DEPLOYMENT OF A THREE-DIMENSIONAL ARRAY OF MICRO-POCKET FISSION DETECTOR TRIADS ($MPFD^3$) FOR REAL-TIME, IN-CORE NEUTRON FLUX MEASUREMENTS IN THE KANSAS STATE UNIVERSITY TRIGA MARK-II NUCLEAR REACTOR

by

MARTIN FRANCIS OHMES

B.S., Kansas State University, 2003
M.S., Kansas State University, 2006

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Mechanical and Nuclear Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2012

Abstract

A Micro-Pocket Fission Detector (MPFD) is a miniaturized type of fission chamber developed for use inside a nuclear reactor. Their unique design allows them to be located between or even inside fuel pins while being built from materials which give them an operational lifetime comparable to or exceeding the life of the fuel. While other types of neutron detectors have been made for use inside a nuclear reactor, the MPFD is the first neutron detector which can survive sustained use inside a nuclear reactor while providing a real-time measurement of the neutron flux.

This dissertation covers the deployment of MPFDs as a large three-dimensional array inside the Kansas State University TRIGA Mark-II Nuclear Reactor for real-time neutron flux measurements. This entails advancements in the design, construction, and packaging of the Micro-Pocket Fission Detector Triads with incorporated Thermocouple, or MPFD^3-T. Specialized electronics and software also had to be designed and built in order to make a functional system capable of collecting real-time data from up to 60 MPFD^3-Ts, or 180 individual MPFDs and 60 thermocouples. Design of the electronics required the development of detailed simulations and analysis for determining the theoretical response of the detectors and determination of their size.

The results of this research shows that MPFDs can operate for extended times inside a nuclear reactor and can be utilized toward the use as distributed neutron detector arrays for advanced reactor control systems and power mapping. These functions are critical for continued gains in efficiency of nuclear power reactors while also improving safety through relatively inexpensive redundancy.

DEPLOYMENT OF A THREE-DIMENSIONAL ARRAY OF MICRO-POCKET FISSION DETECTOR TRIADS ($MPFD^3$) FOR REAL-TIME, IN-CORE NEUTRON FLUX MEASUREMENTS IN THE KANSAS STATE UNIVERSITY TRIGA MARK-II NUCLEAR REACTOR

by

MARTIN FRANCIS OHMES

B.S., Kansas State University, 2003
M.S., Kansas State University, 2006

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Mechanical and Nuclear Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2012

Approved by:

Major Professor
Douglas S. McGregor

Copyright

Martin Francis Ohmes

2012

Abstract

A Micro-Pocket Fission Detector (MPFD) is a miniaturized type of fission chamber developed for use inside a nuclear reactor. Their unique design allows them to be located between or even inside fuel pins while being built from materials which give them an operational lifetime comparable to or exceeding the life of the fuel. While other types of neutron detectors have been made for use inside a nuclear reactor, the MPFD is the first neutron detector which can survive sustained use inside a nuclear reactor while providing a real-time measurement of the neutron flux.

This dissertation covers the deployment of MPFDs as a large three-dimensional array inside the Kansas State University TRIGA Mark-II Nuclear Reactor for real-time neutron flux measurements. This entails advancements in the design, construction, and packaging of the Micro-Pocket Fission Detector Triads with incorporated Thermocouple, or MPFD^3-T. Specialized electronics and software also had to be designed and built in order to make a functional system capable of collecting real-time data from up to 60 MPFD^3-Ts, or 180 individual MPFDs and 60 thermocouples. Design of the electronics required the development of detailed simulations and analysis for determining the theoretical response of the detectors and determination of their size.

The results of this research shows that MPFDs can operate for extended times inside a nuclear reactor and can be utilized toward the use as distributed neutron detector arrays for advanced reactor control systems and power mapping. These functions are critical for continued gains in efficiency of nuclear power reactors while also improving safety through relatively inexpensive redundancy.

Contents

List of Figures **xi**

List of Tables **xvii**

List of Abbreviations **xviii**

Acknowledgements **xxii**

Dedication **xxiv**

A Note to the Reader **xxv**

1 Introduction **1**
1.1 Project Motivation 1
1.2 Other In-Core Detectors 2
1.3 Previous Work 3
1.4 Layout of Dissertation 3

2 Overview of the Project **4**
2.1 The Micro-Pocket Fission Detector Triad with Thermal Monitoring 4
2.2 Deployment in the K-State Nuclear Reactor 8

3 Radiation Signals **12**
3.1 Detector Spectra 12
3.2 Lower Level Discrimination 23
3.3 Response Rate 27
3.4 Detector Efficiency 30
3.5 The Bonus Result 31
3.6 Electron-Ion Induction Timing 33
3.7 Pulse Mode Signals 35
3.8 Current Mode Signals 37

4 Construction and Deployment **39**
4.1 Electrical Contacts and Neutron Reactive Coatings 39
4.1.1 Contact Adhesion and Robustness 39
4.1.2 Characterization of ASCEFI 42
4.2 Detector Assembly 45
4.2.1 Gas Channels 46
4.2.2 Sealing the Detectors 47
4.2.3 Flux-Probe Tube Loading 48
4.3 Detector Array Deployment 52

4.3.1 Flux-Probe Hole Restoration . . . 52
4.3.2 Reactor Upgrade . . . 60
4.3.3 Detector Calibration . . . 64
4.3.4 System Installation . . . 65

5 Electronics 70
5.1 Daughter Boards . . . 70
5.1.1 Pulse Mode Electronics . . . 70
5.1.2 MSV Mode Electronics . . . 78
5.1.3 Current Mode Electronics . . . 80
5.1.4 High Voltage . . . 80
5.1.5 Short Protection . . . 83
5.1.6 Thermocouple Electronics . . . 85
5.1.7 Microcontroller . . . 86
5.2 Mother Board . . . 87
5.2.1 Microcontroller . . . 87
5.2.2 Data Communication . . . 88
5.3 The MPFD^3-T 3D System . . . 88
5.3.1 Front Panel . . . 88
5.3.2 Back Panel . . . 90
5.3.3 Reactor Status . . . 91
5.3.4 Power Distribution . . . 92
5.3.5 Cooling . . . 94

6 Embedded Programming and User Interface 95
6.1 Embedded Programming . . . 95
6.1.1 Daughter Boards . . . 95
6.1.2 Mother Board . . . 96
6.2 The MPFD^3-T 3D System's User Interface Application . . . 106
6.3 Trouble Shooting . . . 125
6.3.1 LCD Screen . . . 125
6.3.2 Stuck on Startup Screen . . . 125
6.3.3 Missing Detectors . . . 126

7 Testing and Results 127
7.1 Detector Dose . . . 127
7.2 Detector Pulses . . . 127
7.3 Detector Spectra . . . 130
7.4 Reactor Tracking . . . 132

8 Future Work 140
8.1 Electronics Calibration . . . 140
8.2 Detector Calibration . . . 142

8.3 Suggestions for Future Systems . . . 142
8.3.1 The MPFD^3-T String Design . . . 142
8.3.2 Daughter Board Preamp . . . 143
8.3.3 Counters . . . 143
8.3.4 Data Transfer Rate . . . 144
8.3.5 Cable Pins . . . 145
8.3.6 Fissionable Coatings . . . 146
8.4 Experiments . . . 147
8.5 Industry . . . 147

9 Conclusions and Outlook **149**

Bibliography **151**

A Additional Analysis **155**
A.1 In-Core Failure Analysis . . . 155
A.2 In-Core By-Product Analysis . . . 176

B Programs **183**
B.1 Programming Languages Utilized . . . 183
B.1.1 MATLAB . . . 183
B.1.2 Visual Basic . . . 183
B.1.3 Microchip C18 . . . 183
B.1.4 SRIM 2003/2006 . . . 184
B.1.5 MCNP . . . 184
B.2 Monte Carlo Pulse Distribution . . . 184
B.2.1 MCPulseDist.m . . . 184
B.2.2 TRIMsim.m . . . 189
B.2.3 IntTRIM.m . . . 197
B.2.4 BatchMCPulse.m . . . 198
B.3 Background Contribution and Lower Level Discriminator Setting . . . 201
B.3.1 EDep.m . . . 201
B.3.2 EDepCompAlpha.m . . . 208
B.4 Pulse Shape . . . 212
B.4.1 PulsePlot.m . . . 212
B.5 ASCEFI . . . 214
B.5.1 user.h . . . 214
B.6 Daughter Board Microcontroller Program . . . 247
B.6.1 main.c . . . 247
B.6.2 main.h . . . 291
B.6.3 config.h . . . 293
B.6.4 config.c . . . 298
B.6.5 globals.h . . . 301

B.6.6 GenericTypeDefs.h . . . 303
B.6.7 Tick.h . . . 309
B.6.8 Tick.c . . . 310
B.6.9 BitBangADC.h . . . 314
B.6.10 BitBangADC.c . . . 314
B.6.11 HVLLD.h . . . 349
B.6.12 HVLLD.c . . . 350
B.6.13 PulseMode.h . . . 356
B.6.14 PulseMode.c . . . 356
B.6.15 convert.h . . . 358
B.6.16 convert.c . . . 359
B.7 Mother Board Microcontroller Program . . . 361
B.7.1 main.c . . . 361
B.7.2 main.h . . . 370
B.7.3 config.h . . . 372
B.7.4 config.c . . . 376
B.7.5 ds2482.c and ds2482.h . . . 379
B.7.6 HardwareProfile.h . . . 379
B.7.7 lcd.h . . . 379
B.7.8 lcd.c . . . 381
B.7.9 mpfd3d.h . . . 398
B.7.10 mpfd3d.c . . . 399
B.7.11 temp.h . . . 437
B.7.12 temp.c . . . 438
B.7.13 FTP.c and FTP.h . . . 439
B.7.14 Helpers.c and Helpers.h . . . 439
B.7.15 HTTP.c and HTTP.h . . . 439
B.7.16 I2CEEPROM.c . . . 440
B.7.17 MPFS.c and MPFS.h . . . 440
B.7.18 TCPIP.h . . . 440
B.7.19 SNTP.c and SNTP.h . . . 440
B.7.20 Tick.c and Tick.h . . . 440
B.8 User Interface Application . . . 441
B.8.1 WriteCalibrationCoefficientsFile.m . . . 441
B.9 Data Analysis . . . 498
B.9.1 LoadMPFDConverted.m . . . 498
B.9.2 ConvertAllFiles.m . . . 503
B.9.3 MPFDData.m . . . 505

C Dimensional Drawings **530**
C.1 MPFD^3-T Cavity Substrate . . . 531
C.2 MPFD^3-T Base Substrate . . . 535
C.3 Contact Evaporation Mask . . . 537

C.4 Ion Mill Substrate Holder 538
C.5 In-Core Housing Tube 541
C.6 Drilling Rig 549
C.7 Reactor Deck Plate 563
C.8 Control Rod Guide Tube Tool 565

D Electronic's Schematics and Layouts **570**
D.1 MPFD^3-T Daughter Board Schematic and Layout 570
D.2 MPFD^3-T Mother Board Schematic and Layout 600

List of Figures

2.1 Basic construction or assembly method for a Micro-Pocket Fission Detector. 5
2.2 Cut-away side view of a Micro-Pocket Fission Detector. 5
2.3 The MPFD^3-T substrates. 6
2.4 Photograph of the MPFD^3-T substrates alongside a penny. 7
2.5 The MPFD^3-T substrates overlaid to show alignment of features. 7
2.6 Cross section of the Kansas State University TRIGA Mark-II Nuclear Reactor pedestal. 8
2.7 MPFD^3-T string deployment inside the reactor. 10

3.1 Geometry used for Monte Carlo simulation of pulse. 13
3.2 Path length distribution using a simplified geometry. 14
3.3 Corrected 3-dimensional geometry used for Monte Carlo simulation of pulse height spectrum. 15
3.4 Path length distribution using the corrected 3-dimensional geometry. 17
3.5 Comparison between the 2D and 3D geometries. 17
3.6 Mass distribution of fission products. 18
3.7 Energy deposition per unit distance for fission fragments in natural uranium. 19
3.8 Energy deposition per unit distance for fission fragments in 1 atm of argon fill gas. 19
3.9 Modified 3D geometry used to account for particle energy loss. 20
3.10 Differential pulse height spectra for various natural uranium neutron reactive coating thicknesses in an MPFD. 22
3.11 Differential pulse height spectra for various natural uranium neutron reactive coating thicknesses in a large detector. 23
3.12 Gamma-ray energy deposition in various chamber widths. 24
3.13 Electron energy deposition in various chamber widths. 24
3.14 Alpha particle energy deposition in various chamber widths. 25
3.15 Maximum energy deposition comparison of background radiation to fission fragments for various chamber widths. 25
3.16 Count rate for natural uranium coated MPFDs and “large” fission chambers with various coating thicknesses. 28
3.17 Longest path in a large detector 31
3.18 Longest path in a large detector is shortened by using an alpha blocking mesh 31
3.19 Intrinsic efficiency comparison between a small chamber and a large chamber. 32
3.20 Histogram of the average charge injection time versus the injection energy. . 34
3.21 Electron and ion drift velocity versus bias voltage. 35
3.22 Estimated current pulse shapes. 36

4.1 Alumina substrate holder for use in the ion mill. 40
4.2 The Millitron ion mill. 41

4.3 SEM image of an uncoated platinum electrode and a natural uranium coated electrode. . . . 42
4.4 SEM images of a ^{235}U unsealed alpha source from Isotope Products. . . . 43
4.5 SEM images of a 93% enriched ^{235}U coated electrode. . . . 44
4.6 AES analysis results of different film thicknesses deposited by ASCEFI. . . . 45
4.7 Drawing of the cavity substrate. . . . 46
4.8 Wiring maps for connecting the MPFD^3-Ts. . . . 49
4.9 Detectors loaded into flux-probe tubes with the connector circuit boards. . . . 50
4.10 Worksheet for how to bend the stainless steel tubing. . . . 51
4.11 Flux-probe design. . . . 52
4.12 Original Flux-probe hole restoration tool. . . . 53
4.13 Ingersoll Rand ARO Par-A-Matic pneumatic self-feed drill used in the restoration of the flux-probe holes. . . . 54
4.14 Upper grid plate layout comparison. . . . 55
4.15 Photo of the reactor with identified fuel and flux-probe hole locations indicated. . . . 56
4.16 Compiled photo analysis results for the upper grid plate. . . . 56
4.17 Flux-probe hole location testing tool. . . . 57
4.18 Drilling rig setup for restoring the flux-probe holes' diameter. . . . 58
4.19 Control valves used to operate the flux-probe hole restoration drilling rig. . . . 59
4.20 Pump sled used with the drilling rig. . . . 60
4.21 Setting up the drilling rig. . . . 61
4.22 Drilling rig with the drill mounted. . . . 61
4.23 Bottom side of the drilling rig. . . . 61
4.24 The drilling rig in action on top of the core. . . . 61
4.25 The control rod guide tube clamp. . . . 63
4.26 MPFD^3-T string installation. . . . 65
4.27 MPFD^3-T 3D System located on 12' level. . . . 66
4.28 The MPFD^3-T array's routing through the reactor tank. . . . 67
4.29 The MPFD^3-T array installed in the reactor core. . . . 68

5.1 Rise time to frequency. . . . 71
5.2 Preamp circuit configuration for AC simulation analysis. . . . 73
5.3 Bode plot of preamp AC simulation analysis. . . . 73
5.4 Preamp circuit configuration for transient simulations. . . . 74
5.5 Settings of the general current source for transient simulation. . . . 75
5.6 Plot of the pulse mode transient simulation results. . . . 76
5.7 Schematic for pulse mode section of the daughter board. . . . 77
5.8 Schematic for MSV mode section of the daughter board. . . . 79
5.9 Schematic for current mode section of the daughter board. . . . 81
5.10 Schematic for high voltage section of the daughter board. . . . 82
5.11 Safety switch to shutdown high voltage if a panel is removed. . . . 83
5.12 Front side of the daughter board. . . . 84

5.13 Back side of the daughter board. 85
5.14 Picture of the front panel of the MPFD^3-T 3D System. 89
5.15 Programming port for daughter boards. 90
5.16 Picture of the back panel of the MPFD^3-T 3D System. 91
5.17 Picture of the probe junction box. 93

6.1 Startup screen of the MPFD^3-T 3D System's LCD. 97
6.2 Enter programming mode screen of the MPFD^3-T 3D System's LCD. . . . 98
6.3 Programming mode screen of the MPFD^3-T 3D System's LCD. 98
6.4 IP address and MAC address screen of the MPFD^3-T 3D System's LCD. . . 99
6.5 Daughter board error screen 1 of the MPFD^3-T 3D System's LCD. 100
6.6 Daughter board error screen 2 of the MPFD^3-T 3D System's LCD. 101
6.7 Current data lines screen of the MPFD^3-T 3D System's LCD. 101
6.8 Connected boards screen of the MPFD^3-T 3D System's LCD. 101
6.9 Sample speed screen of the MPFD^3-T 3D System's LCD. 102
6.10 High voltage screen of the MPFD^3-T 3D System's LCD. 102
6.11 LLD screen of the MPFD^3-T 3D System's LCD. 103
6.12 System temperature screen of the MPFD^3-T 3D System's LCD. 103
6.13 Contrast status screen of the MPFD^3-T 3D System's LCD. 103
6.14 Connected status screen of the MPFD^3-T 3D System's LCD. 104
6.15 Not connected status screen of the MPFD^3-T 3D System's LCD. 104
6.16 Shutdown progress screen of the MPFD^3-T 3D System's LCD. 106
6.17 How to open the User Interface Application via quick links. 107
6.18 How to open the User Interface Application via "All Programs" menu. . . . 107
6.19 The update screen. 108
6.20 Global and local settings missing warning. 108
6.21 Valid path warning. 108
6.22 Storage path screen. 109
6.23 The main screen of the MPFD^3-T 3D System's User Interface Application. . 110
6.24 Settings menu of the User Interface Application. 111
6.25 File menu of the User Interface Application. 111
6.26 Network adapter settings window of the User Interface Application. 111
6.27 Live data window of the User Interface Application. 112
6.28 Live data window showing connection process and settings download. . . . 112
6.29 Board-string connections window of the User Interface Application. 113
6.30 UPS control window. 114
6.31 High voltage settings window. 114
6.32 LLD settings window. 114
6.33 Advanced settings window. 114
6.34 Calibration coefficients window. 115
6.35 Save calibration data stream window. 117
6.36 Plot settings menu and sub menus. 121
6.37 Additional sub menus of the plot settings menu. 121

6.38 Plot settings control window. 123
6.39 File replay control bar. 124

7.1 Oscilloscope screen capture of a pulse signal. 128
7.2 A noisy pulse signal captured on the oscilloscope. 129
7.3 Differential pulse height spectra for various natural uranium neutron reactive coating thicknesses in an MPFD with a 0.5 μm platinum overcoat layer. 130
7.4 Logarithmic plot of spectra collected from a heavy coated and a background MPFD. 131
7.5 Linear plot of spectra collected from a heavy coated and a background MPFD. 132
7.6 Converted log file for the linear channel during a reactor scram. 133
7.7 Select raw data from the MPFD^3-T 3D System replayed in the User Interface Application during a reactor scram. 134
7.8 Select raw data from the MPFD^3-T 3D System replayed in the User Interface Application during a reactor scram. 136
7.9 Pulse mode data for different reactor power levels. 137
7.10 MSV mode data for different reactor power levels. 137
7.11 Current mode data for different reactor power levels. 138
7.12 Current mode data for different reactor power levels. 138

8.1 Cable pin assignment for 37-pin D-SUB connector. 141
8.2 Cable tester for checking conductivity of each wire and connector. 145

A.1 Byproduct log for a MPFD^3-T string with tinned copper wiring. 179
A.2 Byproduct log for a MPFD^3-T string with bare copper wiring. 180
A.3 Byproduct log for a MPFD^3-T string with bare aluminum wiring. 181
A.4 Byproduct log for a MPFD^3-T string with alumel wiring. 182

C.1 MPFD^3-T substrate cavity drawing, page 1. 531
C.2 MPFD^3-T substrate cavity drawing, page 2. 532
C.3 MPFD^3-T substrate cavity drawing, page 3. 533
C.4 MPFD^3-T substrate cavity drawing, page 4. 534
C.5 MPFD^3-T substrate base drawing, page 1. 535
C.6 MPFD^3-T substrate base drawing, page 2. 536
C.7 Contact evaporation mask drawing. 537
C.8 Ion mill substrate holder back view. 538
C.9 Ion mill substrate holder front view. 539
C.10 Ion mill sample rotation drive motor mounting plate. 540
C.11 Flux probe stop weld detail, page 1. 541
C.12 Flux probe tip detail, page 2. 542
C.13 Flux probe stop collar drawing, page 3. 543
C.14 Flux probe bushing weld detail, page 4. 544
C.15 Flux probe bushing drawing, page 5. 545

C.16 Flux probe tip drawing, page 6. 546
C.17 Flux probe coupler bushing weld detail, page 7. 547
C.18 Flux probe coupler bushing detail, page 8. 548
C.19 Drilling rig assembly drawing. 549
C.20 Drilling rig piston clamp assembly drawing. 550
C.21 Drilling rig fuel guide pin drawing. 551
C.22 Drilling rig hold lever drawing. 552
C.23 Drilling rig piston mounting block drawing. 553
C.24 Drilling rig guide plate drawing, page 1. 554
C.25 Drilling rig guide plate dimensions table, page 2. 555
C.26 Drilling rig drill suction assembly drawing. 556
C.27 Drilling rig suction housing drawing. 557
C.28 Drilling rig suction tube short elbow drawing. 558
C.29 Drilling rig drill base drawing. 559
C.30 Drilling rig floating suction tube drawing. 560
C.31 Drilling rig drill exhaust capture block drawing. 561
C.32 Drilling rig test plate drawing. 562
C.33 Reactor deck plate assembly drawing, page 1. 563
C.34 Reactor deck plate assembly drawing, page 2. 564
C.35 Control rod guide tube tool drawing, page 1. 565
C.36 Control rod guide tube tool drawing, page 2. 566
C.37 Control rod guide tube tool drawing, page 3. 567
C.38 Control rod guide tube tool drawing, page 4. 568
C.39 Control rod guide tube tool drawing, page 5. 569

D.1 Schematic for the daughter boards, page 1. 571
D.2 Schematic for the daughter boards, page 2. 572
D.3 Schematic for the daughter boards, page 3. 573
D.4 Schematic for the daughter boards, page 4. 574
D.5 Schematic for the daughter boards, page 5. 575
D.6 Schematic for the daughter boards, page 6. 576
D.7 Schematic for the daughter boards, page 7. 577
D.8 Schematic for the daughter boards, page 8. 578
D.9 Schematic for the daughter boards, page 9. 579
D.10 Schematic for the daughter boards, page 10. 580
D.11 Schematic for the daughter boards, page 11. 581
D.12 Schematic for the daughter boards, page 12. 582
D.13 Schematic for the daughter boards, page 13. 583
D.14 Schematic for the daughter boards, page 14. 584
D.15 Schematic for the daughter boards, page 15. 585
D.16 Schematic for the daughter boards, page 16. 586
D.17 Schematic for the daughter boards, page 17. 587
D.18 Schematic for the daughter boards, page 18. 588

D.19 Top side data and signal plane, layer 1, layout for the daughter boards. . . . 589
D.20 Interior ground plane, layer 2, layout for the daughter boards. 590
D.21 Interior data and signal plane, layer 3, layout for the daughter boards. . . . 591
D.22 Interior data and signal plane, layer 4, layout for the daughter boards. . . . 592
D.23 Interior power and high voltage plane, layer 5, layout for the daughter boards.593
D.24 Bottom side data and signal plane, layer 6, layout for the daughter boards. . 594
D.25 Schematic for the mother board, page 1. 601
D.26 Schematic for the mother board, page 2. 602
D.27 Schematic for the mother board, page 3. 603
D.28 Schematic for the mother board, page 4. 604
D.29 Schematic for the mother board, page 5. 605
D.30 Schematic for the mother board, page 6. 606
D.31 Schematic for the mother board, page 7. 607
D.32 Schematic for the mother board, page 8. 608
D.33 Top side data plane, layer 1, layout for the mother board. 609
D.34 Interior ground plane, layer 2, layout for the mother board. 610
D.35 Interior power plane, layer 3, layout for the mother board. 611
D.36 Bottom side data plane, layer 4, layout for the mother board. 612

List of Tables

3.1 Energy deposition for various chamber widths. 26
3.2 Count rates for central thimble full power operations (250 kW) with different neutron reactive coatings (1 μm platinum overcoat layer). 29
3.3 MPFD^3-T detector efficiency in counts per second per nv. 30
3.4 Charge injection times and energies for various natural uranium neutron reactive coating thicknesses. 34
3.5 Drift times for electrons and ions. 36
3.6 Mean pulse energy with no LLD. 37
3.7 Average output currents. 38

4.1 Comparison of flux probe hole locations based on engineering drawings and the photo analysis results. 57

5.1 Settings and results for preamp transient simulation. 75

6.1 Data structure of MATLAB converted ".cal" file. 118

A.1 ALI and CEDE fission product analysis for gases listed in 10CFR20 Appendix B. 157
A.2 ALI and CEDE fission product analysis for gases not listed in 10CFR20 Appendix B. 158
A.3 ALI and CEDE fission product analysis for particulates listed in 10CFR20 Appendix B. 160
A.4 ALI and CEDE fission product analysis for particulates not listed in 10CFR20 Appendix B. 165

D.1 Bill of materials for the daughter boards, page 1. 595
D.2 Bill of materials for the daughter boards, page 2. 596
D.3 Bill of materials for the daughter boards, page 3. 597
D.4 Bill of materials for the daughter boards, page 4. 598
D.5 Bill of materials for the daughter boards, page 5. 599
D.6 Bill of materials for the mother board, page 1. 613
D.7 Bill of materials for the mother board, page 2. 614

List of Abbreviations

3D 3-Dimensional

AC . . . . . Alternating Current

ADC . . . . . Analog-to-Digital Converter

AES . . . . . Auger Electron Spectroscopy

ALI . . . . . Annual Limit of Intake

Alumina . . . . . Aluminum Oxide (Al_2O_3) ceramic

Å . . . . . Angstrom (10^{-10} meters)

Aqua Regia . . . . . 3:1 $HCl:HNO_3$

ARP . . . . . Address Resolution Protocol

ASCEFI . . . . . Automated System for the Consistent Electroplating of Fissionable Isotopes

AWG . . . . . American Wire Gauge

BOE . . . . . Buffered Oxide Etech or Diluted HF

CEDE . . . . . Committed Effective Dose Equivalent

CMRR . . . . . Common Mode Rejection Ratio

D-Sub . . . . . D-Subminiature Connector

DAC . . . . . Derived Air Concentration

DAC . . . . . Digital-to-Analog Converter

DC . . . . . Direct Current

DHCP . . . . . Dynamic Host Configuration Protocol

DNS . . . . . Domain Name System

DynDNS . . . . . Dynamic Domain Name System

EEPROM . . . . . Electrically Erasable Programmable Read-Only Memory

EMI . . . . . Elecro-Magnetic Interference

FIFO . . . . . First-In First-Out

FPGA . . . . . Field-Programmable Gate Array

FTP . . . . . File Transfer Protocol

GMT . . . . . Greenwich Mean Time

HCl . . . . . Hydrochloric Acid

HF . . . . . Hydrofluoric Acid

HNO_3 . . . . . Nitric Acid

HTTP . . . . . HyperText Transfer Protocol

HV . . . . . High Voltage

ICD . . . . . In-Circuit Debugger

ICMP . . . . . Internet Control Message Protocol

ID . . . . . Inside Diameter

IDE . . . . . Integrated Development Environment

I^2C . . . . . Inter-Integrated Circuit

IP . . . . . Internet Protocol

K-State . . . . . Kansas State University

KSU . . . . . Kansas State University

LCD . . . . . Liquid Crystal Display

LED . . . . . Light Emitting Diode

LLD . . . . . Lower Level Discriminator

MAC . . . . . Media Access Control

MCA . . . . . Multi-Channel Analyzer

MCNP . . . . . Monte Carlo N-Particle

MNE . . . . . Mechanical and Nuclear Engineering Department

MPFD . . . . . Micro-Pocket Fission Detector

$MPFD^3$-T . . . . . MPFD Triad with Thermocouple

- B . . . . . Background - No Coating
- H . . . . . Heavy Natural Uranium Coating
- L . . . . . Light Natural Uranium Coating

T . . . . . Thorium Coating (replaced by L)

U . . . . . 93% Enriched Uranium Coating (replaced by H)

MPFS . . . . . Microchip File System

MSV . . . . . Mean Square Voltage

NIM . . . . . Nuclear Instrumentation Module

OD . . . . . Outside Diameter

PGA . . . . . Programmable Gain Amplifier

Preamp . . . . . Preamplifier

Pro/E . . . . . Pro Engineer

RAM . . . . . Random Access Memory

RMS . . . . . Root-Mean-Square

RSR . . . . . Rotary Specimen Rack

RTV . . . . . Room-Temperature Vulcanization

SAR . . . . . Safety Analysis Report

SEM . . . . . Scanning Electron Microscope

S.M.A.R.T. Lab . . . . . Semiconductor Materials and Radiological Technologies Laboratory

SNTP . . . . . Simple Network Time Protocol

SPI . . . . . Serial Peripheral Interface

SRIM . . . . . Stopping and Range of Ions in Matter

TCP . . . . . Transmission Control Protocol

TRIGA . . . . . Training, Research, and Isotopes - General Atomics

TRIM . . . . . Transport of Ions in Matter

TVS . . . . . Transient Voltage Suppressor

UART . . . . . Universal Asynchronous Receiver/Transmitter

UDP . . . . . User Datagram Protocol

UHP . . . . . Ultra-High Purity

UIA . . . . . User Interface Application

USB . . . . . Universal Serial Bus

UTC . . . . . Coordinated Universal Time

XPS . . . . . X-ray Photoelectron Spectroscopy

Acknowledgments

This project was made possible through the support of the Department of Energy (DOE) Nuclear Engineering Research Initiative (NERI) Grant # DE-FG07-02SF22611 and the Department of Energy (DOE) Innovations in Nuclear Infrastructure and Education (INIE) along with the vision of Prof. Douglas McGregor. The development and fabrication of the Micro-Pocket Fission Detectors (MPFDs) has been accomplished through the Mechanical and Nuclear Engineering (MNE) Department's Semiconductor Materials and Radiological Technologies (S.M.A.R.T.) Laboratory at Kansas State University. Testing of the MPFDs was completed at the Kansas State University TRIGA Mark-II Nuclear Reactor Facility, which is also located in the MNE Department.

Prof. J. Kenneth Shultis and Dr. A.S.M. Sabbir Ahmed provided insight into the design constraints through their theoretical modeling and power density back calculation models (not presented here). Undergraduate assistance in the construction and testing of the MPFDs was provided by David Bruno, Brian Cooper, Rylan Ortiz, Greg Hilgenkamp, Harry McDonald, Ryan Rochel, Clay Bolinger, Jarrett Leeds, Tracy Pinsent, Alan Cebula and Ian Shannon. Significant support was provided by the skilled machinists Jason Selland and Gary Thorton along with administrative support provided by the former department head, Prof. Mohammad Hosni, and the MNE staff. Dr. Phil Ugorowski and the S.M.A.R.T. Lab staff also provided critical assistance throughout this project and their insight over the past ten years has greatly enhanced the learning experience.

The reactor manager, Dr. Jeffrey Guether, and the reactor staff went out of their way to extend the railing around the 12' level where the MPFD^3-T 3D System's electronics are located, which is greatly appreciated. Without the extensive support of P. Michael Whaley (former reactor manager), Dr. Guether and all of the reactor staff, this project never would have been completed.

Several individuals outside of Kansas State University have provided critical assistance

towards the success of the MPFD project. The uranium deposition techniques were developed through the assistance of Kimberlee Hoffert from Reuter Stokes. Dan Kruger and Peter Scheuer of Honeywell, Kansas City Plant provided specialty services and assistance with the firing of Alumina Tape.

The understanding and support for completing this project by FLIR Radiation, Inc., formerly ICx Radiation, Inc., through the management of Andrey Gueorguiev and Paul Bowlus, has been exceptional. Dr. Hartmut Brands has provided extensive assistance with real world electronics design for radiation detection systems.

The assistance of all these individuals and companies is greatly appreciated.

The efforts put forth by Prof. Douglas McGregor, major professor, and the graduate committee, Profs. J. Kenneth Shultis, Andrew Rys, James DeVault and Timothy Bolton, is also greatly appreciated.

Dedication

To all of those who have had an impact on my life,

especially my wife Emily,

and my children Beth and Seth

A Note to the Reader

The design, methods of construction, and basic theoretical development of the Micro-Pocket Fission Detectors may be found in the thesis titled "Micro-Pocket Fission Detector (MPFD) Development for the Kansas State University TRIGA Mark-II Nuclear Reactor" [1]. The detectors had been developed as a sub-part of the complete work described in this dissertation. Only the portions of the prior research which were found to need further development or the methods of construction which needed refinement will be discussed in this work. Otherwise, it is recommended to review the thesis in order to be familiar with the MPFD design.

For the reader's convenience, a copy of the thesis has been included as a supplemental file of the digital version of this dissertation available through Kansas State University's Research Exchange (http://krex.k-state.edu).

Chapter 1

Introduction

Provided in this chapter is a review of the motivation behind the project, along with a brief description of other existing in-core detectors. An examination of the previous work completed for the realization of this project will also be provided. The conclusion of this chapter outlines the remainder of this dissertation.

1.1 Project Motivation

Most reactor operations rely on external detectors to provide a real-time analysis of the current reactor's operation while select in-core detectors are used to determine localized reactor power trends. This configuration of detectors is necessary due to the availability of neutron detectors and their subsequent limitations that will be described in Section §1.2. Because of the limitations of existing neutron detectors, a need has been identified for small in-core neutron detectors that have little to no impact on neutron flux or core reactivity while providing a real-time response. Specifically, the detector needs to respond to reactor transients both at high and low flux levels in real-time. The detectors also need to be insensitive to background radiation so that no induced offsets or increased count rates are generated. Most importantly, the detector must be able to survive the harsh radiation and temperature environment of a nuclear power reactor. For research purposes specifically, the detector and associated assembly must be safe to handle both pre- and post-irradiation. Being inexpensive to manufacture would be an added benefit [1, 2].

From this identified need, the Micro-Pocket Fission Detector was created, and with its capabilities, the realization of a large 3D array of in-core, real-time neutron detectors was possible. Such an array would make it possible to examine and track the localized flux

throughout the reactor not only during steady-state operations but also during fast transient operations. Additionally, the data from these detectors could be utilized to back project the power density of the fuel [2–4]. In the end, the implementation of such a system in the Kansas State University TRIGA Mark-II Nuclear Reactor could show the potential of such a system for use inside a power nuclear reactor, which could not only improve the efficiency of the reactor's operation and lead to increased revenue, but also improve the safety and accuracy through immediate knowledge of localized core neutron flux.

1.2 Other In-Core Detectors

There are numerous types of neutron detectors available on the open market, but only a select few are capable of in-core or even near-core operation. Gas filled neutron detectors, of which fission chambers are a specialized subset of these, are one of the most widely used in-core and near-core neutron detectors. These detectors use either a neutron reactive fill gas or a neutron reactive coating to cause ionization of the fill gas through secondary reaction particles. The electrons and ions are drifted through the chamber by an applied electric field that induces a current pulse. These detectors are normally utilized as near-core detectors due to rapid burn-up of their neutron reactive material in very high neutron flux environments and also due to their sensitivity to high temperatures. Some designs have been made that can be utilized for in-core operation but are typically used only temporarily due to their inherent burn-up and their high cost [5–7].

For in-core neutron monitors, one of the most commonly used is the Self-Powered Neutron Detector, or SPND. The detectors are typically made with either ^{59}Co, ^{195}Pt, ^{51}V, HfO_2, Ag, and ^{103}Rh, which have a subsequent gamma or beta decay after being activated by neutrons. The decay rate is proportional to the neutron flux and is sufficient enough to produce a measurable current [8]. Because the detector's response is dependent on the natural decay rate of its activated emitter material, a detector such as the ^{103}Rh SPND requires up to five minutes to reach equilibrium after reactor transients, therefore limiting

its use for real-time analysis.

Other types of neutron detectors are available. Few, if any, of these detectors, outside of those already discussed, can operate in the high neutron flux environment of a nuclear reactor. A more thorough review of these detectors can be found in the previous work [1].

1.3 Previous Work

The prior work [1] involved the development of the Micro-Pocket Fission Detector. In that research, various forms of substrates and electrode configurations were tested in order to determine the requirements for the MPFD^3-T design discussed in Chapter 2. The electroplating method of depositing fissionable coatings was also developed along with the construction processes anticipated for the deployment of a large 3D array inside the Kansas State University TRIGA Mark-II Nuclear Reactor. Theoretical analysis through simulation studies were performed in order to examine the background radiation immunity of the MPFD design along with investigation of the expected lifetime of the neutron reactive coatings.

1.4 Layout of Dissertation

The following chapters provide an overview of the project with an introduction to the MPFD^3-T design and the proposed array deployment. A series of simulation studies are then utilized to find the expected signal shape of the detectors which is used later for the development of the readout electronics. Construction of the MPFD^3-Ts and deployment of the array then leads into the design of the electronics and software. Data collected from the completed system is then presented along with a review of future work and suggested design changes for future implementations of the system. After the conclusions, several sizable sections are provided in the appendix for analysis required by the Kansas State University TRIGA Mark-II Nuclear Reactor, code utilized in the simulations and for operating the system, drawings of the components, and schematics and layouts of the electronics.

Chapter 2

Overview of the Project

Provided in this chapter is an introduction to the Micro-Pocket Fission Detector (MPFD) technology and also the design of the MPFD^3-T detector. The goal of this project is to use a 3-dimensional array of these detectors inside the Kansas State University TRIGA Mark-II Nuclear Reactor and measure the reactor power in real-time through neutron-flux monitoring. Information concerning the proposed deployment is described in this chapter while provided in the remaining chapters is a description of its execution and results.

2.1 The Micro-Pocket Fission Detector Triad with Thermal Monitoring (MPFD^3-T)

The Micro-Pocket Fission Detector, or MPFD, is a new development in the class of gas detectors known as fission chambers [1–3, 9–13]. Gas detectors operate on the premise that radiation interacts with the fill gas and causes the gas to ionize. The voltage difference applied on the electrodes located on either side of the gas volume causes the gas ions and electrons to drift to the electrodes. The motion of the ions and electrons through the gas volume induces a current, which can then be amplified and measured [5, 14, 15]. Measuring neutrons is difficult because they have no charge, therefore, an intermediate neutron reactive material is commonly utilized to provide a signal or indication of a neutron event. In the case of fission chambers, this reactive material is often applied as a coating on one or both electrodes. Other configurations are possible, but they will not be discussed in the present work [5, 14, 15].

For the MPFD design, a parallel plate electrode structure was selected, which has two equal sized electrodes parallel to each other on either side of the gas volume. The basic con-

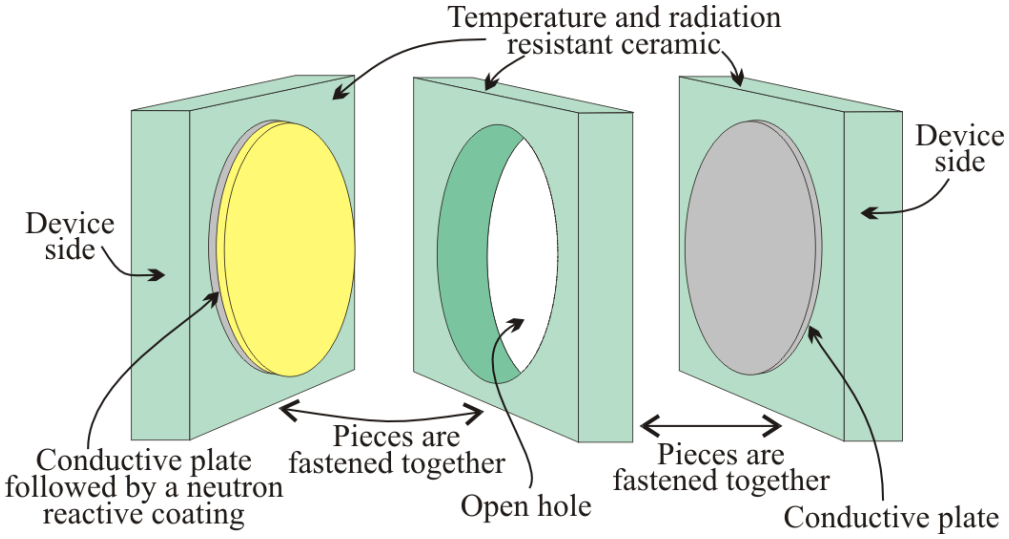


Figure 2.1: Basic construction or assembly method a Micro-Pocket Fission Detector (adapted from [2]).

struction method involves three alumina (Al_2O_3) ceramic substrates. One substrate, herein referred to as the cavity substrate, has a hole in it to form the gas pocket or gas-filled void. This cavity substrate is sandwiched between two base substrates, each of which has an electrode as in Figures 2.1 and 2.2. In this design, one or both electrodes are coated with a neutron reactive material.

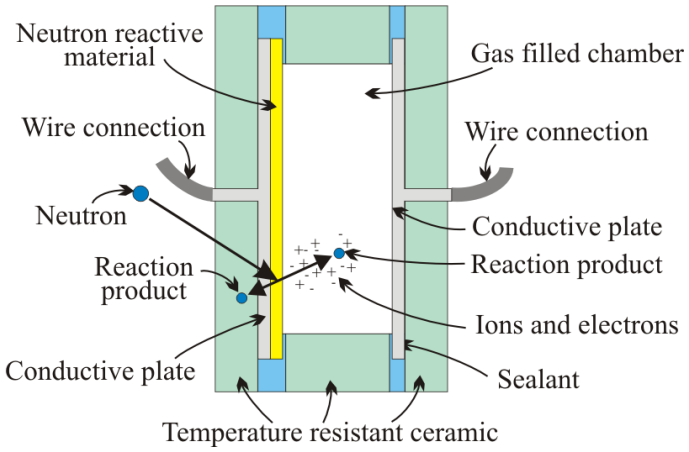


Figure 2.2: Cut-away side view of a Micro-Pocket Fission Detector showing ionization of the fill gas from a neutron induced reaction with the neutron reactive material(adapted from [2]).

Fission chambers usually utilize fissionable or fissile materials such as ^{235}U, ^{238}U, or ^{232}Th because these materials will fission with significantly more energy than other neutron reactive materials such as ^{6}Li or ^{10}B. Argon is the typical fill gas of choice for use inside fission chambers [14]. For MPFDs, argon is trapped inside the gas-filled void created by

the cavity substrate when the three substrates are sealed together.

What sets MPFDs apart from other fission chambers is the size of the MPFD. Historically, fission chambers utilize either high pressure fill gas or large chamber volumes in order to ionize as much gas as possible through the full path length of the fission fragments, between 20 to 30 mm in one atmosphere of argon gas [1, 5, 14, 15]. Analysis has revealed that this is not necessary and that only a partial path length is necessary to deposit enough energy to discriminate against other types of radiation, such as gamma-rays, x-rays, beta particles, and alpha particles. Because of this, the dimensions utilized in the MPFD^3-T design provide a chamber only 1 mm in diameter and 0.5 mm thick [1].

Substrates developed for the MPFD^3-T design are shown in Figure 2.3. The MPFD^3-T designation stands for a triad, or three, Micro-Pocket Fission Detectors with an embedded thermocouple. The red dots in the cavity substrate (top) of Figure 2.3 are the 1 mm diameter holes that form the gas pockets of the three MPFDs. Also in the cavity sub-

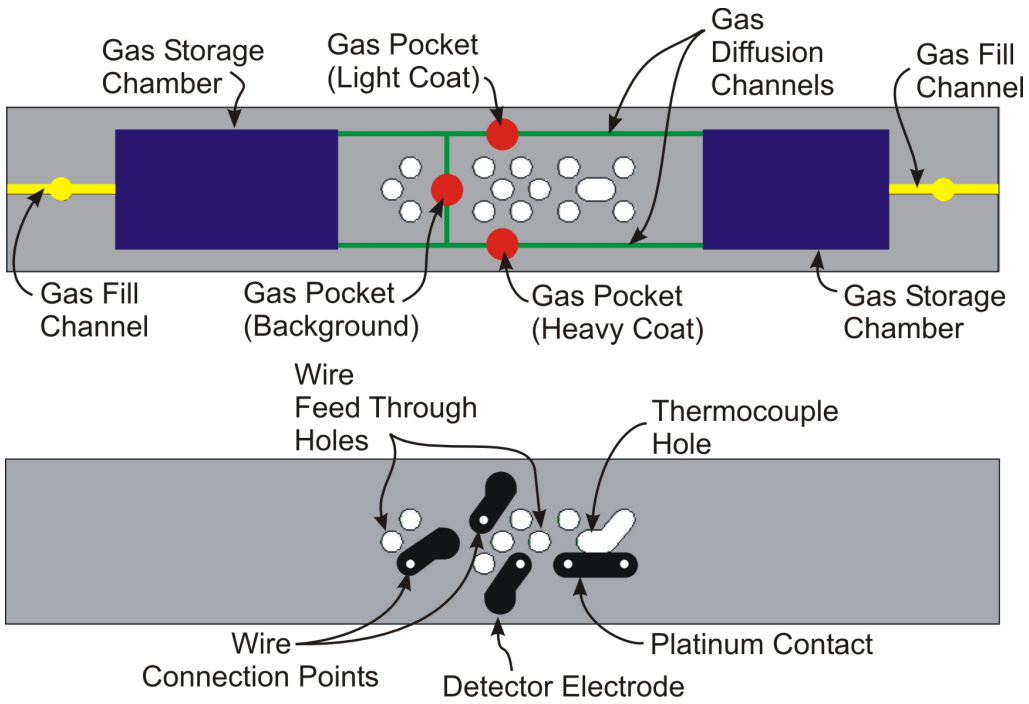


Figure 2.3: The cavity substrate (top) and base substrate (bottom) used for the fabrication of the MPFD^3-T (adapted from [1]).

strate are two large rectangular gas storage chambers (blue) that are connected to the gas pockets via gas diffusion channels (green). Likewise, there are gas fill channels (yellow) that connect the gas storage chambers to the edge of the cavity substrates. While the gas pockets and gas storage chambers are through holes, the gas diffusion channels and gas fill channels are only grooves that have been milled into one side of the cavity substrate's surface. After the cavity and base substrates are assembled, the gas fill channels are intended to be used for evacuating and backfilling the gas pockets with ultra high purity argon fill gas. Once filled, the gas fill channels are plugged in order to trap the gas inside the detectors. The diffusion channels allow fission fragments to diffuse out of the gas pockets and into the gas storage chambers in order to maintain as high a purity of argon as possible inside the detectors.

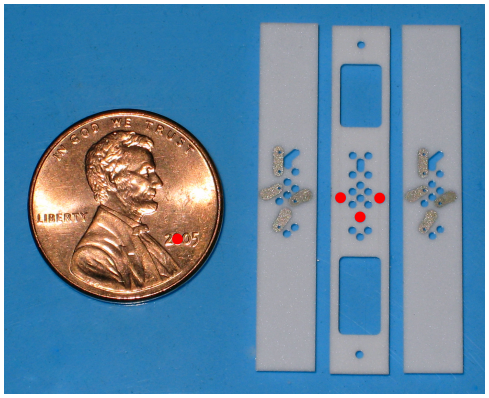


Figure 2.4: Photograph of the MPFD^3-T substrates alongside a penny. The gas pockets and a zero in the date stamp of a U.S. penny are highlighted in red to show their simularity in size (from [13]).

The base substrate (bottom of Figure 2.3) has four platinum contacts that are deposited by platinum evaporation through a shadow mask. Figure 2.4 is a picture of the actual substrates with the evaporated contacts. Three of the contacts form the detector electrodes and provide a point of contact for connection to 0.010" diameter wires. The fourth contact has two holes, one for connection to one leg of a thermocouple and the other for connection to a read-out wire. Both substrates have a series of larger 0.030" diameter holes, which line

Figure 2.5: The MPFD^3-T cavity substrate and base substrates overlaid to show the alignment of their features (from [1]).

up when assembled to form pockets to help secure the wires to the contacts or to provide feedthroughs for routing of the wires. Figure 2.5 illustrates how the holes line up when the substrates are assembled to form the wire connection pockets and feedthroughs.

2.2 Deployment in the K-State Nuclear Reactor

Figure 2.6 is a cross-section view of the pedestal of the TRIGA Mark-II Nuclear Reactor located at Kansas State University. The reactor core sits inside a 22' deep tank of water, with the top of the core residing 16.5' below the top of the tank. The gamma-ray radiation levels in the reactor at full power (1250 kW) can reach approximately 1.2×10^{5} rad s^{-1} with fast (>10 keV) and slow (<0.21 eV) neutron fluxes of 6.0×10^{13} n cm^{-2} s^{-1} and

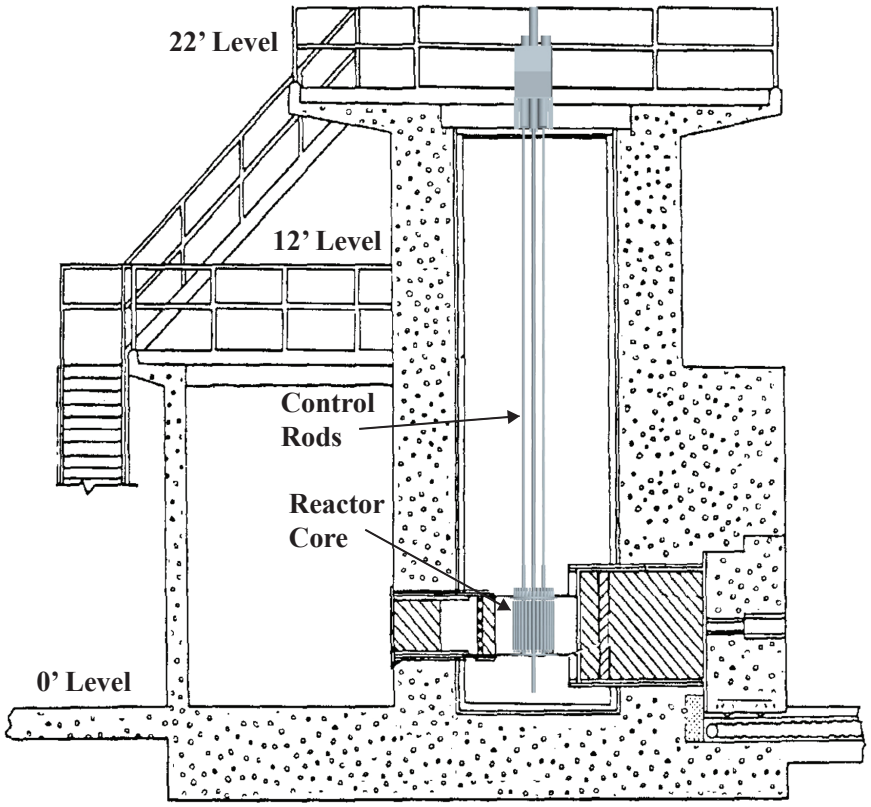
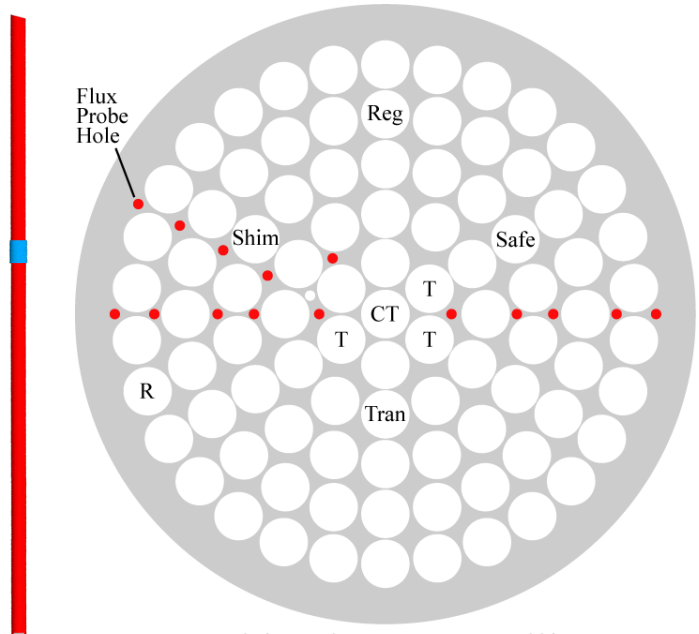


Figure 2.6: Cross section of the Kansas State University TRIGA Mark-II Nuclear Reactor pedestal with an overlaid model of the reactor and control rod assembly (adapted from [16]).

5.0×10^{13} n cm^{-2} s^{-1}, respectively. In order to install the detectors inside the reactor core, all of the materials must be radiation hard. Unfortunately, the preamplifier electronics that are needed to condition the detectors' signals cannot survive the high level of radiation in the reactor core, so the electronics must be located outside the high radiation area near the reactor. The water that surrounds the reactor for cooling purposes also acts as a radiation shield. Because of this shielding, it would have been possible to locate the electronics as close as 6' from the reactor. However, with minimal space inside the reactor tank, the deployment of all the electronics inside the tank would be impractical and would be a hindrance to required reactor maintenance and operations. Because the 22' level of the reactor is utilized extensively for public tours and in-core experiments, the only practical location left for the electronics was on the 12' level of the reactor pedestal (see Figure 2.6). To reach the 12' level location, the wiring from the detectors had to be routed from the reactor core, up the side of the reactor tank wall, across the 22' level, and back down to the electronics on the 12' level, a total distance of approximately 40'.

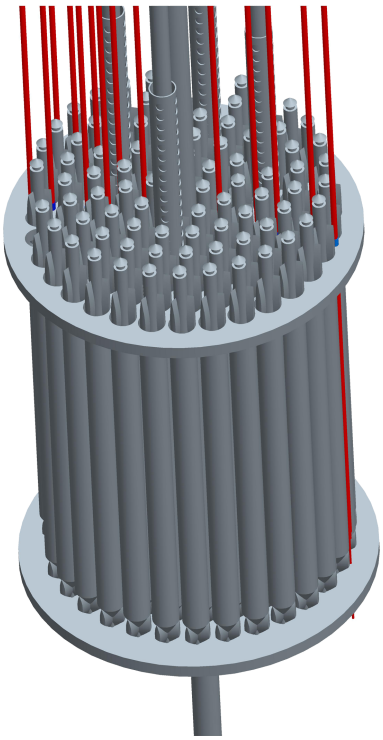
To deploy the detectors inside the reactor, a string of four MPFD^3-Ts were installed inside a flux-probe tube (Figure 2.7a). The reactor is configured through the use of a lower and an upper grid plate, which holds the fuel, control rods, and experimental facilities. Figure 2.7b is a drawing of the upper grid plate, which also shows the flux-probe holes. While the grid plates can support up to 16 flux probes, only 15 of the positions (indicated by red dots) are considered for use in the deployment of the MPFD^3-T array. The upper and lower grid plates have holes located in identical positions, but the holes in the lower grid plate are a smaller diameter than those in the upper grid plate in order to support the fuel. This is also the case for the flux-probe holes. The upper grid plate has 8 mm (0.315") diameter flux-probe holes, while the lower grid plate has 4 mm (0.157") diameter holes [1]. Whereas the fuel rods use the lower grid plate to support their weight and to set their depth inside the reactor, the 1/8" pin at the bottom of the MPFD^3-T string (Figure 2.7a) is only used to align the string. A collar, shown in blue, is welded onto the flux-probe tubes and



Reg	Regulating Rod	R	Rabbit
Safe	Safety Rod	CT	Central Thimble
Shim	Shim Rod	T	Instrumented Fuel
Tran	Transient Rod	●	MPFD^3-T String

(b)

(a)



(c)

Figure 2.7: MPFD^3-T strings (a) are built from an aluminum tube and hold four MPFD^3-Ts inside. These strings are then placed in select flux probe holes located in the upper grid plate (b) of the Kansas State University TRIGA Mark-II Nuclear Reactor as shown in (c) (adapted from [13]).

is designed to rest on the upper grid plate, which aligns the detectors to the fuel. Shown in Figure 2.7c is a model of the reactor with the upper and lower grid plates, fuel, control rods, and the proposed array of MPFD^3-T strings.

Chapter 3

Radiation Signals

While one could simply assemble a variety of radiation detectors and examine their response, there is an advantage of performing theoretical experiments to uncover specific secrets of design changes which could be missed in physical experiments. Some of the effects of design changes that need to be explored deal with the signals produced by the detectors. Once the radiation induced spectrum has been found, it can be used to help determine the type and thickness of the neutron reactive coatings that should be used. With this information and some additional analysis, the basic pulse shape and current output can be identified in order to assist with later development of associated electronics.

3.1 Detector Spectra

Prior analysis has shown that the energy deposition of fission fragments per distance of travel inside the fission chamber's fill gas is approximately uniform due to the chamber's small size [1–3, 9, 10, 12]. Therefore, the differential pulse height spectrum produced by the MPFD can be approximated by finding the distribution of path lengths inside the chamber. Original analysis of this problem used the geometry as shown in Figure 3.1 [1].

The analysis is done using a Monte Carlo simulation where three random numbers are generated for the radial position R, the azimuthal angle Θ and the polar angle ϕ. These values are generated through the equations

$$R = \frac{d}{2}\sqrt{r_1} \tag{3.1}$$

$$\theta = 2\pi r_2 \tag{3.2}$$

$$\phi = \cos^{-1}(2r_3 - 1) \tag{3.3}$$

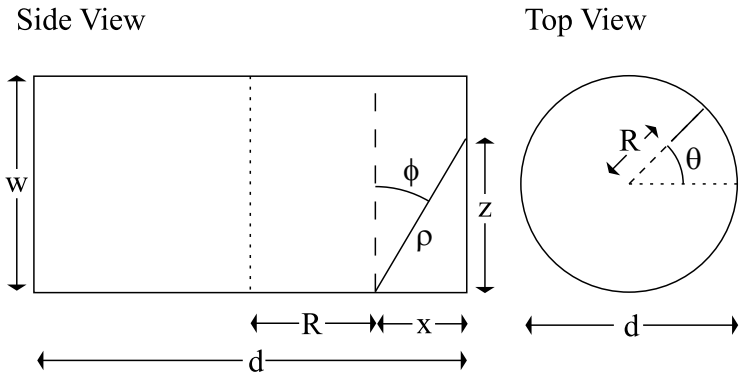


Figure 3.1: Geometry used for Monte Carlo simulation of pulse.

where r_x is a uniform random number from 0 to 1 and d is the chamber diameter. The radial position is found using the square root of the random number in order to maintain a uniform distribution over the circular surface area of the neutron reactive coating. Due to symmetry, the radial path length the particle travels is

$$x = \begin{cases} d/2 - R, & \cos\theta \geq 0 \\ d/2 + R, & \cos\theta < 0 \end{cases} . \tag{3.4}$$

The vertical path length z traveled by the particle is calculated as

$$z = \frac{x}{\tan\phi}. \tag{3.5}$$

However, the vertical path length must be less than the chamber width w, and thus,

$$z_{\max} = \begin{cases} z, & z \leq w \\ w, & z > w \end{cases} . \tag{3.6}$$

The particle path length is the hypotenuse of the triangle formed by the base length x and the height z. Using the rule of similar triangles, the true particle path length is found as

$$\rho = \frac{x}{\sin\phi}\frac{z_{\max}}{z}. \tag{3.7}$$

The resulting path length distribution is shown in Figure 3.2. Several distinctive features can be observed in the plot. From left to right these features include a sharp rise in the

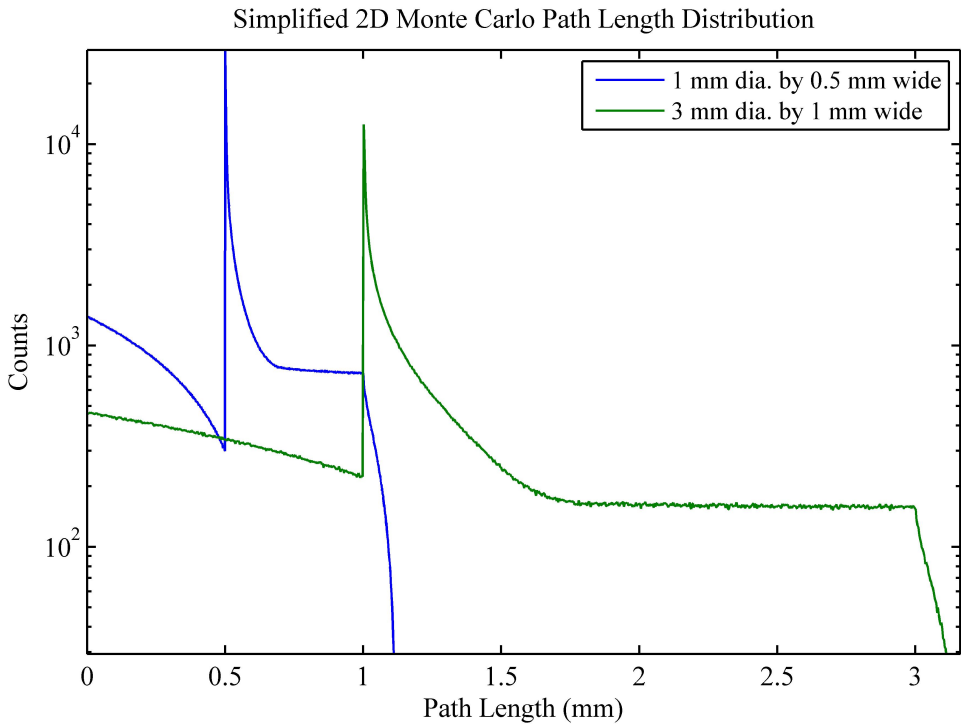


Figure 3.2: Path length distribution using the simplified geometry for a 10^6 particle Monte Carlo simulation.

number of counts at a distance equivalent to the width of the chamber followed by an exponential decay and ending in a plateau. While the primary features of the sharp rise and exponential decay were observed in experimental data, the plateau was not observed, at least not to the sharp detail shown by these simulations. This discontinuity indicates that a more precise model is needed.

The original simplified model does not take into account the full 3-dimensional nature of the particle path. Specifically, the model does not properly include the azimuthal angle in which the particle was emitted. In Figure 3.3, this angle is defined as Θ_2.

For an analysis of this corrected 3-dimensional geometry, four random numbers are generated. These include the radial position R and the azimuthal angle Θ_1 of the fission reaction along with the emission polar angle ϕ and the azimuthal angle Θ_2 of the fission products. The first three are the same as those used in the original simplified analysis.

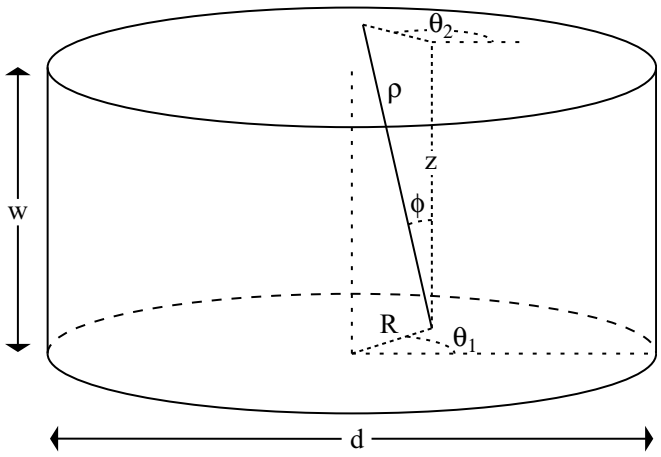


Figure 3.3: Corrected 3-dimensional geometry used for Monte Carlo simulation of pulse height spectrum.

These values are generated through the equations

$$R = \frac{d}{2}\sqrt{r_1} \tag{3.8}$$

$$\theta_1 = 2\pi r_2 \tag{3.9}$$

$$\phi = \cos^{-1}(2r_3 - 1) \tag{3.10}$$

$$\theta_2 = 2\pi r_4 \tag{3.11}$$

again, where r_x is a uniform random number from 0 to 1 and d is the chamber diameter. First, the rectangular coordinates of the fission reaction's location is found through the cylindrical coordinate transformation

$$h = R\cos\theta_1 \tag{3.12}$$

$$k = R\sin\theta_1 \tag{3.13}$$

where h and k are used as the x and y positions since they are an offset from the center axis of the detector chamber. The point at which the emitted fission product will hit the circular wall of the fission chamber may then be found by using the equation of a circle

$$(x_2 - h)^2 + (y_2 - k)^2 = \left(\frac{d}{2}\right)^2 \tag{3.14}$$

and the relation

$$\frac{y_2}{x_2} = \tan\theta_2. \tag{3.15}$$

Using these equations and solving for the unknown x_2, we get

$$x_2 = \cos\theta_2 \left[\sqrt{\frac{d^2}{4} - (h\sin\theta_2 - k\cos\theta_2)^2} + h\cos\theta_2 + k\sin\theta_2 \right]. \tag{3.16}$$

The value for y_2 may now be found by back-substituting x_2 into Equation (3.15). The vertical, or z component, is calculated from spherical coordinate transforms where

$$x_2 = \rho\sin\phi\cos\theta_2 \tag{3.17}$$

and

$$z_2 = \rho\cos\phi = \frac{x_2}{\sin\phi\cos\theta_2}\cos\phi = \frac{x_2}{\tan\phi\cos\theta_2}. \tag{3.18}$$

As in the prior analysis, the vertical path length must be less than the chamber width w, and thus,

$$z_{\max} = \begin{cases} z_2, & z_2 \le w \\ w, & z_2 > w \end{cases}. \tag{3.19}$$

The particle path length is found by combining the Pythagorean theorem and the rule of similar triangles as in

$$\rho = \sqrt{x_2^2 + y_2^2 + z_2^2}\frac{z_{\max}}{z_2}. \tag{3.20}$$

From this version of the analysis the path length distribution shown in Figure 3.4 is found. Shown in Figure 3.5 is an overlay of the plots for the original simplified 2-dimensional and corrected 3-dimensional analysis results of the 1 mm diameter by 0.5 mm wide chamber. From this figure it can be seen that accounting for the emission azimuthal angle in the 3-dimensional analysis removes the plateau feature found in the simplified analysis, which more closely agrees with experimental results. It should be noted that the exact same random numbers were used to produce the plots for both models.

While prior analysis showed that the energy deposition of the fission fragments in the fill gas is nearly constant for the short path lengths, the same analysis also showed that the

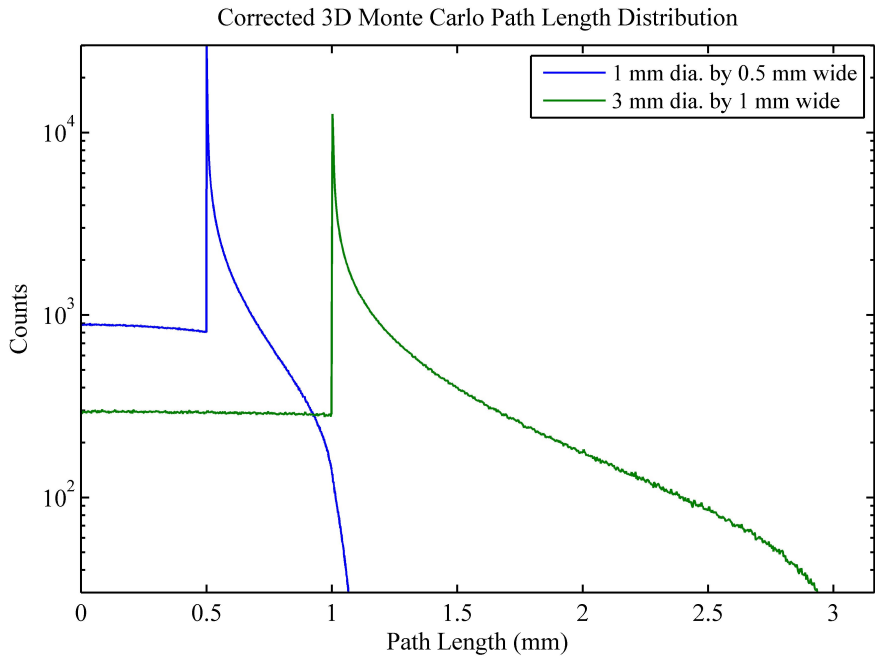


Figure 3.4: Path length distribution using the corrected 3-dimensional geometry for a 10^6 particle Monte Carlo simulation.

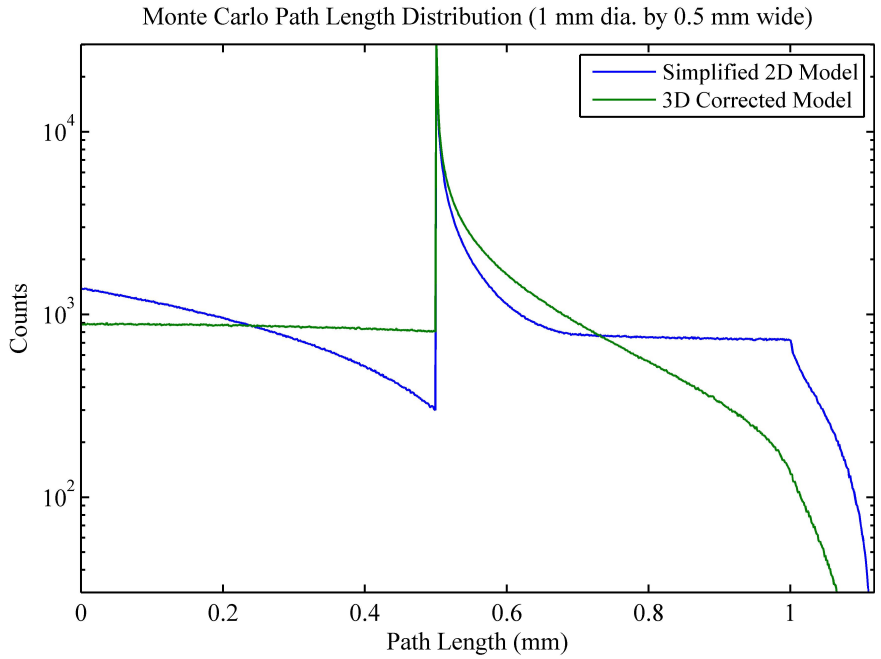


Figure 3.5: Comparison between the path length distributions using the original simplified 2-dimensional geometry and the corrected 3-dimensional geometry for a 10^6 particle Monte Carlo simulation.

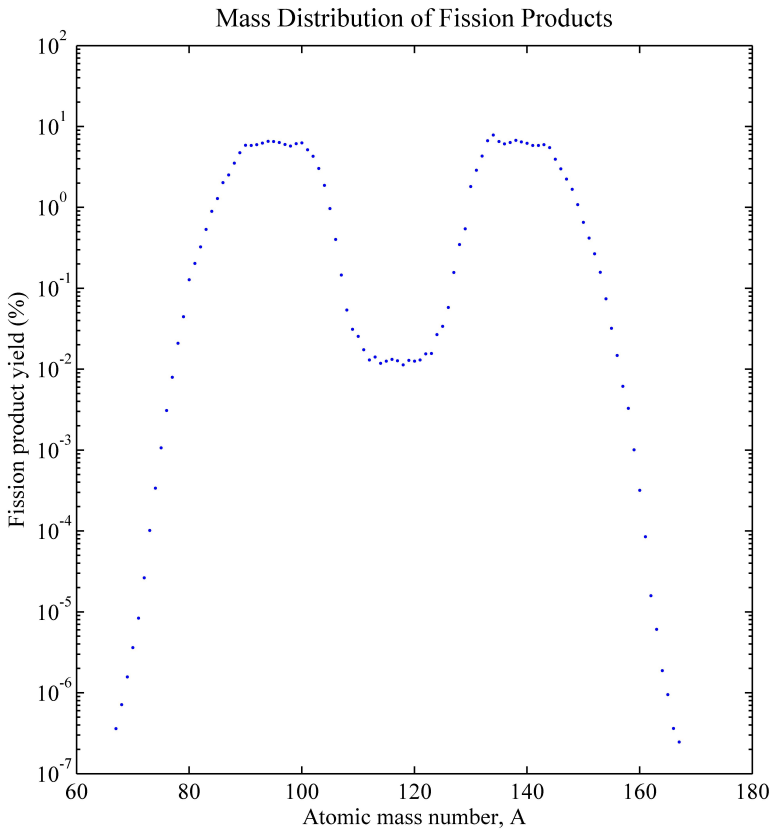


Figure 3.6: Mass distribution of fission products for thermal neutron induced fission of ^{235}U (Adapted from [5, 7].

integrated energy deposition does differ for different fission fragments. This difference is due to the differing fission fragment masses (Figure 3.6) and their corresponding emission energies. The emission energies can be approximated by knowing the masses of both fragments (M_1 and M_2) along with the kinetic energy of the fission fragments (168 MeV) and using the equation

$$E_1 = 168\frac{M_2}{M_1 + M_2}MeV \tag{3.21}$$

where E_1 is the energy of fission fragment 1 [7].

With information from Equation (3.21), simulations to find the energy deposition per unit distance in both the argon fill gas and the uranium neutron reactive coating are possible. The fission fragment will lose a significant amount of energy when traveling through the reactive coating prior to reaching the fill gas (Figure 3.7). This loss causes the energy

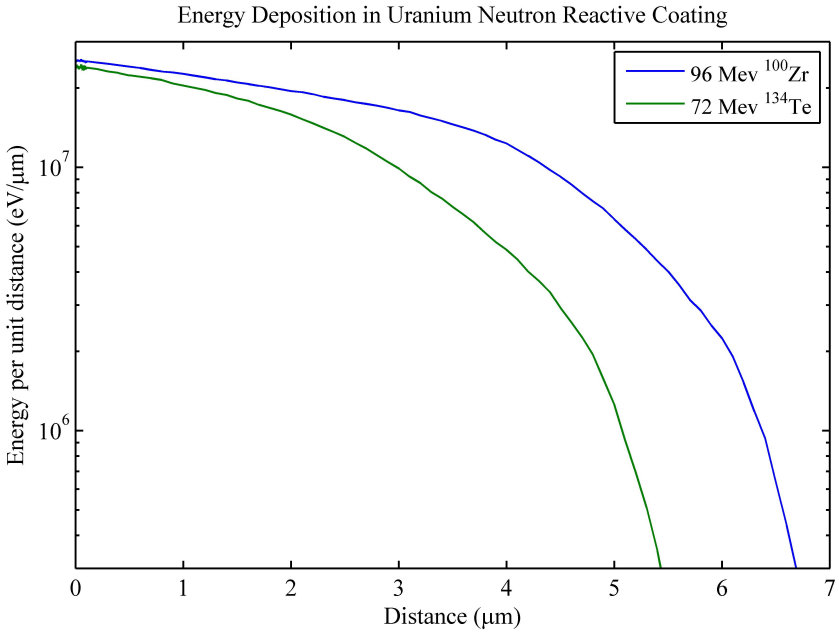


Figure 3.7: Energy deposition per unit distance for fission fragments in natural uranium (density 19.043 g cm^{-3}) [17, 18].

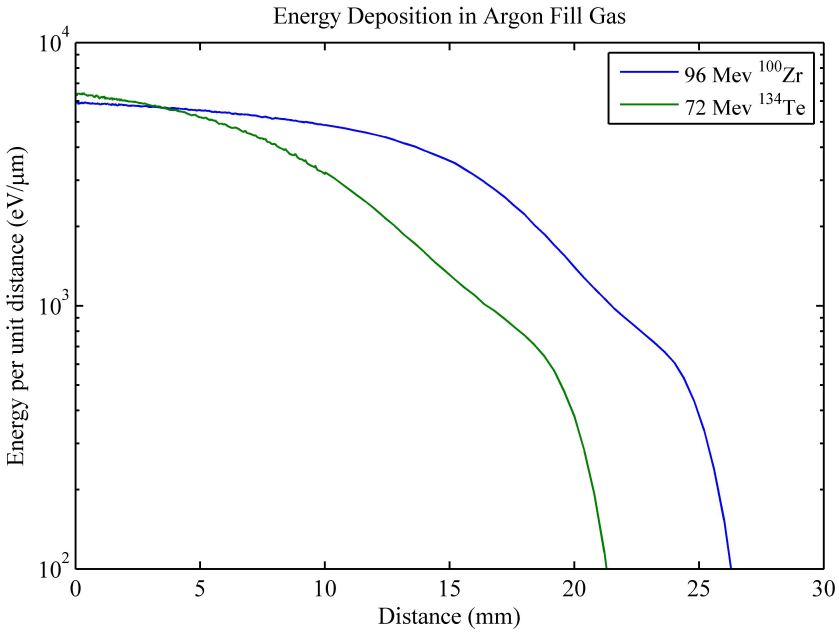


Figure 3.8: Energy deposition per unit distance for fission fragments in 1 atm of argon fill gas [17, 18].

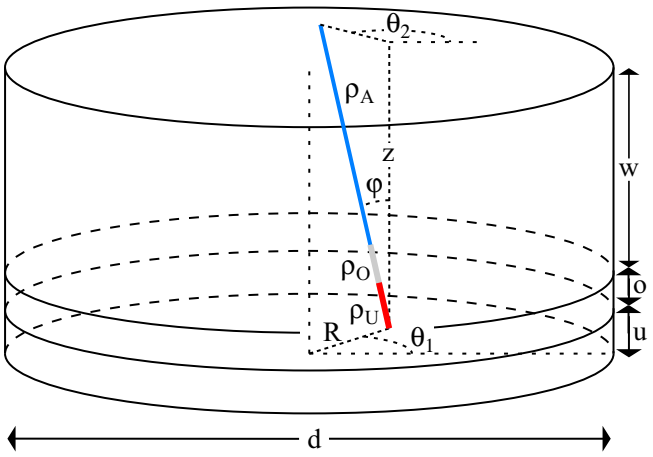


Figure 3.9: Modified 3-dimensional geometry used to account for particle energy loss in the neutron reactive coating and a platinum overcoat layer.

deposition in the fill gas to occur further towards the right of the curve shown in Figure 3.8. Therefore, the total integrated energy deposition in the fill gas is substantially lowered.

Figure 3.9 illustrates the additional parameters u and o which define the neutron reactive coating thickness and the platinum overcoat thickness. A new random number is introduced as

$$w_u = u r_5 \tag{3.22}$$

in which w_u indicates the distance from the top of the neutron reactive coating where the fission reaction occurs. The same analysis equations used for the corrected 3-dimensional model are also relevant here with the substitution that the maximum vertical path length in the neutron reactive coating, the platinum overcoat layer, and the argon fill gas is, respectively,

$$z_{U,\max} = \begin{cases} z_2, & z_2 \leq w_u \\ u - w_u, & z_2 > w_u \end{cases}, \tag{3.23}$$

$$z_{O,\max} = \begin{cases} 0, & z_2 - w_u < 0 \\ z_2 - w_u, & 0 \leq z_2 - w_u \leq o \\ o, & z_2 - w_u > o \end{cases}, \tag{3.24}$$

and

$$z_{A,\max} = \begin{cases} 0, & z_2 - w_u - o < 0 \\ z_2 - w_u - o, & 0 \leq z_2 - w_u - o \leq w \\ w, & z_2 - w_u - o > w \end{cases} . \tag{3.25}$$

The path length in each region may now be found using the methods from before as

$$\rho_U = \sqrt{x_2^2 + y_2^2 + z_2^2}\frac{z_{U,\max}}{z_2}, \tag{3.26}$$

$$\rho_O = \sqrt{x_2^2 + y_2^2 + z_2^2}\frac{z_{O,\max}}{z_2}, \tag{3.27}$$

and

$$\rho_A = \sqrt{x_2^2 + y_2^2 + z_2^2}\frac{z_{A,\max}}{z_2}. \tag{3.28}$$

The energy lost (E_{LU}) to the neutron reactive coating is found by integrating the energy deposition per unit distance, dE_U/dx, for the path length, or

$$E_{LU} = \int_0^{\rho_U} \frac{dE_U}{dx} dx. \tag{3.29}$$

The equivalent distance of energy loss in the platinum overcoat must be found by determining the distance (x_{LO}) that satisfies

$$E_{LU} = \int_0^{x_{LO}} \frac{dE_O}{dx} dx \tag{3.30}$$

where dE_O/dx is the energy deposition per unit distance in the platinum overcoat layer. Once x_{LO} has been found, the energy loss (E_{LO}) in the platinum overcoat layer may be calculated as

$$E_{LO} = \int_{x_{LO}}^{\rho_O + x_{LO}} \frac{dE_O}{dx} dx. \tag{3.31}$$

Then the equivalent distance of energy loss in the argon fill gas must be found by determining the distance (x_{LA}) that satisfies

$$E_{LU} + E_{LO} = \int_0^{x_{LA}} \frac{dE_A}{dx} dx \tag{3.32}$$

where dE_A/dx is the energy deposition per unit distance in the argon fill gas. Once x_{LA} has been found, the total energy deposition (E_D) in the argon fill gas may be calculated as

$$E_D = \int_{x_{LA}}^{\rho_A + x_{LA}} \frac{dE_A}{dx} dx. \tag{3.33}$$

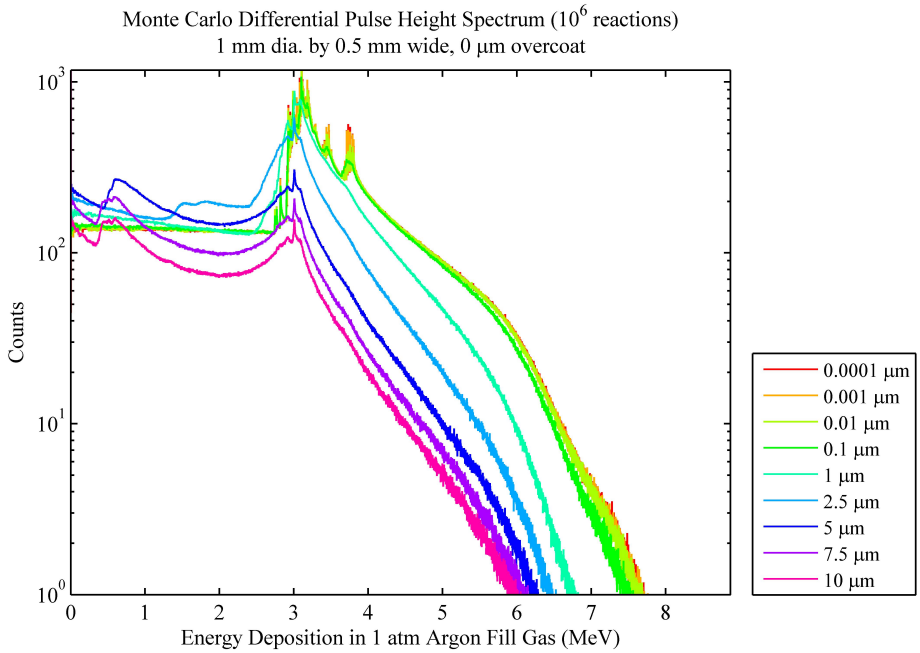


Figure 3.10: Differential pulse height spectra for various natural uranium neutron reactive coating thicknesses per 10^6 events over 8192 channels in an MPFD^3-T sized chamber.

With E_D determined for all of the fission fragment nuclides, it may be multiplied by the nuclide's fission yield and then added to the detector's differential pulse height spectrum. The effects of different natural uranium neutron reactive coating thicknesses for MPFD^3-T sized detectors is shown in Figure 3.10. All of these spectra contain the same total number of counts (1×10^6) but as the neutron reactive coating is made thicker an increasing number of events is shifted into the lowest energy bin.

In order to verify that the simulation was indeed producing reasonable results, a large 30 mm diameter by 30 mm wide detector was simulated. The resulting spectra shown in Figure 3.11 has the expected double hump feature indicating that the simulation is correct.

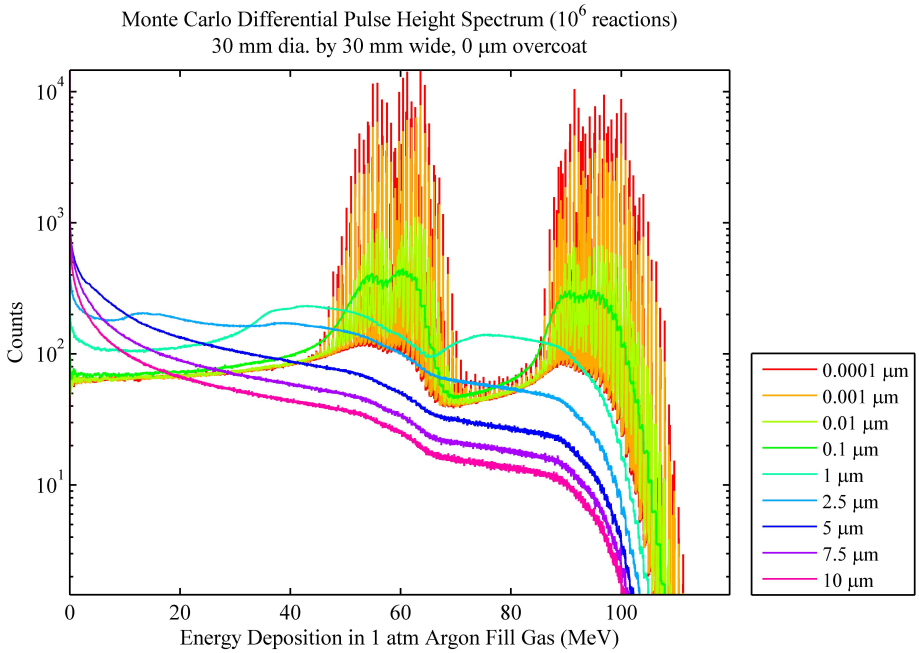


Figure 3.11: Differential pulse height spectra for various natural uranium neutron reactive coating thicknesses per 10^6 events over 8192 channels in a large sized chamber (30 mm diameter by 30 mm wide).

3.2 Lower Level Discrimination

The maximum energy deposited by unwanted events, or background, will determine where the lower level discriminator (LLD) must be set. The events which would constitute background would be electrons, or beta particles, photons, either gamma-rays or x-rays, and alpha particles. While a substantial percentage of the gamma-rays interacting in the fill gas will originate from outside the detector, the remaining background radiation will be generated inside the detector's neutron reactive coating, subsequent fission products or due to secondary reactions. MCNP (Monte Carlo N-Particle - see Section §B.1.5) was previously used to determine the energy deposition from electrons and photons in an argon fill gas environment at 1 atm for different chamber widths (see Figures 3.12 and 3.13) [1]. These plots have been modified to account for a maximum gamma-ray energy of 11.2589 MeV for ^{20}Na and a maximum beta particle energy of 19.636 MeV for ^{150}Er [19]. The shape

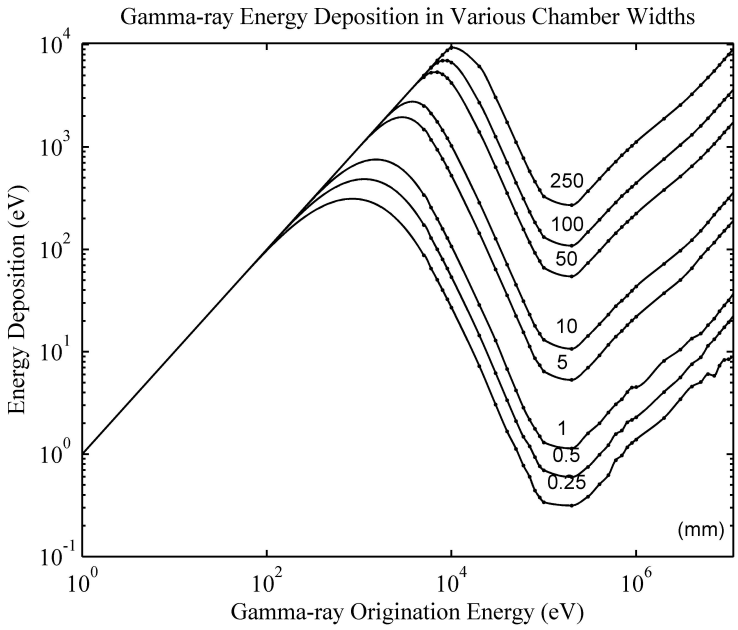


Figure 3.12: Data (dots) and spline fit (lines) of gamma-ray energy deposition in various chamber widths (mm) of argon gas (updated from [1]).

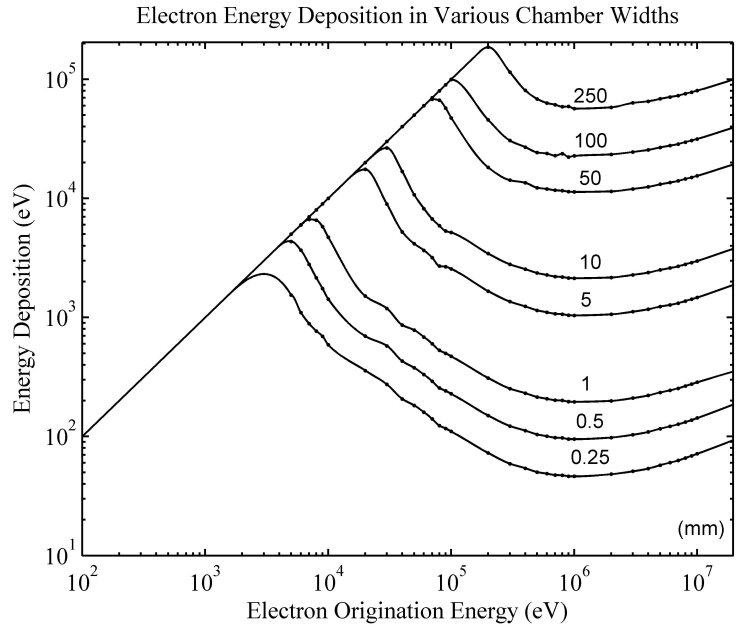


Figure 3.13: Data (dots) and spline fit (lines) of electron energy deposition in various chamber widths (mm) of argon gas (updated from [1]).

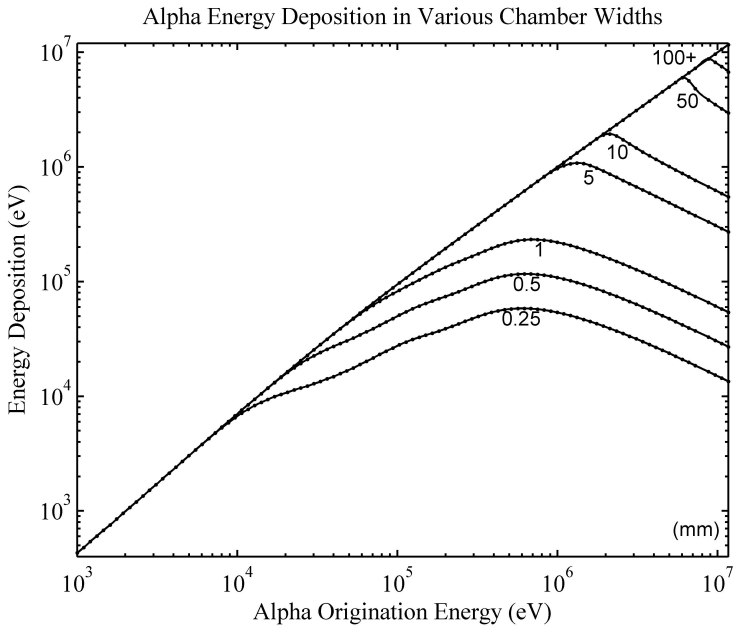


Figure 3.14: Data (dots) and spline fit (lines) of alpha particle energy deposition in various chamber widths (mm) of argon gas.

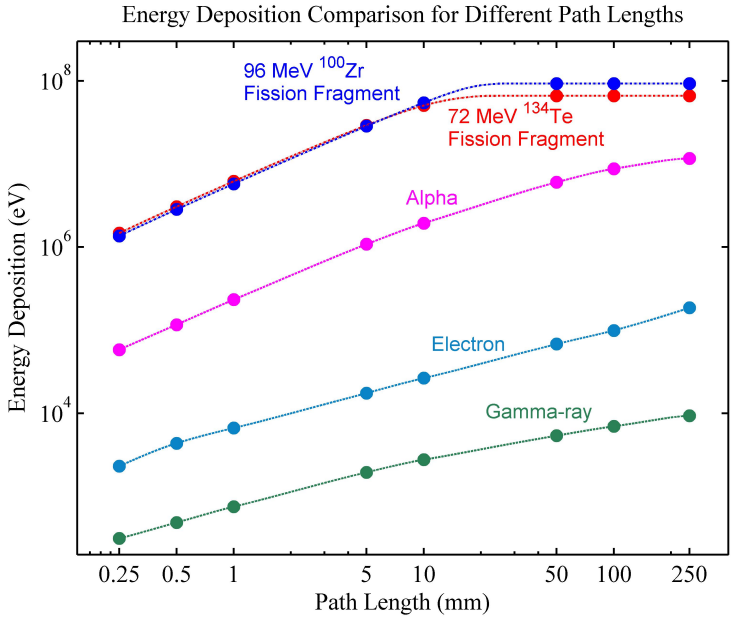


Figure 3.15: Maximum energy deposition comparison of background radiation to fission fragments for various chamber widths.

of Figure 3.12 from low energy to high energy is first restricted by the fact that the energy deposited cannot exceed the origination energy of the gamma-ray. As the energy increases, the photoelectric cross section decreases the energy absorption until absorption is again increased by the addition of the rising Compton scattering cross section and subsequent secondary electrons. A similar TRIM analysis of the alpha-particle energy deposition, shown in Figure 3.14, was added with a maximum alpha-particle energy of 11.65 MeV for ^{212}Po [17, 19, 20]. The maximum possible background energy deposition compared to the energy deposition of some typical fission fragments is shown in Figure 3.15. From this plot it can be seen that the LLD setting will be determined by the alpha particle energy deposition. In addition, the fission products will deposit significantly more energy than any background events, even for very small chamber widths. Table 3.1 lists the maximum energy deposition of the alpha particles, electrons, and gamma-rays of Figure 3.15 compared to the average fission fragment energy deposition based on the curve shown in Figure 3.6.

Table 3.1: Energy deposition for various chamber widths.

Chamber Width (mm)	Average Fission Fragment Energy (keV)	Maximum Alpha Particle Energy (keV)	Maximum Electron Energy (keV)	Maximum Gamma-ray Energy (keV)
0.25	1576.8	58.42	2.316	0.3122
0.5	3145.7	116.70	4.370	0.4856
1	6256.9	233.1	6.665	0.7543
5	29611	1083	17.51	1.954
10	53308	1942	26.54	2.772
50	79841	6021	68.17	5.390
100	79841	8772	99.41	6.983
250	79841	11650	186.2	9.376

3.3 Response Rate

The total number of fissions can be estimated by knowing the neutron reactive coating thickness, the detector's diameter and utilizing the central-thimble full-power fission rate of 1.595×10^{10} fissions per gram for natural uranium [1]. Multiplying the fission rate by the ratio of counts above the LLD versus the total number of counts yields the detector's count rate at full-power (250 kW), or 1.2×10^{13} fast n cm^{-2} s^{-1} and 1.0×10^{13} slow n cm^{-2} s^{-1} [16]. The LLD setting was determined by a constant multiple of the maximum alpha-particle energy deposition (Figure 3.15) for the maximum fill gas path length (ρ_{max}), or

$$\rho_{max} = \sqrt{d^2 + w^2} \tag{3.34}$$

where d is the chamber diameter and w is the chamber width. The results for MPFD^3-T sized detectors (1 mm diameter by 0.5 mm width) with different platinum overcoat thicknesses along with some "large" sized chamber variations are illustrated in Figure 3.16. Several interesting observations that can be made from Figure 3.16. The first observation is the characteristic plateau of the large chambers. It is commonly understood that the maximum uranium coating thickness should be between 2 to 3 mg cm^{-2}, or 1 to 1.6 μm for uranium metal, due to the lack of any efficiency gains in the plateau region [5, 21]. This plateau is due to the higher LLD setting which must be used to discriminate out the background radiation. In addition, as the neutron reactive coating thickness is increased, the fission fragments of neutron reactions occurring deeper in the coating lose enough energy before reaching the fill gas that their pulse signal falls below the LLD setting. This "wall effect," a self-absorption effect, is why most in-core fission chambers are operated in a current mode instead of a pulse mode. In the current mode the alpha particle background current remains approximately constant whereas the current from the neutron-fission reactions will increase linearly with power.

However, using smaller fission chambers where both the width and length, in this case diameter, is reduced and thus minimizing the maximum path length causes the LLD setting

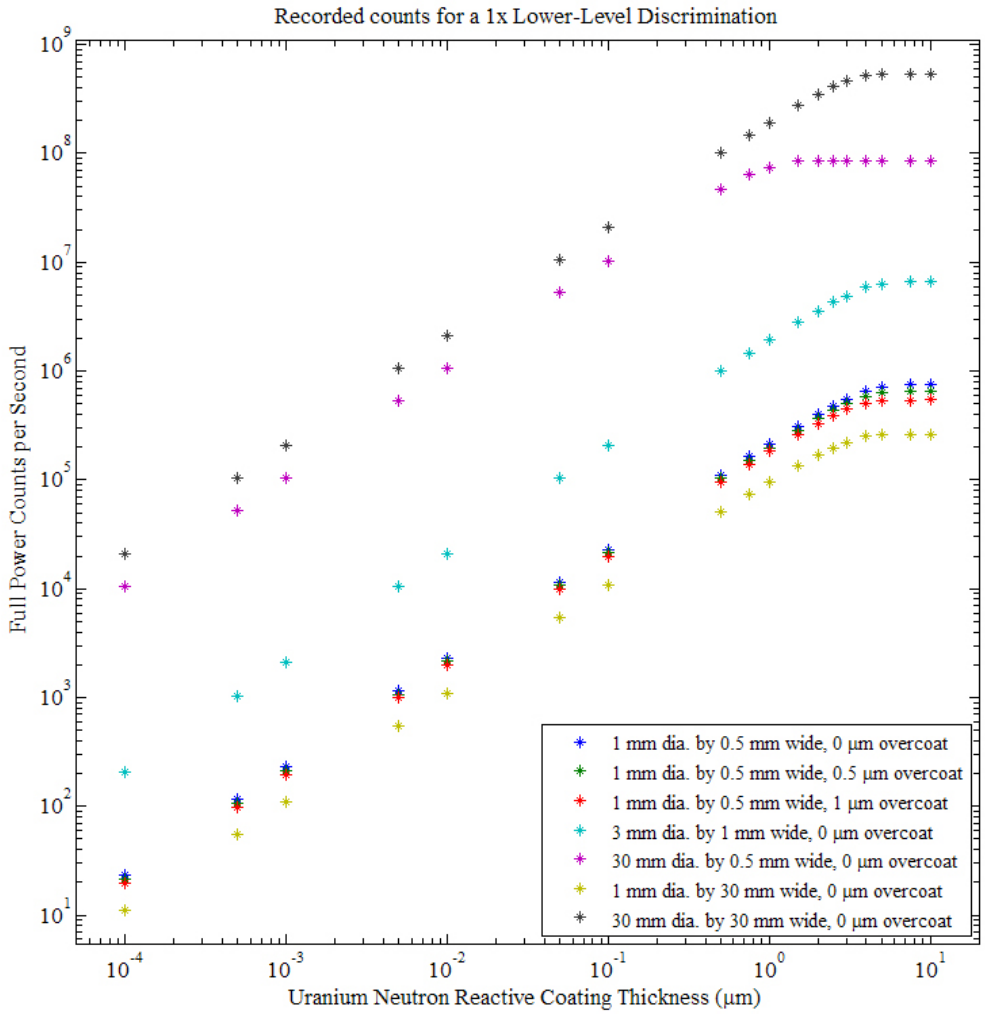


Figure 3.16: Count rate for natural uranium coated MPFDs and "large" fission chambers with various coating thicknesses operated in the central thimble at full power (250 kW).

requirement to also be reduced. While the fission fragment will still lose a significant amount of energy prior to reaching the fill gas, the energy absorbed in the fill gas in a small chamber remains proportionately large to the LLD setting. This different signal shape is why the plateau region for the MPFD^3-T design does not appear until a thickness of just over 5 μm is reached, or nearly 10 mg cm^{-2}. It can also be seen from Figure 3.16 that a platinum overcoat layer has a minimal effect on the MPFD^3-T response because the data points for all three of the 1 mm diameter by 0.5 mm wide detectors lie close to each other.

Table 3.2: Count rates for central thimble full power operations (250 kW) with different neutron reactive coatings (1 μm platinum overcoat layer).

Coating Thickness (μm)	Natural Uranium	93% Enriched ^{235}U*	^{232}Th*
0.5	1.111×10^5	1.310×10^7	938
1	2.142×10^5	2.526×10^7	1809
2	3.960×10^5	4.668×10^7	3344
*based on natural uranium analysis			

Knowing that the platinum overcoat layer has a minimum effect, it is then possible to realize even thicker neutron reactive coatings because the electrode can be built atop the coatings with the neutron reactive coating outside of the chamber. Because uranium naturally oxidizes in air it generates into a charge blocking insulator, which impedes the electron-ion charge motion when the neutron reactive coating is required inside the chamber.

Further, when comparing the response of the 1 mm diameter by 0.5 mm wide MPFD^3-Ts to the large 1 mm diameter by 30 mm wide detectors, it can be seen that the large detector has a worse performance. Because all of these detectors have the same neutron reactive coating surface area, it is strictly an issue of requiring too high an LLD setting which reduces the large detector's pulse mode performance. From experience, the neutron reactive coatings when electroplated perform best near 1 μm thick. Results from Figure 3.16 for natural uranium, 93% enriched ^{235}U, and ^{232}Th neutron reactive coatings are given in Table 3.2 at 0.5, 1, and 2 μm. All of the results in Table 3.2 are based on the natural uranium analysis, but the fission fragment spectrum will be similar enough for a general performance comparison. It is apparent from these results, for the size of detector employed in the MPFD^3-T, that the ^{232}Th coated detectors are impractical because they will not yield high enough count rates for meaningful data. Instead, using a natural uranium coated detector and a 93% enriched ^{235}U coated detector will provide the most information. In addition, the difference in count rates between the two types of detectors will assist in calibrating the

switch between pulse mode and current modes. It has been shown that the detectors will begin to saturate at approximately 56,000 counts per second (1% dead time) and thus it will be necessary to have current mode electronics for both detectors at higher power levels [1].

3.4 Detector Efficiency

While the count rates at full power are a relevant measurement for the Kansas State University TRIGA Mark-II Nuclear Reactor, it has little meaning to the rest of the world. A standard method of describing the performance of fission chambers is to define their efficiency in terms of count rate per nv. Here *n* is defined as the number of neutrons per cubic centimeter and *v* is the average neutron speed in centimeters per second, giving the product commonly known as the neutron flux or n cm^{-2} s^{-1}. Note that the n in nv and in n cm^{-2} s^{-1} are not the same [6]. To convert Table 3.2 into detector efficiencies we only need to divide the count rates by the full power (250 kW) neutron flux in the central thimble, which is 2.21×10^{13} total n cm^{-2} s^{-1} [1, 16]. The resulting detector efficiencies are tabulated in Table 3.3.

Table 3.3: MPFD^3-T detector efficiency in counts per second per nv (n cm^{-2} s^{-1}).

Coating Thickness (μm)	Natural Uranium	93% Enriched ^{235}U*	^{232}Th*
0.5	5.022×10^{-9}	5.921×10^{-7}	4.241×10^{-11}
1	9.686×10^{-9}	1.142×10^{-6}	8.179×10^{-11}
2	1.790×10^{-8}	2.111×10^{-6}	1.512×10^{-10}
*based on natural uranium analysis			

3.5 The *Bonus* Result

When thinking further about the results illustrated in Figure 3.16, it becomes evident that the benefits seen in the MPFD^3-T sized detectors could easily be incorporated into larger detectors. Figure 3.17 demonstrates that the longest path length in a cylindrical type fission chamber is the diagonal of the chamber. It is the distance along this path that must be taken into consideration when setting the LLD since this determines the maximum energy that an alpha particle can deposit in the chamber. In an MPFD^3-T this distance is short enough to allow the LLD to be reduced and thus allows additional lower energy fission fragments to be counted. This same effect can be used in larger fission chambers by simply including an alpha blocking mesh inside the chamber as demonstrated in Figure 3.18. The electrodes do not need to be broken into pixels and because the alpha blocking mesh should be made from an insulative material, it should have no affect on the electric field of the fission chamber. In fact, an alpha blocking mesh made from alumina (Al_2O_3) only needs to be 45 μm thick in order to block the highest energy alpha particles (11.65 MeV for ^{212}Po) [17, 19, 20]. Therefore, while probably needing to be thicker to be structurally sound, thin barriers can

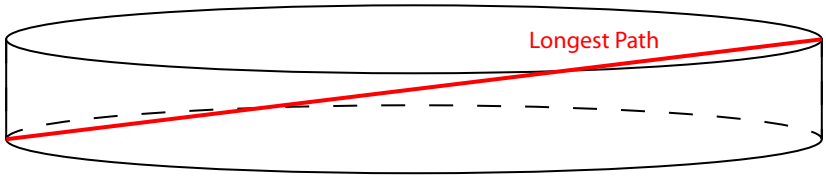


Figure 3.17: The longest path, or the path that will deposit the most energy, is the diagonal of a cylindrical detector.

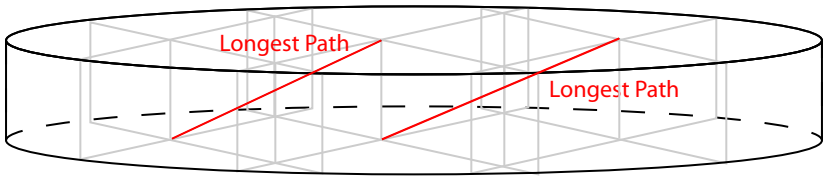


Figure 3.18: Inserting an alpha blocking mesh will divide the longest path lengths and allow the LLD to be lowered, thus improving the detector performance.

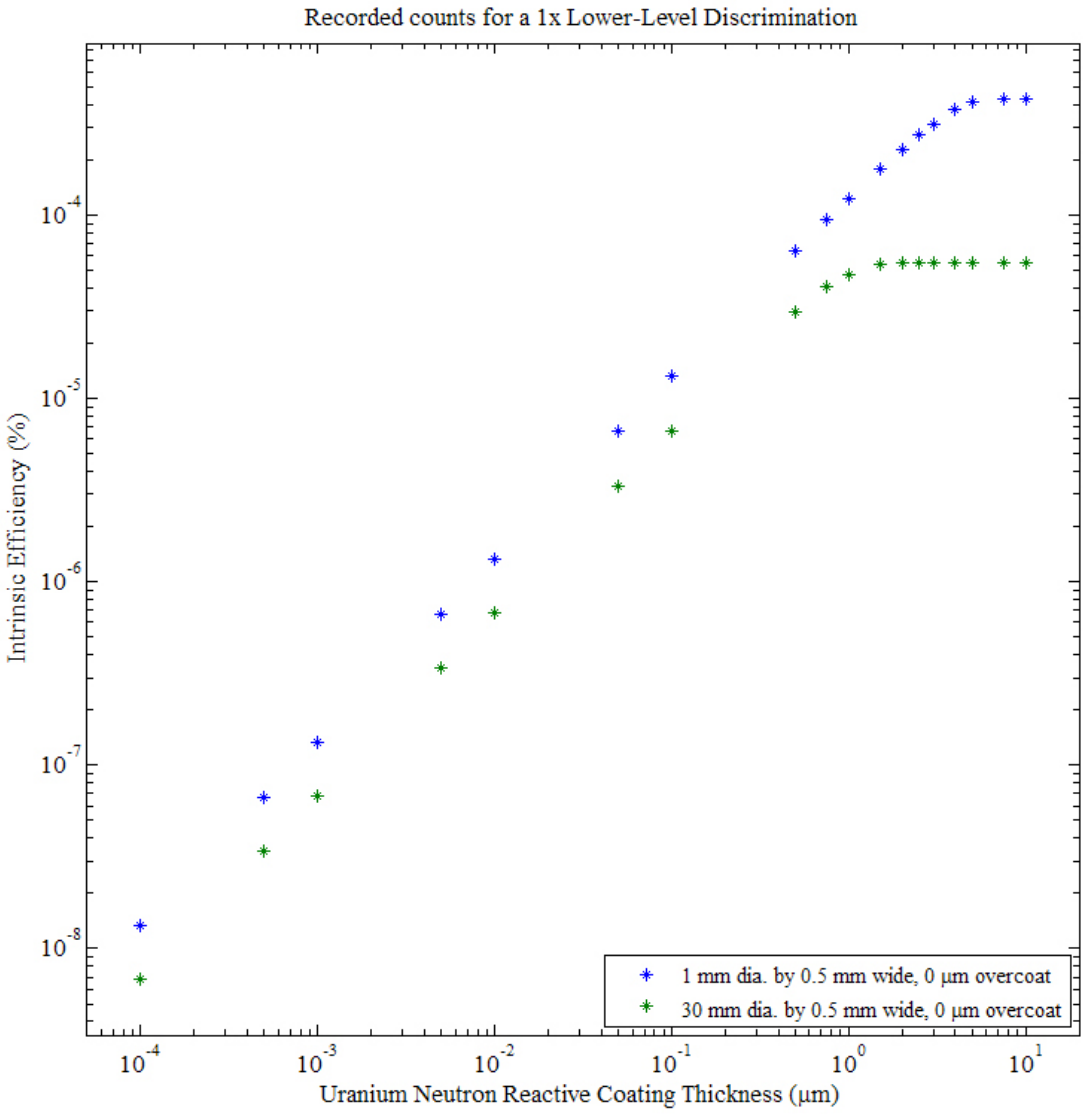


Figure 3.19: Intrinsic efficiency comparison between a small chamber and a large chamber.

be used which have a negligible effect on the overall sensitive surface area of the fission chamber. These alpha barriers could be easily incorporated into almost any existing fission chamber design. For comparison, the count rate per nv normalized to a 1 cm^2 surface area, or the intrinsic efficiency, is shown in Figure 3.19 to demonstrate the improvement of dividing a large chamber into smaller chambers. According to this simplified analysis, the minimum intrinsic efficiency increase would be 96% for coatings less than 0.05 μm

thick and for a 1 μm thick coating the intrinsic efficiency would rise from 4.69×10^{-7} for the large chamber to 1.23×10^{-6} for the meshed chamber, an increase of 2.6 times. With a 7.5 μm thick neutron reactive coating the intrinsic efficiency would further increase from 5.45×10^{-7} up to 4.32×10^{-6}, nearly an increase of 8 times.

3.6 Electron-Ion Induction Timing

The average electron-ion induction time can be found by determining how long it takes the fission fragment to traverse the fill gas in the fission chamber. These calculations only require a small addition to the work already completed in Section §3.1. First, the kinetic energy of the fission fragment must be found when it enters (KE_1) and leaves (KE_2) the fill gas. These are found as

$$KE_1 = E_0 - E_{LU} - E_{LO} \tag{3.35}$$

and

$$KE_2 = KE_1 - E_D \tag{3.36}$$

where E_0 is the fission fragment's original energy, E_{LU} is the energy lost in the neutron reactive coating, E_{LO} is the energy lost in the platinum overcoat layer, and E_D is the energy deposited in the argon fill gas. The velocity (v) of the fission fragment can then be found using the general equation

$$v = c\sqrt{1 - \left(\frac{mc^2}{KE + mc^2}\right)^2} \tag{3.37}$$

where c is the speed of light in a vacuum and m is the fission fragment's mass. Then the average time (t_i) it takes for the fission fragment to cross the fill gas can be found by dividing the path length (ρ_A) by the average velocity as in

$$t_i = \frac{2\rho_A}{v_1 + v_2}. \tag{3.38}$$

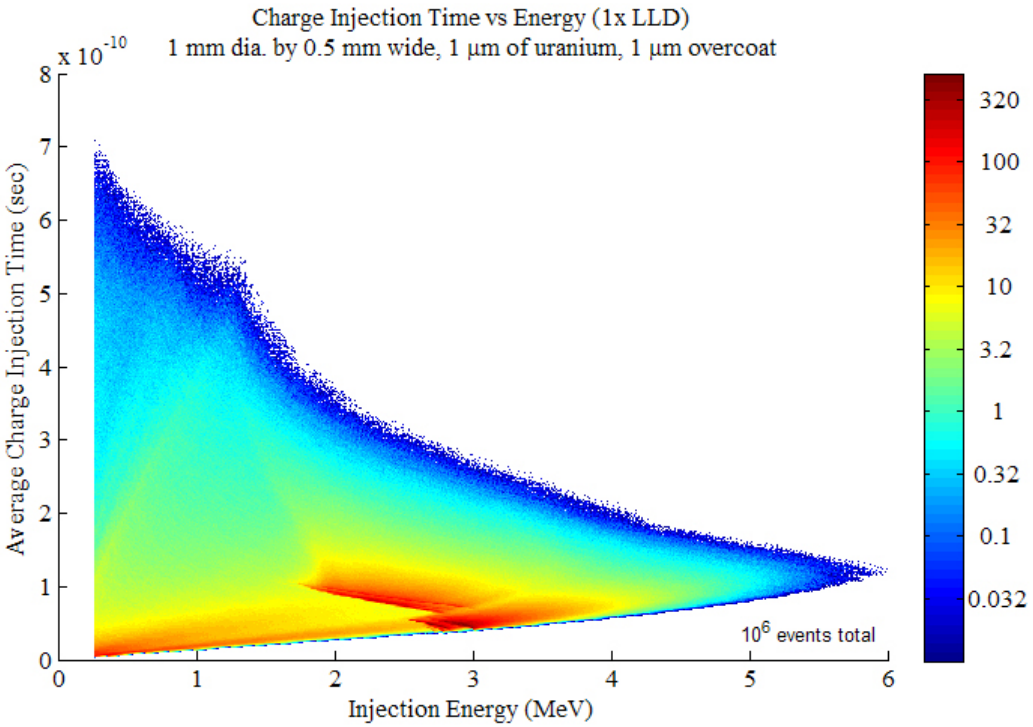


Figure 3.20: Histogram of the average charge injection time versus the injection energy.

Plotting the average induction time versus the energy departed in the fill gas gives the three-dimensional histogram shown in Figure 3.20. Table 3.4 lists the mean, fastest, and slowest charge injection times and associated energies along with the highest energy pulses and its associated injection time for different neutron reactive coating thicknesses.

Table 3.4: Charge injection times and energies for various natural uranium neutron reactive coating thicknesses with a 1 μm platinum overcoat layer.

Coating	Mean Time		Fastest Time		Slowest Time		Highest Energy	
Thickness	Time	Energy	Time	Energy	Time	Energy	Time	Energy
(μm)	(ps)	(MeV)	(ps)	(MeV)	(ps)	(MeV)	(ps)	(MeV)
0.5	61.9	1.852	3.52	0.260	691	0.260	123	6.052
1	64.2	1.683	3.52	0.260	709	0.260	123	5.991
2	67.9	1.376	3.52	0.260	752	0.260	124	6.000

3.7 Pulse Mode Signals

There are three main factors which go into the shape of a fission chamber's pulse. The first is the charge injection time which has already been calculated. The other two were explored in prior work and are due to the ion and electron drift velocities in the fill gas [1]. These drift velocity curves are provided in Figure 3.21 [22–24]. The time (t_d) it will take the electrons and ions to cross the 0.5 mm wide (w) MPFD^3-T is calculated from the drift velocity (v_d) by

$$t_d = \frac{w}{v_d}. \tag{3.39}$$

The drift times for different operating voltages are tabulated for both electrons and ions in Table 3.5. Using this information and the data in Table 3.4, the current pulse shape can be calculated. For simplicity, it is assumed that the injected charge is uniformly distributed

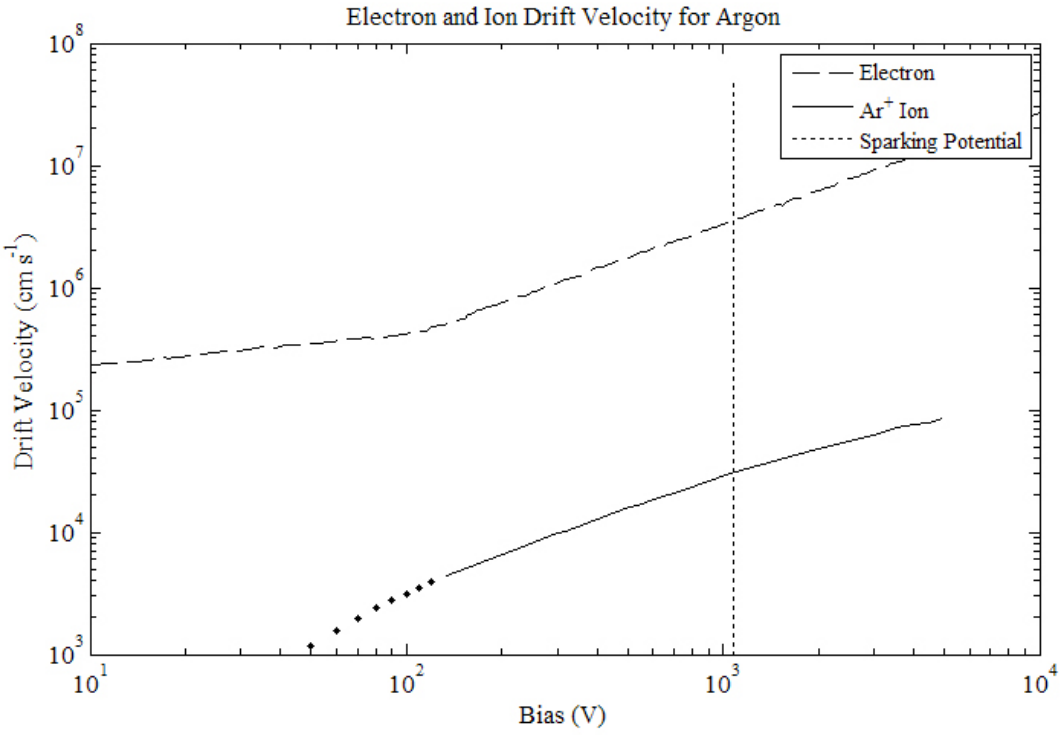


Figure 3.21: Electron and ion drift velocity in 1 atm argon fill gas versus bias voltage (adapted from [1]).

Table 3.5: Drift times for electrons and ions.

Bias (V)	Drift Time (ns)	
	Electron	Ion
200	66.1	7479
300	45.8	5127
400	34.0	3884

across the fill gas chamber, and with diffusion effects ignored, the charges will drift uniformly to the contacts. The maximum current (i) will be formed at the moment all of the charge has been injected and is moving at the drift velocity. Again, for simplicity, this condition is assumed to be instantaneous upon charge injection, and can be expressed as,

$$i = \frac{qE_D}{Wt_d} \tag{3.40}$$

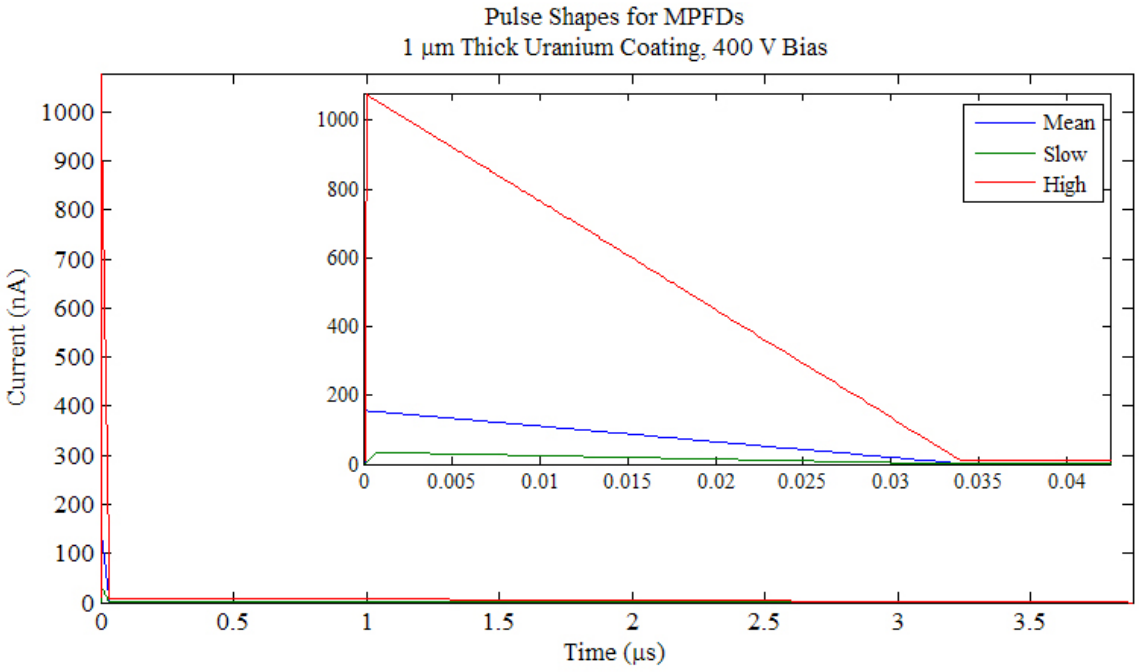


Figure 3.22: Current pulse shape for slow, low energy pulses, average energy pulses, and high energy pulses in a 0.5 mm wide, 1 mm diameter MPFD^3-T with a 1 µm natural uranium coating operated at 400 V.

where q is the charge of an electron (1.622×10^{-19} C), E_D is the energy deposited, and W is the energy dissipation per ion pair value which is 26.4 eV/ion pair for argon gas [5]. Knowing that the current is zero before any charge is injected and returns to zero at time $t_d + t_i$, the approximate pulse shapes for the mean time, slowest time, and highest energy data in Table 3.4 are shown in Figure 3.22. Because both ions and electrons are produced, their signals are summed to produce a single pulse with a fast, high current electron pulse component and a long, low current tail component from the ion pulse.

These pulse shapes are an ideal case and will be distorted when coupled to a preamplifier circuit. However, the knowledge of the idealized case can be coupled into circuit models in order to improve the circuitry design and robustness as will be discussed in Chapter 5.

3.8 Current Mode Signals

As the fission rate increases due to higher reactor power operations the individual pulses will become indistinguishable and the detector will appear to produce a constant current output. The averaged currents ($\bar{i}$) can be calculated through the equation

$$\bar{i} = \frac{qE_D}{Wt_d}F_R t_d = \frac{qE_D}{W}F_R \tag{3.41}$$

where q is the electron charge, E_D is the deposition energy, W is the dissipation energy per ion pair, t_d is the drift time, and F_R is the fission rate. Because the fission rate causes the

Table 3.6: Mean pulse energy with no LLD.

Coating Thickness (μm)	Mean Energy (MeV)
0.5	1.914
1.0	1.757
2.0	1.475

Table 3.7: Average current output for different neutron reactive coatings and thicknesses at different reactor steady-state and peak pulsing power levels.

Coating Material	Coating Thickness (μm)	Steady-State Reactor Power			Pulsing Power	
		1 W	250 kW	1.25 MW	250 MW	3.2 GW
		Current (A)				
Natural Uranium	0.5	1.08×10^{-14}	2.69×10^{-9}	1.35×10^{-8}	2.69×10^{-6}	3.45×10^{-5}
	1	1.96×10^{-14}	4.90×10^{-9}	2.45×10^{-8}	4.90×10^{-6}	6.27×10^{-5}
	2	3.21×10^{-14}	8.02×10^{-9}	4.01×10^{-8}	8.02×10^{-6}	1.03×10^{-4}
93% Enriched ^{235}U*	0.5	1.27×10^{-12}	3.17×10^{-7}	1.59×10^{-6}	3.17×10^{-4}	4.06×10^{-3}
	1	2.31×10^{-12}	5.77×10^{-7}	2.89×10^{-6}	5.77×10^{-4}	7.39×10^{-3}
	2	3.78×10^{-12}	9.45×10^{-7}	4.73×10^{-6}	9.45×10^{-4}	1.21×10^{-2}
Thorium*	0.5	9.09×10^{-17}	2.27×10^{-11}	1.14×10^{-10}	2.27×10^{-8}	2.91×10^{-7}
	1	1.65×10^{-16}	4.13×10^{-11}	2.07×10^{-10}	4.13×10^{-8}	5.29×10^{-7}
	2	2.71×10^{-16}	6.77×10^{-11}	3.39×10^{-10}	6.77×10^{-8}	8.67×10^{-7}
*based on natural uranium analysis						

drift times to cancel out there is no longer a dependence on bias voltage, so long as the bias voltage remains high enough to collect all injected ions and electrons. When operated in current mode, the LLD no longer applies and all pulses above and below the LLD setting contribute to the signal. Therefore, Table 3.6 lists the mean pulse energy which would be integrated by the current mode electronics. Using the fission rates at 250 kW given in Section §3.3, the calculated current levels at various reactor power levels are listed in Table 3.7.

Chapter 4

Construction and Deployment

Scaling up the construction from one MPFD to the construction and deployment of an array of 48 MPFD^3-Ts requires not only some modifications, but also a deeper understanding of the processes. Described in this chapter are the mass production processof these detectors and the steps required to optimize and complete the processes.

4.1 Electrical Contacts and Neutron Reactive Coatings

Several steps were added to the electrode deposition process in order to improve adhesion compared to the prototype detectors built as part of the previous research [1]. Efforts also included calibration and improvement of the electroplating system used to deposit the neutron reactive coatings.

4.1.1 Contact Adhesion and Robustness

Surface contaminants and remnants from prior coating attempts made adhesion of the finalized electrode design difficult when heated. Nearly all of the substrates were initially coated with 100 Å of titanium and 1 μm of platinum using the electron beam (e-beam) evaporator. Unfortunately, analysis of the coatings revealed that the platinum was only 0.1 μm thick due to operator error, which is less than the surface roughness of the alumina substrates. A heated Aqua Regia (3:1 HCl:HNO_3) etch was used on all of the substrates in order to remove the contacts. New contacts were deposited using the correct coating thicknesses, but upon performing the 900°C annealing process, most of the newly deposited contacts peeled off of the alumina surface.

To try and improve adhesion, the ion mill was placed into service, and a custom substrate holder was built (Figure 4.1 and Section §C.4). The holder allowed 36 alumina substrates to be loaded at a time while providing the largest possible surface exposure of the alumina. Once loaded into the ion mill, the processing parameters were adjusted for a 1×10^{-4} Torr nominal base pressure with an argon flow rate of 3.05 (uncalibrated units). The beam current was set to 90 mA at 250 V with an accelerator current of 70 mA at 250 V and a discharge current of 0.50 A at 50 V. The cathode filament operated at 9.6 A with the neutralizer filament current of 15.1 A and an emission current of 83 mA along with the chuck cooling and rotation set to "on". Operation time was approximately 30 minutes.

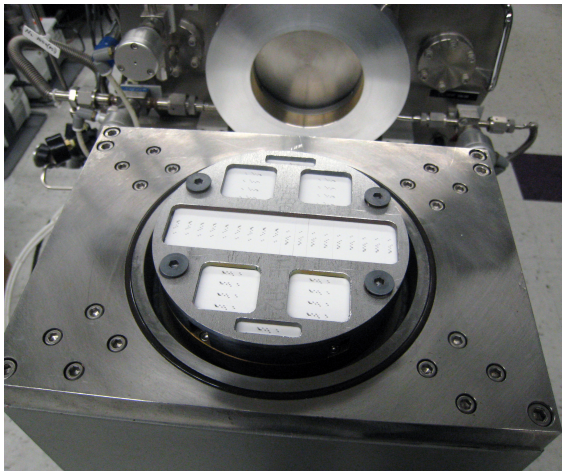


Figure 4.1: Alumina substrate holder for use in the ion mill.

Through the viewing port of the ion mill (Figure 4.2), it eventually became apparent that a faint shadow was becoming visible where the old contacts had been located. This shadow was the thin titanium layer, which the Aqua Regia etch failed to remove. The titanium had not been noticed previously because it had formed an oxide layer that is white, and thus was camouflaged against the white alumina substrate. A buffered oxide etch (BOE), made by diluting hydrofluoric acid (HF), was then used to etch the remaining titanium off of the alumina.

The etched substrates were then cleaned using the ion mill, and new contacts were deposited using the e-beam evaporator. These contacts survived the thermal annealing process with good adhesion. However, attempts to electroplate a natural uranium coating on the contacts failed due to changes in the pH of the electroplating solution during the coating process. Due to the alumina's porous structure, the substrates had apparently absorbed

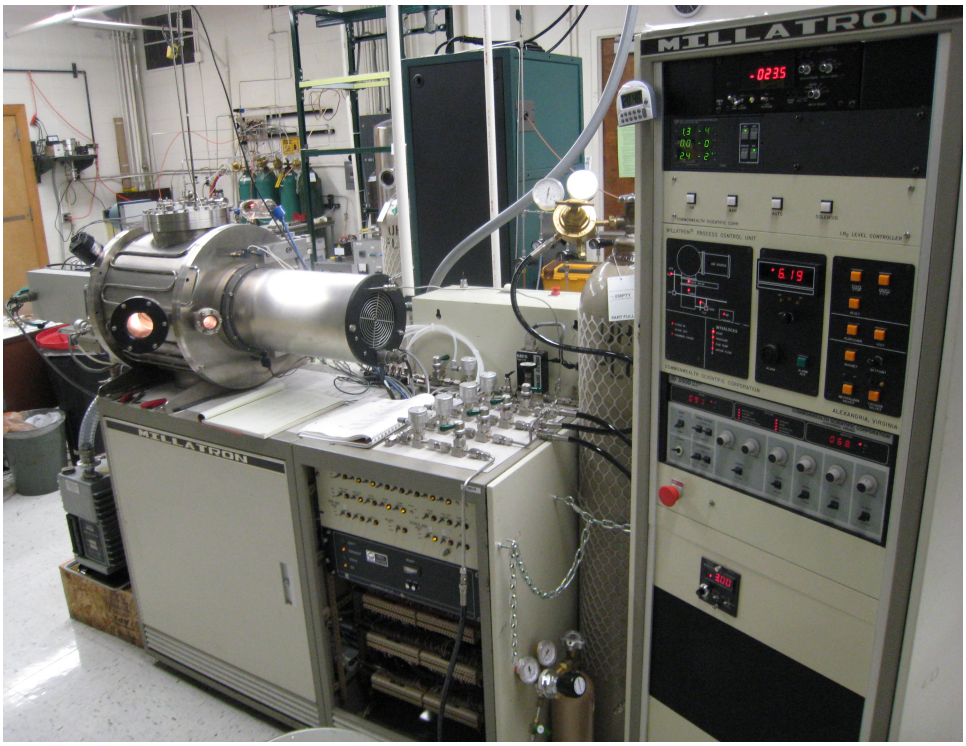


Figure 4.2: Etching of the alumina substrates using the Millitron ion mill.

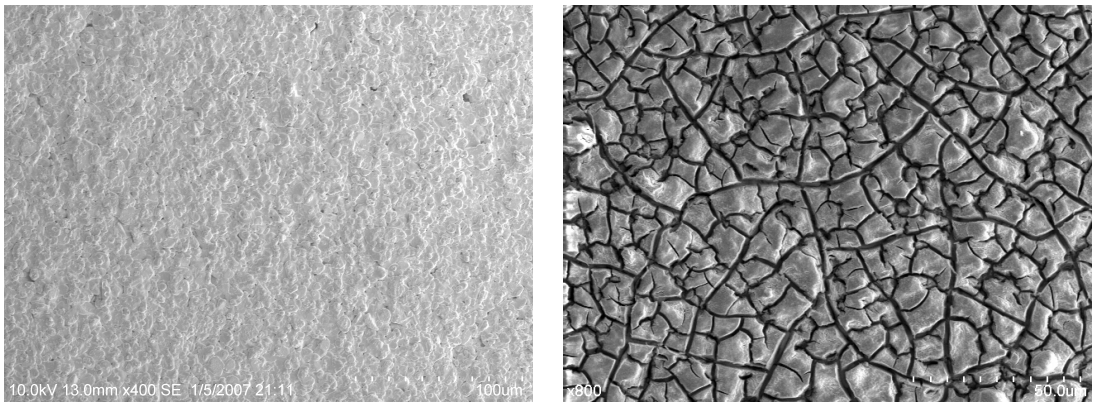
some of the HF during the etch process, and when immersed in the electroplating solution, they released some of the HF into the solution, thus changing the solution's pH level. Because of the HF contamination, it was determined that the alumina substrates could not be reused and new substrates were ordered.

A Dektak^3 profilometer was used to check the surface roughness of the alumina substrates and the platinum electrodes. Two types of measurements were reported by the profilometer: R_a and R_q. R_a is the arithmetic average roughness and R_q is the root-mean-square (RMS) of roughness corresponding to R_a. The new alumina substrates were measured with R_a = 5889 Å and R_q = 7258 Å. After the substrates were cleaned in the ion mill, the measurements were reduced slightly to R_a = 5825 Å and R_q = 7187 Å. Electrodes were then evaporated onto the alumina substrates using approximately 500 Å of titanium, followed by 10 kÅ, or 1 μm, of platinum. The titanium thickness was increased in order to improve adhesion due to the surface roughness of the alumina substrate. The surface roughness of

the electrodes were measured as R_a = 4163 Å and R_q = 5171 Å. After the 900°C annealing process, the surface roughness measurements increased to R_a = 5147 Å and R_q = 6134 Å. The increase in surface roughness is evidence that the annealing process removes the structural defects from the amorphous deposition of the titanium and platinum by reducing the number of voids in the electrode's bulk structure [1, 25]. The electrode resistance was also reduced from 1.682 Ω cm^{-1} down to 0.888 Ω cm^{-1}, which is due to better electrical contact through void reduction in the platinum electrode.

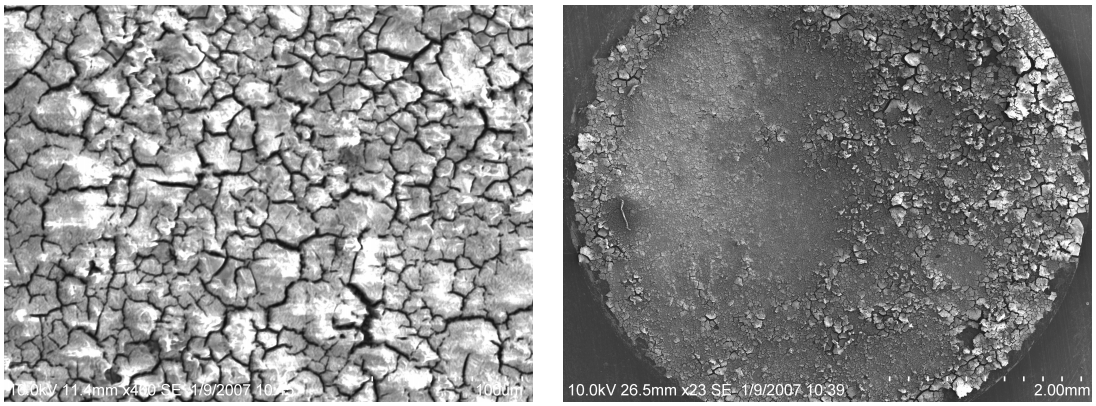
4.1.2 Characterization of ASCEFI

The Automated System for the Consistent Electroplating of Fissionable Isotopes (ASCEFI) was custom designed and built for producing the thin fissionable coating on one of the electrodes of the alumina substrates. The system was designed to provide two different coating options, originally 93% enriched ^{235}U and ^{232}Th. From the analysis in Section §3.3, the ^{232}Th coating would not produce enough counts in the Kansas State University TRIGA Mark-II Nuclear Reactor and so natural uranium and 93% enriched ^{235}U coatings were selected instead. In order not to waste any of the expensive enriched uranium, ASCEFI was initially tested using natural uranium. After the basic problems had been worked



(a) Platinum Contact at 400x Magnification (b) Natural Uranium at 800x Magnification

Figure 4.3: Scanning electron microscope (SEM) image of an (a) uncoated platinum electrode and a (b) natural uranium coated electrode.

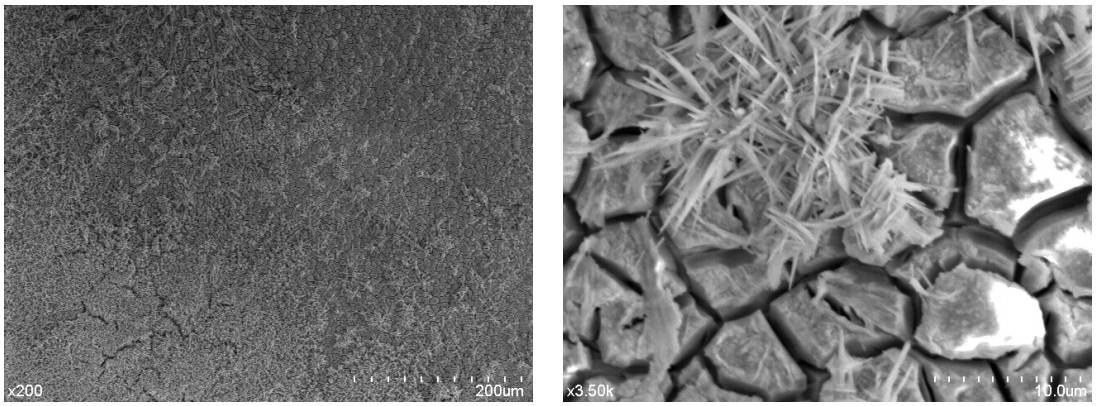


(a) 400x Magnification (b) 23x Magnification

Figure 4.4: SEM images of a ^{235}U unsealed alpha source from Isotope Products. The source was made by electroplating the ^{235}U onto a platinum disk.

out of the system, samples of the electroplated coatings were examined using a scanning electron microscope (SEM). Figure 4.3b is an example of a natural uranium electroplated coating applied to an integral current density of 22.5 C cm^{-2} and magnified at 800x. As a comparison, a ^{235}U unsealed alpha source made by Isotope Products by electroplating a thin layer of ^{235}U onto a platinum disk was also examined using the SEM (Figure 4.4). The "cracked-earth" appearance of the uranium film is evident in both Figure 4.4a and Figure 4.3b, confirming that ASCEFI is working correctly.

With the natural uranium process working, efforts were switched to the 93% enriched ^{235}U coatings. The ^{235}U solution was prepared in the exact same manner as the natural uranium solution. While the ^{235}U coating had the same "cracked earth" appearance as the other coatings, a second layer of spikes had formed as shown in Figure 4.5. X-ray photoelectron spectroscopy (XPS) was performed to try to identify the spikes and any difference in the coating's chemistry. The spikes were identified as an oxalate crystal with magnesium and calcium showing up in the uranium coating. Formic, acetic, and sulfuric acid treatments were tested on the coatings for removal of the oxalate crystals, of which the first two failed to remove the crystals, and the last removed both the crystals and the uranium coating. One thought was that the crystals were forming on the electrode because



(a) 200x Magnification (b) 3500x Magnification

Figure 4.5: SEM images of a 93% enriched ^{235}U coated electrode at (a) 200x magnification and (b) 3500x magnification.

the substrates were at a cooler temperature than the electrolytic solution. Heaters were added to the substrate holders on ASCEFI in order to try to control the crystal growth by warming the substrates slightly above the temperature of the solution. A few sets of coatings were electroplated successfully without crystal formation. However, the problem reappeared. Because the crystal problem could not be solved, it was determined that only the natural uranium coatings would be possible for the purposes of this project. A light coating (17 C cm^{-2}) and a heavy coating (30 C cm^{-2}) of natural uranium were selected for the completion of the project.

Auger Electron Spectroscopy (AES) was performed by Rocky Mountain Laboratories, Inc., on a set of coated substrates in order to try to identify the thickness of the uranium coating in relation to the integral current densities applied by ASCEFI. Shown in Figure 4.6 is the measured thicknesses versus the integrated current densities, along with a least squares line fit. While no error bars were supplied with the measurement data, the roughness of the alumina substrate surface, along with the poor depth resolution in the AES sputter depth profiles, results in inaccurate measurements. For verification, coatings were measured in the SEM, and for 25 C cm^{-2} coatings, the thickness was approximately 3 μm. Again, this thickness was difficult to measure due to surface roughness. Analysis

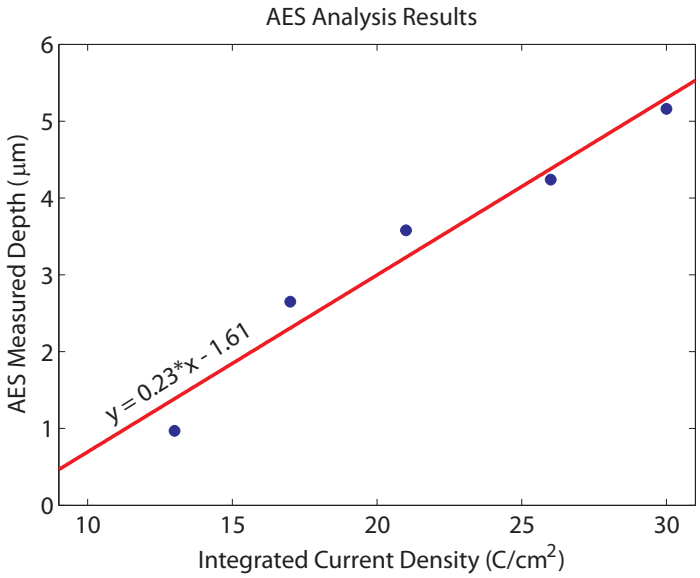


Figure 4.6: Auger Electron Spectroscopy (AES) analysis results for different natural uranium coating thicknesses deposited by ASCEFI.

of freshly deposited coatings using the SEM in the S.M.A.R.T. Laboratory estimated the 17 C cm^{-2} coatings to be 1 μm thick. The reason for the discrepancy is most likely due to the time between when the coatings were deposited and when they were analyzed, which was almost a year. The relatively rapid oxidation of uranium metal in air at room temperature would have increased the thickness of the coatings. The dark black color of the analyzed uranium coatings also confirms that oxidation had occurred. Unfortunately, this analysis cannot be relied upon for determining exactly how much uranium was deposited.

In order to protect the uranium coating from peeling and to improve the conductivity of the uranium coated electrode, 5000 Å of platinum was evaporated on top of the uranium coating. This overcoat layer was annealed at 200°C prior to assembly of the detector.

4.2 Detector Assembly

Described in this section are some of the improvements or changes to the assembly process from what was described in the prior work [1]. Also included in this section is a description of how the individual $MPFD^3$-Ts were assembled into a string of $MPFD^3$-Ts and prepared

for installation into the Kansas State University TRIGA Mark-II Nuclear Reactor.

4.2.1 Gas Channels

The original design of the cavity substrates (Figure 4.7), as presented elsewhere [1], included gas fill and gas diffusion channels. The purpose of the channels is to maintain as high a purity of fill gas as possible by allowing gas to diffuse between the gas pockets and the gas storage chambers. The impurities are generated as fission reactions occur inside the gas pockets, thus generating fission products, which will reduce the charge transport properties of the fill gas. However, when the substrates were originally ordered, no manufacturers were willing to attempt the fabrication of these channels. Several attempts were made to add these channels ourselves, but they all failed, and so the channels were removed from the initial design.

Before reordering the base substrates, options were checked once again for producing the cavity substrates with the channels. Two companies offered to manufacture the substrates based on minor specification changes. After the first company failed to deliver the parts, Ceramic Products, Inc. successfully made and delivered the cavity substrates with the embedded channels.

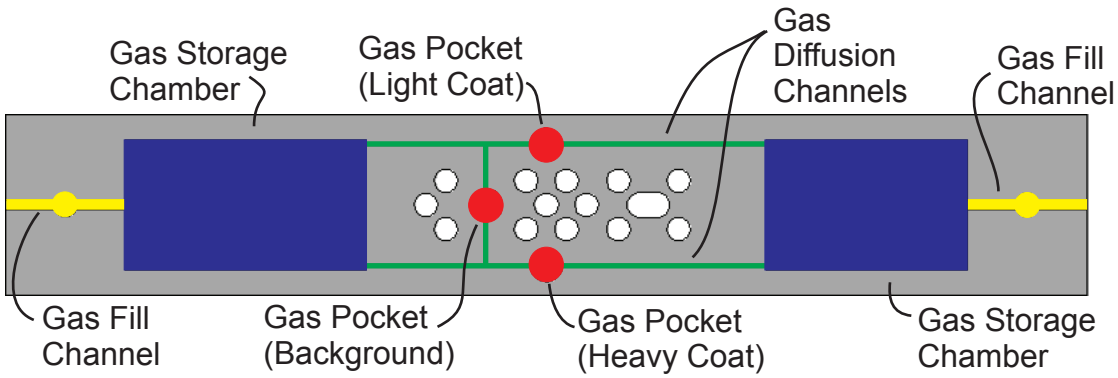


Figure 4.7: Drawing of the cavity substrate (adapted from [1]).

4.2.2 Sealing the Detectors

Utilizing the substrates with gas channels (Section §4.2.1) added an additional layer of complexity to the sealing process. The Aremco 503-VFG-C alumina-based epoxy used to seal the detectors could be applied over the full cavity substrate prior to the addition of the gas channels. Care now had to be taken not only to ensure that no epoxy entered the gas pockets, but also that no epoxy entered the gas channels while maintaining the ability to hermetically seal the detector. Pressing the cavity and base substrates together causes any attempt to avoid the channels to fail, since the epoxy would flow into the channels. Rather than try to avoid filling the channels, a wax was used to fill the channels prior to applying the epoxy. The wax was KIWI brand waxed natural shoe polish, which was loaded into a syringe cartridge and applied using the Zephyrtronics' Air Mill automated dispensing system. The wax would keep any epoxy from entering the gas channels and would evaporate out of the channels during the epoxy's curing process, thus leaving the channels clear.

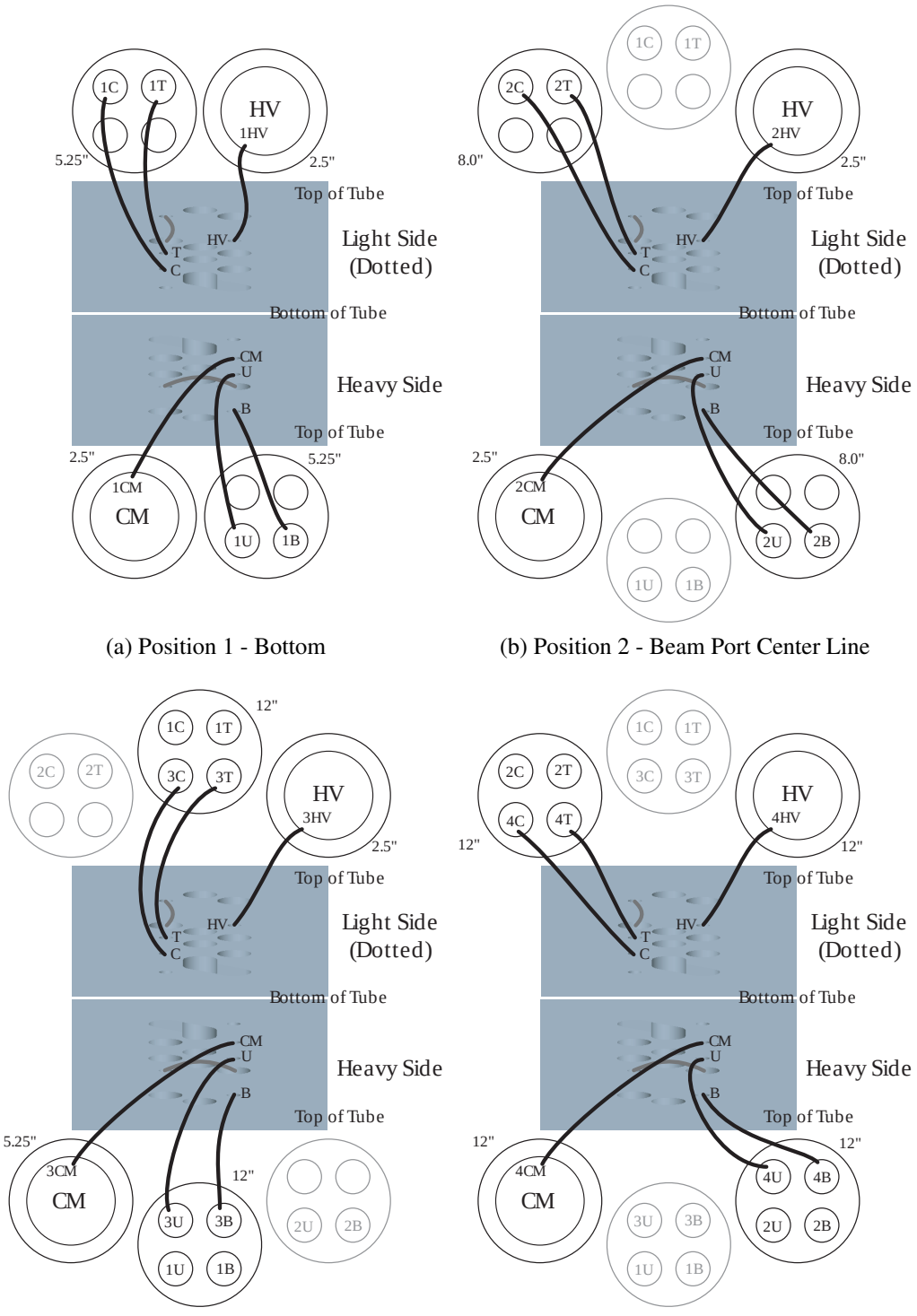
One of the concessions for being able to manufacture the cavity substrates with the gas channels was to reduce the depth of the channels. It was originally intended to plug the gas fill channel (Figure 4.7) with Aremco's 813A alumina epoxy after backfilling the detector with pure argon. This process was now nearly impossible. In addition, it was found that the 503-VFG-C epoxy did a good job of holding the alumina substrates together, but was rarely able to hermetically seal the detectors. Because of these problems, it was decided that the flux-probe tubes would be filled with argon, and the detectors would remain porous, thereby eliminating the necessity of hermetically sealing the detectors. The assembly process of joining the base and cavity substrates together would still be carried out inside the argon-filled glove box. This ensured that any of the detector pockets that did get sealed would have argon trapped inside.

4.2.3 Flux-Probe Tube Loading

Once the detectors were sealed and wires were connected, the individual MPFD^3-Ts had to be loaded into the flux-probe tubes. Original plans included five MPFD^3-Ts per flux-probe tube, but attempts to load this many detectors resulted in too many wires and quickly became impractical if not impossible to accomplish. Instead, only four MPFD^3-Ts were loaded into each tube, one centered 2" above the top of the bottom reflector, one centered on the beam-port center line, one centered on the middle of the meat of the fuel, and one centered 2" below the bottom of the top reflector.

Twenty-four wires per MPFD^3-T string quickly become very difficult to keep track of and without proper planning, shorting of these wires is highly probable. To assist with this process, a set of wiring maps (Figure 4.8) was developed, which showed the orientation and position of the detectors and the alumina insulating tubes. The positions of the alumina insulating tubes and the routing of wires were chosen to minimize the potential of electrical shorts while maintaining a degree of rigidity to keep everything in line. These maps are to be viewed as though you are at the bottom of the tube looking across the face of the detector and into the end of the alumina insulating tubes extending away from you. During fabrication of the detectors, a permanent dot was placed on the base substrate that had the light natural uranium coating while the heavy natural uranium-coated substrate was left bare. In this map, C stands for the thermocouple, CM is the thermocouple common wire, U is the thin coated detector, T is the thick coated detector, B is the background detector, and HV is the common high voltage wire. Tubes with lengths listed next to them identify the lengths of the tubes, while grayed out tubes are continuous when passing that detector.

After all of the wires have been routed and the ends of the wires identified, conductivity tested, and any electrical shorts removed, the detectors are loaded into the tubes as shown in Figure 4.9. Because the alumina insulating tubes are rigid, the end of the wires are threaded through a short, flexible insulating sleeve. The wires are then soldered to the connector circuit boards that provide a transition to shielded and insulated 24 AWG wiring, which



(a) Position 1 - Bottom

(b) Position 2 - Beam Port Center Line

(c) Position 3 - Fuel Center Line

(d) Position 4 - Top

Figure 4.8: Wiring maps for connecting the MPFD^3-Ts and for routing the wires through the alumina insulating tubes.

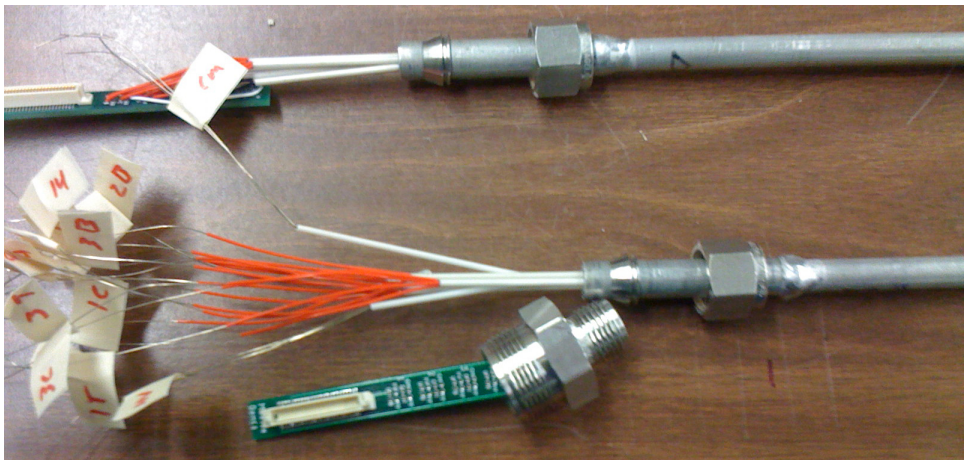


Figure 4.9: Detectors loaded into flux-probe tubes with one set attached to a connector circuit board and the other showing the labeled flying leads.

connects the probes to the electronics. Once soldered, the connector circuit board and the wires are slipped through a bored out 3/8" to 5/8" stainless steel Swagelok connector.

The loaded flux-probe tubes are then completely loaded into a vertical 4" diameter PVC pipe, which has argon slowly flowing into the bottom of the pipe and out a small hole in the top cap. Because the argon is heavier than the air, and since the argon is dry, the flux-probe tubes are dried out and filled with argon. After sitting in the argon for several days, the top of the pipe is removed, and the ends of the flux-probe tubes are plugged using Devcon's Flow-Mix 2 Ton Epoxy (part no. 23145). This two-part epoxy is quick-drying, waterproof, and is low out-gassing, which provides a good seal while not contaminating the fill gas.

The Swagelok connector allows a 5/8" stainless steel tube to be connected to the flux-probe tube. While the alumina insulation and wires will limit some of the neutron streaming through the flux-probe tubes, the stainless steel tubes need to include at least one bend in order to minimize the streaming probability. For the purpose of mechanics and to ease working around the stainless steel and flux-probe tubes, a two-bend option was selected. This configuration allowed the stainless steel tubes to bend shortly after the connection to the flux-probe tubes, angle over to the wall of the reactor tank, and then bend straight up to enter the bottom of the connection boxes located at the top of the tank wall. In order

Priority	Flux Probe	Bend 1		Bend 2		Bend 3		Bend 4		Critical Dimension
		inches*	degrees	inches*	degrees	inches*	degrees	inches*	degrees	
1	A	123	90°	155 - 7/16	90°	---	---	---	---	29.38
2	B	122	90°	155 - 3/4	90°	130 - 9/16	-12.5°	---	---	30.51
3	C	121	90°	156 - 11/16	90°	129 - 9/16	12.5°	---	---	32.50
4	D	120	90°	155 - 1/2	90°	125 - 1/8	-12.5°	---	---	32.38
5	E	119	90°	156 - 9/16	90°	124 - 1/8	12.5°	---	---	34.43
10	F	119	90°	156 - 11/16	90°	135 - 5/8	25°	---	---	34.30
9	G	120	90°	155 - 5/8	90°	143 - 1/2	25°	---	---	32.08
8	H	121	90°	156 - 3/4	90°	134 - 3/4	25°	---	---	32.43
7	I	122	90°	155 - 13/16	90°	142 - 11/16	25°	---	---	30.29
6	J	123	90°	155 - 5/8	90°	150 - 9/16	25°	---	---	28.99
14?	L	123	90°	160	90°	---	---	---	---	33.95
13?	M	122	90°	156 - 13/16	90°	130 - 9/16	12.5°	---	---	31.60
12	N	121	90°	155 - 3/4	90°	129 - 9/16	-12.5°	---	---	31.52
11	O	120	90°	150 - 15/16	90°	125 - 1/8	12.5°	---	---	27.78
15?	P	119	90°	156 - 5/8	90°	124 - 1/8	-12.5°	147 - 3/16	25°	34.32

* To 0° mark (on tool) measured from top of tubing

Note: 16' 7" total vertical length from top of core

Critical Dimension is the horizontal distance between centers looking straight down

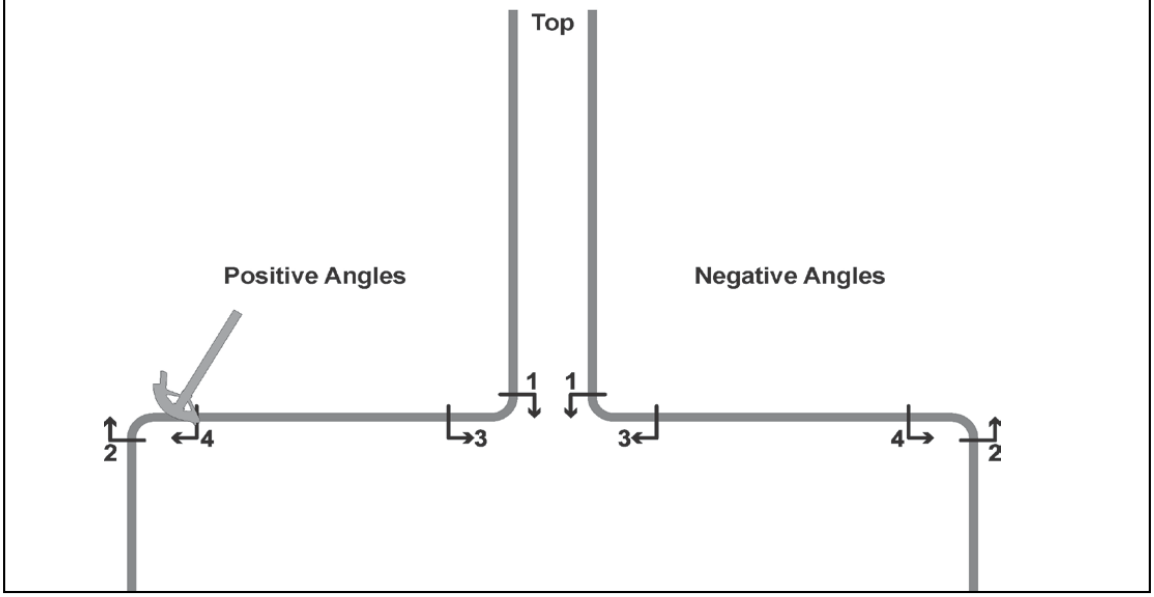


Figure 4.10: Worksheet for how to bend the stainless steel tubing.

to make sure that none of the tubes interfered with each other, the worksheet shown in Figure 4.10 was made. This worksheet shows where and how the conduit bending tool must be positioned in order to bend the stainless steel tubing into the necessary shape. The tubes for flux probes L, M, and P were not made because the shim control rod could come into contact with these tubes, so to be safe, they were left out of the installation plans. While this does reduce the amount of data available, their installation could have impacted the safety of the reactor.

4.3 Detector Array Deployment

Deployment of the MPFD^3-T detector array proved to be far more involved than could have been imagined. Described in the following two sections are some of the side projects that had to be solved prior to deploying the detector array. This discussion is then followed by actual installation, or deployment, of the detectors into the reactor core.

4.3.1 Flux-Probe Hole Restoration

A key to the MPFD^3-T array deployment was the use of 15 of the existing 16 flux-probe holes located in the upper and lower grid plates of the reactor. Available manuals, design descriptions, and technical drawings all specified the upper grid plate to have 8 mm diameter flux-probe holes, while the lower grid plate had 4 mm diameter flux-probe holes. The

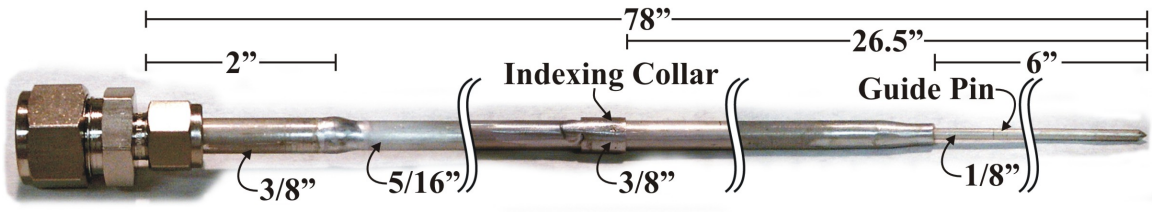


Figure 4.11: Flux-probe design showing (from right to left) the 1/8” lower grid plate pin which is welded inside a 5/16” (0.030” wall) aluminum tube. A 3/8” collar is welded 26.5” from the bottom to provide an indexing guide to the top grid plate. Another 3/8” collar is added to the end of the tube to support a 3/8” to 5/8” Swagelok coupler for connection to a stainless steel tube (Reproduced from [1]).

detector dimensions, the number of wires, and the necessary ceramic insulation limited the inner diameter of the flux probe tubing to be no less than 0.25". Using available standard tubing with a wall thickness of only 0.030" (approximately 1/32") gives an outer tubing diameter of 5/16" (0.3125"), leaving minimal clearance in the 0.315" (8 mm) diameter upper grid plate flux probe holes.

Initial attempts to insert assembled test strings found that the flux-probe holes were too small. The flux-probe tubes were then sanded to reduce their diameter to fit in a 7.5 mm (0.295") diameter hole, but the probes still failed to insert into any of the flux-probe holes. This sanding also reduced the strength of the flux-probe tubes and led to flooding of the flux-probe tubes. Therefore, it was determined that the flux-probe hole diameter would have to be restored to the 8 mm specification.

The first attempt at restoring the hole diameters was made with a reaming bit attached to a 16' extension. A suction collar surrounded the bit and was allowed to float on the extension shaft in order to maintain contact with the upper grid plate (see Figure 4.12). The

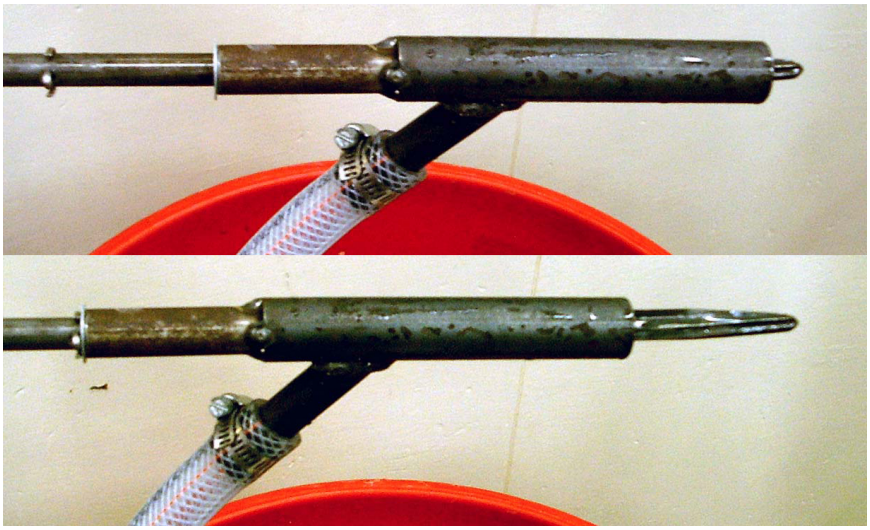
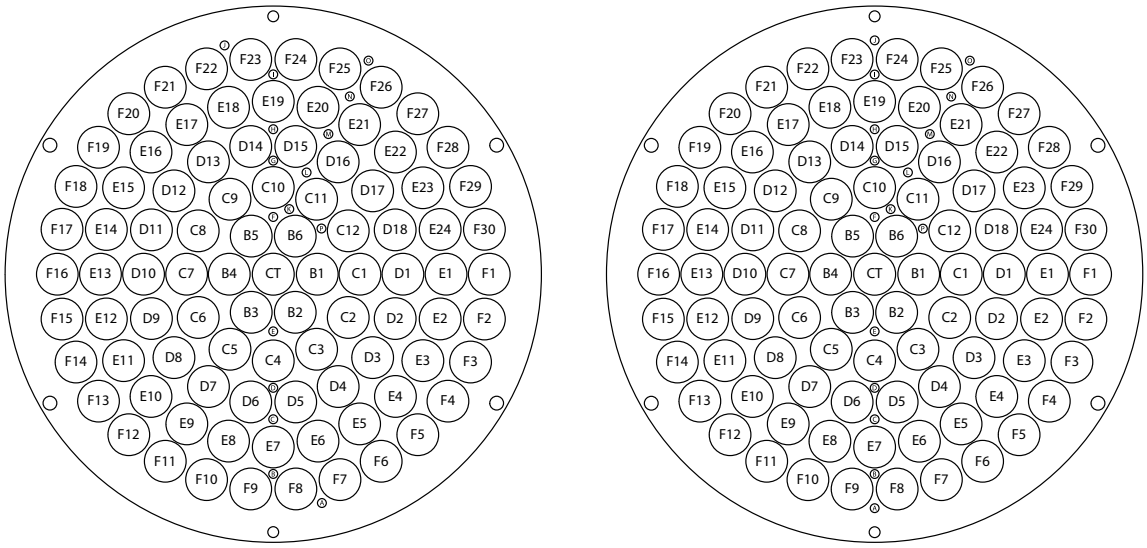


Figure 4.12: Flux-probe hole restoration tool showing the tapered reaming bit and floating suction collar. The collar would sit on the upper grid plate sucking away shavings while the reaming bit drilled out the hole.

Figure 4.13: Ingersoll Rand ARO Par-A-Matic pneumatic self-feed drill used in the restoration of the flux-probe holes.

collar was attached to a hose, which was connected to a filter and then a pump that allowed all of the shavings to be sucked out of the reactor pool and trapped in the filter housing. The reaming bit was powered at the pool's surface by a pneumatic drill, which was chosen in order to reduce the potential of electrocution. When attempting to drill out the flux-probe holes in a test plate, two significant problems were found. The first problem was that, due to the length of the extension, it was difficult to keep the bit vertically straight resulting in over boring the holes. The other problem was that the reaming bit was a different shape than what was used in the calculations. Instead of being a full tapered bit, it was only a half taper resulting in the top surface of the holes to have far more material removed than the bottom surface. Because of these problems, the drilling attempts were abandoned for a later revisit.

Based on the experiences from the first attempt to restore the holes, it was determined that a tool needed to be designed that would make sure the bit stayed vertical and did not over-bore the holes. To do this, the long drill bit idea was abandoned and replaced with a pneumatic self-feeding drill that would rest directly on the upper grid plate. The drill, an Ingersoll Rand ARO Par-A-Matic 8255-A8-3 (see Figure 4.13), not only has an air-powered spindle, but the spindle is connected to an air-powered piston that allows the drill to provide its own downward force when drilling. In order to align the drill to each flux-probe hole, a special platform had to be designed and built. Because of limited space, it was determined that the platform would have to be enlarged to incorporate multiple flux-



(a) From Engineering Drawings (b) From Photo Analysis

Figure 4.14: Upper grid plate design based on (a) engineering drawings and (b) photo analysis of the actual core.

probe holes instead of being adjustable to account for each flux-probe hole's difference in geometry. Due to symmetry, a platform covering only one-half of the core would be necessary in order to restore all of the flux-probe holes. By visually comparing the upper grid plate's engineering drawing to the actual grid plate, one could quickly determine that at least two of the flux-probe hole locations were different (Figure 4.14). Because of this difference, it was felt necessary that the exact location of each flux-probe hole be known, especially due to the new drill's platform design.

It was determined that the easiest and most accurate means of determining the actual flux-probe hole locations was to examine high resolution photographs of the core. The center of each fuel element and flux-probe hole could then be extracted from the photos. A matrix transformation would be found to relate the photo's fuel locations to the drawing coordinates. This matrix transformation, which would account for any perspective distortions, would then be applied to the photo's flux-probe hole locations to give their true drawing coordinates. The trick was to find the fuel and flux-probe hole centers in the photo,

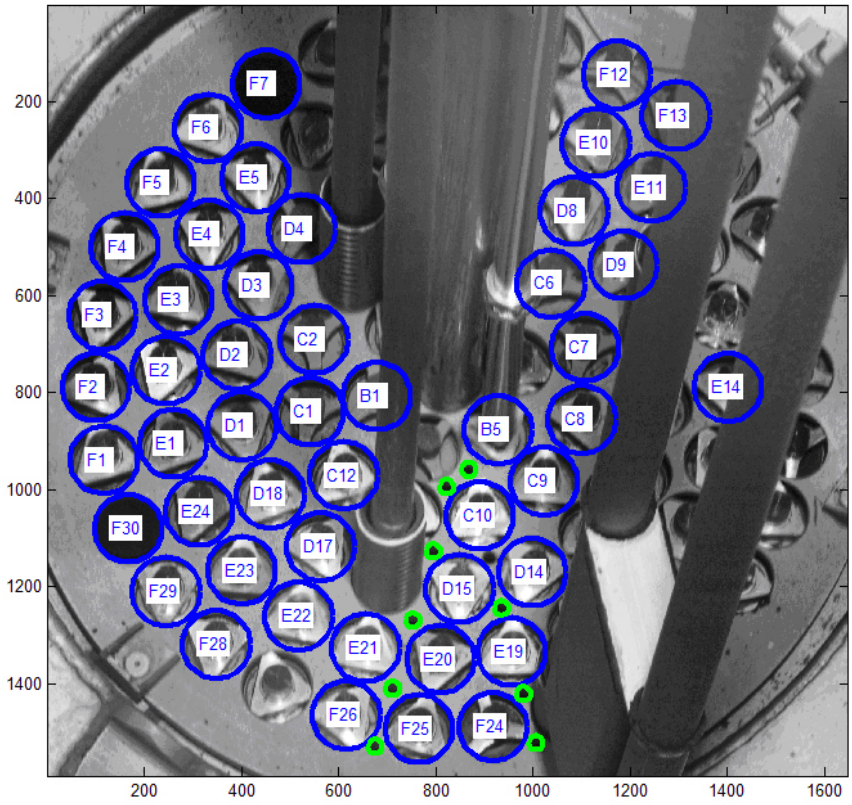


Figure 4.15: Photo of the reactor with identified fuel (blue) and flux-probe hole (green) locations indicated. Axes are in pixels.

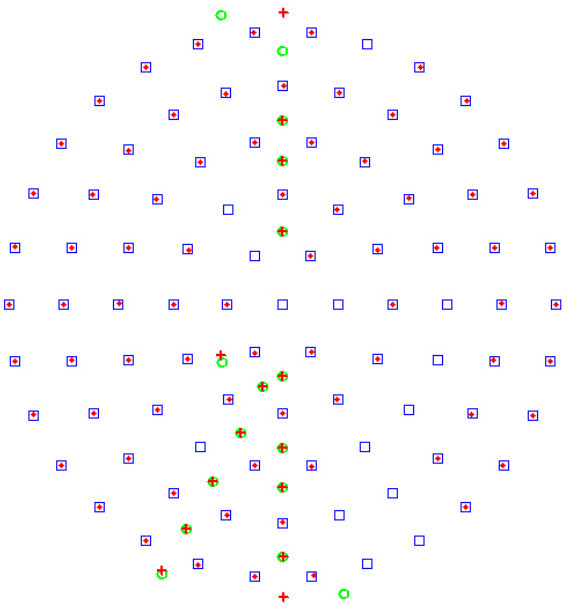


Figure 4.16: Compiled analysis results showing the difference in flux-probe hole locations between the engineering drawings and the analyzed photographs. Locations of fuel and flux-probe holes based on the engineering drawings are indicated with blue squares and green circles, respectively, while the analyzed photo results are shown with red dots and red crosses, respectively.

Table 4.1: Comparison of flux probe hole locations based on engineering drawings and the photo analysis results.

Flux Probe Hole Name	Engineering Drawings		Photo Analysis	
	X	Y	X	Y
A	0.000	8.481	0.000	8.375
B	0.000	7.242	0.000	7.240
C	0.000	5.259	0.000	5.255
D	0.000	4.119	0.000	4.125
E	0.000	2.073	0.000	2.068
F	0.000	-2.073	0.000	-2.068
G	0.000	-4.119	0.000	-4.125
H	0.000	-5.259	0.000	-5.255
I	0.000	-7.242	0.000	-7.240
J	0.000	-8.481	0.000	-8.375
K	-0.582	-2.359	-0.569	-2.367
L	-1.208	-3.690	-1.192	-3.688
M	-2.000	-5.082	-1.988	-5.080
N	-2.775	-6.443	-2.765	-6.441
O	-3.452	-7.744	-3.451	-7.633
P	-1.750	-1.656	-1.778	-1.483

which was accomplished using a MATLAB program published by Tao Peng that utilized a circular Hough transform [26]. Only minor modifications were made to the transform code in order to improve memory efficiency. Additional code was added to provide a user interface that would allow the user to identify each circular object found by the transform as either a specific fuel element, a specific flux-probe hole, or as an invalid object. A completed photo analysis using the user interface is shown in Figure 4.15 along with the compiled results from seven different photos shown in Figure 4.16 and listed in Table 4.1. A

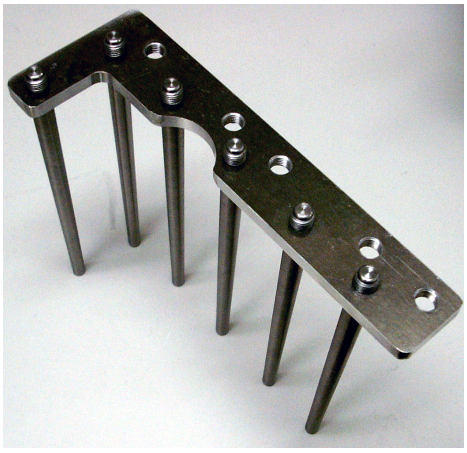


Figure 4.17: Flux-probe hole location testing tool.

flux-probe hole testing tool (Figure 4.17) was built using the analysis results to verify the findings. The tool consisted of 5 to 6 tapered pins connected to a stainless steel cross arm, fixing their locations, followed by 16' of 1/4" pipe (not shown). The pipe allowed the tool to be guided into the flux-probe holes and gave the user feedback as to whether or not the tool was seating securely. The tool indicated that the positions were well known and showed that the flux-probe holes were approximately 7 mm in diameter, based on marks left on the tool by the upper grid plate.

With the actual flux-probe hole locations determined, the drill's platform was then designed. The platform not only had to align the drill to each flux-probe hole while keeping the drill vertical, it also needed to anchor itself to the upper grid plate in order to counter the downward pushing force of the drill. To do the alignment and anchoring of the platform, three alignment posts with built-in pneumatic clamps were de-

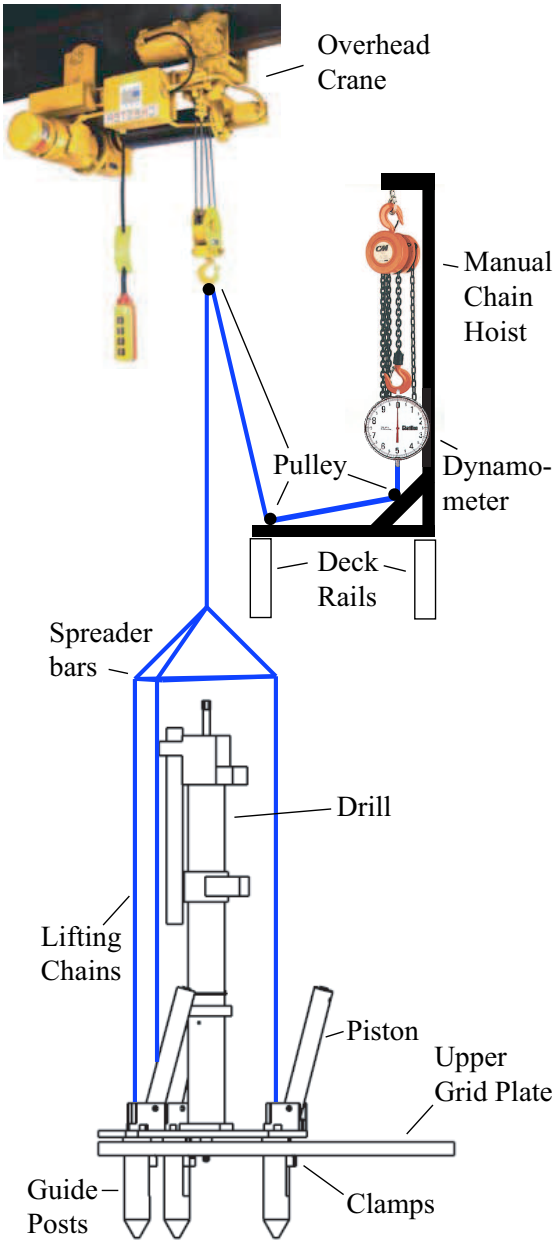


Figure 4.18: Drilling rig setup depicting the drill, its platform, the overhead crane, the manual chain hoist with the dynamometer, and the chain/cable assembly.

signed into the platform. These posts would insert into specific fuel holes in the upper grid plate, and the clamps would come out of the side of the posts to hold against the bottom of the upper grid plate. The post assemblies also included eye bolts which attached to three lifting chains. The chains were kept vertical by using three 8" all-thread bolts above the top of the drill that acted as spreader bars. This kept the center of mass for all drill locations between the three eye bolt lifting points (based on conservative Pro/E calculations assuming a solid steel drill). The overhead crane and a manual chain hoist with an in-line dynamometer were used to lift the drilling rig because the combined weight of the drill and platform was estimated at nearly 40 lbs. The setup of this assembly is depicted in Figure 4.18. Engineering drawings for the drilling rig parts may be found in Section §C.6.

The drill and clamps were controlled via the three valves shown in Figure 4.19. The clamps were either engaged or retracted respectively by the On/Off valve. In addition, the On/Off valve also allowed air to be supplied to the drill only when in the On position. The 3-way valve controlled whether the drill was moving forward (down) or in reverse (up). The final valve was used as a safety and required the operator to hold its lever down in order to operate the drill. This valve could also be used to regulate the speed of the drill. The control valves, alone, required seven air hoses, but since the drill was designed to operate in an air environment, an additional line was required to help capture some of the major air leaks. Capturing these leaks was important since the air supplied to the drill contained oil and since a large flow of bubbles in the water could create virtual beam-port streaming paths through the water.

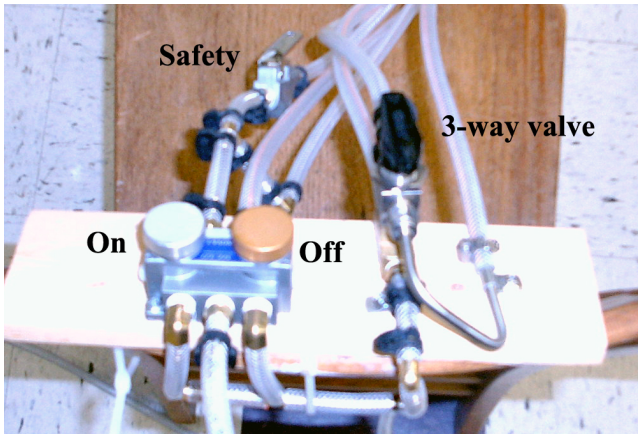
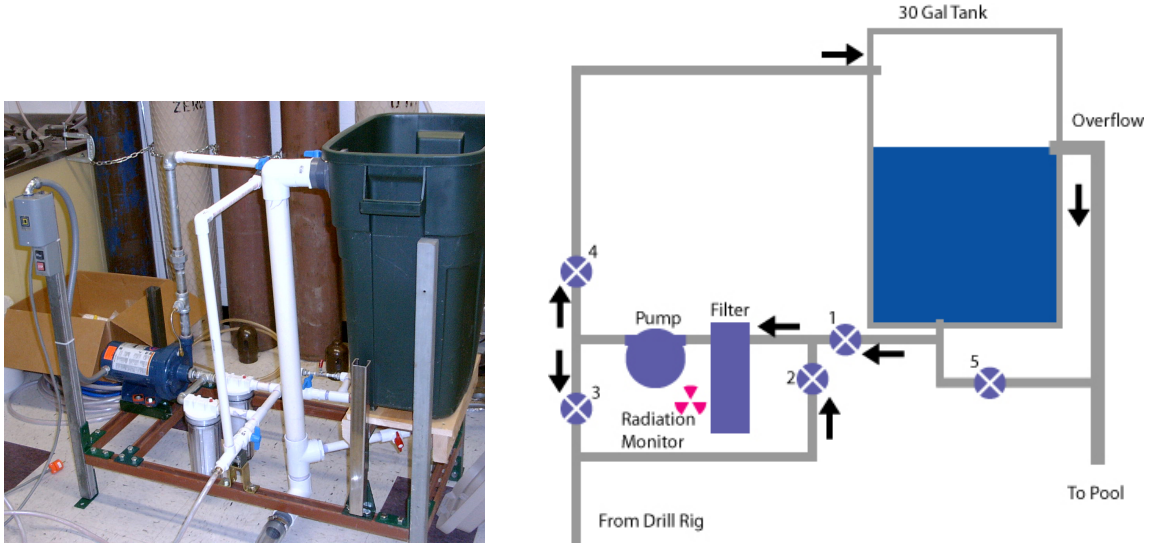


Figure 4.19: Control valves used to operate the flux-probe hole restoration drilling rig.



(a) Pump Sled (b) Conceptual Drawing

Figure 4.20: The (a) actual and the (b) conceptual drawing of the pump sled used with the drilling rig to remove and trap any metal shavings or radioactive material.

There was one more hose contained in the air line bundle, however this hose was used for removing the metal shavings created by the drill. This hose was attached to a pump sled (Figure 4.20), which contained filters for trapping the metal shavings and any radioactive material sucked up by the pump. The pump sled also included a water storage tank, which, through the use of valves, could be used to prime the suction tube and pump system prior to drilling. The pump sled was so successful that it was actually used to clean the tank and filter the primary water once the drilling was completed.

4.3.2 Reactor Upgrade

Restoring the flux-probe holes in the upper grid plate using the drilling rig required the removal of nearly half of the reactor core's fuel rods along with the shim and regulating control rod guide tubes. The timing of the restoration project coincided with the expected approval of the Kansas State University TRIGA Mark-II Nuclear Reactor's new license, which contained requests for an additional control rod and for an increase in the maximum

Figure 4.21: The drilling rig during setup showing the drill rig (drill removed), the manual chain hoist and dynamometer, the pump sled, and the controls block.

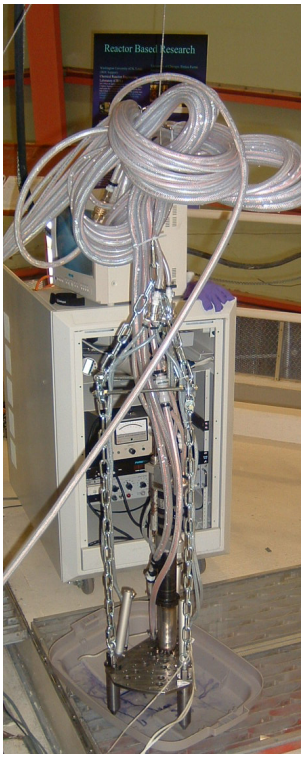


Figure 4.22: The drilling rig with the drill mounted to the platform.

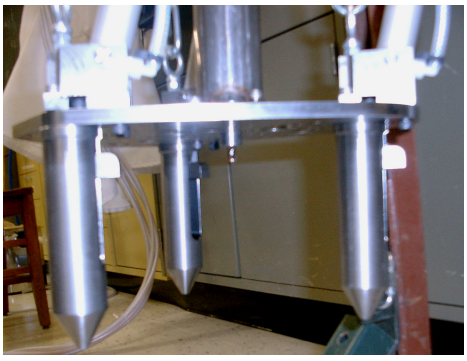


Figure 4.23: Bottom side of the drilling rig showing the clamps engaged and the drill bit with the drill in its full down position.

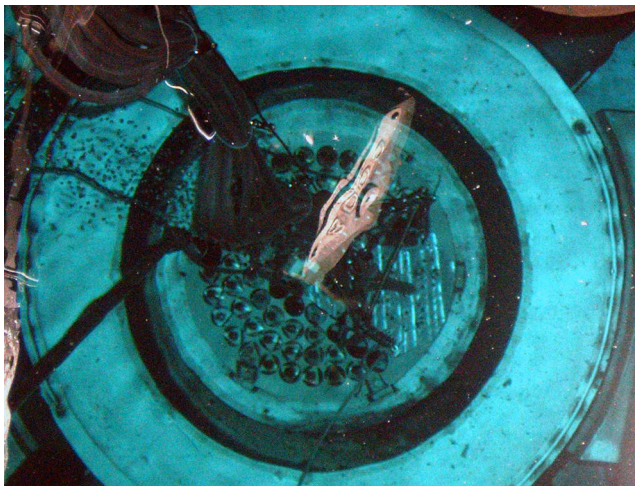


Figure 4.24: Difficult to see due to the reflection of overhead lights, but this is a picture of the drilling rig attached to the reactor core and in the process of restoring the flux-probe holes on the right side of the core.

power limit. The existing license restricted the reactor from increasing the maximum power level but did not forbid the installation of the fourth control rod, so long as the reactivity limits of the reactor were maintained. In addition to the control rod installation, problems with the rotary specimen rack (RSR) irradiation facility required removal of the RSR from around the reactor core in order to try to repair the problem. Because the installation of the new control rod, known as the safety rod, and the RSR repair required many of the same procedures as the flux-probe hole restoration, it was decided to perform these operations together.

The installation of the safety control rod was dependent on the redesign of the deck plate to which the control rod drives are mounted. The design of the new deck plate was supposed to be included with the purchase of the new control rod and drive, but it was discovered that the design was not compatible with the control rod and drive that was ordered and received. When confronted with the problem, General Atomics did not accept responsibility for the problem and refused to correct the deck plate design. In addition, their design placed the transient rod, commonly called the pulse rod, in the "D" ring, which would significantly reduce the rod's reactivity worth. Instead, the reactor manager requested that the pulse rod be positioned in the "C" ring, giving the reactor the capability of reaching pulses of up to \$3 of reactivity worth.

It took several months to build an accurate model of the reactor in Pro/Engineer from the limited schematic data and by scouring the reactor's safety analysis report and historical documents. From this model, a new design for the deck plate was developed with the only viable solution requiring the regulating control rod drive to be located below the other three control rod drives. The full design of the deck plate is not detailed here because this was not the goal of the project, but an assembly drawing is provided in Section §C.7 for reference. Operation of the fourth control rod also required modification to the reactor's control panel in order to activate the rod's control circuitry and position indication.

While the control rod drives are anchored to the deck plate, the actual control rods slide

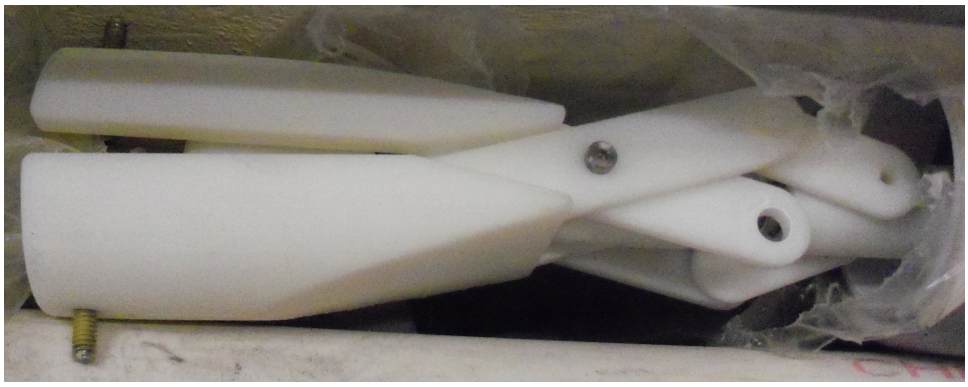


Figure 4.25: The scissor-type clamping tool built for removal and installation of the control rod guide tubes inside the reactor core.

in and out of the reactor core inside control rod guide tubes. These tubes are held in place by the upper grid plate and are anchored into the lower grid plate by a screw that extends a wire below the lower grid plate to keep the control rod from pulling the guide tube out of the lower grid plate. The screw requires an Allen head wrench but unfortunately, there is nothing to keep the guide tube from rotating while trying to withdraw the screw and wire. Because of this, a custom tool also had to be built that would clamp onto the guide tube to prevent the tube from rotating while also not deforming the tube. A scissor-type clamp structure with a hollow tube to accommodate the Allen head wrench was designed to accomplish this requirement. Drawings of the clamp parts are provided in Section §C.8 with a picture of the tool shown in Figure 4.25. Due to the complexity of the hook part of the scissor clamp, the parts were manufactured using the Mechanical and Nuclear Engineering Machine Shop's rapid prototyping tool, which allows the parts to be "printed" using the Pro/Engineer part files without the need for fully detailing the drawings.

Planning for the upgrade and repairs took several months, but the reactor was back online within about three weeks after starting the planned work. Overall, this sidestep of the MPFD^3-T project proved beneficial because it not only restored the upper grid plate flux-probe holes so the MPFD^3-T strings could be installed, but it also upgraded the reactor's controls and instrumentation for the upcoming license changes. This also prepared the

reactor so no modifications to the reactor would be necessary once the MPFD^3-Ts were installed, which would make such modifications very difficult and with high probability would have damaged the MPFD^3-T array.

4.3.3 Detector Calibration

The unique characteristic of low detection efficiency, which makes the MPFD design successful as an in-core neutron detector, also make the MPFD^3-Ts extremely difficult to calibrate. Even with a neutron flux of 1×10^6 n cm^{-2} s^{-1} as found in the beam ports, the count rate for a 1 μm thick natural uranium-coated MPFD^3-T would only be 1.8×10^{-2} counts per second. This rate means that only one count would be registered every 56 seconds on average. For a spectrum with 10,000 counts, it would require more than 154 hours, nearly one week, of continuous reactor operation. This amount of time is impractical and leaves in-core operation as the only possible option for calibration. To do the in-core calibration, each detector would have to be placed in the exact same location inside the reactor with the control rods in the same general configuration and the reactor power at the same set points. Only a relative calibration could be obtained because the actual flux could not be known, even though estimates could be obtained through gold foil or iron wire experiments, or through the use of a calibrated detector such as an self-powered neutron detector (SPND).

While a procedure was prepared to perform this task, the electronics were not yet ready when the MPFD^3-T strings had to be installed in the reactor core, so the calibration was not attempted. Because the detector probes have been installed in the reactor core since December 2008, it would not be advisable to remove the detectors from their in-core positions in order to perform the calibration procedure. The aluminum tubing of the probes is very thin and any forced removal of the tubes from the reactor may result in puncturing the tube wall and subsequent flooding of the detectors.

4.3.4 System Installation

After all of the flux-probe tubes had been loaded with a MPFD^3-T string and sealed, the stainless steel tubing had been bent, and the reactor modifications had been completed, the detectors were ready for in-core installation. The first step was to attach the flux-probe tubes to the stainless steel tubing. A shielded 30 conductor 24 (7/30) AWG cable had already been loaded into each stainless steel tube. A mating connector board (Figure 4.9) was soldered to the flux-probe end of the cable and a 37-pin male D-Sub connector was attached to the other end. With the connector boards mated, they were slid into the end of the stainless steel tube while simultaneously being encapsulated and packed with the Devcon Flow-Mix 2 Ton Epoxy mentioned in Section §4.2.3. The Swagelok compression fitting was then secured to both tubes. The combination of the Swagelok compression fitting and the Devcon epoxy ensured that no water would leak in through the connection of

Figure 4.26: Installation of a MPFD^3-T string into the reactor core. The top of the string is attached to the junction box located at the top of the reactor tank.

Figure 4.27: The MPFD^3-T 3D System located on the 12' level of the reactor pedestal, showing the blue conduit used to route the wires from the detectors down to the electronics.

the two tubes. In addition, the epoxy encapsulating the connector boards and bare wire ends prevents any condensation inside the stainless steel tubing from shorting out the connector boards while also providing high voltage isolation.

Once the epoxy had been allowed to cure, the tubes were installed in the reactor core. The top of the strings were attached to the junction box, located at the top of the reactor tank, by a compression fitting mounted to the bottom of the box. The compression fitting makes its seal to the stainless steel tubing by squeezing a rubber collar around the tube, providing a water-tight and electrically-isolated seal. The junction box is mounted to the top of the reactor using bolts with plastic bushings with a rubber pad in between the box and the top of the tank (Figure 4.26). This setup also provides electrical isolation between the box and the reactor tank, which is critical for electronic noise reduction as will be discussed

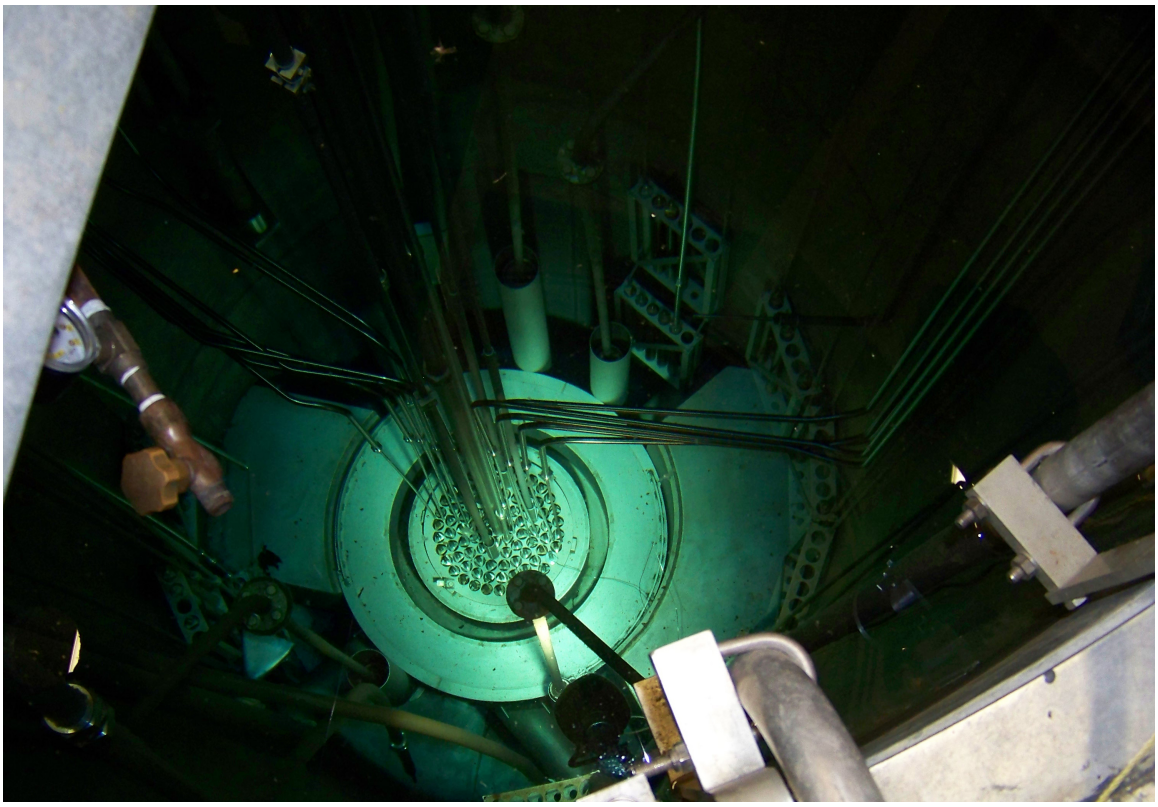


Figure 4.28: The MPFD^3-T array's routing of the stainless steel tubes inside the reactor tank and into the reactor core.

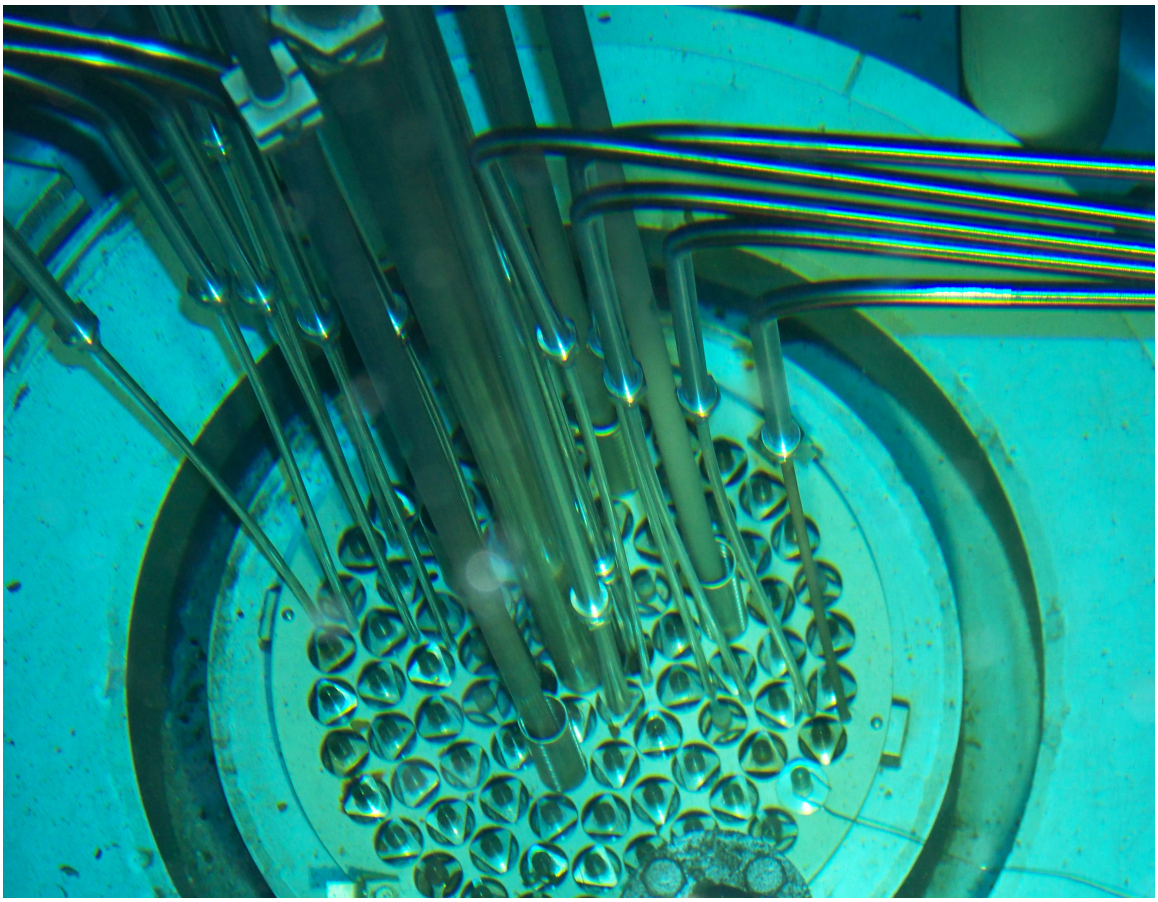


Figure 4.29: A zoomed in view of the MPFD^3-T array installed in the reactor core.

in Section §5.3.4. Attached to the lid of the junction box is an insulated and shielded 2" flexible conduit that routes, protects, and shields five cables across the top side of the reactor deck (22' level) and down the side of the reactor tank pedestal to the MPFD^3-T 3D System located on the 12' level (Figure 4.27).

Figure 4.28 shows how the stainless steel tubes rest against the side of the reactor tank wall from the junction box until the tubes are approximately 6' above the top of the reactor core. The tubes are then bent 90° in order to align vertically with the flux-probe holes where they are again bent at 90°. The Swagelok compression fittings can be seen coupling the stainless steel tubes to the flux-probe tubes in Figure 4.29 prior to the flux-probe tubes entering the upper grid plate of the reactor core. The design minimizes any neutron stream-

ing through the tubes while maintaining sufficient clearance for the removal of the fuel rods for required inspections.

The MPFD^3-T 3D System's electronics and data collection computer are located on the 12' level of the reactor's pedestal in order to minimize the distance to the detectors while not taking up valuable real estate on either the 22' level or the 0' level. The overhang of the 22' level and the System's placement beyond the bulk shield tank also helps protect the MPFD^3-T 3D System's electronics from activities in and around the reactor bay. The overhang also helps protect the System from weather, as seen when the System survived the 2008 tornado that took the roof off of the reactor bay.

Chapter 5

Electronics

A significant portion of this project includes data acquisition from the detector array. While there are many data collection options available for a single detector, collecting data simultaneously from a large number of detectors is not available "off the shelf". To accomplish the goals of this project, it was required that a custom electronics solution be designed and built, which is described in this chapter. Simulations were performed using DesignSoft, Inc. TINA version 9+ PSPICE based simulation software. Schematics and layouts were made using EAGLE (Easily Applicable Graphical Layout Editor) Professional Edition version 4.16.

5.1 Daughter Boards

The daughter board connects to a detector string and collects the pulse mode, MSV mode, current mode, and temperature data. This data is then packaged and forwarded to the mother board.

5.1.1 Pulse Mode Electronics

The first item which must be resolved when designing the preamplifier, or preamp, is to determine the bandwidth requirements based on the radiation signal rise and fall times. These times were found in Sections §3.6 and §3.7 with their range estimates listed in Tables 3.4 and 3.5. The bandwidth can then be determined by estimating the signal's associated frequency (f) through the relation

$$t_r = \frac{\ln(0.9) - \ln(0.1)}{2\pi f} \Rightarrow f = \frac{0.35}{t_r} \tag{5.1}$$

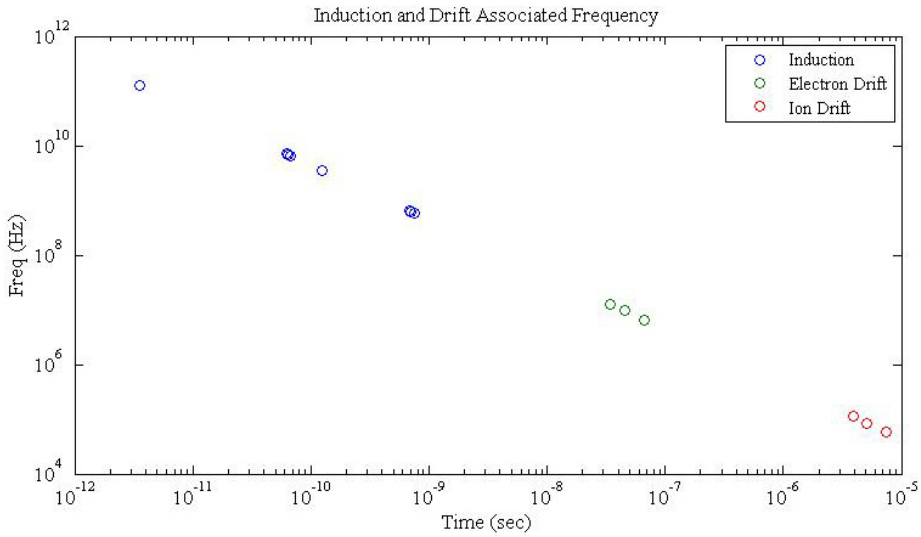


Figure 5.1: Induction and drift times to their associated frequencies.

where t_r is the 10-90% rise time. Because the signals can be estimated as a linear increase or decrease, t_r is simply 80% of the injection and drift times. See Figure 5.1 for the range of frequencies which may be produced by the different detector configurations.

The induction of the charge into the detector has an associated frequency range of 0.58 to 124 GHz, which is outside the bandwidth capabilities of nearly all amplifiers which could be used in a preamp design. Likewise, the associated frequency range of the electron induced signal is between 6.6 and 12.9 MHz, which is near the upper end of amplifier capabilities. The ion induced signal, which is the pulse mode limiting component of the signal, has an associated frequency range of 58 to 113 kHz. Therefore, the preamp design must support signals greater than 58 kHz.

The locations of the detectors and the electronics adds additional considerations which must be taken into account for the preamp design. A generally understood rule for radiation detector electronics is that the preamp should be located as close to the detector as possible, usually within a few inches at most. In the present case, the detectors are located inside a nuclear reactor under 16+ feet of water. Not only does the water prove to be a challenge for sealing the electronics, but also the reactor poses a serious danger to the life

of the electronics due to radiation damage. Due to the number of detectors and thus the volume of their associated electronics, the only practical location for the electronics, as noted previously, is nearly 40 feet from the detectors. This long cable not only induces significant line capacitance but also becomes a large antenna that picks up noise from the control rod drives, cooling pumps, and the electro-magnetic interference (EMI) generated by the reactor itself. Therefore, it is critical to use a differential preamp design for the highest common mode noise rejection instead of the more common single-ended preamp design. An instrumentation amplifier is designed specifically for differential signals with a high common mode rejection ratio (CMRR) and thus is a logical choice.

An additional constraint on the preamp design is that one of the signal lines for the differential input must be the high voltage line of the detector. Therefore, the preamp must be a balanced AC coupled design in order to isolate the preamp from the high voltage. This AC coupling is achieved by using a capacitor between the preamp and each signal line along with a resistor to ground for bleeding off any charge. This circuit forms a high pass filter that is useful for not only blocking the high voltage but also for noise reduction.

After consideration of additional factors such as availability, cost, additional external component requirements, bandwidth, gain, noise, and available simulation models, to name just a few, the Burr-Brown INA103KU instrumentation amplifier was selected as the heart of the preamp design. Using an AC coupling capacitor of 1 nF, the bleed-off resistor must be 2.7 kΩ for a high pass cutoff frequency of 58.9 kHz. Shown in Figure 5.2 is the circuit along with an AC source (high voltage 10 kΩ stand-off resistor in parallel) and a voltmeter for analyzing the circuit's output that is used to generate the bode plot shown in Figure 5.3. This bode plot shows that the gain is uniform within 3 dB from approximately 55.5 kHz up to 2.31 MHz with a 59.6 dB suppression at 60 Hz and 53.5 dB at 120 Hz, which is all critical for minimizing distortion of the desired signals while eliminating any line frequency noise.

The circuit used for simulation of the pulse response or transient response of the preamp is shown in Figure 5.4. In the circuit, R11 (10 kΩ) is the high voltage stand-off resis-

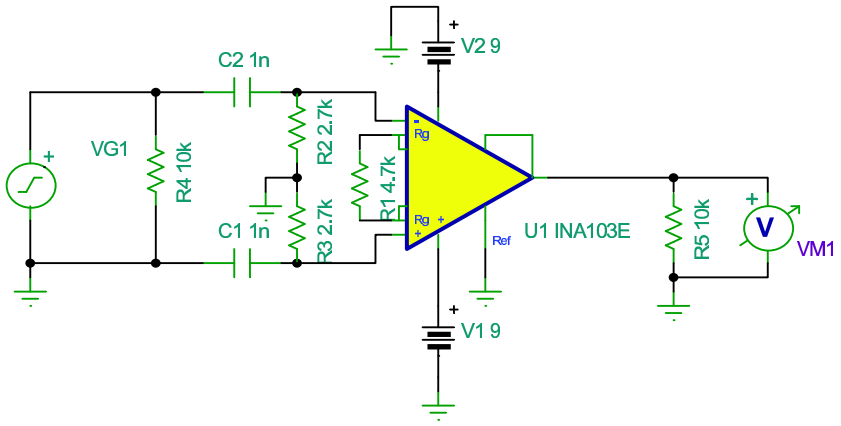


Figure 5.2: Preamp circuit configuration for AC simulation analysis.

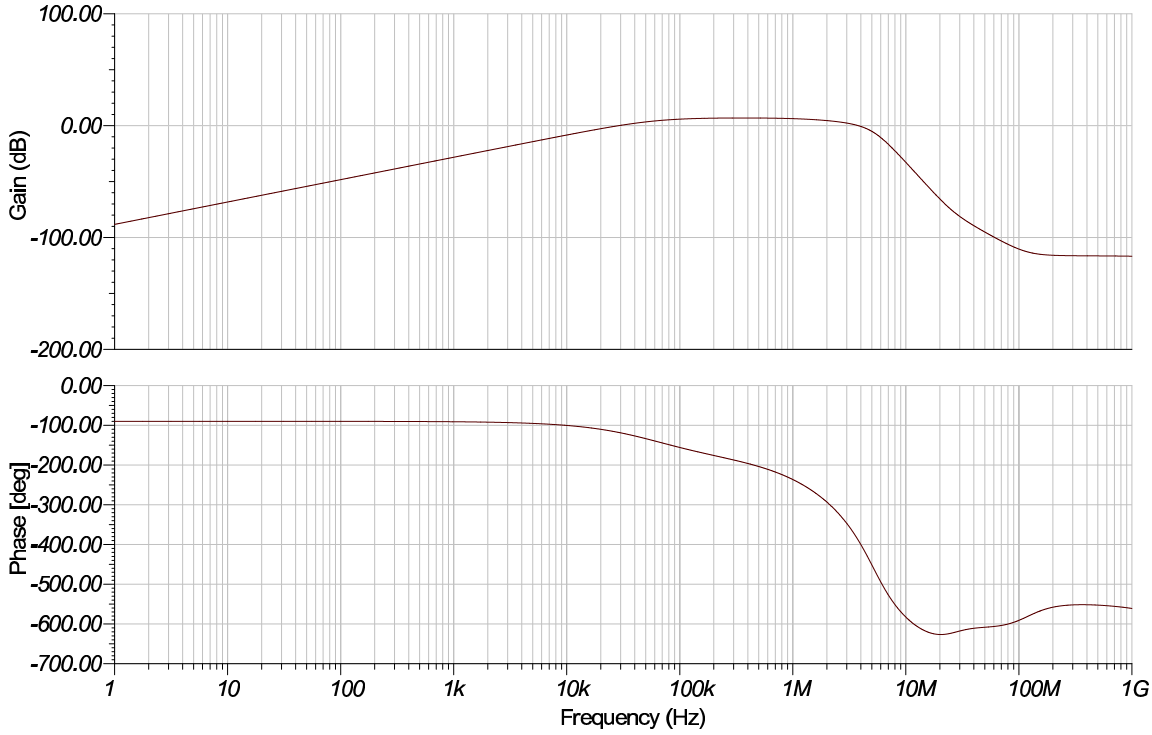


Figure 5.3: Bode plot of preamp AC simulation analysis.

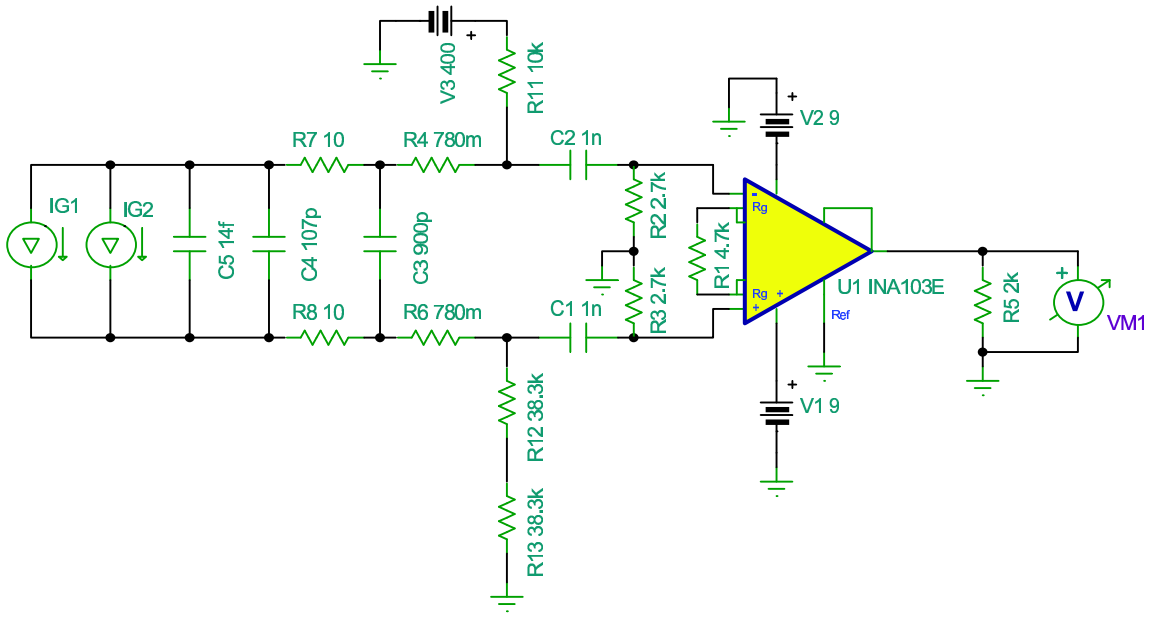


Figure 5.4: Preamp circuit configuration for transient simulation.

tor, while R12 and R13 (both 38.3 kΩ) form a voltage divider used in the current mode electronics (Section §5.1.3). The cable between the electronics and the junction board that sits approximately 6 feet above the reactor is simulated using the resistors R4 and R6 (both 0.78 Ω) along with the capacitor C3 (900 pF). Bare alumel wire sheathed in alumina ceramic insulators connects the junction board to the detectors and is represented by the resistors R7 and R8 (both 10 Ω) along with the capacitor C4 (107 pF). The detector itself has a capacitance of approximately 14 fF (C5) which can be calculated by the parallel-plate capacitor equation

$$C = \frac{k\varepsilon_0 A}{d} \tag{5.2}$$

where k is the relative permittivity of argon at 1 atmosphere or 1.000513, ε_0 is the permittivity of free space or 8.854×10^{-12} F m^{-1}, A is the contact area of the plate, and d is the distance between the plates. The MPFD^3-Ts have a contact diameter of 1 mm so A is $\pi(1\ mm/2)^2$ or 0.785 mm^2 while d is 0.5 mm.

The charge injection is simulated for the electrons and ions by the general current sources IG1 and IG2, respectively. The configuration window of these sources are shown

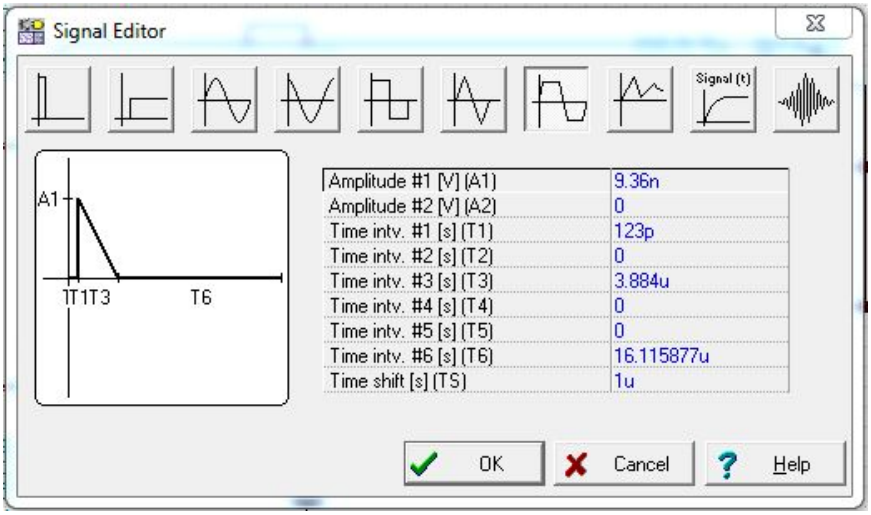


Figure 5.5: Settings of the general current source for transient simulation.

Table 5.1: Settings and results for preamp transient simulation.

Injection Type	Mean Time			Highest Energy		
Inj. Time (ps)	64.2			123		
Energy (MeV)	1.683			5.991		
High Voltage (V)	200	300	400	200	300	400
Elec Drift Time (ns)	66.1	45.8	34.0	66.1	45.8	34.0
Ion Drift Time (ns)	7479	5127	3884	7479	5127	3884
Max Elec Current (nA)	155	223	300	550	794	1070
Max Ion Current (nA)	1.37	1.99	2.63	4.86	7.09	9.36
General Current Source IG1 Settings (all others 0)						
Amplitude #1 [V] (A1)	155n	223n	300n	550n	794n	1.07u
Time intv. #1 [s] (T1)	64.2p	64.2p	64.2p	123p	123p	123p
Time intv. #3 [s] (T3)	66.1n	45.8n	34.0n	66.1n	45.8n	34.0n
Time intv. #6 [s] (T6)	20u - T1 - T3					
Time shift [s] (TS)	1u	1u	1u	1u	1u	1u
General Current Source IG2 Settings (all others 0)						
Amplitude #1 [V] (A1)	1.37n	1.99n	2.63n	4.86n	7.09n	9.36n
Time intv. #1 [s] (T1)	64.2p	64.2p	64.2p	123p	123p	123p
Time intv. #3 [s] (T3)	7.479u	5.127u	3.884u	7.479u	5.127u	3.884u
Time intv. #6 [s] (T6)	20u - T1 - T3					
Time shift [s] (TS)	1u	1u	1u	1u	1u	1u
Results						
Peak Height (μV)	10.58	10.84	11.18	37.57	38.63	39.88

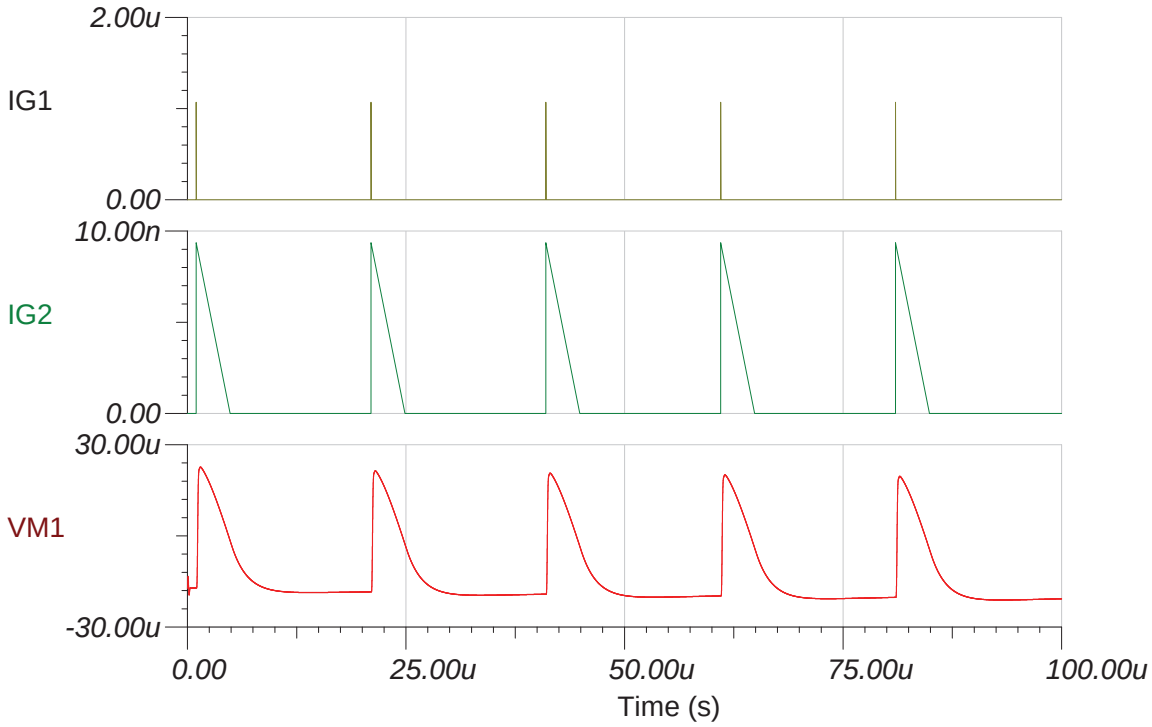


Figure 5.6: Plot of the transient simulation results for the highest energy pulse (5.991 MeV) with 400 V bias from the circuit in Figure 5.4. IG1 and IG2 are the electron and ion induced currents, respectively, while VM1 is the preamp's pulse output signal.

in Figure 5.5 with Table 5.1 summarizing the data from Sections §3.6 and §3.7. This table also provides the configuration parameters for the different pulse modes and provides results from the simulations. An example simulation output is shown in Figure 5.6.

Shown in Figure 5.7 is the schematic of a single channel, or one MPFD, of the daughter board with the pulse mode circuitry highlighted. IC503 and the parts to the left of it should be familiar from the simulation schematic in Figure 5.4. The detector connections are represented by DET_1H and DET_HV. Experimental tests of prototype pulse mode circuits showed that only minor gain was required and this is accomplished through the inverting gain operational amplifiers IC501A and IC501B. IC505C is a unity gain operational amplifier that provides a buffered output for connection to external nuclear instrumentation modules (NIM) and multi-channel analyzers (MCA). Likewise, IC505B provides a unity buffered signal to a Schmitt trigger formed around IC504 with a trigger level defined by

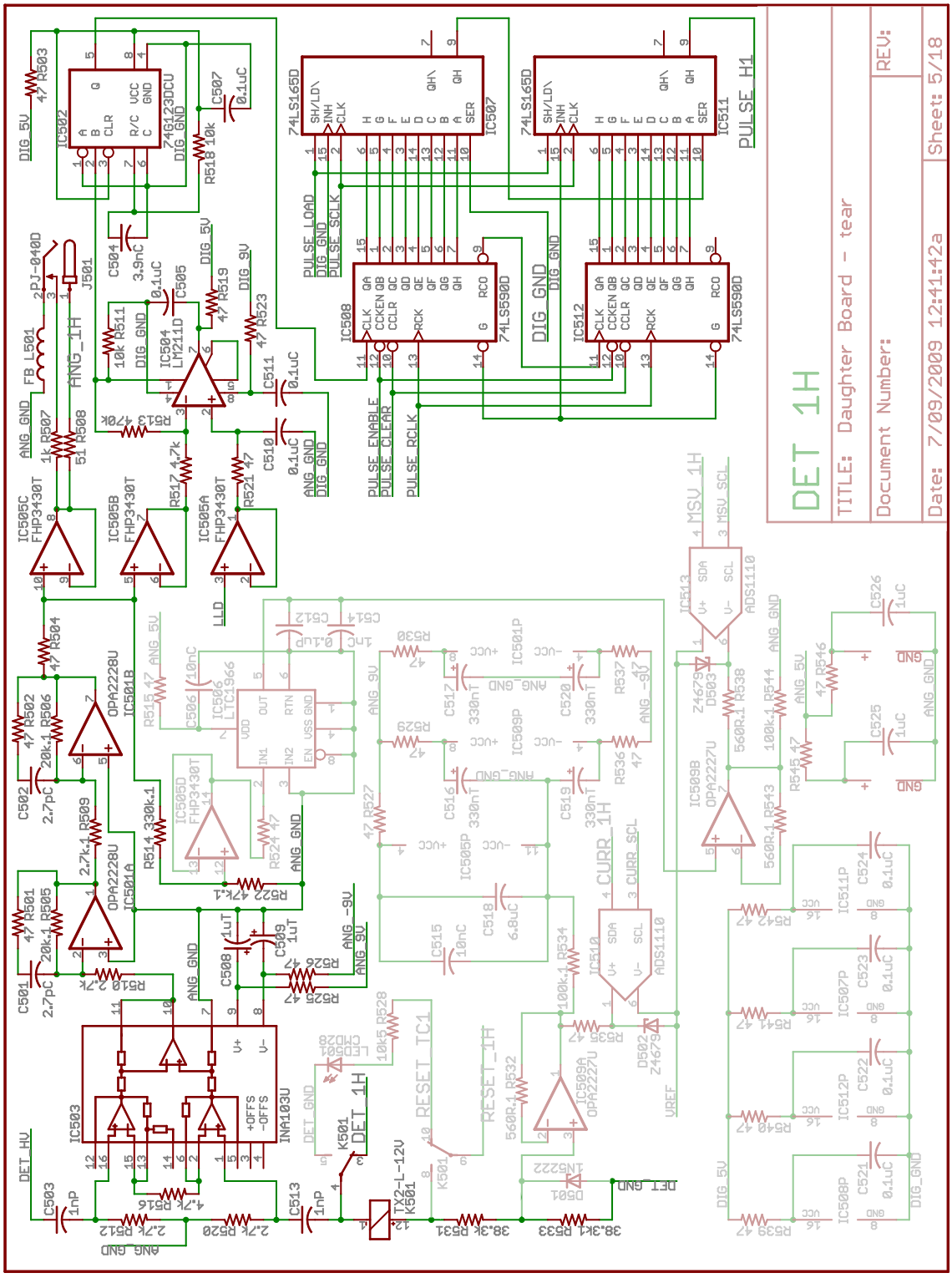


Figure 5.7: Schematic for pulse mode section of the daughter board. See Section §D.1 for full schematics.

a buffered lower level discriminator (LLD) signal from IC505A. IC502 is a retriggerable monostable multivibrator, which when it receives a pulse from the Schmitt trigger will hold the line for approximately 39 μs as defined by C504 $\times$ R518. If there is any jitter from the Schmitt trigger due to noise, IC502 will keep the line triggered in order to minimize duplicate counting of pulses and thus provides system noise reduction. IC508 in series with IC512 provides a 16-bit counter that allows up to 65535 pulses to be recorded between data collection cycles. IC507 and IC511 are parallel to serial shift registers that allow rapid data transfer from the counters to minimize dead time while also reducing 16 data lines into a single data line for querying by the microcontroller.

5.1.2 MSV Mode Electronics

The mean-squared voltage, or MSV, mode electronics (Figure 5.8) utilize the preamplifier and gain sections of the pulse mode electronics (Section §5.1.1). The output of the gain stage of IC501B is stepped down by the voltage divider of R514 and R522, which limits the buffered voltage input of the LTC1966 chip IC506. The LTC1966 has a differential voltage limit of 1 V but provides a true root mean squared (RMS) output. This output is buffered by IC509B for the analog-to-digital converter (ADC) chip IC513. The ADC is an ADS1110 that has a 16-bit conversion over an I^2C serial interface with a conversion rate between 15 and 240 samples per second. In addition to these traits, the ADS1110 was selected for its small SOT23-6 package while needing no additional components due to an onboard oscillator and self-calibration feature. For the large number of channels being built, this chip provides one of the lowest cost overall options. However, its inputs may only operate from -2.048 V to +2.048 V, giving a differential voltage range of 4.096 V, but this is reduced to 512 mV when the internal programmable gain amplifier (PGA) is increased from 1 to 8. In order to maximize the number of available bits while not limiting the capability of increasing the gain, an offset of 0.5 V (VREF) is applied to the positive input terminal. The zener diode D502 provides protection for the inputs of the ADC by

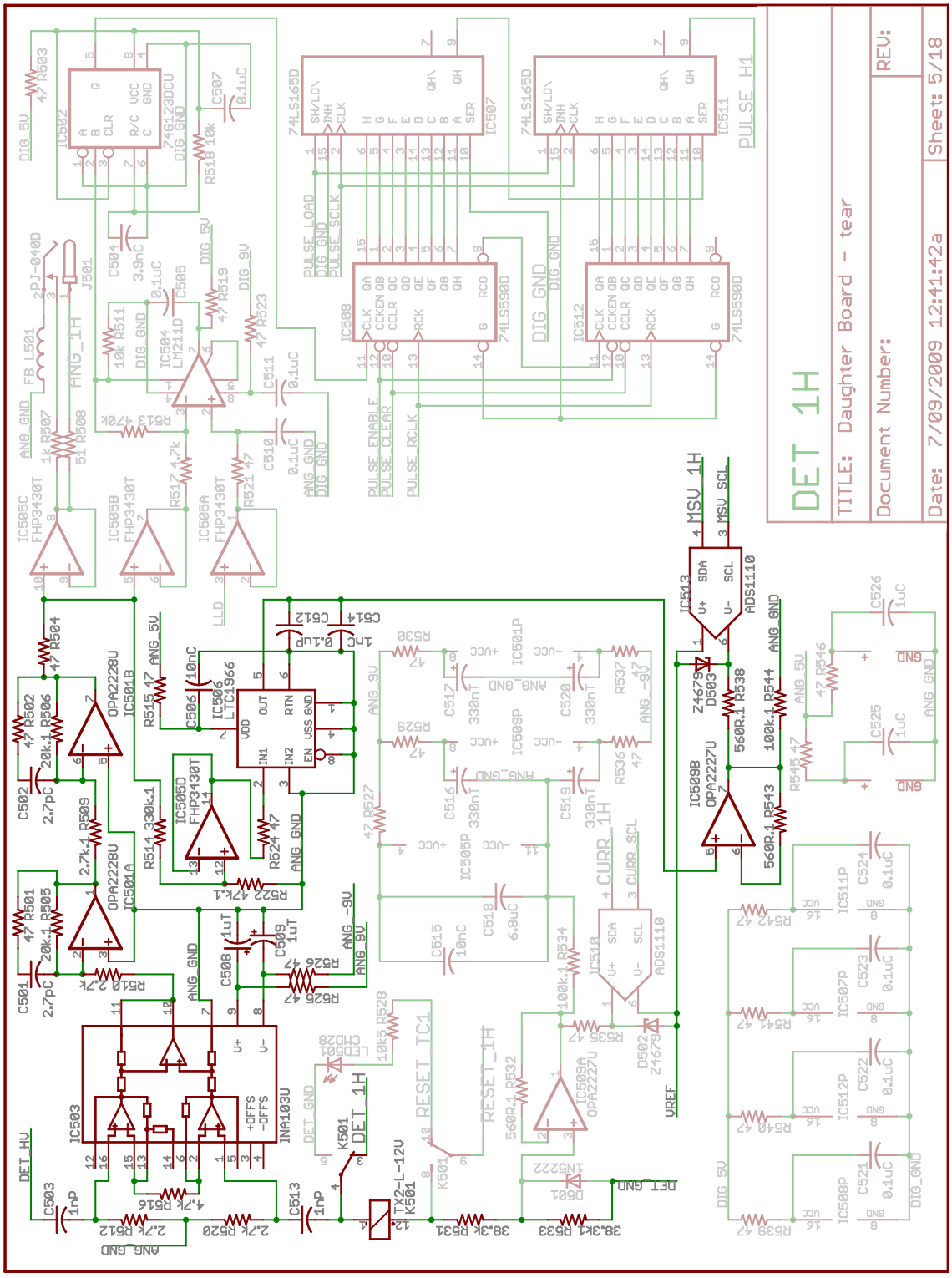


Figure 5.8: Schematic for MSV mode section of the daughter board. See Section §D.1 for full schematics.

limiting the voltage to a maximum 2 V beyond the VREF offset.

5.1.3 Current Mode Electronics

The current mode electronics (Figure 5.9) monitor the current flow generated by the detector through the induced voltage across a 38.3 kΩ stand-off resistor (R533) from the circuit ground. The voltage is limited by a parallel 2.5 V zener diode (D501), allowing a maximum current of 65 μA. This voltage is buffered for reading by the ADC IC510. The ADC is the same version of ADC used for the MSV mode electronics (Section §5.1.2).

5.1.4 High Voltage

The high voltage supply (Figure 5.10) for the detectors is generated by an EMCO F-series high voltage module (F05H) rated for $\pm$500 V at 20 mA. The output voltage is fully adjustable by setting the input voltage between 0 and 12 V. There are two problems when trying to use these modules. The first problem is that the thickness of the module is nearly the equivalent of the spacing between daughter boards. Therefore, the daughter boards must be designed to accommodate two different installation positions for the high voltage module. This allows the installed modules to be offset from each other between neighboring daughter boards in order to eliminate any risk of electrical shorting between boards. The second problem is that the module is not regulated, so a circuit must be built around the module in order to provide a stable controlled output voltage.

The voltage is set by an I^2C controlled digital-to-analog converter (DAC). This DAC (IC203A) is an LTC1669MS8 which has its own voltage reference and requires no additional components. The output of this DAC is connected to an ADG1613 analog switch (K202), which operates based on two cooperative safety systems. The first system is based on a series of C&K Components 11TL4A2 push-pull door switches (Figure 5.11), which open the circuit if one of the enclosure panels is opened. The switches do have the ability to be "pulled" into an on position to allow testing of the live circuit when the panel is open, but this must be done intentionally. The second safety circuit is a loop in the detector cable (see

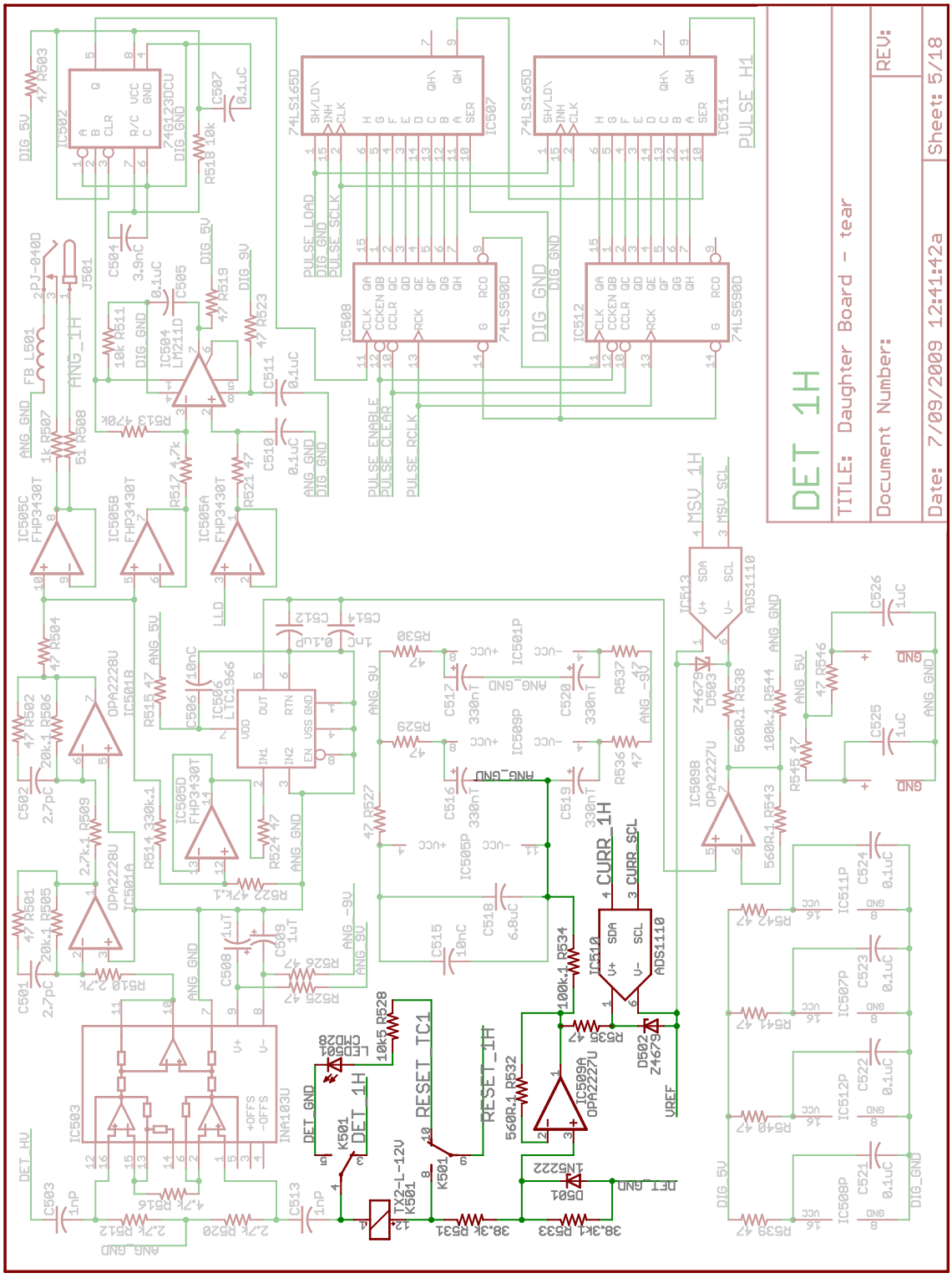


Figure 5.9: Schematic for current mode section of the daughter board. See Section §D.1 for full schematics. Also shows the relay circuit detailed in Section §5.1.5.

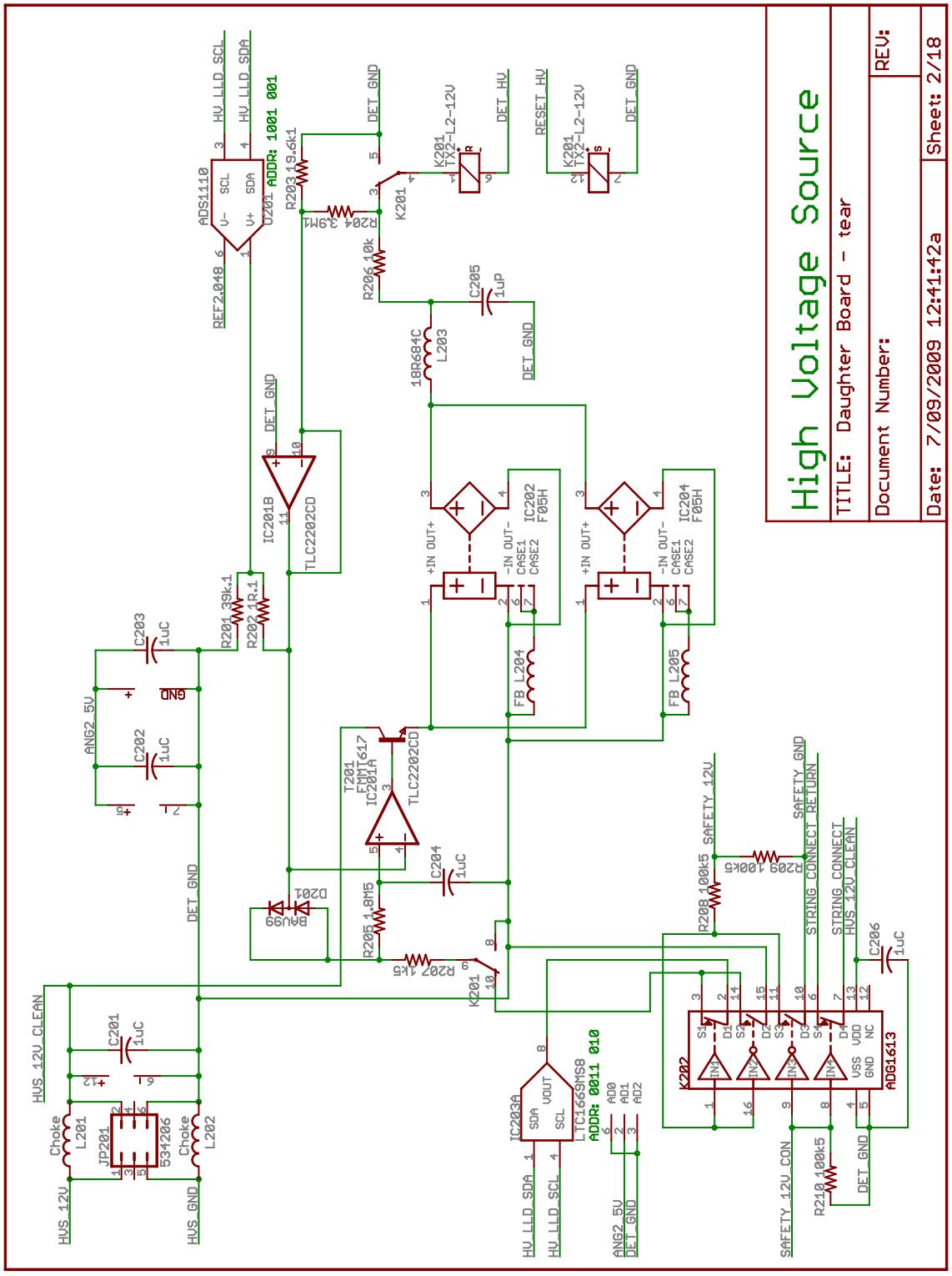


Figure 5.10: Schematic for high voltage section of the daughter board. See Section §D.1 for full schematics.

page 1 of the schematics in Section §D.1). This loop indicates to the circuit that a detector string is connected to the daughter board by raising the voltage on the SAFETY_12V_CON line. When disconnected, the resistor R210 sets the line to ground. If either safety circuit is interrupted, the K202 switch will remove the DAC voltage signal and replace it with a grounded signal line. This ground signal causes the high voltage to shutdown by decreasing back towards zero volts output.

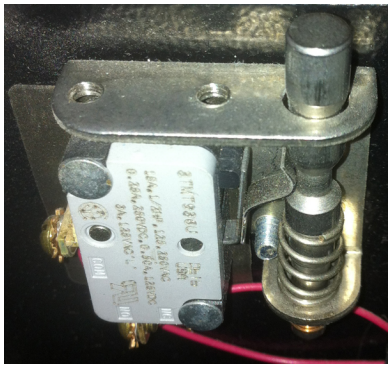


Figure 5.11: Safety switch, a C&K Components 11TL4A2 push-pull door switch, which opens a safety circuit whenever a panel of the enclosure is opened.

The operational amplifier IC201A compares the output voltage of K202 to the output voltage of the high voltage module through a voltage divider of R203 and R204. Another ADC, the same as that used for the MSV and current modes, monitors the same voltage divider and is used for recording purposes only.

5.1.5 Short Protection

Shorting of the high voltage is a real concern with this system. The detectors can easily short out due to the assembly challenge from spacing limitations of the wires, structural damage caused by the high radiation, or flooding of the detector tubes. In Figures 5.9 and 5.10, K501 and K201 are relays where their coils are in the detector circuit line. While the EMCO high voltage module is rated for 20 mA, K501 resides on the ground side of the detector circuit and only sees the current from a single detector. Therefore, a TX2-L-12V relay with a 2.9 mA coil current switch limit was selected. Unless multiple detectors are shorted, this relay should remove the shorted detector from the MPFD^3-T 3D System before overloading the high voltage module. When multiple detectors or the high voltage line is shorted, the relay K201 must react in order to prevent damage. This relay is a TX2-L2-12V

that has a coil switch current limit of 16.7 mA. A coil limit of less than 20 mA was selected since the current output of the high voltage module decreases linearly with decreases in high voltage. The regulating circuit will raise the high voltage in order to compensate for excessive current draw, thereby ensuring safe triggering of the relay. Both relays are latching type relays and can be switched back into the circuit by a signal from the on-board microcontroller. An LED next to each relay is lit during the reset cycle when the relay is successfully switched back into the detector circuit.

Likewise, with up to 20 mA at 500 V per daughter board and a total of 15 daughter boards able to fit into the MPFD^3-T 3D System, giving a total current capability of 300 mA, large enough to be considered lethal [27]. A first layer of user protection is provided by the door switches described in Section §5.1.4. A second passive system is added to each board

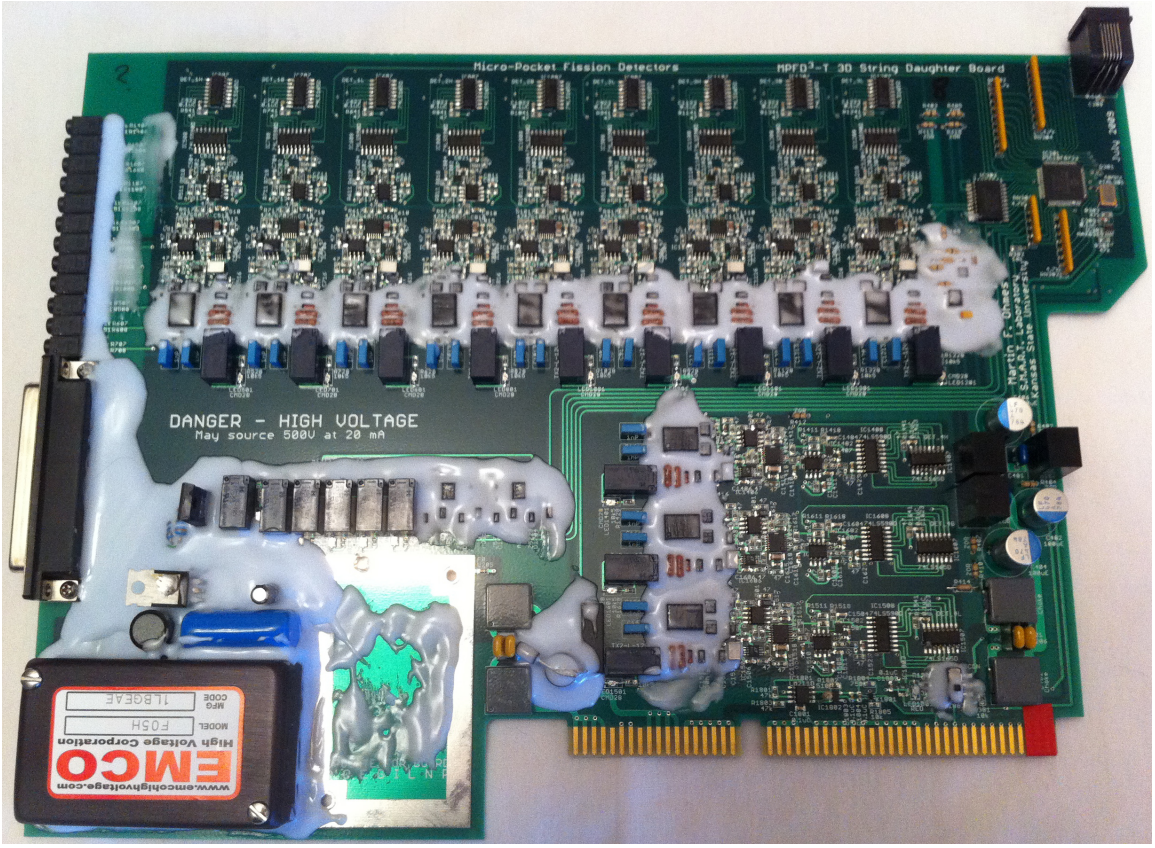


Figure 5.12: Front side of the daughter board with the protective RTV silicone coating.

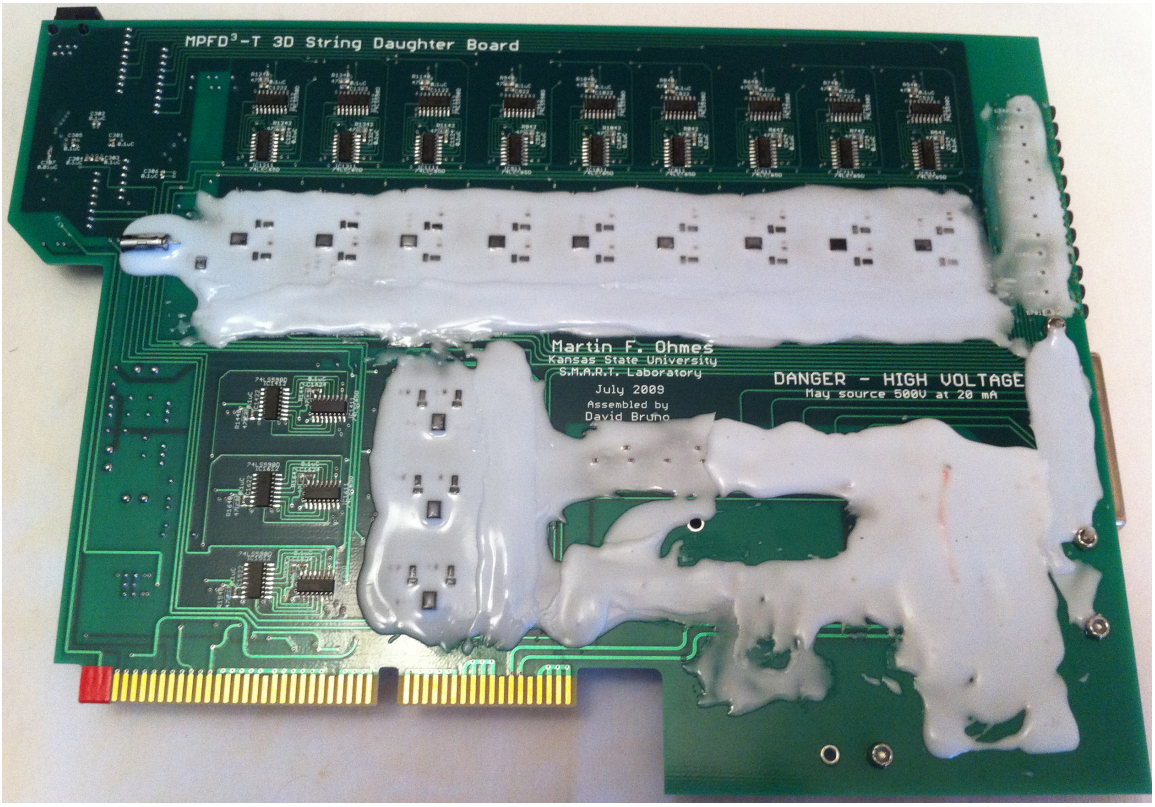


Figure 5.13: Back side of the daughter board with the protective RTV coating.

in order to isolate the user and to provide board level protection. A blue paste like coating spread over certain areas of the daughter board can be seen in Figures 5.12 and 5.13. This coating is an RTV (room-temperature vulcanization) silicone designed for encapsulation and potting in order to seal the surface from moisture, provide a dielectric barrier for high voltage isolation, and to protect sensitive electronics from environmental influences. While this thin coating is enough to protect the user from any high voltage present on the board, its primary purpose is to stabilize the analog circuits.

5.1.6 Thermocouple Electronics

Two short pieces of 0.010 inch diameter iron wire and constantan (copper-nickel) wire are built into the MPFD^3-T substrate to form a J type thermocouple. Usual thermocouple installations utilize special thermocouple extension wire in order to minimize noise and

eliminate as many junctions between dissimilar materials as possible. However, the design constraints do not allow for the usual best practices and thus offset voltages and noise are expected in the thermocouple measuring system. The voltage difference between the two legs of the thermocouple is amplified (IC1701 and IC1702) with a gain of approximately 29.24, yielding approximately 45 mV at 20°C, 170 mV at 100°C, and 331 mV at 200°C [28]. This amplified voltage is read using the ADS1110 ADC. Short protection is also provided using the same TX2-L-12V latching relays as discussed in Section §5.1.5.

5.1.7 Microcontroller

A PIC18F8722 microcontroller (IC301) ties all of the sections of the daughter board together and provides a link to the mother board. While the microcontroller on the motherboard (Section §5.2.1) operates at only 25 MHz, the daughter board microcontroller operates at 40 MHz to allow faster data transfer. The MSV and current mode electronics along with the thermocouple electronics all communicate with the microcontroller through their ADCs using the I^2C data interface while the pulse mode electronics transfer their data through sets of cascaded parallel to serial shift registers. In all, this is 28 I^2C data lines and 4 shift register data lines. Ideally, one would have hardware-based communication for each I^2C interface in order to free up the processor for other tasks. However, unaware of any existing device that could handle 28 I^2C interfaces all operating in parallel, a software implementation commonly known as "bit banging" was utilized. In this instance, a common clock line was supplied to each I^2C device, while a dedicated data line was given to each device. In this manner, all of the devices could be queried at the same time. A similar sort of approach was used to handle the shift register data. To communicate with the mother board microcontroller, a custom data interface was devised and will be explained in more detain in Section §5.2.1.

5.2 Mother Board

The mother board's purpose is to combine the data from each MPFD^3-T string and daughter board set, provide communication to the outside world, and distribute power to each daughter board.

5.2.1 Microcontroller

A PIC18F97J60 microcontroller (IC701) distributes and collects data from each daughter board while providing communication to the user through the panel mount LCD and LEDs along with the built in Ethernet. At the time of construction, the PIC18F97J60 was one of the newest 8-bit microcontrollers from Microchip Technology Inc. that had the unique capability of operating a built-in hardware based Ethernet port. Attached to the microcontroller via I^2C, are a 24FC1025 1024K bit serial EEPROM (IC703), a DS3231 real-time clock (U702), a TC74A5 (IC202) on board temperature sensor, and a DS2482 (IC602) I^2C to 1-wire bus converter to a DS2502 (IC603) which holds the MAC address for the Ethernet port. The front panel (Section §5.3.1) incorporates a liquid crystal display (LCD), reactor status inputs (Section §5.3.3), busy LED indicator (Section §5.2.2), and a standby button which all communicate with the microcontroller. The standby button is tied directly to the microcontroller to provide a "soft off" in order to allow the high voltage on the daughter boards to shut down gracefully.

While all of these peripherals provide important functions, the real purpose of the mother board is to combine the data from all of the daughter boards and send it out the Ethernet port. This is all done through the PIC18F97J60 microcontroller where three data lines transfer data from each daughter board to the mother board and a single fanned out data line transfers data from the mother board to all of the daughter boards. Sequencing of the data transfer is provided by two common data lines and a common trigger line.

5.2.2 Data Communication

Most of the data communication is a software process, especially for the communication between the mother board and the daughter boards. However, there are hardware communication methods that provide information to the user. One such indication is the busy LED located on the front panel of the MPFD^3-T 3D System. This LED utilizes the same trigger data line that instructs the daughter boards to gather a data set for transfer to the mother board. The trigger sets in motion a timed sequence through a 74HC4538D dual channel monostable multivibrator. The trigger turns on the busy LED through the first multivibrator (IC601A) and 32.9 ms later the LED is turned off by the second multivibrator (IC601B). This setup also restricts the busy LED from coming back on for 32.9 ms. The overall purpose of this LED is to show the user how quickly the MPFD^3-T 3D System is collecting data and is used as an indication that the MPFD^3-T 3D System has not locked up and quit working. Another user indication is the "Power On" and "Standby" switches which house an incorporated LED. This LED can be used to indicate the system power status via the microcontroller such as power on, system shutting down, system off, or programming mode. Other components found on the front panel are discussed in Section §5.3.1.

5.3 The MPFD^3-T 3D System

The MPFD^3-T 3D System includes the mother board, daughter boards, and the enclosure for all of these parts called the housing. The system also includes the detectors and their cabling along with the software which makes everything work. The following sections will focus on the housing and the parts attached to it, while the detectors and cabling have been or will be discussed in other sections.

5.3.1 Front Panel

The front panel of the MPFD^3-T 3D System shown in Figure 5.14 has indicators for the power state ("Power On" and "Standby"), busy state ("Busy"), building power state ("AC

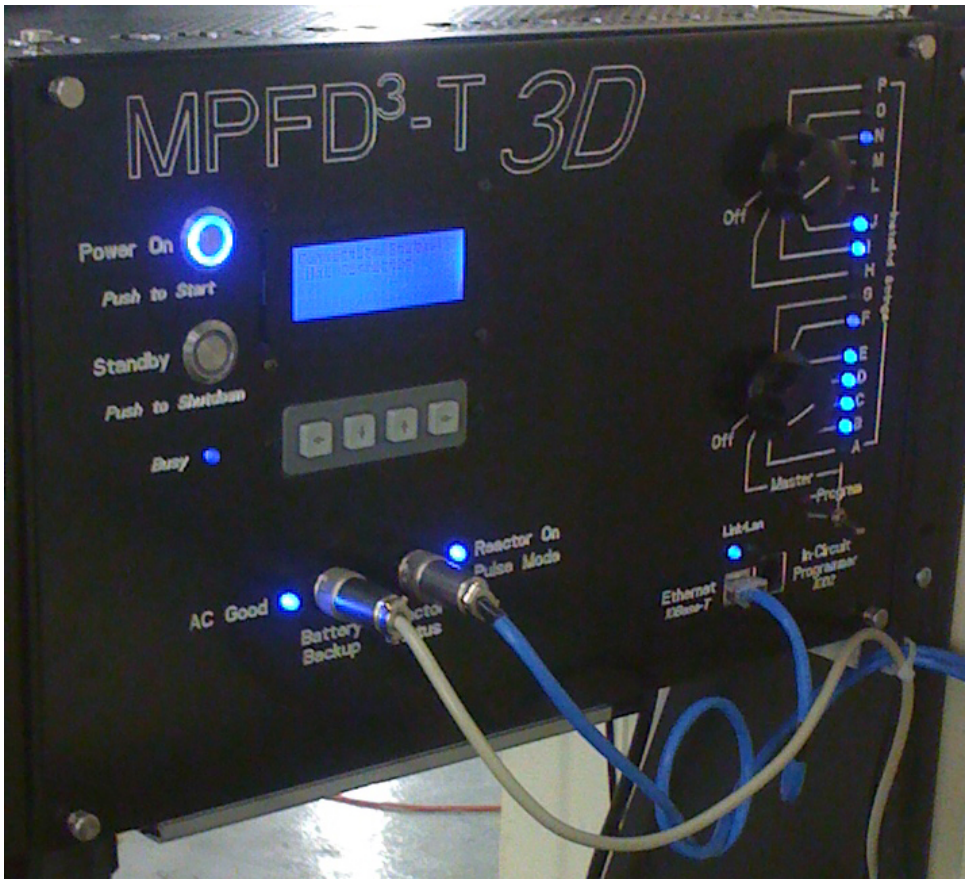


Figure 5.14: Picture of the front panel of the MPFD^3-T 3D System.

Good"), reactor state ("Reactor On" and "Pulse Mode"), and Ethernet state ("Link" and "Lan"). There is also a 20 character by 4 line, or 20×4, liquid crystal display (LCD) that provides additional System status information. The display is part of a Matrix Orbital LK204-25-WB display module that incorporates RS-232 and I^{2}C communications and attachment options for an external keyboard. A four button, left-down-up-right keypad was installed in the front panel and is connected to the LCD module. While the microcontroller (Section §5.2.1) has native I^{2}C communication, the LK204-25-WB's I^{2}C implementation requires excessive delays to be included, which dramatically reduces the effectiveness of the I^{2}C bus. However, there were no additional data pins available on the mother board microcontroller in order to utilize the LK204-25-WB's RS-232 data bus. Therefore, a NXP SC16IS752 demo board was placed between the mother board's I^{2}C bus and the LK204-

25-WB's RS-232 data bus. The SC16IS752 is designed as a I^2C to UART (RS-232) first-in first-out (FIFO) protocol converter. This allows the LK204-25-WB to be used without slowing down the I^2C bus speed.

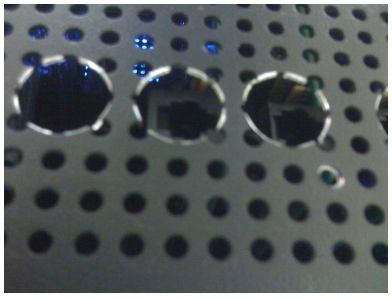


Figure 5.15: Holes in the top cover for access to the programming port on each daughter board.

The right hand side of the front panel has LEDs labeled A-J and L-P to match up with the usable flux probe holes (note that K is excluded). In addition to these LEDs are two large rotary switches, an "In-Circuit Programmer" port and a "Program" switch. The original intention of this port and switch system was to allow programming access to each daughter board and the mother board microcontrollers via the front panel. Unfortunately, the distance of the communication lines between the port, the switches, the routing on the mother board and the routing on the daughter boards proved to be too long for successful programming. Anticipating this potential problem, a programming port was included along the top edge of each daughter board. Holes were added in the top cover of the housing in order to allow access to the programming ports (Figure 5.15). The A-J and L-P LEDs serve a secondary purpose of showing when daughter boards and detector strings are connected. The LED is dim when just the daughter board is installed and bright when the detector string is connected.

5.3.2 Back Panel

The back panel is where the $MPFD^3$-T strings are connected to the System using 37-pin D-SUB connectors. Above this row of connectors are a series of PJ-040D 1.35 mm ID, 3.5 mm OD barrel jacks. These jacks allow external NIM equipment to be utilized for analysis beyond that of the $MPFD^3$-T 3D System electronics capabilities for the pulse mode signals. Slots for unpopulated daughter boards have been taped over in order to improve

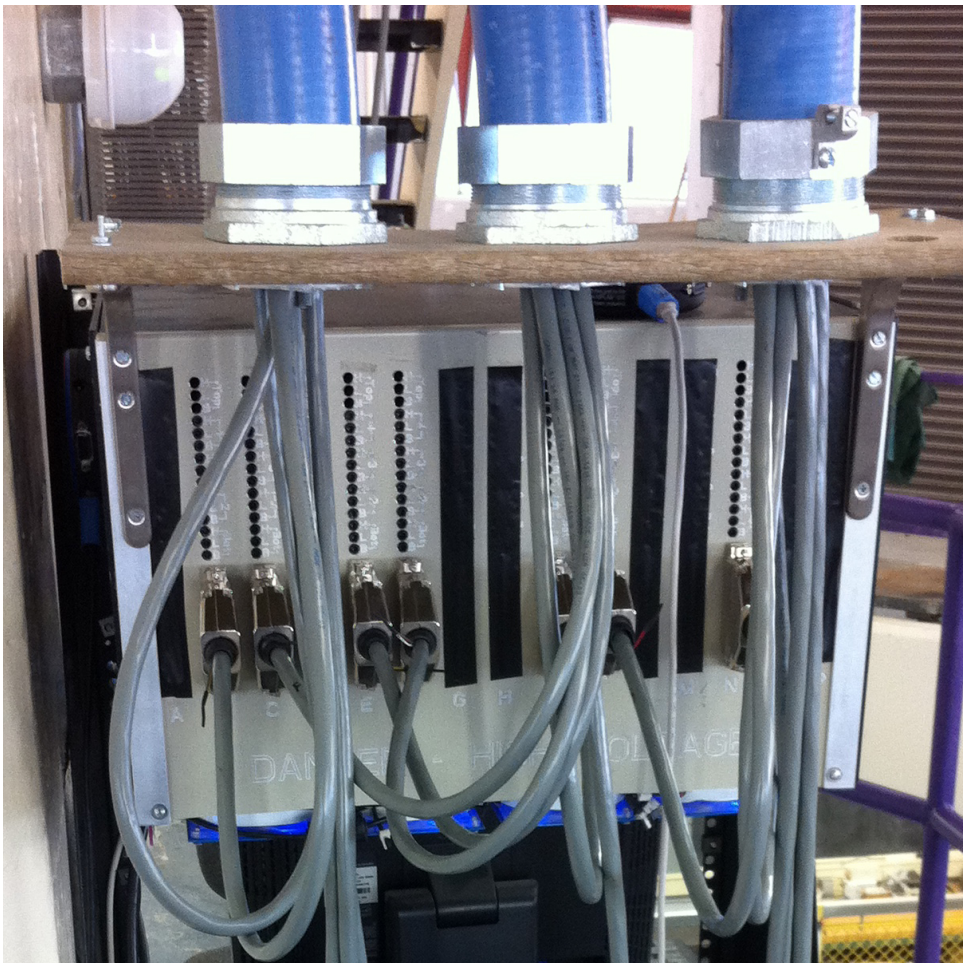


Figure 5.16: Picture of the back panel of the MPFD^3-T 3D System.

airflow inside the housing.

5.3.3 Reactor Status

The reactor status is utilized by the system to control the data communication and external operations. Knowing the reactor status is accomplished through the use of two Hall effect sensors. One sensor is mounted near the control panel key relay that activates the Hall effect sensor whenever the control panel key is switched to the on position. A second sensor is mounted near the pulse mode relay. Through these sensors, the MPFD^3-T 3D System can test the reactor state without affecting the operation of the reactor.

5.3.4 Power Distribution

One of the primary responsibilities of the mother board is to distribute power to all of the daughter boards. The power supply is a medical grade Xgen series power supply built from an XLC powerPac frame with two Xg3 20 A powerMods and one Xg7 5 A powerMod. Together this gives a power supply of 12 V at 45 A. This power supply is attached outside the MPFD^3-T 3D System housing in order to minimize noise from the power line and from the switching circuitry of the power supply. The line power is filtered prior to the power supply through a CyberPower PR2200LCDRT2U uninterruptible power supply (UPS), which also provides up to 1 hour of run time should the building power be interrupted. The UPS provides a USB connection to the data collection computer in addition to a contact closure serial connection, which the MPFD^3-T 3D System monitors. In the event of a building power interruption, this monitored line will change state and an internal software timer will start. If the line state does not change back within 30 minutes, the MPFD^3-T 3D System will initiate a shutdown sequence.

Noise is a primary concern whenever working with power distribution systems. And while the power converter is important, the source of the power is also critical. When performing tests with prototype detectors, significant noise problems were encountered, especially when the reactor's cooling pumps were running, or even when the control rod drives were engaged. Nearly all of the power lines ran through the same cable tray between the control room and the reactor bay. To correct this, an isolated power line was run directly from the MPFD^3-T 3D System all of the way to the breaker panel through its own conduit with a special ground isolated outlet that separates the ground line from the conduit. During the installation of this power line, it was found that the electrical panel had no dedicated safety ground line and instead relied on the conduit between the electrical panel and the building distribution cabinet. A ground wire was added, which cleaned up the power line, not only for the MPFD^3-T 3D System, but also for other experimental facilities in the reactor bay and even the control panel instrumentation.

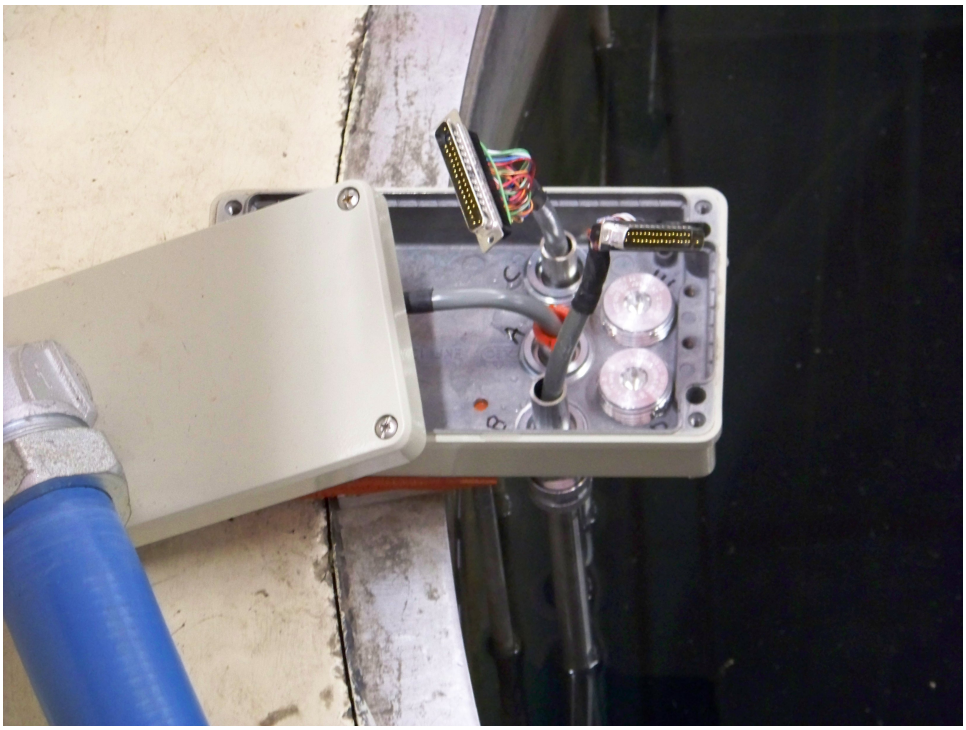


Figure 5.17: Picture of the probe junction box which mounts to the upper edge of the reactor tank and secures up to five MPFD^3-T strings.

As a side note, the reactor itself is actually grounded. When MPFD^3-T strings are loaded into the grid plates of the reactor, they become electrically tied to the reactor. Mounting the top of the MPFD^3-T strings to the edge of the reactor tank then electrically ties the tank and strings together. The problem with this is that the reactor is electrically connected to the tank, which results in a large ground loop. Not only does this ground loop act as an antenna for the EMI present in the reactor bay, but the reactor generates a large number of charged particles that generate noise as well. Since there is no way to isolate the detector strings from the reactor core, the only option for breaking the ground loop is to isolate the MPFD^3-T strings from the reactor tank. This was accomplished by using plastic standoff washers and a foam rubber pad between the tank wall and the junction box which connected up to five MPFD^3-T strings. Each string was held in place by a rubber compression fitting to provide a second layer of isolation.

5.3.5 Cooling

With up to 540 Watts of power being used to power all of the electronics, heat removal from the MPFD^3-T 3D System housing is crucial. The bottom plate of the housing has an array of $1/4$" holes along the back side of the plate. Four 80 mm computer fans with filter covers are attached to cover this area of holes. The top plate, likewise, has the same array of holes but has been oriented so the holes are near the front of the housing. Therefore, air blown in by the fans is forced to flow across all of the daughter boards in order to escape out of the top holes. In order to minimize noise, the fans are located on the outside of the housing and run from their own power supply. This external power supply also allows the fans to continue to run even when the MPFD^3-T 3D System is powered off.

Chapter 6

Embedded Programming and User Interface

It is the programming of the mother board and daughter board microcontrollers along with the User Interface Application that brings all of the hardware together into a functional system. All three of these programming sections are described in the following chapter along with a description of how they all work together.

6.1 Embedded Programming

The mother board and daughter board both use microcontrollers from Microchip Technologies, Inc. and so Microchip's MPLAB IDE version 8.60.00.00 with MPLAB C18 version 3.36 was used for programming. The IDE software organizes all of the program files and provides access to the programmer drivers for the In-Circuit Debugger (ICD). The ICD, commonly referred to as the hockey puck due to its shape, provides a conversion from the computer's USB port to the RJ11 programming port of the microcontrollers. ICD2 and ICD3 programmers were both used. The C18 software functions as a C compiler optimized for Microchip's series 18 microcontrollers.

6.1.1 Daughter Boards

The daughter board program has a narrow focus. It must read the pulse, MSV, and current mode data from all the detectors, set the high voltage and lower level discriminator (LLD), and communicate with the mother board microcontroller. On startup (Section §B.6.1), the microcontroller verifies that the high voltage DAC (Section §5.1.4) is set to zero and the LLD DAC is set high. The relay fuses (Section §5.1.5) are reset once the high voltage

is verified as being off, based on the ADC reading. Then a 10 second delay is initiated in order to wait on the mother board to initialize. Pins 34 through 43 of the daughter board microcontroller have been soldered together in a unique pattern to indicate the serial number of each daughter board. These values are transfered to the mother board once communication between the mother and daughter boards has started.

The daughter board then waits for the trigger signal from the mother board. Once received, the daughter board microcontroller begins transferring data from all of the I^2C ADC devices (Section §B.6.9) and sending any updated settings out to the I^2C high voltage and LLD DAC devices (Section §B.6.11). Next, it stops the pulse mode counters, transfers the current count bits to the parallel-to-serial shift registers, clears the counters, restarts the counters, and then shifts the data to the microcontroller (Section §B.6.13). Conversions (Section §B.6.15) of the ADC values are required in order to account for data shifts caused by gain settings and sample frequency settings.

The data is then transfered through the "TransmitData()" function of the main.c program (Section §B.6.1). This function could be much shorter than the 1814 lines by using a "FOR" function, but this long method was used in order to minimize the overall number of cycles the microcontroller had to go through to complete the function. This optimization allows the $MPFD^3$-T 3D System to operate at 9 samples per second instead of 7 samples per second. Once the data is transfered, the program waits for the next trigger to show up.

6.1.2 Mother Board

The mother board programming is far more complex than the narrowly focused daughter board programming. When the motherboard first powers up, the main program (Section §B.7.1) configures the I^2C interface followed by turning off the "Standby" LED and turning on the "Power On" LED. The system timers (Section §B.7.15) are then initialized. The LCD display (Section §B.7.8) is then initialized and the start up screen in Figure 6.1 is shown.

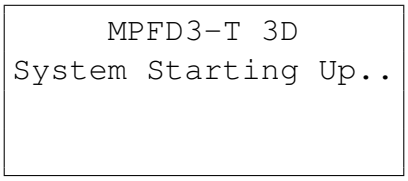


Figure 6.1: Startup screen of the MPFD^3-T 3D System's LCD.

The MAC (Medium Access Control) address is a unique address assigned to every network accessible interface in order to make sure data packets are sent to the correct device. These addresses must be registered through the IEEE Registration Authority Committee as defined by the ISO/IEC 8802 or ANSI/IEEE 802 standards. The Microchip Technology, Inc. PIC18F97J60 microcontroller is capable of Ethernet communication but lacks an assigned MAC address. An individual is not able to purchase a single address from the IEEE Registration Authority, so to solve this problem, one may purchase a special memory chip that has one of these unique MAC addresses stored inside it. The DS2502 is one such device. However, it communicates over a protocol known as the 1-wire bus, so a DS2482 converter is necessary to communicate with the DS2502 on one end and the I^{2}C based microcontroller on the other end. It is at this point that the MAC address is read from the DS2502 through the ds2482 driver (Section §B.7.5) for later use by the microcontroller [29].

The Microchip File System, or MPFS, is initialized (Section §B.7.17), which loads the driver for the 24FC1025 I^{2}C EEPROM (Section §B.7.16). The Microchip Technologies, Inc. TCPIP Stack is then loaded. This stack controls the Ethernet operations and provides the services for Dynamic Host Configuration Protocol (DHCP) client, Address Resolution Protocol (ARP) client and server, Domain Name System (DNS) client, Dynamic DNS (DynDNS) client, Internet Control Message Protocol (ICMP) server, Transmission Control Protocol (TCP) communications layer, User Datagram Protocol (UDP) communications layer, HyperText Transfer Protocol (HTTP) server, and File Transfer Protocol (FTP) server

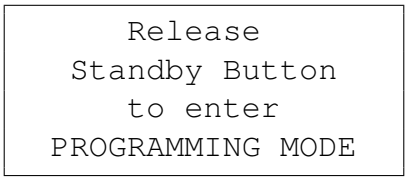


Figure 6.2: Screen of the MPFD^3-T 3D System's LCD shown when the user holds the "Standby" button upon starting the System.

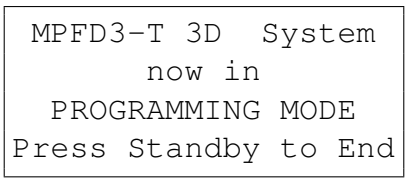


Figure 6.3: Programming mode screen of the MPFD^3-T 3D System's LCD.

[30]. Interrupts that control the system timing and sensing of the "Standby" button are then enabled. If the "Standby" button has been held down since pressing the "Power On" button, the System will be placed into a programming mode at this time by showing the screen in Figure 6.2.

Once the button is released, the screen switches to that shown in Figure 6.3 and the MPFD^3-T 3D System enters a "WHILE" loop where the "Power On" and "Standby" buttons start to alternately flash on and off. This action prevents the MPFD^3-T 3D System from initializing the daughter board communications and holds the daughter boards in a state that is easier to program with the ICD. The programming mode is exited by pressing the "Standby" button, which causes the MPFD^3-T 3D System to shut down.

In most cases, the "Standby" button will not be held down during startup, and so the MPFD^3-T 3D System will continue past this point to perform the initialization of the previously loaded TCPIP Stack. The TCPIP Stack initialization is followed by initialization of the mpfd3d.c task, which prepares the daughter boards for pending communication. It also opens a UDP socket for listening to connection requests from the MPFD^3-T 3D

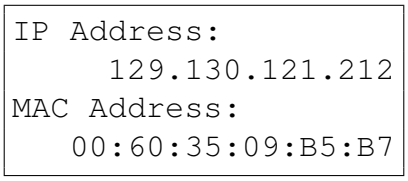


Figure 6.4: IP address and MAC address screen of the MPFD^3-T 3D System's LCD.

System's User Interface Application. Also, the various data arrays and variables are initialized for later use. The microcontroller's clock is then initialized through the SNTP task (Section §B.7.19) by reading the time from the on-board I^{2}C DS3231 real time clock (Section §5.2.1). The on-board I^{2}C TC74A5 temperature sensor is then initialized and prepared for reading.

Dynamic DNS is an online service (http://dyndns.org) that provides support for knowing a remote computer's IP address even though its IP address is dynamically assigned by the network on which it resides. Dynamic DNS is accomplished by having the computer send a notice to the dyndns service of its current IP address. When a user wants to find out the computer's IP address, a request it is made by searching for the static name, such as mpfd.getmyip.com, whereby dyndns looks up this name and returns the associated IP address. It is at this stage that the program configures the DynDNS service.

Now the MPFD^3-T 3D System is ready to enter a continuous "WHILE" loop. During this loop, the TCPIP Stack processes incoming Ethernet packets and sends out replies while the mpfd3d.c controls the daughter boards, collects and packages their data, and controls interactions with the MPFD^3-T 3D System's User Interface Application. One of the first tasks of the TCPIP Stack is to get an IP address from the network's DHCP server, which is done by the DHCP client program. Whenever a change is made to the assigned IP address, the LCD changes its display to indicate the new address as shown by Figure 6.4. Failure to retrieve an IP address will cause the MPFD^3-T 3D System to lock up and block all further activity as a safety precaution. In the event that a lock up occurs, check for network

activity on the "LAN/LINK" LEDs and ensure that the network cable is good and that the network switch in the control room is functioning properly. Recovery from this requires the MPFD^3-T 3D System to be shut down and restarted.

Following successful IP address assignment, the DynDNS.c program transmits the IP address to the dyndns.org servers and also queries the Kansas State University's time server for the current time using the SNTP functions (Section §B.7.19). The returned value is compared to the current time value based on what was previously loaded from the DS3231 real time clock. If the difference is greater than 2 seconds, then the internal time value is adjusted and a unique capability of the DS3231 is implemented that allows the oscillator frequency of the DS3231 to be slightly adjusted in order to maintain an accurate time stamp.

The MPFD^3-T 3D System waits 20 seconds from a successful SNTP query to begin communication with the daughter boards in order to allow them to complete their start up processes. The first task is to query all of the daughter board data lines. Each line has a built-in 47 kΩ pull up resistor that sets the data line high while the first task of the daughter boards is to set each data line low. From this, the mother board can sense which daughter boards are connected. Should a set of data lines not show up as all high (1's) or low (0's), then the series of screens in Figures 6.5, 6.6 and 6.7 are shown. Normally, no error will be present, and so the MPFD^3-T 3D System will continue on and show the connected board positions and serial numbers via the screen in Figure 6.8.

Once communication with the daughter boards has been established, the mpfd3d.c program enters a timed cycle that reacts to the count rate of the pulse mode electronics. If

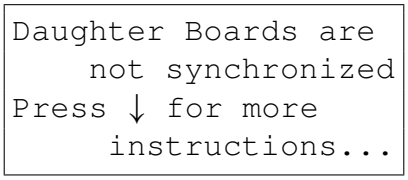


Figure 6.5: First screen shown on the MPFD^3-T 3D System's LCD when the at least one of the data lines is incorrect on initialization.

```
Send data from next
screen (Press ↓) to:
  martin@ohmes.com
then push Standby
```

Figure 6.6: Second screen shown on the MPFD^3-T 3D System's LCD when the at least one of the data lines is incorrect on initialization.

```
Current Data Lines:
A111B000C000D000E000
F000G111H111I000J000
L111M111N000O111P111
```

Figure 6.7: Current data lines screen of the MPFD^3-T 3D System's LCD.

```
Detector Strings:
 [-BCDEF--IJ--N--]
Daughter Boards:
 [-02375--91--a--]
```

Figure 6.8: Connected boards screen of the MPFD^3-T 3D System's LCD.

the most significant bit of any of the connected pulse mode channels is one, indicating a value greater than 32,767, then the samples per second increases to react to a higher count rate. Likewise, if the two most significant bits are both zero, indicating a value less than 16,384, then the samples per second decreases in order to minimize the data flow. The current sample rate can be viewed by pressing the up (↑) or the down (↓) arrow of the LCD keypad in order to scroll to the screen shown in Figure 6.9.

Once the time between samples has expired, the trigger signal will be activated, which causes the daughter boards to initiate their data collection cycles. The time stamp is recorded, and the mother board continues on to other tasks until all of the daughter boards signal that they are ready to transfer data down to the mother board. Once this signal is

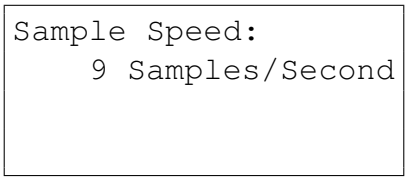


Figure 6.9: Sample speed screen of the MPFD^3-T 3D System's LCD.

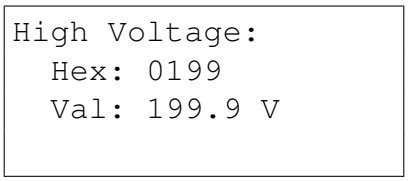


Figure 6.10: High voltage screen of the MPFD^3-T 3D System's LCD.

acknowledged by the mother board, a data transfer dance begins between the mother board and daughter boards in order to upload any new settings to the daughter boards while the collected data is downloaded to the mother board all at the same time. The downloaded data is stored directly into an array of 1327 bytes, which if a connection has been established between the mother board and the MPFD^3-T 3D System User Interface Application, will be transfered to the TCPIP Stack to be sent out over the network as a UDP packet.

The "other tasks" mentioned above consist of responding to LCD keypad requests and operations relating to the TCPIP Stack. The user is able to cycle between the status screens by using the up ($\uparrow$) and down ($\downarrow$) buttons. The screens which are available are the connected daughter boards (Figure 6.8), the high voltage setting (Figure 6.10), the LLD setting (Figure 6.11), the sample speed (Figure 6.9), the internal MPFD^3-T 3D System temperature (Figure 6.12), the current data lines (Figure 6.7), the System IP and MAC addresses (Figure 6.4), and the connection status (Figure 6.15 or Figure 6.14). In addition, the left ($\leftarrow$) and right ($\rightarrow$) buttons can be used to control the contrast of the LCD screen, and when pressed will present a screen similar to Figure 6.13. A value of 127 is normal but may change over time as the LCD ages.

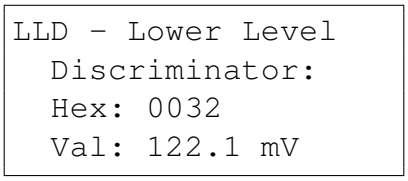


Figure 6.11: Lower level discriminator (LLD) screen of the MPFD^3-T 3D System's LCD.

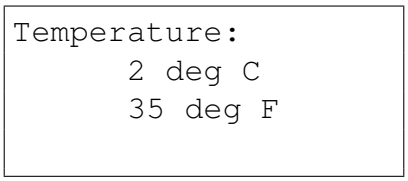


Figure 6.12: System temperature screen of the MPFD^3-T 3D System's LCD.

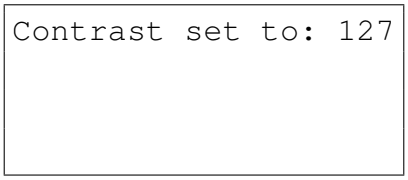


Figure 6.13: Screen shown when the contrast of the MPFD^3-T 3D System's LCD is adjusted.

The TCPIP Stack operations include responding to ping and ARP requests, maintaining the DHCP authorization, and maintaining the real time clock's accuracy through SNTP inquiries. In addition, a small web server is available to quickly determine online availability. The web server can be accessed only when connected to the K-State network by opening a web browser and going to the address http://mpfd3t3d.getmyip.com. UDP and TCP requests will usually be dropped since the MPFD^3-T 3D System does not know what to do with them and so they end up being ignored. However, UDP requests adhering to strict patterns will be processed by the mpfd3d.c program in order to communicate with the MPFD^3-T 3D System's User Interface Application. A "CONN" UDP request will initiate a connection request, whereby the MPFD^3-T 3D System will respond with either a "CONN" response, a "WISH" response, or a "DENY" response. A "CONN" response is issued if

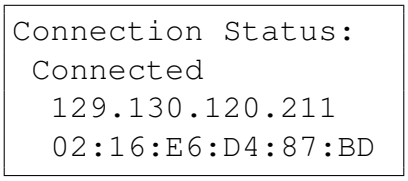


Figure 6.14: Connected status screen of the MPFD^3-T 3D System's LCD.

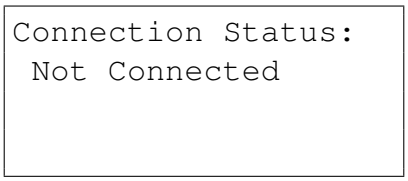


Figure 6.15: Not connected status screen of the MPFD^3-T 3D System's LCD.

no other computer is already connected to the MPFD^3-T 3D System or if the IP address of the requester is the same as the IP address of the currently connected computer. If there is already a computer connected to the MPFD^3-T 3D System then the "WISH" response will be given if the reactor is not in pulse mode. Otherwise, a "DENY" response is sent. The "WISH" response initiates a question box in the User Application Interface that asks the user if they want to take over the connection, thereby disconnecting the other computer from the MPFD^3-T 3D System. When a connection has been established, the LCD screen will change as shown in Figure 6.14. Upon termination of a connection, the LCD screen will likewise change to show Figure 6.15.

Connections require routine requests to be sent from the User Interface Application, otherwise the connection will be closed so the MPFD^3-T 3D System doesn't flood the network with dead packets going to nowhere. One way to maintain contact is for the User Interface Application to send out a "HERE" message, which means "I am here - don't disconnect me please." A "SETT" message may also be sent, which informs the MPFD^3-T 3D System that there are new settings such as high voltage or LLD settings that must be updated. These new settings are sent along with the "SETT" message and the MPFD^3-T 3D

System will respond with a “CHNG” or changing notice. A “DENY” response message will be sent instead if the reactor is currently in pulse mode in order to limit transients during the time sensitive pulse mode operations. Successful upload of the new settings to the daughter boards will cause a “DONE” response message to be sent. Data transfer begins when the user interface sends the “STRT” command and stops when the “STOP” command is sent. Each data packet that is sent from the MPFD^3-T 3D System starts with the key word “DATA” for easy direction in the User Interface Application. Communication is halted when either the System or the User Interface Application sends a “KILL” request. Once the connection is closed, the LCD displays the screen shown in Figure 6.15.

Two standard communication protocol options are available for data transfers: TCP/IP and UDP/IP. The primary difference between these two protocols is that UDP is a connectionless protocol while TCP is a connected protocol. Unlike UDP, TCP must negotiate and verify that a connection exists between the two hosts before sending data. TCP will also verify that each data packet is received intact. UDP does not monitor whether or not the data makes it to the receiving host and thus only sends the data packet out to the network [31]. While TCP will guarantee that the data is transmitted successfully, TCP also requires quite a bit of extra overhead. Therefore, the UDP/IP protocol was selected for transmission of all “DATA” type packets. Even if a “DATA” packet does get lost or corrupted, it is not detrimental to the rest of the system since each packet contains a start and end time stamp for that set of data.

The FTP server of the TCPIP Stack has been modified to work with the 24FC1025 1024K bit I^{2}C serial EEPROM. This allows the User Interface Application to connect to the MPFD^3-T 3D System via FTP for uploading and downloading its settings file. The application of this will be discussed in more detail in Section §6.2.

Shutting down the MPFD^3-T 3D System is initiated by pushing the “Standby” button. This button is sensed by a hardware interrupt that triggers the “OFFButtonPressed()” function of the mpfd3d.c program. If the button is held down for more than 10 seconds, the

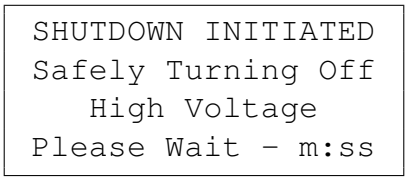


Figure 6.16: Shutdown progress screen of the MPFD^3-T 3D System's LCD where m:ss indicates the minutes and seconds remaining in the shutdown process.

MPFD^3-T 3D System will be forced to shut off immediately. However, a rapid shutdown is not a good idea when working with high voltage because it induces a pulse in the capacitors exposed to the high voltage per

$$i = C\frac{dv}{dt}. \tag{6.1}$$

In Equation (6.1), the rapid decrease in high voltage dv that occurs during a short time period dt for the capacitor of value C will induce a current i. This current goes through neighboring resistors; specifically for C503 in Figure 5.7 the current will generate a voltage across R512, which can damage the preamp. To avoid this risk of damage, the normal shutdown method initiated by briefly pushing "Standby" triggers a sequence of operations, specifically one that causes the high voltage on each daughter board to shut down slowly. Figure 6.16 will update the m:ss labels with the remaining minutes and seconds until the high voltage supply has been shutdown, followed by the MPFD^3-T 3D System powering off.

6.2 The MPFD^3-T 3D System's User Interface Application

The User Interface Application (UIA) was developed using Microsoft Visual Basic 2005 in the Visual Studio 2005 Professional Edition environment. Version control was provided through the add-on BuildVersionIncrement version 2.2.10132.1603 maintained by pipelinecache on CodePlex [32]. The XYGraph (version 3.0.3765.29174) and the Ion-

ic.Zip.Reduced (version 1.9.5.1) .NET plug-ins are utilized to respectively produce real-time plotting and to provide zip compression of the large raw data files. The XYGraph plug-in is developed by componentXtra and requires a paid license for the "Advanced" version of the software [33]. The license allows compiled code to be distributed freely but restricts development to a single registered computer. Ionic.Zip.Reduced is part of the freely available DotNetZip project which is maintained by Cheeso and Jaans on CodePlex [34].

Figure 6.17: How to open the UIA via quick links.

The UIA is a fully compiled and deployed program that can be launched from the start menu or through icons similar to other programs as shown in Figures 6.17 and 6.18. Upon starting the application, it will check an online repository run from a computer located under the MPFD^3-T 3D System. Should a newer version of the application be available, the screen in Figure 6.19 will be shown, allowing the user to download the latest version of the software and install it. Upon installing the software for the first time or after installing an update, windows requires the basic settings to be redefined, which is first indicated by a message box (Figure 6.20) that states that the local settings and/or

Figure 6.18: How to open the UIA via the "All Programs" menu.

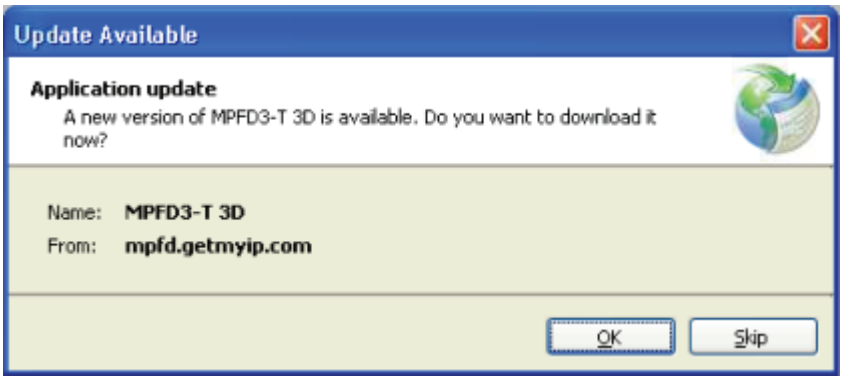


Figure 6.19: Screen shown when a software update is available for installation.

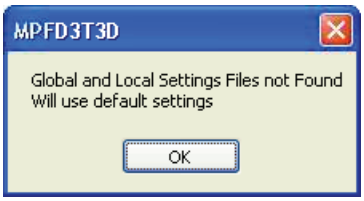


Figure 6.20: Global and local settings missing warning.

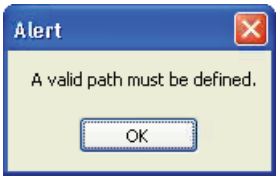


Figure 6.21: Valid path definition warning.

the global settings file have not been defined. An alert (Figure 6.21) will then appear that states that "A valid path must be defined." This path is used to define the storage of all the data collected from the MPFD^3-T 3D System. Once the "OK" button has been pressed, the "Storage Path" window of Figure 6.22 will be displayed. Use the "Browse" button in order to identify which drive should be used for storage. A main folder of "MPFD3T3D" will be created on this drive along with several sub-folders. The "LiveData" sub-folder is used to store the live data as it is collected from the MPFD^3-T 3D System, and it is also used when replaying saved data. All of the processes utilizing this space are automated and no user modifications should take place. A second set of sub-folders are made for permanent storage of the data. These folders are based on the date the data started being recorded as "YYYY\MM\DD" where YYYY is the four digit year, MM is the two digit month, and DD is the two digit day of the month. A similar set of folders are created inside a sub-folder

called “Converted,” which is used to store the files that have been converted from binary to comma separated values (csv) files, which can be read by Microsoft Excel or MATLAB. Calibration data files are also stored in this folder.

The UIA is a typical windows application with a menu bar at the top, a data area in the middle, and a status bar at the bottom, as seen in Figure 6.23. The status bar shows information about the current system status such as whether or not the MPFD^3-T 3D System is connected gathering live data or if old data is in the process of being replayed. Also shown is the “Reactor Status,” the building power condition (“AC Power Status”), and the rate of data collection in samples per second (“Sample Interval”). The “Total Samples” is an indicator of the number of data samples stored in the display array vs the total number of data samples collected since data collection started. If a new data file is started in the live data mode, the number of data samples collected will reset to zero while the total including prior files is shown in parentheses. The current file name is also displayed in the status bar. The data area can contain up to five plots: pulse mode data, mean-squared-voltage mode data, current mode data, temperature data, and high voltage data. These plots and their settings will be discussed later in this section. The menu bar contains the main level options of “File,” “Settings,” “Plots,” “Notes,” and “About.” The “About” option brings up a window that displays the version and copyright information along with a description with contact information. The “Notes” section displays a simple

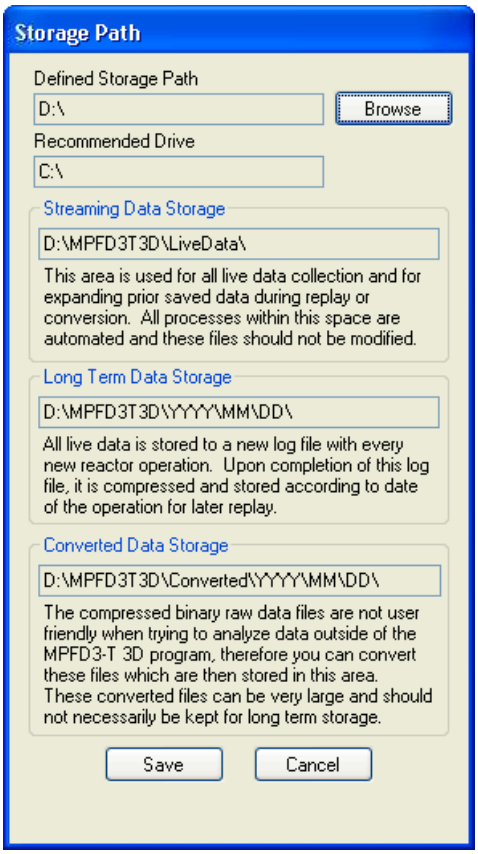


Figure 6.22: Storage path screen for defining where the data is stored.

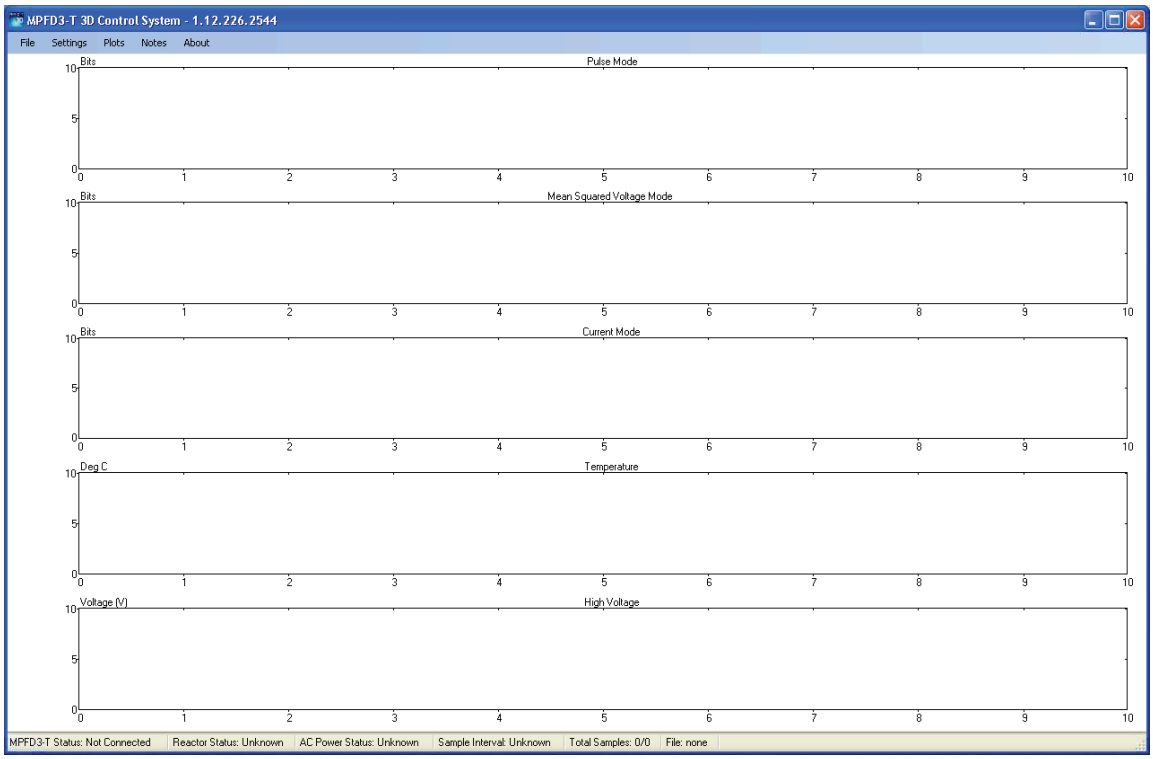


Figure 6.23: The main screen of the MPFD^3-T 3D System's User Interface Application.

text file that can be used to share configuration settings for optimal performance.

To take advantage of the settings storage capability of the MPFD^3-T 3D System (Section §6.1.2), the network configuration must be set through the "Network Adapter" option of the "Settings" menu (Figure 6.24). Selection of this menu option will open the "Network" window shown in Figure 6.26. From the drop down menu under "Network Adapter," the adapter that shows the most reliable connection to the MPFD^3-T 3D System should be selected. This adapter will be used for the routing of all network communications with the MPFD^3-T 3D System. Once saved, the "Live Data" option from the "File" menu (Figure 6.25) will open the window shown in Figure 6.27. The "Connect" button serves two different purposes. If clicked using the left mouse button, the "Connect" button will initiate a connection to the MPFD^3-T 3D System and if successful, will download the last saved settings from the System. This settings download is especially useful for

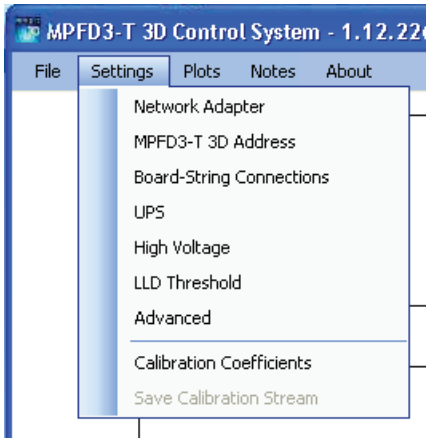


Figure 6.24: Settings menu of the User Interface Application.

Figure 6.25: File menu of the UIA.

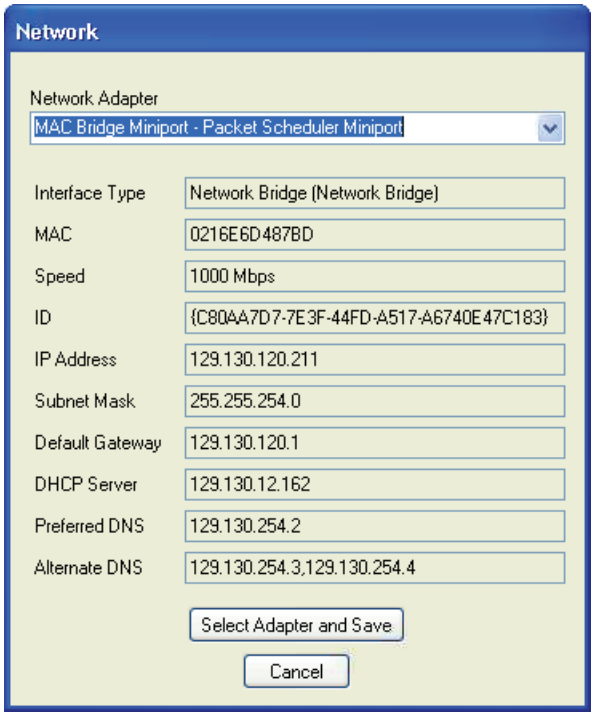


Figure 6.26: Network adapter settings window of the User Interface Application.

restoring settings to a recently updated application but also allows users to maintain consistent settings if switching between different computers. Right clicking the button will force the current settings from the local machine to be uploaded to the MPFD^3-T 3D System. This option will overwrite any settings that were previously saved on the MPFD^3-T 3D System. Shown in Figure 6.28 is the left click connection process that first resolves the IP address of the MPFD^3-T 3D System from its domain name. Then a "CONN" request (see Section §6.1.2 for network message processing) is sent to the MPFD^3-T 3D System and waits for an acceptance message. Once approved, an FTP session is started to download the "settings.set" file from the MPFD^3-T 3D System. The global settings

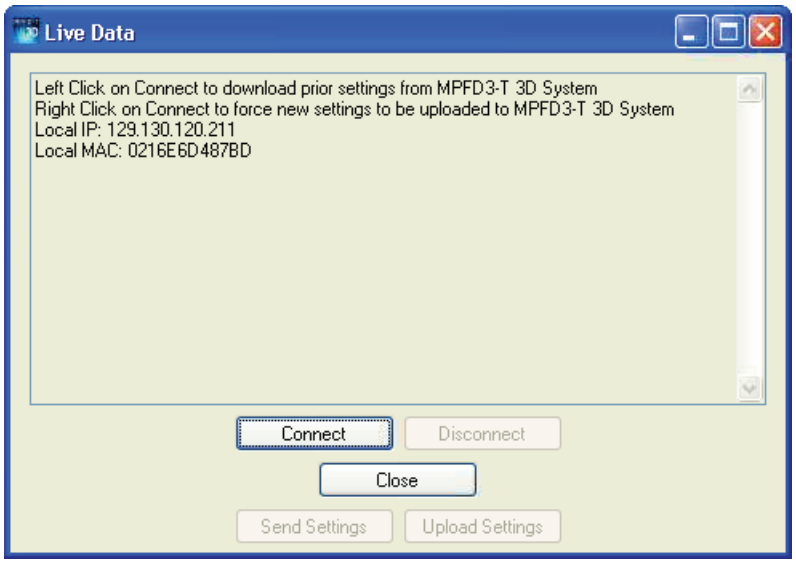


Figure 6.27: Live data window of the User Interface Application.

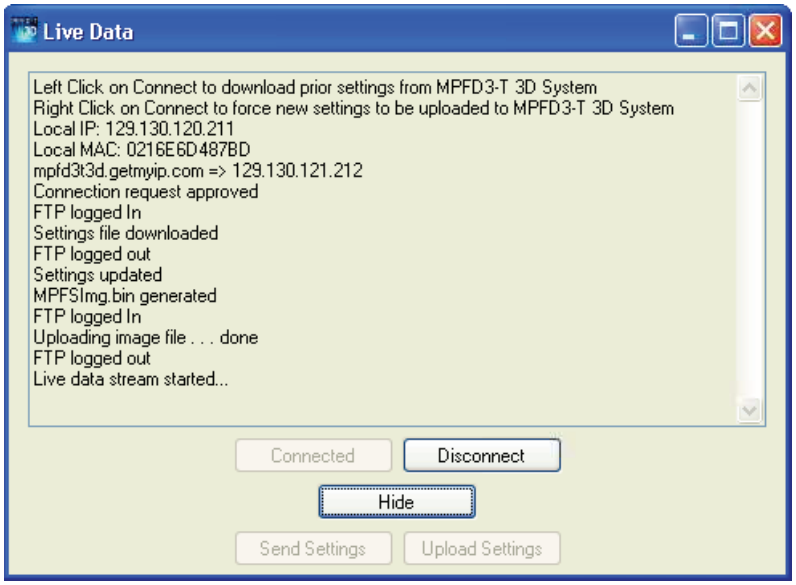


Figure 6.28: Live data window showing connection process and settings download.

are then updated from this downloaded file, followed by the settings file being saved. Then an MPFSImg.bin file is generated that defines how the files are stored on the mother board microcontroller's 24FC1024 I^2C EEPROM. A new FTP session is started in order to transfer the updated MPFSImg.bin and settings.set files back to the $MPFD^3$-T 3D Sys-

tem. Once done, a "STRT" command is sent in order to start the transfer of live data to the UIA.

Configuring the MPFD^3-T 3D System is done through the settings menu (Figure 6.24). The "MPFD3-T 3D Address" menu item opens a window that lets the user define the DNS address of the MPFD^3-T 3D System along with the ability to resolve this into its last registered IP address. Also shown are the receive and transmit ports used by the System, which must be open through the computer's firewall. This window is not shown here for security reasons. The next menu option is the "Board-String Connections," which brings up the window shown in Figure 6.29. Because each daughter board connects to a detector string by a 37-pin D-SUB connector, the "A" string does not have to be connected to the "A" daughter board. Therefore, this window allows the user to inform the UIA as to which string is connected to which daughter board. There are some limitations in that changes may not be made when running in the "Live Data" or "Replay" modes of operations. The window gives some guidelines about how to make the changes.

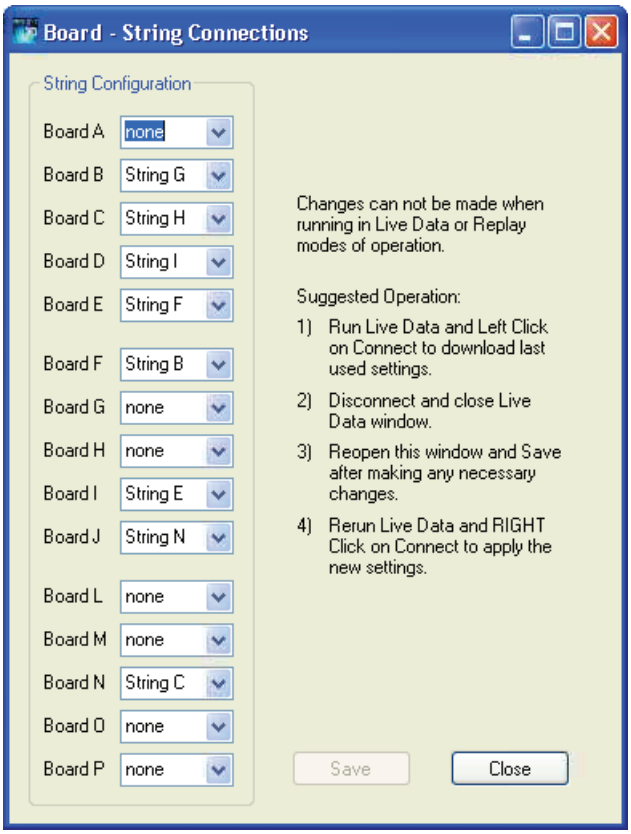


Figure 6.29: Board-string connections window of the User Interface Application.

The next window (Figure 6.30) controls the MPFD^3-T 3D System's response to the UPS. If "Yes" is selected, then the System will begin a timed countdown whenever there is a disruption in the building's power. Once timed out, the MPFD^3-T 3D System will shut

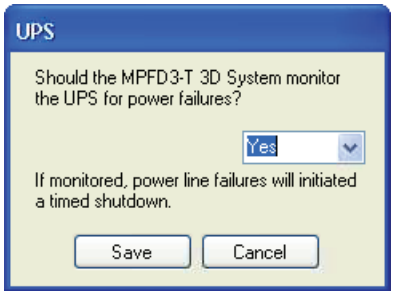


Figure 6.30: The UPS control window.

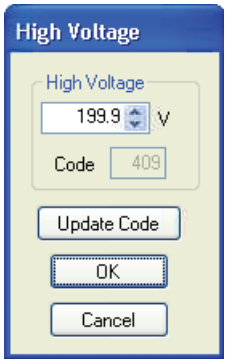


Figure 6.31: The high voltage settings window.

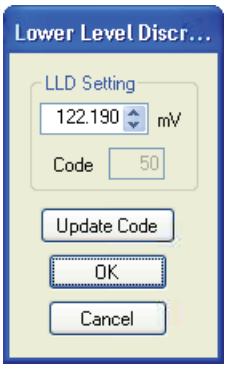


Figure 6.32: The LLD settings window.

down slowly in order to safely shut off the high voltage supply. The high voltage window (Figure 6.31) allows the desired voltage to be either typed or cycled via the up/down numeric box. The "Update Code" button will change the high voltage value to the nearest available value along with showing the corresponding digital-to-analog converter (DAC) code. The lower level discriminator (LLD) is adjusted in the same way as the high voltage through the window shown in Figure 6.32. The advanced settings window (Figure 6.33) allows the user to adjust the gain of the analog-to-digital converters (ADC) for the MSV and current mode operations in addition to the temperature ADCs (Section §5.1.6). There is also a setting that controls whether or not the safety relays (Section §5.1.5) are reset during the MPFD^3-T 3D System startup. Through testing, it was decided to disable all of the advanced settings options except for the temperature ADC gain control, and that is why the other options are grayed out.

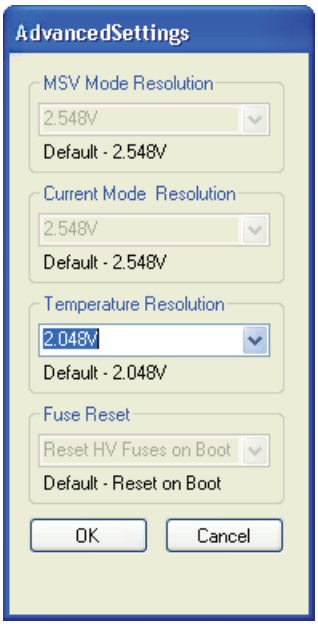


Figure 6.33: The advanced settings window.

The second section of the "Settings" menu (Figure 6.24) deals with the calibration of the MPFD^3-T 3D System. The "Calibration Coefficients" option brings up a large window

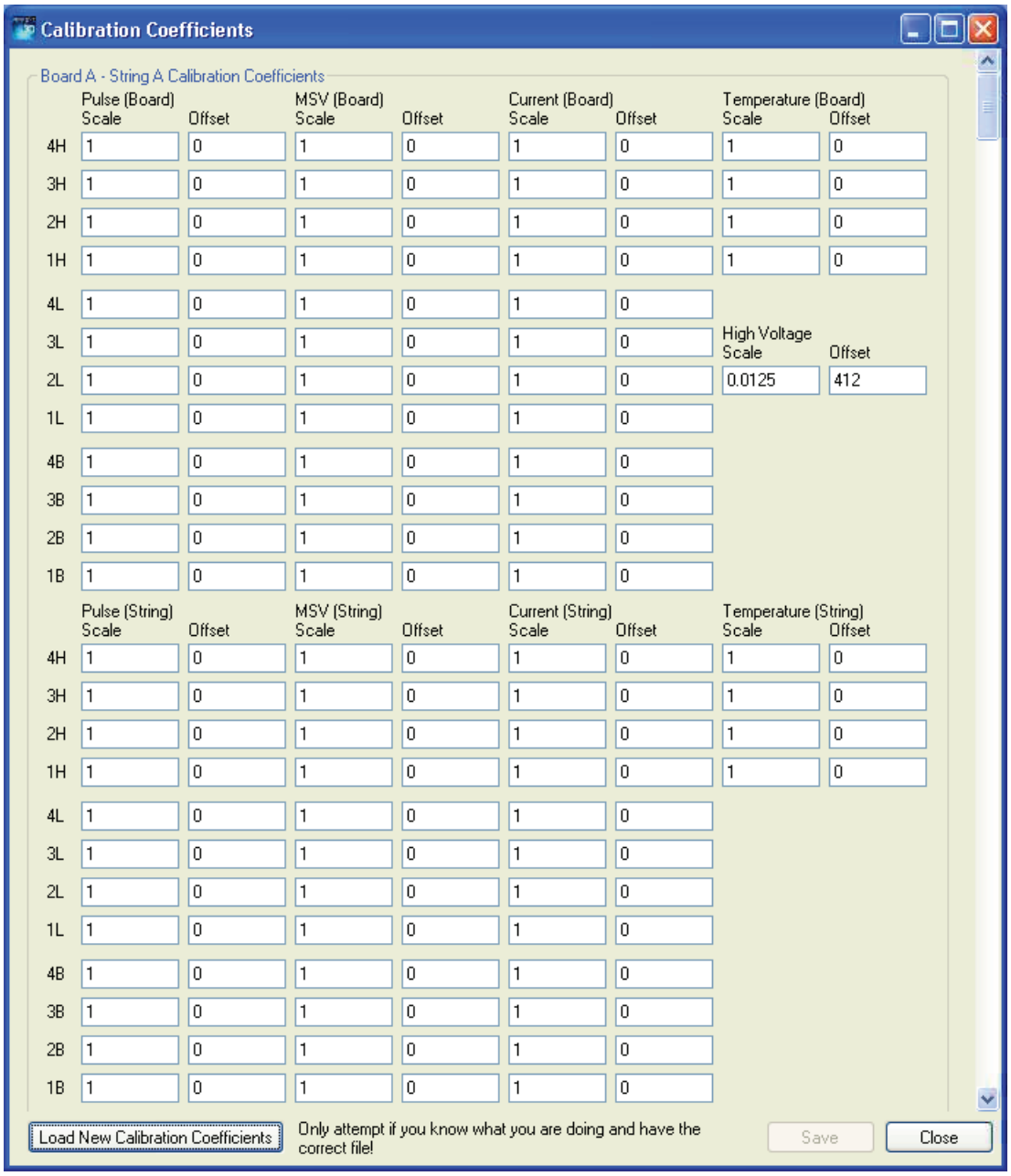


Figure 6.34: Calibration coefficients window of the User Interface Application.

(Figure 6.34) that shows all of the stored calibration coefficients that are currently in use. These coefficients provide a linear scaling through the form

$$y = mx + b \tag{6.2}$$

where m is the "Scale" factor, b is the "Offset" factor, and x is the raw data, while y is the calibrated counts per second output value. The window is sized so that the calibration coefficients for one board and one string will fit on the page at once. In the headings of each table, there is an identifier of (Board) or (String). The purpose of having coefficients for both the boards and the strings is to allow separation of offsets generated by the electronics on the board from any offsets caused by the detectors. Altogether, a total of 2,430 coefficients are necessary to calibrate the full MPFD^3-T 3D System. Because there are so many coefficients and because any changes affect all live and replayed data, the only method to change the coefficients is to load a specially prepared .ccf file. In order to assist in the effort of producing this binary file, a MATLAB program is included in Section §B.8.1 whereby a user may modify and uncomment any coefficient, and when the program is run it will produce the necessary Calibration_Coefficients.ccf file.

The second option of the "Settings" menu's calibration section is the "Save Calibration Stream," which is only available when in the live data mode. This option brings up the window shown in Figure 6.35. The left hand side of the window is similar to the Board-String Connection window in Figure 6.29. The set of drop down menus in the third column of the "String Configuration" section allow the specification of a "Box," which is hardware that can be placed in-line between the detector string and the daughter board. This box option consists of the options: "none," "Control," "Box 1-2," "Box 1-3," "Box 1-4," "Box U-B," and "Box U-T." "Box 1-2" causes the first triad of heavy (H or U), light (L or T), and background (B) detectors to be wired to the second triad's electronics on the daughter board, while the second triad goes to the third triad's electronics, the third triad goes to the fourth triad's electronics, and the fourth triad goes to the first triad's electronics. In the same manner, "Box 1-3" causes the first triad to go to the third triad's electronics, and so on. "Box U-B" maintains the triad's position but causes the heavy coated detector to be wired to the background detector's electronics, and so on. The purpose of these boxes is to assist with trying to determine the calibration coefficients, specifically for zeroing out

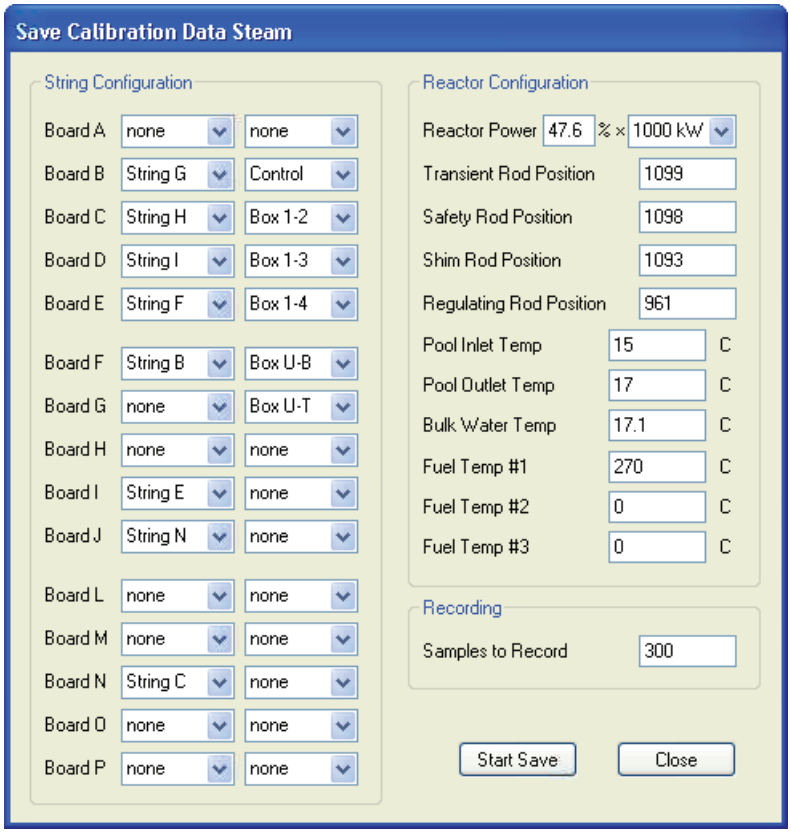


Figure 6.35: Save calibration data stream window of the User Interface Application.

any effects caused by the electronics on the daughter board. The "Reactor Configuration" section allows the user to record the state of all of the reactor's instrumentation at the time the data was recorded. The number of "Samples to Record" determines how many data sets will be saved once the "Start Save" button is pressed. Once committed, a new file with the same name as the live data file currently being utilized will be appended with a number and a ".cal" file extension, and it will be saved in the "Converted" folder. The ".cal" file type is actually a comma-separated values (.csv) file which can be read with Microsoft Excel or with special MATLAB scripts. The reactor and MPFD^3-T 3D System settings are written first to the file followed by sets of binary raw data and converted data. A set of MATLAB scripts (Sections §B.9.1, §B.9.2 and §B.9.3) have been created in order to read the raw ".cal" files and save the data to a MATLAB data structure file (.mat) for faster importing

and analysis. This data structure is shown in Table 6.1.

Table 6.1: Data structure of MATLAB converted ".cal" file. Variables followed by {1,i} represents an array of i samples and (1,1:4) represents an array of 4 columns.

```
data
├─ReactorPower
│  ├─Value
│  ├─Range
│  └─Watts
├─TransientRodPosition
├─SafetyRodPosition
├─ShimRodPosition
├─RegulatingRodPosition
├─PoolInletTemperature
│  ├─degC .2 PoolOutletTemperature
│  └─degC
├─BulkWaterTemperature
│  └─degC
├─FuelTemp1Temperature
│  └─degC
├─FuelTemp2Temperature
│  └─degC
├─FuelTemp3Temperature
│  └─degC
└─BoardA{1,i} through Board{1,i}
   ├─String
   ├─Box
   ├─Present
   ├─StringConnected
   ├─Temperature
   │  ├─Resolution
   │  ├─New(1,1:4)
   │  ├─bits(1,1:4)
   │  ├─mV(1,1:4)
   │  ├─degC(1,1:4)
   │  └─degF(1,1:4)
   └─HV
      ├─Speed
      ├─New
      ├─bits
      └─voltage
```

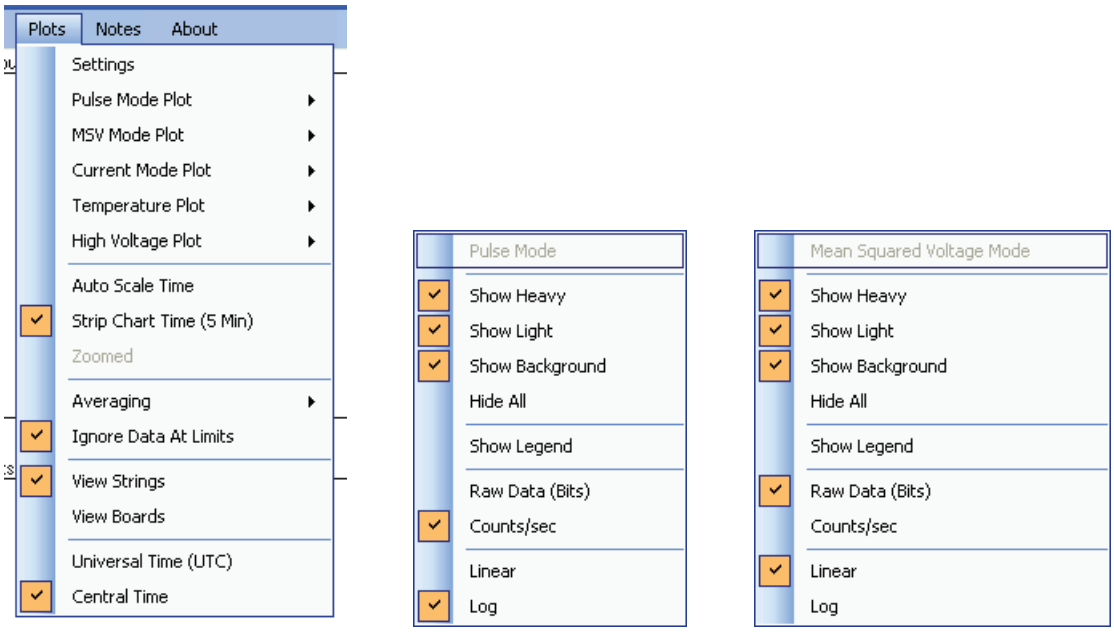
Table 6.1: Data structure of MATLAB converted ".cal" file *(cont.)*.

- data *(cont.)*
  - BoardA{1,i}-Board{1,i} *(cont.)*
    - ADCSpeed
    - MSV
      - L
        - New(1,1:4)
        - bits(1,1:4)
        - cps(1,1:4)
      - B
        - New(1,1:4)
        - bits(1,1:4)
        - cps(1,1:4)
      - H
        - New(1,1:4)
        - bits(1,1:4)
        - cps(1,1:4)
    - Current
      - L
        - New(1,1:4)
        - bits(1,1:4)
        - cps(1,1:4)
      - B
        - New(1,1:4)
        - bits(1,1:4)
        - cps(1,1:4)
      - H
        - New(1,1:4)
        - bits(1,1:4)
        - cps(1,1:4)
    - Pulse
      - L
        - bits(1,1:4)
        - cps(1,1:4)
      - B
        - bits(1,1:4)
        - cps(1,1:4)
      - H
        - bits(1,1:4)
        - cps(1,1:4)

Table 6.1: Data structure of MATLAB converted ".cal" file *(cont.)*

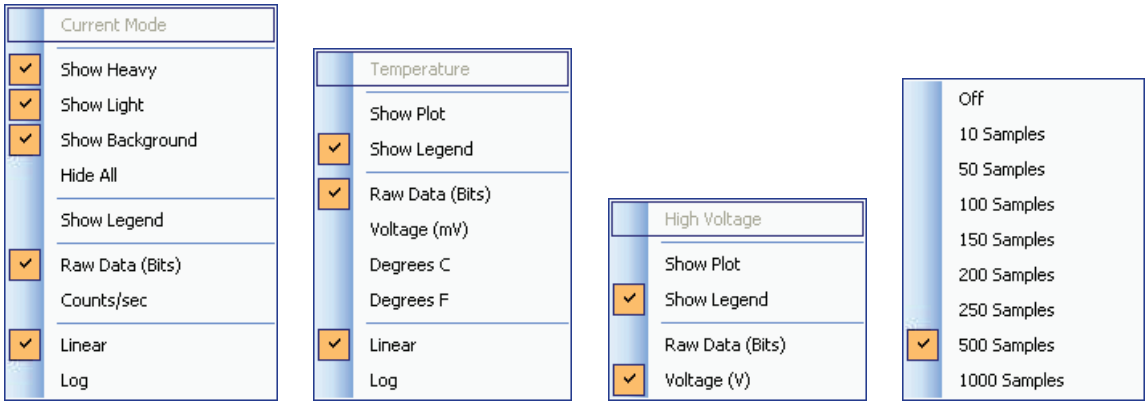
```
data (cont.)
├── BoardA{1,i}-Board{1,i} (cont.)
│   ├── PulseOverflow
│   └── PulseUnderflow
├── RawData(i,1:10608)
├── StartTime(1,i)
│   ├── bits
│   ├── datenum
│   └── datestr
├── EndTime(1,i)
│   ├── bits
│   ├── datenum
│   └── datestr
├── ReactorOn(1,i)
├── ReactorPulse(1,i)
├── ACGood(1,i)
├── SampleRate(1,i)
│   ├── bits
│   └── SPS
├── HVSetpoint(1,i)
│   ├── bits
│   └── value
├── LLDSetpoint(1,i)
│   ├── bits
│   └── value
├── ADCSpeed(1,i)
├── MSVResolution(1,i)
├── CurrentResolution(1,i)
├── TemperatureResolution(1,i)
└── MotherboardTemperature(1,i)
```

The "Plots" menu (Figure 6.36a) gives several controls to the user for configuring what data is shown and how it is displayed in the data section of the main screen (Figure 6.23). The top section of this menu has a sub menu for each of the plots: pulse mode (Figure 6.36b), MSV mode (Figure 6.36c), current mode (Figure 6.37a), temperature (Figure 6.37b), and high voltage (Figure 6.37c). The pulse, MSV, current, temperature, and high voltage sub menus all have the same basic functionality. The top section of the sub



(a) Plots Menu (b) Pulse Mode (c) MSV Mode

Figure 6.36: The User Interface Application's (a) "Plots" menu and the (b) pulse mode and (c) MSV mode plot settings sub menus.



(a) Current Mode (b) Temperature (c) High Voltage (d) Data Averaging

Figure 6.37: The User Interface Application's (a) current mode, (b) temperature, (c) high voltage, and (d) the data averaging plot settings sub menus.

menus lets the user select which plots are shown. Pulse, MSV, and current mode plots allow viewing of only heavy, light, or background detectors, along with any combination of these. If none of the "Show" options are selected, the plot will be removed from the data

section of the main screen and the remaining plots will be rescaled to fill the remaining area. The plots have the capability of showing a legend of all active data lines, but because this legend becomes impractical with the large amount of data available, the sub menu has the option to turn on or off each plot's legend. The next section of the sub menus allows the user to view either the raw data or the scaled data using the calibration coefficients from Figure 6.34. The pulse, MSV, and current mode plots are calibrated to show counts per second while the high voltage is calibrated to show voltage. The temperature plot is calibrated to show voltage in mV and utilizes the J-type thermocouple tables for conversion to either °C or °F [28]. In addition to these settings, all but the high voltage plots allow the users to choose between linear and log plotting. The absolute value of any negative data values are used during the log plotting mode so all positive and negative data is visible.

The next section of the "Plots" menu adjusts how much data is shown based on time. The "Auto Scale Time" option shows all of the available data, while the "Strip Chart Time" shows the last x minutes, which is user-definable through a dialog box that appears when this option is clicked. The plots also allow the user to zoom in on time periods of interest by clicking and drawing a box in one of the plots, which activates the "Zoomed" option and every visible plot will be zoomed in to show the same time sample of data. Clicking on either other option will exit the zoomed mode. "Averaging" opens the sub menu shown in Figure 6.37d, which has preset options for the number of samples to be used in a moving average. This set of options allows the user to reduce any system noise. Likewise, the "Ignore Data At Limits" will eliminate from the plot or moving average any raw data that is equivalent to the each channel's limits.

"Use Strings" and "Use Boards" affects each plot's legend by respectively showing which detector string is associated with each plot line or showing which daughter board channel is associated with each plot line. The associations are made through the "Board-String Connections" window of Figure 6.29. There are currently no options to take any "Box" connections into account.

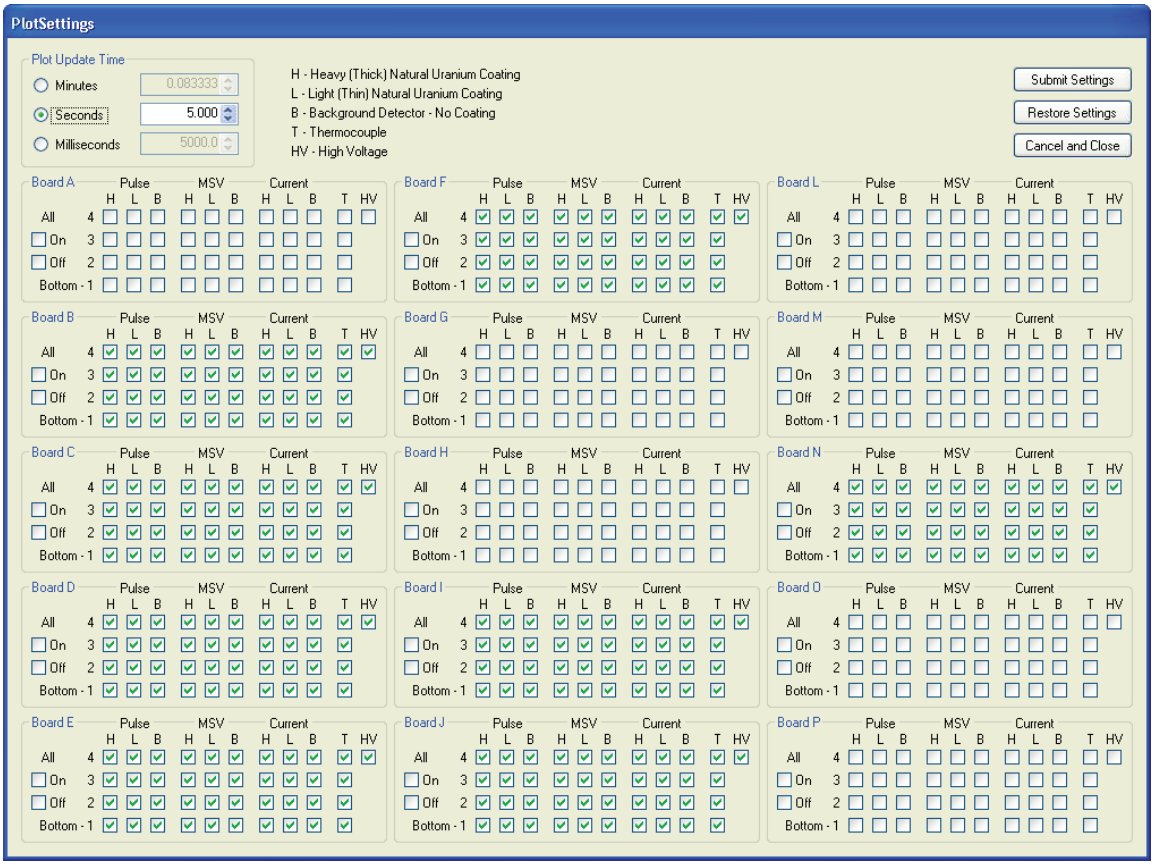


Figure 6.38: The plot settings control window for the User Interface Application.

The last option of the “Plots” menu allows the user to display the time data based on coordinated universal time (UTC) or based on the computer’s local time zone setting. UTC, sometimes referred to Greenwich mean time (GMT), is used as the basis for all of the data’s time stamps and for file generation. Conversion to the local time is only done for display purposes and does not affect the stored data.

The “Settings” option at the top of the “Plots” menu (Figure 6.36a) opens the window shown in Figure 6.38. This window shows each data channel of each daughter board and allows the user to select which channels are shown in the plots and which ones are not. The “Show” options of the “Plots” menu then enables or disables only those channels enabled in the “Settings” window. This window also sets the update rate of the plots. Due to the significant number of calculations that must be computed in order to update the plots, it is

Figure 6.39: The file replay control bar for the User Interface Application.

recommended that a minimum update rate of 5 seconds is utilized. Faster rates can be used, but only with a reduced data set.

When in the "Live Data" mode of operation, any changes to the "Settings" or "Plots" menu options will activate the "Send Settings" or "Upload Settings" buttons in Figures 6.27 and 6.28. Making changes to the "UPS," "High Voltage," "LLD Threshold," or "Advanced" options from the "Settings" menu (Figure 6.24) requires uploading of the changes to the MPFD^3-T 3D System in order to implement them.

Other than the "Live Data" mode of operations, the "File" menu also allows the user to replay files through the "Replay File" option for review of prior reactor operations. This mode adds a file replay control bar to the main menu bar (Figure 6.39) in order to adjust the rate at which the data is processed. The play button (►) causes the data to be replayed in real time, in other words, it takes one second to display one second of data. The double speed button (►►) causes the data to be replayed at twice real time so it takes one second to display two seconds of data. Options are also available for 5 times (►►×5), 25 times (►►×25), and 100 times (►►×100) replay rates. The user may also pause (❙❙) or stop (■) the replay.

Similar to the "Replay File" mode, there is a "Convert File" mode that allows replaying of the data but simultaneously generates a ".csv" file similar to that of the "Collect Calibration Data Stream" operation. The only difference is that there is no reactor data available for inclusion in the converted file.

Last but not least is the memory management that takes place behind the interface. Collecting 1327 bytes of raw data nine times a second adds up to a megabyte of data every 88 seconds and a gigabyte every 25 hours. Plotting this data takes considerably more memory

because not only must the raw data be stored, but the converted data arrays and the plot data arrays must also be stored in RAM. A majority of the arrays utilized are of a first-in, first-out (FIFO) type array. As data is added either through the "Live Data" or "Replay File" modes, the new data can simply be added to the arrays, which are designed to self-expand when their allotted space has been used. This is a dangerous method if left unchecked, because the arrays could quickly expand to use all of the available system memory resources. Therefore, the UIA monitors the available RAM, and if it drops below 100 megabytes, the UIA dequeues or removes array elements until at least 100 megabytes are available. With this capability, the MPFD^3-T 3D System and the User Interface Application is able to monitor reactor operations for days on end without interruption.

6.3 Trouble Shooting

As with any software, there will always be problems. The MPFD^3-T 3D System and the User Interface Application are highly complex and errors are sure to be encountered. Years of testing and experiments have allowed for the identification and solutions to most errors, but there are sure to be more. This section describes some errors that might be encountered from past experience, but which so far have not been solved.

6.3.1 LCD Screen

If the LCD Screen on the MPFD^3-T 3D System fails to start, it may be reset by holding the right arrow key for no more than 2 seconds while powering on the MPFD^3-T 3D System. If successful, the LCD memory should be reset and the proper operating codes reloaded. A subsequent reboot should bring the MPFD^3-T 3D System back to normal operation.

6.3.2 Stuck on Startup Screen

On occasion, the Ethernet module of the MPFD^3-T 3D System will not initialize properly on System startup. Usually, the System will get stuck showing the startup screen (Fig-

ure 6.1). The LAN and LINK LEDs will both be off or there will be no activity. Turn off the System and restart. This restart will normally solve the problem. If it does not, check the Ethernet switch located near the reactor console.

6.3.3 Missing Detectors

Sometimes a working detector may "disappear" from the MPFD^3-T 3D System's User Interface Application. This disappearance can happen when the reactor goes to high power levels that causes a certain detector to exceed the threshold of the safety relay, thus cutting the detector from the circuit. Tripping of a safety relay may also happen when moving fuel or experiments that accidently come into contact with the detector string probes, temporarily causing a wire to contact the probe's wall. Shut down the MPFD^3-T 3D System and restart in order to reset all of the relays. Restarting the MPFD^3-T 3D System, when possible, is recommended prior to collecting any critical data.

Chapter 7

Testing and Results

The MPFD technology has already been proven to work through the development of the detectors presented elsewhere [1]. Presented in this chapter is additional data not only to support the claims of long life and operability, but also demonstrate the MPFD^3-T 3D System's electronics and the system's ability to follow reactor operations.

7.1 Detector Dose

Because the MPFD^3-Ts were installed in the reactor core on December 18, 2008, the reactor has generated a logged 264,316 kWh as of February 23, 2012. According to the Kansas State University TRIGA Mark-II Nuclear Reactor training manual, the fast neutron flux (>10 keV) is 1.2×10^{13} n cm^{-2} s^{-1} in the central thimble and 3.5×10^{12} n cm^{-2} s^{-1} in the outer "F" ring of the reactor at 250 kW. Likewise, the respective slow neutron flux (<0.21 eV) is 1.0×10^{13} n cm^{-2} s^{-1} and 4.3×10^{12} n cm^{-2} s^{-1} while the gamma-ray radiation dose rate is 2.5×10^{4} rad/s and 1.5×10^{4} rad/s [16]. This means that the MPFD^3-Ts have experienced neutron fluences between 1.3×10^{19} to 4.6×10^{19} fast n cm^{-2} and 1.6×10^{19} to 3.8×10^{19} slow n cm^{-2} with a gamma-ray dose between 5.7×10^{10} to 9.5×10^{10} rad.

7.2 Detector Pulses

The auxiliary ports provided by the daughter boards, which are accessible on the back panel (Section §5.3.2), provide buffered access to the preamplifier output of each detector (Section §5.1.1). These ports allow connection of external nuclear instrumentation modules (NIM) and multi-channel analyzers (MCA). Shown in Figure 7.1 is an oscilloscope screen capture from one of the detectors. The buffered output of the detector is connected to

Figure 7.1: Oscilloscope screen capture of a pulse signal from one of the MPFD detectors.

channel 1 (yellow trace) of the oscilloscope and to the input of a Canberra 2022 amplifier. The coarse gain was set to 10, while the fine gain was set to 0.3, providing a total gain of 3. A shaping time of 12 µs and a positive input polarity were used. The unipolar output of the amplifier was connected to channel 2 (green trace) of the oscilloscope along with the input of an ORTEC Trump MCA. The pole-zero adjustment of the amplifier could not be adjusted in order to remove the undershoot seen after the pulse on channel 2 of the oscilloscope screen shot. The pulse should rise above the baseline (0 V) and return to the baseline without going back below if the pole-zero adjustment is set correctly to compensate the preamplifier. Unfortunately, the range of this particular 2022 amplifier's pole-zero could not match the requirements of the preamplifier.

The second oscilloscope screen capture shown in Figure 7.2 is of a pulse that has a large

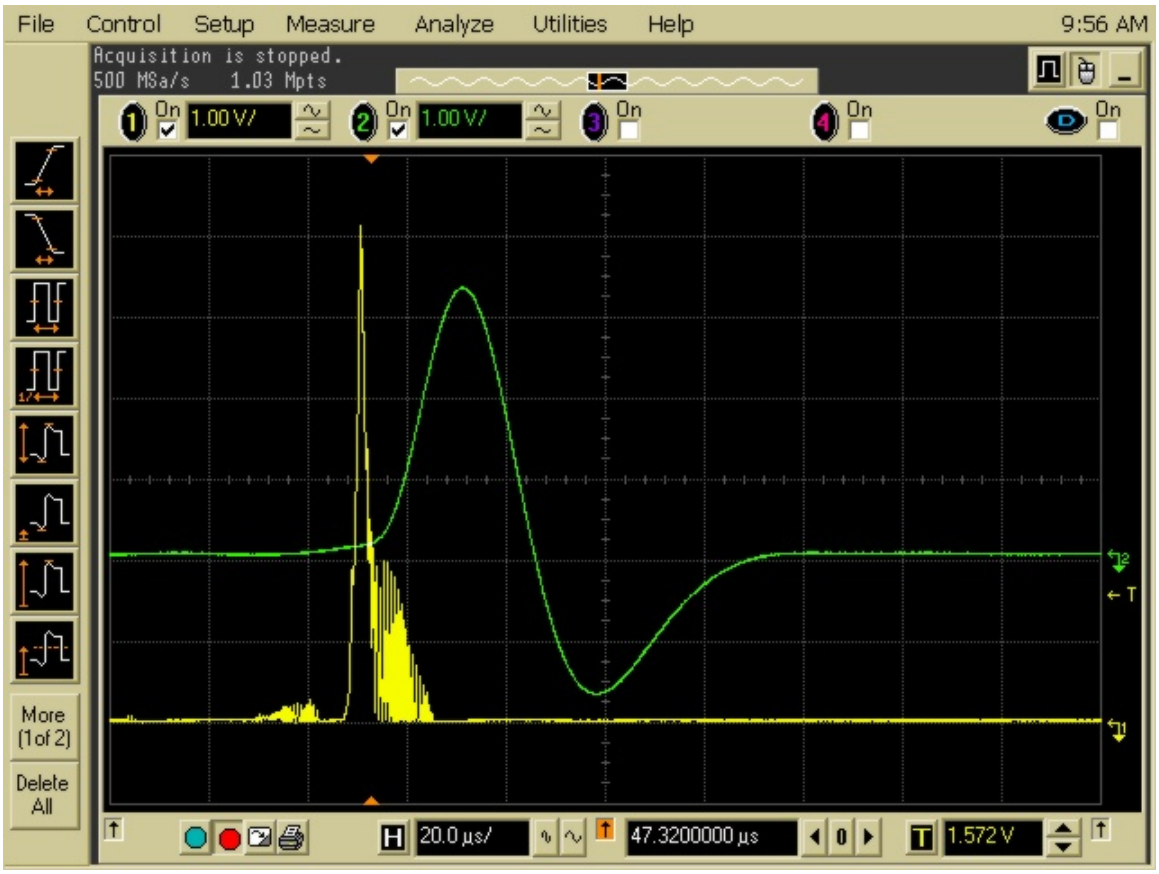


Figure 7.2: Oscilloscope screen capture of a noisy pulse signal from one of the MPFD detectors.

noise component in the tail of the pulse (yellow trace). This noise could trigger multiple counts to be recorded by the pulse mode counter circuitry. As described in Section §5.1.1, a monostable multivibrator resides between the Schmitt trigger and the counters. The Schmitt trigger generates a digital pulse when the analog signal rises above the lower level discriminator (LLD) voltage. The monostable multivibrator is triggered on the rising edge of the Schmitt trigger, but does not return to the off state until the Schmitt trigger's pulse remains in the off state for approximately 39 µs. Therefore, even though the pulse in Figure 7.2 has significant jitter or noise in its tail, only a single count would be registered in the counter.

7.3 Detector Spectra

The preamplifier output provided by the auxiliary ports of the MPFD^3-T 3D System was connected to a Canberra 2022 amplifier and then an ORTEC Trump MCA, as described in Section §7.2. The simulation results in Figure 7.3 shows the expected pulse height spectra based on the analysis of Section §3.1 for different thicknesses of natural uranium coatings. Based on the characterization of the ASCEFI system (Section §4.1.2), the neutron reactive coating was approximately 4 to 5 µm thick for the heavy natural uranium-coated detector, while the background detector should have no coating. If the heavy-coated detector was inadvertently wired to a light natural uranium-coated detector instead, the coating would be approximately 1 to 3 µm thick. Shown in Figure 7.4 is the collected spectrum from a heavy-coated detector and a background detector. Comparison of this figure to the simulation indicates that the coating is on the lighter side due to the rounded bulge on the falling edge,

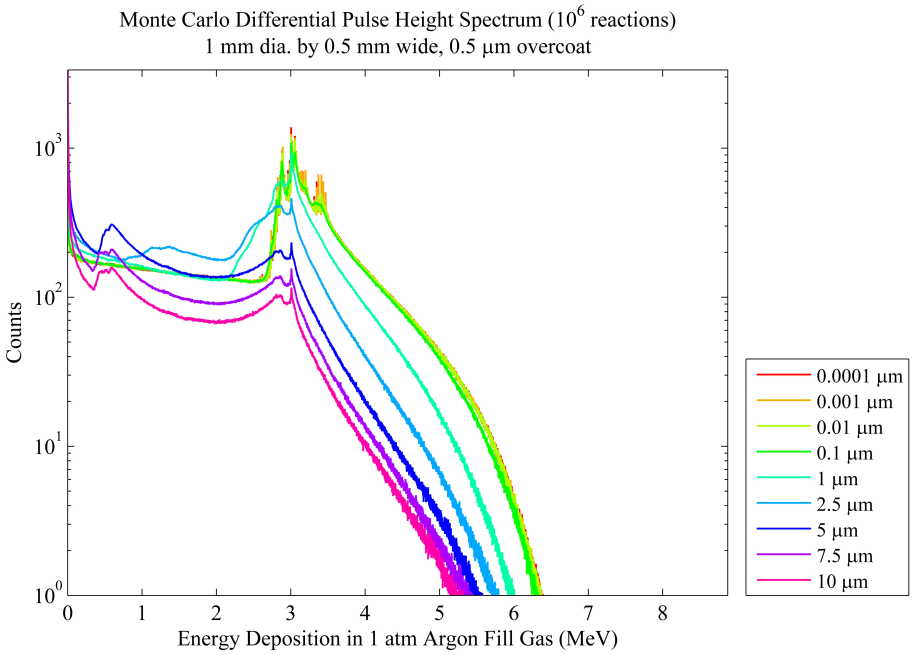


Figure 7.3: Differential pulse height spectra for various natural uranium neutron reactive coating thicknesses with a 0.5 µm platinum overcoat layer per 10^6 events over 8192 channels in an MPFD^3-T sized chamber.

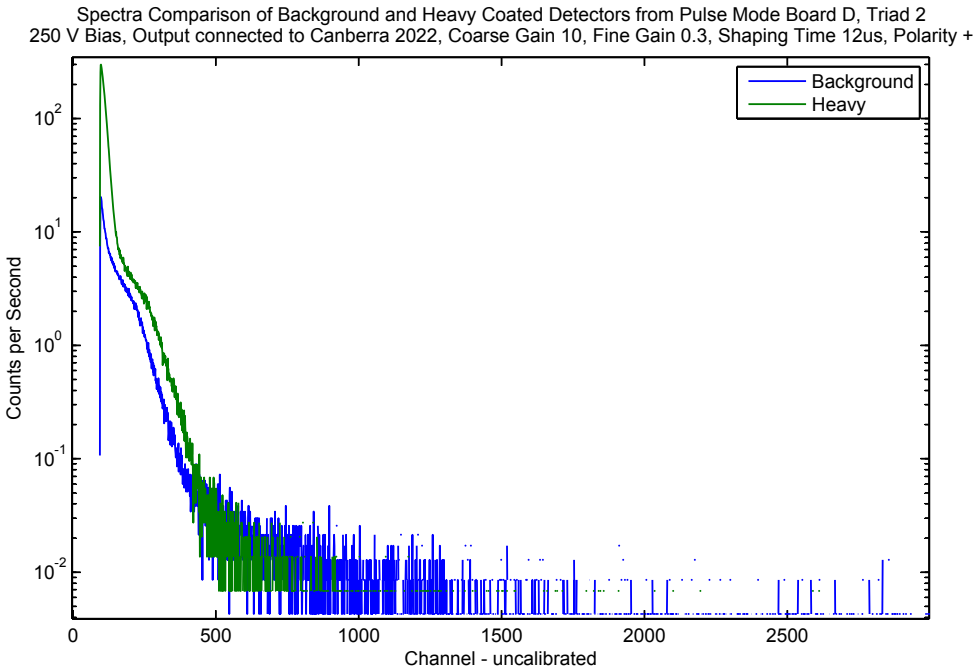


Figure 7.4: Logarithmic plot of spectra collected from a heavy coated and a background MPFD.

or higher energy side, of the spectra. Due to a high rate of counts, which was saturating the MCA, the LLD in the MCA had to be set to channel 94, which is why many of the low energy pulses from Figure 7.3 are not visible in Figure 7.4.

While the background detector's spectrum looks like a significant portion of the heavy-coated detector's spectrum in Figure 7.4, the figure's y-axis is scaled logarithmically in order to exemplify its relation to Figure 7.3. Figure 7.5 is provided to show the linear relation of the two detectors, which indicates that the background detector's spectrum is more than a magnitude less than the heavy-coated detector's spectrum. Even so, it is expected that the background detector would record almost no counts. Comparison of the two spectra in Figure 7.4 indicates that, unless the background detector channel was actually wired to a light-coated detector, there is significant cross-talk between the two detectors. This cross-talk is possible because the two would share a common high voltage wire and can only be ruled out with additional data collection and analysis.

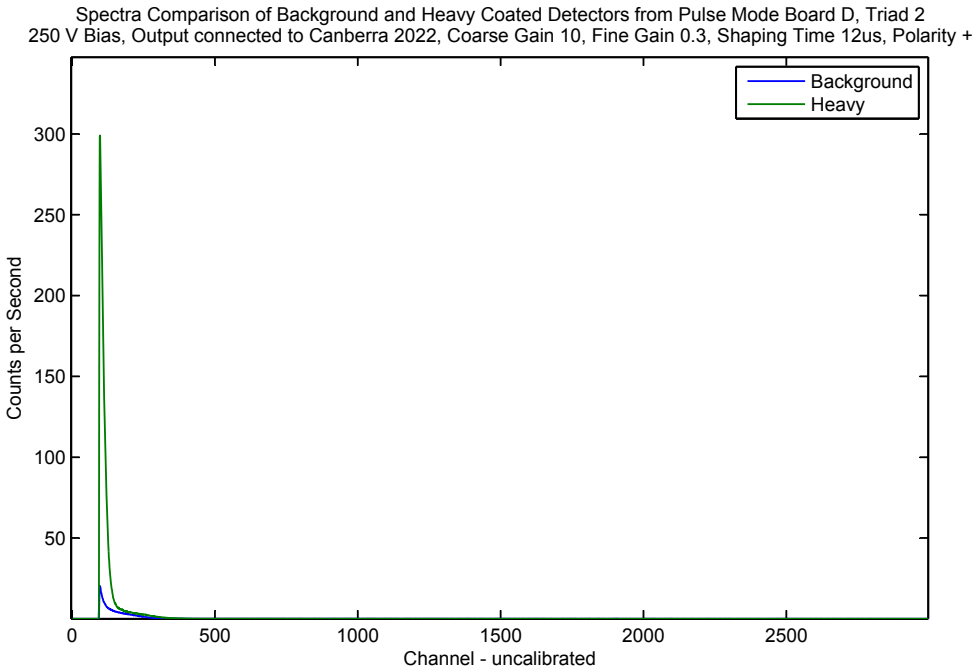


Figure 7.5: Linear plot of spectra collected from a heavy coated and a background MPFD.

7.4 Reactor Tracking

A true test of the MPFD^3-T 3D System and the User Interface Application is to operate the System during reactor transients and verify that the System responds accordingly. To do this, the reactor was operated at full power and then scrammed. The scram action causes all of the control rods to drop into the reactor in under a second. While this is the fastest process to shut down the reactor, the delayed neutrons emitted by the fission fragments slow the reactor to shut down process [7, 16]. The linear channel, one of the reactor's neutron monitoring instruments, is calibrated to display the power of the reactor. The channel has multiple power ranges for the display, specifically 1000 kW, 100 kW, 10 kW, 1 kW, 100 W, 10 W, 1 W, 100 mW, and 10 mW scales, with the display showing the percentage of the selected power range. The digital chart recorder samples the percentage output of the linear channel and records it to a file. Knowing that the reactor power level was in the 1000 kW range, the recorded data was converted and scaled to the actual reactor power as shown in

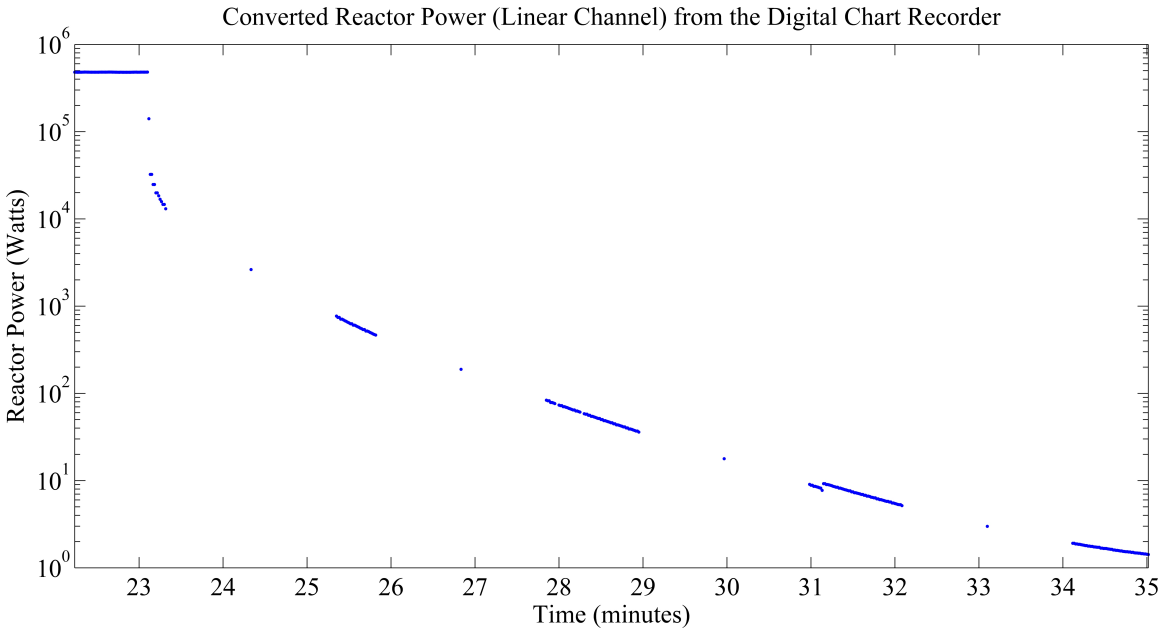


Figure 7.6: Converted log file for the linear channel during a reactor scram from 482 kW.

Figure 7.6. In this figure, the steady-state operation at 482 kW is interrupted by the scram at the displayed time of 23 minutes. The reactor first undergoes a prompt drop in neutrons, and thus, power level, as only the delayed neutrons, or 0.7% of the neutrons, continue to generate power. It is this delayed neutron fraction that limits the reactor power decrease by a factor of e, or approximately 2.718, every 80 seconds.

In comparison to the data recorded from the linear channel, the raw data from the MPFD^3-T 3D System is displayed through the User Interface Application for select detectors in Figure 7.7 with each line representing a single and separate MPFD. The pulse mode, MSV mode, and current mode data are shown, respectively, in the top, middle, and bottom graphs where the x axis displays the time and the y axis displays the measurement magnitude. Because the MPFD^3-T 3D System has not been calibrated, the data magnitude is shown in raw bits, or the value directly reported by the daughter board electronics. The MPFD^3-T 3D System was operated at 200 V and an LLD setting of 2.4 mV with the MPFD^3-T Strings G, H, I, F, B, E, N, and C connected respectively to the daughter boards B, C, D, E, F, I, J, and N. While the data appears extremely noisy in the pulse and MSV

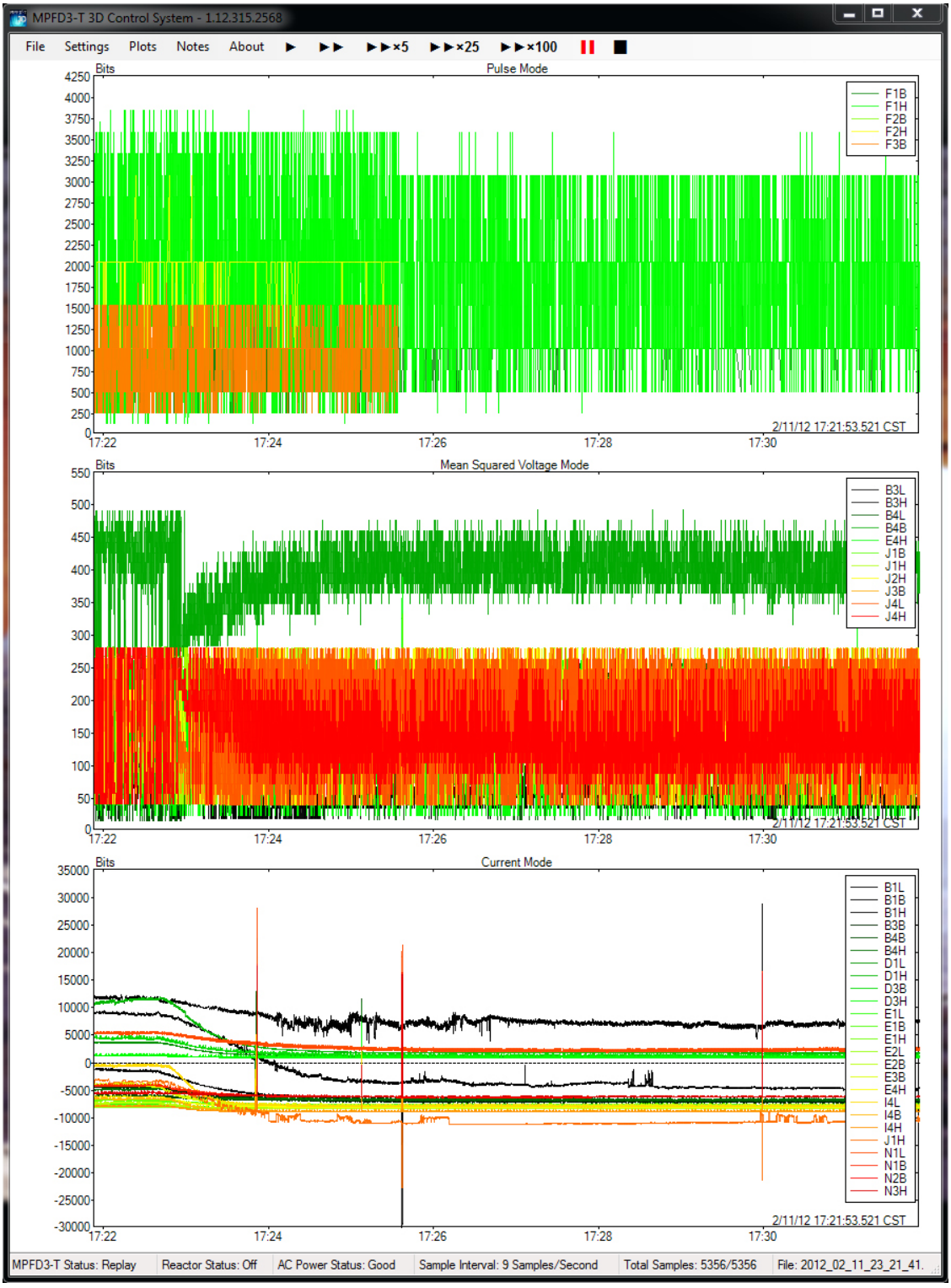


Figure 7.7: Select raw data from the MPFD^3-T 3D System replayed in the User Interface Application during a reactor scram from 482 kW.

modes, the shutdown of the reactor is visible in the current mode graph, with the decrease in power starting at approximately 17:23, corresponding to the decrease in reactor power shown in Figure 7.6. The prompt drop in neutrons is also evident in the MSV mode data shown in the middle plot of Figure 7.7. The noise spike visible in the current mode data to the left of 17:26 corresponds to the "Reactor Status" changing from "On" to "Off," which is caused by the reactor key being turned to the "Off" position on the reactor console. The spikes prior to this spike would be the control rod drives reaching their "Down" position and the control switches being released on the reactor console. It is unclear what would have caused the noise spike at 17:30, but it is expected that this spike is due to the reactor cooling being turned on. From the presence of these spikes, it is clear that the MPFD^3-T 3D System is not completely free of noise. However, from the in-core testing attempted during prototype development, this level of noise is a significant improvement. During these prototype tests, any movement of the control rods would saturate the detectors with noise and testing was impossible if any portion of the cooling system was operating.

While some structure is evident in the MSV data shown in Figure 7.7, application of a moving average is required to smooth the data and reduce the apparent noise. Figure 7.8 is the same data but with a 500 sample moving average applied. Trends are now clearly evident, not only for the current mode data, but also for the MSV and pulse mode data. As the current mode data begins to decrease, the MSV data starts to increase because the RMS converter's input signal is AC-coupled. Because of a reversed polarity connection to the ADC, the signal should actually appear to decrease. The few that do increase is most likely due to a poor connection to the reference voltage on either the ADC or the RMS converter, which will be discussed in Chapter 8. The MSV data returns to its baseline value as the pulses become less frequent in relation to the decrease in reactor power. The further decrease is evident in the pulse mode data as the detectors come out of saturation and then begin to track the reactor.

Data was also collected at different steady-state reactor power levels using the "Save

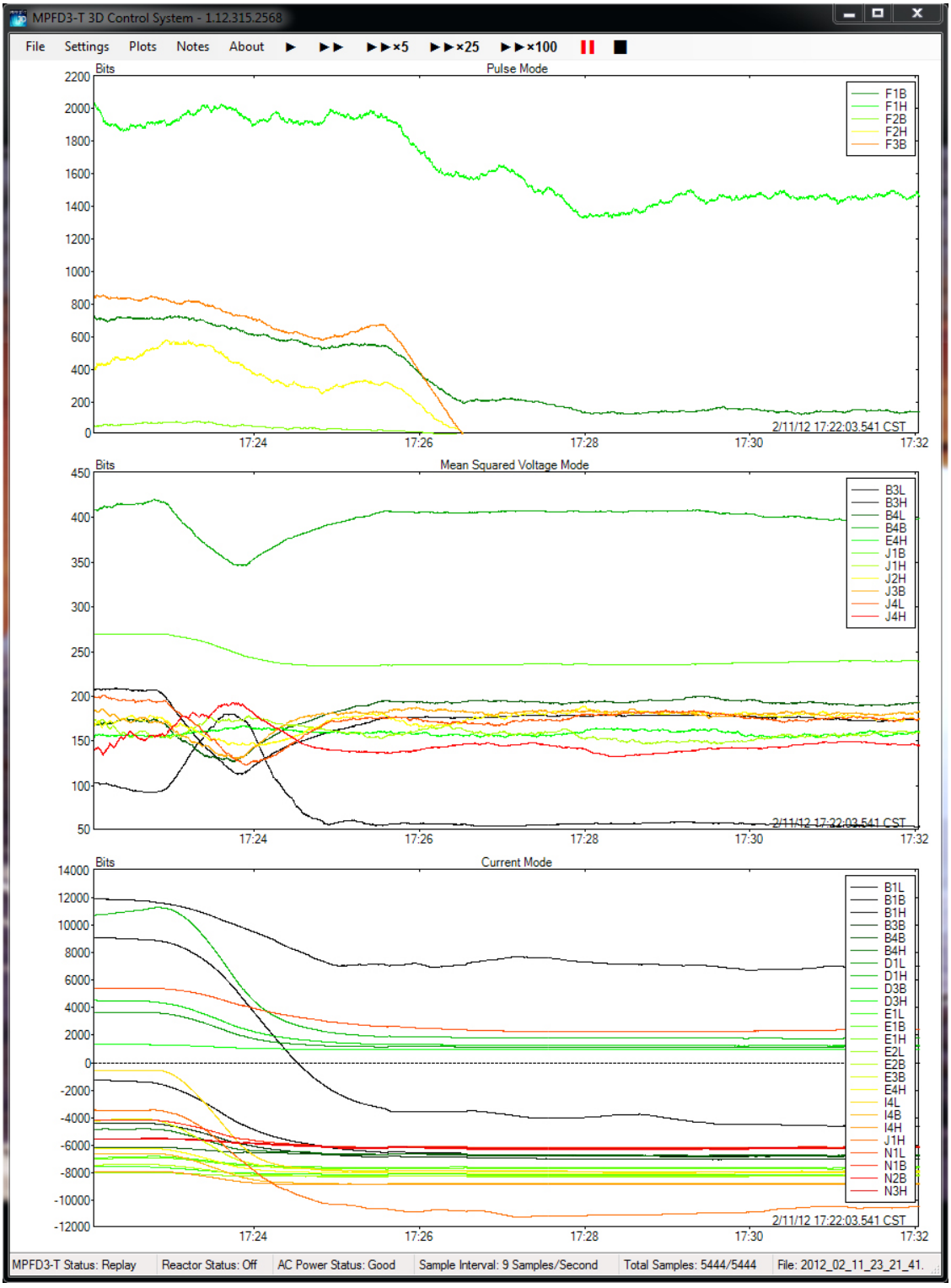


Figure 7.8: Select raw data from the MPFD^3-T 3D System replayed in the User Interface Application during a reactor scram from 482 kW.

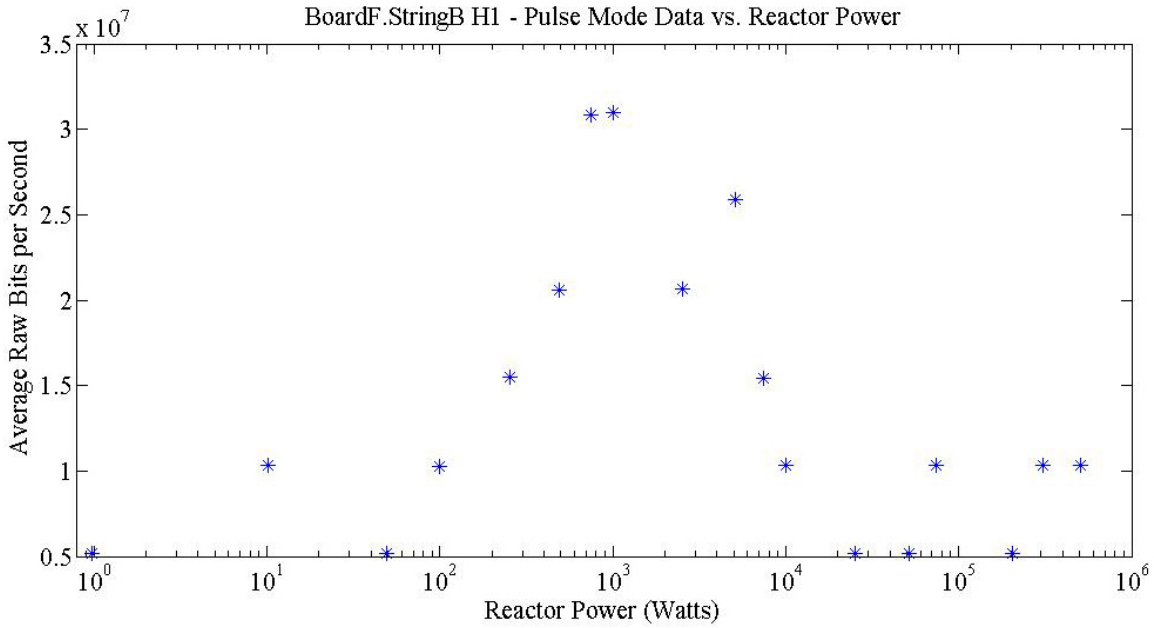


Figure 7.9: Pulse mode data for different reactor power levels when the detector was operated at 250 V with an LLD setting of 100 mV.

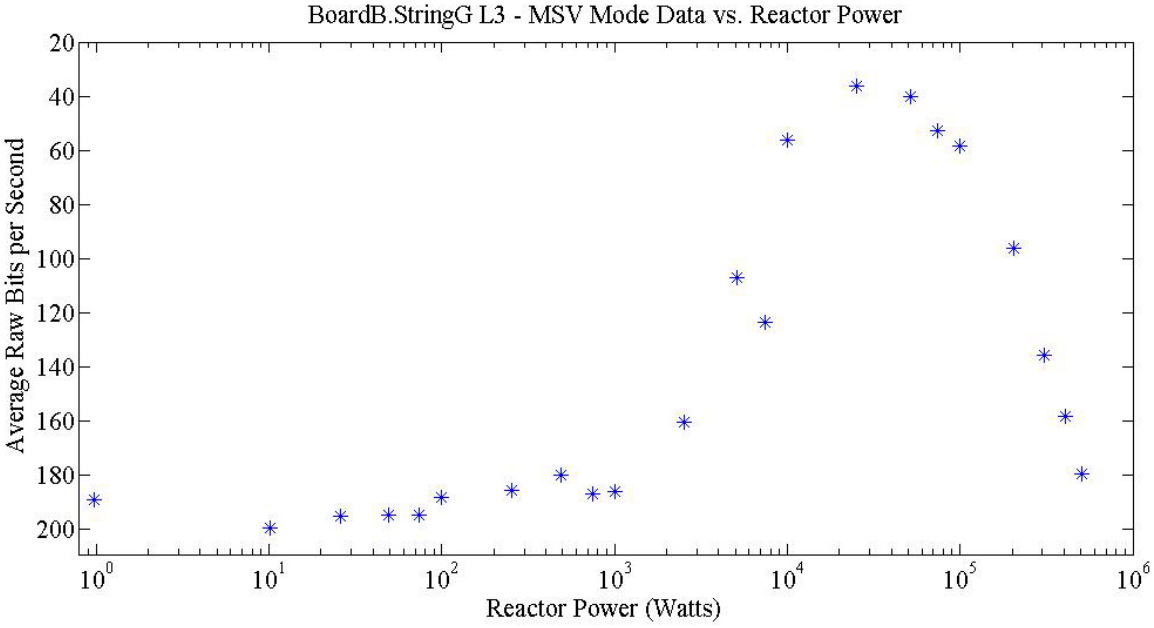


Figure 7.10: MSV mode data for different reactor power levels when the detector was operated at 250 V with an LLD setting of 100 mV. The y-scale has been reversed since the ADC is operated in reverse polarity.

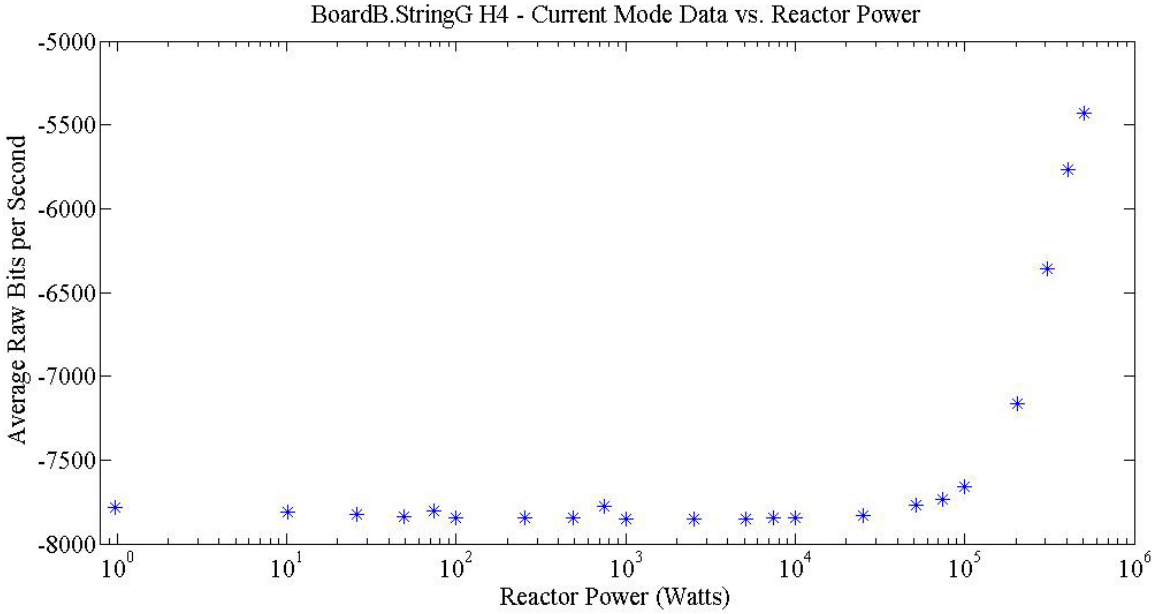


Figure 7.11: Current mode data for different reactor power levels when the detector was operated at 250 V with an LLD setting of 100 mV.

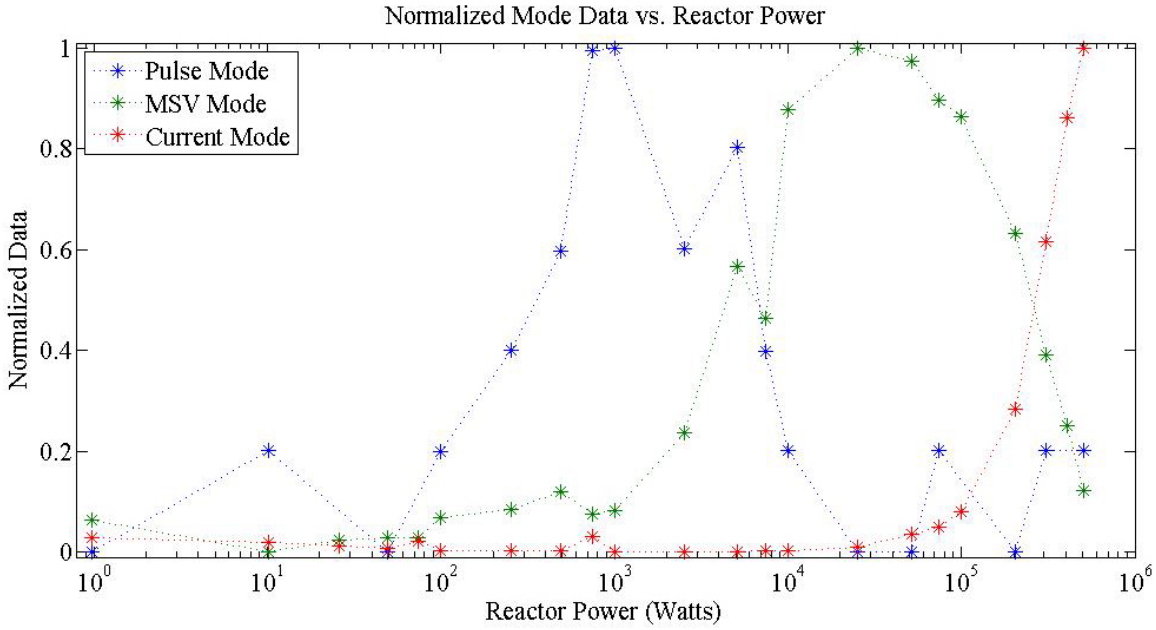


Figure 7.12: Current mode data for different reactor power levels when the detector was operated at 250 V with an LLD setting of 100 mV.

Calibration Stream" described in Section §6.2. This data was converted and compiled using MATLAB. Representative samples of the compiled data are shown for the pulse, MSV, and current modes in Figures 7.9, 7.10 and 7.11. For this set of data, the detectors were operated at 250 V with an LLD setting of 100 mV. Three hundred data samples were collected during each "stream" with the average value being computed for the MSV and current mode plots, while the pulse mode plot is based on the average rate, or per second, value. Again, the units for these plots are in raw bits since the MPFD^3-T 3D System has not yet been calibrated. The three plots have been normalized and superimposed into the plot shown in Figure 7.12. From this plot, it is shown that the full power range is covered by the three modes of operation, pulse mode for the lower power levels, MSV mode for the intermediate power levels, and current mode for the high power levels. Additional work needs to be performed in order to optimize the high voltage and LLD settings, a topic which will be discussed in Chapter 8 along with suggestions on the calibration of the electronics.

Chapter 8

Future Work

The subject of this dissertation, the deployment of the MPFD^3-T 3D System, is just the beginning of this project. The MPFD^3-T 3D System and the User Interface Application were built with long-term use in mind. This chapter focuses on the future tasks that should be or could be performed in order to carry on the research.

8.1 Electronics Calibration

While it is difficult to calibrate the MPFD^3-T 3D System for the intrinsic efficiency of each detector, the MPFD^3-T 3D System can be calibrated so that any differences in the electronics are zeroed out. The easiest method of doing this is to place a pulser on each channel. If preset pulser rates are used, the pulse mode channel can be tested against these rates for compliance. While slight deviation from the pulser rate on any channel is not expected, especially at lower pulser rates, it would not be unexpected to find channels that either do not record any counts or channels that record counts when no pulser is connected directly to that channel, in other words, cross talk. Each daughter board is made up of 1272 components, all of which were soldered onto the board by hand. Poor solder joints or broken surface mount resistors or capacitors could render complete channels useless. Therefore, problems somewhere on each board are not surprising.

The pulser can also be used for testing and calibration of the mean-squared voltage (MSV) mode and current mode electronics by increasing the pulser rate. The use of a pulser will also allow correlation of the MSV and current data to counts per second so long as the pulser is set to an equivalent mean detector pulse shape. Experimental validation will be necessary in order to verify this correlation since each detector may be slightly different.

To assist with design and construction of a pulser, the schematic shown in Figure D.1 may be useful because it provides the D-SUB connector pin assignments for connection between the daughter board and the detector string. However, Figure 8.1 is shown here because it provides additional information about the D-SUB connector, specifically, the standard wire color-code configuration and the relation of the old U, T, B (uranium, thorium, background) and the new H, L, B (heavy, light, background) detector identification (Section §4.1.2). Each cell of Figure 8.1 represents one pin of the D-SUB connector, with the number in the upper left corner of the cell indicating the pin number. Depending on the orientation of the D-SUB connector, the pins may appear backwards so two sets of pins are provided for convenience. The upper right corner lists the old or original detector designation, while the next line lists the new designation associated with the daughter board construction. The bottom three lines are used to identify which wire should be connected to this pin based on the solid color of the wire and the color of any stripes on the wire.

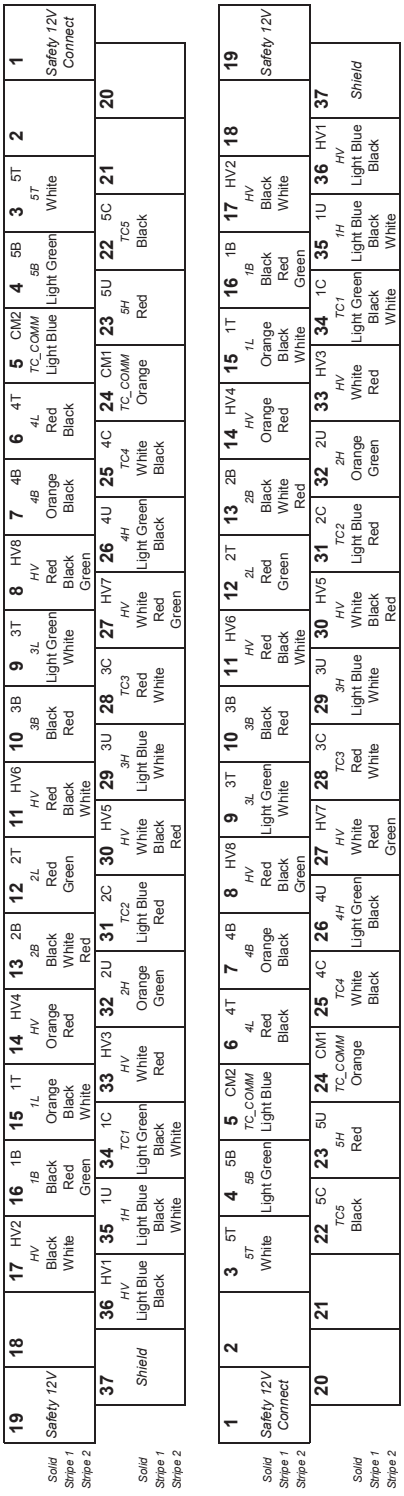


Figure 8.1: Cable pin assignment for 37-pin D-SUB connector.

With the information about the D-SUB connector, the pulser may be included as part of a test board which includes high voltage test points for calibration of the high voltage source. It can also contain J-type thermocouples for calibration of the thermocouple circuits; however, care must be taken to account for any additional thermal junctions that may exist in the actual wiring. In order to collect data from the system, the "Safety 12V Connect" and "Safety 12V" on pins 1 and 19 must be tied together. Connection of these pins will also allow the high voltage to energize, so care must be observed.

8.2 Detector Calibration

Once the electronics have been calibrated, any differences between the detector responses must be due to the detectors, which is especially true if utilizing the "Boxes" available in the calibration window shown in Figure 6.35 and described in Section §6.2. The calibration window also records data concerning not only the reactor's power level, but also the control rod positions and the water temperature. This information may be utilized as part of an MCNP (Monte Carlo Neutral Particles) simulation of the reactor to find the expected neutron flux at each detector's location.

8.3 Suggestions for Future Systems

As with any project, new information is learned as the project progresses. Described in this section are several of the lessons learned after the hardware was already built. If the opportunity to build this system once again became available, these changes would be recommended for the new system.

8.3.1 The MPFD^3-T String Design

While the design of the MPFD^3-T is good for research, it may only be necessary to include one or two detectors at each position for future requirements. The thermocouple may also be removed, depending on the project's requirements. While this redesign could allow

for a more complete vertical coverage of detectors by allowing additional positions to be populated, the main reason to implement any design changes would be to provide isolated high voltage connections to each detector instead of using a common high voltage bus. Utilization of isolated high voltage wires would minimize cross talk and would reduce noise currently present in the MPFD^3-T 3D System.

8.3.2 Daughter Board Preamp

The current preamp design (Section §5.1.1) has one significant vulnerability that was not recognized until the boards had already been produced. The inputs to the INA103KU (IC503 in Figure 5.7) are susceptible to damage during fast high-voltage transients that may occur if the system is not shut down correctly. This problem can be solved by adding a bidirectional transient voltage suppressor (TVS), or two back-to-back zener diodes, across R512 and R520 in Figure 5.7. The TVS will prevent the voltage from exceeding the part's defined breakdown voltage and thus will prevent damage to the INA103KU.

8.3.3 Counters

The electronics were designed around the ADS1110 ADC's maximum sample rate specification of 240 samples per second. For the 16-bit counters, this rate would accommodate up to 15,728,640 counts per second. However, due to limitations encountered in the data transfer rate (Section §8.3.4), the maximum count rate is limited to 589,824 counts per second when the MPFD^3-T 3D System is operated at a rate of 9 samples per second. This should be sufficient. However, there is no provision built into the system if the counter should role over the 16-bit limit in a single sample cycle. The ripple carry output (RCO) on pin 9 of IC512 (Figure 5.7) can be used in conjunction with the "PULSE_ENABLE" data line on pin 12 of IC508 to disable any further counting once the counter had been filled. Unfortunately, this implementation would result in the counter data showing a value of zero instead of a full 65,535, so care must be taken when using the data to account for this overflow.

8.3.4 Data Transfer Rate

The data transfer rate is the primary limiting issue with the current MPFD^3-T 3D System design. Due to the large number of I^2C components on the daughter boards, there is no practical way to pull the data from all of the components simultaneously using the microcontroller, except through software polling. The transfer rate is limited by the execution rate of the software, which requires multiple clock cycles for executing each command, thereby reducing the effective speed of the microcontroller to be significantly less than its 40 MHz oscillator frequency. One option to overcome this is to use a faster processor. Another option, which may be the better option, would be to use a field-programmable gate array (FPGA) or a soft-core FPGA. The FPGA has the ability to perform operations similar to a hardware-based I^2C implementation but maintains the flexibility to combine the data into a single packet for transfer to the motherboard. A microcontroller may or may not be necessary depending on how the FPGA is programmed.

The other point of congestion is the mother board/daughter board interface. During the design phase of the boards, the decision was made to utilize three data lines per daughter board with two clock lines. The intention was to use the clock lines to synchronize the mother board and the daughter boards in a manner similar to the I^2C protocol, but with three times the data flow. Unfortunately, it didn't work out so well, and even with the software optimized as much as possible, the maximum achievable data rate is only 9 samples per second instead of the intended 240 samples per second. One potential solution would be to use an FPGA on the mother board in a manner similar to the daughter board solution. The FPGA would implement a high frequency I^2C bus to each daughter board with a hardware-based I^2C bus to the mother board microcontroller or processor, which would operate at a frequency greater than the current 25 MHz.

If the MPFD^3-T 3D System is ever rebuilt, both of these interfaces need to be researched and redesigned in order to achieve the MPFD^3-T 3D System's intended capabilities. I^2C may also not be the optimal data bus with a high frequency transfer rate of only 400 kHz,

whereas SPI (serial peripheral interface) bus can operate up to 70 MHz. The capabilities and limitations of these other data busses also needs to be taken into consideration.

8.3.5 Cable Pins

The cables are attached to the D-SUB connectors using a pin crimped to each wire. The instructions for the pins and the crimper stated that the wire should not be stripped prior to insertion into the pin. The process of crimping the pin would pierce the insulation and make contact with the conductor. Unfortunately, this was not the case, which was discovered once nearly all of the pins had been crimped and populated. A special test fixture (Figure 8.2) was made to check the conductivity of each wire and crimped connector. The test fixture contains a row of switches, one switch and two LEDs for each data line. When switched on, the selected data line's red LED would turn on, and if the conductivity was good, the green LED would turn on. While all of the wires on all of the cables were tested, and problem wires were fixed, the connections still seemed to be temperamental. The remaining

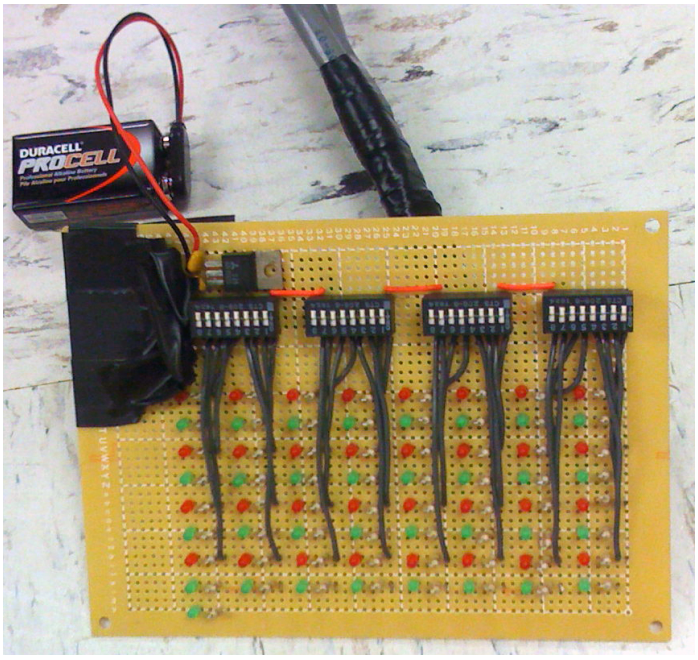


Figure 8.2: Cable tester for checking conductivity of each wire and connector.

problems seemed to lie with the test fixture and were difficult to reproduce consistently. However, the pins cannot be ruled out for disabled or intermittently available detectors or for excessive noise present on all of the detectors in a string due to noise in a high voltage line.

8.3.6 Fissionable Coatings

ASCEFI was more than just an electroplating system; it was an education. This system was used as a teaching platform in order to gain experience with machining, chemistry, electronics, and programming. Unfortunately, dealing with programming and electronics problems, instead of optimizing the coating process, became a time-consuming necessity. While the time and effort spent on ASCEFI proved beneficial in the design and fabrication of the mother and daughter boards for the MPFD^3-T 3D System, it is now clear that there are commercial systems that would have done the electroplating job just as well if not better than ASCEFI. One such controller would be the MPS series produced by American Plating Power (http://www.usplating.com/products_controller_mps.php).

A far better method of producing the fissionable coatings would be to evaporate or sputter the fissionable material onto the electrodes. This method would produce even coatings and would eliminate many of the variables that affect the coating quality when using electroplating techniques. The coatings could even be tailored with different ratios of uranium, thorium, and highly enriched uranium by using multiple targets. This mixture would allow the detectors to achieve the count rate and lifetime needed for a particular application. Unfortunately, such an operation would require the sacrificial use of an evaporator or sputtering machine, because the equipment would become contaminated with radioactivity. Care must also be taken so that the vacuum system does not cause radioactive contamination to the rest of the room or environment because of severe health issues such as the inhalation of uranium.

8.4 Experiments

There are numerous experiments that can be performed with the MPFD^3-T 3D System. However, there are a few that should be run in order to characterize the system once the calibration process in Section §8.1 has been completed. One test is to check for the optimal operating high voltage setting. Too low of a setting will cause incomplete charge collection, while too high of a setting will induce discharges or sparking inside the detector. Tests have shown that the detectors can operate over a wide voltage range with minor changes in performance [11]. The minimum operating voltage of the MPFD^3-T 3D System is 150 V, which should be greater than the minimum voltage required for complete charge collection. While the maximum operating voltage is 500 V, experience has shown that this will often induce enough current to trip many of the protective relays.

Another experiment that needs to be performed is to determine the proper setpoint for the lower level discriminator or LLD. Most fission chambers need to be calibrated to remove the intrinsic beta contribution from the signal by adjusting the LLD. Beam port testing showed that prototype MPFDs would record no counts when the neutron field was removed by a cadmium sheet [11]. This low gamma-ray response is expected, based on the analysis of Section §3.2. However, this result was for a detector that was located outside of the reactor and with a preamp located close to the detector. For the MPFD^3-T 3D System, which has the detectors located throughout the reactor, along with 40' of wire between the detectors and the preamplifiers, the noise will be significantly greater than in the beam port test.

8.5 Industry

The MPFD can fill a significant void in the nuclear industry: real-time, in-core, long-life neutron detectors. MPFDs can fill all of these needs, but first they must be produced commercially and accepted by the industry. Commercial production requires not only the

improvements mentioned in this chapter, but also an understanding and compliance with the regulations and requirements of the nuclear reactor industry.

Chapter 9

Conclusions and Outlook

The goal of deploying a 3D array of MPFD^3-Ts inside the Kansas State University TRIGA Mark-II Nuclear Reactor has been successfully accomplished. Building on the initial work of creating the MPFD^3-T design, refinements were made to improve the construction process and allow for mass production of the detectors. Simulations of the detectors were created and examined in order to gain a better understanding of how the MPFD operated and what should be expected. This information was used during the design of the custom circuitry required to extract and collect data from all of the detectors in pulse, mean-square-voltage (MSV), and current modes. Not only was this data collected for all of the detectors, but data from all three modes of operation along with all of the data from all of the detectors were collected simultaneously. While the MPFD^3-T 3D System did not reach the full sample rate capabilities of the MSV and current mode analog-to-digital converters (ADC), the MPFD^3-T 3D System is capable of recording nine full data sets per second.

Deployment of the array of MPFD^3-T Strings led to many additional tasks that were not in the original scope of the project but still had to be solved. These tasks ranged from something that seemed as simple as constructing water-tight flux-probe tubes to something as complex as building an under-water drilling machine for restoration of the reactor core's flux-probe holes to their specified diameter. Perseverance through these trials and learning experiences eventually proved worthwhile, and the array of detectors was installed in the reactor core.

With the detectors and electronics designed, built and installed, the task switched to creating the software necessary for operation of the MPFD^3-T 3D System. The result was a cooperative set of programs for operating the mother board and daughter boards, along

with a remote User Interface Application, which works with the MPFD^3-T 3D System through an Ethernet connection. The User Interface Application allows users to collect and view the data in real-time, along with the capability to view prior data and convert the data for external analysis.

Testing has shown that the MPFD^3-T 3D System responds to reactor transients in real-time. While there are still many experiments that could be performed to calibrate or even improve the MPFD^3-T 3D System's design, the tests show that the goals of this project have been accomplished.

This new and evolutionary step in fission chamber design is now extended as the first real-time, in-core neutron flux monitoring system that utilizes a 3D array of detectors. The capabilities of this system and its future refinements could have a significant impact on the nuclear industry, especially for nuclear power reactors. The existing method of monitoring and controlling the sizable nuclear power reactor through the use of external or out-of-core neutron flux monitors will be greatly enhanced by the addition of in-core MPFDs. An array of MPFDs deployed throughout a nuclear power reactor would improve not only the overall picture of the reactor's power density during operations, but would also provide localized power density information which will allow higher efficiency and safer operations. Interest in the MPFDs has also been expressed by the research reactor community. Many of their experiments require real-time knowledge of the exact localized neutron flux during an experiment, a unique capability that could be provided by calibrated MPFDs. Therefore, work on this technology should continue, not only in the academic community, but also in the industrial community.

Bibliography

[1] M. F. Ohmes, "Micro-pocket fission detector (MPFD) development for the Kansas State University TRIGA Mark-II Nuclear Reactor," Master's thesis, Kansas State University, Manhattan, 2006.

[2] D. S. McGregor, J. K. Shultis, and P. M. Whaley, "Near-core and in-core neutron radiation monitors for real time neutron flux monitoring and reactor power level measurements," Kansas State University, solicitation no. DE-PS03-02SF22467, DOE NERI, 2002.

[3] M. F. Ohmes, D. S. McGregor, J. K. Shultis, P. M. Whaley, A. S. M. S. Ahmed, C. C. Bolinger, and T. C. Pinsent, "Development of micro-pocket fission detectors (MPFD) for near-core and in-core neutron flux monitoring," in *Proceedings of the SPIE*, vol. 5198, San Diego, January 20 2004, pp. 234–242.

[4] A. S. M. S. Ahmed, "Determining axial fuel-rod power density profiles from in-core neutron flux measurements," Master's thesis, Kansas State University, Manhattan, 2007.

[5] G. F. Knoll, *Radiation Detection and Measurement*, 3rd ed. New York: Wiley, 2000.

[6] Centronic LTD, Radiation Detectors. (2005, April) Fission chambers general data, ref. 313/06. [Online]. Available: http://www.centronic.co.uk/downloads/Fission_Chambers_General_Info.pdf

[7] J. K. Shultis and R. E. Faw, *Fundamentals of Nuclear Science and Engineering*. New York: Marcel Dekker, Inc., 2002.

[8] S. W.H. Todt, "Characteristics of self-powered neutron detectors used in power reactors," in *Proc. of a Specialists' Meeting on In-core Inst. and Reactor Core Assessment, NEA Nuclear Science Committee*, Mito-shi, Japan, October 1996.

[9] D. S. McGregor, J. K. Shultis, M. F. Ohmes, A. S. M. S. Ahmed, R. E. Ortiz, and K. Hoffert, "Micro-pocket fission detectors (MPFD) for near-core and in-core neutron flux monitoring," in *ANS 4th Topical Meeting NPIC & HMIT*, Columbus, Ohio, September 19-22 2004.

[10] D. S. McGregor, M. F. Ohmes, R. E. Ortiz, A. S. M. S. Ahmed, and J. K. Shultis, "Micro-pocket fission detectors (MPFD) for in-core neutron flux monitoring," *Nuclear Instruments & Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors, and Associated Equipment*, no. 554, pp. 494–499, December 2005.

[11] M. F. Ohmes, D. S. McGregor, J. K. Shultis, A. S. M. S. Ahmed, R. E. Ortiz, and R. W. Olsen, "Recent results and fabrication of micro-pocket fission detectors (MPFD)," in *Proceedings of the SPIE*, vol. 6319, San Diego, September 20 2006.

[12] M. F. Ohmes, A. S. M. S. Ahmed, R. E. Ortiz, J. K. Shultis, and D. S. McGregor, "Micro-pocket fission detector (MPFD) performance characteristics," in *Conference Record of the IEEE NSS/MIC*, no. N16-2, San Diego, 2006.

[13] M. F. Ohmes, J. K. Shultis, and D. S. McGregor, "3D real-time in-core neutron flux mapping with micro-pocket fission detectors (MPFD)," in *2007 IEEE Nuclear Science Symposium Conference Record*, vol. 3, no. N39-2, 2007, pp. 2060–2064.

[14] W. D. Allen, *Neutron Detection*. New York, NY: Philosophical Library, Inc., 1960.

[15] N. Tsoulfanidis, *Measurement and Detection of Radiation*, 2nd ed. Washington, DC: Taylor & Francis, 1995.

[16] Kansas State University TRIGA Mark II Nuclear Reactor Facility, *Training Manual*, September 19 2002.

[17] J. F. Ziegler, J. P. Biersack, and U. Littmark, *The Stopping and Range of Ions in Solids*. New York: Pergamon Press, 1985.

[18] J. F. Ziegler and J. P. Biersack, "SRIM-2003," computer program version 2003.26, SRIM.org, 2003.

[19] National Nuclear Data Center, Brookhaven National Laboratory. (2007, June 7) Decay radiation search. [Online]. Available: http://www.nndc.bnl.gov/nudat2/indx_dec.jsp

[20] J. F. Ziegler, M. D. Ziegler, and J. P. Biersack, "SRIM-2006," computer program version 2006.02, SRIM.org, 2006.

[21] T. W. Crane and M. P. Baker, "Neutron detectors," in *Passive Non-destructive Assay of Nuclear Materials*. NUREG/CR-5550, LA-UR-90-732, 1991, ch. 13.

[22] S. F. Biagi, "Accurate solution of the boltzmann transport equation," *Nuclear Instruments & Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors, and Associated Equipment*, no. 273, pp. 533–535, 1988.

[23] S. C. Brown, *Basic data of plasma physics, 1966*, 2nd ed. Cambridge, MA: The MIT Press, 1967.

[24] R. Johnsen, M. T. Leu, and M. A. Biondi, "Mobilities of mass-identified atomic ions in the noble gases," *Physical Review A*, vol. 8, no. 5, pp. 2557–2563, November 1973.

[25] M. Lourenço, J. Serra, M. Nunes, A. Vallêra, and C. Nieto de Castro, "Thin-film characterization for high-temperature applications," *International Journal of Thermophysics*, vol. 19, no. 4, pp. 1253–1265, 1998.

[26] T. Peng. (2007, January 26) Detect circles with various radii in grayscale image via Hough Transform. MATLAB Central. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=9168&objectType=FILE

[27] M. Capelli-Schellpfeffer, "Fair warning," *IEEE Industry Applications Magazine*, vol. 11, pp. 36–40, Jan/Feb 2005.

[28] M. C. Croarkin, W. F. Guthrie, G. W. Burns, M. Kaeser, G. F. Strouse, D. C. Ripple, and S. L. Young, "Temperature-electromotive force reference functions and tables for the letter-designated thermocouple types based on the ITS-90," National Institute of Standards and Technology, Monograph 175, 1993. [Online]. Available: http://srdata.nist.gov/its90/main/

[29] *Application Note 3684: How to Use the DS2482 I^2C 1-Wire Master*, Maxim, Dec 2008. [Online]. Available: http://www.maxim-ic.com/an3684

[30] N. Rajbharti, *AN833: The Microchip TCP/IP Stack*, Microchip Technology, Inc., 2008. [Online]. Available: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en011993

[31] F. Eady, *Networking and Internetworking with Microcontrollers*. Oxford: Elsevier Newnes, 2004.

[32] Pipelinecache. (2010, May 12) BuildVersionIncrement. [Online]. Available: http://autobuildversion.codeplex.com

[33] ComponentXtra. (2011, January 20) XYGraph. [Online]. Available: http://www.componentxtra.com/html/XYGraph.htm

[34] Cheeso and Jaans. (2010, February 9) DotNetZip. [Online]. Available: http://dotnetzip.codeplex.com

[35] Kansas State University TRIGA Mark II Nuclear Reactor Facility, "Accident analysis," in *Safety Analysis Report*, March 2006, ch. 13.

[36] Nuclear Regulatory Commission, "Title 10, Code of Federal Regulations, Part 20 (10CFR20)," June 29 2007.

Appendix A

Additional Analysis

A.1 In-Core Failure Analysis

In order to assure the safety of reactor personnel and the public it is necessary to analyze the effects of any fission fragments released during an MPFD^3-T structural failure. A similar analysis had already been completed for the Kansas State University TRIGA Mark-II Nuclear Reactor in its Safety Analysis Report (SAR) regarding the failure of a fuel element [35]. In the SAR, two different reactor histories were analyzed using the Oak Ridge National Laboratorys Isotope Generation code (ORIGEN) in order to find the fission product build up over a long time period and a short time period. The long time period analysis primarily looked at the build up of long lived radionuclides during a continuous 3.5 kW operation for 40 years, which is based on historical data. The short time period analysis was instead based on the short lived radionuclides generated in a short high-power operation and was simulated by 1250 kW operations for 8 hours a day for 5 straight days, or a continuous average power of 416.67 kW. Both of these simulations generated a radionuclide inventory in activity as curies for 1 tonne of ^{235}U at 1 watt, or Ci W^{-1} tonne^{-1} [35]. For an MPFD^3-T with a maximum neutron reactive coating thickness of 2 μm and a total contact coverage area of 1.98 mm^2 for each chamber, the total mass of uranium is approximately 151 μg. Since the ^{235}U concentration in the natural uranium contact is 0.711% and 93% for the enriched uranium contact, the total ^{235}U mass (m) is 71.6 μg per MPFD^3-T. As previously calculated, the fission rate per gram of natural uranium at 250 kW in the central thimble is approximately 1.60×10^{10} s^{-1} and 1.88×10^{12} s^{-1} for 93% enriched ^{235}U [1]. Linearly scaling these fission rates for 100% enriched ^{235}U yields a fission rate (F) of 8.09×10^{9} fissions s^{-1} g^{-1} kW^{-1}. For a maximum steady-state reactor power (P) of

1250 kW, the effective power (P_{eff}) produced in each MPFD^3-T is then calculated as

$$\begin{aligned} P_{eff} &= PFm \\ &= (1250\ kW)\left(8.09\frac{fissions}{s\cdot ng\cdot kW}\right)\left(33.2\frac{pJ}{fission}\right)(71.6\ \mu g) \\ &= 0.024\ \frac{J}{s} = 0.024\ W. \end{aligned} \tag{A.1}$$

The activity (A) of each nuclide generated in an MPFD^3-T is calculated as

$$A = A_{ORIGEN}\cdot P_{eff}\cdot m \tag{A.2}$$

where A_{ORIGEN} is the ORIGEN activity. ORIGEN identified 688 nuclides in the long run and 677 nuclides in the short run with non-zero activity levels [35]. These nuclides are split into four tables (Tables A.1 to A.4) based on whether they are a gas or particulate and whether they are listed specifically in 10CFR20 Appendix B or not. The total Annual Limit of Intake (ALI) ratio (R) for all of the fission products, gaseous and particulate, is defined as

$$R = \sum_{i=\text{All nuclides}} \frac{A_i}{ALI_i} \tag{A.3}$$

where A_i is the activity of fission product i and ALI_i is its associated ALI. This ratio should not exceed 1.0, which for the long run is calculated to be 5.16×10^{-8} and 2.22×10^{-5} for the short run. In the extremely improbable case that all 75 MPFD^3-Ts would fail simultaneously the R values would be multiplied by 75^2 since the mass is also used to determine the effective power. Even in this case the long run ratio is only 2.90×10^{-4} and the short run ratio is 1.25×10^{-1}.

While the definition of an ALI states that if the ALI has not been exceeded then the Derived Air Concentration (DAC) value will also not be exceeded, the DAC values have been calculated anyways for completeness. A DAC is defined as

> "the product of the concentration of radioactive material in air (expressed as a fraction or multiple of the derived air concentration for each radionuclide)

and the time of exposure to that radionuclide, in hours. A licensee may take 2,000 DAC-hours to represent one ALI, equivalent to a committed effective dose equivalent of 5 rems (0.05 Sv)" [36].

Since the radionuclides will undergo their natural rate of decay, the average air concentration (C_i) of each radionuclide over one year is calculated as

$$\bar{C}_i = \frac{A_{i,0}}{V_{Bay}} \frac{\left(1 - e^{-\lambda t}\right)}{\lambda t} \tag{A.4}$$

where t is the number of seconds in one year (31,557,600 s), λ is the decay rate of the radionuclide, V_{Bay} is the free air volume of the bay (144,000 ft^2 = 4.0776×10^9 mL), and $A_{i,0}$ is the activity of radionuclide i at the time of release. The Committed Effective Dose Equivalent (CEDE) is then calculated as

$$D_i = 5\frac{\bar{C}_i}{DAC_i} \tag{A.5}$$

and

$$D = \sum_{i=\text{All nuclides}} D_i \tag{A.6}$$

where DAC_i is the DAC value defined by 10CFR20 for each radionuclide [36]. The total CEDE (D) for the long run is then determined as 4.61×10^{-8} rem and 1.14×10^{-7} for the short run, both nearly insignificant quantities. Even if all of the MPFD^3-Ts would fail the respective CEDE would only increase to 2.59×10^{-4} and 6.43×10^{-4} rem.

Table A.1: ALI and CEDE fission product analysis for gases listed in 10CFR20 Appendix B [35, 36]. The units for the ORIGEN Activities are Ci/W-tonne while the ALI and MPFD^3-T Activity are μCi and CEDE (Committed Effective Dose Equivalent) is rem. Read 2.60^{-9} as 2.60×10^{-9}.

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{80m}Br	2.60^{-9}	1.90^{-6}	2^{+4}	7^{-6}	4.47^{-15}	3.26^{-12}	5.69^{-22}	4.16^{-19}
^{80}Br	6.10^{-9}	5.41^{-6}	2^{+5}	8^{-5}	1.05^{-14}	9.29^{-12}	7.80^{-24}	6.92^{-21}
^{82}Br	1.03^{-6}	3.64^{-4}	4^{+3}	2^{-6}	1.77^{-12}	6.25^{-10}	6.30^{-18}	2.23^{-15}

Continued on next page...

Table A.1: Gaseous fission products listed in 10CFR20 (*cont.)*

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{83}Br	4.46^{-3}	4.02^{+0}	6^{+4}	3^{-5}	7.66^{-9}	6.91^{-6}	1.23^{-16}	1.11^{-13}
^{84}Br	8.43^{-3}	8.43^{+0}	6^{+4}	2^{-5}	1.45^{-8}	1.45^{-5}	7.75^{-17}	7.75^{-14}
^{3}H	8.23^{-5}	2.36^{-5}	8^{+4}	2^{-5}	1.41^{-10}	4.05^{-11}	8.43^{-15}	2.42^{-15}
^{128}I	1.11^{-7}	1.11^{-4}	1^{+5}	5^{-5}	1.91^{-13}	1.91^{-10}	3.21^{-22}	3.21^{-19}
^{129}I	9.78^{-9}	8.36^{-10}	9^{+0}	4^{-9}	1.68^{-14}	1.44^{-15}	5.15^{-15}	4.40^{-16}
^{130}I	1.96^{-6}	9.56^{-4}	7^{+2}	3^{-7}	3.37^{-12}	1.64^{-9}	2.80^{-17}	1.37^{-14}
^{131}I	2.37^{-2}	2.82^{+0}	5^{+1}	2^{-8}	4.07^{-8}	4.84^{-6}	7.93^{-11}	9.43^{-9}
^{132}I	3.56^{-2}	7.73^{+0}	8^{+3}	3^{-6}	6.12^{-8}	1.33^{-5}	9.38^{-15}	2.04^{-12}
^{133}I	5.68^{-2}	2.32^{+1}	3^{+2}	1^{-7}	9.76^{-8}	3.99^{-5}	4.10^{-12}	1.67^{-9}
^{134}I	6.39^{-2}	6.38^{+1}	5^{+4}	2^{-5}	1.10^{-7}	1.10^{-4}	9.70^{-16}	9.69^{-13}
^{135}I	5.31^{-2}	3.28^{+1}	2^{+3}	7^{-7}	9.12^{-8}	5.63^{-5}	1.73^{-13}	1.07^{-10}
^{81}Kr	1.97^{-15}	2.25^{-16}	—	7^{-4}	3.38^{-21}	3.87^{-22}	5.93^{-27}	6.77^{-28}
^{83m}Kr	4.46^{-3}	3.30^{+0}	—	1^{-2}	7.66^{-9}	5.67^{-6}	2.88^{-19}	2.13^{-16}
^{85m}Kr	1.07^{-2}	7.76^{+0}	—	2^{-5}	1.84^{-8}	1.33^{-5}	8.32^{-16}	6.04^{-13}
^{85}Kr	2.10^{-3}	5.96^{-4}	—	1^{-4}	3.61^{-9}	1.02^{-9}	4.28^{-14}	1.22^{-14}
^{87}Kr	2.17^{-2}	2.14^{+1}	—	5^{-6}	3.73^{-8}	3.68^{-5}	1.91^{-15}	1.88^{-12}
^{88}Kr	3.05^{-2}	2.62^{+1}	—	2^{-6}	5.24^{-8}	4.50^{-5}	1.50^{-14}	1.29^{-11}
^{129m}Xe	4.33^{-11}	5.22^{-9}	—	2^{-4}	7.44^{-17}	8.97^{-15}	1.60^{-23}	1.93^{-21}
^{131m}Xe	2.63^{-4}	3.82^{-3}	—	4^{-4}	4.52^{-10}	6.56^{-9}	6.51^{-17}	9.46^{-16}
^{133m}Xe	1.65^{-3}	3.57^{-1}	—	1^{-4}	2.83^{-9}	6.13^{-7}	3.01^{-16}	6.50^{-14}
^{133}Xe	5.68^{-2}	6.76^{+0}	—	1^{-4}	9.76^{-8}	1.16^{-5}	2.48^{-14}	2.95^{-12}
^{135m}Xe	9.52^{-3}	6.41^{+0}	—	9^{-6}	1.64^{-8}	1.10^{-5}	9.33^{-17}	6.28^{-14}
^{135}Xe	5.52^{-2}	1.96^{+1}	—	1^{-5}	9.48^{-8}	3.37^{-5}	1.74^{-14}	6.18^{-12}
^{138}Xe	5.26^{-2}	5.26^{+1}	—	4^{-6}	9.04^{-8}	9.04^{-5}	1.07^{-15}	1.07^{-12}

Table A.2: ALI and CEDE fission product analysis for gases not listed in 10CFR20 Appendix B [35, 36]. The units for the ORIGEN Activities are Ci/W-tonne while the ALI and MPFD^3-T Activity are μCi and CEDE (Committed Effective Dose Equivalent) is rem. Read 1.21^{-11} as 1.21×10^{-11}.

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{79m}Br	1.21^{-11}	1.21^{-8}	2^{+2}	1^{-7}	2.08^{-17}	2.08^{-14}	5.67^{-26}	5.67^{-23}
^{82m}Br	5.97^{-7}	5.97^{-4}	2^{+2}	1^{-7}	1.03^{-12}	1.03^{-9}	2.12^{-19}	2.12^{-16}
^{84m}Br	1.62^{-4}	1.62^{-1}	2^{+2}	1^{-7}	2.78^{-10}	2.78^{-7}	5.62^{-17}	5.62^{-14}
^{85}Br	1.05^{-2}	1.05^{+1}	2^{+2}	1^{-7}	1.80^{-8}	1.80^{-5}	1.74^{-15}	1.74^{-12}

Continued on next page...

Table A.2: Gaseous fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{86m}Br	8.08^{-3}	8.08^{+0}	2^{+2}	1^{-7}	1.39^{-8}	1.39^{-5}	4.32^{-16}	4.32^{-13}
^{86}Br	8.01^{-3}	8.01^{+0}	2^{+2}	1^{-7}	1.38^{-8}	1.38^{-5}	4.28^{-16}	4.28^{-13}
^{87}Br	1.82^{-2}	1.82^{+1}	2^{+2}	1^{-7}	3.13^{-8}	3.13^{-5}	9.83^{-16}	9.83^{-13}
^{88}Br	2.11^{-2}	2.11^{+1}	2^{+2}	1^{-7}	3.62^{-8}	3.62^{-5}	3.34^{-16}	3.34^{-13}
^{89}Br	1.63^{-2}	1.63^{+1}	2^{+2}	1^{-7}	2.80^{-8}	2.80^{-5}	6.90^{-17}	6.90^{-14}
^{90}Br	1.12^{-2}	1.12^{+1}	2^{+2}	1^{-7}	1.92^{-8}	1.92^{-5}	2.05^{-17}	2.05^{-14}
^{91}Br	3.56^{-3}	3.56^{+0}	2^{+2}	1^{-7}	6.12^{-9}	6.12^{-6}	1.85^{-18}	1.85^{-15}
^{92}Br	1.52^{-4}	1.52^{-1}	2^{+2}	1^{-7}	2.61^{-10}	2.61^{-7}	5.27^{-20}	5.27^{-17}
^{93}Br	4.25^{-5}	4.25^{-2}	2^{+2}	1^{-7}	7.30^{-11}	7.30^{-8}	4.17^{-21}	4.17^{-18}
^{94}Br	2.64^{-6}	2.64^{-3}	2^{+2}	1^{-7}	4.53^{-12}	4.53^{-9}	1.78^{-22}	1.78^{-19}
^{95}Br	4.85^{-8}	4.85^{-5}	2^{+2}	1^{-7}	8.33^{-14}	8.33^{-11}	6.98^{-30}	6.98^{-27}
^{96}Br	2.55^{-9}	2.55^{-6}	2^{+2}	1^{-7}	4.38^{-15}	4.38^{-12}	3.67^{-31}	3.67^{-28}
^{130m}I	6.21^{-7}	6.21^{-4}	2^{+2}	1^{-7}	1.07^{-12}	1.07^{-9}	3.17^{-19}	3.17^{-16}
^{133m}I	1.08^{-3}	1.08^{+0}	2^{+2}	1^{-7}	1.86^{-9}	1.86^{-6}	9.34^{-18}	9.34^{-15}
^{134m}I	3.57^{-3}	3.57^{+0}	2^{+2}	1^{-7}	6.13^{-9}	6.13^{-6}	7.43^{-16}	7.43^{-13}
^{136m}I	1.67^{-2}	1.67^{+1}	2^{+2}	1^{-7}	2.87^{-8}	2.87^{-5}	7.24^{-16}	7.24^{-13}
^{136}I	2.53^{-2}	2.53^{+1}	2^{+2}	1^{-7}	4.35^{-8}	4.35^{-5}	2.04^{-15}	2.04^{-12}
^{137}I	2.73^{-2}	2.73^{+1}	2^{+2}	1^{-7}	4.69^{-8}	4.69^{-5}	6.45^{-16}	6.45^{-13}
^{138}I	1.38^{-2}	1.38^{+1}	2^{+2}	1^{-7}	2.37^{-8}	2.37^{-5}	8.51^{-17}	8.51^{-14}
^{139}I	6.32^{-3}	6.32^{+0}	2^{+2}	1^{-7}	1.09^{-8}	1.09^{-5}	1.40^{-17}	1.40^{-14}
^{140}I	1.89^{-3}	1.89^{+0}	2^{+2}	1^{-7}	3.25^{-9}	3.25^{-6}	1.56^{-18}	1.56^{-15}
^{141}I	2.77^{-4}	2.77^{-1}	2^{+2}	1^{-7}	4.76^{-10}	4.76^{-7}	1.23^{-19}	1.23^{-16}
^{142}I	1.85^{-5}	1.85^{-2}	2^{+2}	1^{-7}	3.18^{-11}	3.18^{-8}	3.56^{-21}	3.56^{-18}
^{143}I	7.47^{-7}	7.47^{-4}	2^{+2}	1^{-7}	1.28^{-12}	1.28^{-9}	1.08^{-28}	1.08^{-25}
^{144}I	2.68^{-8}	2.68^{-5}	2^{+2}	1^{-7}	4.60^{-14}	4.60^{-11}	3.88^{-30}	3.88^{-27}
^{81m}Kr	7.63^{-12}	7.63^{-9}	—	1^{-7}	1.31^{-17}	1.31^{-14}	9.65^{-26}	9.65^{-23}
^{89}Kr	3.91^{-2}	3.91^{+1}	—	1^{-7}	6.72^{-8}	6.72^{-5}	7.12^{-15}	7.12^{-12}
^{90}Kr	3.86^{-2}	3.86^{+1}	—	1^{-7}	6.63^{-8}	6.63^{-5}	1.20^{-15}	1.20^{-12}
^{91}Kr	2.93^{-2}	2.93^{+1}	—	1^{-7}	5.03^{-8}	5.03^{-5}	2.43^{-16}	2.43^{-13}
^{92}Kr	1.29^{-2}	1.29^{+1}	—	1^{-7}	2.22^{-8}	2.22^{-5}	2.29^{-17}	2.29^{-14}
^{93}Kr	4.49^{-3}	4.49^{+0}	—	1^{-7}	7.71^{-9}	7.71^{-6}	5.57^{-18}	5.57^{-15}
^{94}Kr	1.98^{-3}	1.98^{+0}	—	1^{-7}	3.40^{-9}	3.40^{-6}	4.00^{-19}	4.00^{-16}
^{95}Kr	6.75^{-5}	6.75^{-2}	—	1^{-7}	1.16^{-10}	1.16^{-7}	5.67^{-20}	5.67^{-17}
^{96}Kr	1.31^{-5}	1.31^{-2}	—	1^{-7}	2.25^{-11}	2.25^{-8}	1.01^{-21}	1.01^{-18}
^{97}Kr	2.18^{-7}	2.18^{-4}	—	1^{-7}	3.74^{-13}	3.74^{-10}	1.32^{-23}	1.32^{-20}
^{98}Kr	3.40^{-8}	3.40^{-5}	—	1^{-7}	5.84^{-14}	5.84^{-11}	1.50^{-24}	1.50^{-21}
^{134m}Xe	1.98^{-4}	1.98^{-1}	—	1^{-7}	3.40^{-10}	3.40^{-7}	5.54^{-20}	5.54^{-17}
^{137}Xe	5.04^{-2}	5.04^{+1}	—	1^{-7}	8.66^{-8}	8.66^{-5}	1.12^{-14}	1.12^{-11}

Continued on next page...

Table A.2: Gaseous fission products not listed in 10CFR20 (*cont.)*

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{139}Xe	4.32^{-2}	4.32^{+1}	—	1^{-7}	7.42^{-8}	7.42^{-5}	1.66^{-15}	1.66^{-12}
^{140}Xe	3.05^{-2}	3.05^{+1}	—	1^{-7}	5.24^{-8}	5.24^{-5}	3.98^{-16}	3.98^{-13}
^{141}Xe	1.02^{-2}	1.02^{+1}	—	1^{-7}	1.75^{-8}	1.75^{-5}	1.69^{-17}	1.69^{-14}
^{142}Xe	3.12^{-3}	3.12^{+0}	—	1^{-7}	5.36^{-9}	5.36^{-6}	3.61^{-18}	3.61^{-15}
^{143}Xe	4.28^{-4}	4.28^{-1}	—	1^{-7}	7.35^{-10}	7.35^{-7}	1.24^{-19}	1.24^{-16}
^{144}Xe	5.06^{-5}	5.06^{-2}	—	1^{-7}	8.69^{-11}	8.69^{-8}	5.85^{-20}	5.85^{-17}
^{145}Xe	6.73^{-7}	6.73^{-4}	—	1^{-7}	1.16^{-12}	1.16^{-9}	1.22^{-22}	1.22^{-19}
^{146}Xe	1.02^{-7}	1.02^{-4}	—	1^{-7}	1.75^{-13}	1.75^{-10}	1.43^{-23}	1.43^{-20}
^{147}Xe	2.04^{-9}	2.04^{-6}	—	1^{-7}	3.50^{-15}	3.50^{-12}	1.97^{-25}	1.97^{-22}

Table A.3: ALI and CEDE fission product analysis for particulates listed in 10CFR20 Appendix B [35, 36]. The units for the ORIGEN Activities are Ci/W-tonne while the ALI and MPFD^3-T Activity are μCi and CEDE (Committed Effective Dose Equivalent) is rem. Read 2.40^{-11} as 2.40×10^{-11}.

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{110m}Ag	2.40^{-11}	1.09^{-10}	9^{+1}	4^{-8}	4.12^{-17}	1.87^{-16}	7.94^{-19}	3.61^{-18}
^{111}Ag	1.65^{-4}	2.08^{-2}	2^{-1}	1^{-10}	2.83^{-10}	3.57^{-8}	1.03^{-10}	1.29^{-8}
^{112}Ag	1.06^{-4}	3.66^{-2}	2^{-1}	1^{-10}	1.82^{-10}	6.29^{-8}	1.16^{-12}	3.99^{-10}
^{115}Ag	6.17^{-5}	6.17^{-2}	2^{+2}	1^{-7}	1.06^{-10}	1.06^{-7}	7.14^{-17}	7.14^{-14}
^{76}As	1.12^{-9}	4.35^{-7}	1^{+3}	1^{+3}	1.92^{-15}	7.47^{-13}	1.02^{-29}	3.94^{-27}
^{77}As	6.94^{-5}	2.18^{-2}	5^{+3}	2^{-6}	1.19^{-10}	3.74^{-8}	4.68^{-16}	1.47^{-13}
^{78}As	1.39^{-4}	1.24^{-1}	2^{+4}	9^{-6}	2.39^{-10}	2.13^{-7}	8.11^{-18}	7.23^{-15}
^{135m}Ba	1.98^{-9}	7.45^{-7}	2^{-1}	1^{-10}	3.40^{-15}	1.28^{-12}	1.98^{-16}	7.44^{-14}
^{139}Ba	5.43^{-2}	5.33^{+1}	3^{+4}	1^{-5}	9.33^{-8}	9.16^{-5}	2.61^{-15}	2.56^{-12}
^{140}Ba	5.23^{-2}	4.22^{+0}	1^{+3}	6^{-7}	8.98^{-8}	7.25^{-6}	9.28^{-12}	7.49^{-10}
^{141}Ba	4.91^{-2}	4.91^{+1}	7^{+4}	3^{-5}	8.43^{-8}	8.43^{-5}	1.73^{-16}	1.73^{-13}
^{142}Ba	4.88^{-2}	4.88^{+1}	1^{+5}	6^{-5}	8.38^{-8}	8.38^{-5}	5.03^{-17}	5.03^{-14}
^{10}Be	1.31^{-12}	1.49^{-13}	2^{-1}	1^{-10}	2.25^{-18}	2.56^{-19}	2.76^{-17}	3.14^{-18}
^{14}C	5.25^{-11}	6.01^{-12}	2^{+3}	1^{-6}	9.02^{-17}	1.03^{-17}	1.11^{-19}	1.27^{-20}
^{113m}Cd	1.95^{-6}	4.65^{-7}	2^{+0}	1^{-9}	3.35^{-12}	7.99^{-13}	4.01^{-12}	9.56^{-13}
^{115m}Cd	7.69^{-6}	1.94^{-4}	5^{+1}	2^{-8}	1.32^{-11}	3.33^{-10}	1.42^{-13}	3.59^{-12}
^{115}Cd	7.85^{-5}	2.36^{-2}	1^{+3}	5^{-7}	1.35^{-10}	4.05^{-8}	2.91^{-15}	8.76^{-13}
^{117m}Cd	3.16^{-5}	2.56^{-2}	1^{+4}	5^{-6}	5.43^{-11}	4.40^{-8}	7.47^{-18}	6.05^{-15}
^{117}Cd	5.86^{-5}	5.17^{-2}	1^{+4}	5^{-6}	1.01^{-10}	8.88^{-8}	1.01^{-17}	8.95^{-15}

Continued on next page...

Table A.3: Particulate fission products listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{141}Ce	4.93^{-2}	1.57^{+0}	6^{+2}	2^{-7}	8.47^{-8}	2.70^{-6}	6.66^{-11}	2.12^{-9}
^{143}Ce	4.98^{-2}	1.84^{+1}	2^{+3}	7^{-7}	8.55^{-8}	3.16^{-5}	8.17^{-13}	3.02^{-10}
^{144}Ce	4.56^{-2}	1.85^{-1}	1^{+1}	6^{-9}	7.83^{-8}	3.18^{-7}	1.06^{-8}	4.30^{-8}
^{134m}Cs	2.13^{-7}	1.82^{-4}	1^{+5}	6^{-5}	3.66^{-13}	3.13^{-10}	3.57^{-21}	3.05^{-18}
^{134}Cs	2.99^{-7}	4.28^{-7}	1^{+2}	4^{-8}	5.14^{-13}	7.35^{-13}	1.34^{-14}	1.91^{-14}
^{135m}Cs	5.06^{-6}	5.05^{-3}	2^{+5}	8^{-5}	8.69^{-12}	8.67^{-9}	1.94^{-20}	1.93^{-17}
^{135}Cs	6.66^{-7}	5.66^{-8}	1^{+3}	5^{-7}	1.14^{-12}	9.72^{-14}	2.81^{-15}	2.38^{-16}
^{136}Cs	4.44^{-5}	3.50^{-3}	7^{+2}	3^{-7}	7.63^{-11}	6.01^{-9}	1.60^{-14}	1.26^{-12}
^{137}Cs	3.10^{-2}	5.46^{-3}	2^{+2}	6^{-8}	5.33^{-8}	9.38^{-9}	1.08^{-9}	1.89^{-10}
^{138}Cs	5.68^{-2}	5.68^{+1}	6^{+4}	2^{-5}	9.76^{-8}	9.76^{-5}	5.29^{-16}	5.29^{-13}
^{165}Dy	1.13^{-8}	1.02^{-5}	5^{+4}	2^{-5}	1.94^{-14}	1.75^{-11}	4.57^{-22}	4.13^{-19}
^{166}Dy	5.02^{-9}	1.14^{-6}	7^{+2}	3^{-7}	8.62^{-15}	1.96^{-12}	4.73^{-19}	1.07^{-16}
^{152m}Eu	5.48^{-10}	1.49^{-8}	6^{+3}	3^{-6}	9.41^{-16}	2.56^{-14}	5.90^{-22}	1.60^{-20}
^{152}Eu	4.38^{-10}	6.43^{-12}	2^{+1}	1^{-8}	7.52^{-16}	1.10^{-17}	8.99^{-17}	1.32^{-18}
^{154}Eu	1.32^{-8}	5.02^{-9}	2^{+1}	8^{-9}	2.27^{-14}	8.62^{-15}	3.34^{-15}	1.27^{-15}
^{155}Eu	2.75^{-4}	1.85^{-4}	9^{+1}	4^{-8}	4.72^{-10}	3.18^{-10}	1.35^{-11}	9.07^{-12}
^{156}Eu	1.13^{-4}	6.42^{-3}	5^{+2}	2^{-7}	1.94^{-10}	1.10^{-8}	7.15^{-14}	4.06^{-12}
^{157}Eu	5.41^{-5}	2.49^{-2}	5^{+3}	2^{-6}	9.29^{-11}	4.28^{-8}	1.42^{-16}	6.54^{-14}
^{158}Eu	2.69^{-5}	2.69^{-2}	6^{+4}	2^{-5}	4.62^{-11}	4.62^{-8}	3.57^{-19}	3.57^{-16}
^{72}Ga	2.08^{-7}	5.06^{-5}	3^{+3}	1^{-6}	3.57^{-13}	8.69^{-11}	1.01^{-18}	2.45^{-16}
^{73}Ga	8.84^{-7}	6.21^{-4}	2^{+4}	6^{-6}	1.52^{-12}	1.07^{-9}	2.49^{-19}	1.75^{-16}
^{152}Gd	4.73^{-23}	4.77^{-25}	1^{-2}	4^{-12}	8.13^{-29}	8.19^{-31}	2.50^{-26}	2.52^{-28}
^{153}Gd	7.81^{-14}	3.70^{-13}	1^{+2}	6^{-8}	1.34^{-19}	6.36^{-19}	1.70^{-21}	8.03^{-21}
^{159}Gd	8.70^{-6}	3.83^{-3}	6^{+3}	2^{-6}	1.49^{-11}	6.58^{-9}	2.79^{-17}	1.23^{-14}
^{75}Ge	9.41^{-6}	9.23^{-3}	2^{+2}	1^{-7}	1.62^{-11}	1.59^{-8}	4.50^{-17}	4.42^{-14}
^{77}Ge	2.45^{-5}	1.23^{-2}	6^{+3}	2^{-6}	4.21^{-11}	2.11^{-8}	4.80^{-17}	2.41^{-14}
^{78}Ge	1.38^{-4}	1.35^{-1}	2^{+4}	9^{-6}	2.37^{-10}	2.32^{-7}	7.71^{-18}	7.54^{-15}
^{166m}Ho	7.11^{-14}	8.20^{-15}	7^{+0}	3^{-9}	1.22^{-19}	1.41^{-20}	4.99^{-20}	5.76^{-21}
^{166}Ho	5.03^{-9}	7.57^{-7}	2^{+3}	7^{-7}	8.64^{-15}	1.30^{-12}	6.70^{-20}	1.01^{-17}
^{114m}In	1.16^{-13}	2.56^{-12}	6^{+1}	3^{-8}	1.99^{-19}	4.40^{-18}	1.58^{-21}	3.49^{-20}
^{115m}In	7.85^{-5}	2.05^{-2}	4^{+4}	2^{-5}	1.35^{-10}	3.52^{-8}	6.11^{-18}	1.59^{-15}
^{115}In	4.64^{-18}	2.02^{-19}	1^{+0}	6^{-10}	7.97^{-24}	3.47^{-25}	1.61^{-23}	7.01^{-25}
^{116m}In	1.34^{-10}	1.30^{-7}	8^{+4}	3^{-5}	2.30^{-16}	2.23^{-13}	1.40^{-24}	1.36^{-21}
^{117m}In	6.84^{-5}	4.73^{-2}	3^{+4}	1^{-5}	1.17^{-10}	8.13^{-8}	4.72^{-18}	3.27^{-15}
^{117}In	5.39^{-5}	4.03^{-2}	2^{+5}	7^{-5}	9.26^{-11}	6.92^{-8}	1.92^{-19}	1.43^{-16}
^{119m}In	7.01^{-5}	7.01^{-2}	1^{+5}	5^{-5}	1.20^{-10}	1.20^{-7}	1.46^{-19}	1.46^{-16}
^{138}La	1.09^{-16}	1.24^{-17}	4^{+0}	1^{-9}	1.87^{-22}	2.13^{-23}	2.30^{-22}	2.61^{-23}
^{140}La	5.24^{-2}	2.25^{+0}	1^{+3}	6^{-7}	9.00^{-8}	3.87^{-6}	1.22^{-12}	5.24^{-11}

Continued on next page...

Table A.3: Particulate fission products listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{141}La	4.93^{-2}	3.79^{+1}	9^{+3}	4^{-6}	8.47^{-8}	6.51^{-5}	1.68^{-14}	1.29^{-11}
^{142}La	4.97^{-2}	4.83^{+1}	2^{+4}	9^{-6}	8.54^{-8}	8.30^{-5}	2.94^{-15}	2.85^{-12}
^{143}La	4.96^{-2}	4.96^{+1}	9^{+4}	4^{-5}	8.52^{-8}	8.52^{-5}	1.01^{-16}	1.01^{-13}
^{99}Mo	5.06^{-2}	1.31^{+1}	1^{+3}	6^{-7}	8.69^{-8}	2.25^{-5}	1.90^{-12}	4.92^{-10}
^{101}Mo	4.23^{-2}	4.23^{+1}	1^{+5}	6^{-5}	7.27^{-8}	7.27^{-5}	5.92^{-17}	5.92^{-14}
^{93m}Nb	5.37^{-7}	4.47^{-11}	1^{+4}	6^{+6}	9.22^{-13}	7.68^{-17}	1.85^{-28}	1.54^{-32}
^{94}Nb	1.04^{-11}	1.19^{-12}	2^{+1}	6^{-9}	1.79^{-17}	2.04^{-18}	3.65^{-18}	4.18^{-19}
^{95m}Nb	3.78^{-4}	2.44^{-3}	2^{+3}	9^{-7}	6.49^{-10}	4.19^{-9}	1.26^{-14}	8.14^{-14}
^{95}Nb	5.38^{-2}	4.05^{-2}	1^{+3}	5^{-7}	9.24^{-8}	6.96^{-8}	3.13^{-11}	2.36^{-11}
^{96}Nb	5.01^{-6}	2.02^{-3}	2^{+3}	1^{-6}	8.61^{-12}	3.47^{-9}	4.06^{-17}	1.64^{-14}
^{97}Nb	4.93^{-2}	1.98^{+1}	7^{+4}	3^{-5}	8.47^{-8}	3.40^{-5}	6.85^{-16}	2.75^{-13}
^{98}Nb	4.86^{-2}	4.86^{+1}	5^{+4}	2^{-5}	8.35^{-8}	8.35^{-5}	6.69^{-19}	6.69^{-16}
^{147}Nd	1.90^{-2}	1.79^{+0}	8^{+2}	4^{-7}	3.26^{-8}	3.07^{-6}	4.35^{-12}	4.10^{-10}
^{149}Nd	9.11^{-3}	8.74^{+0}	2^{+4}	1^{-5}	1.56^{-8}	1.50^{-5}	5.46^{-16}	5.24^{-13}
^{151}Nd	3.50^{-3}	3.50^{+0}	2^{+5}	8^{-5}	6.01^{-9}	6.01^{-6}	3.08^{-18}	3.08^{-15}
^{107}Pd	5.83^{-9}	6.98^{-10}	4^{+2}	2^{-7}	1.00^{-14}	1.20^{-15}	6.14^{-17}	7.35^{-18}
^{109}Pd	2.50^{-4}	1.19^{-1}	5^{+3}	2^{-6}	4.29^{-10}	2.04^{-7}	5.82^{-16}	2.77^{-13}
^{147}Pm	1.91^{-2}	2.92^{-3}	1^{+2}	5^{-8}	3.28^{-8}	5.02^{-9}	7.07^{-10}	1.08^{-10}
^{148m}Pm	6.32^{-9}	1.69^{-7}	3^{+2}	1^{-7}	1.09^{-14}	2.90^{-13}	2.17^{-17}	5.79^{-16}
^{148}Pm	4.83^{-8}	7.93^{-6}	5^{+2}	2^{-7}	8.30^{-14}	1.36^{-11}	1.08^{-17}	1.77^{-15}
^{149}Pm	9.11^{-3}	2.47^{+0}	2^{+3}	8^{-7}	1.56^{-8}	4.24^{-6}	2.09^{-13}	5.68^{-11}
^{150}Pm	4.53^{-6}	3.96^{-3}	2^{+4}	7^{-6}	7.78^{-12}	6.80^{-9}	6.03^{-19}	5.27^{-16}
^{151}Pm	3.52^{-3}	1.36^{+0}	3^{+3}	1^{-6}	6.05^{-9}	2.34^{-6}	3.46^{-14}	1.34^{-11}
^{142m}Pr	3.04^{-10}	2.41^{-7}	1^{+5}	6^{-5}	5.22^{-16}	4.14^{-13}	4.27^{-25}	3.39^{-22}
^{142}Pr	6.68^{-10}	2.09^{-7}	2^{+3}	8^{-7}	1.15^{-15}	3.59^{-13}	5.53^{-21}	1.73^{-18}
^{143}Pr	4.98^{-2}	2.31^{+0}	7^{+2}	3^{-7}	8.55^{-8}	3.97^{-6}	1.88^{-11}	8.71^{-10}
^{144}Pr	4.56^{-2}	1.86^{-1}	1^{+5}	5^{-5}	7.83^{-8}	3.20^{-7}	9.11^{-17}	3.71^{-16}
^{145}Pr	3.29^{-2}	2.12^{+1}	8^{+3}	3^{-6}	5.65^{-8}	3.64^{-5}	2.27^{-14}	1.46^{-11}
^{147}Pr	1.90^{-2}	1.90^{+1}	2^{+5}	8^{-5}	3.26^{-8}	3.26^{-5}	1.84^{-17}	1.84^{-14}
^{86}Rb	6.33^{-7}	3.63^{-5}	8^{+2}	3^{-7}	1.09^{-12}	6.24^{-11}	3.27^{-16}	1.87^{-14}
^{87}Rb	1.28^{-11}	1.40^{-12}	2^{+3}	6^{-7}	2.20^{-17}	2.40^{-18}	4.49^{-20}	4.91^{-21}
^{88}Rb	3.08^{-2}	2.65^{+1}	6^{+4}	3^{-5}	5.29^{-8}	4.55^{-5}	1.05^{-16}	9.04^{-14}
^{89}Rb	4.05^{-2}	4.05^{+1}	1^{+5}	6^{-5}	6.96^{-8}	6.96^{-5}	5.95^{-17}	5.95^{-14}
^{103m}Rh	2.37^{-2}	6.48^{-1}	1^{+6}	5^{-4}	4.07^{-8}	1.11^{-6}	1.54^{-17}	4.21^{-16}
^{105}Rh	8.53^{-3}	2.57^{+0}	6^{+3}	2^{-6}	1.47^{-8}	4.41^{-6}	5.25^{-14}	1.58^{-11}
^{106m}Rh	2.90^{-9}	3.16^{-6}	3^{+4}	1^{-5}	4.98^{-15}	5.43^{-12}	2.20^{-22}	2.39^{-19}
^{107}Rh	1.37^{-3}	1.37^{+0}	2^{+5}	1^{-4}	2.35^{-9}	2.35^{-6}	1.72^{-18}	1.72^{-15}
^{103}Ru	2.63^{-2}	7.45^{-1}	6^{+2}	3^{-7}	4.52^{-8}	1.28^{-6}	2.85^{-11}	8.09^{-10}

Continued on next page...

Table A.3: Particulate fission products listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{105}Ru	8.53^{-3}	6.23^{+0}	1^{+4}	5^{-6}	1.47^{-8}	1.07^{-5}	2.63^{-15}	1.92^{-12}
^{106}Ru	3.27^{-3}	1.02^{-2}	1^{+1}	5^{-9}	5.62^{-9}	1.75^{-8}	1.00^{-9}	3.12^{-9}
^{122}Sb	3.73^{-10}	9.80^{-8}	1^{+3}	4^{-7}	6.41^{-16}	1.68^{-13}	2.11^{-20}	5.55^{-18}
^{124m}Sb	2.47^{-8}	2.47^{-5}	6^{+5}	2^{-4}	4.24^{-14}	4.24^{-11}	1.11^{-24}	1.11^{-21}
^{124}Sb	4.67^{-8}	8.76^{-7}	2^{+2}	1^{-7}	8.02^{-14}	1.50^{-12}	2.30^{-16}	4.32^{-15}
^{125}Sb	2.49^{-4}	1.77^{-4}	5^{+2}	2^{-7}	4.28^{-10}	3.04^{-10}	2.32^{-12}	1.65^{-12}
^{126m}Sb	2.94^{-6}	2.80^{-3}	2^{+5}	8^{-5}	5.05^{-12}	4.81^{-9}	4.07^{-21}	3.87^{-18}
^{126}Sb	7.44^{-6}	6.16^{-4}	5^{+2}	2^{-7}	1.28^{-11}	1.06^{-9}	3.84^{-15}	3.18^{-13}
^{127}Sb	1.10^{-3}	2.19^{-1}	9^{+2}	4^{-7}	1.89^{-9}	3.76^{-7}	8.81^{-14}	1.75^{-11}
^{128m}Sb	2.84^{-3}	2.84^{+0}	4^{+5}	2^{-4}	4.88^{-9}	4.88^{-6}	8.20^{-19}	8.20^{-16}
^{128}Sb	9.20^{-5}	5.02^{-2}	3^{+3}	1^{-6}	1.58^{-10}	8.62^{-8}	2.90^{-16}	1.58^{-13}
^{129}Sb	5.33^{-3}	3.94^{+0}	9^{+3}	4^{-6}	9.16^{-9}	6.77^{-6}	2.03^{-15}	1.50^{-12}
^{130}Sb	1.73^{-3}	1.73^{+0}	6^{+4}	3^{-5}	2.97^{-9}	2.97^{-6}	1.28^{-17}	1.28^{-14}
^{131}Sb	2.11^{-2}	2.11^{+1}	2^{+4}	1^{-5}	3.62^{-8}	3.62^{-5}	2.81^{-16}	2.81^{-13}
^{79}Se	1.82^{-7}	2.08^{-8}	6^{+2}	2^{-7}	3.13^{-13}	3.57^{-14}	1.92^{-15}	2.19^{-16}
^{81m}Se	6.21^{-5}	6.19^{-2}	7^{+4}	3^{-5}	1.07^{-10}	1.06^{-7}	6.85^{-19}	6.83^{-16}
^{81}Se	1.76^{-3}	1.76^{+0}	2^{+5}	9^{-5}	3.02^{-9}	3.02^{-6}	2.08^{-18}	2.08^{-15}
^{83}Se	1.64^{-3}	1.64^{+0}	1^{+5}	5^{-5}	2.82^{-9}	2.82^{-6}	4.23^{-18}	4.23^{-15}
^{147}Sm	4.49^{-12}	8.20^{-17}	2^{-1}	1^{-10}	7.71^{-18}	1.41^{-22}	9.46^{-17}	1.73^{-21}
^{151}Sm	9.34^{-4}	7.83^{-5}	1^{+2}	4^{-8}	1.60^{-9}	1.35^{-10}	4.90^{-11}	4.11^{-12}
^{153}Sm	1.36^{-3}	4.27^{-1}	3^{+3}	1^{-6}	2.34^{-9}	7.33^{-7}	2.21^{-14}	6.93^{-12}
^{155}Sm	2.76^{-4}	2.76^{-1}	2^{+5}	9^{-5}	4.74^{-10}	4.74^{-7}	3.93^{-19}	3.93^{-16}
^{156}Sm	1.13^{-4}	6.05^{-2}	9^{+3}	4^{-6}	1.94^{-10}	1.04^{-7}	9.21^{-17}	4.93^{-14}
^{119m}Sn	1.35^{-6}	6.35^{-6}	1^{+3}	4^{-7}	2.32^{-12}	1.09^{-11}	4.76^{-15}	2.24^{-14}
^{121m}Sn	2.12^{-8}	3.15^{-9}	5^{+2}	2^{-7}	3.64^{-14}	5.41^{-15}	2.22^{-16}	3.29^{-17}
^{121}Sn	1.07^{-4}	4.15^{-2}	1^{+4}	5^{-6}	1.84^{-10}	7.13^{-8}	2.01^{-16}	7.80^{-14}
^{123m}Sn	9.99^{-5}	9.99^{-2}	1^{+5}	5^{-5}	1.72^{-10}	1.72^{-7}	4.62^{-19}	4.62^{-16}
^{123}Sn	3.67^{-5}	3.25^{-4}	2^{+2}	7^{-8}	6.30^{-11}	5.58^{-10}	4.84^{-13}	4.28^{-12}
^{125}Sn	1.13^{-4}	1.16^{-2}	4^{+2}	1^{-7}	1.94^{-10}	1.99^{-8}	9.06^{-14}	9.30^{-12}
^{126}Sn	1.32^{-7}	1.50^{-8}	6^{+1}	2^{-8}	2.27^{-13}	2.58^{-14}	1.39^{-14}	1.58^{-15}
^{127}Sn	7.02^{-4}	6.52^{-1}	2^{+4}	8^{-6}	1.21^{-9}	1.12^{-6}	6.39^{-17}	5.93^{-14}
^{128}Sn	2.79^{-3}	2.78^{+0}	3^{+4}	1^{-5}	4.79^{-9}	4.78^{-6}	9.52^{-17}	9.48^{-14}
^{87m}Sr	8.45^{-9}	7.30^{-6}	2^{-1}	1^{-10}	1.45^{-14}	1.25^{-11}	8.23^{-17}	7.11^{-14}
^{89}Sr	4.05^{-2}	9.30^{-1}	1^{+2}	6^{-8}	6.96^{-8}	1.60^{-6}	2.82^{-10}	6.47^{-9}
^{90}Sr	2.97^{-2}	5.29^{-3}	4^{+0}	2^{-9}	5.10^{-8}	9.09^{-9}	3.09^{-8}	5.51^{-9}
^{91}Sr	4.94^{-2}	2.64^{+1}	4^{+3}	1^{-6}	8.49^{-8}	4.53^{-5}	1.63^{-13}	8.70^{-11}
^{92}Sr	5.04^{-2}	4.40^{+1}	7^{+3}	3^{-6}	8.66^{-8}	7.56^{-5}	1.58^{-14}	1.38^{-11}
^{160}Tb	2.73^{-10}	4.27^{-9}	2^{+2}	9^{-8}	4.69^{-16}	7.33^{-15}	1.77^{-18}	2.77^{-17}

Continued on next page...

Table A.3: Particulate fission products listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{161}Tb	7.35^{-7}	1.01^{-4}	2^{+3}	7^{-7}	1.26^{-12}	1.73^{-10}	6.04^{-17}	8.30^{-15}
^{99m}Tc	4.43^{-2}	9.79^{+0}	2^{+5}	6^{-5}	7.61^{-8}	1.68^{-5}	1.54^{-15}	3.39^{-13}
^{99}Tc	6.59^{-6}	2.57^{-7}	7^{+2}	3^{-7}	1.13^{-11}	4.41^{-13}	4.63^{-14}	1.80^{-15}
^{101}Tc	4.23^{-2}	4.23^{+1}	3^{+5}	1^{-4}	7.27^{-8}	7.27^{-5}	3.47^{-17}	3.47^{-14}
^{104}Tc	1.53^{-2}	1.53^{+1}	7^{+4}	3^{-5}	2.63^{-8}	2.63^{-5}	5.35^{-17}	5.35^{-14}
^{123m}Te	4.21^{-13}	4.01^{-12}	2^{+2}	9^{-8}	7.23^{-19}	6.89^{-18}	4.08^{-21}	3.89^{-20}
^{123}Te	2.39^{-24}	1.30^{-25}	2^{+2}	8^{-8}	4.11^{-30}	2.23^{-31}	—	—
^{125m}Te	5.83^{-5}	1.02^{-6}	4^{+2}	2^{-7}	1.00^{-10}	1.75^{-12}	1.38^{-13}	2.41^{-15}
^{127m}Te	1.53^{-4}	4.72^{-4}	3^{+2}	1^{-7}	2.63^{-10}	8.11^{-10}	1.25^{-12}	3.86^{-12}
^{127}Te	1.09^{-3}	1.61^{-1}	2^{+4}	7^{-6}	1.87^{-9}	2.77^{-7}	5.13^{-16}	7.58^{-14}
^{129m}Te	7.91^{-4}	2.36^{-2}	2^{+2}	1^{-7}	1.36^{-9}	4.05^{-8}	2.20^{-12}	6.55^{-11}
^{129}Te	5.25^{-3}	3.11^{+0}	6^{+4}	3^{-5}	9.02^{-9}	5.34^{-6}	7.03^{-17}	4.17^{-14}
^{131m}Te	3.04^{-3}	1.15^{+0}	4^{+2}	2^{-7}	5.22^{-9}	1.98^{-6}	1.58^{-13}	5.98^{-11}
^{131}Te	2.13^{-2}	2.09^{+1}	5^{+3}	2^{-6}	3.66^{-8}	3.59^{-5}	1.54^{-15}	1.51^{-12}
^{132}Te	3.54^{-2}	8.32^{+0}	2^{+2}	9^{-8}	6.08^{-8}	1.43^{-5}	1.05^{-11}	2.46^{-9}
^{133m}Te	2.56^{-2}	2.55^{+1}	5^{+3}	2^{-6}	4.40^{-8}	4.38^{-5}	4.10^{-15}	4.08^{-12}
^{133}Te	3.20^{-2}	3.20^{+1}	2^{+4}	9^{-6}	5.50^{-8}	5.50^{-5}	2.57^{-16}	2.57^{-13}
^{134}Te	5.69^{-2}	5.69^{+1}	2^{+4}	1^{-5}	9.77^{-8}	9.77^{-5}	1.38^{-15}	1.38^{-12}
^{90m}Y	1.06^{-7}	8.89^{-5}	1^{+4}	5^{-6}	1.82^{-13}	1.53^{-10}	2.35^{-20}	1.97^{-17}
^{90}Y	2.97^{-2}	2.59^{-3}	6^{+2}	3^{-7}	5.10^{-8}	4.45^{-9}	2.20^{-12}	1.92^{-13}
^{91m}Y	2.86^{-2}	1.41^{+1}	2^{+5}	7^{-5}	4.91^{-8}	2.42^{-5}	1.17^{-16}	5.78^{-14}
^{91}Y	4.94^{-2}	7.71^{-1}	1^{+2}	5^{-8}	8.49^{-8}	1.32^{-6}	4.75^{-10}	7.41^{-9}
^{92}Y	5.05^{-2}	2.81^{+1}	8^{+3}	3^{-6}	8.67^{-8}	4.83^{-5}	2.07^{-14}	1.15^{-11}
^{93}Y	5.44^{-2}	2.86^{+1}	2^{+3}	1^{-6}	9.34^{-8}	4.91^{-5}	1.92^{-13}	1.01^{-10}
^{94}Y	5.26^{-2}	5.26^{+1}	8^{+4}	3^{-5}	9.04^{-8}	9.04^{-5}	1.90^{-16}	1.90^{-13}
^{95}Y	5.35^{-2}	5.35^{+1}	1^{+5}	6^{-5}	9.19^{-8}	9.19^{-5}	5.41^{-17}	5.41^{-14}
^{72}Zn	2.08^{-7}	6.47^{-5}	1^{+3}	5^{-7}	3.57^{-13}	1.11^{-10}	6.71^{-18}	2.09^{-15}
^{93}Zr	9.85^{-7}	9.29^{-8}	6^{+0}	3^{-9}	1.69^{-12}	1.60^{-13}	6.92^{-13}	6.52^{-14}
^{95}Zr	5.38^{-2}	9.72^{-1}	1^{+2}	5^{-8}	9.24^{-8}	1.67^{-6}	5.62^{-10}	1.02^{-8}
^{97}Zr	4.92^{-2}	2.18^{+1}	1^{+3}	5^{-7}	8.45^{-8}	3.74^{-5}	5.73^{-13}	2.54^{-10}

Table A.4: ALI and CEDE fission product analysis for particulates not listed in 10CFR20 Appendix B [35, 36]. The units for the ORIGEN Activities are Ci/W-tonne while the ALI and MPFD^3-T Activity are μCi and CEDE (Committed Effective Dose Equivalent) is rem. Read 3.20^{-17} as 3.20×10^{-17}.

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{108}Ag	3.20^{-17}	1.15^{-18}	2^{+2}	1^{-7}	5.50^{-23}	1.98^{-24}	4.39^{-30}	1.58^{-31}
^{109m}Ag	2.50^{-4}	1.19^{-1}	2^{+2}	1^{-7}	4.29^{-10}	2.04^{-7}	9.59^{-18}	4.56^{-15}
^{110}Ag	2.94^{-11}	2.26^{-8}	2^{+2}	1^{-7}	5.05^{-17}	3.88^{-14}	6.97^{-25}	5.36^{-22}
^{111m}Ag	1.64^{-4}	1.63^{-1}	2^{+2}	1^{-7}	2.82^{-10}	2.80^{-7}	1.03^{-17}	1.02^{-14}
^{113m}Ag	1.04^{-5}	1.04^{-2}	2^{+2}	1^{-7}	1.79^{-11}	1.79^{-8}	6.81^{-19}	6.81^{-16}
^{113}Ag	9.29^{-5}	6.30^{-2}	2^{-1}	1^{-10}	1.60^{-10}	1.08^{-7}	1.71^{-12}	1.16^{-9}
^{114}Ag	9.42^{-5}	9.42^{-2}	2^{+2}	1^{-7}	1.62^{-10}	1.62^{-7}	4.08^{-19}	4.08^{-16}
^{115m}Ag	2.38^{-5}	2.38^{-2}	2^{+2}	1^{-7}	4.09^{-11}	4.09^{-8}	4.28^{-19}	4.28^{-16}
^{116m}Ag	4.42^{-5}	4.42^{-2}	2^{+2}	1^{-7}	7.59^{-11}	7.59^{-8}	4.27^{-19}	4.27^{-16}
^{116}Ag	4.42^{-5}	4.42^{-2}	2^{+2}	1^{-7}	7.59^{-11}	7.59^{-8}	6.84^{-18}	6.84^{-15}
^{117m}Ag	4.51^{-5}	4.51^{-2}	2^{+2}	1^{-7}	7.75^{-11}	7.75^{-8}	2.30^{-19}	2.30^{-16}
^{117}Ag	4.51^{-5}	4.51^{-2}	2^{+2}	1^{-7}	7.75^{-11}	7.75^{-8}	3.16^{-18}	3.16^{-15}
^{118m}Ag	3.26^{-5}	3.26^{-2}	2^{+2}	1^{-7}	5.60^{-11}	5.60^{-8}	6.27^{-20}	6.27^{-17}
^{118}Ag	7.46^{-5}	7.46^{-2}	2^{+2}	1^{-7}	1.28^{-10}	1.28^{-7}	2.87^{-19}	2.87^{-16}
^{119}Ag	6.74^{-5}	6.74^{-2}	2^{+2}	1^{-7}	1.16^{-10}	1.16^{-7}	1.36^{-19}	1.36^{-16}
^{120}Ag	4.68^{-5}	4.68^{-2}	2^{+2}	1^{-7}	8.04^{-11}	8.04^{-8}	5.41^{-20}	5.41^{-17}
^{121}Ag	2.87^{-5}	2.87^{-2}	2^{+2}	1^{-7}	4.93^{-11}	4.93^{-8}	2.21^{-20}	2.21^{-17}
^{122}Ag	1.56^{-5}	1.56^{-2}	2^{+2}	1^{-7}	2.68^{-11}	2.68^{-8}	7.22^{-21}	7.22^{-18}
^{123}Ag	7.84^{-6}	7.84^{-3}	2^{+2}	1^{-7}	1.35^{-11}	1.35^{-8}	2.94^{-21}	2.94^{-18}
^{124}Ag	3.73^{-6}	3.73^{-3}	2^{+2}	1^{-7}	6.41^{-12}	6.41^{-9}	6.18^{-22}	6.18^{-19}
^{125}Ag	7.50^{-7}	7.50^{-4}	2^{+2}	1^{-7}	1.29^{-12}	1.29^{-9}	1.20^{-22}	1.20^{-19}
^{126}Ag	3.36^{-7}	3.36^{-4}	2^{+2}	1^{-7}	5.77^{-13}	5.77^{-10}	3.30^{-23}	3.30^{-20}
^{128}Ag	9.96^{-9}	9.96^{-6}	2^{+2}	1^{-7}	1.71^{-14}	1.71^{-11}	5.56^{-25}	5.56^{-22}
^{79}As	4.26^{-4}	4.26^{-1}	2^{+2}	1^{-7}	7.32^{-10}	7.32^{-7}	2.22^{-16}	2.22^{-13}
^{80}As	1.02^{-3}	1.02^{+0}	2^{+2}	1^{-7}	1.75^{-9}	1.75^{-6}	3.14^{-17}	3.14^{-14}
^{81}As	1.68^{-3}	1.68^{+0}	2^{+2}	1^{-7}	2.89^{-9}	2.89^{-6}	5.38^{-17}	5.38^{-14}
^{82m}As	6.89^{-4}	6.89^{-1}	2^{+2}	1^{-7}	1.18^{-9}	1.18^{-6}	9.29^{-18}	9.29^{-15}
^{82}As	1.81^{-3}	1.81^{+0}	2^{+2}	1^{-7}	3.11^{-9}	3.11^{-6}	3.31^{-17}	3.31^{-14}
^{83}As	3.54^{-3}	3.54^{+0}	2^{+2}	1^{-7}	6.08^{-9}	6.08^{-6}	4.42^{-17}	4.42^{-14}
^{84}As	2.54^{-3}	2.54^{+0}	2^{+2}	1^{-7}	4.36^{-9}	4.36^{-6}	1.10^{-17}	1.10^{-14}
^{85}As	1.71^{-3}	1.71^{+0}	2^{+2}	1^{-7}	2.94^{-9}	2.94^{-6}	3.34^{-18}	3.34^{-15}
^{86}As	9.63^{-4}	9.63^{-1}	2^{+2}	1^{-7}	1.65^{-9}	1.65^{-6}	8.33^{-19}	8.33^{-16}
^{87}As	5.89^{-4}	5.89^{-1}	2^{+2}	1^{-7}	1.01^{-9}	1.01^{-6}	4.25^{-19}	4.25^{-16}
^{88}As	2.05^{-5}	2.05^{-2}	2^{+2}	1^{-7}	3.52^{-11}	3.52^{-8}	5.92^{-27}	5.92^{-24}
^{89}As	1.62^{-6}	1.62^{-3}	2^{+2}	1^{-7}	2.78^{-12}	2.78^{-9}	4.68^{-28}	4.68^{-25}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{136m}Ba	7.32^{-6}	5.77^{-4}	2^{+2}	1^{-7}	1.26^{-11}	9.91^{-10}	2.17^{-21}	1.71^{-19}
^{137m}Ba	2.93^{-2}	7.26^{-3}	2^{+2}	1^{-7}	5.03^{-8}	1.25^{-8}	4.31^{-15}	1.07^{-15}
^{143}Ba	4.43^{-2}	4.43^{+1}	2^{+2}	1^{-7}	7.61^{-8}	7.61^{-5}	6.41^{-16}	6.41^{-13}
^{144}Ba	3.57^{-2}	3.57^{+1}	2^{+2}	1^{-7}	6.13^{-8}	6.13^{-5}	3.95^{-16}	3.95^{-13}
^{145}Ba	1.65^{-2}	1.65^{+1}	2^{+2}	1^{-7}	2.83^{-8}	2.83^{-5}	6.36^{-17}	6.36^{-14}
^{146}Ba	5.54^{-3}	5.54^{+0}	2^{+2}	1^{-7}	9.52^{-9}	9.52^{-6}	1.18^{-17}	1.18^{-14}
^{147}Ba	1.08^{-3}	1.08^{+0}	2^{+2}	1^{-7}	1.86^{-9}	1.86^{-6}	7.28^{-19}	7.28^{-16}
^{148}Ba	1.25^{-4}	1.25^{-1}	2^{+2}	1^{-7}	2.15^{-10}	2.15^{-7}	5.66^{-20}	5.66^{-17}
^{149}Ba	7.54^{-6}	7.54^{-3}	2^{+2}	1^{-7}	1.30^{-11}	1.30^{-8}	2.50^{-21}	2.50^{-18}
^{150}Ba	3.40^{-7}	3.40^{-4}	2^{+2}	1^{-7}	5.84^{-13}	5.84^{-10}	9.82^{-23}	9.82^{-20}
^{152}Ba	1.46^{-10}	1.46^{-7}	2^{+2}	1^{-7}	2.51^{-16}	2.51^{-13}	1.41^{-26}	1.41^{-23}
^{111m}Cd	1.24^{-13}	1.24^{-10}	2^{+2}	1^{-7}	2.13^{-19}	2.13^{-16}	3.47^{-25}	3.47^{-22}
^{118}Cd	1.01^{-4}	1.01^{-1}	2^{+2}	1^{-7}	1.73^{-10}	1.73^{-7}	2.93^{-16}	2.93^{-13}
^{119m}Cd	4.66^{-5}	4.66^{-2}	2^{+2}	1^{-7}	8.00^{-11}	8.00^{-8}	5.93^{-18}	5.93^{-15}
^{119}Cd	4.66^{-5}	4.66^{-2}	2^{+2}	1^{-7}	8.00^{-11}	8.00^{-8}	7.25^{-18}	7.25^{-15}
^{120}Cd	9.61^{-5}	9.61^{-2}	2^{+2}	1^{-7}	1.65^{-10}	1.65^{-7}	4.70^{-18}	4.70^{-15}
^{121}Cd	9.71^{-5}	9.71^{-2}	2^{+2}	1^{-7}	1.67^{-10}	1.67^{-7}	1.26^{-18}	1.26^{-15}
^{122}Cd	9.43^{-5}	9.43^{-2}	2^{+2}	1^{-7}	1.62^{-10}	1.62^{-7}	5.26^{-19}	5.26^{-16}
^{123}Cd	8.73^{-5}	8.73^{-2}	2^{+2}	1^{-7}	1.50^{-10}	1.50^{-7}	1.76^{-19}	1.76^{-16}
^{124}Cd	9.24^{-5}	9.24^{-2}	2^{+2}	1^{-7}	1.59^{-10}	1.59^{-7}	8.00^{-20}	8.00^{-17}
^{125}Cd	4.38^{-5}	4.38^{-2}	2^{+2}	1^{-7}	7.52^{-11}	7.52^{-8}	2.74^{-20}	2.74^{-17}
^{126}Cd	5.56^{-5}	5.56^{-2}	2^{+2}	1^{-7}	9.55^{-11}	9.55^{-8}	2.73^{-20}	2.73^{-17}
^{127}Cd	2.82^{-5}	2.82^{-2}	2^{+2}	1^{-7}	4.84^{-11}	4.84^{-8}	1.00^{-20}	1.00^{-17}
^{128}Cd	1.33^{-5}	1.33^{-2}	2^{+2}	1^{-7}	2.28^{-11}	2.28^{-8}	3.59^{-21}	3.59^{-18}
^{129}Cd	2.49^{-6}	2.49^{-3}	2^{+2}	1^{-7}	4.28^{-12}	4.28^{-9}	6.48^{-22}	6.48^{-19}
^{130}Cd	6.28^{-6}	6.28^{-3}	2^{+2}	1^{-7}	1.08^{-11}	1.08^{-8}	9.82^{-22}	9.82^{-19}
^{131}Cd	5.82^{-7}	5.82^{-4}	2^{+2}	1^{-7}	1.00^{-12}	1.00^{-9}	3.81^{-23}	3.81^{-20}
^{132}Cd	5.21^{-8}	5.21^{-5}	2^{+2}	1^{-7}	8.95^{-14}	8.95^{-11}	4.85^{-24}	4.85^{-21}
^{142}Ce	1.31^{-11}	1.45^{-12}	2^{-1}	1^{-10}	2.25^{-17}	2.49^{-18}	—	—
^{145}Ce	3.29^{-2}	3.29^{+1}	2^{+2}	1^{-7}	5.65^{-8}	5.65^{-5}	5.50^{-15}	5.50^{-12}
^{146}Ce	2.50^{-2}	2.50^{+1}	2^{+2}	1^{-7}	4.29^{-8}	4.29^{-5}	1.95^{-14}	1.95^{-11}
^{147}Ce	1.85^{-2}	1.85^{+1}	2^{+2}	1^{-7}	3.18^{-8}	3.18^{-5}	9.97^{-16}	9.97^{-13}
^{148}Ce	1.28^{-2}	1.28^{+1}	2^{+2}	1^{-7}	2.20^{-8}	2.20^{-5}	5.92^{-16}	5.92^{-13}
^{149}Ce	6.53^{-3}	6.53^{+0}	2^{+2}	1^{-7}	1.12^{-8}	1.12^{-5}	3.27^{-17}	3.27^{-14}
^{150}Ce	2.44^{-3}	2.44^{+0}	2^{+2}	1^{-7}	4.19^{-9}	4.19^{-6}	1.03^{-17}	1.03^{-14}
^{151}Ce	6.16^{-4}	6.16^{-1}	2^{+2}	1^{-7}	1.06^{-9}	1.06^{-6}	5.95^{-19}	5.95^{-16}
^{152}Ce	1.10^{-4}	1.10^{-1}	2^{+2}	1^{-7}	1.89^{-10}	1.89^{-7}	1.48^{-19}	1.48^{-16}
^{153}Ce	1.05^{-5}	1.05^{-2}	2^{+2}	1^{-7}	1.80^{-11}	1.80^{-8}	1.01^{-27}	1.01^{-24}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{154}Ce	6.57^{-7}	6.57^{-4}	2^{+2}	1^{-7}	1.13^{-12}	1.13^{-9}	9.51^{-29}	9.51^{-26}
^{155}Ce	2.70^{-8}	2.70^{-5}	2^{+2}	1^{-7}	4.64^{-14}	4.64^{-11}	3.91^{-30}	3.91^{-27}
^{156}Ce	6.63^{-10}	6.63^{-7}	2^{+2}	1^{-7}	1.14^{-15}	1.14^{-12}	9.60^{-26}	9.60^{-23}
^{157}Ce	1.23^{-11}	1.23^{-8}	2^{+2}	1^{-7}	2.11^{-17}	2.11^{-14}	5.93^{-28}	5.93^{-25}
^{72}Co	2.43^{-9}	2.43^{-6}	2^{+2}	1^{-7}	4.17^{-15}	4.17^{-12}	1.45^{-25}	1.45^{-22}
^{73}Co	1.44^{-10}	1.44^{-7}	2^{+2}	1^{-7}	2.47^{-16}	2.47^{-13}	5.69^{-27}	5.69^{-24}
^{74}Co	3.57^{-10}	3.57^{-7}	2^{+2}	1^{-7}	6.13^{-16}	6.13^{-13}	5.17^{-32}	5.17^{-29}
^{75}Co	6.00^{-11}	6.00^{-8}	2^{+2}	1^{-7}	1.03^{-16}	1.03^{-13}	8.69^{-33}	8.69^{-30}
^{138m}Cs	2.11^{-3}	2.11^{+0}	2^{+2}	1^{-7}	3.62^{-9}	3.62^{-6}	3.53^{-16}	3.53^{-13}
^{139}Cs	5.37^{-2}	5.37^{+1}	2^{+2}	1^{-7}	9.22^{-8}	9.22^{-5}	2.92^{-14}	2.92^{-11}
^{140}Cs	4.88^{-2}	4.88^{+1}	2^{+2}	1^{-7}	8.38^{-8}	8.38^{-5}	2.99^{-15}	2.99^{-12}
^{141}Cs	3.70^{-2}	3.70^{+1}	2^{+2}	1^{-7}	6.36^{-8}	6.36^{-5}	8.87^{-16}	8.87^{-13}
^{142}Cs	2.36^{-2}	2.36^{+1}	2^{+2}	1^{-7}	4.05^{-8}	4.05^{-5}	4.08^{-17}	4.08^{-14}
^{143}Cs	1.30^{-2}	1.30^{+1}	2^{+2}	1^{-7}	2.23^{-8}	2.23^{-5}	2.23^{-17}	2.23^{-14}
^{144}Cs	2.41^{-3}	2.41^{+0}	2^{+2}	1^{-7}	4.14^{-9}	4.14^{-6}	2.35^{-18}	2.35^{-15}
^{145}Cs	5.69^{-4}	5.69^{-1}	2^{+2}	1^{-7}	9.77^{-10}	9.77^{-7}	3.18^{-19}	3.18^{-16}
^{146}Cs	6.21^{-5}	6.21^{-2}	2^{+2}	1^{-7}	1.07^{-10}	1.07^{-7}	1.97^{-20}	1.97^{-17}
^{147}Cs	5.02^{-6}	5.02^{-3}	2^{+2}	1^{-7}	8.62^{-12}	8.62^{-9}	1.14^{-21}	1.14^{-18}
^{148}Cs	1.46^{-7}	1.46^{-4}	2^{+2}	1^{-7}	2.51^{-13}	2.51^{-10}	2.05^{-23}	2.05^{-20}
^{150}Cs	3.28^{-11}	3.28^{-8}	2^{+2}	1^{-7}	5.63^{-17}	5.63^{-14}	1.58^{-27}	1.58^{-24}
^{66}Cu	3.88^{-14}	3.88^{-11}	2^{+2}	1^{-7}	6.67^{-20}	6.67^{-17}	1.92^{-31}	1.92^{-28}
^{72}Cu	1.87^{-7}	1.87^{-4}	2^{+2}	1^{-7}	3.21^{-13}	3.21^{-10}	1.19^{-21}	1.19^{-18}
^{73}Cu	3.84^{-7}	3.84^{-4}	2^{+2}	1^{-7}	6.60^{-13}	6.60^{-10}	1.44^{-21}	1.44^{-18}
^{74}Cu	1.18^{-6}	1.18^{-3}	2^{+2}	1^{-7}	2.03^{-12}	2.03^{-9}	1.81^{-21}	1.81^{-18}
^{75}Cu	1.72^{-6}	1.72^{-3}	2^{+2}	1^{-7}	2.95^{-12}	2.95^{-9}	2.03^{-21}	2.03^{-18}
^{76}Cu	1.76^{-6}	1.76^{-3}	2^{+2}	1^{-7}	3.02^{-12}	3.02^{-9}	1.09^{-21}	1.09^{-18}
^{77}Cu	7.18^{-7}	7.18^{-4}	2^{+2}	1^{-7}	1.23^{-12}	1.23^{-9}	3.24^{-22}	3.24^{-19}
^{78}Cu	3.23^{-7}	3.23^{-4}	2^{+2}	1^{-7}	5.55^{-13}	5.55^{-10}	1.06^{-22}	1.06^{-19}
^{79}Cu	3.85^{-8}	3.85^{-5}	2^{+2}	1^{-7}	6.61^{-14}	6.61^{-11}	6.66^{-24}	6.66^{-21}
^{80}Cu	5.53^{-9}	5.53^{-6}	2^{+2}	1^{-7}	9.50^{-15}	9.50^{-12}	2.40^{-30}	2.40^{-27}
^{165m}Dy	5.70^{-9}	5.70^{-6}	2^{+2}	1^{-7}	9.79^{-15}	9.79^{-12}	4.14^{-22}	4.14^{-19}
^{167m}Er	3.31^{-14}	3.31^{-11}	2^{+2}	1^{-7}	5.69^{-20}	5.69^{-17}	7.33^{-29}	7.33^{-26}
^{159}Eu	8.64^{-6}	8.64^{-3}	2^{+2}	1^{-7}	1.48^{-11}	1.48^{-8}	8.99^{-18}	8.99^{-15}
^{160}Eu	3.11^{-6}	3.11^{-3}	2^{+2}	1^{-7}	5.34^{-12}	5.34^{-9}	1.14^{-19}	1.14^{-16}
^{161}Eu	6.20^{-7}	6.20^{-4}	2^{+2}	1^{-7}	1.07^{-12}	1.07^{-9}	1.55^{-20}	1.55^{-17}
^{162}Eu	1.46^{-7}	1.46^{-4}	2^{+2}	1^{-7}	2.51^{-13}	2.51^{-10}	1.49^{-21}	1.49^{-18}
^{163}Eu	2.76^{-8}	2.76^{-5}	2^{+2}	1^{-7}	4.74^{-14}	4.74^{-11}	1.59^{-22}	1.59^{-19}
^{164}Eu	3.18^{-9}	3.18^{-6}	2^{+2}	1^{-7}	5.46^{-15}	5.46^{-12}	6.11^{-24}	6.11^{-21}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{165}Eu	4.75^{-10}	4.75^{-7}	2^{+2}	1^{-7}	8.16^{-16}	8.16^{-13}	4.58^{-25}	4.58^{-22}
^{74}Ga	2.91^{-6}	2.91^{-3}	2^{+2}	1^{-7}	5.00^{-12}	5.00^{-9}	1.36^{-18}	1.36^{-15}
^{75}Ga	9.38^{-6}	9.38^{-3}	2^{+2}	1^{-7}	1.61^{-11}	1.61^{-8}	1.14^{-18}	1.14^{-15}
^{76}Ga	2.95^{-5}	2.95^{-2}	2^{+2}	1^{-7}	5.07^{-11}	5.07^{-8}	8.27^{-19}	8.27^{-16}
^{77}Ga	6.06^{-5}	6.06^{-2}	2^{+2}	1^{-7}	1.04^{-10}	1.04^{-7}	7.56^{-19}	7.56^{-16}
^{78}Ga	1.36^{-4}	1.36^{-1}	2^{+2}	1^{-7}	2.34^{-10}	2.34^{-7}	6.66^{-19}	6.66^{-16}
^{79}Ga	1.27^{-4}	1.27^{-1}	2^{+2}	1^{-7}	2.18^{-10}	2.18^{-7}	3.67^{-19}	3.67^{-16}
^{80}Ga	1.78^{-4}	1.78^{-1}	2^{+2}	1^{-7}	3.06^{-10}	3.06^{-7}	2.84^{-19}	2.84^{-16}
^{81}Ga	9.48^{-5}	9.48^{-2}	2^{+2}	1^{-7}	1.63^{-10}	1.63^{-7}	1.12^{-19}	1.12^{-16}
^{82}Ga	3.14^{-5}	3.14^{-2}	2^{+2}	1^{-7}	5.39^{-11}	5.39^{-8}	1.81^{-20}	1.81^{-17}
^{83}Ga	7.80^{-6}	7.80^{-3}	2^{+2}	1^{-7}	1.34^{-11}	1.34^{-8}	2.33^{-21}	2.33^{-18}
^{84}Ga	3.70^{-7}	3.70^{-4}	2^{+2}	1^{-7}	6.36^{-13}	6.36^{-10}	3.03^{-23}	3.03^{-20}
^{161}Gd	7.34^{-7}	7.34^{-4}	2^{+2}	1^{-7}	1.26^{-12}	1.26^{-9}	1.55^{-19}	1.55^{-16}
^{162}Gd	2.37^{-7}	2.37^{-4}	2^{+2}	1^{-7}	4.07^{-13}	4.07^{-10}	1.15^{-19}	1.15^{-16}
^{163}Gd	7.89^{-8}	7.89^{-5}	2^{+2}	1^{-7}	1.36^{-13}	1.36^{-10}	5.17^{-21}	5.17^{-18}
^{164}Gd	1.96^{-8}	1.96^{-5}	2^{+2}	1^{-7}	3.37^{-14}	3.37^{-11}	8.50^{-22}	8.50^{-19}
^{165}Gd	7.25^{-9}	7.25^{-6}	2^{+2}	1^{-7}	1.25^{-14}	1.25^{-11}	7.18^{-23}	7.18^{-20}
^{73m}Ge	8.84^{-7}	6.21^{-4}	2^{+2}	1^{-7}	1.52^{-12}	1.07^{-9}	4.25^{-22}	2.99^{-19}
^{75m}Ge	3.87^{-7}	3.87^{-4}	2^{+2}	1^{-7}	6.65^{-13}	6.65^{-10}	1.78^{-20}	1.78^{-17}
^{77m}Ge	5.67^{-5}	5.67^{-2}	2^{+2}	1^{-7}	9.74^{-11}	9.74^{-8}	2.89^{-18}	2.89^{-15}
^{79}Ge	3.38^{-4}	3.38^{-1}	2^{+2}	1^{-7}	5.81^{-10}	5.81^{-7}	6.19^{-18}	6.19^{-15}
^{80}Ge	8.99^{-4}	8.99^{-1}	2^{+2}	1^{-7}	1.54^{-9}	1.54^{-6}	2.55^{-17}	2.55^{-14}
^{81}Ge	1.18^{-3}	1.18^{+0}	2^{+2}	1^{-7}	2.03^{-9}	2.03^{-6}	8.64^{-18}	8.64^{-15}
^{82}Ge	1.12^{-3}	1.12^{+0}	2^{+2}	1^{-7}	1.92^{-9}	1.92^{-6}	4.96^{-18}	4.96^{-15}
^{83}Ge	8.17^{-4}	8.17^{-1}	2^{+2}	1^{-7}	1.40^{-9}	1.40^{-6}	1.45^{-18}	1.45^{-15}
^{84}Ge	1.34^{-4}	1.34^{-1}	2^{+2}	1^{-7}	2.30^{-10}	2.30^{-7}	1.25^{-19}	1.25^{-16}
^{85}Ge	5.21^{-5}	5.21^{-2}	2^{+2}	1^{-7}	8.95^{-11}	8.95^{-8}	2.68^{-20}	2.68^{-17}
^{86}Ge	9.54^{-6}	9.54^{-3}	2^{+2}	1^{-7}	1.64^{-11}	1.64^{-8}	1.38^{-27}	1.38^{-24}
^{87}Ge	1.66^{-6}	1.66^{-3}	2^{+2}	1^{-7}	2.85^{-12}	2.85^{-9}	2.24^{-22}	2.24^{-19}
^{88}Ge	1.57^{-8}	1.57^{-5}	2^{+2}	1^{-7}	2.70^{-14}	2.70^{-11}	4.53^{-30}	4.53^{-27}
^{114}In	2.28^{-13}	1.18^{-10}	2^{+2}	1^{-7}	3.92^{-19}	2.03^{-16}	1.58^{-26}	8.17^{-24}
^{116}In	1.31^{-10}	1.30^{-7}	2^{+2}	1^{-7}	2.25^{-16}	2.23^{-13}	1.78^{-24}	1.76^{-21}
^{118m}In	2.30^{-8}	2.30^{-5}	2^{+2}	1^{-7}	3.95^{-14}	3.95^{-11}	5.91^{-21}	5.91^{-18}
^{118}In	1.01^{-4}	1.01^{-1}	2^{+2}	1^{-7}	1.73^{-10}	1.73^{-7}	4.87^{-19}	4.87^{-16}
^{119}In	2.70^{-5}	2.70^{-2}	2^{-1}	1^{-10}	4.64^{-11}	4.64^{-8}	3.75^{-15}	3.75^{-12}
^{120m}In	4.91^{-5}	4.91^{-2}	2^{-1}	1^{-10}	8.43^{-11}	8.43^{-8}	2.23^{-15}	2.23^{-12}
^{120}In	4.91^{-5}	4.91^{-2}	2^{-1}	1^{-10}	8.43^{-11}	8.43^{-8}	1.42^{-16}	1.42^{-13}
^{121m}In	2.42^{-5}	2.42^{-2}	2^{-1}	1^{-10}	4.16^{-11}	4.16^{-8}	5.46^{-15}	5.46^{-12}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{121}In	8.32^{-5}	8.32^{-2}	2^{+2}	1^{-7}	1.43^{-10}	1.43^{-7}	1.84^{-18}	1.84^{-15}
^{122m}In	8.10^{-6}	8.10^{-3}	2^{-1}	1^{-10}	1.39^{-11}	1.39^{-8}	8.02^{-17}	8.02^{-14}
^{122}In	1.02^{-4}	1.02^{-1}	2^{-1}	1^{-10}	1.75^{-10}	1.75^{-7}	1.48^{-16}	1.48^{-13}
^{123m}In	3.64^{-5}	3.64^{-2}	2^{+2}	1^{-7}	6.25^{-11}	6.25^{-8}	1.68^{-18}	1.68^{-15}
^{123}In	8.43^{-5}	8.43^{-2}	2^{+2}	1^{-7}	1.45^{-10}	1.45^{-7}	4.87^{-19}	4.87^{-16}
^{124}In	1.72^{-4}	1.72^{-1}	2^{+2}	1^{-7}	2.95^{-10}	2.95^{-7}	5.29^{-19}	5.29^{-16}
^{125m}In	5.56^{-5}	5.56^{-2}	2^{+2}	1^{-7}	9.55^{-11}	9.55^{-8}	6.52^{-19}	6.52^{-16}
^{125}In	7.31^{-5}	7.31^{-2}	2^{+2}	1^{-7}	1.26^{-10}	1.26^{-7}	1.55^{-19}	1.55^{-16}
^{126}In	3.35^{-4}	3.35^{-1}	2^{+2}	1^{-7}	5.75^{-10}	5.75^{-7}	4.93^{-19}	4.93^{-16}
^{127m}In	1.89^{-4}	1.89^{-1}	2^{+2}	1^{-7}	3.25^{-10}	3.25^{-7}	6.84^{-19}	6.84^{-16}
^{127}In	1.89^{-4}	1.89^{-1}	2^{+2}	1^{-7}	3.25^{-10}	3.25^{-7}	2.04^{-19}	2.04^{-16}
^{128}In	4.74^{-4}	4.74^{-1}	2^{+2}	1^{-7}	8.14^{-10}	8.14^{-7}	4.10^{-19}	4.10^{-16}
^{129}In	3.20^{-4}	3.20^{-1}	2^{+2}	1^{-7}	5.50^{-10}	5.50^{-7}	1.82^{-19}	1.82^{-16}
^{130}In	5.81^{-4}	5.81^{-1}	2^{+2}	1^{-7}	9.98^{-10}	9.98^{-7}	1.79^{-19}	1.79^{-16}
^{131}In	1.89^{-4}	1.89^{-1}	2^{+2}	1^{-7}	3.25^{-10}	3.25^{-7}	5.09^{-20}	5.09^{-17}
^{132}In	5.64^{-5}	5.64^{-2}	2^{+2}	1^{-7}	9.69^{-11}	9.69^{-8}	1.20^{-20}	1.20^{-17}
^{133}In	3.25^{-6}	3.25^{-3}	2^{+2}	1^{-7}	5.58^{-12}	5.58^{-9}	5.16^{-22}	5.16^{-19}
^{134}In	4.86^{-8}	4.86^{-5}	2^{+2}	1^{-7}	8.35^{-14}	8.35^{-11}	6.55^{-24}	6.55^{-21}
^{144}La	4.51^{-2}	4.51^{+1}	2^{+2}	1^{-7}	7.75^{-8}	7.75^{-5}	1.74^{-15}	1.74^{-12}
^{145}La	3.09^{-2}	3.09^{+1}	2^{+2}	1^{-7}	5.31^{-8}	5.31^{-5}	7.43^{-16}	7.43^{-13}
^{146}La	1.92^{-2}	1.92^{+1}	2^{+2}	1^{-7}	3.30^{-8}	3.30^{-5}	1.16^{-16}	1.16^{-13}
^{147}La	8.91^{-3}	8.91^{+0}	2^{+2}	1^{-7}	1.53^{-8}	1.53^{-5}	3.52^{-17}	3.52^{-14}
^{148}La	2.97^{-3}	2.97^{+0}	2^{+2}	1^{-7}	5.10^{-9}	5.10^{-6}	4.08^{-18}	4.08^{-15}
^{149}La	5.88^{-4}	5.88^{-1}	2^{+2}	1^{-7}	1.01^{-9}	1.01^{-6}	6.80^{-19}	6.80^{-16}
^{150}La	8.23^{-5}	8.23^{-2}	2^{+2}	1^{-7}	1.41^{-10}	1.41^{-7}	6.81^{-20}	6.81^{-17}
^{151}La	6.95^{-6}	6.95^{-3}	2^{+2}	1^{-7}	1.19^{-11}	1.19^{-8}	1.01^{-27}	1.01^{-24}
^{152}La	3.95^{-7}	3.95^{-4}	2^{-1}	1^{-10}	6.79^{-13}	6.79^{-10}	5.72^{-26}	5.72^{-23}
^{153}La	1.15^{-8}	1.15^{-5}	2^{+2}	1^{-7}	1.98^{-14}	1.98^{-11}	1.66^{-30}	1.66^{-27}
^{154}La	2.37^{-10}	2.37^{-7}	2^{+2}	1^{-7}	4.07^{-16}	4.07^{-13}	2.29^{-26}	2.29^{-23}
^{102}Mo	3.52^{-2}	3.52^{+1}	2^{+2}	1^{-7}	6.05^{-8}	6.05^{-5}	2.28^{-14}	2.28^{-11}
^{103}Mo	2.61^{-2}	2.61^{+1}	2^{+2}	1^{-7}	4.48^{-8}	4.48^{-5}	1.71^{-15}	1.71^{-12}
^{104}Mo	1.48^{-2}	1.48^{+1}	2^{+2}	1^{-7}	2.54^{-8}	2.54^{-5}	8.55^{-16}	8.55^{-13}
^{105}Mo	7.63^{-3}	7.63^{+0}	2^{+2}	1^{-7}	1.31^{-8}	1.31^{-5}	3.68^{-16}	3.68^{-13}
^{106}Mo	2.87^{-3}	2.87^{+0}	2^{+2}	1^{-7}	4.93^{-9}	4.93^{-6}	2.32^{-17}	2.32^{-14}
^{107}Mo	9.47^{-4}	9.47^{-1}	2^{+2}	1^{-7}	1.63^{-9}	1.63^{-6}	3.19^{-18}	3.19^{-15}
^{108}Mo	2.04^{-4}	2.04^{-1}	2^{+2}	1^{-7}	3.50^{-10}	3.50^{-7}	2.14^{-19}	2.14^{-16}
^{109}Mo	1.92^{-5}	1.92^{-2}	2^{+2}	1^{-7}	3.30^{-11}	3.30^{-8}	9.79^{-21}	9.79^{-18}
^{110}Mo	3.21^{-6}	3.21^{-3}	2^{+2}	1^{-7}	5.51^{-12}	5.51^{-9}	8.36^{-22}	8.36^{-19}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{111}Mo	2.13^{-7}	2.13^{-4}	2^{+2}	1^{-7}	3.66^{-13}	3.66^{-10}	4.09^{-23}	4.09^{-20}
^{112}Mo	1.47^{-8}	1.47^{-5}	2^{+2}	1^{-7}	2.53^{-14}	2.53^{-11}	2.13^{-30}	2.13^{-27}
^{113}Mo	1.40^{-9}	1.40^{-6}	2^{+2}	1^{-7}	2.40^{-15}	2.40^{-12}	1.35^{-25}	1.35^{-22}
^{114}Mo	5.49^{-11}	5.49^{-8}	2^{+2}	1^{-7}	9.43^{-17}	9.43^{-14}	4.23^{-27}	4.23^{-24}
^{115}Mo	1.37^{-12}	1.37^{-9}	2^{+2}	1^{-7}	2.35^{-18}	2.35^{-15}	7.91^{-29}	7.91^{-26}
^{94m}Nb	3.82^{-9}	3.82^{-6}	2^{+2}	1^{-7}	6.56^{-15}	6.56^{-12}	1.38^{-21}	1.38^{-18}
^{97m}Nb	4.66^{-2}	2.07^{+1}	2^{+2}	1^{-7}	8.00^{-8}	3.56^{-5}	2.42^{-15}	1.08^{-12}
^{98m}Nb	2.31^{-4}	2.31^{-1}	2^{+2}	1^{-7}	3.97^{-10}	3.97^{-7}	6.80^{-16}	6.80^{-13}
^{99m}Nb	1.28^{-3}	1.28^{+0}	2^{+2}	1^{-7}	2.20^{-9}	2.20^{-6}	1.93^{-16}	1.93^{-13}
^{99}Nb	4.93^{-2}	4.93^{+1}	2^{+2}	1^{-7}	8.47^{-8}	8.47^{-5}	7.14^{-16}	7.14^{-13}
^{100m}Nb	2.63^{-2}	2.63^{+1}	2^{+2}	1^{-7}	4.52^{-8}	4.52^{-5}	7.57^{-17}	7.57^{-14}
^{100}Nb	2.63^{-2}	2.63^{+1}	2^{+2}	1^{-7}	4.52^{-8}	4.52^{-5}	3.81^{-17}	3.81^{-14}
^{101}Nb	4.13^{-2}	4.13^{+1}	2^{+2}	1^{-7}	7.09^{-8}	7.09^{-5}	2.82^{-16}	2.82^{-13}
^{102}Nb	3.17^{-2}	3.17^{+1}	2^{+2}	1^{-7}	5.45^{-8}	5.45^{-5}	3.96^{-17}	3.96^{-14}
^{103}Nb	1.85^{-2}	1.85^{+1}	2^{+2}	1^{-7}	3.18^{-8}	3.18^{-5}	2.68^{-17}	2.68^{-14}
^{104}Nb	6.68^{-3}	6.68^{+0}	2^{+2}	1^{-7}	1.15^{-8}	1.15^{-5}	3.09^{-17}	3.09^{-14}
^{105}Nb	1.54^{-3}	1.54^{+0}	2^{+2}	1^{-7}	2.65^{-9}	2.65^{-6}	4.37^{-18}	4.37^{-15}
^{106}Nb	5.67^{-4}	5.67^{-1}	2^{+2}	1^{-7}	9.74^{-10}	9.74^{-7}	5.99^{-19}	5.99^{-16}
^{107}Nb	7.65^{-5}	7.65^{-2}	2^{+2}	1^{-7}	1.31^{-10}	1.31^{-7}	2.43^{-20}	2.43^{-17}
^{108}Nb	4.60^{-6}	4.60^{-3}	2^{+2}	1^{-7}	7.90^{-12}	7.90^{-9}	8.53^{-22}	8.53^{-19}
^{109}Nb	1.25^{-7}	1.25^{-4}	2^{+2}	1^{-7}	2.15^{-13}	2.15^{-10}	2.29^{-23}	2.29^{-20}
^{110}Nb	5.69^{-9}	5.69^{-6}	2^{+2}	1^{-7}	9.77^{-15}	9.77^{-12}	9.33^{-25}	9.33^{-22}
^{111}Nb	9.20^{-11}	9.20^{-8}	2^{+2}	1^{-7}	1.58^{-16}	1.58^{-13}	7.09^{-27}	7.09^{-24}
^{144}Nd	5.86^{-16}	3.75^{-19}	2^{-1}	1^{-10}	1.01^{-21}	6.44^{-25}	1.36^{-20}	8.69^{-24}
^{152}Nd	2.20^{-3}	2.20^{+0}	2^{+2}	1^{-7}	3.78^{-9}	3.78^{-6}	1.45^{-15}	1.45^{-12}
^{153}Nd	1.22^{-3}	1.22^{+0}	2^{+2}	1^{-7}	2.10^{-9}	2.10^{-6}	3.72^{-17}	3.72^{-14}
^{154}Nd	4.63^{-4}	4.63^{-1}	2^{+2}	1^{-7}	7.95^{-10}	7.95^{-7}	1.16^{-17}	1.16^{-14}
^{155}Nd	1.39^{-4}	1.39^{-1}	2^{+2}	1^{-7}	2.39^{-10}	2.39^{-7}	1.19^{-18}	1.19^{-15}
^{156}Nd	2.79^{-5}	2.79^{-2}	2^{+2}	1^{-7}	4.79^{-11}	4.79^{-8}	8.05^{-27}	8.05^{-24}
^{157}Nd	4.25^{-6}	4.25^{-3}	2^{+2}	1^{-7}	7.30^{-12}	7.30^{-9}	4.10^{-28}	4.10^{-25}
^{158}Nd	4.85^{-7}	4.85^{-4}	2^{+2}	1^{-7}	8.33^{-13}	8.33^{-10}	2.34^{-29}	2.34^{-26}
^{159}Nd	2.43^{-8}	2.43^{-5}	2^{+2}	1^{-7}	4.17^{-14}	4.17^{-11}	1.64^{-23}	1.64^{-20}
^{160}Nd	6.25^{-9}	6.25^{-6}	2^{+2}	1^{-7}	1.07^{-14}	1.07^{-11}	1.80^{-24}	1.80^{-21}
^{161}Nd	1.11^{-11}	1.11^{-8}	2^{+2}	1^{-7}	1.91^{-17}	1.91^{-14}	2.13^{-27}	2.13^{-24}
^{72}Ni	8.44^{-8}	8.44^{-5}	2^{+2}	1^{-7}	1.45^{-13}	1.45^{-10}	1.22^{-22}	1.22^{-19}
^{73}Ni	4.13^{-8}	4.13^{-5}	2^{+2}	1^{-7}	7.09^{-14}	7.09^{-11}	3.34^{-23}	3.34^{-20}
^{74}Ni	1.15^{-7}	1.15^{-4}	2^{+2}	1^{-7}	1.98^{-13}	1.98^{-10}	7.53^{-23}	7.53^{-20}
^{75}Ni	6.15^{-8}	6.15^{-5}	2^{+2}	1^{-7}	1.06^{-13}	1.06^{-10}	3.55^{-23}	3.55^{-20}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{76}Ni	2.07^{-8}	2.07^{-5}	2^{+2}	1^{-7}	3.56^{-14}	3.56^{-11}	4.79^{-24}	4.79^{-21}
^{77}Ni	2.71^{-9}	2.71^{-6}	2^{+2}	1^{-7}	4.66^{-15}	4.66^{-12}	3.81^{-31}	3.81^{-28}
^{78}Ni	3.68^{-10}	3.68^{-7}	2^{+2}	1^{-7}	6.32^{-16}	6.32^{-13}	5.33^{-32}	5.33^{-29}
^{107m}Pd	5.87^{-12}	5.86^{-9}	2^{+2}	1^{-7}	1.01^{-17}	1.01^{-14}	1.21^{-25}	1.20^{-22}
^{109m}Pd	1.25^{-4}	1.25^{-1}	2^{+2}	1^{-7}	2.15^{-10}	2.15^{-7}	3.38^{-17}	3.38^{-14}
^{111m}Pd	1.48^{-6}	9.87^{-4}	2^{-1}	1^{-10}	2.54^{-12}	1.70^{-9}	2.82^{-14}	1.88^{-11}
^{111}Pd	1.65^{-4}	1.64^{-1}	2^{+2}	1^{-7}	2.83^{-10}	2.82^{-7}	2.10^{-16}	2.09^{-13}
^{112}Pd	1.06^{-4}	4.46^{-2}	2^{-1}	1^{-10}	1.82^{-10}	7.66^{-8}	7.73^{-12}	3.25^{-9}
^{113}Pd	1.03^{-4}	1.03^{-1}	2^{+2}	1^{-7}	1.77^{-10}	1.77^{-7}	9.68^{-18}	9.68^{-15}
^{114}Pd	9.32^{-5}	9.32^{-2}	2^{+2}	1^{-7}	1.60^{-10}	1.60^{-7}	1.33^{-17}	1.33^{-14}
^{115}Pd	8.02^{-5}	8.02^{-2}	2^{+2}	1^{-7}	1.38^{-10}	1.38^{-7}	1.93^{-18}	1.93^{-15}
^{116}Pd	7.59^{-5}	7.59^{-2}	2^{+2}	1^{-7}	1.30^{-10}	1.30^{-7}	9.28^{-19}	9.28^{-16}
^{117}Pd	8.62^{-5}	8.62^{-2}	2^{+2}	1^{-7}	1.48^{-10}	1.48^{-7}	4.15^{-19}	4.15^{-16}
^{118}Pd	3.01^{-5}	3.01^{-2}	2^{+2}	1^{-7}	5.17^{-11}	5.17^{-8}	5.51^{-20}	5.51^{-17}
^{119}Pd	1.72^{-5}	1.72^{-2}	2^{+2}	1^{-7}	2.95^{-11}	2.95^{-8}	1.52^{-20}	1.52^{-17}
^{120}Pd	5.78^{-6}	5.78^{-3}	2^{-1}	1^{-10}	9.93^{-12}	9.93^{-9}	2.78^{-18}	2.78^{-15}
^{121}Pd	1.65^{-6}	1.65^{-3}	2^{+2}	1^{-7}	2.83^{-12}	2.83^{-9}	2.39^{-28}	2.39^{-25}
^{122}Pd	4.25^{-7}	4.25^{-4}	2^{+2}	1^{-7}	7.30^{-13}	7.30^{-10}	6.15^{-32}	6.15^{-29}
^{123}Pd	9.46^{-8}	9.46^{-5}	2^{+2}	1^{-7}	1.63^{-13}	1.63^{-10}	1.37^{-29}	1.37^{-26}
^{124}Pd	1.89^{-8}	1.89^{-5}	2^{+2}	1^{-7}	3.25^{-14}	3.25^{-11}	3.63^{-24}	3.63^{-21}
^{152m}Pm	3.09^{-5}	3.09^{-2}	2^{+2}	1^{-7}	5.31^{-11}	5.31^{-8}	2.46^{-17}	2.46^{-14}
^{152}Pm	2.23^{-3}	2.23^{+0}	2^{+2}	1^{-7}	3.83^{-9}	3.83^{-6}	5.29^{-16}	5.29^{-13}
^{153}Pm	1.36^{-3}	1.36^{+0}	2^{+2}	1^{-7}	2.34^{-9}	2.34^{-6}	4.24^{-16}	4.24^{-13}
^{154m}Pm	7.68^{-5}	7.68^{-2}	2^{+2}	1^{-7}	1.32^{-10}	1.32^{-7}	1.20^{-17}	1.20^{-14}
^{154}Pm	5.47^{-4}	5.47^{-1}	2^{+2}	1^{-7}	9.40^{-10}	9.40^{-7}	5.37^{-17}	5.37^{-14}
^{155}Pm	2.57^{-4}	2.57^{-1}	2^{+2}	1^{-7}	4.41^{-10}	4.41^{-7}	1.19^{-17}	1.19^{-14}
^{156}Pm	8.84^{-5}	8.84^{-2}	2^{+2}	1^{-7}	1.52^{-10}	1.52^{-7}	2.27^{-18}	2.27^{-15}
^{157}Pm	2.87^{-5}	2.87^{-2}	2^{+2}	1^{-7}	4.93^{-11}	4.93^{-8}	2.91^{-19}	2.91^{-16}
^{158}Pm	7.78^{-6}	7.78^{-3}	2^{+2}	1^{-7}	1.34^{-11}	1.34^{-8}	3.60^{-20}	3.60^{-17}
^{159}Pm	1.08^{-6}	1.08^{-3}	2^{+2}	1^{-7}	1.86^{-12}	1.86^{-9}	1.53^{-21}	1.53^{-18}
^{160}Pm	3.18^{-7}	3.18^{-4}	2^{+2}	1^{-7}	5.46^{-13}	5.46^{-10}	6.11^{-22}	6.11^{-19}
^{161}Pm	4.69^{-9}	4.69^{-6}	2^{+2}	1^{-7}	8.06^{-15}	8.06^{-12}	3.16^{-24}	3.16^{-21}
^{162}Pm	1.87^{-10}	1.87^{-7}	2^{+2}	1^{-7}	3.21^{-16}	3.21^{-13}	9.01^{-26}	9.01^{-23}
^{144m}Pr	5.48^{-4}	2.51^{-3}	2^{+2}	1^{-7}	9.41^{-10}	4.31^{-9}	2.28^{-16}	1.04^{-15}
^{146}Pr	2.50^{-2}	2.50^{+1}	2^{+2}	1^{-7}	4.29^{-8}	4.29^{-5}	3.47^{-14}	3.47^{-11}
^{148}Pr	1.41^{-2}	1.41^{+1}	2^{+2}	1^{-7}	2.42^{-8}	2.42^{-5}	1.85^{-15}	1.85^{-12}
^{149}Pr	8.96^{-3}	8.96^{+0}	2^{+2}	1^{-7}	1.54^{-8}	1.54^{-5}	1.19^{-15}	1.19^{-12}
^{150}Pr	4.95^{-3}	4.95^{+0}	2^{+2}	1^{-7}	8.50^{-9}	8.50^{-6}	2.96^{-17}	2.96^{-14}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{151}Pr	2.48^{-3}	2.48^{+0}	2^{+2}	1^{-7}	4.26^{-9}	4.26^{-6}	4.52^{-17}	4.52^{-14}
^{152}Pr	9.97^{-4}	9.97^{-1}	2^{+2}	1^{-7}	1.71^{-9}	1.71^{-6}	3.07^{-18}	3.07^{-15}
^{153}Pr	2.62^{-4}	2.62^{-1}	2^{+2}	1^{-7}	4.50^{-10}	4.50^{-7}	1.08^{-18}	1.08^{-15}
^{154}Pr	4.58^{-5}	4.58^{-2}	2^{+2}	1^{-7}	7.87^{-11}	7.87^{-8}	1.01^{-19}	1.01^{-16}
^{155}Pr	5.48^{-6}	5.48^{-3}	2^{+2}	1^{-7}	9.41^{-12}	9.41^{-9}	1.58^{-27}	1.58^{-24}
^{156}Pr	4.05^{-7}	4.05^{-4}	2^{+2}	1^{-7}	6.96^{-13}	6.96^{-10}	1.17^{-28}	1.17^{-25}
^{157}Pr	2.23^{-8}	2.23^{-5}	2^{+2}	1^{-7}	3.83^{-14}	3.83^{-11}	6.44^{-24}	6.44^{-21}
^{158}Pr	8.87^{-10}	8.87^{-7}	2^{+2}	1^{-7}	1.52^{-15}	1.52^{-12}	1.70^{-25}	1.70^{-22}
^{159}Pr	1.42^{-11}	1.42^{-8}	2^{+2}	1^{-7}	2.44^{-17}	2.44^{-14}	1.37^{-27}	1.37^{-24}
^{86m}Rb	4.75^{-7}	4.75^{-4}	2^{+2}	1^{-7}	8.16^{-13}	8.16^{-10}	2.79^{-20}	2.79^{-17}
^{90m}Rb	8.33^{-3}	8.33^{+0}	2^{+2}	1^{-7}	1.43^{-8}	1.43^{-5}	2.06^{-15}	2.06^{-12}
^{90}Rb	4.02^{-2}	4.02^{+1}	2^{+2}	1^{-7}	6.91^{-8}	6.91^{-5}	6.05^{-15}	6.05^{-12}
^{91}Rb	4.80^{-2}	4.80^{+1}	2^{+2}	1^{-7}	8.25^{-8}	8.25^{-5}	2.70^{-15}	2.70^{-12}
^{92}Rb	4.05^{-2}	4.05^{+1}	2^{+2}	1^{-7}	6.96^{-8}	6.96^{-5}	1.76^{-16}	1.76^{-13}
^{93}Rb	3.00^{-2}	3.00^{+1}	2^{+2}	1^{-7}	5.15^{-8}	5.15^{-5}	1.69^{-16}	1.69^{-13}
^{94}Rb	1.44^{-2}	1.44^{+1}	2^{+2}	1^{-7}	2.47^{-8}	2.47^{-5}	3.79^{-17}	3.79^{-14}
^{95}Rb	7.59^{-3}	7.59^{+0}	2^{+2}	1^{-7}	1.30^{-8}	1.30^{-5}	2.78^{-18}	2.78^{-15}
^{96}Rb	1.61^{-3}	1.61^{+0}	2^{+2}	1^{-7}	2.77^{-9}	2.77^{-6}	3.16^{-19}	3.16^{-16}
^{97}Rb	2.88^{-4}	2.88^{-1}	2^{+2}	1^{-7}	4.95^{-10}	4.95^{-7}	4.89^{-20}	4.89^{-17}
^{98}Rb	4.29^{-5}	4.29^{-2}	2^{+2}	1^{-7}	7.37^{-11}	7.37^{-8}	4.53^{-21}	4.53^{-18}
^{99}Rb	3.42^{-6}	3.42^{-3}	2^{+2}	1^{-7}	5.87^{-12}	5.87^{-9}	1.66^{-22}	1.66^{-19}
^{100}Rb	1.36^{-7}	1.36^{-4}	2^{+2}	1^{-7}	2.34^{-13}	2.34^{-10}	6.68^{-24}	6.68^{-21}
^{104m}Rh	3.72^{-10}	2.80^{-7}	2^{+2}	1^{-7}	6.39^{-16}	4.81^{-13}	9.33^{-23}	7.02^{-20}
^{104}Rh	1.81^{-9}	5.50^{-7}	2^{+2}	1^{-7}	3.11^{-15}	9.45^{-13}	7.38^{-23}	2.24^{-20}
^{105m}Rh	2.39^{-3}	1.74^{+0}	2^{+2}	1^{-7}	4.11^{-9}	2.99^{-6}	9.88^{-17}	7.19^{-14}
^{106}Rh	3.27^{-3}	1.02^{-2}	2^{+2}	1^{-7}	5.62^{-9}	1.75^{-8}	9.39^{-17}	2.93^{-16}
^{108m}Rh	3.46^{-7}	3.46^{-4}	2^{+2}	1^{-7}	5.94^{-13}	5.94^{-10}	1.20^{-19}	1.20^{-16}
^{108}Rh	5.94^{-4}	5.94^{-1}	2^{+2}	1^{-7}	1.02^{-9}	1.02^{-6}	9.59^{-18}	9.59^{-15}
^{109m}Rh	1.25^{-4}	1.25^{-1}	2^{+2}	1^{-7}	2.15^{-10}	2.15^{-7}	9.63^{-18}	9.63^{-15}
^{109}Rh	2.50^{-4}	2.50^{-1}	2^{+2}	1^{-7}	4.29^{-10}	4.29^{-7}	1.95^{-17}	1.95^{-14}
^{110m}Rh	5.84^{-6}	5.84^{-3}	2^{+2}	1^{-7}	1.00^{-11}	1.00^{-8}	1.60^{-19}	1.60^{-16}
^{110}Rh	1.81^{-4}	1.81^{-1}	2^{+2}	1^{-7}	3.11^{-10}	3.11^{-7}	5.57^{-19}	5.57^{-16}
^{111}Rh	1.63^{-4}	1.63^{-1}	2^{+2}	1^{-7}	2.80^{-10}	2.80^{-7}	1.72^{-18}	1.72^{-15}
^{112}Rh	1.01^{-4}	1.01^{-1}	2^{+2}	1^{-7}	1.73^{-10}	1.73^{-7}	2.04^{-19}	2.04^{-16}
^{113}Rh	8.51^{-5}	8.51^{-2}	2^{+2}	1^{-7}	1.46^{-10}	1.46^{-7}	2.29^{-19}	2.29^{-16}
^{114}Rh	5.71^{-5}	5.71^{-2}	2^{+2}	1^{-7}	9.81^{-11}	9.81^{-8}	9.36^{-20}	9.36^{-17}
^{115}Rh	2.91^{-5}	2.91^{-2}	2^{+2}	1^{-7}	5.00^{-11}	5.00^{-8}	2.78^{-20}	2.78^{-17}
^{116}Rh	1.32^{-5}	1.32^{-2}	2^{+2}	1^{-7}	2.27^{-11}	2.27^{-8}	8.64^{-21}	8.64^{-18}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{117}Rh	3.18^{-5}	3.18^{-2}	2^{+2}	1^{-7}	5.46^{-11}	5.46^{-8}	1.35^{-20}	1.35^{-17}
^{118}Rh	8.90^{-7}	8.90^{-4}	2^{+2}	1^{-7}	1.53^{-12}	1.53^{-9}	2.57^{-22}	2.57^{-19}
^{119}Rh	1.89^{-7}	1.89^{-4}	2^{+2}	1^{-7}	3.25^{-13}	3.25^{-10}	2.74^{-29}	2.74^{-26}
^{120}Rh	2.40^{-8}	2.40^{-5}	2^{+2}	1^{-7}	4.12^{-14}	4.12^{-11}	3.47^{-30}	3.47^{-27}
^{121}Rh	2.76^{-9}	2.76^{-6}	2^{+2}	1^{-7}	4.74^{-15}	4.74^{-12}	4.00^{-31}	4.00^{-28}
^{122}Rh	3.08^{-10}	3.08^{-7}	2^{+2}	1^{-7}	5.29^{-16}	5.29^{-13}	1.48^{-26}	1.48^{-23}
^{107}Ru	1.37^{-3}	1.37^{+0}	2^{+2}	1^{-7}	2.35^{-9}	2.35^{-6}	3.01^{-16}	3.01^{-13}
^{108}Ru	5.94^{-4}	5.94^{-1}	2^{+2}	1^{-7}	1.02^{-9}	1.02^{-6}	1.58^{-16}	1.58^{-13}
^{109}Ru	2.46^{-4}	2.46^{-1}	2^{+2}	1^{-7}	4.23^{-10}	4.23^{-7}	8.17^{-18}	8.17^{-15}
^{110}Ru	1.75^{-4}	1.75^{-1}	2^{+2}	1^{-7}	3.01^{-10}	3.01^{-7}	2.53^{-18}	2.53^{-15}
^{111}Ru	1.27^{-4}	1.27^{-1}	2^{+2}	1^{-7}	2.18^{-10}	2.18^{-7}	2.59^{-19}	2.59^{-16}
^{112}Ru	5.46^{-5}	5.46^{-2}	2^{+2}	1^{-7}	9.38^{-11}	9.38^{-8}	9.22^{-20}	9.22^{-17}
^{113}Ru	2.96^{-5}	2.96^{-2}	2^{+2}	1^{-7}	5.08^{-11}	5.08^{-8}	2.28^{-20}	2.28^{-17}
^{114}Ru	1.05^{-5}	1.05^{-2}	2^{+2}	1^{-7}	1.80^{-11}	1.80^{-8}	5.36^{-21}	5.36^{-18}
^{115}Ru	2.27^{-6}	2.27^{-3}	2^{+2}	1^{-7}	3.90^{-12}	3.90^{-9}	1.62^{-21}	1.62^{-18}
^{116}Ru	3.81^{-7}	3.81^{-4}	2^{+2}	1^{-7}	6.54^{-13}	6.54^{-10}	1.47^{-22}	1.47^{-19}
^{117}Ru	2.53^{-7}	2.53^{-4}	2^{+2}	1^{-7}	4.35^{-13}	4.35^{-10}	7.30^{-23}	7.30^{-20}
^{118}Ru	3.16^{-9}	3.16^{-6}	2^{+2}	1^{-7}	5.43^{-15}	5.43^{-12}	4.57^{-31}	4.57^{-28}
^{120}Ru	1.04^{-11}	1.04^{-8}	2^{+2}	1^{-7}	1.79^{-17}	1.79^{-14}	1.51^{-33}	1.51^{-30}
^{122m}Sb	1.91^{-10}	1.91^{-7}	2^{+2}	1^{-7}	3.28^{-16}	3.28^{-13}	4.62^{-23}	4.62^{-20}
^{130m}Sb	9.99^{-3}	9.99^{+0}	2^{+2}	1^{-7}	1.72^{-8}	1.72^{-5}	3.64^{-15}	3.64^{-12}
^{132m}Sb	8.79^{-3}	8.79^{+0}	2^{+2}	1^{-7}	1.51^{-8}	1.51^{-5}	2.08^{-15}	2.08^{-12}
^{132}Sb	1.38^{-2}	1.38^{+1}	2^{+2}	1^{-7}	2.37^{-8}	2.37^{-5}	2.23^{-15}	2.23^{-12}
^{133}Sb	1.88^{-2}	1.88^{+1}	2^{+2}	1^{-7}	3.23^{-8}	3.23^{-5}	2.72^{-15}	2.72^{-12}
^{134m}Sb	1.92^{-3}	1.92^{+0}	2^{+2}	1^{-7}	3.30^{-9}	3.30^{-6}	1.88^{-17}	1.88^{-14}
^{134}Sb	2.00^{-3}	2.00^{+0}	2^{+2}	1^{-7}	3.44^{-9}	3.44^{-6}	1.50^{-18}	1.50^{-15}
^{135}Sb	1.66^{-3}	1.66^{+0}	2^{+2}	1^{-7}	2.85^{-9}	2.85^{-6}	2.73^{-18}	2.73^{-15}
^{136}Sb	2.52^{-4}	2.52^{-1}	2^{+2}	1^{-7}	4.33^{-10}	4.33^{-7}	1.99^{-19}	1.99^{-16}
^{137}Sb	1.70^{-5}	1.70^{-2}	2^{+2}	1^{-7}	2.92^{-11}	2.92^{-8}	2.46^{-27}	2.46^{-24}
^{138}Sb	1.06^{-6}	1.06^{-3}	2^{+2}	1^{-7}	1.82^{-12}	1.82^{-9}	1.53^{-28}	1.53^{-25}
^{139}Sb	4.93^{-8}	4.93^{-5}	2^{+2}	1^{-7}	8.47^{-14}	8.47^{-11}	7.14^{-30}	7.14^{-27}
^{77m}Se	1.72^{-7}	5.40^{-5}	2^{+2}	1^{-7}	2.95^{-13}	9.28^{-11}	2.88^{-21}	9.03^{-19}
^{79m}Se	4.26^{-4}	4.26^{-1}	2^{+2}	1^{-7}	7.32^{-10}	7.32^{-7}	9.64^{-17}	9.64^{-14}
^{83m}Se	2.78^{-3}	2.78^{+0}	2^{+2}	1^{-7}	4.78^{-9}	4.78^{-6}	1.88^{-16}	1.88^{-13}
^{84}Se	8.28^{-3}	8.28^{+0}	2^{+2}	1^{-7}	1.42^{-8}	1.42^{-5}	1.58^{-15}	1.58^{-12}
^{85m}Se	3.85^{-3}	3.85^{+0}	2^{+2}	1^{-7}	6.61^{-9}	6.61^{-6}	1.19^{-16}	1.19^{-13}
^{85}Se	5.25^{-3}	5.25^{+0}	2^{+2}	1^{-7}	9.02^{-9}	9.02^{-6}	1.60^{-16}	1.60^{-13}
^{86}Se	1.11^{-2}	1.11^{+1}	2^{+2}	1^{-7}	1.91^{-8}	1.91^{-5}	1.61^{-16}	1.61^{-13}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{87}Se	8.26^{-3}	8.26^{+0}	2^{+2}	1^{-7}	1.42^{-8}	1.42^{-5}	4.45^{-17}	4.45^{-14}
^{88}Se	2.85^{-3}	2.85^{+0}	2^{+2}	1^{-7}	4.90^{-9}	4.90^{-6}	4.13^{-18}	4.13^{-15}
^{89}Se	7.41^{-4}	7.41^{-1}	2^{+2}	1^{-7}	1.27^{-9}	1.27^{-6}	2.93^{-19}	2.93^{-16}
^{90}Se	2.83^{-4}	2.83^{-1}	2^{+2}	1^{-7}	4.86^{-10}	4.86^{-7}	4.10^{-26}	4.10^{-23}
^{91}Se	2.89^{-5}	2.89^{-2}	2^{+2}	1^{-7}	4.96^{-11}	4.96^{-8}	7.53^{-21}	7.53^{-18}
^{92}Se	3.75^{-7}	3.75^{-4}	2^{+2}	1^{-7}	6.44^{-13}	6.44^{-10}	1.08^{-28}	1.08^{-25}
^{148}Sm	1.90^{-22}	4.47^{-24}	2^{-1}	1^{-10}	3.26^{-28}	7.68^{-30}	5.12^{-27}	1.21^{-28}
^{157}Sm	5.32^{-5}	5.32^{-2}	2^{+2}	1^{-7}	9.14^{-11}	9.14^{-8}	2.49^{-17}	2.49^{-14}
^{158}Sm	2.53^{-5}	2.53^{-2}	2^{+2}	1^{-7}	4.35^{-11}	4.35^{-8}	8.04^{-18}	8.04^{-15}
^{159}Sm	7.09^{-6}	7.09^{-3}	2^{+2}	1^{-7}	1.22^{-11}	1.22^{-8}	7.96^{-20}	7.96^{-17}
^{160}Sm	2.49^{-6}	2.49^{-3}	2^{+2}	1^{-7}	4.28^{-12}	4.28^{-9}	2.30^{-20}	2.30^{-17}
^{161}Sm	2.32^{-7}	2.32^{-4}	2^{+2}	1^{-7}	3.99^{-13}	3.99^{-10}	1.07^{-21}	1.07^{-18}
^{162}Sm	2.68^{-8}	2.68^{-5}	2^{+2}	1^{-7}	4.60^{-14}	4.60^{-11}	5.15^{-23}	5.15^{-20}
^{163}Sm	2.05^{-9}	2.05^{-6}	2^{+2}	1^{-7}	3.52^{-15}	3.52^{-12}	1.98^{-24}	1.98^{-21}
^{164}Sm	8.45^{-11}	8.45^{-8}	2^{+2}	1^{-7}	1.45^{-16}	1.45^{-13}	4.07^{-26}	4.07^{-23}
^{165}Sm	4.19^{-12}	4.19^{-9}	2^{+2}	1^{-7}	7.20^{-18}	7.20^{-15}	8.05^{-28}	8.05^{-25}
^{125m}Sn	1.36^{-4}	1.36^{-1}	2^{+2}	1^{-7}	2.34^{-10}	2.34^{-7}	7.26^{-17}	7.26^{-14}
^{127m}Sn	3.50^{-4}	3.50^{-1}	2^{+2}	1^{-7}	6.01^{-10}	6.01^{-7}	8.39^{-17}	8.39^{-14}
^{129m}Sn	2.94^{-3}	2.94^{+0}	2^{+2}	1^{-7}	5.05^{-9}	5.05^{-6}	1.17^{-15}	1.17^{-12}
^{129}Sn	1.72^{-3}	1.72^{+0}	2^{+2}	1^{-7}	2.95^{-9}	2.95^{-6}	2.49^{-16}	2.49^{-13}
^{130}Sn	7.45^{-3}	7.45^{+0}	2^{+2}	1^{-7}	1.28^{-8}	1.28^{-5}	1.59^{-15}	1.59^{-12}
^{131}Sn	7.83^{-3}	7.83^{+0}	2^{+2}	1^{-7}	1.35^{-8}	1.35^{-5}	4.60^{-16}	4.60^{-13}
^{132}Sn	4.88^{-3}	4.88^{+0}	2^{+2}	1^{-7}	8.38^{-9}	8.38^{-6}	1.88^{-16}	1.88^{-13}
^{133}Sn	1.41^{-3}	1.41^{+0}	2^{+2}	1^{-7}	2.42^{-9}	2.42^{-6}	1.99^{-18}	1.99^{-15}
^{134}Sn	8.54^{-5}	8.54^{-2}	2^{+2}	1^{-7}	1.47^{-10}	1.47^{-7}	8.53^{-20}	8.53^{-17}
^{135}Sn	1.04^{-5}	1.04^{-2}	2^{+2}	1^{-7}	1.79^{-11}	1.79^{-8}	5.30^{-21}	5.30^{-18}
^{136}Sn	4.51^{-7}	4.51^{-4}	2^{+2}	1^{-7}	7.75^{-13}	7.75^{-10}	1.08^{-22}	1.08^{-19}
^{93}Sr	5.35^{-2}	5.35^{+1}	2^{+2}	1^{-7}	9.19^{-8}	9.19^{-5}	2.32^{-14}	2.32^{-11}
^{94}Sr	4.95^{-2}	4.95^{+1}	2^{+2}	1^{-7}	8.50^{-8}	8.50^{-5}	3.57^{-15}	3.57^{-12}
^{95}Sr	4.56^{-2}	4.56^{+1}	2^{+2}	1^{-7}	7.83^{-8}	7.83^{-5}	1.10^{-15}	1.10^{-12}
^{96}Sr	3.13^{-2}	3.13^{+1}	2^{+2}	1^{-7}	5.38^{-8}	5.38^{-5}	3.20^{-17}	3.20^{-14}
^{97}Sr	1.60^{-2}	1.60^{+1}	2^{+2}	1^{-7}	2.75^{-8}	2.75^{-5}	6.78^{-18}	6.78^{-15}
^{98}Sr	5.79^{-3}	5.79^{+0}	2^{+2}	1^{-7}	9.95^{-9}	9.95^{-6}	3.90^{-18}	3.90^{-15}
^{99}Sr	1.28^{-3}	1.28^{+0}	2^{+2}	1^{-7}	2.20^{-9}	2.20^{-6}	3.58^{-19}	3.58^{-16}
^{100}Sr	1.93^{-4}	1.93^{-1}	2^{+2}	1^{-7}	3.32^{-10}	3.32^{-7}	3.71^{-20}	3.71^{-17}
^{101}Sr	2.42^{-5}	2.42^{-2}	2^{+2}	1^{-7}	4.16^{-11}	4.16^{-8}	2.76^{-21}	2.76^{-18}
^{102}Sr	1.65^{-6}	1.65^{-3}	2^{+2}	1^{-7}	2.83^{-12}	2.83^{-9}	1.10^{-22}	1.10^{-19}
^{103}Sr	4.13^{-8}	4.13^{-5}	2^{+2}	1^{-7}	7.09^{-14}	7.09^{-11}	5.98^{-30}	5.98^{-27}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{104}Sr	7.00^{-10}	7.00^{-7}	2^{+2}	1^{-7}	1.20^{-15}	1.20^{-12}	2.02^{-31}	2.02^{-28}
^{162m}Tb	5.98^{-9}	5.45^{-6}	2^{+2}	1^{-7}	1.03^{-14}	9.36^{-12}	2.63^{-21}	2.39^{-18}
^{162}Tb	2.33^{-7}	2.33^{-4}	2^{+2}	1^{-7}	4.00^{-13}	4.00^{-10}	1.02^{-19}	1.02^{-16}
^{163}Tb	8.29^{-8}	8.29^{-5}	2^{+2}	1^{-7}	1.42^{-13}	1.42^{-10}	9.31^{-20}	9.31^{-17}
^{164}Tb	2.31^{-8}	2.31^{-5}	2^{+2}	1^{-7}	3.97^{-14}	3.97^{-11}	4.00^{-21}	4.00^{-18}
^{165}Tb	1.11^{-8}	1.11^{-5}	2^{+2}	1^{-7}	1.91^{-14}	1.91^{-11}	1.36^{-21}	1.36^{-18}
^{100}Tc	2.00^{-7}	1.99^{-4}	2^{+2}	1^{-7}	3.44^{-13}	3.42^{-10}	3.05^{-21}	3.03^{-18}
^{102m}Tc	1.05^{-5}	1.05^{-2}	2^{+2}	1^{-7}	1.80^{-11}	1.80^{-8}	2.64^{-18}	2.64^{-15}
^{102}Tc	3.52^{-2}	3.52^{+1}	2^{+2}	1^{-7}	6.05^{-8}	6.05^{-5}	1.80^{-16}	1.80^{-13}
^{103}Tc	2.63^{-2}	2.63^{+1}	2^{+2}	1^{-7}	4.52^{-8}	4.52^{-5}	1.37^{-15}	1.37^{-12}
^{105}Tc	8.51^{-3}	8.51^{+0}	2^{+2}	1^{-7}	1.46^{-8}	1.46^{-5}	3.74^{-15}	3.74^{-12}
^{106}Tc	3.26^{-3}	3.26^{+0}	2^{+2}	1^{-7}	5.60^{-9}	5.60^{-6}	1.13^{-16}	1.13^{-13}
^{107}Tc	1.34^{-3}	1.34^{+0}	2^{+2}	1^{-7}	2.30^{-9}	2.30^{-6}	2.73^{-17}	2.73^{-14}
^{108}Tc	5.12^{-4}	5.12^{-1}	2^{+2}	1^{-7}	8.80^{-10}	8.80^{-7}	2.55^{-18}	2.55^{-15}
^{109}Tc	1.32^{-4}	1.32^{-1}	2^{+2}	1^{-7}	2.27^{-10}	2.27^{-7}	1.11^{-19}	1.11^{-16}
^{110}Tc	5.29^{-5}	5.29^{-2}	2^{+2}	1^{-7}	9.09^{-11}	9.09^{-8}	4.67^{-20}	4.67^{-17}
^{111}Tc	1.49^{-5}	1.49^{-2}	2^{+2}	1^{-7}	2.56^{-11}	2.56^{-8}	4.17^{-21}	4.17^{-18}
^{112}Tc	2.51^{-6}	2.51^{-3}	2^{+2}	1^{-7}	4.31^{-12}	4.31^{-9}	7.02^{-22}	7.02^{-19}
^{113}Tc	6.52^{-7}	6.52^{-4}	2^{+2}	1^{-7}	1.12^{-12}	1.12^{-9}	1.07^{-22}	1.07^{-19}
^{114}Tc	7.35^{-8}	7.35^{-5}	2^{+2}	1^{-7}	1.26^{-13}	1.26^{-10}	1.20^{-23}	1.20^{-20}
^{115}Tc	6.09^{-9}	6.09^{-6}	2^{+2}	1^{-7}	1.05^{-14}	1.05^{-11}	5.88^{-25}	5.88^{-22}
^{116}Tc	2.92^{-10}	2.92^{-7}	2^{+2}	1^{-7}	5.02^{-16}	5.02^{-13}	2.53^{-26}	2.53^{-23}
^{117}Tc	5.89^{-11}	5.89^{-8}	2^{+2}	1^{-7}	1.01^{-16}	1.01^{-13}	2.27^{-27}	2.27^{-24}
^{135}Te	2.73^{-2}	2.73^{+1}	2^{+2}	1^{-7}	4.69^{-8}	4.69^{-5}	5.04^{-16}	5.04^{-13}
^{136}Te	1.59^{-2}	1.59^{+1}	2^{+2}	1^{-7}	2.73^{-8}	2.73^{-5}	2.69^{-16}	2.69^{-13}
^{137}Te	3.44^{-3}	3.44^{+0}	2^{+2}	1^{-7}	5.91^{-9}	5.91^{-6}	8.24^{-18}	8.24^{-15}
^{138}Te	7.15^{-4}	7.15^{-1}	2^{+2}	1^{-7}	1.23^{-9}	1.23^{-6}	1.10^{-18}	1.10^{-15}
^{139}Te	1.11^{-4}	1.11^{-1}	2^{+2}	1^{-7}	1.91^{-10}	1.91^{-7}	1.61^{-26}	1.61^{-23}
^{140}Te	1.05^{-5}	1.05^{-2}	2^{+2}	1^{-7}	1.80^{-11}	1.80^{-8}	1.52^{-27}	1.52^{-24}
^{141}Te	3.77^{-7}	3.77^{-4}	2^{+2}	1^{-7}	6.48^{-13}	6.48^{-10}	5.46^{-29}	5.46^{-26}
^{142}Te	6.05^{-9}	6.05^{-6}	2^{+2}	1^{-7}	1.04^{-14}	1.04^{-11}	8.76^{-31}	8.76^{-28}
^{89m}Y	2.44^{-9}	2.44^{-6}	2^{+2}	1^{-7}	4.19^{-15}	4.19^{-12}	3.59^{-23}	3.59^{-20}
^{96}Y	5.04^{-2}	5.04^{+1}	2^{+2}	1^{-7}	8.66^{-8}	8.66^{-5}	3.01^{-16}	3.01^{-13}
^{97}Y	4.23^{-2}	4.23^{+1}	2^{+2}	1^{-7}	7.27^{-8}	7.27^{-5}	1.51^{-16}	1.51^{-13}
^{98}Y	3.00^{-2}	3.00^{+1}	2^{+2}	1^{-7}	5.15^{-8}	5.15^{-5}	1.73^{-17}	1.73^{-14}
^{99}Y	1.69^{-2}	1.69^{+1}	2^{+2}	1^{-7}	2.90^{-8}	2.90^{-5}	2.45^{-17}	2.45^{-14}
^{100}Y	7.28^{-3}	7.28^{+0}	2^{+2}	1^{-7}	1.25^{-8}	1.25^{-5}	5.15^{-18}	5.15^{-15}
^{101}Y	2.24^{-3}	2.24^{+0}	2^{+2}	1^{-7}	3.85^{-9}	3.85^{-6}	9.71^{-19}	9.71^{-16}

Continued on next page...

Table A.4: Particulate fission products not listed in 10CFR20 (*cont.*)

Nuclide	ORIGEN Activity		10CFR20 Limits		MPFD^3-T Activity		MPFD^3-T CEDE	
	Long	Short	ALI	DAC	Long	Short	Long	Short
^{102}Y	4.49^{-4}	4.49^{-1}	2^{+2}	1^{-7}	7.71^{-10}	7.71^{-7}	1.30^{-19}	1.30^{-16}
^{103}Y	4.49^{-5}	4.49^{-2}	2^{+2}	1^{-7}	7.71^{-11}	7.71^{-8}	9.94^{-21}	9.94^{-18}
^{104}Y	2.13^{-6}	2.13^{-3}	2^{+2}	1^{-7}	3.66^{-12}	3.66^{-9}	3.69^{-22}	3.69^{-19}
^{105}Y	6.29^{-8}	6.29^{-5}	2^{+2}	1^{-7}	1.08^{-13}	1.08^{-10}	1.82^{-29}	1.82^{-26}
^{107}Y	2.85^{-10}	2.85^{-7}	2^{+2}	1^{-7}	4.90^{-16}	4.90^{-13}	4.13^{-32}	4.13^{-29}
^{73}Zn	8.59^{-7}	8.59^{-4}	2^{+2}	1^{-7}	1.48^{-12}	1.48^{-9}	1.99^{-20}	1.99^{-17}
^{74}Zn	2.82^{-6}	2.82^{-3}	2^{+2}	1^{-7}	4.84^{-12}	4.84^{-9}	2.60^{-19}	2.60^{-16}
^{75}Zn	8.31^{-6}	8.31^{-3}	2^{+2}	1^{-7}	1.43^{-11}	1.43^{-8}	8.16^{-20}	8.16^{-17}
^{76}Zn	2.10^{-5}	2.10^{-2}	2^{+2}	1^{-7}	3.61^{-11}	3.61^{-8}	1.15^{-19}	1.15^{-16}
^{77}Zn	2.31^{-5}	2.31^{-2}	2^{+2}	1^{-7}	3.97^{-11}	3.97^{-8}	3.11^{-20}	3.11^{-17}
^{78}Zn	3.10^{-5}	3.10^{-2}	2^{+2}	1^{-7}	5.33^{-11}	5.33^{-8}	4.49^{-20}	4.49^{-17}
^{79}Zn	1.24^{-5}	1.24^{-2}	2^{+2}	1^{-7}	2.13^{-11}	2.13^{-8}	1.19^{-20}	1.19^{-17}
^{80}Zn	6.02^{-6}	6.02^{-3}	2^{+2}	1^{-7}	1.03^{-11}	1.03^{-8}	3.13^{-21}	3.13^{-18}
^{81}Zn	1.06^{-6}	1.06^{-3}	2^{+2}	1^{-7}	1.82^{-12}	1.82^{-9}	2.96^{-22}	2.96^{-19}
^{82}Zn	1.05^{-7}	1.05^{-4}	2^{+2}	1^{-7}	1.80^{-13}	1.80^{-10}	1.52^{-29}	1.52^{-26}
^{83}Zn	8.14^{-9}	8.14^{-6}	2^{+2}	1^{-7}	1.40^{-14}	1.40^{-11}	1.18^{-30}	1.18^{-27}
^{90m}Zr	8.70^{-12}	8.70^{-9}	2^{+2}	1^{-7}	1.49^{-17}	1.49^{-14}	6.78^{-27}	6.78^{-24}
^{98}Zr	4.82^{-2}	4.82^{+1}	2^{+2}	1^{-7}	8.28^{-8}	8.28^{-5}	1.42^{-15}	1.42^{-12}
^{99}Zr	4.82^{-2}	4.82^{+1}	2^{+2}	1^{-7}	8.28^{-8}	8.28^{-5}	9.74^{-17}	9.74^{-14}
^{100}Zr	4.46^{-2}	4.46^{+1}	2^{+2}	1^{-7}	7.66^{-8}	7.66^{-5}	3.05^{-16}	3.05^{-13}
^{101}Zr	2.88^{-2}	2.88^{+1}	2^{+2}	1^{-7}	4.95^{-8}	4.95^{-5}	5.54^{-17}	5.54^{-14}
^{102}Zr	1.48^{-2}	1.48^{+1}	2^{+2}	1^{-7}	2.54^{-8}	2.54^{-5}	4.14^{-17}	4.14^{-14}
^{103}Zr	4.45^{-3}	4.45^{+0}	2^{+2}	1^{-7}	7.64^{-9}	7.64^{-6}	5.55^{-18}	5.55^{-15}
^{104}Zr	6.68^{-4}	6.68^{-1}	2^{+2}	1^{-7}	1.15^{-9}	1.15^{-6}	7.73^{-19}	7.73^{-16}
^{105}Zr	5.24^{-5}	5.24^{-2}	2^{+2}	1^{-7}	9.00^{-11}	9.00^{-8}	3.03^{-20}	3.03^{-17}
^{106}Zr	1.93^{-5}	1.93^{-2}	2^{+2}	1^{-7}	3.32^{-11}	3.32^{-8}	2.79^{-27}	2.79^{-24}
^{107}Zr	8.54^{-7}	8.54^{-4}	2^{+2}	1^{-7}	1.47^{-12}	1.47^{-9}	1.24^{-28}	1.24^{-25}
^{108}Zr	1.21^{-8}	1.21^{-5}	2^{+2}	1^{-7}	2.08^{-14}	2.08^{-11}	9.32^{-25}	9.32^{-22}
^{109}Zr	7.15^{-11}	7.15^{-8}	2^{+2}	1^{-7}	1.23^{-16}	1.23^{-13}	1.04^{-32}	1.04^{-29}

A.2 In-Core By-Product Analysis

Before anything is allowed to be irradiated at the Kansas State University TRIGA Mark-II Nuclear Reactor, a byproduct log is required. This log is an overview of the experiment's

composition and an estimate of its radioactivity upon removal from the reactor. The analysis performed in this log becomes the guide to reactor operators on limitations imposed on the reactor's power, time of operations, irradiation locations, size of experiment, and required safety measures. For installation of an MPFD^3-T the experiment is not limited to a single MPFD^3-T but rather a string consisting of up to five MPFD^3-Ts, 40 tinned-copper wires, alumina insulators, and the aluminum housing. Each MPFD^3-T consists of three alumina substrates, platinum contacts with a titanium adhesive under-coating, natural and 93% enriched uranium neutron reactive coatings, argon fill gas, and some nickel based epoxy.

Since there are hundreds of byproduct elements generated by the neutron reactive coatings alone, the full break-out of all byproducts is not shown on the byproduct log (Figure A.1). Instead, all of the byproducts were calculated and summed by parent element. If only one isotope of the parent generated a gamma emitting daughter, then the actual parent isotope is listed.

The cross sections are not the typical 2200 m s^{-1} (0.0253 eV) cross sections but instead are a thermal averaged cross section. These are calculated by taking the integral of the energy dependent flux ($\varphi(E)$) and cross section ($\sigma(E)$) product [1]. This integral is then divided by the integrated slow flux value (φ_t), instead of the total integrated flux value, so that it may be later multiplied by the "Reference Reactor Flux φ_r" value on the byproduct log form (Figure A.1). The thermal averaged cross section calculation may be expressed as

$$\sigma_t = \frac{1}{\varphi_t} \int_0^\infty \sigma\,(E)\,\varphi\,(E)\,dE \tag{A.7}$$

where E is the energy.

Likewise, the gamma-ray energy is actually the intensity averaged gamma-ray energy or

$$E_\gamma = \sum_i Y_i\,E_{\gamma i} \tag{A.8}$$

where Y_i is the yield, or intensity, of gamma-ray i and $E_{\gamma i}$ is the energy of gamma-ray i.

Since the yield is a decimal frequency where a value of 1 equals 100%, then the sum of the yield with the energy automatically constitute an average gamma-ray energy.

The activity (C) is calculated as

$$C = \frac{N_A m}{A}\sigma_t \varphi_r \left(1 - e^{-t_r \frac{\ln(2)}{t_{1/2}}}\right) e^{-t_c \frac{\ln(2)}{t_{1/2}}} \tag{A.9}$$

where N_A is Avogadro's constant, m is the mass, A is the mass number, or more accurately, the isotopic mass, t_r is the irradiation time, t_c is the cooling time, and $t_{1/2}$ is the isotope's half-life. The isotopic exposure rate ($\dot{X}$) in R/h at 1 foot is calculated using the approximation

$$\dot{X} = \frac{6CEN}{r^2} \tag{A.10}$$

where C is in Ci, E is the gamma-ray energy in MeV, N is the yield, and r is the distance in feet. Since the product EN is the intensity averaged gamma-ray energy (E_γ) and r is equal to 1, Equation (A.10) is reduced to

$$\dot{X} = 6CE_\gamma. \tag{A.11}$$

For the uranium, the fission fragments identified in the analysis of Section §A.1 has been included. Examination of the byproduct log shown in Figure A.1 reveals that the primary exposure comes from the tinned copper wiring, specifically from the 44 year half-life for ^{121m}Sn. Due to this issue, the tinned copper wire may be replaced by a bare copper wire Figure A.2 or bare aluminum wire Figure A.3. However, due to oxidation issues with both of these and the aluminum being difficult to solder, an alumel alloy (95% Ni, 2% Al, 2% Mn, 1% Si - Figure A.4) commonly used as one leg of a K-type thermocouple, is going to be used instead.

BYPRODUCT LOG

KSUTMII-4
Rev 3/2003

Target Type	Micro-Pocket Fission Detector ($MPFD^3$-T) String
Target Composition	U, Al, O, Ar, Ti, Pt, Cu, Sn, Ni
Reference Target Mass, m_r	1602.65 mg
Reference Reactor Power, P_r	1250 kW
Reference Reactor Flux, φ_r	5.00E+13 $cm^{-2}s^{-1}$
Reference Activation Time, t_r	525960 minutes (1 year)
Reference Cooling Time	10080 minutes (7 days)
Irradiation Facility	In-core flux probe holes (based on CT flux)

Parent Isotope	Mass (g)	Cross-Section (barns)	Half life (ms,s,m, h,d,a)	Gamma Energy* (MeV)	Calculated Activity[1,2] (Ci)	Isotopic exposure rate at 1 foot[1] (R/h)
Uranium	7.56E-04	---	---	---	4.71E-11	3.10E-10
Al-27	8.55E+02	2.09E-01	134.484 s	1.779	0.00E+00	0.00E+00
O-18	7.43E+02	1.49E+00	26.88 s	0.940	0.00E+00	0.00E+00
Ar-40	1.62E-03	1.61E-03	6576.6 s	1.284	4.02E-30	3.10E-29
Ti-50	3.81E-05	1.98E-06	345.6 s	0.369	0.00E+00	0.00E+00
Platinum	3.62E-03	---	---	---	1.98E-03	9.26E-04
Copper	4.63E+00	---	---	---	1.73E-02	1.93E-02
Tin	1.53E-01	---	---	---	7.31E-02	4.45E-02
Ni-64	1.83E-01	1.36E+00	9061.88 s	0.558	2.39E-22	8.02E-22

Total calculated exposure rate at 1 foot	0.0646 R/h
Actual exposure rate at 1 foot	
Actual γ exposure rate at contact	
Actual γ and β exposure rate at contact	
$m_r P_r t_r$	17561036.37 g kW m
Exempt quantity[2]	Yes ____ No X

[1] See beginning of logbook for equations and examples.

[2] See beginning of logbook for quantities exempted by Kansas Regulation 28-35-192A6.

* Intensity averaged gamma radiation energy

Figure A.1: Byproduct log for a $MPFD^3$-T string with tinned copper wiring.

BYPRODUCT LOG

KSUTMII-4
Rev 3/2003

Target Type	Micro-Pocket Fission Detector ($MPFD^3$-T) String
Target Composition	U, Al, O, Ar, Ti, Pt, Cu, Ni
Reference Target Mass, m_r	1602.68 mg
Reference Reactor Power, P_r	1250 kW
Reference Reactor Flux, φ_r	5.00E+13 $cm^{-2}s^{-1}$
Reference Activation Time, t_r	525960 minutes (1 year)
Reference Cooling Time	10080 minutes (7 days)
Irradiation Facility	In-core flux probe holes (based on CT flux)

Parent Isotope	Mass (g)	Cross-Section (barns)	Half life (ms,s,m, h,d,a)	Gamma Energy* (MeV)	Calculated Activity[1,2] (Ci)	Isotopic exposure rate at 1 foot[1] (R/h)
Uranium	7.56E-04	---	---	---	4.71E-11	3.10E-10
Al-27	8.55E+02	2.09E-01	134.484 s	1.779	0.00E+00	0.00E+00
O-18	7.43E+02	1.49E+00	26.88 s	0.940	0.00E+00	0.00E+00
Ar-40	1.62E-03	1.61E-03	6576.6 s	1.284	4.02E-30	3.10E-29
Ti-50	3.81E-05	1.98E-06	345.6 s	0.369	0.00E+00	0.00E+00
Platinum	3.62E-03	---	---	---	1.98E-03	9.26E-04
Copper	4.82E+00	---	---	---	1.80E-02	2.00E-02
Ni-64	1.83E-01	1.36E+00	9061.88 s	0.558	2.39E-22	8.02E-22

Total calculated exposure rate at 1 foot: 0.0210 R/h

Actual exposure rate at 1 foot:

Actual γ exposure rate at contact:

Actual γ and β exposure rate at contact:

$m_r P_r t_r$: 17561414.68 g kW m

Exempt quantity[2]: Yes ____ No X

[1] See beginning of logbook for equations and examples.

[2] See beginning of logbook for quantities exempted by Kansas Regulation 28-35-192A6.

* Intensity averaged gamma radiation energy

Figure A.2: Byproduct log for a $MPFD^3$-T string with bare copper wiring.

BYPRODUCT LOG

KSUTMII-4
Rev 3/2003

Target Type	Micro-Pocket Fission Detector ($MPFD^3$-T) String
Target Composition	U, Al, O, Ar, Ti, Pt, Ni
Reference Target Mass, m_r	1598.91 mg
Reference Reactor Power, P_r	1250 kW
Reference Reactor Flux, φ_r	5.00E+13 $cm^{-2}s^{-1}$
Reference Activation Time, t_r	525960 minutes (1 year)
Reference Cooling Time	10080 minutes (7 days)
Irradiation Facility	In-core flux probe holes (based on CT flux)

Parent Isotope	Mass (g)	Cross-Section (barns)	Half life (ms,s,m, h,d,a)	Gamma Energy* (MeV)	Calculated Activity[1,2] (Ci)	Isotopic exposure rate at 1 foot[1] (R/h)
Uranium	7.56E-04	---	---	---	4.71E-11	3.10E-10
Al-27	8.56E+02	2.09E-01	134.484 s	1.779	0.00E+00	0.00E+00
O-18	7.43E+02	1.49E+00	26.88 s	0.940	0.00E+00	0.00E+00
Ar-40	1.62E-03	1.61E-03	6576.6 s	1.284	4.02E-30	3.10E-29
Ti-50	3.81E-05	1.98E-06	345.6 s	0.369	0.00E+00	0.00E+00
Platinum	3.62E-03	---	---	---	1.98E-03	9.26E-04
Ni-64	1.83E-01	1.36E+00	9061.88 s	0.558	2.39E-22	8.02E-22

Total calculated exposure rate at 1 foot	0.0009 R/h
Actual exposure rate at 1 foot	
Actual γ exposure rate at contact	
Actual γ and β exposure rate at contact	
$m_r P_r t_r$	17520049.85 g kW m
Exempt quantity[2]	Yes ____ No X

[1] See beginning of logbook for equations and examples.

[2] See beginning of logbook for quantities exempted by Kansas Regulation 28-35-192A6.

* Intensity averaged gamma radiation energy

Figure A.3: Byproduct log for a $MPFD^3$-T string with bare aluminum wiring.

BYPRODUCT LOG

KSUTMII-4
Rev 3/2003

Target Type	Micro-Pocket Fission Detector ($MPFD^3$-T) String
Target Composition	U, Al, O, Ar, Ti, Pt, Cu, Mn, Si, Ni
Reference Target Mass, m_r	1597.82 mg
Reference Reactor Power, P_r	1250 kW
Reference Reactor Flux, φ_r	5.00E+13 $cm^{-2}s^{-1}$
Reference Activation Time, t_r	525960 minutes (1 year)
Reference Cooling Time	10080 minutes (7 days)
Irradiation Facility	In-core flux probe holes (based on CT flux)

Parent Isotope	Mass (g)	Cross-Section (barns)	Half life (ms,s,m, h,d,a)	Gamma Energy* (MeV)	Calculated Activity[1,2] (Ci)	Isotopic exposure rate at 1 foot[1] (R/h)
Uranium	7.56E-04	---	---	---	4.71E-11	3.10E-10
Al-27	8.55E+02	2.09E-01	134.484 s	1.779	0.00E+00	0.00E+00
O-18	7.43E+02	1.49E+00	26.88 s	0.940	0.00E+00	0.00E+00
Ar-40	1.62E-03	1.61E-03	6576.6 s	1.284	4.02E-30	3.10E-29
Ti-50	3.81E-05	1.98E-06	345.6 s	0.369	0.00E+00	0.00E+00
Platinum	3.62E-03	---	---	---	1.98E-03	9.26E-04
Mn-55	9.38E-02	1.20E+01	9.28E+03	1.691	4.09E-19	4.15E-18
Si-30	1.45E-03	9.64E-02	9.44E+03	0.001	1.95E-22	1.04E-24
Ni-64	4.31E-02	1.36E+00	9.06E+03	0.558	6.06E-21	2.03E-20

Total calculated exposure rate at 1 foot	0.0009 R/h
Actual exposure rate at 1 foot	
Actual γ exposure rate at contact	
Actual γ and β exposure rate at contact	
$m_r P_r t_r$	17508089.6 g kW m
Exempt quantity[2]	Yes ____ No X

[1] See beginning of logbook for equations and examples.

[2] See beginning of logbook for quantities exempted by Kansas Regulation 28-35-192A6.

* Intensity averaged gamma radiation energy

Figure A.4: Byproduct log for a $MPFD^3$-T string with alumel wiring.

Appendix B

Programs

This chapter includes several programs developed for this research project.

B.1 Programming Languages Utilized

The following sections contain a list of software used throughout this research project.

B.1.1 MATLAB

MATLAB is a programming language designed for handling matrix operations and math routines. The version used for the analysis in Chapter 4 is 7.4.0.287 (R2007a), January 29, 2007 while analysis performed in Chapters 6 and 7 used version 7.10.0.499 (R2010a), February 5, 2010 for 32 and 64 bit windows.

B.1.2 Visual Basic

Microsoft Visual Basic is a programming language that allows users to program by laying out windows through standardized buttons and components and then provide the controlling program code. This package is usually associated with Microsoft Visual Studio. Microsoft Visual Studio 2005 Professional Edition with Microsoft Visual Basic 2005 was used for this development. See Section §6.2 for more information and specification of additional packages utilized.

B.1.3 Microchip C18

Microchip's MPLAB IDE version 8.60.00.00 development environment with Microchip's MPLAB C18 version 3.36 compiler was used for the programming the Microchip series 18

microcontrollers located on the mother board and daughter boards.

B.1.4 SRIM 2003/2006

While not a true programming language, SRIM does provide a script enabled execution interface via a .IN file.

B.1.5 MCNP

MCNP (Monte Carlo Neutral Particles) version 4C is a program developed for simulating radiation interactions with matter.

B.2 Monte Carlo Pulse Distribution

This set of programs was utilized to determine the detector spectrum.

B.2.1 MCPulseDist.m

This program is used to generate the data for the detector spectra found in Figures 3.2, 3.4 and 3.5. The input for this program is generated through TRIMsim.m (Section §B.2.2) and IntTRIM.m (Section §B.2.3). MCPulseDist.m may be run in a batch mode via BatchMCPulse.m (Section §B.2.4).

Begin MCPulseDist.m

```
function MCPulseDist(varargin)

u = 0.1; % Uranium Thickness (um)
o = 0.5; % Platinum Overcoat Thickness (um)
d = 1; % Chamber Diameter {mm}
w = 0.5; % Chamber Width {mm}
if nargin > 0
    u = varargin{1}(1);
end
if nargin > 1
    o = varargin{2}(1);
end
if nargin > 2
    d = varargin{3}(1);
end
```

```
if nargin > 3
    w = varargin{4}(1);
end
if nargin > 5
    noplot = varargin{6}(1);
else
    noplot = 0;
end
if noplot == 0
    figure(1);
    clf('reset');
end
if nargin > 4
    s = varargin{5}(1);
else
    s = 1000000; % Number of iterations (int)
end
T = evalc('m = feature(''memstats'')'); % Find amount of RAM
T1 = findstr(T,'Total:');
T2 = findstr(T,'MB');
T2 = T2(1,gt(T2,T1(1,1)));
m = max(10000,10000*round(str2num(T(1,T1(1,1)+6:T2(1,1)-1))/1024));  % itteration chunk
clear T T1 T2;

filepathD = [pwd,'\'];
fname = [filepathD,'Spec_d',num2str(round(d*1000)),'um_w',num2str(round(w*1000)),' ...
    um_u',num2str(round(u*10000)),'A_o',num2str(round(o*10000)),'A.mat'];
lname = [filepathD,'Spec_d',num2str(round(d*1000)),'um_w',num2str(round(w*1000)),' ...
    um_u',num2str(round(u*10000)),'A_o',num2str(round(o*10000)),'A.loc'];
sname = [filepathD(1,1),':\MATSpace\MCPulse\SpecTemp.mat'];
if exist(fileparts(sname),'dir') == 0
    mkdir(fileparts(sname));
end
if exist(sname,'file') == 0
    contrun = 0;
    sd = 0;
    disp('...starting');
else
    d0 = d; w0 = w; u0 = u; o0 = o;
    load(sname);
    if d0 == d && w0 == w && u0 == u && o0 == o
        contrun = 1;
        disp('...resuming');
    else
        contrun = 0;
        sd = 0;
        d = d0; w = w0; u = u0; o = o0;
        disp('...starting');
    end
end
load([filepathD,'U235.mat'],'data');
```

```
data = data/2;
xx = 0;
load([filepathD,'IntIoniz.mat'],'IntIoniz','IntIonizU','IntIonizPt','xx','xxU ...
    ','xxPt');
I = gt(data,0); % Remove intensities <= 0
IntIoniz = IntIoniz(I,:);
IntIonizPt = IntIonizPt(I,:);
IntIonizU = IntIonizU(I,:);
data = data(:,I);
clear I;

IntIoniz = IntIoniz/1e6; % convert to MeV
IntIonizU = IntIonizU/1e6;
IntIonizPt = IntIonizPt/1e6;

nbins = 1024;
binmax = sqrt(d^2+w^2);
engmax = sqrt(d^2+(w+(u+o)/1000)^2); % distance for maximum energy calculation
if engmax > xx(1,end)
    engmax = max(max(IntIoniz));
else
    i = 1;
    while xx(1,i) < engmax
        i = i + 1;
    end
    engmax = max(IntIoniz(:,i));
end
if contrun == 0 % Initialize variables for new run
    hist3D = 0;
    histFis = 0;
    histJKS = 0;
    tet = 0;
end

Imax = min([IntIoniz(:,end) IntIonizU(:,end) IntIonizPt(:,end)],[],2);
Inum = size(IntIoniz,1);
IIlast = Inum*ones(1,Inum);
for i = 1:Inum
    IntIoniz(i,:) = min(IntIoniz(i,:),Imax(i,1)*ones(1,size(IntIoniz,2)));
    for j = size(IntIoniz,2):-1:2
        if IntIoniz(i,j) ~= IntIoniz(i,j-1)
            IIlast(1,i) = j;
            break;
        end
    end
end
Inum = size(IntIonizPt,1);
IIlastPt = Inum*ones(1,Inum);
for i = 1:Inum
    IntIonizPt(i,:) = min(IntIonizPt(i,:),Imax(i,1)*ones(1,size(IntIonizPt,2)));
    for j = size(IntIonizPt,2):-1:2
```

```
        if IntIonizPt(i,j) ~= IntIonizPt(i,j-1)
            IIlastPt(1,i) = j;
            break;
        end
    end
end
for i = 1:size(IntIonizU,1)
    IntIonizU(i,:) = min(IntIonizU(i,:),Imax(i,1)*ones(1,size(IntIonizU,2)));
end

if exist('tet','var') ~= 1
    tet = 0;
end
tic;

while sd < s
    m = min(m,s-sd); % adjust itteration chunk if needed
    r = rand(1,m).^0.5*d/2; % radius of source (mm)
    zh = rand(1,m)*u/1000; %height location of source generation in uranium coating (mm)
    t1 = 2*pi()*rand(1,m); %azimuthal source angle (0-2pi radians)
    t2 = 2*pi()*rand(1,m); %azimuthal emission angle (0-2pi radians)
    f2 = pi()/2*rand(1,m); %polar emission angle (0-pi/2 radians)

    ct1 = cos(t1);
    st1 = sin(t1);
    ct2 = cos(t2);
    ct22 = ct2.^2;
    st2 = sin(t2);
    st22 = st2.^2;
    tt2 = tan(t2);
    cf2 = cos(f2);
    sf2 = sin(f2);
    tf2 = tan(f2);

    h = r.*ct1;
    k = r.*st1;

    x = ct2.*((d^2/4 - h.^2.*st22 + 2*h.*k.*st2.*ct2 - k.^2.*ct22).^0.5 + h.*ct2 + k.*st2);
    y = tt2.*x;
    z = x.*cf2./(sf2.*ct2);

    wm = w*ones(1,m);
    z3 = min(z,wm);

    x3 = x.*z3./z;
    y3 = y.*z3./z;

    p = (x3.^2 + y3.^2 + z3.^2).^0.5;
    x3 = x3-h;
    y3 = y3-k;
```

```
% Fission product skewing and coating correction
% uranium coating
wmu = (u/1000*ones(1,m)-zh);
zu = min(z,wmu);
xu = x.*zu./z;
yu = y.*zu./z;
pu = (xu.^2 + yu.^2 + zu.^2).^0.5;

% platinum overcoat layer
wmo = (o/1000)*ones(1,m);
zo = max(min(z-zu,wmo),0);
xo = x.*zo./z;
yo = y.*zo./z;
po = (xo.^2 + yo.^2 + zo.^2).^0.5;

% argon fill gas
wma = w*ones(1,m);
za = max(min(z-zu-zo,wma),0);
xa = x.*za./z;
ya = y.*za./z;
pa = (xa.^2 + ya.^2 + za.^2).^0.5;

% Coating energy absorption
ea = zeros(size(IntIoniz,1),m);
for i = 1:size(IntIoniz,1)
    eu = max(0,interp1(xxU,IntIonizU(i,:),pu,'pchip',IntIonizU(i,end)));
    do = max(0,interp1(IntIonizPt(i,1:IIlastPt(1,i)),xxPt(1,1:IIlastPt(1,i)),eu,'pchip',' ...
        extrap'));
    eo = max(0,interp1(xxPt,IntIonizPt(i,:),do+po,'pchip',IntIonizPt(i,end)));
    da = max(0,interp1(IntIoniz(i,1:IIlast(1,i)),xx(1,1:IIlast(1,i)),eo,'pchip','extrap'));
    ea(i,:) = max(0,interp1(xx,IntIoniz(i,:),da+pa,'pchip',IntIoniz(i,end)));
    ea(i,:) = max(ea(i,:)-eo,zeros(1,m));
end

% Generate histogram and plot for energy deposition
[histFis,binsFis] = plothist(histFis,ea,data,engmax,0,8192,3,3,1,noplot,'Energy Depositon  ...
    Spectrum','Energy Deposition (MeV)','Number of paths');

% Generate histogram and plot for 3D Spectrum
[hist3D,bins3D] = plothist(hist3D,p,1,binmax,0,nbins,1,3,1,noplot,'3D Spectrum','Path  ...
    Length (mm)','Number of paths');

% Shultis code - 2D spectrum
b = d/2 - sign(ct2).*r;
zi = b./tf2;
zm = min(zi,wm);
si = (b./sf2).*(zm./zi); % Shultis result

% Generate histogram and plot for 2D Spectrum
[histJKS,binsJKS] = plothist(histJKS,si,1,binmax,0,nbins,2,3,1,noplot,'Shultis Spectrum', ...
    'Path Length (mm)','Number of paths');
```

```
    sd = sd + m;
    et = toc;
    tet = tet + et;
    tic;
    save(sname,'hist3D','bins3D','histFis','binsFis','histJKS','binsJKS','d ...
        ','w','u','o','sd','tet');
    edate = datestr(now+(s-sd)*(et/m)/86400);
    disp(['u=',num2str(u),' w=',num2str(w),' d=',num2str(d),' o=',num2str(o),' n=', ...
        num2str(sd),' ',num2str(et),' sec (',num2str(tet),' sec) Est. completion ...
        = ',edate]);
end
copyfile(sname,fname,'f');
delete(lname);
et = toc;
end
```

End MCPulseDist.m

B.2.2 TRIMsim.m

This program operates the SRIM program (Section §B.1.4) for all of the fission fragments and stores the results for later use.

Begin TRIMsim.m

```
clear all; clc; close all;

%% set operation range
istart = 185; %set to 0 for normal operation
iend = 185; %set to 0 for normal operation

%% Load data
filepathC = 'C:\Program Files\SRIM 2003\';
filepathD = [pwd,'\'];
% Load A,Z,data from ENDEF U235 fission product yields (FisFragRead.m)
load([filepathD,'U235.mat']);
ZA = Z*1000+A;
N = A-Z;
Mass = A;
fid = fopen([filepathD,'mass_mas03.txt'],'r');
if fid == -1
    return
else
    for i = 1:1:40
        if feof(fid)
            break
        end
        c = fgetl(fid);
```

```
    end
    i = 1;
    while 1
        if feof(fid)
            break
        end
        c = fgetl(fid);
        mN(1,i) = str2num(c(1,7:9));
        mZ(1,i) = str2num(c(1,12:14));
        mA(1,i) = str2num(c(1,17:19));
        mName(i,:) = c(1,21:22);
        mMc = c(1,97:112);
        mMc = strrep(mMc,'  ','');
        mMc = strrep(mMc,'#','');
        mM(1,i) = str2num(mMc)/1E6;
        mZA = mZ*1000+mA;
        i = i + 1;
    end
end
fclose(fid);
% Find FP masses
for i = 1:1:size(ZA,2)
    M2 = [233-A(1,i) 234-A(1,i)];
    for j = 1:1:size(mZA,2)
        if Z(1,i) == mZ(1,j)
            Name(i,:) = mName(j,:);
        end
        if ZA(1,i) == mZA(1,j)
            Mass(1,i) = mM(1,j);
        end
        if mZ(1,j) == 92 - Z(1,i)
            if mN(1,j) == 141 - N(1,i)
                M2(1,1) = mM(1,j);
            elseif mN(1,j) == 142 - N(1,i)
                M2(1,2) = mM(1,j);
            end
        end
    end
    Mass(2,i) = mean(M2);
    E(1,i) = 168000*Mass(2,i)/(Mass(2,i)+Mass(1,i));
end
Mass(3,:) = Mass(1,:)+Mass(2,:);
clear('M2','data','mA','mM','mMc','mN','mName','mZ','mZA');

count = 1000;
t = [1.9 10.1 40]*1E7;
Dist = zeros(size(Z,2),size(t,2)*100);
Ioniz = Dist;
Recoil = Dist;
t2 = [0.00010 0.010]*1E7;
DistU = zeros(size(Z,2),size(t2,2)*100);
```

```
IonizU = DistU;
RecoilU = DistU;
t3 = [0.00010 0.010]*1E7;
DistPt = zeros(size(Z,2),size(t2,2)*100);
IonizPt = DistPt;
RecoilPt = DistPt;
tic;
icurr = 0;
if exist([filepathD,'Ioniz.mat'],'file') == 0
    if istart == 0
        istart = 1;
    end
else
    load([filepathD,'Ioniz.mat']);
    if istart == 0
        istart = i+1;
    else
        icurr = i;
    end
end
if iend == 0
    iend = size(Z,2);
end
for i = istart:iend
    et = toc;
    tic;
    edate = datestr(now+(size(Z,2)-i+1)*et/86400);
    disp(['i=',num2str(i),'  ',Name(i,:),'  ',num2str(ZA(1,i)),' last run time = ', ...
        num2str(et),' sec Est. completion = ',edate]);
    d = 1;
    clear('iIoniz');
    for j = 1:size(t,2)
        fid = fopen([filepathC,'TRIMAUTO'], 'w');
        if fid == -1
            return
        else
            fprintf(fid,'1\r\r');
            fprintf(fid,'TRIMAUTO allows the running of TRIM in batch mode ( ...
                without any keyboard inputs).\r');
            fprintf(fid,'This feature is controlled by the number in line #1 ...
                (above).\r');
            fprintf(fid,' 0 = Normal TRIM - New Calculation based on TRIM.IN ...
                made by setup program.\r');
            fprintf(fid,' 1 = Auto TRIM - TRIM based on TRIM.IN. No inputs  ...
                required. Terminates after all ions.\r');
            fprintf(fid,' 2 = RESUME - Resume old TRIM calculation based on  ...
                files: SRIM Restore\\*.SAV.\r\r');
            fprintf(fid,' Line #2 of this file is the Directory of Resumed  ...
                data, e.g. A:\\TRIM2\\\r');
            fprintf(fid,' If empty, the default is the \''\''SRIM\\SRIM  ...
                Restore\''\'' directory.\r\r');
```

```
fprintf(fid,'See the file TRIMAUTO.TXT for more details.');
fclose(fid);
fid = fopen([filepathC,'TRIM.IN'], 'w');
if fid == -1
    return
else
    fprintf(fid,'==> SRIM-2003.26 This file controls TRIM  ...
        Calculations.\r');
    fprintf(fid,'Ion: Z1 , M1, Energy (keV), Angle,Number,Bragg  ...
        Corr,AutoSave Number.\r');
    fprintf(fid,' %3i %7.3f %9.2f 0 %6i 0 %6i\r',Z(1,i),Mass(1,i),E(1,i), ...
        count,count);
    fprintf(fid,'Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7= ...
        Neutrons), Random Number Seed, Reminders\r');
    fprintf(fid,' 1 0\r');
    fprintf(fid,'Diskfiles (0=no,1=yes): Ranges, Backscatt,  ...
        Transmit, Sputtered, Recoils (0=none;1=Some;2=All  ...
        Details;>10=EXYZ Output)\r');
    fprintf(fid,' 0 0 0 0 0\r');
    fprintf(fid,'Target material : Number of Elements & Layers\r');
    fprintf(fid,'"%s (%9.2f) into FillGas " 1 1\r',Name(i,:),E(1,i));
    fprintf(fid,'Target Energies (eV): Binding, Surface,  ...
        Individual Displacement\r');
    fprintf(fid,' 1 2 5\r');
    fprintf(fid,'PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.)  ...
        [=0 0 for Viewing Full Target]\r');
    fprintf(fid,' 5 0 0\r');
    fprintf(fid,'Target Elements: Z Mass(amu)\r');
    fprintf(fid,'Atom 1 = Ar= 18 39.948\r');
    fprintf(fid,'Layer Layer Name / Width Density Ar(18)\r');
    fprintf(fid,'Numb. Description (Ang) (g/cm3) Stoich\r');
    fprintf(fid,' 1 "FillGas" %9.2E .001784 1\r',t(1,j));
    fprintf(fid,' 0 Target layer phases (0=Solid, 1=Gas)');
    fclose(fid);
    cd(filepathC);
    status = system('TRIM.exe');
    cd(filepathD);
    if status == 0
        fid = fopen([filepathC,'IONIZ.txt'],'r');
        if fid == -1
            return
        else
            for x = 1:1:26
                if feof(fid)
                    break
                end
                c = fgetl(fid);
            end
            while 1
                if feof(fid)
                    break
```

```
                        end
                        c = fgetl(fid);
                        iIoniz(1,d) = str2num(c(1,1:11))/1E7;
                        iIoniz(2,d) = str2num(c(1,14:24));
                        iIoniz(3,d) = str2num(c(1,27:37));
                        d = d + 1;
                    end
                end
            end
        end
    end
end
iIoniz = (sortrows(iIoniz',1))';
Dist(i,:) = iIoniz(1,:); % mm
Ioniz(i,:) = iIoniz(2,:); % eV/ang
Recoil(i,:) = iIoniz(3,:); % eV/ang
d = 1;
clear('iIoniz');
for j = 1:size(t2,2)
    fid = fopen([filepathC,'TRIMAUTO'],'w');
    if fid == -1
        return
    else
        fprintf(fid,'1\r\r');
        fprintf(fid,'TRIMAUTO allows the running of TRIM in batch mode ( ...
            without any keyboard inputs).\r');
        fprintf(fid,'This feature is controlled by the number in line #1 ...
             (above).\r');
        fprintf(fid,' 0 = Normal TRIM - New Calculation based on TRIM.IN ...
             made by setup program.\r');
        fprintf(fid,' 1 = Auto TRIM - TRIM based on TRIM.IN. No inputs  ...
            required. Terminates after all ions.\r');
        fprintf(fid,' 2 = RESUME - Resume old TRIM calculation based on  ...
            files: SRIM Restore\\*.SAV.\r\r');
        fprintf(fid,' Line #2 of this file is the Directory of Resumed  ...
            data, e.g. A:\\TRIM2\\\r');
        fprintf(fid,' If empty, the default is the \''\''SRIM\\SRIM  ...
            Restore\''\'' directory.\r\r');
        fprintf(fid,'See the file TRIMAUTO.TXT for more details.');
        fclose(fid);
        fid = fopen([filepathC,'TRIM.IN'],'w');
        if fid == -1
            return
        else
            fprintf(fid,'==> SRIM-2003.26 This file controls TRIM  ...
                Calculations.\r');
            fprintf(fid,'Ion: Z1 , M1, Energy (keV), Angle,Number,Bragg  ...
                Corr,AutoSave Number.\r');
            fprintf(fid,' %3i %7.3f %9.2f 0 %6i 0 %6i\r',Z(1,i),Mass(1,i),E(1,i), ...
                count,count);
            fprintf(fid,'Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7= ...
```

```
                Neutrons), Random Number Seed, Reminders\r');
            fprintf(fid,' 1 0\r');
            fprintf(fid,'Diskfiles (0=no,1=yes): Ranges, Backscatt,  ...
                Transmit, Sputtered, Recoils (0=none;1=Some;2=All  ...
                Details;>10=EXYZ Output)\r');
            fprintf(fid,' 0 0 0 0 0\r');
            fprintf(fid,'Target material : Number of Elements & Layers\r');
            fprintf(fid,'"%s (%9.2f) into FisCoat " 1 1\r',Name(i,:),E(1,i));
            fprintf(fid,'Target Energies (eV): Binding, Surface,  ...
                Individual Displacement\r');
            fprintf(fid,' 3 5.42 25\r');
            fprintf(fid,'PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.)  ...
                [=0 0 for Viewing Full Target]\r');
            fprintf(fid,' 5 0 0\r');
            fprintf(fid,'Target Elements: Z Mass(amu)\r');
            fprintf(fid,'Atom 1 = U = 92 238.03\r');
            fprintf(fid,'Layer Layer Name / Width Density Ar(18)\r');
            fprintf(fid,'Numb. Description (Ang) (g/cm3) Stoich\r');
            fprintf(fid,' 1 "FisCoat" %9.2E 19.043 1\r',t2(1,j));
            fprintf(fid,' 0 Target layer phases (0=Solid, 1=Gas)');
            fclose(fid);
            cd(filepathC);
            status = system('TRIM.exe');
            cd(filepathD);
            if status == 0
                fid = fopen([filepathC,'IONIZ.txt'],'r');
                if fid == -1
                    return
                else
                    for x = 1:1:26
                        if feof(fid)
                            break
                        end
                        c = fgetl(fid);
                    end
                    while 1
                        if feof(fid)
                            break
                        end
                        c = fgetl(fid);
                        iIoniz(1,d) = str2num(c(1,1:11))/1E7;
                        iIoniz(2,d) = str2num(c(1,14:24));
                        iIoniz(3,d) = str2num(c(1,27:37));
                        d = d + 1;
                    end
                end
            end
        end
    end
end
iIoniz = (sortrows(iIoniz',1))';
```

```
DistU(i,:) = iIoniz(1,:); % mm
IonizU(i,:) = iIoniz(2,:); % eV/ang
RecoilU(i,:) = iIoniz(3,:); % eV/ang
d = 1;
clear('iIoniz');
for j = 1:size(t3,2)
    fid = fopen([filepathC,'TRIMAUTO'], 'w');
    if fid == -1
        return
    else
        fprintf(fid,'1\r\r');
        fprintf(fid,'TRIMAUTO allows the running of TRIM in batch mode ( ...
            without any keyboard inputs).\r');
        fprintf(fid,'This feature is controlled by the number in line #1 ...
             (above).\r');
        fprintf(fid,' 0 = Normal TRIM - New Calculation based on TRIM.IN ...
             made by setup program.\r');
        fprintf(fid,' 1 = Auto TRIM - TRIM based on TRIM.IN. No inputs  ...
            required. Terminates after all ions.\r');
        fprintf(fid,' 2 = RESUME - Resume old TRIM calculation based on  ...
            files: SRIM Restore\\*.SAV.\r\r');
        fprintf(fid,' Line #2 of this file is the Directory of Resumed  ...
            data, e.g. A:\\TRIM2\\\r');
        fprintf(fid,' If empty, the default is the \''\''SRIM\\SRIM  ...
            Restore\''\'' directory.\r\r');
        fprintf(fid,'See the file TRIMAUTO.TXT for more details.');
        fclose(fid);
        fid = fopen([filepathC,'TRIM.IN'], 'w');
        if fid == -1
            return
        else
            fprintf(fid,'==> SRIM-2003.26 This file controls TRIM  ...
                Calculations.\r');
            fprintf(fid,'Ion: Z1 , M1, Energy (keV), Angle,Number,Bragg  ...
                Corr,AutoSave Number.\r');
            fprintf(fid,' %3i %7.3f %9.2f 0 %6i 0 %6i\r',Z(1,i),Mass(1,i),E(1,i), ...
                count,count);
            fprintf(fid,'Cascades(1=No;2=Full;3=Sputt;4-5=Ions;6-7= ...
                Neutrons), Random Number Seed, Reminders\r');
            fprintf(fid,' 1 0\r');
            fprintf(fid,'Diskfiles (0=no,1=yes): Ranges, Backscatt,  ...
                Transmit, Sputtered, Recoils (0=none;1=Some;2=All  ...
                Details;>10=EXYZ Output)\r');
            fprintf(fid,' 0 0 0 0 0\r');
            fprintf(fid,'Target material : Number of Elements & Layers\r');
            fprintf(fid,'"%s (%9.2f) into PtCap " 1 1\r',Name(i,:),E(1,i));
            fprintf(fid,'Target Energies (eV): Binding, Surface,  ...
                Individual Displacement\r');
            fprintf(fid,' 3 5.86 25\r');
            fprintf(fid,'PlotType (0-5); Plot Depths: Xmin, Xmax(Ang.)  ...
                [=0 0 for Viewing Full Target]\r');
```

```
                    fprintf(fid,' 5 0 0\r');
                    fprintf(fid,'Target Elements: Z Mass(amu)\r');
                    fprintf(fid,'Atom 1 = Pt = 78 195.08\r');
                    fprintf(fid,'Layer Layer Name / Width Density Ar(18)\r');
                    fprintf(fid,'Numb. Description (Ang) (g/cm3) Stoich\r');
                    fprintf(fid,' 1 "PtCap" %9.2E 21.45 1\r',t3(1,j));
                    fprintf(fid,' 0 Target layer phases (0=Solid, 1=Gas)');
                    fclose(fid);
                    cd(filepathC);
                    status = system('TRIM.exe');
                    cd(filepathD);
                    if status == 0
                        fid = fopen([filepathC,'IONIZ.txt'],'r');
                        if fid == -1
                            return
                        else
                            for x = 1:1:26
                                if feof(fid)
                                    break
                                end
                                c = fgetl(fid);
                            end
                            while 1
                                if feof(fid)
                                    break
                                end
                                c = fgetl(fid);
                                iIoniz(1,d) = str2num(c(1,1:11))/1E7;
                                iIoniz(2,d) = str2num(c(1,14:24));
                                iIoniz(3,d) = str2num(c(1,27:37));
                                d = d + 1;
                            end
                        end
                    end
                end
            end
        end
    end
    iIoniz = (sortrows(iIoniz',1))';
    DistPt(i,:) = iIoniz(1,:); % mm
    IonizPt(i,:) = iIoniz(2,:); % eV/ang
    RecoilPt(i,:) = iIoniz(3,:); % eV/ang
    if icurr ~= 0
        ithis = i;
        i = icurr;
    end
    save([filepathD,'Ioniz.mat'],'Dist','Ioniz','Recoil','DistU','IonizU',' ...
        RecoilU','DistPt','IonizPt','RecoilPt','Name','Mass','E','i');
    if icurr ~= 0
        i = ithis;
    end
end
```

```
et = toc;
```

End TRIMsim.m

B.2.3 IntTRIM.m

This program integrates the energy loss per unit distance data from TRIMsim.m (Section §B.2.2) in order to get data which gives the total energy at a certain distance. This integrated data is then utilized by MCPulseDist.m (Section §B.2.1).

Begin IntTRIM.m

```
clear all; close all; clc;

filepathD = 'D:\Research\MPFD\Data\MCPulseDist\';
%load([filepathD,'U235.mat']);
load([filepathD,'Ioniz2.mat']);
clear('A','E','Mass','Name','Z','data');

for i = size(Ioniz,2):-1:1
    if sum(Ioniz(:,i)) == 0
        Ioniz(:,i) = [];
        Dist(:,i) = [];
    end
end
for i = size(IonizU,2):-1:1
    if sum(IonizU(:,i)) == 0
        IonizU(:,i) = [];
        DistU(:,i) = [];
    end
end
for i = size(IonizPt,2):-1:1
    if sum(IonizPt(:,i)) == 0
        IonizPt(:,i) = [];
        DistPt(:,i) = [];
    end
end
Dist = Dist*1e3; % convert from mm to um
DistU = DistU*1e3;
DistPt = DistPt*1e3;
Ioniz = Ioniz*1e4; % convert from eV/angstrom to eV/um
IonizU = IonizU*1e4;
IonizPt = IonizPt*1e4;

x = max(max(max(Dist))/10000,1);
xx = x:x:max(max(Dist));

IntIoniz = zeros(size(Dist,1),size(xx,2)+1);
```

```
for i = 1:size(Dist,1)
    yy = interp1(Dist(i,:),Ioniz(i,:),xx,'pchip');
    for j = 1:size(yy,2)
        IntIoniz(i,j+1) = IntIoniz(i,j)+yy(1,j)*x;
    end
end

xU = max(max(max(DistU))/10000,0.001);
xxU = xU:xU:max(max(DistU));

IntIonizU = zeros(size(DistU,1),size(xxU,2)+1);
for i = 1:size(DistU,1)
    yy = interp1(DistU(i,:),IonizU(i,:),xxU,'pchip');
    for j = 1:size(yy,2)
        IntIonizU(i,j+1) = IntIonizU(i,j)+yy(1,j)*xU;
    end
end

xPt = max(max(max(DistPt))/10000,0.001);
xxPt = xPt:xPt:max(max(DistPt));

IntIonizPt = zeros(size(DistPt,1),size(xxPt,2)+1);
for i = 1:size(DistPt,1)
    yy = interp1(DistPt(i,:),IonizPt(i,:),xxPt,'pchip');
    for j = 1:size(yy,2)
        IntIonizPt(i,j+1) = IntIonizPt(i,j)+yy(1,j)*xPt;
    end
end

xx = [0 xx/1000]; % mm
xxU = [0 xxU/1000]; % mm
xxPt = [0 xxPt/1000]; % mm
Dist = Dist/1000; % mm
DistU = DistU/1000; % mm
DistPt = DistPt/1000; % mm
Ioniz = Ioniz/10000; % eV/angstrom
IonizU = IonizU/10000; % eV/angstrom
IonizPt = IonizPt/10000; % eV/angstrom
save([filepathD,'IntIoniz.mat'],'IntIoniz','xx','Dist','Ioniz','IntIonizU',' ...
    xxU','DistU','IonizU','IntIonizPt','xxPt','DistPt','IonizPt');
```

End IntTRIM.m

B.2.4 BatchMCPulse.m

This program runs MCPulseDist.m (Section §B.2.1) in a batch mode and allows simultaneous operations on multiple computers where the starting path is synchronized between the computers.

Begin BatchMCPulse.m

```
clear all; clc; close all;
% dbstop if error;
% dbstop in MCPulseDist at 200

noplot = 1;
filepathD = [pwd,'\'];
s = 1000000; % Number of iterations (int)
um = [1e-4 5e-4 1e-3 5e-3 1e-2 5e-2 1e-1 0.5 0.75 1 1.5 2 2.5 3 4 5 7.5 10]; % 25 50 75 100 250 ...
    500 750 1000 2500 5000
sm = size(um,2);
om = [0.0*ones(1,sm) 0.5*ones(1,sm) 1.0*ones(1,sm) 0.0*ones(1,sm) 0.0*ones(1,sm) 0.0*ones(1,sm) ...
    0.0*ones(1,sm)]; % Platinum Overcoat Thickness (um)
dm = [1.0*ones(1,sm) 1.0*ones(1,sm) 1.0*ones(1,sm) 3.0*ones(1,sm) 30.0*ones(1,sm) 1.0*ones(1,sm) ...
    30.0*ones(1,sm)]; % Chamber Diameter {mm}
wm = [0.5*ones(1,sm) 0.5*ones(1,sm) 0.5*ones(1,sm) 1.0*ones(1,sm) 30.0*ones(1,sm) 30.0*ones(1, ...
    sm) 0.5*ones(1,sm)]; % Chamber Width {mm}
um = [um um um um um um um];

% um = fliplr(um); om = fliplr(om); dm = fliplr(dm); wm = fliplr(wm);

ciname = [filepathD(1,1),':\MATSpace\MCPulse\CompID.mat'];
if exist(fileparts(ciname),'dir') == 0
    mkdir(fileparts(ciname));
end
if exist(ciname,'file') == 0
    compid = [num2str(now),num2str(rand),num2str(rand)];
    save(ciname,'compid');
else
    load(ciname,'compid');
end

cnt = 0;
for im = 1:size(um,2)
    fname = [filepathD,'Spec_d',num2str(round(dm(1,im)*1000)),'um_w',num2str(round(wm(1, ...
        im)*1000)),'um_u',num2str(round(um(1,im)*10000)),'A_o',num2str(round(om(1,im) ...
        *10000)),'A.mat'];
    if exist(fname,'file') == 0
        cnt = cnt + 1;
    else
        load(fname,'-mat','sd');
        if sd < s
            cnt = cnt + 1;
        end
    end
end
disp([num2str(cnt),' runs left to go']);

sname = [filepathD(1,1),':\MATSpace\MCPulse\SpecTemp.mat'];
sok = 0;
if exist(sname,'file') == 2
```

```
    load(sname,'-mat','sd','u','o','d','w')
    if sd < s
        for im = 1:size(um,2)
            if um(1,im) == u && om(1,im) == o && dm(1,im) == d && wm(1,im) == w
                um = [u um];
                om = [o om];
                dm = [d dm];
                wm = [w wm];
                disp(['...SpecTemp found - ',num2str(sd),' of ',num2str(s)]);
                sok = 1;
                break;
            end
        end
    end
end
for im = 1:size(um,2)
    fname = [filepathD,'Spec_d',num2str(round(dm(1,im)*1000)),'um_w',num2str(round(wm(1, ...
        im)*1000)),'um_u',num2str(round(um(1,im)*10000)),'A_o',num2str(round(om(1,im) ...
        *10000)),'A.mat'];
    lname = [filepathD,'Spec_d',num2str(round(dm(1,im)*1000)),'um_w',num2str(round(wm(1, ...
        im)*1000)),'um_u',num2str(round(um(1,im)*10000)),'A_o',num2str(round(om(1,im) ...
        *10000)),'A.loc'];
    disp(['d = ',num2str(round(dm(1,im)*1000)),'um w = ',num2str(round(wm(1,im)*1000)) ...
        ,'um u = ',num2str(round(um(1,im)*10000)),'A o = ',num2str(round(om(1,im) ...
        *10000)),'A']);
    eok = 0;
    if exist(fname,'file') == 0 || sok == 1 && im == 1
        fok = 1;
    else
        load(fname,'-mat','sd')
        if sd < s
            fok = 1;
            disp(['...histories extended - ',num2str(sd),' of ',num2str(s)]);
            eok = 1;
        else
            fok = 0;
            disp('...already done');
        end
    end
    if fok == 1;
        if exist(lname,'file') == 0
            disp('...no lock found');
            fok = 1;
        else
            fid = fopen(lname,'r');
            tline = str2num(['0',fgetl(fid)]);
            mline = fgetl(fid);
            fclose(fid);
            if tline < now-1
                disp('...lock expired');
                fok = 1;
```

```
            elseif strcmp(mline,compid) == 1
                disp('...my lock found');
                fok = 1;
            else
                fok = 0;
            end
        end
        if fok == 1
            if eok == 1
                copyfile(fname,sname,'f');
            end
            fid = fopen(lname,'w');
            fprintf(fid,'%s\n',num2str(now));
            fprintf(fid,'%s\n',compid);
            fclose(fid);
            MCPulseDist(um(1,im),om(1,im),dm(1,im),wm(1,im),s,noplot);
        else
            disp('...locked');
        end
    end
end
```

End BatchMCPulse.m

B.3 Background Contribution and Lower Level Discriminator Setting

B.3.1 EDep.m

This program generated Figures 3.12, 3.13 and 3.14 by compiling MCNP and SRIM data for the energy deposition over a specific distance.

Begin EDep.m

```
% ...
    %%%%
```

```
clc;

CS = [0.25, 0.5, 1, 5, 10, 50, 100, 250]; % width (mm)
GammaMaxE = 11258900; %eV for Na-20
ElecMaxE = 19636000; %eV for Er-150 (Beta+) 10063 keV for B-14 (Beta-) 4554.2 keV for K-48 ( ...
    Elec)
AlphaMaxE = 11650000; %eV for Po-212

filepathD = [pwd,'\'];
%% Average Fission Fragment Energy Deposition
load([filepathD(1,1),':\Research\MPFD\Data\MCPulseDist\U235.mat'],'data');
load([filepathD(1,1),':\Research\MPFD\Data\MCPulseDist\IntIoniz.mat'],' ...
    IntIoniz','xx');
I = gt(data,0); % Remove intensities <= 0
IntIoniz = IntIoniz(I,:);
data = data(:,I);

Inum = size(IntIoniz,1);
IIlast = Inum*ones(1,Inum);
for i = 1:Inum
    for j = size(IntIoniz,2):-1:2
        if IntIoniz(i,j) ~= IntIoniz(i,j-1)
            IIlast(1,i) = j;
            break;
        end
    end
end
ff = zeros(size(IntIoniz,1),size(CS,2));
for i = 1:size(IntIoniz,1)
    ff(i,:) = interp1(xx(1,1:IIlast(1,i)),IntIoniz(i,1:IIlast(1,i)),CS,'pchip',IntIoniz(i,IIlast(1,i)));
end
s = (data*ff)'/sum(data)
clear I IntIoniz data xx Inum IIlast ff;

%%% Gamma Energy Dep %%%
load([filepathD,'Gamma.dat']); % load data
Gd = abs(Gamma); % remove negatives
G = log10(Gd); % take log of data
gxa = (0:0.00001:min(G(:,1)))';
gxb = (gxa(end,1):0.00001:log10(GammaMaxE))';
gx = [gxa;gxb];
% y = (0:0.00001:log10(1e8))';
gxy = gx;
z1a = (0:0.005:min(G(:,1)))';
z1b = (z1a(end,1):0.005:max(G(:,1)))';
z1 = [z1a;z1b];
g1 = z1;
for i = 2:9;
    j = 1;
    d = 0;
    while d == 0
```

```
            k = (0:0.005:j)';
            P = union([k k],[G(:,1) G(:,i)],'rows');
% g1(:,i) = spline(P(:,1),P(:,2),z1);
            g1a = spline(P(:,1),P(:,2),z1a);
            g1b = pchip(P(:,1),P(:,2),z1b);
            g1(:,2) = [g1a;g1b];
            d = 1;
            j = j + 0.005;
            for m = 1:size(g1,1)
                if (g1(m,2) > g1(m,1) && j < G(size(G,1),1))
                    d = 0;
                end
            end
        end
        gxya = spline(P(:,1),P(:,2),gxa);
        gxyb = pchip(P(:,1),P(:,2),gxb);
        gxy(:,i) = [gxya;gxyb];
% gxy(:,i) = spline(P(:,1),P(:,2),gx);
end
G = 10.^gxy;
clear g1a g1b gxya gxyb z1a z1b gxa gxb gxy g1 gx k z1 P Gamma;
[Gmax,GImax] = max(G(1:min(size(G,1),ceil(log10(GammaMaxE)/(log10(G(2,1))-log10(G(1,1))))) ...
    ,2:9));
Gmax = [Gmax' Gmax'];
Gmax(:,1) = G(GImax,1);
Gmax

figure(1); clf;
loglog(G(:,1),G(:,9),'-k',Gd(:,1),Gd(:,9),'.k',G(:,1),G(:,8),'-k',Gd(:,1),Gd(:,8),'.k',G(:,1),G(:,7), ...
    '-k',Gd(:,1),Gd(:,7),'.k',G(:,1),G(:,6),'-k',Gd(:,1),Gd(:,6),'.k',G(:,1),G(:,5),'-k',Gd(:,1), ...
    Gd(:,5),'.k',G(:,1),G(:,4),'-k',Gd(:,1),Gd(:,4),'.k',G(:,1),G(:,3),'-k',Gd(:,1),Gd(:,3),'.k ...
    ',G(:,1),G(:,2),'-k',Gd(:,1),Gd(:,2),'.k');
title('Gamma-ray Energy Deposition in Various Chamber Widths');
ylabel('Energy Deposition (eV)');
xlabel('Gamma-ray Origination Energy (eV)');
xlim([1 GammaMaxE]);
ylim([0.1 max(max(G(:,2:9)))*1.1]);
setplot(12,12,11);

annotation(gcf,'textbox',...
    'Position',[0.8268 0.1067 0.06488 0.0746],...
    'String',{'(mm)'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6715 0.1814 0.07202 0.0746],...
    'String',{'0.25'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6778 0.2243 0.05952 0.0746],...
    'String',{'0.5'},'FitHeightToText','on','LineStyle','none');
```

```
annotation(gcf,'textbox',...
    'Position',[0.683 0.2672 0.04914 0.0746],...
    'String',{'1'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.683 0.3719 0.04914 0.0746],...
    'String',{'5'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6813 0.4291 0.05238 0.0746],...
    'String',{'10'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6813 0.5362 0.05238 0.0746],...
    'String',{'50'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6751 0.591 0.06488 0.0746],...
    'String',{'100'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6751 0.66 0.06488 0.0746],...
    'String',{'250'},'FitHeightToText','on','LineStyle','none');

%%% Electron Energy Dep %%%
load 'Electron.dat';
Ed = abs(Electron); % remove negatives
E = log10(Ed); % take log of data
exa = (0:0.00001:min(E(:,1)))';
exb = (exa(end,1):0.00001:log10(ElecMaxE))';
ex = [exa;exb];
% y = (0:0.00001:log10(1e8))';
exy = ex;
y1a = (0:0.005:min(E(:,1)))';
y1b = (y1a(end,1):0.005:max(E(:,1)))';
y1 = [y1a;y1b];
e1 = y1;
for i = 2:9;
    j = 1;
    d = 0;
    while d == 0
        k = (0:0.005:j)';
        Q = union([k k],[E(:,1) E(:,i)],'rows');
        e1a = spline(Q(:,1),Q(:,2),y1a);
        e1b = pchip(Q(:,1),Q(:,2),y1b);
        e1(:,2) = [e1a;e1b];
        d = 1;
        j = j + 0.005;
        for m = 1:size(e1,1)
            if (e1(m,2) > e1(m,1) && j < E(size(E,1),1))
                d = 0;
```

```
            end
        end
    end
    exya = spline(Q(:,1),Q(:,2),exa);
    exyb = pchip(Q(:,1),Q(:,2),exb);
    exy(:,i) = [exya;exyb];
end
E = 10.^exy;
clear e1a e1b exya exyb exa exb y1a y1b exy e1 k e1 Q Electron;
[Emax,EImax] = max(E(1:min(size(E,1),ceil(log10(ElecMaxE)/(log10(E(2,1))-log10(E(1,1))))),2:9));
Emax = [Emax' Emax'];
Emax(:,1) = E(EImax,1);
Emax

figure(2); clf;
loglog(E(:,1),E(:,9),'-k',Ed(:,1),Ed(:,9),'.k',E(:,1),E(:,8),'-k',Ed(:,1),Ed(:,8),'.k',E(:,1),E(:,7), ...
    '-k',Ed(:,1),Ed(:,7),'.k',E(:,1),E(:,6),'-k',Ed(:,1),Ed(:,6),'.k',E(:,1),E(:,5),'-k',Ed(:,1),Ed ...
    (:,5),'.k',E(:,1),E(:,4),'-k',Ed(:,1),Ed(:,4),'.k',E(:,1),E(:,3),'-k',Ed(:,1),Ed(:,3),'.k',E(:,1) ...
    ,E(:,2),'-k',Ed(:,1),Ed(:,2),'.k');
title('Electron Energy Deposition in Various Chamber Widths');
ylabel('Energy Deposition (eV)');
xlabel('Electron Origination Energy (eV)');
xlim([100 ElecMaxE]);
ylim([10 max(max(E(:,2:9)))*1.1]);
setplot(12,12,11);

annotation(gcf,'textbox',...
    'Position',[0.8268 0.1067 0.06488 0.0746],...
    'String',{'(mm)'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6996 0.2195 0.07202 0.0746],...
    'String',{'0.25'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7059 0.2767 0.05952 0.0746],...
    'String',{'0.5'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7111 0.3362 0.04914 0.0746],...
    'String',{'1'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7111 0.4695 0.04914 0.0746],...
    'String',{'5'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7095 0.5291 0.05238 0.0746],...
    'String',{'10'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
```

```
    'Position',[0.7095 0.6624 0.05238 0.0746],...
    'String',{'50'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7032 0.7219 0.06488 0.0746],...
    'String',{'100'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7032 0.7933 0.06488 0.0746],...
    'String',{'250'},'FitHeightToText','on','LineStyle','none');

%%% Alpha Energy Dep %%%
load 'Ioniz.mat';
% Ioniz = Ioniz + Recoil;
t = [0.53 3.7 10.1 50.3]*1E7;
for i = 1:size(t,2)
    d0(1+(i-1)*100:i*100,1) = t(1,i)/100:t(1,i)/100:t(1,i);
end
[dx,L] = sort(d0,1);
i = 0;
L = [L (1:1:size(L,1))'];
while i == 0
    dx = ones(size(L,1),1) & [lt(L(1:end-1,1),L(2:end,1));1] & ne(mod(L(:,1),100),0);
% dx = ones(size(L,1),1) & ne(mod(L(:,1),100),0);
    if sum(~dx) > 0
        i = 0;
        L = L([dx],:);
    else
        i = 1;
% d1 = d0([L([dx],1)],1);
    end
end
Ioniz = Ioniz(:,[L(:,2)]);
Dist = Dist(:,[L(:,2)]);
k = [96];
for i = 1:size(Ioniz,1)
    j = find(le(Ioniz(i,:),0),1,'first');
    if ~isempty(j)
        Ioniz(i,j:end) = zeros(1,size(Ioniz,2)-j+1);
        if j <= 5
            k = [k j];
        end
    end
end
Ioniz([k],:) = [];
Dist([k],:) = [];
E(:,[k]) = [];

% Rs = [122 225];
% for i = size(Rs,2):-1:1
% Ioniz(:,Rs(1,i)) = [];
```

```
% Dist(:,Rs(1,i)) = [];
% end
for i = size(Ioniz,2):-1:1
    if sum(Ioniz(:,i)) == 0
        Ioniz(:,i) = [];
        Dist(:,i) = [];
    end
end
Dist = Dist*1e3; % convert from mm to um
Ioniz = Ioniz*1e4; % convert from eV/angstrom to eV/um
x = max(max(max(Dist))/10000,1);
xx = x:x:max(max(Dist));

IntIoniz = zeros(size(Dist,1),size(xx,2)+1);
for i = 1:size(Dist,1)
    yy = interp1(Dist(i,:),Ioniz(i,:),xx,'spline');
    for j = 1:size(yy,2)
        IntIoniz(i,j+1) = IntIoniz(i,j)+yy(1,j)*x;
    end
end
xx = [0 xx/1000]; % mm
Dist = Dist/1000; % mm
Ioniz = Ioniz/10000; % eV/angstrom
Alpha = zeros(size(Dist,1),size(CS,2)+1);
Alpha(:,1) = E'*1000;
for i = 1:size(IntIoniz,1)
    Alpha(i,2:size(CS,2)+1) = interp1(xx,IntIoniz(i,:),CS,'pchip');
    Alpha(i,2:size(CS,2)+1) = min([max(Alpha(i,1))*ones(1,size(CS,2));abs(Alpha(i,2:end))],[],1);
end

Ad = abs(Alpha); % remove negatives
A = log10(Ad); % take log of data
ax = (0:0.00001:log10(AlphaMaxE))';
A = 10.^ax;
for i = 2:9
    A(:,i) = spline(Ad(:,1),Ad(:,i),A(:,1));
end

[Amax,AImax] = nanmax(A(1:min(size(A,1),ceil(log10(AlphaMaxE)/(log10(A(2,1))-log10(A(1,1))))) ...
    ,2:9));
Amax = [Amax' Amax'];
Amax(:,1) = A(AImax,1);
Amax

figure(3); clf;
loglog(A(:,1),A(:,9),'-k',Ad(:,1),Ad(:,9),'.k',A(:,1),A(:,8),'-k',Ad(:,1),Ad(:,8),'.k',A(:,1),A(:,7), ...
    '-k',Ad(:,1),Ad(:,7),'.k',A(:,1),A(:,6),'-k',Ad(:,1),Ad(:,6),'.k',A(:,1),A(:,5),'-k',Ad(:,1), ...
    Ad(:,5),'.k',A(:,1),A(:,4),'-k',Ad(:,1),Ad(:,4),'.k',A(:,1),A(:,3),'-k',Ad(:,1),Ad(:,3),'.k ...
    ',A(:,1),A(:,2),'-k',Ad(:,1),Ad(:,2),'.k');
% loglog(Ad(:,1),Ad(:,9),'.k',Ad(:,1),Ad(:,8),'.k',Ad(:,1),Ad(:,7),'.k',Ad(:,1),Ad(:,6),'.k',Ad(:,1),Ad(:,5),'. ...
    k',Ad(:,1),Ad(:,4),'.k',Ad(:,1),Ad(:,3),'.k',Ad(:,1),Ad(:,2),'.k');
```

```
title('Alpha Energy Deposition in Various Chamber Widths');
ylabel('Energy Deposition (eV)');
xlabel('Alpha Origination Energy (eV)');
xlim([1000 AlphaMaxE]);
ylim([400 1.2e7]);
setplot(12,12,11);

annotation(gcf,'textbox',...
    'Position',[0.8268 0.1067 0.06488 0.0746],...
    'String',{'(mm)'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6233 0.4409 0.07202 0.0746],...
    'String',{'0.25'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6367 0.4957 0.05952 0.0746],...
    'String',{'0.5'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.6616 0.5505 0.04914 0.0746],...
    'String',{'1'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7205 0.6648 0.04914 0.0746],...
    'String',{'5'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.7474 0.7053 0.05238 0.0746],...
    'String',{'10'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.8313 0.7862 0.05238 0.0746],...
    'String',{'50'},'FitHeightToText','on','LineStyle','none');

annotation(gcf,'textbox',...
    'Position',[0.8019 0.8553 0.06488 0.0746],...
    'String',{'100+'},'FitHeightToText','on','LineStyle','none');

Ep = E'*1000;
a = (Ep - max(IntIoniz,[],2))./Ep;
```

End EDep.m

B.3.2 EDepCompAlpha.m

This program uses data compiled by EDep.m (Section §B.3.1) to compare the maximum energy deposition of background to the energy deposition by some typical fission fragments

(Figure 3.15).

Begin EDepCompAlpha.m

```
close all;
clear all;
clc;

G = [0.25 3.122408488616467e+002;
     0.5 4.855597763990426e+002;
     1 7.543216324904238e+002;
     5 1.953650396386453e+003;
     10 2.772492097454128e+003;
     50 5.390179999481123e+003;
     100 6.982650000000001e+003;
     250 9.375929999999995e+003];
E = [0.25 2.315867714481273e+003;
     0.5 4.370087504549922e+003;
     1 6.665129999855697e+003;
     5 1.751030000000003e+004;
     10 2.653849999933362e+004;
     50 6.816539999810261e+004;
     100 9.941059999999995e+004;
     250 1.861919999999995e+005];
A = [0.25 5.842435331852041e+004;
     0.5 1.166937948006100e+005;
     1 2.330792375977684e+005;
     5 1.083301142666263e+006;
     10 1.941862547342161e+006;
     50 6.020989857766805e+006;
     100 8.772439554271489e+006;
     250 1.164984105255834e+007];
load('fiss_products.mat','ff1','ff2');
F = [ff2 ff1(:,2)]; clear ff1 ff2;

F = [F zeros(size(F,1),2)];
for i = 2:size(F,1)
    F(i,4) = (F(i,2)+F(i-1,2))*(F(i,1)-F(i-1,1))*1E7/2+F(i-1,4);
    F(i,5) = (F(i,3)+F(i-1,3))*(F(i,1)-F(i-1,1))*1E7/2+F(i-1,5);
end
FI = pchip(F(:,1),F(:,4),G(:,1));
FB = pchip(F(:,1),F(:,5),G(:,1));
C = 10.^(log10(0.25):0.001:log10(250));
FIC = pchip(F(:,1),F(:,4),C);
FBC = pchip(F(:,1),F(:,5),C);
GC = pchip(G(:,1),G(:,2),C);
EC = pchip(E(:,1),E(:,2),C);
AC = pchip(A(:,1),A(:,2),C);
figure(1);
clf('reset');
loglog1 = loglog(G(:,1),G(:,2),E(:,1),E(:,2),A(:,1),A(:,2),G(:,1),FI,G(:,1),FB,C,GC,C,EC,C,AC,C,FIC,C,  ...
    FBC);
```

```
setplot(12,12,11);
set(loglog1(1),...
  'Color',[0.1686 0.5059 0.3373],...
  'LineStyle','none',...
  'Marker','.',...
  'MarkerSize',20);
set(loglog1(2),...
  'Color',[0.04314 0.5176 0.7804],...
  'LineStyle','none',...
  'Marker','.',...
  'MarkerSize',20);
set(loglog1(3),...
  'Color',[1 0 1],...
  'LineStyle','none',...
  'Marker','.',...
  'MarkerSize',20);
set(loglog1(4),...
  'Color',[1 0 0],...
  'LineStyle','none',...
  'Marker','.',...
  'MarkerSize',20);
set(loglog1(5),...
  'Color',[0 0 1],...
  'LineStyle','none',...
  'Marker','.',...
  'MarkerSize',20);
set(loglog1(6),...
  'Color',[0.1686 0.5059 0.3373],...
  'LineStyle','--');
set(loglog1(7),...
  'Color',[0.04314 0.5176 0.7804],...
  'LineStyle','--');
set(loglog1(8),...
  'Color',[1 0 1],...
  'LineStyle','--');
set(loglog1(9),...
  'Color',[1 0 0],...
  'LineStyle','-.');
set(loglog1(10),...
  'Color',[0 0 1],...
  'LineStyle','-.');

annotation1 = annotation(...
  gcf,'textbox',...
  'Position',[0.316 0.7937 0.2077 0.1222],...
  'EdgeColor','none',...
  'Color',[0 0 1],...
  'String',{'96 MeV ^1^0^0Zr','Fission Fragment'},...
  'FitHeightToText','on');
annotation2 = annotation(...
  gcf,'textbox',...
```

```
  'Position',[0.5593 0.7213 0.2077 0.1222],...
  'EdgeColor','none',...
  'Color',[1 0 0],...
  'String',{'72 MeV ^1^3^4Te','Fission Fragment'},...
  'FitHeightToText','on');
 annotation3 = annotation(...
  gcf,'textbox',...
  'Position',[[0.5096 0.6298 0.2077 0.0746]],...
  'EdgeColor','none',...
  'Color',[1 0 1],...
  'String',{'Alpha'},...
  'FitHeightToText','on');
 annotation4 = annotation(...
  gcf,'textbox',...
  'Position',[0.5396 0.3974 0.2077 0.0746],...
  'EdgeColor','none',...
  'Color',[0.04314 0.5176 0.7804],...
  'String',{'Electron'},...
  'FitHeightToText','on');
 annotation5 = annotation(...
  gcf,'textbox',...
  'Position',[0.604 0.2903 0.2077 0.0746],...
  'EdgeColor','none',...
  'Color',[0.1686 0.5059 0.3373],...
  'String',{'Gamma-ray'},...
  'FitHeightToText','on');
title('Energy Deposition Comparison for Different Path Lengths');
xlabel('Path Length (mm)');
ylabel('Energy Deposition (eV)');
set(gca,'XTick',[0.25;0.5;1;5;10;50;100;250]);
axis([0.15 400 2E2 3E8]);

RIG = FIC./GC;
RIE = FIC./EC;
RIA = FIC./AC;
RBG = FBC./GC;
RBE = FBC./EC;
RBA = FBC./AC;
figure(2);
loglog1 = loglog(C,RIG,C,RIE,C,RIA,C,RBG,C,RBE,C,RBA);
set(gca,'XTick',[0.25;0.5;1;5;10;50;100;250]);
legend('I:G','I:E','I:A','B:G','B:E','B:A');
[X,I] = max(RIG);
RIGmax = [C(I) X]
[X,I] = max(RIE);
RIEmax = [C(I) X]
[X,I] = max(RIA);
RIAmax = [C(I) X]
[X,I] = max(RBG);
RBGmax = [C(I) X]
[X,I] = max(RBE);
```

```
RBEmax = [C(I) X]
[X,I] = max(RBA);
RBAmax = [C(I) X]

Gx = pchip(G(:,1),G(:,2),(sqrt(0.5^2+1^2)))
GR = max(pchip(G(:,1),FI(:,1),(sqrt(0.5^2+1^2))))/Gx
Ex = pchip(E(:,1),E(:,2),(sqrt(0.5^2+1^2)))
ER = max(pchip(G(:,1),FI(:,1),(sqrt(0.5^2+1^2))))/Ex
Ax = pchip(A(:,1),A(:,2),(sqrt(0.5^2+1^2)))
AR = max(pchip(G(:,1),FI(:,1),(sqrt(0.5^2+1^2))))/Ax
```

End EDepCompAlpha.m

B.4 Pulse Shape

B.4.1 PulsePlot.m

This program is used to generate an estimated pulse shape plot and calculates the current output for different neutron reactive coating thicknesses and bias settings.

Begin PulsePlot.m

```
clear all; close all; clc;

% Settings
DetCoat = 0.5; % um of uranium
DetBias = 400; % V

% Data
q = 1.60217653e-19; % Coulombs (electron charge)
Bias = [200 300 400]; % V
IonDriftTime = [7479 5127 3884]*1e-9; % sec
ElecDriftTime = [66.1 45.8 34.0]*1e-9; % sec

Coating = [0.5 1 2];
IndEnergyMean = [1.852 1.683 1.376]*1e6; % eV
IndTimeMean = [61.9 64.2 67.9]*1e-12; % sec
IndEnergySlow = [0.260 0.260 0.260]*1e6; % eV
IndTimeSlow = [691 709 752]*1e-12; % sec
IndEnergyHigh = [6.052 5.991 6.000]*1e6; % eV
IndTimeHigh = [123 123 124]*1e-12; % sec
IndEnergyFast = [0.260 0.260 0.260]*1e6; % eV
IndTimeFast = [3.52 3.52 3.52]*1e-12; % sec

% CurrentT = [1.0025 1.1065 1.3232]*1e-10; % sec
% CurrentE = [1.9141 1.7571 1.4747]*1e6; % eV

W = 26.4; % eV/ion-electron pair for argon
```

```
c = eq(Coating,DetCoat);
b = eq(Bias,DetBias);
ElecIonPairs = [IndEnergyMean([c]) IndEnergySlow([c]) IndEnergyHigh([c])]/W; % ion−electron pairs
IndTimes = [IndTimeMean([c]) IndTimeSlow([c]) IndTimeHigh([c])]; %sec
IonCurrent = q∗ElecIonPairs./IonDriftTime;
ElecCurrent = q∗ElecIonPairs./ElecDriftTime;

for i = 1:size(ElecCurrent,2)
    figure(i);
    ElecData = [logspace(log10(IndTimes(1,i)/100),log10(IndTimes(1,i)),100) logspace(log10( ...
        IndTimes(1,i)),log10(ElecDriftTime(1,[b])),500)];
    ElecData(2,:) = [ElecData(1,1:100)∗ElecCurrent(1,i)/IndTimes(1,i) (ones(1,500)∗ElecCurrent(1,i) ...
        − ElecData(1,101:600)∗ElecCurrent(1,i)/ElecDriftTime(1,[b]))];
    IonData = [logspace(log10(IndTimes(1,i)/100),log10(IndTimes(1,i)),100) logspace(log10(IndTimes ...
        (1,i)),log10(ElecDriftTime(1,[b])),500) logspace(log10(ElecDriftTime(1,[b])),log10( ...
        IonDriftTime(1,[b])),500)];
    IonData(2,:) = [IonData(1,1:100)∗IonCurrent(1,i)/IndTimes(1,i) (ones(1,1000)∗IonCurrent(1,i) − ...
        IonData(1,101:1100)∗IonCurrent(1,i)/IonDriftTime(1,[b]))];
    TotData{i} = IonData; TotData{i}(2,1:600) = ElecData(2,1:600)+TotData{i}(2,1:600);
    plot(ElecData(1,:),ElecData(2,:),'.r',IonData(1,:),IonData(2,:),'.b',TotData{i}(1,:),TotData{i ...
        }(2,:),'.g');
end

figure(i+1);
% subplot(5,5,1:25);
clrs = colormap(lines(i));
maxC = 0;
maxT = 0;
i = 1;
for i = 1:size(TotData,2)
    plot(TotData{i}(1,:)∗1e6,TotData{i}(2,:)∗1e9,'-','Color',clrs(i,:));
    maxT = max([maxT TotData{i}(1,:)∗1e6]);
    maxC = max([maxC TotData{i}(2,:)∗1e9]);
    hold on;
end
hold off;
% axis([0 maxT 0 maxC]);
xlabel('Time (\mus)');
ylabel('Current (nA)');
title({'Pulse Shapes for MPFDs';[num2str(DetCoat),' \mum Thick Uranium ...
    Coating, ',num2str(DetBias),' V Bias']});
set(gcf,'Position',[560 504 877 444]);
setplot(12,12,11);

figure(i+2);
% subplot(5,5,[8:10,13:15,18:20]);
for i = 1:size(TotData,2)
    plot(TotData{i}(1,:)∗1e6,TotData{i}(2,:)∗1e9,'-','Color',clrs(i,:));
    hold on;
end
hold off;
```

```
axis([0 ElecDriftTime(1,[b])*1.25e6 0 maxC]);
legend('Mean','Slow','High');
% xlabel('Time (\mus)');
% ylabel('Current (nA)');
set(gcf,'Position',[751 569 633 300]);
setplot(11,11,11);

figure(i+3);
IndTime = [IndTimeMean,IndTimeSlow,IndTimeHigh,IndTimeFast];
IndFreq = 0.35./(0.8*IndTime);
ElecFreq = 0.35./(0.8*ElecDriftTime);
IonFreq = 0.35./(0.8*IonDriftTime);
loglog(IndTime,IndFreq,'d',ElecDriftTime,ElecFreq,'o',IonDriftTime,IonFreq,'+')
xlabel('Time (sec)')
ylabel('Freq (Hz)')
title('Induction and Drift Associated Frequency')
legend('Induction','Electron Drift','Ion Drift')
set(gcf,'Position',[560 504 877 444]);
setplot(12,12,11);

NatUFR = 15951178491.6912/250000*19.043*pi()*(1^2)/4*DetCoat*1e-6; % fissions/sec per watt
EnrUFR = 1879893221956.69/250000*19.050*pi()*(1^2)/4*DetCoat*1e-6; % fissions/sec per watt
ThFR = 218774435.145849/250000*11.724*pi()*(1^2)/4*DetCoat*1e-6; % fissions/sec per watt

% CurrentT = [1.0030 1.1075 1.3217]*1e-10; % sec
CurrentE = [1.8598 1.6906 1.3845]*1e6; % eV

Pwr = [1 250e3 1.25e6 250e6 3.2e9] % Reactor Power - watts
CurrentNatU = (q*(CurrentE(1,[c])/W)+q*(CurrentE(1,[c])/W))*Pwr*NatUFR
CurrentEnrU = (q*(CurrentE(1,[c])/W)+q*(CurrentE(1,[c])/W))*Pwr*EnrUFR
CurrentTh = (q*(CurrentE(1,[c])/W)+q*(CurrentE(1,[c])/W))*Pwr*ThFR
```

End PulsePlot.m

B.5 ASCEFI

B.5.1 user.h

The user.h program is used by the Ez-Ware software for customization of the ASCEFI controller.

Begin user.h

```
////////////// Uranium and Thorium Electroplating System \\\\\\\\\\\\\\
// Code for MP200 controller unit
// Developed by: Martin F. Ohmes
// at: Semiconductor Materials and Radiological Technologies
// (S.M.A.R.T.) Laboratory, Dept. of Mechanical and
```

```
// Nuclear Engineering, Kansas State University
// when: December 2003 – September 2007
////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\


// Function Declarations Begin
#define MAXV 2000 // < 5 VDC
#define MINV 0 //
#define CRm 1145.4371 //1143.9541
#define CRb 1.1798737 //0.25855412
#define Fm 25385.525 //25352.584
#define Fb 23.084053 //1.9887108
#define TRate 39
/******************** 64k Variable Definitions ********************/
unsigned int ui0,ui1;
unsigned int rbui;
unsigned int crcui, lastbit, crc, thisbyte, shift, ser1_finished;
unsigned int i_wait;
unsigned int twv_stat;
unsigned int twv_now;
unsigned int motnlnow;
unsigned int motn_dir;
unsigned int peri_pwr;
unsigned int peri_now;
unsigned int motn_loc;
unsigned int motn_sol;
unsigned int motnldir;
unsigned int version;
unsigned int startup;
unsigned int versnow;
/******************** +– 32K Variable Definitions ********************/
/******************** 4 Gig Variable Definitions ********************/
unsigned long int ser_con;
unsigned long int ser_tot;
unsigned long int mixtime;
unsigned long int ser_tmp;
/******************** 2 Gig Variable Definitions ********************/
long int k1t,k1h,k1m,k1s,k1l; // time vars
long int timrmn;
/******************** Float Gig Variable Definitions ********************/
float fl1,fl2,fl3,fl4;
float wb_temp;
float wb_set;
float wb_cn9_2;
float wb_cn9_3;
/******************** Character Array Variable Definitions ********************/
char sts[2] = "a";
unsigned char mess[20], tmess[20], reply[512];
char tempstr[9];
char tempstr2[9];
char serial_data[25];
```

```
char tprogram[72];
char uprogram[72];
char measpgm[72] = " ...
    40.005301003011111020.00.0244013.000017.000021.000026.000030.000000.000 ...
    ";
char pgm17[8] = "017.000";
char pgm30[8] = "030.000";
char SFntName[4]; // 3 character short name for FRONT material
char LFntName[8]; // 7 character long name for FRONT material
char SBckName[4]; // 3 character short name for BACK material
char LBckName[8]; // 7 character long name for BACK material
// 0 1 2 3 4 5 6 7
// 01234567890123456789012345678901234567890123456789012345678901234567890123456789
char runp_scr[81] = "*** Run Program ***F1-Front F5-ExitF2-Back F3-> Frt/ ...
    Bck <-F4 ";
char setp_scr[81] = "** Setup Program **F1-Front F2-Back F5-Exit ";
char stght_scr[81] = "F1-Set WB Temp F2-Turn ON Heaters F3-Turn OFF Heaters ...
     ";
/******************** String Variable Definitions ********************/
/******************** Character Variable Definitions ********************/
/******************** Decimal Equivalent Variable Definitions ********************/
unsigned char uc0,uc1;
unsigned char ucx,ucxs,ucy,ucys,gik0,gik1;
unsigned char stuc;
unsigned char rmuc1,rmuc2;
unsigned char smucp,smuce,smuc1,smuc2;
unsigned char rsuc0,rsuc1;
unsigned char svuc2,svuc3;
unsigned char rp0;

/******************** Function Declarations ********************/
void menu();
void power();
void setup();
void init();
void init_all17();
void init_all30();
void init_meas();
void scr_bath_names();
void runp();
void setp();
void syss();
void lock();
void bath();
void peri();
void motn();
void possetup();
unsigned char getikey(char scr_menu[81]);
unsigned char scankeyL();
void run_bath();
void run_bath2();
```

```
void run_bath_set();
void run_stg_off();
void run_stg_temp();
void stg_security();
void send_mess(unsigned char im, unsigned char ir);
void crc_calc(unsigned int crcloc);
void run_peri();
void clrperi();
void run_loc();
void run_motn();
void run_sol();
void run_ser();
void set_prog(unsigned char spuc1);
void set_val(unsigned char starti, unsigned char endi, unsigned char loci, unsigned char svuc1);
void runpver(unsigned char pver);
void datapver(unsigned char pver);
void runningp(unsigned char rprg);
void settmp(unsigned char prog, unsigned char starti, unsigned char endi);
void temperr();
void plating(unsigned char pos, unsigned char tprg);
unsigned char codescr(unsigned char cs);
void timecalc();
void printscr(char *scrstr);
void bath_names();

void init_ser1();
interrupt (unsigned char) ser1_int();

////////////////////// Primary Screens Begin \\\\\\\\\\\\\\\\\\\\\\

/* Main Menu */
void menu() {
  while (1) {
    printscr(main_scr);
    uc0 = getikey(main_scr);
    if (uc0 == F1) {
      setp(); // F1 - Setup Programs
    } else if (uc0 == F2) {
      runp(); // F2 - Run Program
    } else if (uc0 == F3) {
      setup(); // F3 - Setup - Initialize System (Reset)
    } else if (uc0 == F5) {
      lock(); // F5 - Lock System
    } else if (uc0 == F6) {
      motn(); // F6 - Move - Motion Control
    } else if (uc0 == F7) {
      peri(); // F7 - Pump - Peristalic Pump
    } else if (uc0 == F8) {
      bath(); // F8 - Water Bath
    }
  }
```

```
}

/* Power Motherboard */
void power() {
  clrscrn();
  gbl_port = 3; gbl_col = 3; printfn("ASCEFI Starting Please wait");
  gbl_col = 42;
  if (versnow != version) {
    printfn("Version %u Update",version);
    init();
  } else {
    startup += 1;
    printfn(" Version %u.%u",version,startup);
  }
  //gbl_col = 61; printfn("Zeroing Outputs ");
  wranout(MINV,PLATEV); // Set plating voltage to 0
  wrdigout(0,POWER);
  wrdigout(0,SOLENOID);
  wrdigout(0,STGHEAT); // Turn off Stage Heaters
  wrdigout(0,PERIPWR);
  wrdigout(0,PERIDIR);
  wrdigout(0,TWV_ON);
  err_str = wr_stprs_out(1,STRTELEC,HIGH);
  err_str = wr_stprs_out(1,WB_ON,LOW);
  err_str = wr_stprs_out(1,LOC1,LOW);
  err_str = wr_stprs_out(1,LOC2,LOW);
  err_str = wr_stprs_out(1,LOC3,LOW);
  err_str = wr_stprs_out(1,SER_DATA,LOW);
  err_str = wr_stprs_out(1,SER_CLK,HIGH);
  err_str = wr_stprs_out(1,U_TH,HIGH);
  //gbl_col = 61; printfn("Power Motherboard");
  wrdigout(1,POWER); // Motherboard Power
  //inproc = 0;
  if (twv_now != 0 || peri_now != 0) {
    //gbl_col = 61; printfn("Clearing Fluid Lines");
    clrperi();
    //gbl_col = 61; printfn("3-Way Valves Closed ");
    twv_stat = 0; // 3-Way Valve & U_Th Select
    twv_now = 1;
    run_peri();
  } else {
    twv_stat = 0;
    peri_pwr = 0;
  }
  //gbl_col = 61; printfn("Moving platform ");
  motn_loc = 0;
  motn_sol = 1;
  run_motn();
  run_sol();
  //gbl_col = 61; printfn("Zeroing Serial ");
  ser_tot = 10;
```

```
    run_ser();
    err_str = wr_stprs_out(1,STRTELEC,LOW); // Inverted
    //gbl_col = 61; printfn("Updating Bath ");
    mess[0] = 0x01;
    run_bath_set();
    scr_bath_names();
    //gbl_col = 61; printfn("Reset Stage Heater");
    mess[0] = 0x02; run_stg_off();
    mess[0] = 0x03; run_stg_off();
    wrdigout(1,STGHEAT); // Turn off Stage Heaters
}

/* Initialize System (Reset) */
void setup() {
    while (1) {
        printscr(sys_scr);
        gbl_col = 73; printfn("%u.%u",version,startup);
        uc0 = getikey(sys_scr);
        if (uc0 == F1) { // F1 - Initialize Variables
            do {
                printscr(init_scr);
                uc1 = getikey(init_scr);
                if (uc1 == F1) {
                    if (codescr(2) == 10) {
                        init();
                        power();
                    }
                } else if (uc1 == F2) {
                    init_all17();
                    uc1 = F5;
                } else if (uc1 == F3) {
                    init_all30();
                    uc1 = F5;
                } else if (uc1 == F4) {
                    init_meas();
                    uc1 = F5;
                }
            } while (uc1 != F5);
        } else if (uc0 == F2) { // F2 - Set Fluid Names
            bath_names();
        } else if (uc0 == F3) { // Recirc fluids
            do {
                peri_pwr = (peri_pwr + 1) % 3;
                do {
                    twv_stat = (twv_stat + 1) % 3;
                    uc0 = 0;
                    do {
                        uc0++;
                            adelay(1000);
                        uc1 = scankeyL();
                    } while (uc1 != F5 && uc0 <= 120);
```

```
            } while (uc1 != F5);
          } while (uc1 != F5);
        } else if (uc0 == F4) {
          printscr(stght_scr);
          do {
            uc1 = scankeyL();
            if (uc1 == F1) { // Set WB Temp
              mess[0] = 0x02; run_stg_temp();
              mess[0] = 0x03; run_stg_temp();
            } else if (uc1 == F2) { // Turn ON Heaters
              wrdigout(1,STGHEAT); // Turn on Stage Heaters
            } else if (uc1 == F3) { // Turn OFF Heaters
              mess[0] = 0x02; run_stg_off();
              mess[0] = 0x03; run_stg_off();
              wrdigout(1,STGHEAT); // Turn off Stage Heaters
            }
            gbl_col = 77;
            if (STGHOT == OFF) {
              printfn("h");
            } else {
              printfn("H");
            }
            if (STGCOLD == OFF) {
              printfn("c");
            } else {
              printfn("C");
            }
          } while (uc1 != F5);

        } else if (uc0 == F5) { // F5 - Exit
          break;
        }
      }
    }

    void init_all17() {
      init_meas();
      // 00.0 = Water Bath Temperature (xx.x C)
      // 11 = Minimum Mixing Time (xx minutes)
      // 22 = System Drying Time after plating (xx minutes)
      // 33 = Minimum Bath Flowing Time (xx minutes)
      // 444 = Hold Time After Plating (xxx seconds)
      // 234567 = Plating Positions (posloc2 ... posloc7)
      // 66.6 = Target Current Density (xx.x mA/cm2)
      // 7.7777 = Surface Area (x.xxxx cm2)
      // xxx.xxx = Total Current (xxx.xxx A/cm2)
      // 0 1 2 3 4 5 6 7
      // 01234567890123456789012345678901234567890123456789012345678901234567890
      // 00.011223344423456766.67.7777222.222333.333444.444555.555666.666777.777
      ucys = 29;
      ucy = 0;
```

```
  while (ucys < 71) {
    uprogram[ucys] = pgm17[ucy];
    tprogram[ucys] = pgm17[ucy];
    ucy = (ucy + 1) % 7;
    ucys = ucys + 1;
  }
  //strcpy(uprogram ...
       ,"40.005301003011111020.00.0244017.000017.000017.000017.000017.000000.000");
  //strcpy(tprogram ...
       ,"40.005301003011111020.00.0244017.000017.000017.000017.000017.000000.000");
}

void init_all30() {
  init_meas();
  ucys = 29;
  ucy = 0;
  while (ucys < 71) {
    uprogram[ucys] = pgm30[ucy];
    tprogram[ucys] = pgm30[ucy];
    ucy = (ucy + 1) % 7;
    ucys = ucys + 1;
  }
  //strcpy(uprogram ...
       ,"40.005301003011111020.00.0244030.000030.000030.000030.000030.000000.000");
  //strcpy(tprogram ...
       ,"40.005301003011111020.00.0244030.000030.000030.000030.000030.000000.000");
}

void init_meas() {
  for (ucx = 0; ucx < 72; ucx++) {
    uprogram[ucx] = measpgm[ucx];
    tprogram[ucx] = measpgm[ucx];
  }
  //strcpy(uprogram ...
       ,"40.005301003011111020.00.0244013.000017.000021.000026.000030.000000.000");
  //strcpy(tprogram ...
       ,"40.005301003011111020.00.0244013.000017.000021.000026.000030.000000.000");
}

void init() {
  init_meas();
  strcpy(SFntName,"Nat"); // 3 character short name for FRONT material
  strcpy(LFntName,"Uranium"); // 7 character long name for FRONT material
  strcpy(SBckName,"Enr"); // 3 character short name for BACK material
  strcpy(LBckName,"EnrU235"); // 7 character long name for BACK material
  startup = 1;
  versnow = version;
  wb_set = 40.0;
}

void scr_bath_names() {
```

```
  for (uc1 = 0; uc1 < 7; uc1++) {
    runp_scr[23+uc1] = LFntName[uc1];
    setp_scr[23+uc1] = LFntName[uc1];
    runp_scr[43+uc1] = LBckName[uc1];
    setp_scr[43+uc1] = LBckName[uc1];
    if (uc1 < 3) {
      runp_scr[66+uc1] = SFntName[uc1];
      runp_scr[70+uc1] = SBckName[uc1];
    }
  }
}

void bath_names() {
  unsigned char i1,i2;
  // strcpy(SFntName[4],"Nat"); // 3 character short name for FRONT material
  // strcpy(LFntName[8],"Uranium"); // 7 character long name for FRONT material
  // strcpy(SBckName[4],"Enr"); // 3 character short name for BACK material
  // strcpy(LBckName[8],"EnrU235"); // 7 character long name for BACK material
  i1 = 1; // 1 - Front, 2 - Back
  i2 = 0;
  do {
  // 01234567890123
  //char names_scr[81] = " [Uranium] [Nat] Front Bath F1-Save1-Up 4-Down 3-Next2-Upper 5- ...
       Lower ";
    if (i1 == 1) {
      if (i2 > 6) {
        sts[0] = SFntName[i2-11];
      } else {
        sts[0] = LFntName[i2];
      }
    } else {
      if (i2 > 6) {
        sts[0] = SBckName[i2-11];
      } else {
        sts[0] = LBckName[i2];
      }
    }
    gbl_col = 1;
    if (i1 == 1) {
      printfn(" [%s] [%s] Front Bath F1-Save1-Up 4-Down 3-Next",LFntName, ...
          SFntName);
    } else {
      printfn(" [%s] [%s] Back Bath F1-Save1-Up 4-Down 3-Next",LBckName, ...
          SBckName);
    }
    if (sts[0] >= 65 && sts[0] <= 90) {
      printfn("5-Lower");
    } else if (sts[0] >= 97 && sts[0] <= 122) {
      printfn("2-Upper");
    } else {
      printfn("  ");
```

```
}
printfn("  6-Back");
gbl_col = i2+4; printfn("%c",sts[0]);
curs_on(i2+4);
do {
   uc1 = scankeyL();
} while (uc1 == 255);
curs_off();
if (uc1 == F1 || uc1 == ENTER) {
   if (i1 == 1) {
      i1 = 2;
      i2 = 0;
   } else {
      uc1 = F5;
   }
} else if (uc1 == 3) {// Next
   if (i2 < 13) {i2++;}
   if (i2 == 7) {i2 = 11;}
} else if (uc1 == 6) {// Back
   if (i2 > 0) {i2--;}
   if (i2 == 10) {i2 = 6;}
} else if (uc1 != F5) {
   if (uc1 == 1) { // Up
      sts[0] = sts[0] + 1;
   } else if (uc1 == 4) { // Down
      sts[0] = sts[0] - 1;
   } else if (uc1 == 2) { // Upper
      if (sts[0] >= 97 && sts[0] <= 122) {
         sts[0] = sts[0] - 32;
      }
   } else if (uc1 == 5) { // Lower
      if (sts[0] >= 65 && sts[0] <= 90) {
         sts[0] = sts[0] + 32;
      }
   } else if (uc1 == CLEAR) { // Space
      sts[0] = 32;
   }
   if (sts[0] == 33) {sts[0] = 48;}
   if (sts[0] == 58) {sts[0] = 97;}
   if (sts[0] == 123) {sts[0] = 65;}
   if (sts[0] == 91) {sts[0] = 32;}
   if (sts[0] == 31) {sts[0] = 90;}
   if (sts[0] == 64) {sts[0] = 122;}
   if (sts[0] == 96) {sts[0] = 57;}
   if (sts[0] == 46) {sts[0] = 32;}
   if (i1 == 1) {
      if (i2 > 6) {
         SFntName[i2-11] = sts[0];
      } else {
         LFntName[i2] = sts[0];
      }
```

```
        } else {
          if (i2 > 6) {
            SBckName[i2−11] = sts[0];
          } else {
            LBckName[i2] = sts[0];
          }
        }
      }
    } while (uc1 != F5);
    scr_bath_names();
}

/* Run Program − Main Menu */
void runp() {
    while (1) {
      printscr(runp_scr);
      uc0 = getikey(main_scr);
      if (uc0 == F1) {
        runpver(1); // F1 − Front [Uranium] Program
        break;
      } else if (uc0 == F2) {
        runpver(2); // F2 − Back [Thorium] Program
        break;
      } else if (uc0 == F3) {
        runpver(3); // F3 − Front/Back [Uranium/Thorium] Program
        break;
      } else if (uc0 == F4) {
        runpver(4); // F4 − Back/Front [Thorium/Uranium] Program
        break;
      } else if (uc0 == F5) {
        break; // F5 − Exit
      }
    }
}

unsigned char codescr(unsigned char cs) {
    gbl_port = 3;
    if (cs == 4) {
      gbl_col = 61; printfn("Authorize:");
    } else {
      printscr(lock_scr);
      gbl_col = 43;
    }
    if (cs == 2) { // Run Program
      printfn("Authorize Start");
    } else if (cs == 3) { // Set Program & Reset Variables
      printfn("Authorize Change");
    }
    ui1 = 0;
    for (uc0 = 72; uc0 <= 78; uc0 = uc0 + 2) {
      do {ui0 = getikey(lock_scr);} while (ui0 == 255);
```

```
      gbl_col = uc0; printfn("*");
      ui1 = ui1*10 + ui0;
    }
    if (ui1 == 6733 && (cs == 1 || cs == 4)) { // Lock Screen
      cs = 10;
    } else if (ui1 == 1184 && cs == 2) { // Run Program
      cs = 10;
    } else if (ui1 == 9778 && cs == 3) { // Set Program
      cs = 10;
    }
    return(cs);
}

/* Authorize Run Program */
void runpver(unsigned char pver) {
    if (codescr(2) == 10) {
      clrscrn();
      fl1 = 0;
      if (motnlnow != 7) {
        fl1 = (6-motnlnow)*19+16;
        if (motnlnow == 0) {
          fl1 += 127;
        }
      }
      settmp(1,4,5); // Get program data for minimum solution mixing time
      fl1 = fl1 + (atof(tempstr))*60;
      settmp(1,6,7); // Get program data for system drying time
      fl1 = fl1 + (atof(tempstr))*60;
      settmp(1,19,22); // Get program data for current density
      wb_cn9_2 = atof(tempstr);
      settmp(1,23,28); // Get program data for surface area
      wb_cn9_3 = atof(tempstr); // surface area
      if (pver != 2) { // Uranium used
        datapver(1);
      }
      if (pver != 1) { // Thorium used
        datapver(2);
      }
      k1t = fl1 + 300;
      timecalc();
      gbl_col = 1; printfn("**Program Running**F8-Continue Program F5-Exit  ...
          Estimate: %4li:%02li:%02li",k1h,k1m,k1s);
      do { // Load Samples
        uc1 = scankeyL();
        } while (uc1 != F8 && uc1 != F5);
      if (uc1 == F8) {
        clrlne(3,1);
        settmp(1,4,5); // Get program data for minimum solution mixing time
        mixtime = atol(tempstr);
        mixtime *= 60;
        if (pver == 1 || pver == 3) {
```

```
            runningp(1);
          } else {
            runningp(2);
          }
          mixtime = 0;
          if (pver == 3) {runningp(2);}
          if (pver == 4) {runningp(1);}
          mess[0] = 0x02; run_stg_off(); // Turn off Front Stage Heater
          mess[0] = 0x03; run_stg_off(); // Turn off Back Stage Heater
          wrdigout(1,STGHEAT); // Turn off Stage Heaters
          // 12345678901234567890123456789012345678901234567890123456789012345678901234567890
          gbl_col = 21; printfn(" Plating Complete Shutting Down ");
          clrperi();
          motn_loc = 7;
          run_motn(); // Move to hold
          settmp(1,6,7); // Get program data for minimum drying time after plating
          mixtime = atol(tempstr);
          mixtime *= 60;
          k1t = mixtime;
          at_cdown = sec_1;
          do {
            timecalc();
            gbl_col = 61; printfn("System Drying:%3li:%02li",k1m,k1s);
            while (k1t == mixtime - (sec_1 - at_cdown)) {
              if (sec_1 >= 2591999) {mixtime = mixtime - (2591999 - at_cdown);}
            }
            k1t = mixtime - (sec_1 - at_cdown);
          } while (k1t > 0);// 12345678901234567890123456789012345678901234567890
          gbl_col = 41; printfn(" Waiting on Stage to Cool ");
          do {
          } while (STGCOLD == ON);
          gbl_col = 41; printfn(" Moving to LOAD ");
          motn_loc = 0;
          run_motn();
          lock();
        }
      }
    }

    void datapver(unsigned char pver) {
      settmp(pver,8,9); // Get program data for minimum flow time before plating
      fl1 = fl1 + (atof(tempstr))*60;
      settmp(pver,10,12); // Get program data for minimum drying time after plating
      fl3 = atof(tempstr);
      uc0 = 7;
      for (rp0 = 2; rp0 <= 6; rp0 = rp0 + 1) {
        settmp(pver,11+rp0,11+rp0); // Get Locations
        if (atoi(tempstr) == 1) {
          uc1 = rp0 - 2;
          settmp(pver,29+7*uc1,35+7*uc1); // Get program data for total current
          fl4 = uc0;
```

```
        fl4 -= rp0;
        if (fl4 < 0) { fl4 = 0 - fl4;}
        fl1 = fl1 + fl4*19 + fl3 + atof(tempstr)*1000/wb_cn9_2 + wb_cn9_2*wb_cn9_3*TRate + 51 + 1.9* ...
              atof(tempstr)*1000*(wb_cn9_2*wb_cn9_3*Fm - Fb)/wb_cn9_2/16777200; // 25+10+16
        uc0 = rp0;
      }
    }
    fl1 = fl1 + (7-uc0)*19 + 16;
}

/* Temperature Error */
void temperr() {
    gbl_col = 41; printfn(" *** ERROR *** ");
    clrperi();
    while (1) {
    }
}

void clrperi() {
    peri_pwr = 2;
    run_peri(); // Drain lines
    peri_now = 1;
    twv_stat = 1;
    run_peri(); // Open U 3-way valve
    peri_now = 1;
    twv_stat = 2;
    run_peri(); // Open Th 3-way valve
    twv_stat = 0;
    run_peri(); // Close 3-way valve
    peri_pwr = 0;
    run_peri(); // Turn off peristalic pump
}

void timecalc() {
    k1h = k1t/3600;
    k1m = (k1t-k1h*3600)/60;
    k1s = k1t-k1h*3600-k1m*60;
}

/* Run through the plating program */
void runningp(unsigned char rprg) {
    gbl_col = 21; printfn(" Setting up for Run ");
    clrlne(4,1);
    //inproc = 0;
    twv_stat = 0;
    peri_pwr = 0;
    run_peri(); // Turn off Peristalic pump
    settmp(rprg,0,3); // Get program data for temperature
    wb_set = atof(tempstr);
    mess[0] = 0x01;
    run_bath_set();
```

```
wrdigout(1,STGHEAT); // Turn on Stage Heaters
mess[0] = 0x02; run_stg_temp(); // Set Front Stage Heater Temp
mess[0] = 0x03; run_stg_temp(); // Set Back Stage Heater Temp
motn_loc = 7;
run_motn();
gbl_col = 21; printfn("Target Temp: %4.1lf C",wb_set);

k1t = mixtime;
rp0 = 0;
uc0 = 255;
do {
  mess[0] = 0x01; run_bath();
  gbl_col = 41; printfn("Temp Now: %4.1f C",wb_temp);
  if (((wb_temp < wb_set - 0.2) && rp0 == 0) || (STGHOT == OFF)) { // Mixing Temp not reached
    peri_pwr = 0;
    adelay(5000);
  } else if (peri_pwr == 0) { // Mixing Temp reached, start timer
    adelay(1000);
    peri_pwr = 1;
    at_cdown = sec_1;
    rp0 = 1;
  } else { // Keep running until timer runs out
    timecalc();
    while (k1t == mixtime - (sec_1 - at_cdown)) {
      if (sec_1 >= 2591999) {mixtime = mixtime - (2591999 - at_cdown);}
    }
    gbl_col = 61;
    if (mixtime - (sec_1 - at_cdown) >= 0) {
      k1t = mixtime - (sec_1 - at_cdown);
      printfn("Mix Left: %4li:%02li",k1m,k1s);
    } else {
      k1t = 0;
      printfn(" Waiting on Temp ");
    }
    if (sec_1 - at_cdown - mixtime > 1800) {
      temperr();
    }
  }
  run_peri();
  uc0 = scankeyL();
} while ((k1t > 0 || wb_temp < wb_set - 0.2 || wb_temp > wb_set + 0.2 || STGHOT == OFF) && uc0  ...
      != F5);

settmp(rprg,8,9); // Get program data for minimum flow time before plating
mixtime = atol(tempstr);
mixtime *= 60;
at_cdown = sec_1; // Reset Timer
twv_stat = rprg;
run_peri(); // Start Flow to Plating Bath
k1t = mixtime;
do {
```

```
      mess[0] = 0x01; run_bath();
      gbl_col = 41; printfn("Temp Now: %4.1f C",wb_temp);
      timecalc();
      while (k1t == mixtime -(sec_1 - at_cdown)) {
        if (sec_1 >= 2591999) {mixtime = mixtime - (2591999 - at_cdown);}
      }
      gbl_col = 61;
      if (mixtime - (sec_1 - at_cdown) >= 0) {
        k1t = mixtime - (sec_1 - at_cdown);
        printfn("Bath Flow: %4li:%02li",k1m,k1s);
      } else {
        k1t = 0;
        printfn(" Waiting on Temp ");
      }
      if (sec_1 - at_cdown - mixtime > 1800) {
        temperr();
      }
      uc0 = scankeyL();
    } while ((k1t > 0 || wb_temp < wb_set - 0.2 || wb_temp > wb_set + 0.2 || STGHOT == OFF) && uc0  ...
          != F5); // Run until minimum flow time in bath reached
    clrlne(2,1);
    clrlne(3,1);
    clrlne(4,1);
    settmp(rprg,10,12); // Get program data for minimum drying time after plating
    mixtime = 1;
    mixtime *= atol(tempstr);
    settmp(rprg,19,22); // Get program data for current density
    fl1 = atof(tempstr);
    settmp(rprg,23,28); // Get program data for surface area
    fl2 = atof(tempstr); // surface area
    gbl_col = 21; if (rprg == 1) {printfn(" %s Position ",LFntName);} else {printfn(" %s  ...
        Position ",LBckName);}
    for (rp0 = 2; rp0 <= 7; rp0 = rp0 + 1) {
      settmp(rprg,11+rp0,11+rp0); // Get Locations
      if (atoi(tempstr) == 1) {
        motn_loc = rp0;
        gbl_col = 39; printfn("%u",motn_loc);
        motn_loc--;
        plating(rp0,rprg);
      }
    }
}

void plating(unsigned char pos, unsigned char tprg) {
    wrdigout(1,STGHEAT); // Turn on Stage Heaters
    mess[0] = 0x02; run_stg_temp(); // Set Front Stage Heater Temp
    mess[0] = 0x03; run_stg_temp(); // Set Back Stage Heater Temp
    at_cdown = sec_1; // Reset Timer
    uc0 = 255;
    while ((wb_temp < wb_set - 0.2 || wb_temp > wb_set + 0.2 || STGHOT == OFF) && uc0 != F5) {
      uc0 = scankeyL();
```

```
  if (sec_1 >= 2591999) {at_cdown = sec_1;}
  if (sec_1 − at_cdown > 1800) {
    temperr();
  }
  gbl_col = 61; printfn(" Waiting on Temp ");
  adelay(1000);
}
gbl_col = 61; printfn("Preparing for EPlate");
twv_stat = 0;
run_peri(); // Close 3−Way Valves
wranout(MINV,PLATEV); // Set plating voltage to 0
run_motn(); // Move to proper position
motn_sol = 0; run_sol(); // Lower the platform
twv_stat = tprg;
run_peri(); // Start Flow to Plating Bath
adelay(1000);
pos = pos − 2;
settmp(tprg,29+7∗pos,31+7∗pos); // Get program data for total current
ser_tmp = 1000UL;
ser_tmp ∗=(atof(tempstr));
settmp(tprg,33+7∗pos,35+7∗pos); // Get program data for total current
ser_tmp += atof(tempstr);
fl3 = fl1∗fl2∗Fm − Fb;
ser_tmp = ser_tmp∗fl3/fl1;
gbl_col = 61; printfn("Rate: 0.00 mA/cm2");
ui0 = fl1∗fl2∗CRm − CRb;
if (fl1 − (ui0−CRb)/(CRm∗fl2) > (ui0+1−CRb)/(CRm∗fl2) − fl1) {
  ui0 = ui0 + 1;
}
if (ui0 > MAXV) {
  ui0 = MAXV;
}
ui1 = 0;
uc0 = 255;
k11 = 0;
adelay(8000);
while (ser_tmp > 0 && uc0 != F5) {
  if (ser_tmp > 16777200) {
    if (ser_tmp < 33554400) {
      ser_tot = ser_tmp/2;
    } else {
      ser_tot = 16777200;
    }
  } else {
    ser_tot = ser_tmp;
  }
  timrmn = ser_tmp/fl3 + fl1∗fl2∗TRate + 1.9∗ser_tmp/16777200;
  at_cdown = sec_1;
  ser_tmp = ser_tmp − ser_tot;
  run_ser();
  do {
```

```
        if (sec_1 >= 2591999) {timrmn = timrmn - (2591999 - at_cdown);}
        k1t = timrmn - (sec_1 - at_cdown);
        k1t = labs(k1t);
        if (ui1 < ui0) {
          ui1 = ui1 + 1;
          wranout(ui1,PLATEV);
          fl4 = (ui1-CRb)/(CRm*fl2);
          gbl_col = 61; printfn("Rate: %6.2f mA/cm2",fl4);
        }

        if (k1t != k1l) {
          timecalc();
          gbl_col = 41; printfn("Estimate: %4li:%02li:%02li",k1h,k1m,k1s);
          k1l = k1t;
        }
        uc0 = scankeyL();
      } while (ELECDONE != 0 && uc0 != F5);
      ui1 = 0;
      wranout(0,PLATEV);
    }
    clrlne(4,1);
    clrlne(3,1);
    twv_stat = 0;
    run_peri();
    motn_sol = 1; run_sol(); // Raise the platform
    adelay(1000);
    motn_sol = 0; run_sol(); // Lower the platform
    adelay(1000);
    motn_sol = 1; run_sol(); // Raise the platform
    adelay(1000);
    motn_sol = 0; run_sol(); // Lower the platform
    adelay(1000);
    motn_sol = 1; run_sol(); // Raise the platform
    at_cdown = sec_1; // Reset Timer
    k1t = mixtime;
    do {
      mess[0] = 0x01; run_bath();
      timecalc();
      gbl_col = 41; printfn("Temp Now: %5.2f CDrying: %3li:%02li",wb_temp,k1m,k1s);
      while (k1t == mixtime - (sec_1 - at_cdown)) {
        if (sec_1 >= 2591999) {mixtime = mixtime - (2591999 - at_cdown);}
      }
      k1t = mixtime - (sec_1 - at_cdown);
    } while (k1t > 0);
    clrlne(3,1); clrlne(4,1);
}

/* Setup Automated Programs - Main Menu */
void setp() {
  if(codescr(3) == 10){
    while (1) {
```

```
        printscr(setp_scr);
        uc0 = getikey(setp_scr);
        if (uc0 == 17 || uc0 == 18) { // F1 - Uranium F2 - Thorium
          set_prog(uc0);
        } else if (uc0 == F5) {
          break;
        }
      }
    }
}

void set_prog(unsigned char spuc1) {
  // 0 1 2 3 4
  // 012345678901234567890123456789012345
  // 00.011223344400000066.67.7777888.888
  // 0123 = Water Bath Temperature (xx.x C)
  // 45 = Minimum Mixing Time (xx minutes)
  // 67 = System Drying Time after plating (xx minutes)
  // 89 = Minimum Bath Flowing Time (xx minutes)
  // 012 = Hold Time After Plating (xxx seconds)
  // 345678 = Plating Positions (posloc2 ... posloc7)
  // 9012 = Target Current Density (xx.x mA/cm2)
  // 345678 = Surface Area (x.xxxx cm2)
  // 9012345 = Total Current (xxx.xxx A)
  clrscrn();
  gbl_col = 1;
  if (spuc1 == F1) {printfn("- %s Program -",LFntName);} else {printfn("- %s Program - ...
      ",LBckName);}
  gbl_col = 43; printfn("Water Bath Temp? __._ C");
  set_val(0,3,68,spuc1);
  gbl_col = 41; printfn("Minimum Mixing Time? __ minutes");
  set_val(4,5,66,spuc1);
  gbl_col = 41; printfn("Min Bath Flow Time? ");
  set_val(8,9,66,spuc1);
  gbl_col = 41; printfn("Hold Time Aftr Plate ___ seconds");
  set_val(10,12,65,spuc1);
  gbl_col = 41; printfn(" System Dry Time? __ minutes");
  set_val(6,7,66,spuc1);
  gbl_col = 43; printfn("Current Density? __._ mA/cm2");
  set_val(19,22,65,spuc1);
  gbl_col = 43; printfn(" Surface Area? _._____ cm2");
  set_val(23,28,66,spuc1);
  for (stuc = 0; stuc <= 28; stuc++) {
    if (stuc == 13) {
      stuc = 19;
    }
    if (spuc1 == 17) {
      tprogram[stuc] = uprogram[stuc];
    } else {
      uprogram[stuc] = tprogram[stuc];
    }
```

```
}
gbl_col = 21; printfn("Active Positions: ");
gbl_col = 41;
if (spuc1 == 17) {
  printfn(" 6 5 4 3 2 ");
} else {
  printfn(" 2 3 4 5 6 ");
}
clrlne(4,1);
for (uc0 = 13; uc0 <= 17; uc0++) {
  settmp(spuc1-16,uc0,uc0);
  if (spuc1 == 17) {
    gbl_col = 61 + (3*(18 - uc0));
  } else {
    gbl_col = 61 + (3*(uc0 - 12));
  }
  if (atoi(tempstr) == 1) {
    printfn("Y");
  } else {
    printfn("-");
  }
}
while (1) {
  do {
    uc0 = scankeyL();
  } while (uc0 == 255);
  if (uc0 <= 6 && uc0 >= 2) {
    uc0 += 11;
    settmp(spuc1-16,uc0,uc0);
    if (spuc1 == 17) {
      gbl_col = 61 + (3*(18 - uc0));
      if (atoi(tempstr) == 0) {
        uprogram[(uc0)] = '1';
        printfn("Y");
      } else {
        uprogram[(uc0)] = '0';
        printfn("-");
      }
    } else {
      gbl_col = 61 + (3*(uc0 - 12));
      if (atoi(tempstr) == 0) {
        tprogram[(uc0)] = '1';
        printfn("Y");
      } else {
        tprogram[(uc0)] = '0';
        printfn("-");
      }
    }
  } else if (uc0 == ENTER) {
    break;
  }
```

```
    }
    settmp(spuc1−16,19,22); // Get program data for current density
    fl1 = atof(tempstr);
    settmp(spuc1−16,23,28); // Get program data for surface area
    fl2 = atof(tempstr); // surface area
    for (uc0 = 0; uc0 < 5; uc0 = uc0 + 1) {
      settmp(spuc1−16,13+uc0,13+uc0); // Get Locations
      if (atoi(tempstr) == 1) {
        do {
          clrlne(2,1);
          gbl_col = 41; printfn("Total Charge DensityPOS%u: ___.___ C/cm2",uc0+2);
          set_val(29+7∗uc0,35+7∗uc0,68,spuc1);

          settmp(spuc1−16,29+7∗uc0,31+7∗uc0); // Get program data for total current
          ser_tmp = 1000UL;
          ser_tmp ∗= (atof(tempstr));
          settmp(spuc1−16,33+7∗uc0,35+7∗uc0); // Get program data for total current
          ser_tmp += atof(tempstr);
          k1t = ser_tmp/fl1 + fl1∗fl2∗TRate/2;
          timecalc();
          gbl_col = 21; printfn("Est Time: %3li:%02li:%02li Ok:ENTER No:CLEAR ",k1h ...
                ,k1m,k1s);
          do {
            uc1 = scankeyL();
          } while (uc1 == 255);
        } while (uc1 != ENTER);
      }
    }
}

void set_val(unsigned char starti, unsigned char endi, unsigned char loci, unsigned char svuc1) {
    settmp(svuc1 − 16,starti,endi);
    svuc3 = 0;
    do {
      gbl_col = loci; for (svuc2 = 0; svuc2 <= endi − starti; svuc2++) {printfn("%c",tempstr[svuc2]);}
      gbl_col = loci + svuc3; gbl_dec = 0; curs_on(loci + svuc3);
      do {
        svuc2 = scankeyL();
      } while (svuc2 > ENTER);
      curs_off();
      if (svuc2 == CLEAR) { // Clear Key Pressed
        tempstr[svuc3] = tempstr2[svuc3];
        if (svuc3 > 0) {svuc3−−;}
        if (tempstr[svuc3] == '.') {svuc3−−;}
      } else if (svuc3 <= endi − starti && svuc2 < 10) { // Number Entered
        ui2a_c31(svuc2,sts,1);
        tempstr[svuc3] = sts[0];
        svuc3 = svuc3 + 1;
        if (tempstr[svuc3] == '.') {svuc3++;}
      }
      if (svuc3 > (endi − starti)) {svuc3−−;}
```

```
  } while (svuc2 != ENTER); // Enter Key Pressed
  for (svuc3 = 0; svuc3 <= endi - starti; svuc3++) { // Store Result
    if (svuc1 == 17) {
      uprogram[(svuc3 + starti)] = tempstr[svuc3];
    } else {
      tprogram[(svuc3 + starti)] = tempstr[svuc3];
    }
  }
}

/* Code entering function for unlocking system */
void lock(){
  while (codescr(1) != 10) {}
  gik0 = 255;
}

////////////////////// Primary Screens End \\\\\\\\\\\\\\\\\\\\\\
////////////////////// Secondary Screens Begin \\\\\\\\\\\\\\\\\\\\\\

void printscr(char *scrstr) {
  gbl_port = 3; gbl_col = 1; printfn(scrstr);
}

/* Manual Control - Sub Menues */
/* Water Bath Manual Control */
void bath() {
  printscr(bathu_scr);
  mess[0] = 0x01;
  do {
    run_bath();
    printscr(bath_scr);
    gbl_col = 34; printfn("%6.2f",wb_temp);
    run_bath2();
    gbl_col = 55; printfn("%5.2f",wb_set);
    uc0 = getikey(bath_scr);
    if (uc0 == F1) {
      gbl_col = 55; printfn("___.___"); gbl_col = 55; gbl_dec = 2; curs_on(55); scanf("%f",&fl1);  ...
          curs_off();
      if (fl1 >= 0.0 && fl1 < 100.0) {
        printscr(bathu_scr);
        wb_set = fl1;
        run_bath_set();
      }
    }
  } while (uc0 != F5);
}

/* Peristalic Pump Manual Control */
void peri() {
  while (1) {
    printscr(peri_scr);
```

```
      gbl_col = 37;
      if (twv_stat == 1) {
        printfn("%s]",SFntName);
      } else if (twv_stat == 2) {
        printfn("%s]",SBckName);
      } else {
        printfn("OFF]");
      }
      gbl_col = 45;
      if (peri_pwr == 1) {
        printfn("FORWARD]");
      } else if (peri_pwr == 2) {
        printfn("REVERSE]");
      } else {
        printfn("OFF]  ");
      }
      uc0 = getikey(peri_scr);
      if (uc0 == F1) {
        twv_stat = (twv_stat + 1) % 3;
      } else if (uc0 == F2) {
        peri_pwr = (peri_pwr + 1) % 3;
      } else if (uc0 == F3) {
        clrscrn(); gbl_col = 5; printfn("Processing...");
        run_peri();
      } else if (uc0 == F5) {
        break;
      }
    }
    peri_pwr = peri_now;
    twv_stat = twv_now;
}

/* Motion Control Manual Control */
void motn() {
    motn_loc = motnlnow;
    while(1) {
      printscr(motn_scr);
      if (motn_loc == 0) {
        gbl_col = 21;
        printfn("[F1-LOAD]");
      } else if (motn_loc == 7) {
        gbl_col = 32;
        printfn("[F2-HOLD]");
      } else {
        gbl_col = 40+motn_loc*3;
        printfn("[%1u]",(motn_loc + 1));
      }
      if (STGCOLD == ON) {
        gbl_col = 21;
        printfn("STAGE HOT");
      }
```

```
      gbl_col = 64;
      if (motn_sol == 0) {printfn("UP ");}
      uc0 = getikey(motn_scr);
      if (uc0 == F1 && STGCOLD == OFF) {
        motn_loc = 0;
      } else if (uc0 == F2) {
        motn_loc = 7;
      } else if (uc0 >= 2 && uc0 <= 6) {
        motn_loc = uc0 - 1;
      } else if (uc0 == F3) {
        motn_sol = (motn_sol + 1) % 2;
        clrscrn();
        gbl_col = 5; printfn("Processing...");
        run_sol();
      } else if (uc0 == F5) {
        break;
      }
      while (motn_loc != motnlnow) {
        gbl_port = 3;
        clrscrn();
        gbl_col = 7; printfn("Move to:");
        gbl_col = 29;
        if (motn_loc == 0) {
          printfn("LOAD");
        } else if (motn_loc == 7) {
          printfn("HOLD");
        } else {
          printfn("POS%1u",(motn_loc + 1));
        }
        gbl_col = 41;
        printfn(" F1-Yes F5-No ");
        do {
          uc0 = scankeyL();
        } while (uc0 == 255);
        if (uc0 == F1) {
          gbl_col = 41;
          printfn("  ");
          gbl_col = 6; printfn("Moving to:");
          run_motn();
        } else if (uc0 == F5) {
          motn_loc = motnlnow;
        }
      }
    }
  }

////////////////////// Functions Begin \\\\\\\\\\\\\\\\\\\\\\

/* ScankeyL - Return Char, Check for leaks */
unsigned char scankeyL() {
  unsigned char skl;
```

```
   skl = scankey();
   if (NOLEAK == OFF) { // leak detected
      clrscrn();
      gbl_col = 1; printfn("-- LEAK DETECTED --");
      clrperi();
      gbl_col = 46; printfn("Repair and Press F5 to continue");
      do { } while (scankey() != F5);
      skl = 255;
   }
   return(skl);
}

/* Getikey - Return Integer */
unsigned char getikey(char scr_menu[81]) {
   i_wait = 0;
   gik1 = 0;
   do {
      gik0 = scankeyL();
      i_wait = i_wait + 1;
      if (i_wait == 20000 || (i_wait == 50 && gik1 == 1)) {
         gik1 = 1;
         i_wait = 0;
         if (ucx >= 15) {
            ucx = 15;
            ucxs = 0;
         }
         if (ucx <= 1) {
            ucx = 1;
            ucxs = 1;
         }
         if (ucy >= 6) {
            ucy = 6;
            ucys = 0;
         }
         if (ucy <= 0) {
            ucy = 0;
            ucys = 1;
         }
         if (ucxs == 1) { ucx = ucx + 1; } else { ucx = ucx - 1; }
         if (ucys == 1) { ucy = ucy + 2; } else { ucy = ucy - 2; }
         clrscrn();
         gbl_col = (ucy * 10) + ucx; printfn("ASCEFI");
      }
   } while (gik0 == 255);
   if (gik1 == 1) {
      gik1 = 0;
      gik0 = 255;
      if (strcmp(scr_menu,lock_scr) != 0) {
         lock();
      }
      printscr(scr_menu);
```

```
  }
  return(gik0);
}

/* Load tempstr & tempstr2 with select values from programs */
void settmp(unsigned char prog, unsigned char starti, unsigned char endi) {
  if (prog == 1) {
    for (stuc = 0; stuc <= endi - starti; stuc++) {
      tempstr[stuc] = uprogram[(stuc + starti)];
      tempstr2[stuc] = uprogram[(stuc + starti)];
      tempstr[stuc+1] = sts[1];
    }
  } else if (prog == 2) {
    for (stuc = 0; stuc <= endi - starti; stuc++) {
      tempstr[stuc] = tprogram[(stuc + starti)];
      tempstr2[stuc] = tprogram[(stuc + starti)];
      tempstr[stuc+1] = sts[1];
    }
  }
}

/* Update Optoswitches for new location */
void run_loc() {
  // CW is right-hand thread from motor (location 7)
  // CCW is left-hand thread from motor (location 0)
  if (motn_loc == motnlnow) {
    motn_dir = 2;
  } else {
    if (motn_loc > motnlnow) {
      motn_dir = CW1;
    } else {
      motn_dir = CCW1;
    }
    adelay(1000);

    if ((motn_loc % 2) == 0) {
      err_str = wr_stprs_out(1,LOC1,ON);
    } else {
      err_str = wr_stprs_out(1,LOC1,OFF);
    }
    if (motn_loc == 0 || motn_loc == 1 || motn_loc == 5) {
      err_str = wr_stprs_out(1,LOC2,ON);
    } else {
      err_str = wr_stprs_out(1,LOC2,OFF);
    }
    if (motn_loc >= 0 && motn_loc <= 3) {
      err_str = wr_stprs_out(1,LOC3,ON);
    } else {
      err_str = wr_stprs_out(1,LOC3,OFF);
    }
  }
```

```
  motnlnow = motn_loc;
}

/* Update Motion Controls */
void run_motn() {
  stop_stprs(1,0);
  run_loc();
  if (motn_dir != 2) {
    motn_sol = 1; run_sol(); // Raise the platform
    if (CWLIM == ON || CCWLIM == ON) {
      rmuc2 = 1;
    } else {
      rmuc2 = 0;
    }
    if ((rmuc2 == 1 && motnldir != motn_dir) || rmuc2 == 0) {
      motnldir = motn_dir;
      err_str = spd_ops(1,149,motnldir,(STEPPER1_SPEED/(0.025*STEPPER1_UNIT))); // Start ...
          movement
      for (rmuc1 = 0; rmuc1 <= 180; rmuc1 = rmuc1 + 5) {
        err_str = spd_ops(1,136,1,(STEPPER1_SPEED/(sin(rmuc1/360.0*pye)*7.5*STEPPER1_UNIT) ...
            )); // Accelerate
        adelay(30);
        if (CWLIM > rmuc2 || CCWLIM > rmuc2) {
          stop_stprs(1,0);
        }
      }
      while (CWLOC == OFF && CCWLOC == OFF && CWLIM <= rmuc2 && CCWLIM <= ...
          rmuc2) {
        if (rmuc2 == 1 && CWLIM == OFF && CCWLIM == OFF) {
          rmuc2 = 0;
        }
      }
      for (rmuc1 = 0; rmuc1 <= 170; rmuc1 = rmuc1 + 5) {
        if (CWLIM == ON || CCWLIM == ON) {
          stop_stprs(1,0);
        }
        err_str = spd_ops(1,136,1,(STEPPER1_SPEED/(cos(rmuc1/360.0*pye)*7.5*STEPPER1_UNIT) ...
            )); // Decelerate
        adelay(40);
      }
      while (CWLOC == ON && CCWLOC == ON && CWLIM == OFF && CCWLIM == OFF) {
        adelay(1700);
      }
      stop_stprs(1,0);
      if (CWLIM == ON || CCWLIM == ON) {
        rmuc2 = 1;
        motnldir = (motn_dir + 1)%2;
        err_str = spd_ops(1,149,motnldir,(STEPPER1_SPEED/(0.025*STEPPER1_UNIT))); // Start ...
            movement
        for (rmuc1 = 5; rmuc1 <= 180; rmuc1 = rmuc1 + 5) {
          err_str = spd_ops(1,136,1,(STEPPER1_SPEED/(sin(rmuc1/360.0*pye)*7.5* ...
```

```
                STEPPER1_UNIT))); // Accelerate
            adelay(30);
            if (CWLIM > rmuc2 || CCWLIM > rmuc2) {
              stop_stprs(1,0);
            }
          }
          while (CWLOC == OFF && CCWLOC == OFF && CWLIM <= rmuc2 && CCWLIM <=  ...
              rmuc2) {
            if (rmuc2 == 1 && CWLIM == OFF && CCWLIM == OFF) {
              rmuc2 = 0;
            }
          }
          for (rmuc1 = 0; rmuc1 <= 170; rmuc1 = rmuc1 + 5) {
            if (CWLIM == ON || CCWLIM == ON) {
              stop_stprs(1,0);
            }
            err_str = spd_ops(1,136,1,(STEPPER1_SPEED/(cos(rmuc1/360.0*pye)*7.5* ...
                STEPPER1_UNIT))); // Decelerate
            adelay(40);
          }
          while (CWLOC == ON && CCWLOC == ON && CWLIM == OFF && CCWLIM == OFF) {
            adelay(1700);
          }
          stop_stprs(1,0);
          if (CWLIM == ON || CCWLIM == ON) {
            if (motnldir == CW1) { motnlnow = 7;}
            if (motnldir == CCW1) { motnlnow = 0;}
            clrscrn();
            gbl_col = 5; printfn("MOTION ERROR Position %u Not Found Press any  ...
                key...",(motn_loc+1));
            do { } while (scankeyL() == 255);
          }
        }
      }
    }
  }

/* Update Solenoid Controls */
void run_sol() {
  wrdigout(motn_sol,SOLENOID); // Raise or Lower Solenoids
  if (motn_sol) {
    do { } while (SOLUP == OFF);
  } else {
    do { } while (SOLDWN == OFF);
  }
}

/* Update Water Bath Controls */
void run_bath() {
  mess[1] = 0x03;
  mess[2] = 0x00; mess[3] = 0x01;
```

```
  mess[4] = 0x00; mess[5] = 0x01;
  send_mess(5,4);
  wb_temp = ((reply[3]<<8) + reply[4]) / 100.0;
}

void run_bath2() {
  mess[1] = 0x03;
  mess[2] = 0x00; mess[3] = 0x02;
  mess[4] = 0x00; mess[5] = 0x01;
  send_mess(5,4);
  wb_set = ((reply[3]<<8) + reply[4]) / 100.0;
}

void run_bath_set() {
  mess[1] = 0x06;
  mess[2] = 0x00; mess[3] = 0x02;
  rbui = wb_set*100;
  mess[4] = (rbui >> 8) & 0xff;
  mess[5] = rbui & 0xff;
  send_mess(5,5);
  adelay(1000);
}

void run_stg_temp() {
  rbui = 1; stg_security();

  mess[1] = 0x06; // Set SEt.3 (Set point for 3)
  mess[2] = 0x02; mess[3] = 0xe8; // Stage Hot set point
  rbui = (wb_set+8)*10; // Water Bath + 9 C
  mess[4] = (rbui >> 8) & 0xff;
  mess[5] = rbui & 0xff;
  send_mess(5,5);

  mess[2] = 0x00; mess[3] = 0x81; // Set SEt.2 (Set point for 2)
  mess[4] = 0x01; // Stage Cool set point
  mess[5] = 0x2c;
  send_mess(5,5);

  mess[3] = 0x7f;
  rbui = (wb_set+10)*10; // Water Bath + 10 C
  mess[4] = (rbui >> 8) & 0xff;
  mess[5] = rbui & 0xff;
  send_mess(5,5);

  rbui = 0; stg_security();
}

void run_stg_off() { // Turn off stage heaters
  rbui = 1; stg_security();
  mess[1] = 0x06;
  mess[2] = 0x00; mess[3] = 0x7f;
```

```
  mess[4] = 0x00; // 20.0 degrees C
  mess[5] = 0xc8;
  send_mess(5,5);
  rbui = 0; stg_security();
}

void stg_security() {
  if (rbui == 1) {
    mess[2] = 0x15; // Security Enter
  } else {
    mess[2] = 0x16; // Security Exit
  }
  mess[3] = 0x00;
  mess[4] = 0x00; mess[5] = 0x00;
  send_mess(5,5);
}

void send_mess(unsigned char im, unsigned char ir) {
  smucp = IP;
  smuce = IE;
  for (smuc2 = 0; smuc2 <= im; smuc2 = smuc2 + 1) {
    tmess[smuc2] = mess[smuc2];
  }
  crc_calc(im);
  mess[im+1]=tmess[im+1];
  mess[im+2]=tmess[im+2];
  smuc1 = 0;
  while (smuc1 < 10) {
    smuc1 = smuc1 + 1;
    reply[1] = 0x00;
    for (smuc2 = 0; smuc2 <= im + 2; smuc2 = smuc2 + 1) {
      tmess[smuc2] = mess[smuc2];
    }
    gbl_port = 1;
    clr_ri();
    clr_ti();
    ser1_finished = 0;
    for (smuc2 = 0; smuc2 <= im + 2; smuc2 = smuc2 + 1) {
      putchar(tmess[smuc2]);
    }
    init_ser1();
    ser1_int();
    while (ser1_finished == 0) { }
    if (bcd2int(reply[1],1) == bcd2int(mess[1],1)) {
      for (smuc2 = 0; smuc2 <= ir; smuc2 = smuc2 + 1) {
        tmess[smuc2] = reply[smuc2];
      }
      crc_calc(ir);
      if (bcd2int(reply[ir+1],1) == bcd2int(tmess[ir+1],1) && bcd2int(reply[ir+2],1) == bcd2int(tmess[  ...
            ir+2],1)) {
        smuc1 = 10;
```

```
        }
      }
    }
    IP = smucp;
    IE = smuce;
    gbl_port = 3;
}

void crc_calc(unsigned int crcloc) {
    crc=0xffff;
    for (crcui=0; crcui <= crcloc; crcui = crcui + 1) {
      thisbyte = 0x00ff & tmess[crcui];
      crc = crc ^ thisbyte;
      for (shift = 1; shift <= 8; shift = shift + 1) {
        lastbit = crc & 0x0001;
        crc = (crc >> 1) & 0x7fff;
        if (lastbit == 0x0001) {
          crc = crc ^ 0xa001;
        }
      }
    }
    tmess[crcloc+1] = crc & 0xff;
    tmess[crcloc+2] = (crc >> 8) & 0xff;
}

/* Update Peristalic Pump Controls */
void run_peri() {
    if ((twv_now != 0 && twv_stat != twv_now && peri_now == 1) || (peri_now == 1 && peri_pwr !=  ...
        peri_now)) {
      wrdigout(0,PERIPWR);
      adelay(1000);
      wrdigout(0,PERIDIR); // PERIDIR = REVERSE;
      adelay(1000);
      wrdigout(1,PERIPWR);
      adelay(20000);
      wrdigout(0,TWV_ON);
      adelay(1000);
      if (twv_stat == 2) {
        err_str = wr_stprs_out(1,U_TH,ON);
        adelay(1000);
        wrdigout(1,TWV_ON);
      } else if (twv_stat == 1) {
        err_str = wr_stprs_out(1,U_TH,OFF);
        adelay(1000);
        wrdigout(1,TWV_ON);
      } else {
        err_str = wr_stprs_out(1,U_TH,OFF);
      }
      if (peri_pwr != 2) {
        wrdigout(0,PERIPWR);
        adelay(1000);
```

```
      }
      if (peri_pwr == 1){
        wrdigout(1,PERIDIR); // PERIDIR = FORWARD;
        adelay(1000);
        wrdigout(1,PERIPWR);
      }
      twv_now = twv_stat;
      peri_now = peri_pwr;
    }
    if (twv_stat != twv_now) {
      wrdigout(0,TWV_ON);
      adelay(1000);
      if (twv_stat == 2) {
        err_str = wr_stprs_out(1,U_TH,ON);
        adelay(1000);
        wrdigout(1,TWV_ON);
      } else if (twv_stat == 1) {
        err_str = wr_stprs_out(1,U_TH,OFF);
        adelay(1000);
        wrdigout(1,TWV_ON);
      } else {
        err_str = wr_stprs_out(1,U_TH,OFF);
      }
      twv_now = twv_stat;
    }
    if (peri_now != peri_pwr) {
      wrdigout(0,PERIPWR);
      adelay(1000);
      if (peri_pwr == 2) {
        wrdigout(0,PERIDIR); // PERIDIR = REVERSE;
      } else {
        wrdigout(1,PERIDIR); // PERIDIR = FORWARD;
      }
      adelay(1000);
      if (peri_pwr > 0) {
        wrdigout(1,PERIPWR);
      }
      peri_now = peri_pwr;
    }
}

/* Send Serial Data to Shift Registers */
void run_ser() {
    serial_data = "000000000000000000000000";
    ser_con = ser_tot;
    for (rsuc1 = 0; rsuc1 < 24; rsuc1 = rsuc1 + 1) {
      rsuc0 = ser_con % 2;
      ser_con = (ser_con - rsuc0) / 2;
      if (rsuc0 == 1) {
        serial_data[rsuc1] = '1';
      } else {
```

```
        serial_data[rsuc1] = '0';
      }
    }
    err_str = wr_stprs_out(1,SER_DATA,LOW);
    for (rsuc1 = 24; rsuc1 > 0; rsuc1 = rsuc1 - 1) {
      err_str = wr_stprs_out(1,SER_CLK,HIGH);
      err_str = wr_stprs_out(1,STRTELEC,HIGH);
      adelay(40);
      err_str = wr_stprs_out(1,SER_CLK,LOW);
      err_str = wr_stprs_out(1,STRTELEC,LOW);
      adelay(20);
      if (serial_data[rsuc1-1] == '0') {
        err_str = wr_stprs_out(1,SER_DATA,LOW);
      } else {
        err_str = wr_stprs_out(1,SER_DATA,HIGH);
      }
      adelay(20);
    }
    err_str = wr_stprs_out(1,SER_CLK,HIGH);
}

/* Create an interrupt for Com1 */
interrupt (6) ser1_int()
{
    asm clr ie.7; //clears only the MSB
    asm mov dptr,#_reply; // Load the address
    asm wait_fpr_flag: //This is about 20MS timer (#7fh)
    asm mov r5,#7fh //Load it

    asm wait_fptr_flag1:
    asm mov r6,#0ffH

    asm wait_fptr_flag2:
    asm jb scon.0,read_fpr_flag1;
    asm djnz r6,wait_fptr_flag2;
    asm djnz r5,wait_fptr_flag1;
    asm ljmp finish_ser_flag; //Timer is expired

    asm read_fpr_flag1:
    asm clr scon.0;
    asm mov a,sbuf;
    asm movx @dptr,a; //Save the data
    asm inc dptr;
    asm ljmp wait_fpr_flag //Keep readig until timer expires

    asm finish_ser_flag:
    asm mov a,#00h; //Mark the end
    asm movx @dptr,a;
    asm mov dptr,#_ser1_finished; //Load the finish flag
    asm mov a,#1h;
    asm movx @dptr,a; // let it know it is finished receiving (Set ser1_finished = 1)
```

```
  SCON = SCON | 0x10; // Reset the interrupt
  asm exit_int: // Exit
  asm clr scon.1;
}

void init_ser1()
{
  asm orl ip,#10h;
  asm orl ie,#90h;
}
```

End user.h

B.6 Daughter Board Microcontroller Program

Not all of the code is documented here due to size. A full copy of the program may be requested from the author via email at martin@ohmes.com.

B.6.1 main.c

This is the main program, or entry point, for the daughter board's microcontroller.

Begin main.c

```
#define __MAIN_C

// NOTE: DEBUG switch
// Must be Down in order to operate correctly

/************ Header Files ************/
#include <P18F8722.H>
#include "main.h"
#include <stdlib.h>
#include "GenericTypeDefs.h"
#include "Compiler.h"
#include "globals.h"
#include "config.h"
#include "HVLLD.h"
#include "convert.h"
#include "BitBangADC.h"
#include "PulseMode.h"
#include "Tick.h"
#include "PulseMode.h"

static volatile BOOL SyncTmr = TRUE;
static volatile unsigned char TMR3CountDown = 0x13u;
BOOL SyncTimer(void);
```

```
void ResetSyncTimer(void);
void MPFD_Write_Done(void);
void MPFD_Read_Done(void);
void ResetRelays(void);
void Board_Identify(void);
BOOL TransmitData(void);
BOOL IsTMR3CountDownZero(void);
BOOL IsTMR3CountDownNotZero(void);
void ResetTimer3(void);
void StopTimer3(void);
void SetMPFDArrayBit(unsigned char MPFDByte, unsigned char MPFDBit, unsigned int MPFDData  ...
    );

/*****************************************\
* Function: void MyHighISR(void)
* Input: Occurs on a timer overflow
* or TRIGGER rising edge
* Output: Increment timers
* or set TRIGGER_FLAG
* Overview: Although MyHighISR is a high priority interrupt, use
* interruptlow pragma so that the compiler will not use
* retfie FAST.
\*****************************************/
#pragma interruptlow HighISR // declare function as high priority isr
void HighISR(void) {
    if (PIR3bits.TMR4IF == 1u) {
        PIR3bits.TMR4IF = 0; // Clear flag
        T4CONbits.TMR4ON = 0; // Timer4 off
        SyncTmr = FALSE;
    }
    if (PIR2bits.TMR3IF) {
        PIR2bits.TMR3IF = 0; // Clear flag
        --TMR3CountDown;
        if (TMR3CountDown == 0u) {
            T3CONbits.TMR3ON = 0; // Stops Timer3
            TRIGGER_TRIS = 0;
        } else {
            T3CONbits.TMR3ON = 0; // Stops Timer3
            TMR3H = 0x00u;
            TMR3L = 0x00u; // Set for 1/19 second timeout
            T3CONbits.TMR3ON = 1; // Start Timer3
        }
    }
}

/*****************************************\
* Function: void MyLowISR(void)
* Input: Occurs on a timer overflow
* Output: Increment timers
* Overview:
\*****************************************/
```

```
#pragma interruptlow LowISR// declare function as low priority isr
void LowISR(void) {
    TickUpdate();
}

/************ Interrupt Vector Sections ************/
#pragma code lowVector=0x18
void LowVector(void) {_asm goto LowISR _endasm}
#pragma code highVector=0x08
void HighVector(void) {_asm goto HighISR _endasm}

/*****************************************\
* Function: void main(void)
\*****************************************/
#pragma code /* return to default code section */

static DWORD TickSeconds;
static far unsigned char BoardID = 0x00u;
static far unsigned short long HVData = 0x000000u;
static far unsigned int HVvOut;

static far unsigned int HV_Setpoint = 0x0000u;
static far unsigned int HV_NewSetpoint = 0x0000u;
static far unsigned char VoltageChanging = 0x00u;
static far unsigned int LLD_Setpoint = 0x0000u;
static far unsigned int LLD_NewSetpoint = 0x0000u;
static far unsigned char Speed = 0x00u;
static far unsigned char Speed_New = 0x00u;
static far unsigned char MSV_Res = 0x00u;
static far unsigned char MSV_Res_New = 0x00u;
static far unsigned char Curr_Res = 0x00u;
static far unsigned char Curr_Res_New = 0x00u;
static far unsigned char Temp_Res = 0x00u;
static far unsigned char Temp_Res_New = 0x00u;
static far unsigned char NextResetRelays = 0x00u;
static far unsigned char DoRelayReset = 0x00u;

static unsigned char CollectDataC = 0x00u;
static unsigned char CollectDataK = 0x00u;
static unsigned char CollectDataD = 0x00u;
static unsigned char CollectDataW = 0x00u;

typedef union _MPFDArray {
    unsigned int v[44];
} MPFDArray;

static MPFDArray DataArray;

void main(void) {
    unsigned int PLH = 0x0000u;
    unsigned char uc0;
```

```
CONFIG_Init();
Init_ReadWrite();
Init_System();

while(1) {
    TRIGGER_TRIS = 0;
    WRITE_TRIS = 0;
    READ_TRIS = 0;
    ResetTimer3();

    if ((DataArray.v[0] & 0x8000u) == 0x8000u) {
        /*** Load New LLD Setpoint ***/
        if (LLD_NewSetpoint != LLD_Setpoint) {
            if (SetLLD(LLD_NewSetpoint) == 0x01u) {
                LLD_Setpoint = LLD_NewSetpoint;
            }
        }

        Speed = Speed_New;
        MSV_Res = MSV_Res_New;
        Curr_Res = Curr_Res_New;
        Temp_Res = Temp_Res_New;

        if (STRING_STATUS_IN != 1u) {
            HV_NewSetpoint = 0x0000u;
        }
        HV_Setpoint = HV_NewSetpoint;
    }

    for (uc0 = 0u; uc0 <= 43u; uc0++) {
        DataArray.v[uc0] = 0x0000u;
    }

    if (HV_Setpoint == 0x0032u) {
        HV_Setpoint = 0x0033u;
    }

    if (ChangeHVreturn(HV_Setpoint) == HV_Setpoint) {
        VoltageChanging = 0x00u;
    } else {
        VoltageChanging = 0x01u;
    }
    StopTimer3();

    while (!TW_IN_BOOL); // Wait until TW_IN goes high

    /*** Data Transfer Process ***/
    TRIGGER_TRIS = 1;
    while (!TRIGGER_IN_BOOL); // Wait for Trigger to go high
    TR1_OUT = 0;
```

```
ResetTimer3();

/*** Load Output Data Array ***/
DataArray.v[1] |= ((unsigned int)STRING_STATUS_IN) << 15u;
DataArray.v[1] |= ((unsigned int)Speed) << 11u;

ADC_getdata();
ADC_config();
GetPulseData();
DataArray.v[3] = convertHV(DataArray.v[3], (((unsigned char)(DataArray.v[0])) & 0x03u)  ...
    << 2);
for (CollectDataC = 4u; CollectDataC <= 7u; CollectDataC++) {
    DataArray.v[CollectDataC] = convertTemp(DataArray.v[CollectDataC], Speed);
}
for (CollectDataC = 8u; CollectDataC <= 19u; CollectDataC++) {
    PLH |= DataArray.v[CollectDataC];
}
for (CollectDataC = 20u; CollectDataC <= 31u; CollectDataC++) {
    DataArray.v[CollectDataC] = convertMSV(DataArray.v[CollectDataC], Speed);
}
for (CollectDataC = 32u; CollectDataC <= 43u; CollectDataC++) {
    DataArray.v[CollectDataC] = convertCURR(DataArray.v[CollectDataC], Speed);
}
DataArray.v[0] |= (PLH & 0xC000u) >> 10;

TR1_OUT = 1;
while (TW_IN_BOOL && TRIGGER_IN_BOOL);
ResetSyncTimer();

if (TransmitData() == FALSE) {
    TRIGGER_TRIS = 0;
    READ_TRIS = 0;
    WRITE_TRIS = 0;
    TR1_OUT = 0;
    TR2_OUT = 0;
    TR3_OUT = 0;
    DataArray.v[0] &= 0x7FFFu;
} else {
    MPFD_Read_Done();

    if (TRIGGER_IN_BOOL) {
        DataArray.v[0] |= 0x8000u;
    } else {
        DataArray.v[0] &= 0x7FFFu;
    }

    MPFD_Write_Done();

    MPFD_Read_Done();
```

```
            while (SyncTimer() && (TRIGGER_IN_BOOL));
        }
        StopTimer3();
    }
}

BOOL TransmitData(void) {
    TR1_OUT = (DataArray.v[0] >> 5) & 0x0001u;
    TR2_OUT = (DataArray.v[0] >> 4) & 0x0001u;
    TR3_OUT = (DataArray.v[0] >> 3) & 0x0001u;
    MPFD_Write_Done(); // 0
    // Ignore first TW_IN
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[0] >> 2) & 0x0001u;
    TR2_OUT = (DataArray.v[0] >> 1) & 0x0001u;
    TR3_OUT = DataArray.v[0] & 0x0001u;
    MPFD_Write_Done(); // 1
    Speed_New = 0x00u | (TW_IN << 3u);
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[1] >> 15) & 0x0001u;
    TR2_OUT = (DataArray.v[1] >> 14) & 0x0001u;
    TR3_OUT = (DataArray.v[1] >> 13) & 0x0001u;
    MPFD_Write_Done(); // 2
    if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
        return(FALSE);
    }
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[1] >> 12) & 0x0001u;
    TR2_OUT = (DataArray.v[1] >> 11) & 0x0001u;
    TR3_OUT = (DataArray.v[1] >> 10) & 0x0001u;
    MPFD_Write_Done(); // 3
    Speed_New |= (TW_IN << 2u);
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[1] >> 9) & 0x0001u;
    TR2_OUT = (DataArray.v[1] >> 8) & 0x0001u;
    TR3_OUT = (DataArray.v[1] >> 7) & 0x0001u;
    MPFD_Write_Done(); // 4
    if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
        return(FALSE);
    }
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[1] >> 6) & 0x0001u;
    TR2_OUT = (DataArray.v[1] >> 5) & 0x0001u;
    TR3_OUT = (DataArray.v[1] >> 4) & 0x0001u;
    MPFD_Write_Done(); // 5
    MSV_Res_New = 0x00u | (TW_IN << 1u);
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[1] >> 3) & 0x0001u;
    TR2_OUT = (DataArray.v[1] >> 2) & 0x0001u;
    TR3_OUT = (DataArray.v[1] >> 1) & 0x0001u;
    MPFD_Write_Done(); // 6
```

```
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[1] & 0x0001u;
TR2_OUT = (DataArray.v[2] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[2] >> 14) & 0x0001u;
MPFD_Write_Done(); // 7
MSV_Res_New |= TW_IN;
MPFD_Read_Done();
TR1_OUT = (DataArray.v[2] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[2] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[2] >> 11) & 0x0001u;
MPFD_Write_Done(); // 8
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[2] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[2] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[2] >> 8) & 0x0001u;
MPFD_Write_Done(); // 9
Curr_Res_New = 0x00u | (TW_IN << 1u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[2] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[2] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[2] >> 5) & 0x0001u;
MPFD_Write_Done(); // 10
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[2] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[2] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[2] >> 2) & 0x0001u;
MPFD_Write_Done(); // 11
Curr_Res_New |= TW_IN;
MPFD_Read_Done();
TR1_OUT = (DataArray.v[2] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[2] & 0x0001u;
TR3_OUT = (DataArray.v[3] >> 15) & 0x0001u;
MPFD_Write_Done(); // 12
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[3] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[3] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[3] >> 12) & 0x0001u;
MPFD_Write_Done(); // 13
Temp_Res_New = 0x00u | (TW_IN << 1u);
```

```
MPFD_Read_Done();
TR1_OUT = (DataArray.v[3] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[3] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[3] >> 9) & 0x0001u;
MPFD_Write_Done(); // 14
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[3] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[3] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[3] >> 6) & 0x0001u;
MPFD_Write_Done(); // 15
Temp_Res_New |= TW_IN;
MPFD_Read_Done();
TR1_OUT = (DataArray.v[3] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[3] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[3] >> 3) & 0x0001u;
MPFD_Write_Done(); // 16
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[3] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[3] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[3] & 0x0001u;
MPFD_Write_Done(); // 17
LLD_NewSetpoint = 0x0000u | (((unsigned int)TW_IN) << 9u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[4] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[4] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[4] >> 13) & 0x0001u;
MPFD_Write_Done(); // 18
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[4] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[4] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[4] >> 10) & 0x0001u;
MPFD_Write_Done(); // 19
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 8u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[4] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[4] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[4] >> 7) & 0x0001u;
MPFD_Write_Done(); // 20
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
```

```
TR1_OUT = (DataArray.v[4] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[4] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[4] >> 4) & 0x0001u;
MPFD_Write_Done(); // 21
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 7u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[4] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[4] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[4] >> 1) & 0x0001u;
MPFD_Write_Done(); // 22
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[4] & 0x0001u;
TR2_OUT = (DataArray.v[5] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[5] >> 14) & 0x0001u;
MPFD_Write_Done(); // 23
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 6u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[5] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[5] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[5] >> 11) & 0x0001u;
MPFD_Write_Done(); // 24
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[5] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[5] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[5] >> 8) & 0x0001u;
MPFD_Write_Done(); // 25
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 5u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[5] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[5] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[5] >> 5) & 0x0001u;
MPFD_Write_Done(); // 26
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[5] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[5] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[5] >> 2) & 0x0001u;
MPFD_Write_Done(); // 27
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 4u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[5] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[5] & 0x0001u;
TR3_OUT = (DataArray.v[6] >> 15) & 0x0001u;
```

```
MPFD_Write_Done(); // 28
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[6] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[6] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[6] >> 12) & 0x0001u;
MPFD_Write_Done(); // 29
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 3u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[6] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[6] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[6] >> 9) & 0x0001u;
MPFD_Write_Done(); // 30
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[6] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[6] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[6] >> 6) & 0x0001u;
MPFD_Write_Done(); // 31
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 2u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[6] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[6] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[6] >> 3) & 0x0001u;
MPFD_Write_Done(); // 32
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[6] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[6] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[6] & 0x0001u;
MPFD_Write_Done(); // 33
LLD_NewSetpoint |= (((unsigned int)TW_IN) << 1u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[7] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[7] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[7] >> 13) & 0x0001u;
MPFD_Write_Done(); // 34
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[7] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[7] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[7] >> 10) & 0x0001u;
MPFD_Write_Done(); // 35
```

```
LLD_NewSetpoint |= ((unsigned int)TW_IN);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[7] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[7] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[7] >> 7) & 0x0001u;
MPFD_Write_Done(); // 36
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[7] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[7] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[7] >> 4) & 0x0001u;
MPFD_Write_Done(); // 37
HV_NewSetpoint = 0x0000u | (((unsigned int)TW_IN) << 9u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[7] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[7] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[7] >> 1) & 0x0001u;
MPFD_Write_Done(); // 38
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[7] & 0x0001u;
TR2_OUT = (DataArray.v[8] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[8] >> 14) & 0x0001u;
MPFD_Write_Done(); // 39
HV_NewSetpoint |= (((unsigned int)TW_IN) << 8u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[8] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[8] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[8] >> 11) & 0x0001u;
MPFD_Write_Done(); // 40
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[8] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[8] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[8] >> 8) & 0x0001u;
MPFD_Write_Done(); // 41
HV_NewSetpoint |= (((unsigned int)TW_IN) << 7u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[8] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[8] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[8] >> 5) & 0x0001u;
MPFD_Write_Done(); // 42
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
```

```
MPFD_Read_Done();
TR1_OUT = (DataArray.v[8] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[8] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[8] >> 2) & 0x0001u;
MPFD_Write_Done(); // 43
HV_NewSetpoint |= (((unsigned int)TW_IN) << 6u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[8] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[8] & 0x0001u;
TR3_OUT = (DataArray.v[9] >> 15) & 0x0001u;
MPFD_Write_Done(); // 44
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[9] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[9] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[9] >> 12) & 0x0001u;
MPFD_Write_Done(); // 45
HV_NewSetpoint |= (((unsigned int)TW_IN) << 5u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[9] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[9] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[9] >> 9) & 0x0001u;
MPFD_Write_Done(); // 46
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[9] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[9] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[9] >> 6) & 0x0001u;
MPFD_Write_Done(); // 47
HV_NewSetpoint |= (((unsigned int)TW_IN) << 4u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[9] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[9] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[9] >> 3) & 0x0001u;
MPFD_Write_Done(); // 48
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[9] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[9] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[9] & 0x0001u;
MPFD_Write_Done(); // 49
HV_NewSetpoint |= (((unsigned int)TW_IN) << 3u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[10] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[10] >> 14) & 0x0001u;
```

```
TR3_OUT = (DataArray.v[10] >> 13) & 0x0001u;
MPFD_Write_Done(); // 50
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[10] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[10] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[10] >> 10) & 0x0001u;
MPFD_Write_Done(); // 51
HV_NewSetpoint |= (((unsigned int)TW_IN) << 2u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[10] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[10] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[10] >> 7) & 0x0001u;
MPFD_Write_Done(); // 52
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[10] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[10] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[10] >> 4) & 0x0001u;
MPFD_Write_Done(); // 53
HV_NewSetpoint |= (((unsigned int)TW_IN) << 1u);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[10] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[10] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[10] >> 1) & 0x0001u;
MPFD_Write_Done(); // 54
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[10] & 0x0001u;
TR2_OUT = (DataArray.v[11] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[11] >> 14) & 0x0001u;
MPFD_Write_Done(); // 55
HV_NewSetpoint |= ((unsigned int)TW_IN);
MPFD_Read_Done();
TR1_OUT = (DataArray.v[11] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[11] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[11] >> 11) & 0x0001u;
MPFD_Write_Done(); // 56
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[11] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[11] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[11] >> 8) & 0x0001u;
```

```
MPFD_Write_Done(); // 57
//NextResetRelays = TW_IN;
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[11] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[11] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[11] >> 5) & 0x0001u;
MPFD_Write_Done(); // 58
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[11] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[11] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[11] >> 2) & 0x0001u;
MPFD_Write_Done(); // 59
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[11] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[11] & 0x0001u;
TR3_OUT = (DataArray.v[12] >> 15) & 0x0001u;
MPFD_Write_Done(); // 60
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[12] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[12] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[12] >> 12) & 0x0001u;
MPFD_Write_Done(); // 61
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[12] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[12] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[12] >> 9) & 0x0001u;
MPFD_Write_Done(); // 62
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[12] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[12] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[12] >> 6) & 0x0001u;
MPFD_Write_Done(); // 63
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
```

```
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[12] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[12] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[12] >> 3) & 0x0001u;
MPFD_Write_Done(); // 64
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[12] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[12] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[12] & 0x0001u;
MPFD_Write_Done(); // 65
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[13] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[13] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[13] >> 13) & 0x0001u;
MPFD_Write_Done(); // 66
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[13] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[13] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[13] >> 10) & 0x0001u;
MPFD_Write_Done(); // 67
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[13] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[13] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[13] >> 7) & 0x0001u;
MPFD_Write_Done(); // 68
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[13] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[13] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[13] >> 4) & 0x0001u;
MPFD_Write_Done(); // 69
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
```

```
TR1_OUT = (DataArray.v[13] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[13] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[13] >> 1) & 0x0001u;
MPFD_Write_Done(); // 70
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[13] & 0x0001u;
TR2_OUT = (DataArray.v[14] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[14] >> 14) & 0x0001u;
MPFD_Write_Done(); // 71
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[14] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[14] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[14] >> 11) & 0x0001u;
MPFD_Write_Done(); // 72
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[14] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[14] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[14] >> 8) & 0x0001u;
MPFD_Write_Done(); // 73
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[14] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[14] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[14] >> 5) & 0x0001u;
MPFD_Write_Done(); // 74
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[14] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[14] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[14] >> 2) & 0x0001u;
MPFD_Write_Done(); // 75
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[14] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[14] & 0x0001u;
TR3_OUT = (DataArray.v[15] >> 15) & 0x0001u;
```

```
MPFD_Write_Done(); // 76
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[15] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[15] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[15] >> 12) & 0x0001u;
MPFD_Write_Done(); // 77
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[15] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[15] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[15] >> 9) & 0x0001u;
MPFD_Write_Done(); // 78
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[15] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[15] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[15] >> 6) & 0x0001u;
MPFD_Write_Done(); // 79
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[15] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[15] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[15] >> 3) & 0x0001u;
MPFD_Write_Done(); // 80
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[15] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[15] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[15] & 0x0001u;
MPFD_Write_Done(); // 81
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[16] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[16] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[16] >> 13) & 0x0001u;
MPFD_Write_Done(); // 82
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
```

```
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[16] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[16] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[16] >> 10) & 0x0001u;
MPFD_Write_Done(); // 83
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[16] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[16] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[16] >> 7) & 0x0001u;
MPFD_Write_Done(); // 84
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[16] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[16] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[16] >> 4) & 0x0001u;
MPFD_Write_Done(); // 85
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[16] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[16] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[16] >> 1) & 0x0001u;
MPFD_Write_Done(); // 86
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[16] & 0x0001u;
TR2_OUT = (DataArray.v[17] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[17] >> 14) & 0x0001u;
MPFD_Write_Done(); // 87
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[17] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[17] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[17] >> 11) & 0x0001u;
MPFD_Write_Done(); // 88
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[17] >> 10) & 0x0001u;
```

```
TR2_OUT = (DataArray.v[17] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[17] >> 8) & 0x0001u;
MPFD_Write_Done(); // 89
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[17] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[17] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[17] >> 5) & 0x0001u;
MPFD_Write_Done(); // 90
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[17] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[17] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[17] >> 2) & 0x0001u;
MPFD_Write_Done(); // 91
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[17] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[17] & 0x0001u;
TR3_OUT = (DataArray.v[18] >> 15) & 0x0001u;
MPFD_Write_Done(); // 92
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[18] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[18] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[18] >> 12) & 0x0001u;
MPFD_Write_Done(); // 93
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[18] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[18] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[18] >> 9) & 0x0001u;
MPFD_Write_Done(); // 94
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[18] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[18] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[18] >> 6) & 0x0001u;
MPFD_Write_Done(); // 95
```

```
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[18] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[18] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[18] >> 3) & 0x0001u;
MPFD_Write_Done(); // 96
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[18] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[18] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[18] & 0x0001u;
MPFD_Write_Done(); // 97
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[19] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[19] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[19] >> 13) & 0x0001u;
MPFD_Write_Done(); // 98
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[19] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[19] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[19] >> 10) & 0x0001u;
MPFD_Write_Done(); // 99
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[19] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[19] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[19] >> 7) & 0x0001u;
MPFD_Write_Done(); // 100
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[19] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[19] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[19] >> 4) & 0x0001u;
MPFD_Write_Done(); // 101
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
```

```
MPFD_Read_Done();
TR1_OUT = (DataArray.v[19] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[19] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[19] >> 1) & 0x0001u;
MPFD_Write_Done(); // 102
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[19] & 0x0001u;
TR2_OUT = (DataArray.v[20] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[20] >> 14) & 0x0001u;
MPFD_Write_Done(); // 103
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[20] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[20] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[20] >> 11) & 0x0001u;
MPFD_Write_Done(); // 104
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[20] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[20] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[20] >> 8) & 0x0001u;
MPFD_Write_Done(); // 105
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[20] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[20] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[20] >> 5) & 0x0001u;
MPFD_Write_Done(); // 106
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[20] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[20] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[20] >> 2) & 0x0001u;
MPFD_Write_Done(); // 107
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[20] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[20] & 0x0001u;
```

```
TR3_OUT = (DataArray.v[21] >> 15) & 0x0001u;
MPFD_Write_Done(); // 108
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[21] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[21] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[21] >> 12) & 0x0001u;
MPFD_Write_Done(); // 109
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[21] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[21] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[21] >> 9) & 0x0001u;
MPFD_Write_Done(); // 110
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[21] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[21] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[21] >> 6) & 0x0001u;
MPFD_Write_Done(); // 111
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[21] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[21] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[21] >> 3) & 0x0001u;
MPFD_Write_Done(); // 112
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[21] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[21] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[21] & 0x0001u;
MPFD_Write_Done(); // 113
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[22] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[22] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[22] >> 13) & 0x0001u;
MPFD_Write_Done(); // 114
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
```

```
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[22] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[22] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[22] >> 10) & 0x0001u;
MPFD_Write_Done(); // 115
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[22] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[22] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[22] >> 7) & 0x0001u;
MPFD_Write_Done(); // 116
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[22] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[22] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[22] >> 4) & 0x0001u;
MPFD_Write_Done(); // 117
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[22] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[22] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[22] >> 1) & 0x0001u;
MPFD_Write_Done(); // 118
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[22] & 0x0001u;
TR2_OUT = (DataArray.v[23] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[23] >> 14) & 0x0001u;
MPFD_Write_Done(); // 119
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[23] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[23] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[23] >> 11) & 0x0001u;
MPFD_Write_Done(); // 120
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
```

```
TR1_OUT = (DataArray.v[23] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[23] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[23] >> 8) & 0x0001u;
MPFD_Write_Done(); // 121
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[23] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[23] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[23] >> 5) & 0x0001u;
MPFD_Write_Done(); // 122
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[23] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[23] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[23] >> 2) & 0x0001u;
MPFD_Write_Done(); // 123
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[23] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[23] & 0x0001u;
TR3_OUT = (DataArray.v[24] >> 15) & 0x0001u;
MPFD_Write_Done(); // 124
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[24] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[24] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[24] >> 12) & 0x0001u;
MPFD_Write_Done(); // 125
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[24] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[24] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[24] >> 9) & 0x0001u;
MPFD_Write_Done(); // 126
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[24] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[24] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[24] >> 6) & 0x0001u;
```

```
MPFD_Write_Done(); // 127
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[24] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[24] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[24] >> 3) & 0x0001u;
MPFD_Write_Done(); // 128
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[24] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[24] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[24] & 0x0001u;
MPFD_Write_Done(); // 129
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[25] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[25] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[25] >> 13) & 0x0001u;
MPFD_Write_Done(); // 130
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[25] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[25] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[25] >> 10) & 0x0001u;
MPFD_Write_Done(); // 131
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[25] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[25] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[25] >> 7) & 0x0001u;
MPFD_Write_Done(); // 132
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[25] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[25] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[25] >> 4) & 0x0001u;
MPFD_Write_Done(); // 133
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
```

```
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[25] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[25] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[25] >> 1) & 0x0001u;
MPFD_Write_Done(); // 134
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[25] & 0x0001u;
TR2_OUT = (DataArray.v[26] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[26] >> 14) & 0x0001u;
MPFD_Write_Done(); // 135
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[26] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[26] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[26] >> 11) & 0x0001u;
MPFD_Write_Done(); // 136
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[26] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[26] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[26] >> 8) & 0x0001u;
MPFD_Write_Done(); // 137
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[26] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[26] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[26] >> 5) & 0x0001u;
MPFD_Write_Done(); // 138
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[26] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[26] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[26] >> 2) & 0x0001u;
MPFD_Write_Done(); // 139
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[26] >> 1) & 0x0001u;
```

```
TR2_OUT = DataArray.v[26] & 0x0001u;
TR3_OUT = (DataArray.v[27] >> 15) & 0x0001u;
MPFD_Write_Done(); // 140
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[27] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[27] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[27] >> 12) & 0x0001u;
MPFD_Write_Done(); // 141
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[27] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[27] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[27] >> 9) & 0x0001u;
MPFD_Write_Done(); // 142
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[27] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[27] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[27] >> 6) & 0x0001u;
MPFD_Write_Done(); // 143
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[27] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[27] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[27] >> 3) & 0x0001u;
MPFD_Write_Done(); // 144
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[27] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[27] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[27] & 0x0001u;
MPFD_Write_Done(); // 145
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[28] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[28] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[28] >> 13) & 0x0001u;
MPFD_Write_Done(); // 146
```

```
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[28] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[28] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[28] >> 10) & 0x0001u;
MPFD_Write_Done(); // 147
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[28] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[28] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[28] >> 7) & 0x0001u;
MPFD_Write_Done(); // 148
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[28] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[28] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[28] >> 4) & 0x0001u;
MPFD_Write_Done(); // 149
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[28] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[28] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[28] >> 1) & 0x0001u;
MPFD_Write_Done(); // 150
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[28] & 0x0001u;
TR2_OUT = (DataArray.v[29] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[29] >> 14) & 0x0001u;
MPFD_Write_Done(); // 151
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[29] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[29] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[29] >> 11) & 0x0001u;
MPFD_Write_Done(); // 152
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
```

```
MPFD_Read_Done();
TR1_OUT = (DataArray.v[29] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[29] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[29] >> 8) & 0x0001u;
MPFD_Write_Done(); // 153
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[29] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[29] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[29] >> 5) & 0x0001u;
MPFD_Write_Done(); // 154
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[29] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[29] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[29] >> 2) & 0x0001u;
MPFD_Write_Done(); // 155
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[29] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[29] & 0x0001u;
TR3_OUT = (DataArray.v[30] >> 15) & 0x0001u;
MPFD_Write_Done(); // 156
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[30] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[30] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[30] >> 12) & 0x0001u;
MPFD_Write_Done(); // 157
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[30] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[30] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[30] >> 9) & 0x0001u;
MPFD_Write_Done(); // 158
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[30] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[30] >> 7) & 0x0001u;
```

```
TR3_OUT = (DataArray.v[30] >> 6) & 0x0001u;
MPFD_Write_Done(); // 159
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[30] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[30] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[30] >> 3) & 0x0001u;
MPFD_Write_Done(); // 160
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[30] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[30] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[30] & 0x0001u;
MPFD_Write_Done(); // 161
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[31] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[31] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[31] >> 13) & 0x0001u;
MPFD_Write_Done(); // 162
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[31] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[31] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[31] >> 10) & 0x0001u;
MPFD_Write_Done(); // 163
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[31] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[31] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[31] >> 7) & 0x0001u;
MPFD_Write_Done(); // 164
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[31] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[31] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[31] >> 4) & 0x0001u;
MPFD_Write_Done(); // 165
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
```

```
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[31] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[31] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[31] >> 1) & 0x0001u;
MPFD_Write_Done(); // 166
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[31] & 0x0001u;
TR2_OUT = (DataArray.v[32] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[32] >> 14) & 0x0001u;
MPFD_Write_Done(); // 167
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[32] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[32] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[32] >> 11) & 0x0001u;
MPFD_Write_Done(); // 168
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[32] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[32] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[32] >> 8) & 0x0001u;
MPFD_Write_Done(); // 169
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[32] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[32] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[32] >> 5) & 0x0001u;
MPFD_Write_Done(); // 170
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[32] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[32] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[32] >> 2) & 0x0001u;
MPFD_Write_Done(); // 171
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
```

```
TR1_OUT = (DataArray.v[32] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[32] & 0x0001u;
TR3_OUT = (DataArray.v[33] >> 15) & 0x0001u;
MPFD_Write_Done(); // 172
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[33] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[33] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[33] >> 12) & 0x0001u;
MPFD_Write_Done(); // 173
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[33] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[33] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[33] >> 9) & 0x0001u;
MPFD_Write_Done(); // 174
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[33] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[33] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[33] >> 6) & 0x0001u;
MPFD_Write_Done(); // 175
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[33] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[33] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[33] >> 3) & 0x0001u;
MPFD_Write_Done(); // 176
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[33] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[33] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[33] & 0x0001u;
MPFD_Write_Done(); // 177
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[34] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[34] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[34] >> 13) & 0x0001u;
```

```
MPFD_Write_Done(); // 178
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[34] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[34] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[34] >> 10) & 0x0001u;
MPFD_Write_Done(); // 179
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[34] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[34] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[34] >> 7) & 0x0001u;
MPFD_Write_Done(); // 180
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[34] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[34] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[34] >> 4) & 0x0001u;
MPFD_Write_Done(); // 181
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[34] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[34] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[34] >> 1) & 0x0001u;
MPFD_Write_Done(); // 182
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[34] & 0x0001u;
TR2_OUT = (DataArray.v[35] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[35] >> 14) & 0x0001u;
MPFD_Write_Done(); // 183
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[35] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[35] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[35] >> 11) & 0x0001u;
MPFD_Write_Done(); // 184
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
```

```
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[35] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[35] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[35] >> 8) & 0x0001u;
MPFD_Write_Done(); // 185
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[35] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[35] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[35] >> 5) & 0x0001u;
MPFD_Write_Done(); // 186
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[35] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[35] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[35] >> 2) & 0x0001u;
MPFD_Write_Done(); // 187
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[35] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[35] & 0x0001u;
TR3_OUT = (DataArray.v[36] >> 15) & 0x0001u;
MPFD_Write_Done(); // 188
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[36] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[36] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[36] >> 12) & 0x0001u;
MPFD_Write_Done(); // 189
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[36] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[36] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[36] >> 9) & 0x0001u;
MPFD_Write_Done(); // 190
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[36] >> 8) & 0x0001u;
```

```
TR2_OUT = (DataArray.v[36] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[36] >> 6) & 0x0001u;
MPFD_Write_Done(); // 191
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[36] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[36] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[36] >> 3) & 0x0001u;
MPFD_Write_Done(); // 192
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[36] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[36] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[36] & 0x0001u;
MPFD_Write_Done(); // 193
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[37] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[37] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[37] >> 13) & 0x0001u;
MPFD_Write_Done(); // 194
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[37] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[37] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[37] >> 10) & 0x0001u;
MPFD_Write_Done(); // 195
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[37] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[37] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[37] >> 7) & 0x0001u;
MPFD_Write_Done(); // 196
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[37] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[37] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[37] >> 4) & 0x0001u;
MPFD_Write_Done(); // 197
```

```
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[37] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[37] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[37] >> 1) & 0x0001u;
MPFD_Write_Done(); // 198
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[37] & 0x0001u;
TR2_OUT = (DataArray.v[38] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[38] >> 14) & 0x0001u;
MPFD_Write_Done(); // 199
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[38] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[38] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[38] >> 11) & 0x0001u;
MPFD_Write_Done(); // 200
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[38] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[38] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[38] >> 8) & 0x0001u;
MPFD_Write_Done(); // 201
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[38] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[38] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[38] >> 5) & 0x0001u;
MPFD_Write_Done(); // 202
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[38] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[38] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[38] >> 2) & 0x0001u;
MPFD_Write_Done(); // 203
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
```

```
MPFD_Read_Done();
TR1_OUT = (DataArray.v[38] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[38] & 0x0001u;
TR3_OUT = (DataArray.v[39] >> 15) & 0x0001u;
MPFD_Write_Done(); // 204
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[39] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[39] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[39] >> 12) & 0x0001u;
MPFD_Write_Done(); // 205
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[39] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[39] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[39] >> 9) & 0x0001u;
MPFD_Write_Done(); // 206
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[39] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[39] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[39] >> 6) & 0x0001u;
MPFD_Write_Done(); // 207
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[39] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[39] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[39] >> 3) & 0x0001u;
MPFD_Write_Done(); // 208
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[39] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[39] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[39] & 0x0001u;
MPFD_Write_Done(); // 209
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[40] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[40] >> 14) & 0x0001u;
```

```
TR3_OUT = (DataArray.v[40] >> 13) & 0x0001u;
MPFD_Write_Done(); // 210
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[40] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[40] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[40] >> 10) & 0x0001u;
MPFD_Write_Done(); // 211
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[40] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[40] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[40] >> 7) & 0x0001u;
MPFD_Write_Done(); // 212
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[40] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[40] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[40] >> 4) & 0x0001u;
MPFD_Write_Done(); // 213
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[40] >> 3) & 0x0001u;
TR2_OUT = (DataArray.v[40] >> 2) & 0x0001u;
TR3_OUT = (DataArray.v[40] >> 1) & 0x0001u;
MPFD_Write_Done(); // 214
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = DataArray.v[40] & 0x0001u;
TR2_OUT = (DataArray.v[41] >> 15) & 0x0001u;
TR3_OUT = (DataArray.v[41] >> 14) & 0x0001u;
MPFD_Write_Done(); // 215
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[41] >> 13) & 0x0001u;
TR2_OUT = (DataArray.v[41] >> 12) & 0x0001u;
TR3_OUT = (DataArray.v[41] >> 11) & 0x0001u;
MPFD_Write_Done(); // 216
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
```

```
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[41] >> 10) & 0x0001u;
TR2_OUT = (DataArray.v[41] >> 9) & 0x0001u;
TR3_OUT = (DataArray.v[41] >> 8) & 0x0001u;
MPFD_Write_Done(); // 217
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[41] >> 7) & 0x0001u;
TR2_OUT = (DataArray.v[41] >> 6) & 0x0001u;
TR3_OUT = (DataArray.v[41] >> 5) & 0x0001u;
MPFD_Write_Done(); // 218
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[41] >> 4) & 0x0001u;
TR2_OUT = (DataArray.v[41] >> 3) & 0x0001u;
TR3_OUT = (DataArray.v[41] >> 2) & 0x0001u;
MPFD_Write_Done(); // 219
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[41] >> 1) & 0x0001u;
TR2_OUT = DataArray.v[41] & 0x0001u;
TR3_OUT = (DataArray.v[42] >> 15) & 0x0001u;
MPFD_Write_Done(); // 220
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[42] >> 14) & 0x0001u;
TR2_OUT = (DataArray.v[42] >> 13) & 0x0001u;
TR3_OUT = (DataArray.v[42] >> 12) & 0x0001u;
MPFD_Write_Done(); // 221
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[42] >> 11) & 0x0001u;
TR2_OUT = (DataArray.v[42] >> 10) & 0x0001u;
TR3_OUT = (DataArray.v[42] >> 9) & 0x0001u;
MPFD_Write_Done(); // 222
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
```

```
TR1_OUT = (DataArray.v[42] >> 8) & 0x0001u;
TR2_OUT = (DataArray.v[42] >> 7) & 0x0001u;
TR3_OUT = (DataArray.v[42] >> 6) & 0x0001u;
MPFD_Write_Done(); // 223
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[42] >> 5) & 0x0001u;
TR2_OUT = (DataArray.v[42] >> 4) & 0x0001u;
TR3_OUT = (DataArray.v[42] >> 3) & 0x0001u;
MPFD_Write_Done(); // 224
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[42] >> 2) & 0x0001u;
TR2_OUT = (DataArray.v[42] >> 1) & 0x0001u;
TR3_OUT = DataArray.v[42] & 0x0001u;
MPFD_Write_Done(); // 225
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[43] >> 15) & 0x0001u;
TR2_OUT = (DataArray.v[43] >> 14) & 0x0001u;
TR3_OUT = (DataArray.v[43] >> 13) & 0x0001u;
MPFD_Write_Done(); // 226
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[43] >> 12) & 0x0001u;
TR2_OUT = (DataArray.v[43] >> 11) & 0x0001u;
TR3_OUT = (DataArray.v[43] >> 10) & 0x0001u;
MPFD_Write_Done(); // 227
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[43] >> 9) & 0x0001u;
TR2_OUT = (DataArray.v[43] >> 8) & 0x0001u;
TR3_OUT = (DataArray.v[43] >> 7) & 0x0001u;
MPFD_Write_Done(); // 228
if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
    return(FALSE);
}
MPFD_Read_Done();
TR1_OUT = (DataArray.v[43] >> 6) & 0x0001u;
TR2_OUT = (DataArray.v[43] >> 5) & 0x0001u;
TR3_OUT = (DataArray.v[43] >> 4) & 0x0001u;
```

```
    MPFD_Write_Done(); // 229
    if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
        return(FALSE);
    }
    MPFD_Read_Done();
    TR1_OUT = (DataArray.v[43] >> 3) & 0x0001u;
    TR2_OUT = (DataArray.v[43] >> 2) & 0x0001u;
    TR3_OUT = (DataArray.v[43] >> 1) & 0x0001u;
    MPFD_Write_Done(); // 230
    if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
        return(FALSE);
    }
    MPFD_Read_Done();
    TR1_OUT = DataArray.v[43] & 0x0001u;
    TR2_OUT = (BoardID >> 3) & 0x0001u;
    TR3_OUT = (BoardID >> 2) & 0x0001u;
    MPFD_Write_Done(); // 231
    if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
        return(FALSE);
    }
    MPFD_Read_Done();
    TR1_OUT = (BoardID >> 1) & 0x0001u;
    TR2_OUT = BoardID & 0x0001u;
    TR3_OUT = VoltageChanging;
    MPFD_Write_Done(); // 232
    if (TW_IN_BOOL || !TRIGGER_IN_BOOL) {
        return(FALSE);
    }
    MPFD_Read_Done();
    TR1_OUT = 0;
    TR2_OUT = 0;
    TR3_OUT = 0;
    MPFD_Write_Done();
    return(TRUE);
}

unsigned char GetSpeedNew(void) {
    return Speed_New;
}

void Init_System(void) {
    unsigned char uc0;

    TickInit();

    // Enable Global High Priority Interrupts
    INTCONbits.GIEH = 1;
    // Enable Global Low Priority Interrupts
    INTCONbits.GIEL = 1;

    TR1_OUT = 0;
```

```
    TR2_OUT = 0;
    TR3_OUT = 0;

    swi2c_stop();

    Speed = 0x00u;
    Speed_New = 0x00u;
    MSV_Res = 0x00u;
    MSV_Res_New = 0x00u;
    Curr_Res = 0x00u;
    Curr_Res_New = 0x00u;
    Temp_Res = 0x00u;
    Temp_Res_New = 0x00u;

    ResetTimer3();

    Board_Identify();

    /********* Find I2C Addresses for ADCs *********/
    FindADCAddr();

    /********* Clear High Voltage *********/
    HV_Setpoint = 0x0000u;
    HV_NewSetpoint = 0x0000u;
    do {
        ResetTimer3();
    } while(SetHV(0x0000u) == 0x00u);
    do {
        ResetTimer3();
    } while (ConfigHVADC(DataRate240SPS, ADCGain1) == 0x00u);

    /********* Set LLD High *********/
    do {
        ResetTimer3();
    } while(SetLLD(0x03FFu) == 0x00u);
    LLD_Setpoint = 0x03FFu;
    LLD_NewSetpoint = 0x03FFu;

    ResetRelays();
    StopTimer3();

    /********* Provide a time delay for system to startup *********/
    // one second time delay
    for (uc0 = 0u; uc0 < 10u; ++uc0) {
        TickWaitSecond();
    }
}

void ResetRelays(void) {
    /*** Set High Voltage to 0 ***/
    HV_Setpoint = 0x0000u;
```

```
    if (SetHV(HV_Setpoint) == 1u) {
        /*** Configure HV ADC ***/
        if (ConfigHVADC(DataRate240SPS, ADCGain1) == 0x01u) {
            /*** Verify HV is off ***/
            do {
                ResetTimer3();
                TickWait240();
                HVData = ReadHVADC();
                HVvOut = ((unsigned short long)(HVData >> 8));
                HVvOut = convertHV(HVvOut,DataRate240SPS);
                HVvOut = convertHVoffset(HVvOut);
            } while (HVvOut > 0x0A3Eu); // Greater than 20 V
            StopTimer3();
            /*** Reset Relays ***/
            RELAY_RESET_TRIS = 0;
            RELAY_RESET_OUT = 1;
            TickWaitSecond();
            RELAY_RESET_OUT = 0;
        }
    }
}

BOOL SyncTimer(void) {
    return SyncTmr;
}

void ResetSyncTimer(void) {
    TMR4 = 0x00u; // Timer4 Register
    SyncTmr = TRUE;
    T4CONbits.TMR4ON = 1; // Timer4 on
}

void MPFD_Write_Done(void) {
    /*** Set Write high and wait for line to read high ***/
    while (SyncTmr && TRIGGER_IN_BOOL);
    WRITE_TRIS = 1;

    while (!WRITE_IN_BOOL && TRIGGER_IN_BOOL);
    READ_TRIS = 0;

    TMR4 = 0x00u; // Timer4 Register
    SyncTmr = TRUE;
    T4CONbits.TMR4ON = 1; // Timer4 on
}

void MPFD_Read_Done(void) {
    /*** Set Read high and wait for line to read high ***/
    while (SyncTmr && TRIGGER_IN_BOOL);
    READ_TRIS = 1;

    while (!READ_IN_BOOL && TRIGGER_IN_BOOL);
```

```
    WRITE_TRIS = 0;

    TMR4 = 0x00u; // Timer4 Register
    SyncTmr = TRUE;
    T4CONbits.TMR4ON = 1; // Timer4 on
}

void Board_Identify(void) {
    BoardID = 0x00u;
    BoardID34_OUT = 0;
    BoardID36_OUT = 1;
    BoardID = BoardID | (BoardID35_IN << 3);
    BoardID36_OUT = 0;
    BoardID38_OUT = 1;
    BoardID = BoardID | (BoardID37_IN << 2);
    BoardID38_OUT = 0;
    BoardID40_OUT = 1;
    BoardID = BoardID | (BoardID39_IN << 1);
    BoardID41_OUT = 0;
    BoardID43_OUT = 1;
    BoardID = BoardID | BoardID42_IN;
    ++BoardID;
}

BOOL IsTMR3CountDownZero(void) {
    if (TMR3CountDown > 0u) {
        return(FALSE);
    } else {
        return(TRUE);
    }
}

BOOL IsTMR3CountDownNotZero(void) {
    if (TMR3CountDown > 0u) {
        return(TRUE);
    } else {
        return(FALSE);
    }
}

void ResetTimer3(void) {
    T3CONbits.TMR3ON = 0; // Stops Timer3
    TMR3H = 0x00u;
    TMR3L = 0x00u; // Set for 1/19 second timeout
    TMR3CountDown = 0x13u; // Trigger is set to 0 once TMR3CountDown reaches 0
    T3CONbits.TMR3ON = 1; // Start Timer3
}

void StopTimer3(void) {
    T3CONbits.TMR3ON = 0; // Stops Timer3
    TMR3CountDown = 0x13u;
```

```
}

void SetMPFDArrayBit(unsigned char MPFDByte, unsigned char MPFDBit, unsigned int MPFDData ...
    ) {
    DataArray.v[MPFDByte] |= MPFDData << MPFDBit;
}
```

End main.c

B.6.2 main.h

This is the header file for the program main.c (Section §B.6.1) of the daughter board's microcontroller.

Begin main.h

```
#ifndef __main_H
#define __main_H

#include <P18F8722.H>

//#ifdef __18F8722_H
/************ Configuration bits ************/
/* Oscillator Selection */
#pragma config OSC = HS

/* Fail-Safe Clock Monitor Enable bit */
// Section 25.4
#pragma config FCMEN = OFF

/* Internal/External Oscillator Switchover bit */
#pragma config IESO = OFF

/* Power-up Timer Enable bit */
// Section 4.5
#pragma config PWRT = OFF

/* Brown-out Reset Enable bits */
// Section 4.4, page 54
#pragma config BOREN = ON

/* Brown-out Voltage bits */
// Section 28.1, page 380
#pragma config BORV = 3

/* Watchdog Timer Enable bit */
// Section 25.2, page 314
#pragma config WDT = OFF, WDTPS = 32768
```

```
/* Processor Data Memory Mode Select bits */
// Section 5.1.1, page 65
#pragma config MODE = MC // Microcontroller Mode

/* Address Bus Width Select bits */
#pragma config ADDRBW = ADDR8BIT

/* Data Bus Width Select bit */
#pragma config DATABW = DATA16BIT

/* External Bus Data Wait Enable bit */
#pragma config WAIT = OFF

/* MCLR Pin Enable bit */
#pragma config MCLRE = ON

/* Low-Power Timer1 Oscillator Enable bit */
#pragma config LPT1OSC = OFF

/* ECCP MUX bit */
#pragma config ECCPMX = PORTH

/* CCP2 MUX bit */
#pragma config CCP2MX = PORTC

/* Stack Full/Underflow Reset Enable bit */
#pragma config STVREN = ON

/* Single-Supply ICSP Enable bit */
#pragma config LVP = OFF

/* Boot Block Size Select bits */
#pragma config BBSIZ = BB2K

/* Extended Instruction Set Enable bit */
#pragma config XINST = ON

/* Background Debugger Enable bit */
#pragma config DEBUG = ON

/* Code Protection bits */
#pragma config CP0 = OFF
#pragma config CP1 = OFF
#pragma config CP2 = OFF
#pragma config CP3 = OFF
#pragma config CP4 = OFF
#pragma config CP5 = OFF
#pragma config CP6 = OFF
#pragma config CP7 = OFF
#pragma config CPB = OFF
#pragma config CPD = OFF
```

```
#pragma config WRT0 = OFF
#pragma config WRT1 = OFF
#pragma config WRT2 = OFF
#pragma config WRT3 = OFF
#pragma config WRT4 = OFF
#pragma config WRT5 = OFF
#pragma config WRT6 = OFF
#pragma config WRT7 = OFF
#pragma config WRTB = OFF
#pragma config WRTC = OFF
#pragma config WRTD = OFF
#pragma config EBTR0 = OFF
#pragma config EBTR1 = OFF
#pragma config EBTR2 = OFF
#pragma config EBTR3 = OFF
#pragma config EBTR4 = OFF
#pragma config EBTR5 = OFF
#pragma config EBTR6 = OFF
#pragma config EBTR7 = OFF
#pragma config EBTRB = OFF


/************ Function Prototypes ************/
void HighVector(void);
void LowVector(void);
void MyHighISR(void);
void MyLowISR(void);
void main(void);
void Init_System(void);
void ChangeSpeed(void);
void AdjustLLD(void);
void ChangeTempRes(void);
void AdjustHighVoltage(void);
void LoadNewHighVoltage(void);
void ResetRelays(void);
void NormalDataCollection(void);
unsigned char GetSpeedNew(void);


#endif // __main_H
```

End main.h

B.6.3 config.h

This is the header file for the program config.c (Section §B.6.4) of the daughter board's microcontroller.

Begin config.h

```
#ifndef __config_H
#define __config_H

#include <P18F8722.H>
#include "Compiler.h"

//#define LCD_DEBUG
// PORT – READ IN
// LAT – WRITE OUT
// TRIS IN/OUT
#define IN 1
#define OUT 0

#define MSV_4L_TRIS (TRISAbits.TRISA0) // IN/OUT
#define MSV_4L_IN ((unsigned char)PORTAbits.RA0)
#define MSV_4L_OUT (LATAbits.LATA0)
#define CURR_4L_TRIS (TRISAbits.TRISA1) // IN/OUT
#define CURR_4L_IN ((unsigned char)PORTAbits.RA1)
#define CURR_4L_OUT (LATAbits.LATA1)
#define MSV_4B_TRIS (TRISAbits.TRISA2) // IN/OUT
#define MSV_4B_IN ((unsigned char)PORTAbits.RA2)
#define MSV_4B_OUT (LATAbits.LATA2)
#define CURR_4B_TRIS (TRISAbits.TRISA3) // IN/OUT
#define CURR_4B_IN ((unsigned char)PORTAbits.RA3)
#define CURR_4B_OUT (LATAbits.LATA3)
#define BoardID34_TRIS (TRISAbits.TRISA4) // OUT
#define BoardID34_OUT (LATAbits.LATA4)
#define PULSE_LBH4_TRIS (TRISAbits.TRISA5) // IN/OUT
#define PULSE_LBH4_IN ((unsigned char)PORTAbits.RA4)
#define NotUsedA6_TRIS (TRISAbits.TRISA6) // OUT
#define NotUsedA6_OUT (LATAbits.LATA6)

#define READ_TRIS (TRISBbits.TRISB0) // IN/OUT (SET_READY)
#define READ_OUT (LATBbits.LATB0)
#define READ_IN ((unsigned char)PORTBbits.RB0)
#define READ_IN_BOOL (PORTBbits.RB0)
#define NotUsedB1_TRIS (TRISBbits.TRISB1) // IN (SET_DONE)
#define NotUsedB1_IN ((unsigned char)PORTBbits.RB1)
#define NotUsedJ3_TRIS (TRISJbits.TRISJ3) // IN (READY)
#define NotUsedJ3_IN ((unsigned char)PORTJbits.RJ3)
#define WRITE_TRIS (TRISBbits.TRISB2) // IN/OUT (DONE)
#define WRITE_OUT (LATBbits.LATB2)
#define WRITE_IN ((unsigned char)PORTBbits.RB2)
#define WRITE_IN_BOOL (PORTBbits.RB2)
#define TRIGGER_TRIS (TRISBbits.TRISB3) // IN/OUT
#define TRIGGER_OUT (LATBbits.LATB3)
#define TRIGGER_IN ((unsigned char)PORTBbits.RB3)
#define TRIGGER_IN_BOOL (PORTBbits.RB3)
#define WD_TRIS (TRISBbits.TRISB4) // OUT
#define WD_OUT (LATBbits.LATB4)
```

```
#define NotUsedB5_TRIS (TRISBbits.TRISB5) // OUT
#define NotUsedB5_OUT (LATBbits.LATB5)

#define BoardID36_TRIS (TRISCbits.TRISC0) // OUT
#define BoardID36_OUT (LATCbits.LATC0)
#define BoardID35_TRIS (TRISCbits.TRISC1) // IN
#define BoardID35_IN ((unsigned char)PORTCbits.RC1)
#define BoardID43_TRIS (TRISCbits.TRISC2) // OUT
#define BoardID43_OUT (LATCbits.LATC2)
#define HV_LLD_SCL_TRIS (TRISCbits.TRISC3) // IN/OUT
#define HV_LLD_SCL_IN ((unsigned char)PORTCbits.RC3)
#define HV_LLD_SCL_OUT (LATCbits.LATC3)
#define HV_LLD_SDA_TRIS (TRISCbits.TRISC4) // IN/OUT
#define HV_LLD_SDA_IN ((unsigned char)PORTCbits.RC4)
#define HV_LLD_SDA_OUT (LATCbits.LATC4)
#define STRING_STATUS_TRIS (TRISCbits.TRISC5) // IN
#define STRING_STATUS_IN ((unsigned char)PORTCbits.RC5)
#define BoardID37_TRIS (TRISCbits.TRISC6) // IN
#define BoardID37_IN ((unsigned char)PORTCbits.RC6)
#define BoardID38_TRIS (TRISCbits.TRISC7) // OUT
#define BoardID38_OUT (LATCbits.LATC7)

#define CURR_2H_TRIS (TRISDbits.TRISD0) // IN/OUT
#define CURR_2H_IN ((unsigned char)PORTDbits.RD0)
#define CURR_2H_OUT (LATDbits.LATD0)
#define MSV_1L_TRIS (TRISDbits.TRISD1) // IN/OUT
#define MSV_1L_IN ((unsigned char)PORTDbits.RD1)
#define MSV_1L_OUT (LATDbits.LATD1)
#define CURR_1L_TRIS (TRISDbits.TRISD2) // IN/OUT
#define CURR_1L_IN ((unsigned char)PORTDbits.RD2)
#define CURR_1L_OUT (LATDbits.LATD2)
#define MSV_1B_TRIS (TRISDbits.TRISD3) // IN/OUT
#define MSV_1B_IN ((unsigned char)PORTDbits.RD3)
#define MSV_1B_OUT (LATDbits.LATD3)
#define CURR_1B_TRIS (TRISDbits.TRISD4) // IN/OUT
#define CURR_1B_IN ((unsigned char)PORTDbits.RD4)
#define CURR_1B_OUT (LATDbits.LATD4)
#define MSV_1H_TRIS (TRISDbits.TRISD5) // IN/OUT
#define MSV_1H_IN ((unsigned char)PORTDbits.RD5)
#define MSV_1H_OUT (LATDbits.LATD5)
#define CURR_1H_TRIS (TRISDbits.TRISD6) // IN/OUT
#define CURR_1H_IN ((unsigned char)PORTDbits.RD6)
#define CURR_1H_OUT (LATDbits.LATD6)

#define PULSE_LBH1_TRIS (TRISEbits.TRISE0) // IN
#define PULSE_LBH1_IN ((unsigned char)PORTEbits.RE0)
#define MSV_3L_TRIS (TRISEbits.TRISE1) // IN/OUT
#define MSV_3L_IN ((unsigned char)PORTEbits.RE1)
#define MSV_3L_OUT (LATEbits.LATE1)
#define CURR_3H_TRIS (TRISEbits.TRISE2) // IN/OUT
#define CURR_3H_IN ((unsigned char)PORTEbits.RE2)
```

```
#define CURR_3H_OUT (LATEbits.LATE2)
#define MSV_2L_TRIS (TRISEbits.TRISE3) // IN/OUT
#define MSV_2L_IN ((unsigned char)PORTEbits.RE3)
#define MSV_2L_OUT (LATEbits.LATE3)
#define CURR_2L_TRIS (TRISEbits.TRISE4) // IN/OUT
#define CURR_2L_IN ((unsigned char)PORTEbits.RE4)
#define CURR_2L_OUT (LATEbits.LATE4)
#define MSV_2B_TRIS (TRISEbits.TRISE5) // IN/OUT
#define MSV_2B_IN ((unsigned char)PORTEbits.RE5)
#define MSV_2B_OUT (LATEbits.LATE5)
#define CURR_2B_TRIS (TRISEbits.TRISE6) // IN/OUT
#define CURR_2B_IN ((unsigned char)PORTEbits.RE6)
#define CURR_2B_OUT (LATEbits.LATE6)
#define MSV_2H_TRIS (TRISEbits.TRISE7) // IN/OUT
#define MSV_2H_IN ((unsigned char)PORTEbits.RE7)
#define MSV_2H_OUT (LATEbits.LATE7)

#define MSV_4H_TRIS (TRISFbits.TRISF0) // IN/OUT
#define MSV_4H_IN ((unsigned char)PORTFbits.RF0)
#define MSV_4H_OUT (LATFbits.LATF0)
#define CURR_4H_TRIS (TRISFbits.TRISF1) // IN/OUT
#define CURR_4H_IN ((unsigned char)PORTFbits.RF1)
#define CURR_4H_OUT (LATFbits.LATF1)
#define TEMP_4_TRIS (TRISFbits.TRISF2) // IN/OUT
#define TEMP_4_IN ((unsigned char)PORTFbits.RF2)
#define TEMP_4_OUT (LATFbits.LATF2)
#define TEMP_SCL_TRIS (TRISFbits.TRISF3) // OUT
#define TEMP_SCL_OUT (LATFbits.LATF3)
#define CURR_SCL_TRIS (TRISFbits.TRISF4) // OUT
#define CURR_SCL_OUT (LATFbits.LATF4)
#define MSV_SCL_TRIS (TRISFbits.TRISF5) // OUT
#define MSV_SCL_OUT (LATFbits.LATF5)
#define PULSE_CLEAR_TRIS (TRISFbits.TRISF6) // OUT
#define PULSE_CLEAR_OUT (LATFbits.LATF6)
#define PULSE_RCLK_TRIS (TRISFbits.TRISF7) // OUT
#define PULSE_RCLK_OUT (LATFbits.LATF7)

#define PULSE_LBH2_TRIS (TRISGbits.TRISG0) // IN
#define PULSE_LBH2_IN ((unsigned char)PORTGbits.RG0)
#define PULSE_LBH3_TRIS (TRISGbits.TRISG1) // IN
#define PULSE_LBH3_IN ((unsigned char)PORTGbits.RG1)
#define PULSE_SCLK_TRIS (TRISGbits.TRISG2) // OUT
#define PULSE_SCLK_OUT (LATGbits.LATG2)
#define PULSE_LOAD_TRIS (TRISGbits.TRISG3) // OUT
#define PULSE_LOAD_OUT (LATGbits.LATG3)
#define PULSE_ENABLE_TRIS (TRISGbits.TRISG4) // OUT
#define PULSE_ENABLE_OUT (LATGbits.LATG4)

#define MSV_3H_TRIS (TRISHbits.TRISH0) // IN/OUT
#define MSV_3H_IN ((unsigned char)PORTHbits.RH0)
#define MSV_3H_OUT (LATHbits.LATH0)
```

```
#define CURR_3B_TRIS (TRISHbits.TRISH1) // IN/OUT
#define CURR_3B_IN ((unsigned char)PORTHbits.RH1)
#define CURR_3B_OUT (LATHbits.LATH1)
#define MSV_3B_TRIS (TRISHbits.TRISH2) // IN/OUT
#define MSV_3B_IN ((unsigned char)PORTHbits.RH2)
#define MSV_3B_OUT (LATHbits.LATH2)
#define CURR_3L_TRIS (TRISHbits.TRISH3) // IN/OUT
#define CURR_3L_IN ((unsigned char)PORTHbits.RH3)
#define CURR_3L_OUT (LATHbits.LATH3)
#define RELAY_RESET_TRIS (TRISHbits.TRISH4) // OUT
#define RELAY_RESET_OUT (LATHbits.LATH4)
#define TEMP_1_TRIS (TRISHbits.TRISH5) // IN/OUT
#define TEMP_1_IN ((unsigned char)PORTHbits.RH5)
#define TEMP_1_OUT (LATHbits.LATH5)
#define TEMP_2_TRIS (TRISHbits.TRISH6) // IN/OUT
#define TEMP_2_IN ((unsigned char)PORTHbits.RH6)
#define TEMP_2_OUT (LATHbits.LATH6)
#define TEMP_3_TRIS (TRISHbits.TRISH7) // IN/OUT
#define TEMP_3_IN ((unsigned char)PORTHbits.RH7)
#define TEMP_3_OUT (LATHbits.LATH7)

#define TR1_TRIS (TRISDbits.TRISD7) // OUT
#define TR1_OUT (LATDbits.LATD7)
#define TR1_IN ((unsigned char)PORTDbits.RD7)
#define TR2_TRIS (TRISJbits.TRISJ0) // OUT
#define TR2_OUT (LATJbits.LATJ0)
#define TR2_IN ((unsigned char)PORTJbits.RJ0)
#define TR3_TRIS (TRISJbits.TRISJ1) // OUT
#define TR3_OUT (LATJbits.LATJ1)
#define TR3_IN ((unsigned char)PORTJbits.RJ1)
#define TW_TRIS (TRISJbits.TRISJ2) // IN
#define TW_IN ((unsigned char)PORTJbits.RJ2)
#define TW_IN_BOOL (PORTJbits.RJ2)
#define BoardID39_TRIS (TRISJbits.TRISJ4) // IN
#define BoardID39_IN ((unsigned char)PORTJbits.RJ4)
#define BoardID40_TRIS (TRISJbits.TRISJ5) // OUT
#define BoardID40_OUT (LATJbits.LATJ5)
#define BoardID41_TRIS (TRISJbits.TRISJ6) // OUT
#define BoardID41_OUT (LATJbits.LATJ6)
#define BoardID42_TRIS (TRISJbits.TRISJ7) // IN
#define BoardID42_IN ((unsigned char)PORTJbits.RJ7)

void CONFIG_Init(void);
void Init_ReadWrite(void);

#endif
```

End config.h

B.6.4 config.c

This is the config program for the daughter board's microcontroller.

Begin config.c

```
#include "config.h"

//static volatile ReadWrite_States ReadWriteState;
//static volatile unsigned char TRIGGER_SET;
//static volatile unsigned char WRITE_SET;
//static volatile unsigned char READ_SET;

//extern unsigned char getdebug1(void);
//extern unsigned char getdebug2(void);

void CONFIG_Init(void) {

RELAY_RESET_OUT = 0;
RELAY_RESET_TRIS = OUT;

READ_OUT = 0;
READ_TRIS = 0;
NotUsedB1_TRIS = IN;
NotUsedJ3_TRIS = IN;
WRITE_OUT = 0;
WRITE_TRIS = 0;
TRIGGER_OUT = 0;
TRIGGER_TRIS = 0;
WD_TRIS = OUT;
WD_OUT = 0;
TR1_OUT = 0;
TR1_TRIS = OUT;
TR2_OUT = 0;
TR2_TRIS = OUT;
TR3_OUT = 0;
TR3_TRIS = OUT;
TW_TRIS = IN;

STRING_STATUS_TRIS = IN;

BoardID35_TRIS = IN;
BoardID37_TRIS = IN;
BoardID39_TRIS = IN;
BoardID42_TRIS = IN;
BoardID34_OUT = 1;
BoardID34_TRIS = OUT;
BoardID36_OUT = 1;
BoardID36_TRIS = OUT;
BoardID38_OUT = 1;
BoardID38_TRIS = OUT;
```

```
BoardID40_OUT = 1;
BoardID40_TRIS = OUT;
BoardID41_OUT = 1;
BoardID41_TRIS = OUT;
BoardID43_OUT = 1;
BoardID43_TRIS = OUT;

NotUsedA6_OUT = 0;
NotUsedA6_TRIS = OUT;
NotUsedB5_OUT = 0;
NotUsedB5_TRIS = OUT;

HV_LLD_SCL_TRIS = OUT;
HV_LLD_SCL_OUT = 0;
HV_LLD_SDA_TRIS = OUT;
HV_LLD_SDA_OUT = 0;
TEMP_SCL_TRIS = OUT;
TEMP_SCL_OUT = 1;
CURR_SCL_TRIS = OUT;
CURR_SCL_OUT = 1;
MSV_SCL_TRIS = OUT;
MSV_SCL_OUT = 1;

PULSE_SCLK_OUT = 0;
PULSE_SCLK_TRIS = OUT;
PULSE_LOAD_OUT = 0;
PULSE_LOAD_TRIS = OUT;
PULSE_CLEAR_OUT = 0;
PULSE_CLEAR_TRIS = OUT;
PULSE_RCLK_OUT = 0;
PULSE_RCLK_TRIS = OUT;
PULSE_ENABLE_OUT = 0;
PULSE_ENABLE_TRIS = OUT;

MSV_1H_TRIS = IN;
MSV_1H_OUT = 0;
CURR_1H_TRIS = IN;
CURR_1H_OUT = 0;
MSV_1L_TRIS = IN;
MSV_1L_OUT = 0;
CURR_1L_TRIS = IN;
CURR_1L_OUT = 0;
MSV_1B_TRIS = IN;
MSV_1B_OUT = 0;
CURR_1B_TRIS = IN;
CURR_1B_OUT = 0;
PULSE_LBH1_TRIS = IN;
TEMP_1_TRIS = IN;
TEMP_1_OUT = 0;

MSV_2H_TRIS = IN;
```

```
MSV_2H_OUT = 0;
CURR_2H_TRIS = IN;
CURR_2H_OUT = 0;
MSV_2L_TRIS = IN;
MSV_2L_OUT = 0;
CURR_2L_TRIS = IN;
CURR_2L_OUT = 0;
MSV_2B_TRIS = IN;
MSV_2B_OUT = 0;
CURR_2B_TRIS = IN;
CURR_2B_OUT = 0;
PULSE_LBH2_TRIS = IN;
TEMP_2_TRIS = IN;
TEMP_2_OUT = 0;

MSV_3H_TRIS = IN;
MSV_3H_OUT = 0;
CURR_3H_TRIS = IN;
CURR_3H_OUT = 0;
MSV_3L_TRIS = IN;
MSV_3L_OUT = 0;
CURR_3L_TRIS = IN;
CURR_3L_OUT = 0;
MSV_3B_TRIS = IN;
MSV_3B_OUT = 0;
CURR_3B_TRIS = IN;
CURR_3B_OUT = 0;
PULSE_LBH3_TRIS = IN;
TEMP_3_TRIS = IN;
TEMP_3_OUT = 0;

MSV_4H_TRIS = IN;
MSV_4H_OUT = 0;
CURR_4H_TRIS = IN;
CURR_4H_OUT = 0;
MSV_4L_TRIS = IN;
MSV_4L_OUT = 0;
CURR_4L_TRIS = IN;
CURR_4L_OUT = 0;
MSV_4B_TRIS = IN;
MSV_4B_OUT = 0;
CURR_4B_TRIS = IN;
CURR_4B_OUT = 0;
PULSE_LBH4_TRIS = IN;
TEMP_4_TRIS = IN;
TEMP_4_OUT = 0;

ADCON0 = 0;
ADCON1 = 0b00001111;
}
```

```
void Init_ReadWrite(void) {
    READ_OUT = 0;
    READ_TRIS = 0;
    WRITE_OUT = 0;
    WRITE_TRIS = 0;
    TRIGGER_OUT = 0;
    TRIGGER_TRIS = 0;

    RCONbits.IPEN = 1; // Enale priority levels on interrups
    INTCON2bits.RBPU = 1; // Disable PORTB pull−ups
    INTCON2bits.RBIP = 0; // RB Port Change Interrupt Priority bit − not used
    INTCONbits.RBIF = 0; // Clear RB Port Change Interrupt Flag
    INTCONbits.RBIE = 0; // Disable RB Port Change Interrupt

    // READ
    INTCONbits.INT0IF = 0; // Clear Interrupt Flag
    INTCON2bits.INTEDG0 = 1; // Interrupt on rising edge
    //INTCONbits.INT0IE = 1; // Enable Interrupt
    INTCONbits.INT0IE = 0; // Disable Interrupt

    // WRITE
    INTCON2bits.INTEDG2 = 1; // Interrupt on rising edge
    INTCON3bits.INT2IP = 1; // High priority
    INTCON3bits.INT2IF = 0; // Clear Interrupt Flag
    //INTCON3bits.INT2IE = 1; // Enable Interrupt
    INTCON3bits.INT2IE = 0; // Disable Interrupt

    // TRIGGER
    INTCON2bits.INTEDG3 = 1; // Interrupt on rising edge
    INTCON2bits.INT3IP = 1; // High priority
    INTCON3bits.INT3IF = 0; // Clear Interrupt Flag
    //INTCON3bits.INT3IE = 1; // Enable Interrupt
    INTCON3bits.INT3IE = 0; // Disable Interrupt

    // INT1 − Not Used
    INTCON2bits.INTEDG1 = 1; // Rising Edge
    INTCON3bits.INT1IE = 0; // Disable
    INTCON3bits.INT1IF = 0; // Clear Flag
    INTCON3bits.INT1IP = 0; // Low Priority

}
```

End config.c

B.6.5 globals.h

This file actually does nothing for the program but shows how the data structure for all of the detectors is organized for transfer to the mother board.

Begin globals.h

```
#ifndef __globals_H
#define __globals_H

/************ Global Variables ************/
// 1 – String Connected
// 2 – Pulse Overflow
// t – Temp Resolution
// h – HV speed
// 3 – Pulse Underflow
// 4 – Speed
// 5 – HV New
// 6 – Temp New
// 7 – MSV New
// 8 – Curr New
// 9 – HV Data
// A – Temp Data
// B – Pulse Data
// C – MSV Data
// D – Curr Data
// 5432109876543210
// DataArray[0] = 0b000000000023tthh
// DataArray[1] = 0b1445666677777777
// DataArray[2] = 0b7777888888888888
// DataArray[3] = 0b9999999999999999
// DataArray[4] = 0bAAAAAAAAAAAAAAAA – TEMP1
// DataArray[5] = 0bAAAAAAAAAAAAAAAA – TEMP2
// DataArray[6] = 0bAAAAAAAAAAAAAAAA – TEMP3
// DataArray[7] = 0bAAAAAAAAAAAAAAAA – TEMP4
// DataArray[8] = 0bBBBBBBBBBBBBBBBB – Pulse 1L
// DataArray[9] = 0bBBBBBBBBBBBBBBBB – Pulse 1B
// DataArray[10] = 0bBBBBBBBBBBBBBBBB – Pulse 1H
// DataArray[11] = 0bBBBBBBBBBBBBBBBB – Pulse 2L
// DataArray[12] = 0bBBBBBBBBBBBBBBBB – Pulse 2B
// DataArray[13] = 0bBBBBBBBBBBBBBBBB – Pulse 2H
// DataArray[14] = 0bBBBBBBBBBBBBBBBB – Pulse 3L
// DataArray[15] = 0bBBBBBBBBBBBBBBBB – Pulse 3B
// DataArray[16] = 0bBBBBBBBBBBBBBBBB – Pulse 3H
// DataArray[17] = 0bBBBBBBBBBBBBBBBB – Pulse 4L
// DataArray[18] = 0bBBBBBBBBBBBBBBBB – Pulse 4B
// DataArray[19] = 0bBBBBBBBBBBBBBBBB – Pulse 4H
// DataArray[20] = 0bCCCCCCCCCCCCCCCC – MSV 1L
// DataArray[21] = 0bCCCCCCCCCCCCCCCC – MSV 1B
// DataArray[22] = 0bCCCCCCCCCCCCCCCC – MSV 1H
// DataArray[23] = 0bCCCCCCCCCCCCCCCC – MSV 2L
// DataArray[24] = 0bCCCCCCCCCCCCCCCC – MSV 2B
// DataArray[25] = 0bCCCCCCCCCCCCCCCC – MSV 2H
// DataArray[26] = 0bCCCCCCCCCCCCCCCC – MSV 3L
// DataArray[27] = 0bCCCCCCCCCCCCCCCC – MSV 3B
// DataArray[28] = 0bCCCCCCCCCCCCCCCC – MSV 3H
// DataArray[29] = 0bCCCCCCCCCCCCCCCC – MSV 4L
```

```
// DataArray[30] = 0bCCCCCCCCCCCCCCCC – MSV 4B
// DataArray[31] = 0bCCCCCCCCCCCCCCCC – MSV 4H
// DataArray[32] = 0bDDDDDDDDDDDDDDDD – Curr 1L
// DataArray[33] = 0bDDDDDDDDDDDDDDDD – Curr 1B
// DataArray[34] = 0bDDDDDDDDDDDDDDDD – Curr 1H
// DataArray[35] = 0bDDDDDDDDDDDDDDDD – Curr 2L
// DataArray[36] = 0bDDDDDDDDDDDDDDDD – Curr 2B
// DataArray[37] = 0bDDDDDDDDDDDDDDDD – Curr 2H
// DataArray[38] = 0bDDDDDDDDDDDDDDDD – Curr 3L
// DataArray[39] = 0bDDDDDDDDDDDDDDDD – Curr 3B
// DataArray[40] = 0bDDDDDDDDDDDDDDDD – Curr 3H
// DataArray[41] = 0bDDDDDDDDDDDDDDDD – Curr 4L
// DataArray[42] = 0bDDDDDDDDDDDDDDDD – Curr 4B
// DataArray[43] = 0bDDDDDDDDDDDDDDDD – Curr 4H

#endif //__globals_H
```

End globals.h

B.6.6 GenericTypeDefs.h

This file of type definitions are used to define several uncommon variable types for use throughout the programs of the daughter board's microcontroller.

Begin GenericTypeDefs.h

```
/********************************************************************
 *
 * Generic Type Definitions
 *
 ********************************************************************
 * FileName: GenericTypeDefs.h
 * Dependencies: None
 * Processor: PIC18, PIC24, dsPIC, PIC32
 * Compiler: Microchip C18, C30, C32
 * Company: Microchip Technology, Inc.
 *
 * Software License Agreement
 *
 * The software supplied herewith by Microchip Technology Incorporated
 * (the "Company") is intended and supplied to you, the Company's
 * customer, for use solely and exclusively with products manufactured
 * by the Company.
 *
 * The software is owned by the Company and/or its supplier, and is
 * protected under applicable copyright laws. All rights are reserved.
 * Any use in violation of the foregoing restrictions may subject the
 * user to criminal sanctions under applicable laws, as well as to
 * civil liability for the breach of the terms and conditions of this
```

```
 * license.
 *
 * THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
 * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
 * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
 * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 ********************************************************************
 * File Description:
 *
 * Change History:
 * Rev Date Description
 * 1.1 09/11/06 Add base signed types
 * 1.2 02/28/07 Add QWORD, LONGLONG, QWORD_VAL
 * 1.3 02/06/08 Add def's for PIC32
 * 1.4 08/08/08 Remove LSB/MSB Macros, adopted by Peripheral lib
 * 1.5 08/14/08 Simplify file header
 ********************************************************************/

#ifndef __GENERIC_TYPE_DEFS_H_
#define __GENERIC_TYPE_DEFS_H_

typedef enum _BOOL { FALSE = 0, TRUE } BOOL; // Undefined size

#ifndef NULL
#define NULL 0//((void *)0)
#endif

#define PUBLIC // Function attributes
#define PROTECTED
#define PRIVATE static

typedef unsigned char BYTE; // 8-bit unsigned
typedef unsigned short int WORD; // 16-bit unsigned
typedef unsigned long DWORD; // 32-bit unsigned
typedef unsigned long long QWORD; // 64-bit unsigned
typedef signed char CHAR; // 8-bit signed
typedef signed short int SHORT; // 16-bit signed
typedef signed long LONG; // 32-bit signed
typedef signed long long LONGLONG; // 64-bit signed

/* Alternate definitions */
typedef void VOID;

typedef char CHAR8;
typedef unsigned char UCHAR8;

/* Processor & Compiler independent, size specific definitions */
// To Do: We need to verify the sizes on each compiler. These
```

```
// may be compiler specific, we should either move them
// to "compiler.h" or #ifdef them for compiler type.
typedef signed int INT;
typedef signed char INT8;
typedef signed short int INT16;
typedef signed long int INT32;
typedef signed long long INT64;

typedef unsigned int UINT;
typedef unsigned char UINT8;
typedef unsigned short int UINT16;
typedef unsigned long int UINT32; // other name for 32-bit integer
typedef unsigned long long UINT64;

typedef union _BYTE_VAL
{
    BYTE Val;
    struct
    {
        unsigned char b0:1;
        unsigned char b1:1;
        unsigned char b2:1;
        unsigned char b3:1;
        unsigned char b4:1;
        unsigned char b5:1;
        unsigned char b6:1;
        unsigned char b7:1;
    } bits;
} BYTE_VAL, BYTE_BITS;

typedef union _WORD_VAL
{
    WORD Val;
    BYTE v[2];
    struct
    {
        BYTE LB;
        BYTE HB;
    } byte;
    struct
    {
        unsigned char b0:1;
        unsigned char b1:1;
        unsigned char b2:1;
        unsigned char b3:1;
        unsigned char b4:1;
        unsigned char b5:1;
        unsigned char b6:1;
        unsigned char b7:1;
        unsigned char b8:1;
        unsigned char b9:1;
```

```
            unsigned char b10:1;
            unsigned char b11:1;
            unsigned char b12:1;
            unsigned char b13:1;
            unsigned char b14:1;
            unsigned char b15:1;
        } bits;
} WORD_VAL, WORD_BITS;

typedef union _DWORD_VAL
{
        DWORD Val;
        WORD w[2];
        BYTE v[4];
        struct
        {
            WORD LW;
            WORD HW;
        } word;
        struct
        {
            BYTE LB;
            BYTE HB;
            BYTE UB;
            BYTE MB;
        } byte;
        struct
        {
            WORD_VAL low;
            WORD_VAL high;
        }wordUnion;
        struct
        {
            unsigned char b0:1;
            unsigned char b1:1;
            unsigned char b2:1;
            unsigned char b3:1;
            unsigned char b4:1;
            unsigned char b5:1;
            unsigned char b6:1;
            unsigned char b7:1;
            unsigned char b8:1;
            unsigned char b9:1;
            unsigned char b10:1;
            unsigned char b11:1;
            unsigned char b12:1;
            unsigned char b13:1;
            unsigned char b14:1;
            unsigned char b15:1;
            unsigned char b16:1;
            unsigned char b17:1;
```

```
            unsigned char b18:1;
            unsigned char b19:1;
            unsigned char b20:1;
            unsigned char b21:1;
            unsigned char b22:1;
            unsigned char b23:1;
            unsigned char b24:1;
            unsigned char b25:1;
            unsigned char b26:1;
            unsigned char b27:1;
            unsigned char b28:1;
            unsigned char b29:1;
            unsigned char b30:1;
            unsigned char b31:1;
        } bits;
} DWORD_VAL;

typedef union _QWORD_VAL
{
        QWORD Val;
        DWORD d[2];
        WORD w[4];
        BYTE v[8];
        struct
        {
            DWORD LD;
            DWORD HD;
        } dword;
        struct
        {
            WORD LW;
            WORD HW;
            WORD UW;
            WORD MW;
        } word;
        struct
        {
            unsigned char b0:1;
            unsigned char b1:1;
            unsigned char b2:1;
            unsigned char b3:1;
            unsigned char b4:1;
            unsigned char b5:1;
            unsigned char b6:1;
            unsigned char b7:1;
            unsigned char b8:1;
            unsigned char b9:1;
            unsigned char b10:1;
            unsigned char b11:1;
            unsigned char b12:1;
            unsigned char b13:1;
```

```
    unsigned char b14:1;
    unsigned char b15:1;
    unsigned char b16:1;
    unsigned char b17:1;
    unsigned char b18:1;
    unsigned char b19:1;
    unsigned char b20:1;
    unsigned char b21:1;
    unsigned char b22:1;
    unsigned char b23:1;
    unsigned char b24:1;
    unsigned char b25:1;
    unsigned char b26:1;
    unsigned char b27:1;
    unsigned char b28:1;
    unsigned char b29:1;
    unsigned char b30:1;
    unsigned char b31:1;
    unsigned char b32:1;
    unsigned char b33:1;
    unsigned char b34:1;
    unsigned char b35:1;
    unsigned char b36:1;
    unsigned char b37:1;
    unsigned char b38:1;
    unsigned char b39:1;
    unsigned char b40:1;
    unsigned char b41:1;
    unsigned char b42:1;
    unsigned char b43:1;
    unsigned char b44:1;
    unsigned char b45:1;
    unsigned char b46:1;
    unsigned char b47:1;
    unsigned char b48:1;
    unsigned char b49:1;
    unsigned char b50:1;
    unsigned char b51:1;
    unsigned char b52:1;
    unsigned char b53:1;
    unsigned char b54:1;
    unsigned char b55:1;
    unsigned char b56:1;
    unsigned char b57:1;
    unsigned char b58:1;
    unsigned char b59:1;
    unsigned char b60:1;
    unsigned char b61:1;
    unsigned char b62:1;
    unsigned char b63:1;
} bits;
```

```
} QWORD_VAL;

#endif //__GENERIC_TYPE_DEFS_H_
```

End GenericTypeDefs.h

B.6.7 Tick.h

This is the header file for the program Tick.c (Section §B.6.8) for the daughter board's microcontroller.

Begin Tick.h

```
#ifndef __TICK_H
#define __TICK_H

#include "GenericTypeDefs.h"

// All TICKS are stored as 64−bit unsigned integers
typedef DWORD TICK;

#define TICKS_PER_SECOND (5000000ul)
#define TICKS_PER_100k (52ul)
#define TICKS_PER_QUARTER (13ul)
#define TICKS_PER_240 (20834ul)

#define TICKS_PER_2QUARTER (TICKS_PER_QUARTER∗2ul)
#define TICKS_PER_3QUARTER (TICKS_PER_QUARTER∗3ul)
#define TICKS_PER_4QUARTER (TICKS_PER_QUARTER∗4ul)
#define TickDiffSecond (0xFFFFFFFFul − TICKS_PER_SECOND)
#define TickDiff100k (0xFFFFFFFFul − TICKS_PER_100k)
#define TickDiff240 (0xFFFFFFFFul − TICKS_PER_240)
#define TickDiffQuarter (0xFFFFFFFFul − TICKS_PER_QUARTER)
#define TickDiff2Quarter (0xFFFFFFFFul − TICKS_PER_2QUARTER)
#define TickDiff3Quarter (0xFFFFFFFFul − TICKS_PER_3QUARTER)
#define TickDiff4Quarter (0xFFFFFFFFul − TICKS_PER_4QUARTER)

void TickInit(void);
void TickUpdate(void);
DWORD TickGet(void);
void TickWaitSecond(void);
void TickWait240(void);
void TickWait100k(void);
void TickWait1Quarter(void);
void TickWait2Quarter(void);
void TickWait3Quarter(void);
void TickWait4Quarter(void);

#endif // __TICK_H
```

End Tick.h

B.6.8 Tick.c

This is the Tick.c program that controls the timing for various functions of the daughter board's microcontroller.

Begin Tick.c

```
#include "Tick.h"
#include "GenericTypeDefs.h"
#include <P18F8722.H>

extern BOOL IsTMR3CountDownZero(void);
extern BOOL IsTMR3CountDownNotZero(void);

static volatile WORD OverflowTicks;
static DWORD_VAL InternalTicks;
static DWORD Timeout2Q, Timeout3Q, Timeout4Q;
static unsigned char Timeover2Q, Timeover3Q, Timeover4Q;

void TickInit(void) {
    InternalTicks.Val = 0ul;
    OverflowTicks = 0u;
    // Timer0 On/Off Control bit
    // – 1 Enables Timer0
    // – 0 Stops Timer0
    T0CONbits.TMR0ON = 0;
    // Timer0 8–bit/16–bit Control bit
    // – 1 Timer0 is configured as an 8–bit timer/counter
    // – 0 Timer0 is configured as a 16–bit timer/counter
    T0CONbits.T08BIT = 0;
    // Timer0 Clock Source Select bit
    // – 1 Transition on T0CKI pin,
    // – 0 Internal instruction cycle clock (CLKO)
    T0CONbits.T0CS = 0;
    // Timer0 Source Edge Select bit
    // – 1 high–to–low
    // – 0 low–to–high
    T0CONbits.T0SE = 0;
    // Timer0 Prescaler Assignment bit
    // – 1 Timer0 prescaler is NOT assigned
    // – 0 Timer0 prescaler is assigned
    T0CONbits.PSA = 0;
    // Timer0 Prescaler Select bits
    T0CONbits.T0PS0 = 0;
    T0CONbits.T0PS1 = 0;
    T0CONbits.T0PS2 = 0;

    INTCON2bits.TMR0IP = 0; // TMR0 Overflow Interrupt Low Priority
```

```
    INTCONbits.TMR0IF = 0; // Clear Interrupt Flag
    INTCONbits.TMR0IE = 1; // Interrupt Enable

    // TMR0H must be set before TMR0L when in 16 bit mode
    TMR0H = 0;
    TMR0L = 0;

    // Timer0 On/Off Control bit
    // – 1 Enables Timer0
    // – 0 Stops Timer0
    T0CONbits.TMR0ON = 1;

    // TIMER 3
    PIE2bits.TMR3IE = 0; // Disable interrupt
    PIR2bits.TMR3IF = 0; // Clear interrupt flag
    IPR2bits.TMR3IP = 1; // High priority interrupt
    T3CONbits.RD16 = 0; // 8 bit read/write
    T3CONbits.T3CKPS1 = 1;
    T3CONbits.T3CKPS1 = 1; // 1:8 Prescale value
    T3CONbits.TMR3CS = 0; // Use Timer1 oscillator
    T3CONbits.T3SYNC = 1; // Do not sync
    T3CONbits.TMR3ON = 0; // Stops Timer3
    PIE2bits.TMR3IE = 1; // Enable interrupt

    // TIMER 4
    PIE3bits.TMR4IE = 0; // Disable interrupt
    IPR3bits.TMR4IP = 1; // High priority interrupt
    PIR3bits.TMR4IF = 0; // Clear interrupt flag
    T4CONbits.T4OUTPS3 = 0;
    T4CONbits.T4OUTPS2 = 0;
    T4CONbits.T4OUTPS1 = 1;
    T4CONbits.T4OUTPS0 = 0; // 1:2 Postscalar
    T4CONbits.T4CKPS1 = 0;
    T4CONbits.T4CKPS0 = 1; // 1:4 Prescalar
    PR4 = 0x3Bu; // Period Register
    TMR4 = 0x00u; // Timer4 Register
    T4CONbits.TMR4ON = 0; // Timer4 off
    PIE3bits.TMR4IE = 1; // Enable interrupt
}

void TickUpdate(void) {
    // Increment internal high tick counter
    ++OverflowTicks;
    //Reset
    INTCONbits.TMR0IF = 0;
}

DWORD TickGet(void) {
    // TMR0L must be read before TMR0H when in 16 bit mode
    INTCONbits.TMR0IE = 1; // Interrupt Enable
    Nop();
```

```
    INTCONbits.TMR0IE = 0; // Interrupt Disable
    InternalTicks.v[0] = TMR0L;
    InternalTicks.v[1] = TMR0H;
    InternalTicks.w[1] = OverflowTicks;
    INTCONbits.TMR0IE = 1; // Interrupt Enable
    return InternalTicks.Val;
}

void TickWaitSecond(void) {
    DWORD Timeout, T2;

    Timeout = TickGet();
    if (TickDiffSecond <= Timeout) {
        Timeout = TICKS_PER_SECOND - (0xFFFFFFFFul - Timeout);
        while (TickGet() > TickDiffSecond);
    } else {
        Timeout = Timeout + TICKS_PER_SECOND;
    }
    while (TickGet() < Timeout);
}

void TickWait240(void) {
    DWORD Timeout;

    Timeout = TickGet();
    if (TickDiff240 <= Timeout) {
        Timeout = TICKS_PER_240 - (0xFFFFFFFFul - Timeout);
        while (TickGet() > TickDiff240 && IsTMR3CountDownNotZero());
    } else {
        Timeout = Timeout + TICKS_PER_240;
    }
    while (TickGet() < Timeout && IsTMR3CountDownNotZero());
}

void TickWait100k(void) {
    DWORD Timeout;

    Timeout = TickGet();
    if (TickDiff100k <= Timeout) {
        Timeout = TICKS_PER_100k - (0xFFFFFFFFul - Timeout);
        while (TickGet() > TickDiff100k);
    } else {
        Timeout = Timeout + TICKS_PER_100k;
    }
    while (TickGet() < Timeout);
}

void TickWait1Quarter(void) {
    DWORD Timeout;

    Timeout = TickGet();
```

```
    if (TickDiffQuarter <= Timeout) {
        Timeout4Q = TICKS_PER_4QUARTER - (0xFFFFFFFFul - Timeout);
        Timeover4Q = 1u;
        Timeout3Q = TICKS_PER_3QUARTER - (0xFFFFFFFFul - Timeout);
        Timeover3Q = 1u;
        Timeout2Q = TICKS_PER_2QUARTER - (0xFFFFFFFFul - Timeout);
        Timeover2Q = 1u;
        Timeout = TICKS_PER_QUARTER - (0xFFFFFFFFul - Timeout);
        while (TickGet() > TickDiffQuarter && IsTMR3CountDownNotZero());
    } else {
        if (TickDiff2Quarter <= Timeout) {
            Timeout4Q = TICKS_PER_4QUARTER - (0xFFFFFFFFul - Timeout);
            Timeover4Q = 1u;
            Timeout3Q = TICKS_PER_3QUARTER - (0xFFFFFFFFul - Timeout);
            Timeover3Q = 1u;
            Timeout2Q = TICKS_PER_2QUARTER - (0xFFFFFFFFul - Timeout);
            Timeover2Q = 1u;
        } else {
            if (TickDiff3Quarter <= Timeout) {
                Timeout4Q = TICKS_PER_4QUARTER - (0xFFFFFFFFul - Timeout);
                Timeover4Q = 1u;
                Timeout3Q = TICKS_PER_3QUARTER - (0xFFFFFFFFul - Timeout);
                Timeover3Q = 1u;
            } else {
                if (TickDiff4Quarter <= Timeout) {
                    Timeout4Q = TICKS_PER_4QUARTER - (0xFFFFFFFFul - Timeout);
                    Timeover4Q = 1u;
                } else {
                    Timeout4Q = Timeout + TICKS_PER_4QUARTER;
                    Timeover4Q = 0u;
                }
                Timeout3Q = Timeout + TICKS_PER_3QUARTER;
                Timeover3Q = 0u;
            }
            Timeout2Q = Timeout + TICKS_PER_2QUARTER;
            Timeover2Q = 0u;
        }
        Timeout = Timeout + TICKS_PER_QUARTER;
    }
    while (TickGet() < Timeout && IsTMR3CountDownNotZero());
}

void TickWait2Quarter(void) {
    if (Timeover2Q == 1u) {
        while (TickGet() > TickDiff2Quarter && IsTMR3CountDownNotZero());
    }
    while (TickGet() < Timeout2Q && IsTMR3CountDownNotZero());
}

void TickWait3Quarter(void) {
    if (Timeover3Q == 1u) {
```

```
        while (TickGet() > TickDiff3Quarter && IsTMR3CountDownNotZero());
    }
    while (TickGet() < Timeout3Q && IsTMR3CountDownNotZero());
}

void TickWait4Quarter(void) {
    if (Timeover4Q == 1u) {
        while (TickGet() > TickDiff4Quarter && IsTMR3CountDownNotZero());
    }
    while (TickGet() < Timeout4Q && IsTMR3CountDownNotZero());
}
```

End Tick.c

B.6.9 BitBangADC.h

This is the header file for the program BitBangADC.c (Section §B.6.10) for the daughter board's microcontroller.

Begin BitBangADC.h

```
#ifndef __BitBangADC_H
#define __BitBangADC_H

#include "config.h"
#include "globals.h"
#include "HVLLD.h"

void ADC_config(void);
void ADC_getdata(void);
void FindADCAddr(void);
unsigned char GetHVADCAddr(void);

#endif // __BitBangADC_H
```

End BitBangADC.h

B.6.10 BitBangADC.c

This is the BitBangADC.c program that controls the data collection from all of the I^2C ADCs to the daughter board's microcontroller.

Begin BitBangADC.c

```
#include "config.h"
#include "BitBangADC.h"
```

```
#include "globals.h"
#include "GenericTypeDefs.h"
#include "Tick.h"

static BYTE_VAL Addr_MSV_1H, Addr_MSV_1L, Addr_MSV_1B;
static BYTE_VAL Addr_MSV_2H, Addr_MSV_2L, Addr_MSV_2B;
static BYTE_VAL Addr_MSV_3H, Addr_MSV_3L, Addr_MSV_3B;
static BYTE_VAL Addr_MSV_4H, Addr_MSV_4L, Addr_MSV_4B;
static BYTE_VAL Addr_CURR_1H, Addr_CURR_1L, Addr_CURR_1B;
static BYTE_VAL Addr_CURR_2H, Addr_CURR_2L, Addr_CURR_2B;
static BYTE_VAL Addr_CURR_3H, Addr_CURR_3L, Addr_CURR_3B;
static BYTE_VAL Addr_CURR_4H, Addr_CURR_4L, Addr_CURR_4B;
static BYTE_VAL Addr_TEMP_1, Addr_TEMP_2, Addr_TEMP_3, Addr_TEMP_4;
static BYTE_VAL Addr_HV;

extern unsigned char GetSpeedNew(void);
extern void SetMPFDArrayBit(unsigned char MPFDByte, unsigned char MPFDBit, unsigned int   ...
      MPFDData);

//static far MPFDArray LDataArray;

void BBi2c_start(void);
void BBi2c_stop(void);
void BBi2c_write(void);
void BBi2c_read(unsigned char k);
void BBi2c_readNew(void);
void BBi2c_readSpeed1(void);
void BBi2c_readSpeed0(void);
void BBi2c_readGain1(void);
void BBi2c_readGain0(void);
void BBi2c_Addr(unsigned char ReadWrite);

void BBi2c_start(void) {
    /*** SDA1 ***/
    MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
        CURR_1L_TRIS = 1;
        CURR_1B_TRIS = 1;
        CURR_2H_TRIS = 1;
        CURR_2L_TRIS = 1;
```

```
    CURR_2B_TRIS = 1;
    CURR_3H_TRIS = 1;
    CURR_3L_TRIS = 1;
    CURR_3B_TRIS = 1;
    CURR_4H_TRIS = 1;
    CURR_4L_TRIS = 1;
    CURR_4B_TRIS = 1;
    TEMP_1_TRIS = 1;
    TEMP_2_TRIS = 1;
    TEMP_3_TRIS = 1;
    TEMP_4_TRIS = 1;
HV_LLD_SDA_TRIS = 1;

    TickWait1Quarter();

/*** SCL1 ***/
LATF |= 0b00111000;
HV_LLD_SCL_TRIS = 1;

    TickWait2Quarter();

/*** SDA0 ***/
MSV_1H_TRIS = 0;
    MSV_1L_TRIS = 0;
    MSV_1B_TRIS = 0;
    MSV_2H_TRIS = 0;
    MSV_2L_TRIS = 0;
    MSV_2B_TRIS = 0;
    MSV_3H_TRIS = 0;
    MSV_3L_TRIS = 0;
    MSV_3B_TRIS = 0;
    MSV_4H_TRIS = 0;
    MSV_4L_TRIS = 0;
    MSV_4B_TRIS = 0;
    CURR_1H_TRIS = 0;
    CURR_1L_TRIS = 0;
    CURR_1B_TRIS = 0;
    CURR_2H_TRIS = 0;
    CURR_2L_TRIS = 0;
    CURR_2B_TRIS = 0;
    CURR_3H_TRIS = 0;
    CURR_3L_TRIS = 0;
    CURR_3B_TRIS = 0;
    CURR_4H_TRIS = 0;
    CURR_4L_TRIS = 0;
    CURR_4B_TRIS = 0;
    TEMP_1_TRIS = 0;
    TEMP_2_TRIS = 0;
    TEMP_3_TRIS = 0;
    TEMP_4_TRIS = 0;
HV_LLD_SDA_TRIS = 0;
```

```
        TickWait3Quarter();

    /*** SCL0 ***/
    LATF &= 0b11000111;
    HV_LLD_SCL_TRIS = 0;

        TickWait4Quarter();
}

void BBi2c_stop(void) {
    /*** SDA0 ***/
    MSV_1H_TRIS = 0;
        MSV_1L_TRIS = 0;
        MSV_1B_TRIS = 0;
        MSV_2H_TRIS = 0;
        MSV_2L_TRIS = 0;
        MSV_2B_TRIS = 0;
        MSV_3H_TRIS = 0;
        MSV_3L_TRIS = 0;
        MSV_3B_TRIS = 0;
        MSV_4H_TRIS = 0;
        MSV_4L_TRIS = 0;
        MSV_4B_TRIS = 0;
        CURR_1H_TRIS = 0;
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

        TickWait2Quarter();

    /*** SDA1 ***/
    MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
```

```
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
        CURR_1L_TRIS = 1;
        CURR_1B_TRIS = 1;
        CURR_2H_TRIS = 1;
        CURR_2L_TRIS = 1;
        CURR_2B_TRIS = 1;
        CURR_3H_TRIS = 1;
        CURR_3L_TRIS = 1;
        CURR_3B_TRIS = 1;
        CURR_4H_TRIS = 1;
        CURR_4L_TRIS = 1;
        CURR_4B_TRIS = 1;
        TEMP_1_TRIS = 1;
        TEMP_2_TRIS = 1;
        TEMP_3_TRIS = 1;
        TEMP_4_TRIS = 1;
    HV_LLD_SDA_TRIS = 1;

        TickWait3Quarter();
}

void BBi2c_write(void) {
        TickWait1Quarter();

    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

        TickWait3Quarter();

    /*** SCL0 ***/
    LATF &= 0b11000111;
    HV_LLD_SCL_TRIS = 0;

        TickWait4Quarter();
}

void ADC_config(void) {
        BBi2c_start();
        // Write Address – 0b10010010
    BBi2c_Addr(0x00u);
```

```
    // Write Config
    /*** ST/!DRDY – Start Conversion ***/
/*** SDA1 ***/
MSV_1H_TRIS = 1;
    MSV_1L_TRIS = 1;
    MSV_1B_TRIS = 1;
    MSV_2H_TRIS = 1;
    MSV_2L_TRIS = 1;
    MSV_2B_TRIS = 1;
    MSV_3H_TRIS = 1;
    MSV_3L_TRIS = 1;
    MSV_3B_TRIS = 1;
    MSV_4H_TRIS = 1;
    MSV_4L_TRIS = 1;
    MSV_4B_TRIS = 1;
    CURR_1H_TRIS = 1;
    CURR_1L_TRIS = 1;
    CURR_1B_TRIS = 1;
    CURR_2H_TRIS = 1;
    CURR_2L_TRIS = 1;
    CURR_2B_TRIS = 1;
    CURR_3H_TRIS = 1;
    CURR_3L_TRIS = 1;
    CURR_3B_TRIS = 1;
    CURR_4H_TRIS = 1;
    CURR_4L_TRIS = 1;
    CURR_4B_TRIS = 1;
    TEMP_1_TRIS = 1;
    TEMP_2_TRIS = 1;
    TEMP_3_TRIS = 1;
    TEMP_4_TRIS = 1;
HV_LLD_SDA_TRIS = 1;

    BBi2c_write();

    /*** 6 ***/
/*** SDA0 ***/
MSV_1H_TRIS = 0;
    MSV_1L_TRIS = 0;
    MSV_1B_TRIS = 0;
    MSV_2H_TRIS = 0;
    MSV_2L_TRIS = 0;
    MSV_2B_TRIS = 0;
    MSV_3H_TRIS = 0;
    MSV_3L_TRIS = 0;
    MSV_3B_TRIS = 0;
    MSV_4H_TRIS = 0;
    MSV_4L_TRIS = 0;
    MSV_4B_TRIS = 0;
    CURR_1H_TRIS = 0;
```

```
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

        BBi2c_write();

        /*** 5 ***/
    /*** SDA0 ***/
    MSV_1H_TRIS = 0;
        MSV_1L_TRIS = 0;
        MSV_1B_TRIS = 0;
        MSV_2H_TRIS = 0;
        MSV_2L_TRIS = 0;
        MSV_2B_TRIS = 0;
        MSV_3H_TRIS = 0;
        MSV_3L_TRIS = 0;
        MSV_3B_TRIS = 0;
        MSV_4H_TRIS = 0;
        MSV_4L_TRIS = 0;
        MSV_4B_TRIS = 0;
        CURR_1H_TRIS = 0;
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;
```

```
        BBi2c_write();

        /*** Single Conversion Mode ***/
    /*** SDA1 ***/
    MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
        CURR_1L_TRIS = 1;
        CURR_1B_TRIS = 1;
        CURR_2H_TRIS = 1;
        CURR_2L_TRIS = 1;
        CURR_2B_TRIS = 1;
        CURR_3H_TRIS = 1;
        CURR_3L_TRIS = 1;
        CURR_3B_TRIS = 1;
        CURR_4H_TRIS = 1;
        CURR_4L_TRIS = 1;
        CURR_4B_TRIS = 1;
        TEMP_1_TRIS = 1;
        TEMP_2_TRIS = 1;
        TEMP_3_TRIS = 1;
        TEMP_4_TRIS = 1;
    HV_LLD_SDA_TRIS = 1;

        BBi2c_write();
        if ((GetSpeedNew() & 0b00001000u) == 0b00001000u) {
                /*** DR1 ***/
        /*** SDA1 ***/
        MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
```

```
CURR_1L_TRIS = 1;
CURR_1B_TRIS = 1;
CURR_2H_TRIS = 1;
CURR_2L_TRIS = 1;
CURR_2B_TRIS = 1;
CURR_3H_TRIS = 1;
CURR_3L_TRIS = 1;
CURR_3B_TRIS = 1;
CURR_4H_TRIS = 1;
CURR_4L_TRIS = 1;
CURR_4B_TRIS = 1;
TEMP_1_TRIS = 1;
TEMP_2_TRIS = 1;
TEMP_3_TRIS = 1;
TEMP_4_TRIS = 1;
HV_LLD_SDA_TRIS = 1;
} else {
        /*** DR1 ***/
/*** SDA0 ***/
MSV_1H_TRIS = 0;
MSV_1L_TRIS = 0;
MSV_1B_TRIS = 0;
MSV_2H_TRIS = 0;
MSV_2L_TRIS = 0;
MSV_2B_TRIS = 0;
MSV_3H_TRIS = 0;
MSV_3L_TRIS = 0;
MSV_3B_TRIS = 0;
MSV_4H_TRIS = 0;
MSV_4L_TRIS = 0;
MSV_4B_TRIS = 0;
CURR_1H_TRIS = 0;
CURR_1L_TRIS = 0;
CURR_1B_TRIS = 0;
CURR_2H_TRIS = 0;
CURR_2L_TRIS = 0;
CURR_2B_TRIS = 0;
CURR_3H_TRIS = 0;
CURR_3L_TRIS = 0;
CURR_3B_TRIS = 0;
CURR_4H_TRIS = 0;
CURR_4L_TRIS = 0;
CURR_4B_TRIS = 0;
TEMP_1_TRIS = 0;
TEMP_2_TRIS = 0;
TEMP_3_TRIS = 0;
TEMP_4_TRIS = 0;
HV_LLD_SDA_TRIS = 0;
}

BBi2c_write();
```

```
if ((GetSpeedNew() & 0b00000100u) == 0b00000100u) {
        /*** DR0 ***/
/*** SDA1 ***/
MSV_1H_TRIS = 1;
MSV_1L_TRIS = 1;
MSV_1B_TRIS = 1;
MSV_2H_TRIS = 1;
MSV_2L_TRIS = 1;
MSV_2B_TRIS = 1;
MSV_3H_TRIS = 1;
MSV_3L_TRIS = 1;
MSV_3B_TRIS = 1;
MSV_4H_TRIS = 1;
MSV_4L_TRIS = 1;
MSV_4B_TRIS = 1;
CURR_1H_TRIS = 1;
CURR_1L_TRIS = 1;
CURR_1B_TRIS = 1;
CURR_2H_TRIS = 1;
CURR_2L_TRIS = 1;
CURR_2B_TRIS = 1;
CURR_3H_TRIS = 1;
CURR_3L_TRIS = 1;
CURR_3B_TRIS = 1;
CURR_4H_TRIS = 1;
CURR_4L_TRIS = 1;
CURR_4B_TRIS = 1;
TEMP_1_TRIS = 1;
TEMP_2_TRIS = 1;
TEMP_3_TRIS = 1;
TEMP_4_TRIS = 1;
HV_LLD_SDA_TRIS = 1;
} else {
        /*** DR0 ***/
/*** SDA0 ***/
MSV_1H_TRIS = 0;
MSV_1L_TRIS = 0;
MSV_1B_TRIS = 0;
MSV_2H_TRIS = 0;
MSV_2L_TRIS = 0;
MSV_2B_TRIS = 0;
MSV_3H_TRIS = 0;
MSV_3L_TRIS = 0;
MSV_3B_TRIS = 0;
MSV_4H_TRIS = 0;
MSV_4L_TRIS = 0;
MSV_4B_TRIS = 0;
CURR_1H_TRIS = 0;
CURR_1L_TRIS = 0;
CURR_1B_TRIS = 0;
```

```
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
        HV_LLD_SDA_TRIS = 0;
        }

        BBi2c_write();

        /*** PGA1 ***/
    /*** SDA0 ***/
    MSV_1H_TRIS = 0;
        MSV_1L_TRIS = 0;
        MSV_1B_TRIS = 0;
        MSV_2H_TRIS = 0;
        MSV_2L_TRIS = 0;
        MSV_2B_TRIS = 0;
        MSV_3H_TRIS = 0;
        MSV_3L_TRIS = 0;
        MSV_3B_TRIS = 0;
        MSV_4H_TRIS = 0;
        MSV_4L_TRIS = 0;
        MSV_4B_TRIS = 0;
        CURR_1H_TRIS = 0;
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

        BBi2c_write();
```

```
    /*** PGA0 ***/
/*** SDA0 ***/
MSV_1H_TRIS = 0;
    MSV_1L_TRIS = 0;
    MSV_1B_TRIS = 0;
    MSV_2H_TRIS = 0;
    MSV_2L_TRIS = 0;
    MSV_2B_TRIS = 0;
    MSV_3H_TRIS = 0;
    MSV_3L_TRIS = 0;
    MSV_3B_TRIS = 0;
    MSV_4H_TRIS = 0;
    MSV_4L_TRIS = 0;
    MSV_4B_TRIS = 0;
    CURR_1H_TRIS = 0;
    CURR_1L_TRIS = 0;
    CURR_1B_TRIS = 0;
    CURR_2H_TRIS = 0;
    CURR_2L_TRIS = 0;
    CURR_2B_TRIS = 0;
    CURR_3H_TRIS = 0;
    CURR_3L_TRIS = 0;
    CURR_3B_TRIS = 0;
    CURR_4H_TRIS = 0;
    CURR_4L_TRIS = 0;
    CURR_4B_TRIS = 0;
    TEMP_1_TRIS = 0;
    TEMP_2_TRIS = 0;
    TEMP_3_TRIS = 0;
    TEMP_4_TRIS = 0;
HV_LLD_SDA_TRIS = 0;

    BBi2c_write();

    /*** Ack ***/
/*** SDA0 ***/
MSV_1H_TRIS = 0;
    MSV_1L_TRIS = 0;
    MSV_1B_TRIS = 0;
    MSV_2H_TRIS = 0;
    MSV_2L_TRIS = 0;
    MSV_2B_TRIS = 0;
    MSV_3H_TRIS = 0;
    MSV_3L_TRIS = 0;
    MSV_3B_TRIS = 0;
    MSV_4H_TRIS = 0;
    MSV_4L_TRIS = 0;
    MSV_4B_TRIS = 0;
    CURR_1H_TRIS = 0;
    CURR_1L_TRIS = 0;
```

```
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

        BBi2c_write();
        BBi2c_stop();
}

void ADC_getdata(void) {
        BBi2c_start();
        // Read Address – 0b10010011
    BBi2c_Addr(0x01u);

        /*** Set for input ***/
    /*** SDA1 ***/
    MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
        CURR_1L_TRIS = 1;
        CURR_1B_TRIS = 1;
        CURR_2H_TRIS = 1;
        CURR_2L_TRIS = 1;
        CURR_2B_TRIS = 1;
        CURR_3H_TRIS = 1;
        CURR_3L_TRIS = 1;
        CURR_3B_TRIS = 1;
        CURR_4H_TRIS = 1;
        CURR_4L_TRIS = 1;
        CURR_4B_TRIS = 1;
```

```
        TEMP_1_TRIS = 1;
        TEMP_2_TRIS = 1;
        TEMP_3_TRIS = 1;
        TEMP_4_TRIS = 1;
    HV_LLD_SDA_TRIS = 1;

        BBi2c_read(15u);
        BBi2c_read(14u);
        BBi2c_read(13u);
        BBi2c_read(12u);
        BBi2c_read(11u);
        BBi2c_read(10u);
        BBi2c_read(9u);
        BBi2c_read(8u);

        /*** Ack ***/
    /*** SDA0 ***/
    MSV_1H_TRIS = 0;
        MSV_1L_TRIS = 0;
        MSV_1B_TRIS = 0;
        MSV_2H_TRIS = 0;
        MSV_2L_TRIS = 0;
        MSV_2B_TRIS = 0;
        MSV_3H_TRIS = 0;
        MSV_3L_TRIS = 0;
        MSV_3B_TRIS = 0;
        MSV_4H_TRIS = 0;
        MSV_4L_TRIS = 0;
        MSV_4B_TRIS = 0;
        CURR_1H_TRIS = 0;
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

        BBi2c_write();

        /*** Set for input ***/
    /*** SDA1 ***/
```

```
MSV_1H_TRIS = 1;
    MSV_1L_TRIS = 1;
    MSV_1B_TRIS = 1;
    MSV_2H_TRIS = 1;
    MSV_2L_TRIS = 1;
    MSV_2B_TRIS = 1;
    MSV_3H_TRIS = 1;
    MSV_3L_TRIS = 1;
    MSV_3B_TRIS = 1;
    MSV_4H_TRIS = 1;
    MSV_4L_TRIS = 1;
    MSV_4B_TRIS = 1;
    CURR_1H_TRIS = 1;
    CURR_1L_TRIS = 1;
    CURR_1B_TRIS = 1;
    CURR_2H_TRIS = 1;
    CURR_2L_TRIS = 1;
    CURR_2B_TRIS = 1;
    CURR_3H_TRIS = 1;
    CURR_3L_TRIS = 1;
    CURR_3B_TRIS = 1;
    CURR_4H_TRIS = 1;
    CURR_4L_TRIS = 1;
    CURR_4B_TRIS = 1;
    TEMP_1_TRIS = 1;
    TEMP_2_TRIS = 1;
    TEMP_3_TRIS = 1;
    TEMP_4_TRIS = 1;
HV_LLD_SDA_TRIS = 1;

    BBi2c_read(7u);
    BBi2c_read(6u);
    BBi2c_read(5u);
    BBi2c_read(4u);
    BBi2c_read(3u);
    BBi2c_read(2u);
    BBi2c_read(1u);
    BBi2c_read(0u);

    /*** Ack ***/
/*** SDA0 ***/
MSV_1H_TRIS = 0;
    MSV_1L_TRIS = 0;
    MSV_1B_TRIS = 0;
    MSV_2H_TRIS = 0;
    MSV_2L_TRIS = 0;
    MSV_2B_TRIS = 0;
    MSV_3H_TRIS = 0;
    MSV_3L_TRIS = 0;
    MSV_3B_TRIS = 0;
    MSV_4H_TRIS = 0;
```

```
        MSV_4L_TRIS = 0;
        MSV_4B_TRIS = 0;
        CURR_1H_TRIS = 0;
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

        BBi2c_write();

        /*** Set for input ***/
    /*** SDA1 ***/
    MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
        CURR_1L_TRIS = 1;
        CURR_1B_TRIS = 1;
        CURR_2H_TRIS = 1;
        CURR_2L_TRIS = 1;
        CURR_2B_TRIS = 1;
        CURR_3H_TRIS = 1;
        CURR_3L_TRIS = 1;
        CURR_3B_TRIS = 1;
        CURR_4H_TRIS = 1;
        CURR_4L_TRIS = 1;
        CURR_4B_TRIS = 1;
        TEMP_1_TRIS = 1;
        TEMP_2_TRIS = 1;
        TEMP_3_TRIS = 1;
```

```
        TEMP_4_TRIS = 1;
    HV_LLD_SDA_TRIS = 1;

        BBi2c_readNew(); // ST/!DRDY
        BBi2c_write(); // 0
        BBi2c_write(); // 0
        BBi2c_write(); // SC
        BBi2c_readSpeed1(); // DR1
        BBi2c_readSpeed0(); // DR0
        BBi2c_readGain1(); // PGA1
        BBi2c_readGain0(); // PGA0

        /*** Ack ***/
    /*** SDA0 ***/
    MSV_1H_TRIS = 0;
        MSV_1L_TRIS = 0;
        MSV_1B_TRIS = 0;
        MSV_2H_TRIS = 0;
        MSV_2L_TRIS = 0;
        MSV_2B_TRIS = 0;
        MSV_3H_TRIS = 0;
        MSV_3L_TRIS = 0;
        MSV_3B_TRIS = 0;
        MSV_4H_TRIS = 0;
        MSV_4L_TRIS = 0;
        MSV_4B_TRIS = 0;
        CURR_1H_TRIS = 0;
        CURR_1L_TRIS = 0;
        CURR_1B_TRIS = 0;
        CURR_2H_TRIS = 0;
        CURR_2L_TRIS = 0;
        CURR_2B_TRIS = 0;
        CURR_3H_TRIS = 0;
        CURR_3L_TRIS = 0;
        CURR_3B_TRIS = 0;
        CURR_4H_TRIS = 0;
        CURR_4L_TRIS = 0;
        CURR_4B_TRIS = 0;
        TEMP_1_TRIS = 0;
        TEMP_2_TRIS = 0;
        TEMP_3_TRIS = 0;
        TEMP_4_TRIS = 0;
    HV_LLD_SDA_TRIS = 0;

        BBi2c_write();

        BBi2c_stop();
}

void BBi2c_readSpeed1(void) {
        TickWait1Quarter();
```

```
    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

    SetMPFDArrayBit(0u, 1u, ((unsigned int) HV_LLD_SDA_IN));
        TickWait3Quarter();

    /*** SCL0 ***/
    LATF &= 0b11000111;
    HV_LLD_SCL_TRIS = 0;

        TickWait4Quarter();
}

void BBi2c_readSpeed0(void) {
        TickWait1Quarter();

    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

    SetMPFDArrayBit(0u, 0u, ((unsigned int) HV_LLD_SDA_IN));
        TickWait3Quarter();

    /*** SCL0 ***/
    LATF &= 0b11000111;
    HV_LLD_SCL_TRIS = 0;

        TickWait4Quarter();
}

void BBi2c_readGain1(void) {
        TickWait1Quarter();

    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

    SetMPFDArrayBit(0u, 3u, ((unsigned int) TEMP_1_IN));
        TickWait3Quarter();

    /*** SCL0 ***/
    LATF &= 0b11000111;
    HV_LLD_SCL_TRIS = 0;

        TickWait4Quarter();
}

void BBi2c_readGain0(void) {
        TickWait1Quarter();
```

```
    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

    SetMPFDArrayBit(0u, 2u, ((unsigned int) TEMP_1_IN));
        TickWait3Quarter();

    /*** SCL0 ***/
    LATF &= 0b11000111;
    HV_LLD_SCL_TRIS = 0;

        TickWait4Quarter();
}

void BBi2c_readNew(void) {
        TickWait1Quarter();

    /*** SCL1 ***/
    LATF |= 0b00111000;
    HV_LLD_SCL_TRIS = 1;

        // 1 – String Connected
        // 2 – Pulse Overflow
        // t – Temp Resolution
        // h – HV speed
        // 3 – Pulse Underflow
        // 4 – Speed
        // 5 – HV New
        // 6 – Temp New
        // 7 – MSV New
        // 8 – Curr New

        // 5432109876543210
        // LDataArray.v[0] = 0b000000000012tthh
        // LDataArray.v[1] = 0b3445666677777777
        // LDataArray.v[2] = 0b7777888888888888

    SetMPFDArrayBit(1u, 12u, ((unsigned int) HV_LLD_SDA_IN));
    SetMPFDArrayBit(1u, 11u, ((unsigned int) TEMP_1_IN));
    SetMPFDArrayBit(1u, 10u, ((unsigned int) TEMP_2_IN));
    SetMPFDArrayBit(1u, 9u, ((unsigned int) TEMP_3_IN));
    SetMPFDArrayBit(1u, 8u, ((unsigned int) TEMP_4_IN));

    SetMPFDArrayBit(1u, 7u, ((unsigned int) MSV_1L_IN));
    SetMPFDArrayBit(1u, 6u, ((unsigned int) MSV_1B_IN));
    SetMPFDArrayBit(1u, 5u, ((unsigned int) MSV_1H_IN));
    SetMPFDArrayBit(1u, 4u, ((unsigned int) MSV_2L_IN));
    SetMPFDArrayBit(1u, 3u, ((unsigned int) MSV_2B_IN));
    SetMPFDArrayBit(1u, 2u, ((unsigned int) MSV_2H_IN));
    SetMPFDArrayBit(1u, 1u, ((unsigned int) MSV_3L_IN));
```

```
        SetMPFDArrayBit(1u, 0u, ((unsigned int) MSV_3B_IN));
        SetMPFDArrayBit(2u, 15u, ((unsigned int) MSV_3H_IN));
        SetMPFDArrayBit(2u, 14u, ((unsigned int) MSV_4L_IN));
        SetMPFDArrayBit(2u, 13u, ((unsigned int) MSV_4B_IN));
        SetMPFDArrayBit(2u, 12u, ((unsigned int) MSV_4H_IN));

        SetMPFDArrayBit(2u, 11u, ((unsigned int) CURR_1L_IN));
        SetMPFDArrayBit(2u, 10u, ((unsigned int) CURR_1B_IN));
        SetMPFDArrayBit(2u, 9u, ((unsigned int) CURR_1H_IN));
        SetMPFDArrayBit(2u, 8u, ((unsigned int) CURR_2L_IN));
        SetMPFDArrayBit(2u, 7u, ((unsigned int) CURR_2B_IN));
        SetMPFDArrayBit(2u, 6u, ((unsigned int) CURR_2H_IN));
        SetMPFDArrayBit(2u, 5u, ((unsigned int) CURR_3L_IN));
        SetMPFDArrayBit(2u, 4u, ((unsigned int) CURR_1B_IN));
        SetMPFDArrayBit(2u, 3u, ((unsigned int) CURR_1H_IN));
        SetMPFDArrayBit(2u, 2u, ((unsigned int) CURR_4L_IN));
        SetMPFDArrayBit(2u, 1u, ((unsigned int) CURR_4B_IN));
        SetMPFDArrayBit(2u, 0u, ((unsigned int) CURR_4H_IN));
            TickWait3Quarter();

        /*** SCL0 ***/
        LATF &= 0b11000111;
        HV_LLD_SCL_TRIS = 0;

            TickWait4Quarter();
}

void BBi2c_read(unsigned char k) {
            TickWait1Quarter();

        /*** SCL1 ***/
        LATF |= 0b00111000;
        HV_LLD_SCL_TRIS = 1;

        SetMPFDArrayBit(3u, k, ((unsigned int) HV_LLD_SDA_IN));

        SetMPFDArrayBit(4u, k, ((unsigned int) TEMP_1_IN));
        SetMPFDArrayBit(5u, k, ((unsigned int) TEMP_2_IN));
        SetMPFDArrayBit(6u, k, ((unsigned int) TEMP_3_IN));
        SetMPFDArrayBit(7u, k, ((unsigned int) TEMP_4_IN));

        SetMPFDArrayBit(20u, k, ((unsigned int) MSV_1L_IN));
        SetMPFDArrayBit(21u, k, ((unsigned int) MSV_1B_IN));
        SetMPFDArrayBit(22u, k, ((unsigned int) MSV_1H_IN));

        SetMPFDArrayBit(23u, k, ((unsigned int) MSV_2L_IN));
        SetMPFDArrayBit(24u, k, ((unsigned int) MSV_2B_IN));
        SetMPFDArrayBit(25u, k, ((unsigned int) MSV_2H_IN));

        SetMPFDArrayBit(26u, k, ((unsigned int) MSV_3L_IN));
        SetMPFDArrayBit(27u, k, ((unsigned int) MSV_3B_IN));
```

```
        SetMPFDArrayBit(28u, k, ((unsigned int) MSV_3H_IN));

        SetMPFDArrayBit(29u, k, ((unsigned int) MSV_4L_IN));
        SetMPFDArrayBit(30u, k, ((unsigned int) MSV_4B_IN));
        SetMPFDArrayBit(31u, k, ((unsigned int) MSV_4H_IN));

        SetMPFDArrayBit(32u, k, ((unsigned int) CURR_1L_IN));
        SetMPFDArrayBit(33u, k, ((unsigned int) CURR_1B_IN));
        SetMPFDArrayBit(34u, k, ((unsigned int) CURR_1H_IN));

        SetMPFDArrayBit(35u, k, ((unsigned int) CURR_2L_IN));
        SetMPFDArrayBit(36u, k, ((unsigned int) CURR_2B_IN));
        SetMPFDArrayBit(37u, k, ((unsigned int) CURR_2H_IN));

        SetMPFDArrayBit(38u, k, ((unsigned int) CURR_3L_IN));
        SetMPFDArrayBit(39u, k, ((unsigned int) CURR_3B_IN));
        SetMPFDArrayBit(40u, k, ((unsigned int) CURR_3H_IN));

        SetMPFDArrayBit(41u, k, ((unsigned int) CURR_4L_IN));
        SetMPFDArrayBit(42u, k, ((unsigned int) CURR_4B_IN));
        SetMPFDArrayBit(43u, k, ((unsigned int) CURR_4H_IN));

            TickWait3Quarter();

        /*** SCL0 ***/
        LATF &= 0b11000111;
        HV_LLD_SCL_TRIS = 0;

            TickWait4Quarter();
}

void FindADCAddr(void) {
    unsigned char i;

        BBi2c_stop();

    Addr_MSV_1H.Val = 0x90u;
    Addr_MSV_1L.Val = 0x90u;
    Addr_MSV_1B.Val = 0x90u;
    Addr_MSV_2H.Val = 0x90u;
    Addr_MSV_2L.Val = 0x90u;
    Addr_MSV_2B.Val = 0x90u;
    Addr_MSV_3H.Val = 0x90u;
    Addr_MSV_3L.Val = 0x90u;
    Addr_MSV_3B.Val = 0x90u;
    Addr_MSV_4H.Val = 0x90u;
    Addr_MSV_4L.Val = 0x90u;
    Addr_MSV_4B.Val = 0x90u;
    Addr_CURR_1H.Val = 0x90u;
    Addr_CURR_1L.Val = 0x90u;
    Addr_CURR_1B.Val = 0x90u;
```

```
Addr_CURR_2H.Val = 0x90u;
Addr_CURR_2L.Val = 0x90u;
Addr_CURR_2B.Val = 0x90u;
Addr_CURR_3H.Val = 0x90u;
Addr_CURR_3L.Val = 0x90u;
Addr_CURR_3B.Val = 0x90u;
Addr_CURR_4H.Val = 0x90u;
Addr_CURR_4L.Val = 0x90u;
Addr_CURR_4B.Val = 0x90u;
Addr_TEMP_1.Val = 0x90u;
Addr_TEMP_2.Val = 0x90u;
Addr_TEMP_3.Val = 0x90u;
Addr_TEMP_4.Val = 0x90u;
Addr_HV.Val = 0x90u;

for (i = 0u; i < 8u; ++i) {
    BBi2c_start();
    /*** SDA - bit 7 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b7;
    MSV_1L_TRIS = Addr_MSV_1L.bits.b7;
    MSV_1B_TRIS = Addr_MSV_1B.bits.b7;
    MSV_2H_TRIS = Addr_MSV_2H.bits.b7;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b7;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b7;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b7;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b7;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b7;
    MSV_4H_TRIS = Addr_MSV_4H.bits.b7;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b7;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b7;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b7;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b7;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b7;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b7;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b7;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b7;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b7;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b7;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b7;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b7;
    CURR_4L_TRIS = Addr_CURR_4L.bits.b7;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b7;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b7;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b7;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b7;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b7;
    HV_LLD_SDA_TRIS = Addr_HV.bits.b7;

    BBi2c_write();
    /*** SDA - bit 6 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b6;
```

```
MSV_1L_TRIS = Addr_MSV_1L.bits.b6;
MSV_1B_TRIS = Addr_MSV_1B.bits.b6;
MSV_2H_TRIS = Addr_MSV_2H.bits.b6;
MSV_2L_TRIS = Addr_MSV_2L.bits.b6;
MSV_2B_TRIS = Addr_MSV_2B.bits.b6;
MSV_3H_TRIS = Addr_MSV_3H.bits.b6;
MSV_3L_TRIS = Addr_MSV_3L.bits.b6;
MSV_3B_TRIS = Addr_MSV_3B.bits.b6;
MSV_4H_TRIS = Addr_MSV_4H.bits.b6;
MSV_4L_TRIS = Addr_MSV_4L.bits.b6;
MSV_4B_TRIS = Addr_MSV_4B.bits.b6;
CURR_1H_TRIS = Addr_CURR_1H.bits.b6;
CURR_1L_TRIS = Addr_CURR_1L.bits.b6;
CURR_1B_TRIS = Addr_CURR_1B.bits.b6;
CURR_2H_TRIS = Addr_CURR_2H.bits.b6;
CURR_2L_TRIS = Addr_CURR_2L.bits.b6;
CURR_2B_TRIS = Addr_CURR_2B.bits.b6;
CURR_3H_TRIS = Addr_CURR_3H.bits.b6;
CURR_3L_TRIS = Addr_CURR_3L.bits.b6;
CURR_3B_TRIS = Addr_CURR_3B.bits.b6;
CURR_4H_TRIS = Addr_CURR_4H.bits.b6;
CURR_4L_TRIS = Addr_CURR_4L.bits.b6;
CURR_4B_TRIS = Addr_CURR_4B.bits.b6;
TEMP_1_TRIS = Addr_TEMP_1.bits.b6;
TEMP_2_TRIS = Addr_TEMP_2.bits.b6;
TEMP_3_TRIS = Addr_TEMP_3.bits.b6;
TEMP_4_TRIS = Addr_TEMP_4.bits.b6;
HV_LLD_SDA_TRIS = Addr_HV.bits.b6;

BBi2c_write();
/*** SDA - bit 5 ***/
MSV_1H_TRIS = Addr_MSV_1H.bits.b5;
MSV_1L_TRIS = Addr_MSV_1L.bits.b5;
MSV_1B_TRIS = Addr_MSV_1B.bits.b5;
MSV_2H_TRIS = Addr_MSV_2H.bits.b5;
MSV_2L_TRIS = Addr_MSV_2L.bits.b5;
MSV_2B_TRIS = Addr_MSV_2B.bits.b5;
MSV_3H_TRIS = Addr_MSV_3H.bits.b5;
MSV_3L_TRIS = Addr_MSV_3L.bits.b5;
MSV_3B_TRIS = Addr_MSV_3B.bits.b5;
MSV_4H_TRIS = Addr_MSV_4H.bits.b5;
MSV_4L_TRIS = Addr_MSV_4L.bits.b5;
MSV_4B_TRIS = Addr_MSV_4B.bits.b5;
CURR_1H_TRIS = Addr_CURR_1H.bits.b5;
CURR_1L_TRIS = Addr_CURR_1L.bits.b5;
CURR_1B_TRIS = Addr_CURR_1B.bits.b5;
CURR_2H_TRIS = Addr_CURR_2H.bits.b5;
CURR_2L_TRIS = Addr_CURR_2L.bits.b5;
CURR_2B_TRIS = Addr_CURR_2B.bits.b5;
CURR_3H_TRIS = Addr_CURR_3H.bits.b5;
CURR_3L_TRIS = Addr_CURR_3L.bits.b5;
```

```
CURR_3B_TRIS = Addr_CURR_3B.bits.b5;
CURR_4H_TRIS = Addr_CURR_4H.bits.b5;
CURR_4L_TRIS = Addr_CURR_4L.bits.b5;
CURR_4B_TRIS = Addr_CURR_4B.bits.b5;
TEMP_1_TRIS = Addr_TEMP_1.bits.b5;
TEMP_2_TRIS = Addr_TEMP_2.bits.b5;
TEMP_3_TRIS = Addr_TEMP_3.bits.b5;
TEMP_4_TRIS = Addr_TEMP_4.bits.b5;
HV_LLD_SDA_TRIS = Addr_HV.bits.b5;

BBi2c_write();
/*** SDA - bit 4 ***/
MSV_1H_TRIS = Addr_MSV_1H.bits.b4;
MSV_1L_TRIS = Addr_MSV_1L.bits.b4;
MSV_1B_TRIS = Addr_MSV_1B.bits.b4;
MSV_2H_TRIS = Addr_MSV_2H.bits.b4;
MSV_2L_TRIS = Addr_MSV_2L.bits.b4;
MSV_2B_TRIS = Addr_MSV_2B.bits.b4;
MSV_3H_TRIS = Addr_MSV_3H.bits.b4;
MSV_3L_TRIS = Addr_MSV_3L.bits.b4;
MSV_3B_TRIS = Addr_MSV_3B.bits.b4;
MSV_4H_TRIS = Addr_MSV_4H.bits.b4;
MSV_4L_TRIS = Addr_MSV_4L.bits.b4;
MSV_4B_TRIS = Addr_MSV_4B.bits.b4;
CURR_1H_TRIS = Addr_CURR_1H.bits.b4;
CURR_1L_TRIS = Addr_CURR_1L.bits.b4;
CURR_1B_TRIS = Addr_CURR_1B.bits.b4;
CURR_2H_TRIS = Addr_CURR_2H.bits.b4;
CURR_2L_TRIS = Addr_CURR_2L.bits.b4;
CURR_2B_TRIS = Addr_CURR_2B.bits.b4;
CURR_3H_TRIS = Addr_CURR_3H.bits.b4;
CURR_3L_TRIS = Addr_CURR_3L.bits.b4;
CURR_3B_TRIS = Addr_CURR_3B.bits.b4;
CURR_4H_TRIS = Addr_CURR_4H.bits.b4;
CURR_4L_TRIS = Addr_CURR_4L.bits.b4;
CURR_4B_TRIS = Addr_CURR_4B.bits.b4;
TEMP_1_TRIS = Addr_TEMP_1.bits.b4;
TEMP_2_TRIS = Addr_TEMP_2.bits.b4;
TEMP_3_TRIS = Addr_TEMP_3.bits.b4;
TEMP_4_TRIS = Addr_TEMP_4.bits.b4;
HV_LLD_SDA_TRIS = Addr_HV.bits.b4;

BBi2c_write();
/*** SDA - bit 3 ***/
MSV_1H_TRIS = Addr_MSV_1H.bits.b3;
MSV_1L_TRIS = Addr_MSV_1L.bits.b3;
MSV_1B_TRIS = Addr_MSV_1B.bits.b3;
MSV_2H_TRIS = Addr_MSV_2H.bits.b3;
MSV_2L_TRIS = Addr_MSV_2L.bits.b3;
MSV_2B_TRIS = Addr_MSV_2B.bits.b3;
MSV_3H_TRIS = Addr_MSV_3H.bits.b3;
```

```
MSV_3L_TRIS = Addr_MSV_3L.bits.b3;
MSV_3B_TRIS = Addr_MSV_3B.bits.b3;
MSV_4H_TRIS = Addr_MSV_4H.bits.b3;
MSV_4L_TRIS = Addr_MSV_4L.bits.b3;
MSV_4B_TRIS = Addr_MSV_4B.bits.b3;
CURR_1H_TRIS = Addr_CURR_1H.bits.b3;
CURR_1L_TRIS = Addr_CURR_1L.bits.b3;
CURR_1B_TRIS = Addr_CURR_1B.bits.b3;
CURR_2H_TRIS = Addr_CURR_2H.bits.b3;
CURR_2L_TRIS = Addr_CURR_2L.bits.b3;
CURR_2B_TRIS = Addr_CURR_2B.bits.b3;
CURR_3H_TRIS = Addr_CURR_3H.bits.b3;
CURR_3L_TRIS = Addr_CURR_3L.bits.b3;
CURR_3B_TRIS = Addr_CURR_3B.bits.b3;
CURR_4H_TRIS = Addr_CURR_4H.bits.b3;
CURR_4L_TRIS = Addr_CURR_4L.bits.b3;
CURR_4B_TRIS = Addr_CURR_4B.bits.b3;
TEMP_1_TRIS = Addr_TEMP_1.bits.b3;
TEMP_2_TRIS = Addr_TEMP_2.bits.b3;
TEMP_3_TRIS = Addr_TEMP_3.bits.b3;
TEMP_4_TRIS = Addr_TEMP_4.bits.b3;
HV_LLD_SDA_TRIS = Addr_HV.bits.b3;

BBi2c_write();
/*** SDA - bit 2 ***/
MSV_1H_TRIS = Addr_MSV_1H.bits.b2;
MSV_1L_TRIS = Addr_MSV_1L.bits.b2;
MSV_1B_TRIS = Addr_MSV_1B.bits.b2;
MSV_2H_TRIS = Addr_MSV_2H.bits.b2;
MSV_2L_TRIS = Addr_MSV_2L.bits.b2;
MSV_2B_TRIS = Addr_MSV_2B.bits.b2;
MSV_3H_TRIS = Addr_MSV_3H.bits.b2;
MSV_3L_TRIS = Addr_MSV_3L.bits.b2;
MSV_3B_TRIS = Addr_MSV_3B.bits.b2;
MSV_4H_TRIS = Addr_MSV_4H.bits.b2;
MSV_4L_TRIS = Addr_MSV_4L.bits.b2;
MSV_4B_TRIS = Addr_MSV_4B.bits.b2;
CURR_1H_TRIS = Addr_CURR_1H.bits.b2;
CURR_1L_TRIS = Addr_CURR_1L.bits.b2;
CURR_1B_TRIS = Addr_CURR_1B.bits.b2;
CURR_2H_TRIS = Addr_CURR_2H.bits.b2;
CURR_2L_TRIS = Addr_CURR_2L.bits.b2;
CURR_2B_TRIS = Addr_CURR_2B.bits.b2;
CURR_3H_TRIS = Addr_CURR_3H.bits.b2;
CURR_3L_TRIS = Addr_CURR_3L.bits.b2;
CURR_3B_TRIS = Addr_CURR_3B.bits.b2;
CURR_4H_TRIS = Addr_CURR_4H.bits.b2;
CURR_4L_TRIS = Addr_CURR_4L.bits.b2;
CURR_4B_TRIS = Addr_CURR_4B.bits.b2;
TEMP_1_TRIS = Addr_TEMP_1.bits.b2;
TEMP_2_TRIS = Addr_TEMP_2.bits.b2;
```

```
TEMP_3_TRIS = Addr_TEMP_3.bits.b2;
TEMP_4_TRIS = Addr_TEMP_4.bits.b2;
HV_LLD_SDA_TRIS = Addr_HV.bits.b2;

BBi2c_write();
/*** SDA - bit 1 ***/
MSV_1H_TRIS = Addr_MSV_1H.bits.b1;
MSV_1L_TRIS = Addr_MSV_1L.bits.b1;
MSV_1B_TRIS = Addr_MSV_1B.bits.b1;
MSV_2H_TRIS = Addr_MSV_2H.bits.b1;
MSV_2L_TRIS = Addr_MSV_2L.bits.b1;
MSV_2B_TRIS = Addr_MSV_2B.bits.b1;
MSV_3H_TRIS = Addr_MSV_3H.bits.b1;
MSV_3L_TRIS = Addr_MSV_3L.bits.b1;
MSV_3B_TRIS = Addr_MSV_3B.bits.b1;
MSV_4H_TRIS = Addr_MSV_4H.bits.b1;
MSV_4L_TRIS = Addr_MSV_4L.bits.b1;
MSV_4B_TRIS = Addr_MSV_4B.bits.b1;
CURR_1H_TRIS = Addr_CURR_1H.bits.b1;
CURR_1L_TRIS = Addr_CURR_1L.bits.b1;
CURR_1B_TRIS = Addr_CURR_1B.bits.b1;
CURR_2H_TRIS = Addr_CURR_2H.bits.b1;
CURR_2L_TRIS = Addr_CURR_2L.bits.b1;
CURR_2B_TRIS = Addr_CURR_2B.bits.b1;
CURR_3H_TRIS = Addr_CURR_3H.bits.b1;
CURR_3L_TRIS = Addr_CURR_3L.bits.b1;
CURR_3B_TRIS = Addr_CURR_3B.bits.b1;
CURR_4H_TRIS = Addr_CURR_4H.bits.b1;
CURR_4L_TRIS = Addr_CURR_4L.bits.b1;
CURR_4B_TRIS = Addr_CURR_4B.bits.b1;
TEMP_1_TRIS = Addr_TEMP_1.bits.b1;
TEMP_2_TRIS = Addr_TEMP_2.bits.b1;
TEMP_3_TRIS = Addr_TEMP_3.bits.b1;
TEMP_4_TRIS = Addr_TEMP_4.bits.b1;
HV_LLD_SDA_TRIS = Addr_HV.bits.b1;

BBi2c_write();
/*** SDA - bit 0 ***/
MSV_1H_TRIS = Addr_MSV_1H.bits.b0;
MSV_1L_TRIS = Addr_MSV_1L.bits.b0;
MSV_1B_TRIS = Addr_MSV_1B.bits.b0;
MSV_2H_TRIS = Addr_MSV_2H.bits.b0;
MSV_2L_TRIS = Addr_MSV_2L.bits.b0;
MSV_2B_TRIS = Addr_MSV_2B.bits.b0;
MSV_3H_TRIS = Addr_MSV_3H.bits.b0;
MSV_3L_TRIS = Addr_MSV_3L.bits.b0;
MSV_3B_TRIS = Addr_MSV_3B.bits.b0;
MSV_4H_TRIS = Addr_MSV_4H.bits.b0;
MSV_4L_TRIS = Addr_MSV_4L.bits.b0;
MSV_4B_TRIS = Addr_MSV_4B.bits.b0;
CURR_1H_TRIS = Addr_CURR_1H.bits.b0;
```

```
CURR_1L_TRIS = Addr_CURR_1L.bits.b0;
CURR_1B_TRIS = Addr_CURR_1B.bits.b0;
CURR_2H_TRIS = Addr_CURR_2H.bits.b0;
CURR_2L_TRIS = Addr_CURR_2L.bits.b0;
CURR_2B_TRIS = Addr_CURR_2B.bits.b0;
CURR_3H_TRIS = Addr_CURR_3H.bits.b0;
CURR_3L_TRIS = Addr_CURR_3L.bits.b0;
CURR_3B_TRIS = Addr_CURR_3B.bits.b0;
CURR_4H_TRIS = Addr_CURR_4H.bits.b0;
CURR_4L_TRIS = Addr_CURR_4L.bits.b0;
CURR_4B_TRIS = Addr_CURR_4B.bits.b0;
TEMP_1_TRIS = Addr_TEMP_1.bits.b0;
TEMP_2_TRIS = Addr_TEMP_2.bits.b0;
TEMP_3_TRIS = Addr_TEMP_3.bits.b0;
TEMP_4_TRIS = Addr_TEMP_4.bits.b0;
HV_LLD_SDA_TRIS = Addr_HV.bits.b0;

BBi2c_write();

/*** Ack ***/
/*** Set to inputs ***/
MSV_1H_TRIS = 1;
MSV_1L_TRIS = 1;
MSV_1B_TRIS = 1;
MSV_2H_TRIS = 1;
MSV_2L_TRIS = 1;
MSV_2B_TRIS = 1;
MSV_3H_TRIS = 1;
MSV_3L_TRIS = 1;
MSV_3B_TRIS = 1;
MSV_4H_TRIS = 1;
MSV_4L_TRIS = 1;
MSV_4B_TRIS = 1;
CURR_1H_TRIS = 1;
CURR_1L_TRIS = 1;
CURR_1B_TRIS = 1;
CURR_2H_TRIS = 1;
CURR_2L_TRIS = 1;
CURR_2B_TRIS = 1;
CURR_3H_TRIS = 1;
CURR_3L_TRIS = 1;
CURR_3B_TRIS = 1;
CURR_4H_TRIS = 1;
CURR_4L_TRIS = 1;
CURR_4B_TRIS = 1;
TEMP_1_TRIS = 1;
TEMP_2_TRIS = 1;
TEMP_3_TRIS = 1;
TEMP_4_TRIS = 1;
HV_LLD_SDA_TRIS = 1;
```

```
/*** SCL1 ***/
LATF |= 0b00111000;
HV_LLD_SCL_TRIS = 1;

TickWait1Quarter();

/*** Check if Ack – if Nack then increment address ***/
if (MSV_1H_IN == 1u) {
    Addr_MSV_1H.Val += 2u;
}
if (MSV_1L_IN == 1u) {
    Addr_MSV_1L.Val += 2u;
}
if (MSV_1B_IN == 1u) {
    Addr_MSV_1B.Val += 2u;
}
if (MSV_2H_IN == 1u) {
    Addr_MSV_2H.Val += 2u;
}
if (MSV_2L_IN == 1u) {
    Addr_MSV_2L.Val += 2u;
}
if (MSV_2B_IN == 1u) {
    Addr_MSV_2B.Val += 2u;
}
if (MSV_3H_IN == 1u) {
    Addr_MSV_3H.Val += 2u;
}
if (MSV_3L_IN == 1u) {
    Addr_MSV_3L.Val += 2u;
}
if (MSV_3B_IN == 1u) {
    Addr_MSV_3B.Val += 2u;
}
if (MSV_4H_IN == 1u) {
    Addr_MSV_4H.Val += 2u;
}
if (MSV_4L_IN == 1u) {
    Addr_MSV_4L.Val += 2u;
}
if (MSV_4B_IN == 1u) {
    Addr_MSV_4B.Val += 2u;
}

if (CURR_1H_IN == 1u) {
    Addr_CURR_1H.Val += 2u;
}
if (CURR_1L_IN == 1u) {
    Addr_CURR_1L.Val += 2u;
}
if (CURR_1B_IN == 1u) {
```

```
        Addr_CURR_1B.Val += 2u;
}
if (CURR_2H_IN == 1u) {
        Addr_CURR_2H.Val += 2u;
}
if (CURR_2L_IN == 1u) {
        Addr_CURR_2L.Val += 2u;
}
if (CURR_2B_IN == 1u) {
        Addr_CURR_2B.Val += 2u;
}
if (CURR_3H_IN == 1u) {
        Addr_CURR_3H.Val += 2u;
}
if (CURR_3L_IN == 1u) {
        Addr_CURR_3L.Val += 2u;
}
if (CURR_3B_IN == 1u) {
        Addr_CURR_3B.Val += 2u;
}
if (CURR_4H_IN == 1u) {
        Addr_CURR_4H.Val += 2u;
}
if (CURR_4L_IN == 1u) {
        Addr_CURR_4L.Val += 2u;
}
if (CURR_4B_IN == 1u) {
        Addr_CURR_4B.Val += 2u;
}

if (TEMP_1_IN == 1u) {
        Addr_TEMP_1.Val += 2u;
}
if (TEMP_2_IN == 1u) {
        Addr_TEMP_2.Val += 2u;
}
if (TEMP_3_IN == 1u) {
        Addr_TEMP_3.Val += 2u;
}
if (TEMP_4_IN == 1u) {
        Addr_TEMP_4.Val += 2u;
}

if (HV_LLD_SDA_IN == 1u) {
        Addr_HV.Val += 2u;
}

TickWait3Quarter();

/*** SCL0 ***/
LATF &= 0b11000111;
```

```
            HV_LLD_SCL_TRIS = 0;

            TickWait4Quarter();

            BBi2c_stop();
        }

}

void BBi2c_Addr(unsigned char ReadWrite) {
    // write - ReadWrite = 0x00u;
    // read - ReadWrite = 0x01u;
    BYTE_VAL RW;
    RW.Val = ReadWrite;

        /*** SDA - bit 7 ***/
        MSV_1H_TRIS = Addr_MSV_1H.bits.b7;
        MSV_1L_TRIS = Addr_MSV_1L.bits.b7;
        MSV_1B_TRIS = Addr_MSV_1B.bits.b7;
        MSV_2H_TRIS = Addr_MSV_2H.bits.b7;
        MSV_2L_TRIS = Addr_MSV_2L.bits.b7;
        MSV_2B_TRIS = Addr_MSV_2B.bits.b7;
        MSV_3H_TRIS = Addr_MSV_3H.bits.b7;
        MSV_3L_TRIS = Addr_MSV_3L.bits.b7;
        MSV_3B_TRIS = Addr_MSV_3B.bits.b7;
        MSV_4H_TRIS = Addr_MSV_4H.bits.b7;
        MSV_4L_TRIS = Addr_MSV_4L.bits.b7;
        MSV_4B_TRIS = Addr_MSV_4B.bits.b7;
        CURR_1H_TRIS = Addr_CURR_1H.bits.b7;
        CURR_1L_TRIS = Addr_CURR_1L.bits.b7;
        CURR_1B_TRIS = Addr_CURR_1B.bits.b7;
        CURR_2H_TRIS = Addr_CURR_2H.bits.b7;
        CURR_2L_TRIS = Addr_CURR_2L.bits.b7;
        CURR_2B_TRIS = Addr_CURR_2B.bits.b7;
        CURR_3H_TRIS = Addr_CURR_3H.bits.b7;
        CURR_3L_TRIS = Addr_CURR_3L.bits.b7;
        CURR_3B_TRIS = Addr_CURR_3B.bits.b7;
        CURR_4H_TRIS = Addr_CURR_4H.bits.b7;
        CURR_4L_TRIS = Addr_CURR_4L.bits.b7;
        CURR_4B_TRIS = Addr_CURR_4B.bits.b7;
        TEMP_1_TRIS = Addr_TEMP_1.bits.b7;
        TEMP_2_TRIS = Addr_TEMP_2.bits.b7;
        TEMP_3_TRIS = Addr_TEMP_3.bits.b7;
        TEMP_4_TRIS = Addr_TEMP_4.bits.b7;
        HV_LLD_SDA_TRIS = Addr_HV.bits.b7;

    BBi2c_write();
        /*** SDA - bit 6 ***/
        MSV_1H_TRIS = Addr_MSV_1H.bits.b6;
        MSV_1L_TRIS = Addr_MSV_1L.bits.b6;
        MSV_1B_TRIS = Addr_MSV_1B.bits.b6;
```

```
    MSV_2H_TRIS = Addr_MSV_2H.bits.b6;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b6;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b6;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b6;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b6;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b6;
    MSV_4H_TRIS = Addr_MSV_4H.bits.b6;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b6;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b6;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b6;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b6;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b6;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b6;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b6;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b6;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b6;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b6;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b6;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b6;
    CURR_4L_TRIS = Addr_CURR_4L.bits.b6;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b6;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b6;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b6;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b6;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b6;
    HV_LLD_SDA_TRIS = Addr_HV.bits.b6;

BBi2c_write();
    /*** SDA - bit 5 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b5;
    MSV_1L_TRIS = Addr_MSV_1L.bits.b5;
    MSV_1B_TRIS = Addr_MSV_1B.bits.b5;
    MSV_2H_TRIS = Addr_MSV_2H.bits.b5;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b5;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b5;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b5;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b5;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b5;
    MSV_4H_TRIS = Addr_MSV_4H.bits.b5;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b5;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b5;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b5;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b5;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b5;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b5;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b5;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b5;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b5;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b5;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b5;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b5;
```

```
    CURR_4L_TRIS = Addr_CURR_4L.bits.b5;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b5;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b5;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b5;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b5;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b5;
    HV_LLD_SDA_TRIS = Addr_HV.bits.b5;

BBi2c_write();
    /*** SDA - bit 4 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b4;
    MSV_1L_TRIS = Addr_MSV_1L.bits.b4;
    MSV_1B_TRIS = Addr_MSV_1B.bits.b4;
    MSV_2H_TRIS = Addr_MSV_2H.bits.b4;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b4;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b4;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b4;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b4;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b4;
    MSV_4H_TRIS = Addr_MSV_4H.bits.b4;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b4;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b4;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b4;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b4;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b4;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b4;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b4;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b4;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b4;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b4;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b4;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b4;
    CURR_4L_TRIS = Addr_CURR_4L.bits.b4;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b4;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b4;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b4;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b4;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b4;
    HV_LLD_SDA_TRIS = Addr_HV.bits.b4;

BBi2c_write();
    /*** SDA - bit 3 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b3;
    MSV_1L_TRIS = Addr_MSV_1L.bits.b3;
    MSV_1B_TRIS = Addr_MSV_1B.bits.b3;
    MSV_2H_TRIS = Addr_MSV_2H.bits.b3;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b3;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b3;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b3;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b3;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b3;
```

```
    MSV_4H_TRIS = Addr_MSV_4H.bits.b3;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b3;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b3;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b3;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b3;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b3;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b3;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b3;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b3;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b3;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b3;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b3;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b3;
    CURR_4L_TRIS = Addr_CURR_4L.bits.b3;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b3;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b3;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b3;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b3;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b3;
    HV_LLD_SDA_TRIS = Addr_HV.bits.b3;

BBi2c_write();
    /*** SDA - bit 2 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b2;
    MSV_1L_TRIS = Addr_MSV_1L.bits.b2;
    MSV_1B_TRIS = Addr_MSV_1B.bits.b2;
    MSV_2H_TRIS = Addr_MSV_2H.bits.b2;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b2;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b2;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b2;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b2;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b2;
    MSV_4H_TRIS = Addr_MSV_4H.bits.b2;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b2;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b2;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b2;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b2;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b2;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b2;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b2;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b2;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b2;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b2;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b2;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b2;
    CURR_4L_TRIS = Addr_CURR_4L.bits.b2;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b2;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b2;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b2;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b2;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b2;
```

```
    HV_LLD_SDA_TRIS = Addr_HV.bits.b2;

BBi2c_write();
    /*** SDA - bit 1 ***/
    MSV_1H_TRIS = Addr_MSV_1H.bits.b1;
    MSV_1L_TRIS = Addr_MSV_1L.bits.b1;
    MSV_1B_TRIS = Addr_MSV_1B.bits.b1;
    MSV_2H_TRIS = Addr_MSV_2H.bits.b1;
    MSV_2L_TRIS = Addr_MSV_2L.bits.b1;
    MSV_2B_TRIS = Addr_MSV_2B.bits.b1;
    MSV_3H_TRIS = Addr_MSV_3H.bits.b1;
    MSV_3L_TRIS = Addr_MSV_3L.bits.b1;
    MSV_3B_TRIS = Addr_MSV_3B.bits.b1;
    MSV_4H_TRIS = Addr_MSV_4H.bits.b1;
    MSV_4L_TRIS = Addr_MSV_4L.bits.b1;
    MSV_4B_TRIS = Addr_MSV_4B.bits.b1;
    CURR_1H_TRIS = Addr_CURR_1H.bits.b1;
    CURR_1L_TRIS = Addr_CURR_1L.bits.b1;
    CURR_1B_TRIS = Addr_CURR_1B.bits.b1;
    CURR_2H_TRIS = Addr_CURR_2H.bits.b1;
    CURR_2L_TRIS = Addr_CURR_2L.bits.b1;
    CURR_2B_TRIS = Addr_CURR_2B.bits.b1;
    CURR_3H_TRIS = Addr_CURR_3H.bits.b1;
    CURR_3L_TRIS = Addr_CURR_3L.bits.b1;
    CURR_3B_TRIS = Addr_CURR_3B.bits.b1;
    CURR_4H_TRIS = Addr_CURR_4H.bits.b1;
    CURR_4L_TRIS = Addr_CURR_4L.bits.b1;
    CURR_4B_TRIS = Addr_CURR_4B.bits.b1;
    TEMP_1_TRIS = Addr_TEMP_1.bits.b1;
    TEMP_2_TRIS = Addr_TEMP_2.bits.b1;
    TEMP_3_TRIS = Addr_TEMP_3.bits.b1;
    TEMP_4_TRIS = Addr_TEMP_4.bits.b1;
    HV_LLD_SDA_TRIS = Addr_HV.bits.b1;

BBi2c_write();
    /*** SDA - bit 0 ***/
    MSV_1H_TRIS = RW.bits.b0;
    MSV_1L_TRIS = RW.bits.b0;
    MSV_1B_TRIS = RW.bits.b0;
    MSV_2H_TRIS = RW.bits.b0;
    MSV_2L_TRIS = RW.bits.b0;
    MSV_2B_TRIS = RW.bits.b0;
    MSV_3H_TRIS = RW.bits.b0;
    MSV_3L_TRIS = RW.bits.b0;
    MSV_3B_TRIS = RW.bits.b0;
    MSV_4H_TRIS = RW.bits.b0;
    MSV_4L_TRIS = RW.bits.b0;
    MSV_4B_TRIS = RW.bits.b0;
    CURR_1H_TRIS = RW.bits.b0;
    CURR_1L_TRIS = RW.bits.b0;
    CURR_1B_TRIS = RW.bits.b0;
```

```
        CURR_2H_TRIS = RW.bits.b0;
        CURR_2L_TRIS = RW.bits.b0;
        CURR_2B_TRIS = RW.bits.b0;
        CURR_3H_TRIS = RW.bits.b0;
        CURR_3L_TRIS = RW.bits.b0;
        CURR_3B_TRIS = RW.bits.b0;
        CURR_4H_TRIS = RW.bits.b0;
        CURR_4L_TRIS = RW.bits.b0;
        CURR_4B_TRIS = RW.bits.b0;
        TEMP_1_TRIS = RW.bits.b0;
        TEMP_2_TRIS = RW.bits.b0;
        TEMP_3_TRIS = RW.bits.b0;
        TEMP_4_TRIS = RW.bits.b0;
        HV_LLD_SDA_TRIS = RW.bits.b0;

    BBi2c_write();

    /*** Ack ***/
        /*** Set to inputs ***/
        MSV_1H_TRIS = 1;
        MSV_1L_TRIS = 1;
        MSV_1B_TRIS = 1;
        MSV_2H_TRIS = 1;
        MSV_2L_TRIS = 1;
        MSV_2B_TRIS = 1;
        MSV_3H_TRIS = 1;
        MSV_3L_TRIS = 1;
        MSV_3B_TRIS = 1;
        MSV_4H_TRIS = 1;
        MSV_4L_TRIS = 1;
        MSV_4B_TRIS = 1;
        CURR_1H_TRIS = 1;
        CURR_1L_TRIS = 1;
        CURR_1B_TRIS = 1;
        CURR_2H_TRIS = 1;
        CURR_2L_TRIS = 1;
        CURR_2B_TRIS = 1;
        CURR_3H_TRIS = 1;
        CURR_3L_TRIS = 1;
        CURR_3B_TRIS = 1;
        CURR_4H_TRIS = 1;
        CURR_4L_TRIS = 1;
        CURR_4B_TRIS = 1;
        TEMP_1_TRIS = 1;
        TEMP_2_TRIS = 1;
        TEMP_3_TRIS = 1;
        TEMP_4_TRIS = 1;
        HV_LLD_SDA_TRIS = 1;

    BBi2c_write();
}
```

```
unsigned char GetHVADCAddr(void) {
    return Addr_HV.Val;
    //return 0x92u;
}
```

End BitBangADC.c

B.6.11 HVLLD.h

This is the header file for the program HVLLD.c (Section §B.6.12) for the daughter board's microcontroller.

Begin HVLLD.h

```
#ifndef __HVLLD_H
#define __HVLLD_H

#include "GenericTypeDefs.h"
#include "config.h"

#define LLD_SET_ADDR 0b01001110u
#define LLD_READ_ADDR 0b01001111u
#define HV_SET_ADDR 0b01000100u
#define HV_READ_ADDR 0b01000101u
#define DataRate240SPS 0b00000000u
#define DataRate60SPS 0b00000100u
#define DataRate30SPS 0b00001000u
#define DataRate15SPS 0b00001100u
#define ADCGain1 0b00000000u
#define ADCGain2 0b00000001u
#define ADCGain4 0b00000010u
#define ADCGain8 0b00000011u

unsigned char hbyte(unsigned int data);
unsigned char lbyte(unsigned int data);
unsigned char SetHV(unsigned int HVSetPoint);
unsigned char SetLLD(unsigned int LLDSetPoint);
unsigned short long ReadHVADC(void);
unsigned char ConfigHVADC(unsigned char HVADCSpeed,unsigned char HVADCGain);
void swi2c_start(void);
void swi2c_stop(void);
void swi2c_writebit(void);
unsigned char swi2c_read(unsigned char ack);
unsigned char swi2c_write(unsigned char data);
void ChangeHV(unsigned int HV_Setpoint);
void SetHVvoid(unsigned int HVSetPoint);
void ConfigHVADCvoid(unsigned char HVADCSpeed,unsigned char HVADCGain);
unsigned int ChangeHVreturn(unsigned int HV_Setpoint);
```

```
#endif
```

End HVLLD.h

B.6.12 HVLLD.c

This is the HVLLD.c program that controls the reading and writing of data for the high voltage and lower level discriminator DAC on the daughter board.

Begin HVLLD.c

```
#include "HVLLD.h"
#include "config.h"
#include "Tick.h"
#include "BitBangADC.h"

static unsigned int Current_HV_Setpoint = 0x0001u;

unsigned char hbyte(unsigned int data) {
    return((data >> 8u) & 0xFFu);
}

unsigned char lbyte(unsigned int data) {
    return(data & 0xFFu);
}

void swi2c_start(void) {
    HV_LLD_SDA_TRIS = 1;
    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    TickWait2Quarter();
    HV_LLD_SDA_TRIS = 0;
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();
    return;
}

void swi2c_stop(void) {
    HV_LLD_SDA_TRIS = 0;
    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    TickWait2Quarter();
    HV_LLD_SDA_TRIS = 1;
    TickWait3Quarter();
    return;
}
```

```
void swi2c_writebit(void) {
    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();
    return;
}

unsigned char swi2c_read(unsigned char ack) {
    unsigned char d;

    HV_LLD_SDA_TRIS = 1;
    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d = 0x00u | (((unsigned char)HV_LLD_SDA_IN) << 7u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN) << 6u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN) << 5u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN) << 4u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN) << 3u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
```

```
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN) << 2u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN) << 1u);
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    d |= (((unsigned char)HV_LLD_SDA_IN));
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();

    HV_LLD_SDA_TRIS = ack;
    swi2c_writebit();
    return d;
}

unsigned char swi2c_write(unsigned char data) {
    unsigned char ack;

    HV_LLD_SDA_TRIS = ((data >> 7u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = ((data >> 6u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = ((data >> 5u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = ((data >> 4u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = ((data >> 3u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = ((data >> 2u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = ((data >> 1u) & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = (data & 0x01u);
    swi2c_writebit();
    HV_LLD_SDA_TRIS = 1;
```

```
    TickWait1Quarter();
    HV_LLD_SCL_TRIS = 1;
    while (HV_LLD_SCL_IN == 0u);
    ack = HV_LLD_SDA_IN;
    TickWait3Quarter();
    HV_LLD_SCL_TRIS = 0;
    TickWait4Quarter();
    return ack;
}


unsigned char SetHV(unsigned int HVSetPoint) {
    unsigned char HVSetPointLS, HVSetPointMS;

    HVSetPointLS = lbyte(HVSetPoint);
    HVSetPointMS = hbyte(HVSetPoint);

    swi2c_start();
    if (swi2c_write(HV_SET_ADDR) != 0u) {
        swi2c_stop();
        return(0u);
    }
    if (swi2c_write(0x04u) != 0u) {
        swi2c_stop();
        return(0u);
    }
    if (swi2c_write(HVSetPointLS) != 0u) {
        swi2c_stop();
        return(0u);
    }
    if (swi2c_write(HVSetPointMS) != 0u) {
        swi2c_stop();
        return(0u);
    }
    swi2c_stop();
    return(1u);
}

unsigned char SetLLD(unsigned int LLDSetPoint) {
    unsigned char LLDSetPointLS, LLDSetPointMS;

    LLDSetPointLS = lbyte(LLDSetPoint);
    LLDSetPointMS = hbyte(LLDSetPoint);

    swi2c_start();
    if (swi2c_write(LLD_SET_ADDR) != 0u) {
        swi2c_stop();
        return(0);
    }
    if (swi2c_write(0x04u) != 0u) {
        swi2c_stop();
```

```
            return(0);
        }
        if (swi2c_write(LLDSetPointLS) != 0u) {
            swi2c_stop();
            return(0);
        }
        if (swi2c_write(LLDSetPointMS) != 0u) {
            swi2c_stop();
            return(0);
        }
        swi2c_stop();
        return(1u);
}

unsigned short long ReadHVADC(void) {
        unsigned short long HVADCData = 0x000000u;
        unsigned char HVADCAddr;
        unsigned char tmp;

        HVADCAddr = (GetHVADCAddr() | 0x01u);
        swi2c_start();
        if (swi2c_write(HVADCAddr) != 0u) {
            swi2c_stop();
            return(0u);
        }
        tmp = swi2c_read(0u);
        HVADCData = (HVADCData | (((unsigned short long)tmp) << 16));
        tmp = swi2c_read(0u);
        HVADCData = (HVADCData | (((unsigned short long)tmp) << 8));
        tmp = swi2c_read(0u);
        swi2c_stop();
        HVADCData = (HVADCData | ((unsigned short long)tmp));
        return(HVADCData);
}

unsigned char ConfigHVADC(unsigned char HVADCSpeed,unsigned char HVADCGain) {
        swi2c_start();
        if (swi2c_write(GetHVADCAddr()) != 0u) {
            swi2c_stop();
            return(0);
        }
        if (swi2c_write(0b10000000u | HVADCSpeed | HVADCGain) != 0u) {
            swi2c_stop();
            return(0);
        }
        swi2c_stop();
        return(1);
}

void ChangeHV(unsigned int HV_Setpoint) {
        if (HV_Setpoint > Current_HV_Setpoint) {
```

```
            SetHVvoid(++Current_HV_Setpoint);
            if (Current_HV_Setpoint == 0x0032u) {
                RELAY_RESET_OUT = 1;
            } else {
                RELAY_RESET_OUT = 0;
            }
        } else if (HV_Setpoint < Current_HV_Setpoint) {
            SetHVvoid(--Current_HV_Setpoint);
            RELAY_RESET_OUT = 0;
        } else {
            RELAY_RESET_OUT = 0;
        }
        return;
}

unsigned int ChangeHVreturn(unsigned int HV_Setpoint) {
        if (HV_Setpoint > Current_HV_Setpoint) {
            SetHVvoid(++Current_HV_Setpoint);
            if (Current_HV_Setpoint == 0x0032u) {
                RELAY_RESET_OUT = 1;
            } else {
                RELAY_RESET_OUT = 0;
            }
        } else if (HV_Setpoint < Current_HV_Setpoint) {
            SetHVvoid(--Current_HV_Setpoint);
            RELAY_RESET_OUT = 0;
        } else {
            RELAY_RESET_OUT = 0;
        }
        return(Current_HV_Setpoint);
}

void SetHVvoid(unsigned int HVSetPoint) {
        unsigned char HVSetPointLS, HVSetPointMS;

        HVSetPointLS = lbyte(HVSetPoint);
        HVSetPointMS = hbyte(HVSetPoint);

        swi2c_start();
        if (swi2c_write(HV_SET_ADDR) != 0u) {
            swi2c_stop();
            return;
        }
        if (swi2c_write(0x04u) != 0u) {
            swi2c_stop();
            return;
        }
        if (swi2c_write(HVSetPointLS) != 0u) {
            swi2c_stop();
            return;
        }
```

```
    if (swi2c_write(HVSetPointMS) != 0u) {
        swi2c_stop();
        return;
    }
    swi2c_stop();
    return;
}

void ConfigHVADCvoid(unsigned char HVADCSpeed,unsigned char HVADCGain) {
    swi2c_start();
    if (swi2c_write(GetHVADCAddr()) != 0u) {
        swi2c_stop();
        return;
    }
    if (swi2c_write(0b10000000u | HVADCSpeed | HVADCGain) != 0u) {
        swi2c_stop();
        return;
    }
    swi2c_stop();
    return;
}
```

End HVLLD.c

B.6.13 PulseMode.h

This is the header file for the program PulseMode.c (Section §B.6.14) for the daughter board's microcontroller.

Begin PulseMode.h

```
#ifndef __PulseMode_H
#define __PulseMode_H

#include "config.h"
#include "globals.h"

void GetPulseData(void);

#endif // __PulseMode_H
```

End PulseMode.h

B.6.14 PulseMode.c

This is the PulseMode.c program that controls the data collection from the pulse mode

counters on the daughter board.

Begin PulseMode.c

```
#include "PulseMode.h"
#include "config.h"

extern void SetMPFDArrayBit(unsigned char MPFDByte, unsigned char MPFDBit, unsigned int ...
    MPFDData);

void Pulse_Enable(void) {
    PULSE_ENABLE_OUT = 0;
}


void Pulse_Disable(void) {
    PULSE_ENABLE_OUT = 1;
}

void Pulse_Latch(void) {
    PULSE_RCLK_OUT = 1;
    Nop();
    Nop();
    Nop();
    PULSE_RCLK_OUT = 0;
}

void Pulse_Clear(void) {
    PULSE_CLEAR_OUT = 0;
    Nop();
    Nop();
    Nop();
    PULSE_CLEAR_OUT = 1;
}

void Pulse_Load(void) {
    PULSE_LOAD_OUT = 0;
    Nop();
    Nop();
    Nop();
    PULSE_LOAD_OUT = 1;
}

void Pulse_Next_Bit(void) {
    PULSE_SCLK_OUT = 1;
    Nop();
    Nop();
    Nop();
    PULSE_SCLK_OUT = 0;
}

void GetPulseData(void) {
```

```
    unsigned char c,k;

    Pulse_Disable();
    Pulse_Latch();
    Pulse_Clear();
    Pulse_Enable();
    Pulse_Load();
    for (c = 0u; c <= 2u; c++) {
        Pulse_Next_Bit();
        Nop(); // No Data
        for (k = 0u; k <= 15u; k++) {
            Pulse_Next_Bit();
            SetMPFDArrayBit((8u + c), (15u − k), ((unsigned int) PULSE_LBH1_IN));
            SetMPFDArrayBit((11u + c), (15u − k), ((unsigned int) PULSE_LBH2_IN));
            SetMPFDArrayBit((14u + c), (15u − k), ((unsigned int) PULSE_LBH3_IN));
            SetMPFDArrayBit((17u + c), (15u − k), ((unsigned int) PULSE_LBH4_IN));
        }
    }
    Pulse_Next_Bit();
}
```

End PulseMode.c

B.6.15 convert.h

This is the header file for the program convert.c (Section §B.6.16) for the daughter board's microcontroller.

Begin convert.h

```
#ifndef __convert_H
#define __convert_H

#include "HVLLD.h"

unsigned int convertCURR(unsigned int input, unsigned char DataRate);
unsigned int convertMSV(unsigned int input, unsigned char DataRate);
unsigned int convertHV(unsigned int input, unsigned char DataRate);
unsigned int convertHVoffset(unsigned int input);
unsigned int convertTemp(unsigned int input, unsigned char DataRate);
unsigned int convertPulse(unsigned int input);

#endif // __convert_H
```

End convert.h

B.6.16 convert.c

This is the convert.c program that converts the I^2C ADC data to account for gain and sample frequency.

Begin convert.c

```
#include "convert.h"
#include "HVLLD.h"

unsigned int convertCURR(unsigned int input, unsigned char DataRate) {

    switch (DataRate) {
        case DataRate240SPS:
            input = (((0x0FFFu & input) << 4u) & 0xFFF0);
            break;
        case DataRate60SPS:
            input = (((0x3FFFu & input) << 2u) & 0xFFFC);
            break;
        case DataRate30SPS:
            input = (((0x7FFFu & input) << 1u) & 0xFFFE);
            break;
        default:
            break;
    }
    /**** (0 to 2.548) − 0.5 = −0.5 to 2.048 ****/
    return input;
}

unsigned int convertMSV(unsigned int input, unsigned char DataRate) {

    switch (DataRate) {
        case DataRate240SPS:
            input = (((0x0FFFu & input) << 4u) & 0xFFF0);
            break;
        case DataRate60SPS:
            input = (((0x3FFFu & input) << 2u) & 0xFFFC);
            break;
        case DataRate30SPS:
            input = (((0x7FFFu & input) << 1u) & 0xFFFE);
            break;
        default:
            break;
    }
    /**** 0.5 − (0 to 2.548) = 0.5 to −2.048 ****/
    return input;
}

unsigned int convertHV(unsigned int input, unsigned char DataRate) {
```

```
    switch (DataRate) {
        case DataRate240SPS:
            input = (((0x0FFFu & input) << 4u) & 0xFFF0);
            break;
        case DataRate60SPS:
            input = (((0x3FFFu & input) << 2u) & 0xFFFC);
            break;
        case DataRate30SPS:
            input = (((0x7FFFu & input) << 1u) & 0xFFFE);
            break;
        default:
            break;
    }
    /**** (0 to 4.096) - 2.048 = -2.048 to 2.048 ****/
    return input;
}

unsigned int convertHVoffset(unsigned int input) {
    /**** (0 to 4.096) - 2.048 = -2.048 to 2.048 ****/
    if ((input & 0x8000u) == 0x8000u) {
        input = (input & 0x7FFFu);
    } else {
        input = (input | 0x8000u);
    }
    return input;
}

unsigned int convertTemp(unsigned int input, unsigned char DataRate) {
    switch (DataRate) {
        case DataRate240SPS:
            input = (((0x0FFFu & input) << 4u) & 0xFFF0);
            break;
        case DataRate60SPS:
            input = (((0x3FFFu & input) << 2u) & 0xFFFC);
            break;
        case DataRate30SPS:
            input = (((0x7FFFu & input) << 1u) & 0xFFFE);
            break;
        default:
            break;
    }
    /**** (0 to 4.096) - 2.048 = -2.048 to 2.048 ****/
    return input;
}

unsigned int convertPulse(unsigned int input) {
    return input;
}
```

End convert.c

B.7 Mother Board Microcontroller Program

This code is based on and developed from TCPIP Demo App and the TCPIP Stack from Microchip Technology, Inc. as published in "Microchip TCPIP Stack v5.10 Installer.zip." Only the code which has been modified from the original TCPIP stack will be documented here, although the source code cannot be provided due to the copyright and license restrictions. A full copy of the program may be requested from the author via email at martin@ohmes.com.

B.7.1 main.c

This is the main program, or entry point, for the mother board's microcontroller.

Begin main.c

```
#define THIS_IS_STACK_APPLICATION 1
#define _18F97J60

/************ Header Files ************/
#include <p18F97J60.h>
#include "main.h"
#include <stdlib.h>
#include "i2c.h"
#include "config.h"
#include "TCPIPConfig.h"
#include "TCPIP.h"
#include "lcd.h"
#include "ds2482.h"
#include "private.h"
#include "mpfd3d.h"
#include "GenericTypeDefs.h"
#include "XEEPROM.h"
#include "temp.h"

APP_CONFIG AppConfig;
BYTE AN0String[8];
//BYTE myDHCPBindCount = 0xFF;
static volatile BOOL SyncTmr = TRUE;
//static volatile BOOL SyncTmrLong = TRUE;
static volatile BOOL SecondTimer3 = TRUE;
static volatile unsigned char InProgrammingMode = 2u;
BOOL SyncTimer(void);
//BOOL SyncTimerLong(void);
void ResetSyncTimerShort(void);
//void ResetSyncTimerLong(void);
```

```
BOOL GetSecondTimer3(void);
void ResetSecondTimer3(void);

#pragma interruptlow LowISR
void LowISR(void) {
    TickUpdate();
    if (Shutdown_Mode_Selected()) {
        OFF_LED_OUT = TickSecondsIsEven();
    }
}

#pragma interruptlow HighISR
void HighISR(void) {
    if (PIR3bits.TMR4IF) {
        T4CONbits.TMR4ON = 0; // Timer4 off
        PIR3bits.TMR4IF = 0; // Clear flag
        SyncTmr = FALSE;
    }
    if (PIR1bits.TMR2IF) {
        T2CONbits.TMR2ON = 0; // Timer2 off
        PIR1bits.TMR2IF = 0; // Clear flag
        TRIGGER_TRIS = 0;
    }
    if (PIR2bits.TMR3IF) {
        T3CONbits.TMR3ON = 0; // Stops Timer3
        PIR2bits.TMR3IF = 0; // Clear flag
        SecondTimer3 = FALSE;
        TRIGGER_TRIS = 0;
    }
    if (INTCONbits.INT0IF) {
        if (InProgrammingMode == 0u) {
            OFFButtonPressed();
        }
        INTCONbits.INT0IF = 0;
    }
}

#pragma code lowVector=0x18
void LowVector(void){_asm goto LowISR _endasm}
#pragma code highVector=0x8
void HighVector(void){_asm goto HighISR _endasm}
#pragma code // Return to default code section

IP_ADDR GetIP(void);
MAC_ADDR GetMAC(void);

void main(void) {
    static TICK t = 0u;
    static DWORD dwLastIP = 0u;
    unsigned char uc0 = 0u;
    DWORD Timer;
```

```
InProgrammingMode = 2u;
CONFIG_Init();
SSP1ADD = 0x3D;
OpenI2C1(MASTER, SLEW_OFF); // 100 kHz

INTCON2bits.RBPU = 1; // No pullup

// OFF Button
INTCON2bits.INTEDG0 = 1; // Rising Edge
INTCONbits.INT0IF = 0; // Clear Flag
INTCONbits.INT0IE = 1; // Enable

Init_ReadWrite();

StandbyLED_OFF;
PowerLED_ON;

// Initialize stack−related hardware components that may be
// required by the UART configuration routines
TickInit();

// Initialize LCD
lcdinit();

/**** Load MAC Address ****/
LoadMACfromDS2502();

/**** Initialize Microchip File System ****/
MPFSInit();

// Initialize Stack and application related NV variables into AppConfig.
InitAppConfig();

OSCTUNE = 0x40;

// Enable Interrupts
RCONbits.IPEN = 1; // Enable interrupt priorities
INTCONbits.GIEH = 1;
INTCONbits.GIEL = 1;

if (OFF_BUTTON_IN_BOOL) {
    StandbyLED_ON;
    ChangeLCDScreen(LCD_NONE);
    lcdPrint(LCD_ClearScreen);
    // 12345678901234567890
    lcdString(" Release ");
    lcdString(" Standby Button ");
    lcdString(" to enter ");
    lcdString(" PROGRAMMING MODE ");
    while (InProgrammingMode == 2u) {
```

```
            ResetSecondTimer3();
            while ((!OFF_BUTTON_IN_BOOL) && (PIR2bits.TMR3IF == 0u) && (SecondTimer3));
            if (!OFF_BUTTON_IN_BOOL) {
                InProgrammingMode = 1u;
            }
        }
        ChangeLCDScreen(LCD_NONE);
        lcdPrint(LCD_ClearScreen);
        // 12345678901234567890
        lcdString(" MPFD3-T 3D System ");
        lcdString(" now in ");
        lcdString(" PROGRAMMING MODE ");
        lcdString("Press Standby to End");
        while (!OFF_BUTTON_IN_BOOL) {
            GetTickCopy();
            ON_LED_OUT = TickSecondsIsEven();
            OFF_LED_OUT = TickSecondsIsEven();
        }
        POWER_OFF_OUT = 1;
    }
    InProgrammingMode = 0u;

    StackInit();
    MPFD3DInit();
    RTCInit();
    InitTemp();

    #if defined(STACK_USE_DYNAMICDNS_CLIENT)
        DDNSClient.Host.szROM = (ROM BYTE*)DDNSHost;
        DDNSClient.Username.szROM = (ROM BYTE*)DDNSUsername;
        DDNSClient.Password.szROM = (ROM BYTE*)DDNSPassword;
        DDNSClient.ROMPointers.Host = 1;
        DDNSClient.ROMPointers.Username = 1;
        DDNSClient.ROMPointers.Password = 1;
        DDNSSetService(DYNDNS_ORG);
    #endif

    while (1) {
        lcdTask();

        // This task performs normal stack task including checking
        // for incoming packet, type of packet and calling
        // appropriate stack entity to process it.
        StackTask();

        // This tasks invokes each of the core stack application tasks
        StackApplications();

        MPFD3DTask();

        // Process application specific tasks here.
```

```
// For this demo app, this will include the Generic TCP
// client and servers, and the SNMP, Ping, and SNMP Trap
// demos. Following that, we will process any IO from
// the inputs on the board itself.
// Any custom modules or processing you need to do should
// go here.
#if defined(STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE)
GenericTCPClient();
#endif

#if defined(STACK_USE_GENERIC_TCP_SERVER_EXAMPLE)
GenericTCPServer();
#endif

#if defined(STACK_USE_SMTP_CLIENT)
SMTPDemo();
#endif

#if defined(STACK_USE_SNMP_SERVER) && !defined(SNMP_TRAP_DISABLED)
SNMPTrapDemo();
if(gSendTrapFlag)
    SNMPSendTrap();
#endif

#if defined(STACK_USE_BERKELEY_API)
BerkeleyTCPClientDemo();
BerkeleyTCPServerDemo();
BerkeleyUDPClientDemo();
#endif

// If the local IP address has changed (ex: due to DHCP lease change)
// write the new IP address to the LCD display, UART, and Announce
// service
if(dwLastIP != AppConfig.MyIPAddr.Val) {
    dwLastIP = AppConfig.MyIPAddr.Val;

    #if defined(STACK_USE_UART)
        putrsUART((ROM char*)"\r\nNew IP Address: ");
    #endif

    #if defined(STACK_USE_UART)
        putrsUART((ROM char*)"\r\n");
    #endif

    #if defined(STACK_USE_ANNOUNCE)
        AnnounceIP();
    #endif

    ChangeLCDScreen(LCD_MY_IP);
    LCDForceUpdate();
}
```

```
    }
}

/*********************************************************************
 * Function: void InitAppConfig(void)
 *
 * PreCondition: MPFSInit() is already called.
 *
 * Input: None
 *
 * Output: Write/Read non-volatile config variables.
 *
 * Side Effects: None
 *
 * Overview: None
 *
 * Note: None
 ********************************************************************/
// MAC Address Serialization using a MPLAB PM3 Programmer and
// Serialized Quick Turn Programming (SQTP).
// The advantage of using SQTP for programming the MAC Address is it
// allows you to auto-increment the MAC address without recompiling
// the code for each unit. To use SQTP, the MAC address must be fixed
// at a specific location in program memory. Uncomment these two pragmas
// that locate the MAC address at 0x1FFF0. Syntax below is for MPLAB C
// Compiler for PIC18 MCUs. Syntax will vary for other compilers.
//#pragma romdata MACROM=0x1FFF0
//static ROM BYTE SerializedMACAddress[6] = {MY_DEFAULT_MAC_BYTE1,  ...
      MY_DEFAULT_MAC_BYTE2, MY_DEFAULT_MAC_BYTE3, MY_DEFAULT_MAC_BYTE4,  ...
      MY_DEFAULT_MAC_BYTE5, MY_DEFAULT_MAC_BYTE6};
static BYTE SerializedMACAddress[6] = {MY_DEFAULT_MAC_BYTE1,  ...
      MY_DEFAULT_MAC_BYTE2, MY_DEFAULT_MAC_BYTE3, MY_DEFAULT_MAC_BYTE4,  ...
      MY_DEFAULT_MAC_BYTE5, MY_DEFAULT_MAC_BYTE6};
//#pragma romdata

static void InitAppConfig(void) {
    BYTE c;

    AppConfig.Flags.bIsDHCPEnabled = TRUE;
    AppConfig.Flags.bInConfigMode = TRUE;
    memcpy((void*)&AppConfig.MyMACAddr, (void*)SerializedMACAddress, sizeof(AppConfig.  ...
        MyMACAddr));
    AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 |  ...
        MY_DEFAULT_IP_ADDR_BYTE2<<8ul | MY_DEFAULT_IP_ADDR_BYTE3<<16ul |  ...
        MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
    AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
    AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 | MY_DEFAULT_MASK_BYTE2<<8  ...
        ul | MY_DEFAULT_MASK_BYTE3<<16ul | MY_DEFAULT_MASK_BYTE4<<24ul;
    AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
    AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 | MY_DEFAULT_GATE_BYTE2  ...
        <<8ul | MY_DEFAULT_GATE_BYTE3<<16ul | MY_DEFAULT_GATE_BYTE4<<24ul;
```

```
AppConfig.PrimaryDNSServer.Val = MY_DEFAULT_PRIMARY_DNS_BYTE1 |  ...
    MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul | MY_DEFAULT_PRIMARY_DNS_BYTE3  ...
    <<16ul | MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1 |  ...
    MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul |  ...
    MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul |  ...
    MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;

// SNMP Community String configuration
#if defined(STACK_USE_SNMP_SERVER)
    BYTE i;
    static ROM char * ROM cReadCommunities[] = SNMP_READ_COMMUNITIES;
    static ROM char * ROM cWriteCommunities[] = SNMP_WRITE_COMMUNITIES;
    ROM char * strCommunity;

    for(i = 0; i < SNMP_MAX_COMMUNITY_SUPPORT; i++) {
        // Get a pointer to the next community string
        strCommunity = cReadCommunities[i];
        if(i >= sizeof(cReadCommunities)/sizeof(cReadCommunities[0]))
            strCommunity = "";

        // Ensure we don't buffer overflow. If your code gets stuck here,
        // it means your SNMP_COMMUNITY_MAX_LEN definition in TCPIPConfig.h
        // is either too small or one of your community string lengths
        // (SNMP_READ_COMMUNITIES) are too large. Fix either.
        if(strlenpgm(strCommunity) >= sizeof(AppConfig.readCommunity[0]))
            while(1);

        // Copy string into AppConfig
        strcpypgm2ram((char*)AppConfig.readCommunity[i], strCommunity);

        // Get a pointer to the next community string
        strCommunity = cWriteCommunities[i];
        if(i >= sizeof(cWriteCommunities)/sizeof(cWriteCommunities[0]))
            strCommunity = "";

        // Ensure we don't buffer overflow. If your code gets stuck here,
        // it means your SNMP_COMMUNITY_MAX_LEN definition in TCPIPConfig.h
        // is either too small or one of your community string lengths
        // (SNMP_WRITE_COMMUNITIES) are too large. Fix either.
        if(strlenpgm(strCommunity) >= sizeof(AppConfig.writeCommunity[0]))
            while(1);

        // Copy string into AppConfig
        strcpypgm2ram((char*)AppConfig.writeCommunity[i], strCommunity);
    }
#endif

// Load the default NetBIOS Host Name
memcpypgm2ram(AppConfig.NetBIOSName, (ROM void*)MY_DEFAULT_HOST_NAME, 16);
FormatNetBIOSName(AppConfig.NetBIOSName);
```

```
}

IP_ADDR GetIP(void) {
    return AppConfig.MyIPAddr;
}

MAC_ADDR GetMAC(void) {
    return AppConfig.MyMACAddr;
}

void LoadMACfromDS2502(void) {
    unsigned char uc0, uc1;
    unsigned char uca0[0x0Du];

    for (uc1 = 0x00u; uc1 <= 0x0Cu; ++uc1) {
        uca0[uc1] = 0xFFu;
    }
    DS2482_detect(MAC_I2C_ADDR);
    if (OWReset() == 0x01u) {
    }
    OWWriteByte(0xCCu);
    OWWriteByte(0xF0u);
    OWWriteByte(0x00u);
    OWWriteByte(0x00u);
    uc0 = OWReadByte(); // Get CRC
    OWBlock(uca0, 0x0Du);
    uc0 = 0x00u;
    for (uc1 = 0x0Au; uc1 >= 0x05u; uc1--) {
        SerializedMACAddress[uc0] = uca0[uc1];
        uc0++;
    }
}

WORD HTTPGetVar(BYTE var, WORD ref, BYTE *val) {
    // Identify variable.
    // Is it RB5 ?
    if ( var == 4u ) {
        // We will simply return 1 if RB5 is high,
        // or 0 if low.
        if ( PORTBbits.RB5 )
            *val = '1';
        else
            *val = '0';
        // Tell HTTP that this is last byte of
        // variable value.
        return HTTP_END_OF_VAR;
    } else {
    // Check for other variables...
    //...
    }
}
```

```
void HTTPExecCmd(BYTE *argv, BYTE argc) {
    BYTE i;
    // Go through each parameters for current form command.
    // We are skipping form action name, so i starts at 1...
    for (i = 1; i < argc; i++ ) {
        // Identify parameter.
        if ( argv[i] == 'P' ) { // Is this power level?
            // Next parameter is the Power level value.
            //PowerLevel = atoi(argv[++i]);
        } else if ( argv[i] == 'L' ) { // Is this Low Power Setting?
            //LowPowerSetting = atoi(argv[++i]);
        } else if ( argv[i] == 'H' ) { // Is this High Power Setting?
            //HighPowerSetting = atoi(argv[++i]);
        }
        // If another page is to be displayed as a result of this command, copy
        // its upper case name into argv[0]
        // strcpy(argv[0], RESULTS.CGI);
    }
}

BOOL SyncTimer(void) {
    return SyncTmr;
}

void ResetSyncTimerShort(void) {
    TMR4 = 0x00u; // Timer4 Register
    SyncTmr = TRUE;
    T4CONbits.TMR4ON = 1; // Timer4 on
}

BOOL GetSecondTimer3(void) {
    return SecondTimer3;
}

void ResetSecondTimer3(void) {
    T3CONbits.TMR3ON = 0; // Stops Timer3
    TMR3H = 0x80u;
    TMR3L = 0x00u; // Set for 1 second timeout
    SecondTimer3 = TRUE;
    T3CONbits.TMR3ON = 1; // Start Timer3
}

void MPFD_Write_Done(void) {
    /*** Set Write high and wait for line to read high ***/
    while (SyncTmr && TRIGGER_IN_BOOL);
    WRITE_TRIS = 1;

    TMR2 = 0x00u; // Timer2 Register
    T2CONbits.TMR2ON = 1; // Timer2 on
```

```
    while ((!WRITE_IN_BOOL) && TRIGGER_IN_BOOL);
    T2CONbits.TMR2ON = 0; // Timer2 on
    READ_TRIS = 0;

    TMR4 = 0x00u; // Timer4 Register
    SyncTmr = TRUE;
    T4CONbits.TMR4ON = 1; // Timer4 on
}

void MPFD_Read_Done(void) {
    /*** Set Read high and wait for line to read high ***/
    while (SyncTmr && TRIGGER_IN_BOOL);
    READ_TRIS = 1;

    TMR2 = 0x00u; // Timer2 Register
    T2CONbits.TMR2ON = 1; // Timer2 on

    while ((!READ_IN_BOOL) && TRIGGER_IN_BOOL);
    T2CONbits.TMR2ON = 0; // Timer2 on
    WRITE_TRIS = 0;

    TMR4 = 0x00u; // Timer4 Register
    SyncTmr = TRUE;
    T4CONbits.TMR4ON = 1; // Timer4 on
}
```

End main.c

B.7.2 main.h

This is the header file for the program main.c (Section §B.7.1) of the mother board's microcontroller.

Begin main.h

```
#ifndef __main_H
#define __main_H

#include <P18F97J60.H>

/************ Configuration bits ************/
/* Background Debugger Enable Bit */
#pragma config DEBUG = ON

/* Extended Instruction Set Enable Bit */
#pragma config XINST = ON
//#pragma config XINST = OFF

/* Stack Overflow/Underflow Reset Enable Bit */
```

```
#pragma config STVR = ON

/* Watchdog Timer Enable Bit */
#pragma config WDT = OFF

/* Code Protection Bit */
#pragma config CP0 = OFF

/* Fail-Safe Clock Monitor Enable Bit */
#pragma config FCMEN = OFF

/* Two-Speed Start-Up */
#pragma config IESO = OFF

/* Default/Reset System Clock Select Bit */
#pragma config FOSC2 = ON
//OSCCON |= 0x01; // System clock tied to Timer1

/* Oscillator Selection bits */
#pragma config FOSC = EC

/* Watchdog Timer Postscale Select bits */
#pragma config WDTPS = 1

/* External Bus Wait Enable bit */
#pragma config WAIT = OFF

/* Data Bus Width Select bit */
#pragma config BW = 8

/* External Memory Bus Configuration bits */
#pragma config MODE = MM

/* External Address Bus Shift Enable bit */
#pragma config EASHFT = OFF

/* Ethernet LED Enable bit */
#pragma config ETHLED = ON

/* ECCP MUX bit */
#pragma config ECCPMX = OFF

/* ECCP2 MUX bit */
#pragma config CCP2MX = OFF

#if !defined(THIS_IS_STACK_APPLICATION)
    extern BYTE AN0String[8];
#endif

static void InitAppConfig(void);
void LoadMACfromDS2502(void);
```

```
#endif // __main_H
```

End main.h

B.7.3 config.h

This is the header file for the program config.c (Section §B.7.4) of the mother board's microcontroller.

Begin config.h

```
#ifndef __config_H
#define __config_H

#include <P18F97J60.H>

// PORT − READ IN
// LAT − WRITE OUT
// TRIS IN/OUT
#define IN 1
#define OUT 0

#define TRL1_TRIS (TRISAbits.TRISA2) // IN
#define TRL1_IN ((unsigned char)PORTAbits.RA2)
#define TRJ1_TRIS (TRISAbits.TRISA3) // IN
#define TRJ1_IN ((unsigned char)PORTAbits.RA3)
#define TRM2_TRIS (TRISAbits.TRISA4) // IN
#define TRM2_IN ((unsigned char)PORTAbits.RA4)
#define TRM3_TRIS (TRISAbits.TRISA5) // IN
#define TRM3_IN ((unsigned char)PORTAbits.RA5)

#define OFF_BUTTON_TRIS (TRISBbits.TRISB0) // IN
#define OFF_BUTTON_IN ((unsigned char)PORTBbits.RB0)
#define OFF_BUTTON_IN_BOOL (PORTBbits.RB0)
#define NotUsedB1_TRIS (TRISBbits.TRISB1) // OUT (SET_READY)
#define NotUsedB1_OUT (LATBbits.LATB1)
#define READ_TRIS (TRISBbits.TRISB2) // IN/OUT (READY)
#define READ_OUT (LATBbits.LATB2)
#define READ_IN ((unsigned char)PORTBbits.RB2)
#define READ_IN_BOOL (PORTBbits.RB2)
#define WRITE_TRIS (TRISBbits.TRISB3) // IN/OUT (DONE)
#define WRITE_OUT (LATBbits.LATB3)
#define WRITE_IN ((unsigned char)PORTBbits.RB3)
#define WRITE_IN_BOOL (PORTBbits.RB3)

#define ON_LED_TRIS (TRISBbits.TRISB4) // OUT
#define ON_LED_OUT (LATBbits.LATB4)
#define OFF_LED_TRIS (TRISBbits.TRISB5) // OUT
```

```
#define OFF_LED_OUT (LATBbits.LATB5)

#define RTC_TRIS (TRISCbits.TRISC0) // IN
#define RTC_IN ((unsigned char)PORTCbits.RC0)
#define TRM1_TRIS (TRISCbits.TRISC1) // IN
#define TRM1_IN ((unsigned char)PORTCbits.RC1)
#define TRP1_TRIS (TRISCbits.TRISC2) // IN
#define TRP1_IN ((unsigned char)PORTCbits.RC2)
#define SCL_TRIS (TRISCbits.TRISC3) // IN/OUT
#define SCL_IN ((unsigned char)PORTCbits.RC3)
#define SCL_OUT (LATCbits.LATC3)
#define SDA_TRIS (TRISCbits.TRISC4) // IN/OUT
#define SDA_IN ((unsigned char)PORTCbits.RC4)
#define SDA_OUT (LATCbits.LATC4)
#define UPS_AC_GOOD_TRIS (TRISCbits.TRISC5) // IN
#define UPS_AC_GOOD_IN ((unsigned char)PORTCbits.RC5)
#define TRN3_TRIS (TRISCbits.TRISC6) // IN
#define TRN3_IN ((unsigned char)PORTCbits.RC6)
#define TRN2_TRIS (TRISCbits.TRISC7) // IN
#define TRN2_IN ((unsigned char)PORTCbits.RC7)

#define TRB1_TRIS (TRISDbits.TRISD0) // IN
#define TRB1_IN ((unsigned char)PORTDbits.RD0)
#define TRA3_TRIS (TRISDbits.TRISD1) // IN
#define TRA3_IN ((unsigned char)PORTDbits.RD1)
#define TRA2_TRIS (TRISDbits.TRISD2) // IN
#define TRA2_IN ((unsigned char)PORTDbits.RD2)
#define TRA1_TRIS (TRISDbits.TRISD3) // IN
#define TRA1_IN ((unsigned char)PORTDbits.RD3)
#define MWP_TRIS (TRISDbits.TRISD4) // IN/OUT
#define MWP_IN ((unsigned char)PORTDbits.RD4)
#define MWP_OUT (LATDbits.LATD4)
#define MSDA_TRIS (TRISDbits.TRISD5) // IN/OUT
#define MSDA_IN ((unsigned char)PORTDbits.RD5)
#define MSDA_OUT (LATDbits.LATD5)
#define MSCL_TRIS (TRISDbits.TRISD6) // IN/OUT
#define MSCL_IN ((unsigned char)PORTDbits.RD6)
#define MSCL_OUT (LATDbits.LATD6)
#define RESET_ALL_TRIS (TRISDbits.TRISD4) // OUT
#define RESET_ALL_OUT (LATDbits.LATD4)

#define TW_TRIS (TRISEbits.TRISE0) // OUT
#define TW_OUT (LATEbits.LATE0)
#define TRE1_TRIS (TRISEbits.TRISE1) // IN
#define TRE1_IN ((unsigned char)PORTEbits.RE1)
#define TRD3_TRIS (TRISEbits.TRISE2) // IN
#define TRD3_IN ((unsigned char)PORTEbits.RE2)
#define TRC1_TRIS (TRISEbits.TRISE3) // IN
#define TRC1_IN ((unsigned char)PORTEbits.RE3)
#define TRC2_TRIS (TRISEbits.TRISE4) // IN
#define TRC2_IN ((unsigned char)PORTEbits.RE4)
```

```
#define TRC3_TRIS (TRISEbits.TRISE5) // IN
#define TRC3_IN ((unsigned char)PORTEbits.RE5)
#define TRB3_TRIS (TRISEbits.TRISE6) // IN
#define TRB3_IN ((unsigned char)PORTEbits.RE6)
#define TRB2_TRIS (TRISEbits.TRISE7) // IN
#define TRB2_IN ((unsigned char)PORTEbits.RE7)

#define TRF2_TRIS (TRISFbits.TRISF0) // IN
#define TRF2_IN ((unsigned char)PORTFbits.RF0)
#define TRJ2_TRIS (TRISFbits.TRISF1) // IN
#define TRJ2_IN ((unsigned char)PORTFbits.RF1)
#define TRH1_TRIS (TRISFbits.TRISF2) // IN
#define TRH1_IN ((unsigned char)PORTFbits.RF2)
#define TRH2_TRIS (TRISFbits.TRISF3) // IN
#define TRH2_IN ((unsigned char)PORTFbits.RF3)
#define TRH3_TRIS (TRISFbits.TRISF4) // IN
#define TRH3_IN ((unsigned char)PORTFbits.RF4)
#define TRG3_TRIS (TRISFbits.TRISF5) // IN
#define TRG3_IN ((unsigned char)PORTFbits.RF5)
#define TRG2_TRIS (TRISFbits.TRISF6) // IN
#define TRG2_IN ((unsigned char)PORTFbits.RF6)
#define TRG1_TRIS (TRISFbits.TRISF7) // IN
#define TRG1_IN ((unsigned char)PORTFbits.RF7)

#define MMCU_WD_TRIS (TRISGbits.TRISG0) // OUT
#define MMCU_WD_OUT (LATGbits.LATG0)
#define POWER_OFF_TRIS (TRISGbits.TRISG1) // OUT
#define POWER_OFF_OUT (LATGbits.LATG1)
#define TRP2_TRIS (TRISGbits.TRISG2) // IN
#define TRP2_IN ((unsigned char)PORTGbits.RG2)
#define TRP3_TRIS (TRISGbits.TRISG3) // IN
#define TRP3_IN ((unsigned char)PORTGbits.RG3)
#define TRF3_TRIS (TRISGbits.TRISG4) // IN
#define TRF3_IN ((unsigned char)PORTGbits.RG4)
#define TRF1_TRIS (TRISGbits.TRISG5) // IN
#define TRF1_IN ((unsigned char)PORTGbits.RG5)
#define NotUsedG6_TRIS (TRISGbits.TRISG6) // OUT (SET_DONE)
#define NotUsedG6_OUT (LATGbits.LATG6)
#define TRL2_TRIS (TRISGbits.TRISG7) // IN
#define TRL2_IN ((unsigned char)PORTGbits.RG7)

#define TRD2_TRIS (TRISHbits.TRISH0) // IN
#define TRD2_IN ((unsigned char)PORTHbits.RH0)
#define TRD1_TRIS (TRISHbits.TRISH1) // IN
#define TRD1_IN ((unsigned char)PORTHbits.RH1)
#define TRE3_TRIS (TRISHbits.TRISH2) // IN
#define TRE3_IN ((unsigned char)PORTHbits.RH2)
#define TRE2_TRIS (TRISHbits.TRISH3) // IN
#define TRE2_IN ((unsigned char)PORTHbits.RH3)
#define TRJ3_TRIS (TRISHbits.TRISH4) // IN
#define TRJ3_IN ((unsigned char)PORTHbits.RH4)
```

```
#define TRI1_TRIS (TRISHbits.TRISH5) // IN
#define TRI1_IN ((unsigned char)PORTHbits.RH5)
#define TRI2_TRIS (TRISHbits.TRISH6) // IN
#define TRI2_IN ((unsigned char)PORTHbits.RH6)
#define TRI3_TRIS (TRISHbits.TRISH7) // IN
#define TRI3_IN ((unsigned char)PORTHbits.RH7)

#define TRO2_TRIS (TRISJbits.TRISJ0) // IN
#define TRO2_IN ((unsigned char)PORTJbits.RJ0)
#define TRO1_TRIS (TRISJbits.TRISJ1) // IN
#define TRO1_IN ((unsigned char)PORTJbits.RJ1)
#define REACTOR_PULSE_TRIS (TRISJbits.TRISJ2) // IN
#define REACTOR_PULSE_IN ((unsigned char)PORTJbits.RJ2)
#define REACTOR_ON_TRIS (TRISJbits.TRISJ3) // IN
#define REACTOR_ON_IN ((unsigned char)PORTJbits.RJ3)
#define TRN1_TRIS (TRISJbits.TRISJ4) // IN
#define TRN1_IN ((unsigned char)PORTJbits.RJ4)
#define TRO3_TRIS (TRISJbits.TRISJ5) // IN
#define TRO3_IN ((unsigned char)PORTJbits.RJ5)
#define TRIGGER_TRIS (TRISJbits.TRISJ6) // OUT/IN
#define TRIGGER_OUT (LATJbits.LATJ6)
#define TRIGGER_IN ((unsigned char)PORTJbits.RJ6)
#define TRIGGER_IN_BOOL (PORTJbits.RJ6)
#define TRL3_TRIS (TRISJbits.TRISJ7) // IN
#define TRL3_IN ((unsigned char)PORTJbits.RJ7)

/************** MACROS **************/
#define PowerLED_ON ON_LED_OUT = 0;
#define PowerLED_OFF ON_LED_OUT = 1;
#define StandbyLED_ON OFF_LED_OUT = 1;
#define StandbyLED_OFF OFF_LED_OUT = 0;

/************** I2C Addresses **************/
#define RTC_READ 0b11010001u
#define RTC_WRITE 0b11010000u
#define RTC_I2C_ADDR 0b11010000u

#define MEM_READ_B0 0b10100111u // first 512k
#define MEM_WRITE_B0 0b10100110u // first 512k
#define MEM_READ_B1 0b10101111u // second 512k
#define MEM_WRITE_B1 0b10101110u // second 512k

#define MAC_READ 0b00110001u // DS2482 to DS2502
#define MAC_WRITE 0b00110000u
#define MAC_I2C_ADDR 0b00110000u

#define LCD_READ 0b10010001u // LCD through I2C-UART Link
#define LCD_WRITE 0b10010000u

#define TEMP_I2C_ADDR 0b10011010u // TC74A5
```

```
void CONFIG_Init(void);
void Init_ReadWrite(void);

#endif
```

End config.h

B.7.4 config.c

This is the config program for the mother board's microcontroller.

Begin config.c

```
#include "config.h"
#include "lcd.h"
#include "GenericTypeDefs.h"
#include "Tick.h"
#include "SNTP.h"
#include "mpfd3d.h"

void CONFIG_Init(void) {
    POWER_OFF_OUT = 0;
    POWER_OFF_TRIS = OUT;
    OFF_BUTTON_TRIS = IN;
    ON_LED_TRIS = OUT;
    ON_LED_OUT = 1;
    OFF_LED_TRIS = OUT;
    OFF_LED_OUT = 1;

    RTC_TRIS = IN;
    RESET_ALL_OUT = 1;
    RESET_ALL_TRIS = OUT;

    NotUsedB1_TRIS = OUT;
    NotUsedB1_OUT = 1;
    TRIGGER_OUT = 0;
    TRIGGER_TRIS = 0;
    READ_OUT = 0;
    READ_TRIS = 0;
    WRITE_OUT = 0;
    WRITE_TRIS = 0;
    NotUsedG6_TRIS = OUT;
    NotUsedG6_OUT = 1;

    UPS_AC_GOOD_TRIS = IN;
    REACTOR_PULSE_TRIS = IN;
    REACTOR_ON_TRIS = IN;

    SCL_TRIS = IN;
    SCL_OUT = 0;
```

```
SDA_TRIS = IN;
SDA_OUT = 0;

MWP_TRIS = OUT;
MWP_OUT = 1; // Must be kept low for write operations
MSDA_TRIS = IN;
MSDA_OUT = 0;
MSCL_TRIS = IN;
MSCL_OUT = 0;

MMCU_WD_TRIS = OUT;

TW_TRIS = 0;
TW_OUT = 0;

TRA1_TRIS = IN;
TRA2_TRIS = IN;
TRA3_TRIS = IN;

TRB1_TRIS = IN;
TRB2_TRIS = IN;
TRB3_TRIS = IN;

TRC1_TRIS = IN;
TRC2_TRIS = IN;
TRC3_TRIS = IN;

TRD1_TRIS = IN;
TRD2_TRIS = IN;
TRD3_TRIS = IN;

TRE1_TRIS = IN;
TRE2_TRIS = IN;
TRE3_TRIS = IN;

TRF1_TRIS = IN;
TRF2_TRIS = IN;
TRF3_TRIS = IN;

TRG1_TRIS = IN;
TRG2_TRIS = IN;
TRG3_TRIS = IN;

TRH1_TRIS = IN;
TRH2_TRIS = IN;
TRH3_TRIS = IN;

TRI1_TRIS = IN;
TRI2_TRIS = IN;
TRI3_TRIS = IN;
```

```
    TRJ1_TRIS = IN;
    TRJ2_TRIS = IN;
    TRJ3_TRIS = IN;

    TRL1_TRIS = IN;
    TRL2_TRIS = IN;
    TRL3_TRIS = IN;

    TRM1_TRIS = IN;
    TRM2_TRIS = IN;
    TRM3_TRIS = IN;

    TRN1_TRIS = IN;
    TRN2_TRIS = IN;
    TRN3_TRIS = IN;

    TRO1_TRIS = IN;
    TRO2_TRIS = IN;
    TRO3_TRIS = IN;

    TRP1_TRIS = IN;
    TRP2_TRIS = IN;
    TRP3_TRIS = IN;

    ADCON0bits.ADON = 0;
    ADCON1 = 0b00001111;
}

void Init_ReadWrite(void) {
    READ_OUT = 0;
    READ_TRIS = 0;
    WRITE_OUT = 0;
    WRITE_TRIS = 0;
    TRIGGER_OUT = 0;
    TRIGGER_TRIS = 0;

    RCONbits.IPEN = 1; // Enale priority levels on interrups
    INTCON2bits.RBPU = 1; // Disable PORTB pull−ups

    // READ
    INTCON3bits.INT2IF = 0; // Clear Interrupt Flag
    INTCON2bits.INTEDG2 = 1; // Interrupt on rising edge
    INTCON3bits.INT2IP = 1; // High priority
    INTCON3bits.INT2IE = 0; // Disable Interrupt

    // WRITE
    INTCON3bits.INT3IF = 0; // Clear Interrupt Flag
    INTCON2bits.INTEDG3 = 1; // Interrupt on rising edge
    INTCON2bits.INT3IP = 1; // High priority
    INTCON3bits.INT3IE = 0; // Disable Interrupt
}
```

End config.c

B.7.5 ds2482.c and ds2482.h

The driver ds2482.c and ds2482.h contain the necessary code for communication with Maxim's DS2482 I^2C to 1-wire bus converter. This code is from the application note 3684 from Maxim [29].

B.7.6 HardwareProfile.h

This is the header file which works with the program config.c (Section §B.7.4) of the mother board's microcontroller.

Begin HardwareProfile.h

```
#ifndef __HARDWARE_PROFILE_H
#define __HARDWARE_PROFILE_H

#include "GenericTypeDefs.h"
#include "Compiler.h"
#include "config.h"

#define GetSystemClock() (25000000ul) // Hz
#define GetInstructionClock() (GetSystemClock()/4)
#define GetPeripheralClock() GetInstructionClock()

#define EEPROM_SCL_TRIS MSCL_TRIS
#define EEPROM_SDA_TRIS MSDA_TRIS
#define EEPROM_SCL_IO MSCL_OUT
#define EEPROM_SDA_IO MSDA_TRIS
#define EEPROM_SPICON2bits SSP2CON2bits
#define MPFS_EEPROM_CLOCK 0x0E // SSP2ADD
#define EEPROM_CONTROL MEM_WRITE_B0

#endif
```

End HardwareProfile.h

B.7.7 lcd.h

This is the header file for the program lcd.c (Section §B.7.8) of the mother board's microcontroller.

Begin lcd.h

```
#ifndef __lcd_H
#define __lcd_H

#define LCD_AutoScrollOn 0x51u
#define LCD_AutoScrollOff 0x52u
#define LCD_ClearScreen 0x58u
#define LCD_AutoLineWrapOn 0x43u
#define LCD_AutoLineWrapOff 0x44u
#define LCD_AutoTransmitKeyPressesOn 0x41u
#define LCD_PollKeys 0x26u
#define LCD_ClearKeyBuffer 0x45u
#define LCD_SetDebounce 0x55u
#define LCD_KeyAutoRepeat 0x7Eu
#define LCD_KeyAutoRepeatOff 0x60u
#define LCD_SetSaveBrightness 0x98u
#define LCD_SetSaveContrast 0x91u
#define LCD_SetContrast 0x50u
#define LCD_DisplayOn 0x42u
#define LCD_SetRemember 0x93u
#define LCD_ChangeBaudRate 0x39u

#define LCD_ChA 0b00000000u
//#define LCD_ChB 0b00000000u // for debugging send to terminal output
#define LCD_ChB 0b00000010u

//#define LCD_DEBUG
//#define LCD_SERIAL_REPEAT

typedef enum _LCD_SCREENS
{
    LCD_NONE,
    LCD_MY_IP,
    LCD_CONNECTION,
    LCD_BOARDS,
    LCD_HIGH_VOLTAGE,
    LCD_LLD,
    LCD_SPEED,
    LCD_TEMPERATURE,
    LCD_VIEWCURRENTDATALINES,
    LCD_LAST
} LCD_SCREENS;

void lcdwait(unsigned char secs);
void lcdinit(void);
void uartconfig(unsigned char addr, unsigned char chan, unsigned char reg, unsigned char data);
void lcdPrint(unsigned char uc0);
void lcd2Print(unsigned char data1,unsigned char data2);
void serialType(unsigned char data);
void lcdType(unsigned char data);
void lcdTypeHex(unsigned char data);
```

```
void serialString(const rom char * str);
void lcdString(const rom char * str);
void lcdStringRam(char * str);
void lcdSetKeys(void);
void lcdEmpty(void);
unsigned char uartSpace(unsigned char chan);
void lcdCursor(unsigned char col, unsigned char row);
void lcdTask(void);
void ChangeLCDScreen(LCD_SCREENS tmp);
void LCDForceUpdate(void);
LCD_SCREENS GetLCDScreen(void);
void showdatalines(void);
void lcdStartupScreen(void);
void serial_debug(unsigned char screen, unsigned char step);
void lcdDBsync(void);
void lcdDBsync1(void);
void lcdDBsync2(void);
void lcdDBsync3(void);

#endif // __lcd_H
```

End lcd.h

B.7.8 lcd.c

This is the lcd program for the mother board's microcontroller.

Begin lcd.c

```
#include "lcd.h"
#include "config.h"
#include "GenericTypeDefs.h"
#include "HardwareProfile.h"
#include "Tick.h"
#include "i2c.h"
#include "string.h"
#include "StackTsk.h"
#include "mpfd3d.h"
#include "temp.h"

extern IP_ADDR GetIP(void);
extern MAC_ADDR GetMAC(void);
extern WORD GetConnectedBoards(void);

/***** SC16IS752 General Register Set *****/
#define RHR 0x00u
#define THR 0x00u
#define IER 0x01u
#define FCR 0x02u
#define IIR 0x02u
```

```
#define LCR 0x03u
#define MCR 0x04u
#define LSR 0x05u
#define MSR 0x06u
#define SPR 0x07u
#define TCR 0x06u
#define TLR 0x07u
#define TXLVL 0x08u
#define RXLVL 0x09u
#define IODir 0x0Au
#define IOState 0x0Bu
#define IOIntEna 0x0Cu
#define IOControl 0x0Eu
#define EFCR 0x0Fu

/***** SC16IS752 Special Register Set *****/
#define DLL 0x00u
#define DLH 0x01u

/***** SC16IS752 Enhanced Register Set *****/
#define EFR 0x02u
#define XON1 0x04u
#define XON2 0x05u
#define XOFF1 0x06u
#define XOFF2 0x07u

static LCD_SCREENS lcd_current_screen = LCD_NONE;
static LCD_SCREENS lcd_last_screen = LCD_LAST;
static BOOL force_update = FALSE;
static unsigned char lcdstarted = 0u;
static DWORD LCD_Tick_Tran = 0u;
static DWORD LCD_Tick_Delay = 0u;
static DWORD LCD_Tick_Read = 0u;
static DWORD LCD_Lim_Tran = 0u;
static DWORD LCD_Lim_Delay = 0u;
static DWORD LCD_Lim_Read = 0u;
static unsigned char LCD_Contrast = 128u;

void lcdwait(unsigned char secs) {
    unsigned char uc0;
    DWORD Timer, Timer2;
    DWORD Timeout;

    Timeout = secs;
    Timeout *= TICK_SECOND;
    Timer = TickGet();
    uc0 = 0u;
    do {
        Timer2 = TickGet();
        if (Timer2 >= Timer) {
            if ((Timer2 - Timer) >= Timeout) {
```

```
                uc0 = 1u;
            }
        } else {
            if (((0xFFFFFFFFu - Timer) + Timer2) >= Timeout) {
                uc0 = 1u;
            }
        }
    } while (uc0 == 0u);
    return;
}

void lcdinit(void) {
    unsigned char uc0,uc1;

    lcdwait(1u);
    uartconfig(LCD_WRITE,LCD_ChA,LCR,0x00u); // LCR - 19200 baud, 8N1 - General Register ...
        Set
    uartconfig(LCD_WRITE,LCD_ChB,LCR,0x00u); // LCR - 19200 baud, 8N1 - General Register ...
        Set
    uartconfig(LCD_WRITE,LCD_ChA,IOControl,0b00000010u); // IOControl - Reset
    uc1 = 0;
    do {
        lcdwait(1u);
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA | (IOControl << 3u));
        RestartI2C1();
        WriteI2C1(LCD_READ);
        uc0 = ReadI2C1();
        NotAckI2C1();
        StopI2C1();
        ++uc1;
    } while (uc0 & 0x07u == 0x07u && uc1 < 10u);
    if (uc1 >= 10u) {
        StandbyLED_ON;
        PowerLED_OFF;
        POWER_OFF_OUT = 1;
    }

    uartconfig(LCD_WRITE,LCD_ChA,LCR,0x00u); // LCR - 19200 baud, 8N1 - General Register ...
        Set
    uartconfig(LCD_WRITE,LCD_ChB,LCR,0x00u); // LCR - 19200 baud, 8N1 - General Register ...
        Set
    uartconfig(LCD_WRITE,LCD_ChA,IOControl,0b00000000u); // IOControl - Set for GPIO
    uartconfig(LCD_WRITE,LCD_ChA,IODir,0xFFu); // IODir - Set all pins to outputs
    uartconfig(LCD_WRITE,LCD_ChA,IOState,0x00u); // IOState - Turn all LEDs ON

    lcdwait(1u);
    uartconfig(LCD_WRITE,LCD_ChA,LCR,0xBFu); // Set LCR for modification of Enhanced register ...
        set
    uartconfig(LCD_WRITE,LCD_ChA,EFR,0b00010000u); // Enable Hardware Flow control and ...
```

```
    Enanced Functions (EFR[4])
uartconfig(LCD_WRITE,LCD_ChA,LCR,0b10000011u); // LCR - 19200 baud, 8N1 - Special ...
    Register Set
uartconfig(LCD_WRITE,LCD_ChA,DLL,0x0Cu); // Low byte for division register (19200 buad)
uartconfig(LCD_WRITE,LCD_ChA,DLH,0x00u); // High byte for division register (19200 buad)
uartconfig(LCD_WRITE,LCD_ChA,LCR,0b00000011u); // LCR - 19200 baud, 8N1 - General ...
    Register Set
uartconfig(LCD_WRITE,LCD_ChA,IER,0x00u); // IER - Interrupt Enable Register
uartconfig(LCD_WRITE,LCD_ChA,FCR,0b00000110u); // FCR - FIFO Control Register ( ...
    Disabled)
lcdwait(1u);
uartconfig(LCD_WRITE,LCD_ChA,MCR,0b10000000u); // MCR - Modem Control Register ( ...
    Loopback)
uartconfig(LCD_WRITE,LCD_ChA,FCR,0b00000001u); // FCR - FIFO Control Register (Enabled ...
    )
uartconfig(LCD_WRITE,LCD_ChA,EFCR,0b00000000u); // EFCR - Extra Features Control ...
    Register

uartconfig(LCD_WRITE,LCD_ChB,LCR,0xBFu); // Set LCR for modification of Enhanced register ...
    set
uartconfig(LCD_WRITE,LCD_ChB,EFR,0b00010000u); // Enable Enanced Functions (EFR[4])
uartconfig(LCD_WRITE,LCD_ChB,LCR,0b10000011u); // LCR - 19200 baud, 8N1 - Special ...
    Register Set
uartconfig(LCD_WRITE,LCD_ChB,DLL,0x0Cu); // Low byte for division register (19200 buad)
uartconfig(LCD_WRITE,LCD_ChB,DLH,0x00u); // High byte for division register (19200 buad)
uartconfig(LCD_WRITE,LCD_ChB,LCR,0b00000011u); // LCR - 19200 baud, 8N1 - General ...
    Register Set
uartconfig(LCD_WRITE,LCD_ChB,IER,0x00u); // IER - Interrupt Enable Register
uartconfig(LCD_WRITE,LCD_ChB,FCR,0b00000110u); // FCR - FIFO Control Register (Disabled ...
    )
lcdwait(1u);
lcdwait(1u);
lcdwait(1u);
lcdwait(1u);
lcdwait(1u);
uartconfig(LCD_WRITE,LCD_ChB,MCR,0b10000000u); // MCR - Modem Control Register ( ...
    Loopback)
uartconfig(LCD_WRITE,LCD_ChB,FCR,0b00000001u); // FCR - FIFO Control Register (Enabled ...
    )
uartconfig(LCD_WRITE,LCD_ChB,EFCR,0b00000000u); // EFCR - Extra Features Control ...
    Register

uartconfig(LCD_WRITE,LCD_ChA,IOState,0xFFu); // IOState - Turn all LEDs OFF

lcdwait(1u);
lcd2Print(LCD_SetRemember,0x01u);
lcd2Print(LCD_ChangeBaudRate,0x33u);
lcdStartupScreen();
lcd2Print(LCD_DisplayOn,0x00u);
lcdPrint(LCD_AutoScrollOn);
lcdPrint(LCD_AutoLineWrapOn);
```

```
    lcdPrint(LCD_AutoTransmitKeyPressesOn);
    lcdPrint(LCD_ClearKeyBuffer);
    lcd2Print(LCD_SetDebounce,0x0Au);
    lcd2Print(LCD_KeyAutoRepeat,0x00u);
    lcdPrint(LCD_KeyAutoRepeatOff);
    lcdSetKeys();
    lcd2Print(LCD_SetSaveBrightness,0xFFu);
    lcd2Print(LCD_SetSaveContrast,LCD_Contrast);

    #if defined(LCD_DEBUG)
        serialString("Loaded");
    #endif
    // 01234567890123456789
}

void uartconfig(unsigned char addr, unsigned char chan, unsigned char reg, unsigned char data) {
    StartI2C1();
    WriteI2C1(addr);
    WriteI2C1(chan | (reg << 3u));
    WriteI2C1(data);
    StopI2C1();
}

void lcdPrint(unsigned char data) {
    while (uartSpace(LCD_ChB) < 2u) {
        Nop();
    }
    StartI2C1();
    WriteI2C1(LCD_WRITE);
    WriteI2C1(LCD_ChB);
    WriteI2C1(0xFEu);
    WriteI2C1(data);
    StopI2C1();
    #if defined(LCD_DEBUG) || defined(LCD_SERIAL_REPEAT)
        while (uartSpace(LCD_ChA) < 2u) {
        Nop();
        }
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA);
        WriteI2C1(0xFEu);
        WriteI2C1(data);
        StopI2C1();
    #endif
}

void serialType(unsigned char data) {
    if (uartSpace(LCD_ChA) >= 1u) {
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA);
```

```
        WriteI2C1(data);
        StopI2C1();
    }
}

void lcdType(unsigned char data) {
    while (uartSpace(LCD_ChB) < 1u) {
        Nop();
    }
    StartI2C1();
    WriteI2C1(LCD_WRITE);
    WriteI2C1(LCD_ChB);
    WriteI2C1(data);
    StopI2C1();
    #if defined(LCD_DEBUG) || defined(LCD_SERIAL_REPEAT)
        while (uartSpace(LCD_ChA) < 1u) {
            Nop();
        }
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA);
        WriteI2C1(data);
        StopI2C1();
    #endif
}

void serialString(const rom char * str) {
    unsigned char i;
    i = strlenpgm(str);
    if (uartSpace(LCD_ChA) >= i) {
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA);
        while (i > 0u) {
            WriteI2C1(*str);
            str++;
            i--;
        }
        StopI2C1();
    }
}

void lcdString(const rom char * str) {
    unsigned char i;
    i = strlenpgm(str);
    while (i > 0u) {
        lcdType(*str);
        str++;
        i--;
    }
}
```

```
void lcdStringRam(char * str) {
    unsigned char i;
    i = strlen(str);
    while (i > 0u) {
        lcdType(*str);
        str++;
        i--;
    }
}

void lcd2Print(unsigned char data1,unsigned char data2) {
    while (uartSpace(LCD_ChB) < 3u) {
        Nop();
    }
    StartI2C1();
    WriteI2C1(LCD_WRITE);
    WriteI2C1(LCD_ChB);
    WriteI2C1(0xFEu);
    WriteI2C1(data1);
    WriteI2C1(data2);
    StopI2C1();
    #if defined(LCD_DEBUG) || defined(LCD_SERIAL_REPEAT)
        while (uartSpace(LCD_ChA) < 3u) {
        Nop();
        }
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA);
        WriteI2C1(0xFEu);
        WriteI2C1(data1);
        WriteI2C1(data2);
        StopI2C1();
    #endif
}

void lcdTypeHex(unsigned char data) {
    unsigned char data1,data2;
    while (uartSpace(LCD_ChB) < 2u) {
        Nop();
    }
    data1 = data >> 4u;
    if (data1 <= 9u) {
        data1 += 0x30u;
    } else {
        data1 += (0x41u - 10u);
    }
    data2 = data & 0x0F;
    if (data2 <= 9u) {
        data2 += 0x30u;
    } else {
```

```
            data2 += (0x41u - 10u);
        }
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChB);
        WriteI2C1(data1);
        WriteI2C1(data2);
        StopI2C1();
        #if defined(LCD_DEBUG) || defined(LCD_SERIAL_REPEAT)
            while (uartSpace(LCD_ChA) < 2u) {
                Nop();
            }
            StartI2C1();
            WriteI2C1(LCD_WRITE);
            WriteI2C1(LCD_ChA);
            WriteI2C1(data1);
            WriteI2C1(data2);
            StopI2C1();
        #endif
    }

    void lcdStartupScreen(void) {
        lcdPrint(0x40u);
        // 12345678901234567890
        lcdString(" MPFD3-T 3D ");
        lcdString("System Starting Up..");
        lcdString("  ");
        lcdString("  ");
    }


    void lcdSetKeys(void) {
        //WatchDog;
        //lcdwait(1u);
        lcdType(0xFEu);
        lcdType(0xD5u);

        lcdType(0x3Fu);
        lcdType(0x3Fu);
        lcdType(0x3Fu);
        lcdType(0x3Fu);
        lcdType(0x3Fu);

        lcdType(0xC8u);
        lcdType(0xC8u);
        lcdType(0xC8u);
        lcdType(0xC8u);
        lcdType(0xC8u);

        lcdType(0xC6u);
        lcdType(0xC6u);
```

```
        lcdType(0xC6u);
        lcdType(0xC6u);
        lcdType(0xC6u);

        lcdType(0xC5u);
        lcdType(0xC5u);
        lcdType(0xC5u);
        lcdType(0xC5u);
        lcdType(0xC5u);

        lcdType(0xC7u);
        lcdType(0xC7u);
        lcdType(0xC7u);
        lcdType(0xC7u);
        lcdType(0xC7u);

        lcdType(0x3Fu);
        lcdType(0x3Fu);
        lcdType(0x3Fu);
        lcdType(0x3Fu);
        lcdType(0x3Fu);

        lcdType(0xC8u);
        lcdType(0xC8u);
        lcdType(0xC8u);
        lcdType(0xC8u);
        lcdType(0xC8u);

        lcdType(0xC6u);
        lcdType(0xC6u);
        lcdType(0xC6u);
        lcdType(0xC6u);
        lcdType(0xC6u);

        lcdType(0xC5u);
        lcdType(0xC5u);
        lcdType(0xC5u);
        lcdType(0xC5u);
        lcdType(0xC5u);

        lcdType(0xC7u);
        lcdType(0xC7u);
        lcdType(0xC7u);
        lcdType(0xC7u);
        lcdType(0xC7u);
}

void lcdEmpty(void) {
    unsigned char uc0;
    do {
        StartI2C1();
```

```
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChB | (LSR << 3u));
        RestartI2C1();
        WriteI2C1(LCD_READ);
        uc0 = ReadI2C1();
        NotAckI2C1();
        StopI2C1();
    } while ((uc0 & 0x60u) != 0x60u);
}

unsigned char uartSpace(unsigned char chan) {
    unsigned char uc;
    StartI2C1();
    WriteI2C1(LCD_WRITE);
    WriteI2C1(chan | (TXLVL << 3u));
    RestartI2C1();
    WriteI2C1(LCD_READ);
    uc = ReadI2C1();
    NotAckI2C1();
    StopI2C1();

    return uc;
}

void lcdCursor(unsigned char col, unsigned char row) {
    while (uartSpace(LCD_ChB) < 4u) {
        Nop();
    }
    StartI2C1();
    WriteI2C1(LCD_WRITE);
    WriteI2C1(LCD_ChB);
    WriteI2C1(0xFEu);
    WriteI2C1(0x47u);
    WriteI2C1(col);
    WriteI2C1(row);
    StopI2C1();
    #if defined(LCD_DEBUG)
        while (uartSpace(LCD_ChA) < 4u) {
            Nop();
        }
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChA);
        WriteI2C1(0xFEu);
        WriteI2C1(0x47u);
        WriteI2C1(col);
        WriteI2C1(row);
        StopI2C1();
    #endif
}
```

```
void lcdTask(void) {
    unsigned char uc0, i;
    IP_ADDR ipaddr;
    MAC_ADDR macaddr;
    WORD w;
    unsigned char Last_LCD_Contrast;
    LCD_SCREENS Last_lcd_screen;

    StartI2C1();
    WriteI2C1(LCD_WRITE);
    WriteI2C1(LCD_ChB | (RXLVL << 3u));
    RestartI2C1();
    WriteI2C1(LCD_READ);
    i = ReadI2C1();
    NotAckI2C1();
    StopI2C1();

    Last_LCD_Contrast = LCD_Contrast;
    while (i > 0u) {
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChB);
        RestartI2C1();
        WriteI2C1(LCD_READ);
        uc0 = ReadI2C1();
        NotAckI2C1();
        StopI2C1();
        Last_lcd_screen = lcd_current_screen;
        if ((uc0 == 0xC5u) || (uc0 == 0xCEu)) { // up
            lcd_current_screen--;
        } else if ((uc0 == 0xC6u) || (uc0 == 0xC9u)) { // down
            lcd_current_screen++;
        }
        if ((Last_lcd_screen != LCD_NONE) && (Last_lcd_screen != LCD_LAST) && (Last_lcd_screen  ...
             != lcd_current_screen)) {
            if (lcd_current_screen == LCD_NONE) {
                lcd_current_screen = LCD_LAST - 1;
            } else if (lcd_current_screen == LCD_LAST) {
                lcd_current_screen = LCD_NONE + 1;
            }
        }
        if ((uc0 == 0xC7u) || (uc0 == 0xCFu)) { // right
            if (LCD_Contrast < 255u) {
                LCD_Contrast = LCD_Contrast + 1u;
            }
        }
        if ((uc0 == 0xC8u) || (uc0 == 0xD0u)) { // left
            if (LCD_Contrast > 0u) {
                LCD_Contrast = LCD_Contrast - 1u;
            }
        }
```

```
    #if defined(LCD_DEBUG) || defined(LCD_SERIAL_REPEAT)
        if (uartSpace(LCD_ChA) > 2u) {
            serialType(uc0);
        }
    #endif
    i--;
}
if (Last_LCD_Contrast != LCD_Contrast) {
    lcd2Print(LCD_SetContrast,LCD_Contrast);
    lcd_current_screen = LCD_NONE;
    lcdPrint(LCD_ClearScreen);
    lcdString("Contrast set to: ");
    lcdType((LCD_Contrast / 100u) + 0x30u);
    lcdType((LCD_Contrast / 10u) % 10u + 0x30u);
    lcdType((LCD_Contrast % 10u) + 0x30u);
    Last_LCD_Contrast = LCD_Contrast;
}

if (lcd_current_screen != lcd_last_screen) {
    force_update = TRUE;
}

if (force_update == TRUE) {
    switch(lcd_current_screen) {
        case LCD_NONE:
            break;
        case LCD_LAST:
            break;
        case LCD_MY_IP:
            lcdPrint(LCD_ClearScreen);
            // 01234567890123456789
            lcdString("IP Address:");
            lcdCursor(6u,2u);
            ipaddr = GetIP();
            lcdType((ipaddr.v[0] / 100u) + 0x30u);
            lcdType((ipaddr.v[0] / 10u) % 10u + 0x30u);
            lcdType((ipaddr.v[0] % 10u) + 0x30u);
            lcdType('.');
            lcdType((ipaddr.v[1] / 100u) + 0x30u);
            lcdType((ipaddr.v[1] / 10u) % 10u + 0x30u);
            lcdType((ipaddr.v[1] % 10u) + 0x30u);
            lcdType('.');
            lcdType((ipaddr.v[2] / 100u) + 0x30u);
            lcdType((ipaddr.v[2] / 10u) % 10u + 0x30u);
            lcdType((ipaddr.v[2] % 10u) + 0x30u);
            lcdType('.');
            lcdType((ipaddr.v[3] / 100u) + 0x30u);
            lcdType((ipaddr.v[3] / 10u) % 10u + 0x30u);
            lcdType((ipaddr.v[3] % 10u) + 0x30u);
            lcdCursor(1u,3u);
            lcdString("MAC Address:");
```

```
        lcdCursor(4u,4u);
        macaddr = GetMAC();
        lcdTypeHex(macaddr.v[0]);
        lcdType(':');
        lcdTypeHex(macaddr.v[1]);
        lcdType(':');
        lcdTypeHex(macaddr.v[2]);
        lcdType(':');
        lcdTypeHex(macaddr.v[3]);
        lcdType(':');
        lcdTypeHex(macaddr.v[4]);
        lcdType(':');
        lcdTypeHex(macaddr.v[5]);
        break;
    case LCD_CONNECTION:
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("Connection Status:");
        if (MPFD3DConnected() == 1u) {
            lcdCursor(2u,2u);
            lcdString("Connected");
            lcdCursor(3u,3u);
            MPFD3DTypeIPAddress();
            lcdCursor(3u,4u);
            MPFD3DTypeMACAddress();
        } else {
            lcdCursor(2u,2u);
            lcdString("Not Connected");
        }
        break;
    case LCD_BOARDS:
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("Detector Strings:");
        lcdCursor(2u,2u);
        lcdType('[');
        w = GetConnectedBoards();
        for (i = 0x00u; i < 0x10u; ++i) {
            if (i != 0x0Au) {
                if (((w >> ((WORD)i)) & 0x0001u) == 0x0001u) {
                    lcdType('-');
                } else {
                    lcdType(0x41u + i);
                }
            }
        }
        lcdType(']');
        lcdCursor(1u,3u);
        lcdString("Daughter Boards:");
        lcdCursor(2u,4u);
        lcdType('[');
```

```
        for (i = 0x00u; i < 0x0Fu; ++i) {
            uc0 = GetBoardIDs(i);
            if (uc0 == 0x0Fu || uc0 == 0x00u) {
                lcdType('-');
            } else {
                if (uc0 >= 0x0Bu) {
                    lcdType(0x56u + uc0);
                } else {
                    lcdType(0x2Fu + uc0);
                }
            }
        }
        lcdType(']');
        //ABCDEFGHIJLMNOP]");
        break;
    case LCD_HIGH_VOLTAGE:
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("High Voltage:");
        lcdCursor(3u,2u);
        lcdString("Hex: ");
        MPFD3DTypeHVbits();
        lcdCursor(3u,3u);
        lcdString("Val: ");
        MPFD3DTypeHVvalue();
        lcdString(" V");
        break;
    case LCD_LLD:
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("LLD - Lower Level");
        lcdCursor(3u,2u);
        lcdString("Discriminator:");
        lcdCursor(3u,3u);
        lcdString("Hex: ");
        MPFD3DTypeLLDbits();
        lcdCursor(3u,4u);
        lcdString("Val: ");
        MPFD3DTypeLLDvalue();
        lcdString(" mV");
        break;
    case LCD_SPEED:
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("Sample Speed:");
        lcdCursor(1u,2u);
        MPFD3DTypeSpeed();
        break;
    case LCD_TEMPERATURE:
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
```

```
                lcdString("Temperature:");
                lcdCursor(7u,2u);
                TempTypeC();
                lcdString(" deg C");
                lcdCursor(7u,3u);
                TempTypeF();
                lcdString(" deg F");
                break;
            case LCD_VIEWCURRENTDATALINES:
                lcdPrint(LCD_ClearScreen);
                // 01234567890123456789
                lcdString("Current Data Lines: ");
                showdatalines();
                break;
            default:
                break;
        }
        force_update = FALSE;
    }
    lcd_last_screen = lcd_current_screen;
}

void ChangeLCDScreen(LCD_SCREENS tmp) {
    if (lcd_current_screen != tmp) {
        if (tmp == LCD_NONE) {
            lcdPrint(LCD_ClearScreen);
        }
        lcd_current_screen = tmp;
    }
}

void LCDForceUpdate(void) {
    force_update = TRUE;
}

LCD_SCREENS GetLCDScreen(void) {
    return lcd_current_screen;
}

void lcdDBsync(void) {
    unsigned char uc0 = 0x00u;
    unsigned char uc1 = 0x00u;
    lcdDBsync1();
    while (!Shutdown_Mode_Selected()) {
        StartI2C1();
        WriteI2C1(LCD_WRITE);
        WriteI2C1(LCD_ChB);
        RestartI2C1();
        WriteI2C1(LCD_READ);
        uc0 = ReadI2C1();
        NotAckI2C1();
```

```
            StopI2C1();
            if (uc0 == 0xC6u) {
                uc1 = (uc1 + 1u) % 3u;
                if (uc1 == 0x00u) {
                    lcdDBsync1();
                } else if (uc1 == 0x01u) {
                    lcdDBsync2();
                } else {
                    lcdDBsync3();
                }
            } else if (uc0 == 0xC5u) {
                uc1 = (uc1 + 2u) % 3u;
                if (uc1 == 0x00u) {
                    lcdDBsync1();
                } else if (uc1 == 0x01u) {
                    lcdDBsync2();
                } else {
                    lcdDBsync3();
                }
            }
        }
        StandbyLED_ON;
        PowerLED_OFF;
        POWER_OFF_OUT = 1;
}

void lcdDBsync1(void) {
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("Daughter Boards are ");
        lcdString(" not synchronized");
        lcdString("Press ");
        lcdType(0xC6u);
        lcdString(" for more ");
        lcdString(" instructions...");
}

void lcdDBsync2(void) {
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
        lcdString("Send data from next ");
        lcdString("screen (Press ");
        lcdType(0xC6u);
        lcdString(") to:");
        lcdString(" martin@ohmes.com ");
        lcdString("then push Standby");
}

void lcdDBsync3(void) {
        lcdPrint(LCD_ClearScreen);
        // 01234567890123456789
```

```
    lcdString("Current Data Lines: ");
    showdatalines();
}

void showdatalines(void) {
    lcdType('A');
    lcdType(0x30u + (unsigned char)TRA1_IN);
    lcdType(0x30u + (unsigned char)TRA2_IN);
    lcdType(0x30u + (unsigned char)TRA3_IN);
    lcdType('B');
    lcdType(0x30u + (unsigned char)TRB1_IN);
    lcdType(0x30u + (unsigned char)TRB2_IN);
    lcdType(0x30u + (unsigned char)TRB3_IN);
    lcdType('C');
    lcdType(0x30u + (unsigned char)TRC1_IN);
    lcdType(0x30u + (unsigned char)TRC2_IN);
    lcdType(0x30u + (unsigned char)TRC3_IN);
    lcdType('D');
    lcdType(0x30u + (unsigned char)TRD1_IN);
    lcdType(0x30u + (unsigned char)TRD2_IN);
    lcdType(0x30u + (unsigned char)TRD3_IN);
    lcdType('E');
    lcdType(0x30u + (unsigned char)TRE1_IN);
    lcdType(0x30u + (unsigned char)TRE2_IN);
    lcdType(0x30u + (unsigned char)TRE3_IN);
    lcdType('F');
    lcdType(0x30u + (unsigned char)TRF1_IN);
    lcdType(0x30u + (unsigned char)TRF2_IN);
    lcdType(0x30u + (unsigned char)TRF3_IN);
    lcdType('G');
    lcdType(0x30u + (unsigned char)TRG1_IN);
    lcdType(0x30u + (unsigned char)TRG2_IN);
    lcdType(0x30u + (unsigned char)TRG3_IN);
    lcdType('H');
    lcdType(0x30u + (unsigned char)TRH1_IN);
    lcdType(0x30u + (unsigned char)TRH2_IN);
    lcdType(0x30u + (unsigned char)TRH3_IN);
    lcdType('I');
    lcdType(0x30u + (unsigned char)TRI1_IN);
    lcdType(0x30u + (unsigned char)TRI2_IN);
    lcdType(0x30u + (unsigned char)TRI3_IN);
    lcdType('J');
    lcdType(0x30u + (unsigned char)TRJ1_IN);
    lcdType(0x30u + (unsigned char)TRJ2_IN);
    lcdType(0x30u + (unsigned char)TRJ3_IN);
    lcdType('L');
    lcdType(0x30u + (unsigned char)TRL1_IN);
    lcdType(0x30u + (unsigned char)TRL2_IN);
    lcdType(0x30u + (unsigned char)TRL3_IN);
    lcdType('M');
    lcdType(0x30u + (unsigned char)TRM1_IN);
```

```
    lcdType(0x30u + (unsigned char)TRM2_IN);
    lcdType(0x30u + (unsigned char)TRM3_IN);
    lcdType('N');
    lcdType(0x30u + (unsigned char)TRN1_IN);
    lcdType(0x30u + (unsigned char)TRN2_IN);
    lcdType(0x30u + (unsigned char)TRN3_IN);
    lcdType('O');
    lcdType(0x30u + (unsigned char)TRO1_IN);
    lcdType(0x30u + (unsigned char)TRO2_IN);
    lcdType(0x30u + (unsigned char)TRO3_IN);
    lcdType('P');
    lcdType(0x30u + (unsigned char)TRP1_IN);
    lcdType(0x30u + (unsigned char)TRP2_IN);
    lcdType(0x30u + (unsigned char)TRP3_IN);
}

void serial_debug(unsigned char screen, unsigned char step) {
    #if defined(LCD_DEBUG)
    unsigned char i;
    for (i = 0u; i < 100u; ++i) {
        Nop();
    }
    serialType(0x0Du);
    serialType(0x0Au);
    serialType((screen / 100u) + 0x30u);
    serialType((screen / 10u) % 10u + 0x30u);
    serialType((screen % 10u) + 0x30u);
    serialType(0x2Du);
    serialType((step / 100u) + 0x30u);
    serialType((step / 10u) % 10u + 0x30u);
    serialType((step % 10u) + 0x30u);
    #endif
}
```

End lcd.c

B.7.9 mpfd3d.h

This is the header file for the program mpfd3d.c (Section §B.7.10) of the mother board's microcontroller.

Begin mpfd3d.h

```
#ifndef __mpfd3d_H
#define __mpfd3d_H

#include "GenericTypeDefs.h"

void MPFD3DInit(void);
```

```
void MPFD3DTask(void);
unsigned char MPFD3DConnected(void);
void MPFD3DTypeIPAddress(void);
void MPFD3DTypeMACAddress(void);
void MPFD3DTypeHVbits(void);
void MPFD3DTypeHVvalue(void);
void MPFD3DTypeLLDbits(void);
void MPFD3DTypeLLDvalue(void);
void MPFD3DTypeSpeed(void);
void OFFButtonPressed(void);
BOOL Shutdown_Mode_Selected(void);
WORD GetConnectedBoards(void);
unsigned char GetBoardIDs(unsigned char i);

#endif // __mpfd3d_H
```

End mpfd3d.h

B.7.10 mpfd3d.c

This is the mpfd3d program for the mother board's microcontroller.

Begin mpfd3d.c

```
#include "mpfd3d.h"
#include "TCPIPConfig.h"
#include "TCPIP.h"
#include "UDP.h"
#include "lcd.h"
#include "SNTP.h"
#include "temp.h"
#include "GenericTypeDefs.h"
#include "XEEPROM.h"
#include "stdio.h"
#include "config.h"

#define MPFD3D_RECEIVE_PORT (22223u) // Send data to MPFD3D on this port
#define MPFD3D_TRANSMIT_PORT (22224u) // Get data from MPFD3D on this port
#define MAX_MPFD3D_CMD_STRING_LEN (128u)
#define MAX_MPFD3D_ARGS (7u)
#define MPFD3D_TIMEOUT (180ull * TICK_SECOND) // 3 minutes
#define MPFD3D_MAX_TIMEOUT (600ull * TICK_SECOND) // 10 minutes
#define MPFD3D_SHUTDOWN_TIMEOUT (180ull) // 3 minutes (180 seconds)
static QWORD timeout_tick;

static UDP_SOCKET MPFD3DSocket;
static UDP_SOCKET MPFD3DRespondSocket;
static UDP_SOCKET MPFD3DExtraSocket;
static IP_ADDR MPFD3DIPAddress;
static IP_ADDR MPFD3DConnIPAddress;
```

```
static MAC_ADDR MPFD3DConnMACAddress;

// Commands coming from Remote MPFD3D Program
typedef enum _MPFD3D_COMMAND
{
    MPFD3D_CMD_CONNECT,
    MPFD3D_CMD_REFRESH,
    MPFD3D_CMD_TAKEOVER,
    MPFD3D_CMD_SETTINGS,
    MPFD3D_CMD_DATA_START,
    MPFD3D_CMD_DATA_STOP,
    MPFD3D_CMD_QUIT,
    MPFD3D_CMD_KILL,
    MPFD3D_CMD_UNKNOWN,
    MPFD3D_CMD_NONE
} MPFD3D_COMMAND;

// Each entry in following table must match with that of FTP_COMMAND enum.
static ROM char * ROM MPFD3DCommandString[] =
{
    "CONN", // MPFD3D_CMD_CONNECT
    "HERE", // MPFD3D_CMD_REFRESH
    "TAKE", // MPFD3D_CMD_TAKEOVER
    "SETT", // MPFD3D_CMD_SETTINGS
    "STRT", // MPFD3D_CMD_DATA_START
    "STOP", // MPFD3D_CMD_DATA_STOP
    "QUIT", // MPFD3D_CMD_QUIT
    "KILL" // MPFD3D_CMD_KILL
};
#define MPFD3D_COMMAND_TABLE_SIZE ( sizeof(MPFD3DCommandString)/sizeof(  ...
    MPFD3DCommandString[0]) )

// Response commands generated by MPFD3D System and sent to Remote MPFD3D Program
typedef enum _MPFD3D_RESPONSE
{
    MPFD3D_RESP_CONNECT_OK, // Connection granted
    MPFD3D_RESP_DENY, // Deny request
    MPFD3D_RESP_WISH, // Ask if wish to takeover connection
    MPFD3D_RESP_KILL, // Terminate connection
    MPFD3D_RESP_CHANGING, // Changing settings
    MPFD3D_RESP_DONE, // Setting change completed
    MPFD3D_RESP_TIME, // Timout has occured since last Remote Activity
    MPFD3D_RESP_DATA, // Send Data set
    MPFD3D_RESP_NONE // This must always be the last
                     // There is no corresponding string.
} MPFD3D_RESPONSE;

// Each entry in following table must match with FTP_RESPONE enum
static ROM char * ROM MPFD3DResponseString[] =
{
    "CONN", // MPFD3D_RESP_CONNECT_OK
```

```
    "DENY", // MPFD3D_RESP_DENY
    "WISH", // MPFD3D_RESP_WISH
    "KILL", // MPFD3D_RESP_KILL
    "CHNG", // MPFD3D_RESP_CHANGING
    "DONE", // MPFD3D_RESP_DONE
    "TIME", // MPFD3D_RESP_TIME
    "DATA" // MPFD3D_RESP_DATA
};

// State Machine
typedef enum _SM_MPFD3D
{
    SM_MPFD3D_NOT_CONNECTED,
    SM_MPFD3D_CONNECTED,
    SM_MPFD3D_RESPOND,
    SM_MPFD3D_WISH,
    SM_MPFD3D_TAKEOVER,
    SM_MPFD3D_TIMEOUT,
    SM_MPFD3D_KILL
} SM_MPFD3D;

static MPFD3D_COMMAND MPFD3DCommand;
static MPFD3D_RESPONSE MPFD3DResponse;
static SM_MPFD3D smMPFD3D; // MPFD3D server FSM state
#pragma udata DATA_RAM
static far BYTE MPFD_Data_Array[1327u];
#pragma udata
static BOOL MPFD_Data_Ready = FALSE;
static BOOL MPFD_REFRESH_SETTINGS = FALSE;

static unsigned char MPFD3Dstatus;
static TICK lastMPFD3DActivity; // Timeout keeper.
static BYTE MPFD3DString[MAX_MPFD3D_CMD_STRING_LEN+2u];
static BYTE MPFD3DStringLen;
static BYTE *MPFD3D_argv[MAX_MPFD3D_ARGS]; // Parameters for a udp command
static BYTE MPFD3D_argc; // Total number of params for a udp command

extern NODE_INFO remoteNode;
static NODE_INFO connectedNode;
static NODE_INFO takeoverNode;
static NODE_INFO denyNode;

MPFD3D_COMMAND ParseMPFD3DCommand(BYTE *cmd);
BYTE ParseMPFD3DString(void);
void ExecuteMPFD3DCommand(MPFD3D_COMMAND cmd);
void ExecuteMPFD3DResponse(void);

static QWORD NextTimeStamp = 0u;
static QWORD ThisTimeStamp = 0u;
static QWORD TimeDiff = 0u;
static QWORD_VAL TriggeredTimeStamp;
```

```
static QWORD_VAL LastTriggeredTimeStamp;
static QWORD ShutdownTimeStamp = 0u;
static far unsigned int HV_Setpoint = 0x0000u;
static far unsigned int HV_NewSetpoint = 0x0000u;
static far unsigned int LLD_Setpoint = 0x0000u;
static far unsigned int LLD_NewSetpoint = 0x0000u;
static far unsigned char Speed = 0x00u;
static far unsigned char Speed_New = 0x00u;
static far unsigned char MSV_Res = 0x00u;
static far unsigned char MSV_Res_New = 0x00u;
static far unsigned char Curr_Res = 0x00u;
static far unsigned char Curr_Res_New = 0x00u;
static far unsigned char Temp_Res = 0x00u;
static far unsigned char Temp_Res_New = 0x00u;
static far unsigned char ResetHVFuses = 0x00u;
static WORD_VAL ConnectedBoards = 0x0000u;
static unsigned int BoardMask = 0x0000u;
static far unsigned char BoardID[15];
static unsigned char HVuc1 = 0u;
static LCD_SCREENS LastScreen;
static unsigned char MonitorUPS = 0u;

#define SPS15 0b00000011u
#define SPS30 0b00000010u
#define SPS60 0b00000001u
#define SPS240 0b00000000u

//QWORD TicksPerSample[] = {262144ull,131072ull,65536ull,32768ull,16384ull, 8192ull, 4370ull,   ...
      2185ull, 1093ull, 547ull, 274ull};
//unsigned char SpeedEquivalent[] = { SPS15, SPS15, SPS15, SPS15, SPS15, SPS15, SPS15, SPS30,   ...
      SPS60, SPS240, SPS240};
//static ROM char * ROM SPSString[] =
//{// 01234567890123456789
// " 4 Seconds/Sample", // 262144
// " 2 Seconds/Sample", // 131072
// " 1 Sample/Second", // 65536
// " 2 Samples/Second", // 32768
// " 4 Samples/Second", // 16384
// " 8 Samples/Second", // 8192
// " 15 Samples/Second", // 4370
// " 30 Samples/Second", // 2185
// " 60 Samples/Second", // 1093
// " 120 Samples/Second", // 547
// " 240 Samples/Second" // 274
//};
far QWORD TicksPerSample[] = {262144ull,131072ull,65536ull,32768ull,21846ull,16384ull,13108ull  ...
      ,10923ull, 9363ull, 8192ull, 7282ull, 6554ull, 5958ull, 5462ull, 5042ull, 4370ull};
far unsigned char SpeedEquivalent[] = { SPS15, SPS15, SPS15, SPS15, SPS15, SPS15, SPS15, SPS15,   ...
      SPS15, SPS15, SPS15, SPS15, SPS15, SPS15, SPS15, SPS15};
far static ROM char * ROM SPSString[] =
{// 01234567890123456789
```

```
    " 4 Seconds/Sample",// 262144
    " 2 Seconds/Sample",// 131072
    " 1 Sample/Second",// 65536
    " 2 Samples/Second",// 32768
    " 3 Samples/Second",// 21846
    " 4 Samples/Second",// 16384
    " 5 Samples/Second",// 13108
    " 6 Samples/Second",// 10923
    " 7 Samples/Second",// 9363
    " 8 Samples/Second",// 8192
    " 9 Samples/Second",// 7282
    " 10 Samples/Second",// 6554
    " 11 Samples/Second",// 5958
    " 12 Samples/Second",// 5462
    " 13 Samples/Second",// 5042
    " 15 Samples/Second" // 4370
};

static far unsigned char SpeedIndex = 0u;
static far unsigned char SpeedIndexNew = 0u;
static far unsigned char SpeedIndexLast = 0u;

typedef enum _MPFD3D_DATA_TASK
{
    MPFD3D_INITIATED,
    MPFD3D_DAUGHTERS_INITIALIZED,
    MPFD3D_DAUGHTERS_TRIGGER,
    MPFD3D_DAUGHTERS_WAIT,
    MPFD3D_SETTINGS,
    MPFD3D_COLLECT_DATA,
    MPFD3D_SHUTDOWN
} MPFD3D_DATA_TASK;

static MPFD3D_DATA_TASK DataTask = MPFD3D_INITIATED;
static BOOL SendData = FALSE;
static unsigned char NumBoards = 0;
static volatile BOOL SHUTDOWN_MODE_SELECTED = FALSE;
static DWORD PowerFailureTime = 0;
static unsigned char StoredSettings[5];
static unsigned char DataSubTask = 0u;
static unsigned char SettingsTask = 0x00u;

void mpfd3d_daughters_ready(void);
extern void MPFD_Write_Done(void);
extern void MPFD_Read_Done(void);
void mpfd3d_collect_data(void);
extern BOOL SyncTimer(void);
extern void ResetSyncTimerShort(void);
extern BOOL GetSecondTimer3(void);
extern void ResetSecondTimer3(void);
```

```
void MPFD3DInit(void) {
    unsigned int i;

    TW_OUT = 0;
    TW_TRIS = 0;
    connectedNode.IPAddr.Val = 0x00000000;
    MPFD3DRespondSocket = INVALID_UDP_SOCKET;
    MPFD3DExtraSocket = INVALID_UDP_SOCKET;
    MPFD3DSocket = UDPOpen(MPFD3D_TRANSMIT_PORT, NULL,   ...
        MPFD3D_TRANSMIT_PORT);
    smMPFD3D = SM_MPFD3D_NOT_CONNECTED;
    MPFD3DStringLen = 0;
    DataTask = MPFD3D_INITIATED;
    PowerFailureTime = TickGetDiv64K();
    SendData = FALSE;
    for (i = 0u; i < 1342u; ++i) {
        MPFD_Data_Array[i] = 0u;
    }
    LastTriggeredTimeStamp = GetUTCTimeStampVal();
    TriggeredTimeStamp = GetUTCTimeStampVal();
    ShutdownTimeStamp = 0u;
    for (i = 0u; i < 15u; ++i) {
        BoardID[i] = 0x0Fu;
    }
    return;
}

void MPFD3DTask(void) {
    BYTE v;
    unsigned char i,uc0,uc1;
    unsigned char rw = 0u;
    WORD_VAL ui1, ui2, ui3;

    /***** Determine if running on batteries *****/
    if ((((unsigned char)UPS_AC_GOOD_IN) == 0x01u) || (((unsigned char)MonitorUPS) == 0x00u))
        PowerFailureTime = TickGetDiv64K();
    else if ((TickGetDiv64K() - PowerFailureTime) > (15ul*60ul)) // if battery powered for more than   ...
        5 minutes then trigger shutdown
        SHUTDOWN_MODE_SELECTED = TRUE;

    /***** Read UDP packets and determine if task necessary *****/
    if (UDPIsGetReady(MPFD3DSocket) > 0u) {
        timeout_tick = MPFD3D_TIMEOUT;
        while( UDPGet(&v ) ) {
            MPFD3DString[MPFD3DStringLen++] = v;
            if ( MPFD3DStringLen == MAX_MPFD3D_CMD_STRING_LEN )
                MPFD3DStringLen = 0u;
        }
        if ( v == '\n' ) {
            MPFD3DString[MPFD3DStringLen] = '\0';
            MPFD3DStringLen = 0u;
```

```
            if (ParseMPFD3DString() < 1u)
                return;
            MPFD3DCommand = ParseMPFD3DCommand(MPFD3D_argv[0]);
        }
        if (((MPFD3DCommand < MPFD3D_CMD_UNKNOWN) && (smMPFD3D !=  ...
             SM_MPFD3D_NOT_CONNECTED)) || (MPFD3DCommand <=  ...
             MPFD3D_CMD_REFRESH))
            ExecuteMPFD3DCommand(MPFD3DCommand);
    }
    if ((smMPFD3D == SM_MPFD3D_CONNECTED) && ((TickGet() - lastMPFD3DActivity) >  ...
         timeout_tick)) {
        timeout_tick = timeout_tick + (30ull * TICK_SECOND);
        if (timeout_tick >= MPFD3D_MAX_TIMEOUT)
            smMPFD3D = SM_MPFD3D_KILL;
        else
            smMPFD3D = SM_MPFD3D_TIMEOUT;
    }

    /***** Shutdown initiated - close remote connection *****/
    if ((smMPFD3D == SM_MPFD3D_CONNECTED) && SHUTDOWN_MODE_SELECTED)
        smMPFD3D = SM_MPFD3D_KILL;

    /***** Send UDP Response to remote connection *****/
    if (smMPFD3D > SM_MPFD3D_CONNECTED)
        ExecuteMPFD3DResponse();

    if (smMPFD3D == SM_MPFD3D_NOT_CONNECTED)
        SendData = FALSE;

    /***** Perform daughter board data operations *****/
    ThisTimeStamp = GetUTCTimeStamp();

    /***** Check if early shutdown is necessary *****/
    if (SHUTDOWN_MODE_SELECTED) {
        if (SNTP_has_run()) {
            if ((DataTask < MPFD3D_DAUGHTERS_INITIALIZED) && (smMPFD3D ==  ...
                 SM_MPFD3D_NOT_CONNECTED)) {
                DataTask = MPFD3D_SHUTDOWN;
                NextTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u]*0x0Au); // 10  ...
                    Second delay
                ShutdownTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u]*0x0Au); // 10  ...
                    seconds
                ChangeLCDScreen(LCD_NONE);
                lcdPrint(LCD_ClearScreen);
                // 12345678901234567890
                lcdString(" SHUTDOWN INITIATED ");
                lcdString(" Safely Turning Off ");
                lcdString(" High Voltage ");
                lcdString(" Please Wait - 0:10 ");
            }
        } else {
```

```
            StandbyLED_ON;
            PowerLED_OFF;
            POWER_OFF_OUT = 1;
        }
    }

    if (ThisTimeStamp >= NextTimeStamp) {
        switch(DataTask) {
            /***** Give the system some time to initialize everything *****/
            case MPFD3D_INITIATED:
                if (SNTP_has_run()) {
                    DataTask = MPFD3D_DAUGHTERS_INITIALIZED;
                    NextTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u]*20ull); // 20  ...
                        Second delay
                    #if defined(LCD_DEBUG)
                        serial_debug(99u,0u);
                    #endif
                    READ_TRIS = 0;
                    WRITE_TRIS = 0;
                    TRIGGER_TRIS = 0;
                    READ_OUT = 0;
                    WRITE_OUT = 0;
                    TRIGGER_OUT = 0;
                    TW_TRIS = 0;
                }
                break;

            /***** Trigger and wait for daughters to respond *****/
            case MPFD3D_DAUGHTERS_INITIALIZED:
                #if defined(LCD_DEBUG)
                    serial_debug(99u,1u);
                #endif
                mpfd3d_daughters_ready();
                DataSubTask = 0;
                DataTask = MPFD3D_DAUGHTERS_TRIGGER;

            /***** Set Trigger and wait for daughter boards to show they are ready *****/
            case MPFD3D_DAUGHTERS_TRIGGER:
                #if defined(LCD_DEBUG)
                    serial_debug(99u,2u);
                #endif
                READ_TRIS = 0;
                WRITE_TRIS = 0;
                TRIGGER_TRIS = 1;
                ResetSecondTimer3();
                TW_TRIS = 1;
                if ((REACTOR_ON_IN == 1u) && (REACTOR_PULSE_IN == 1u)) {
                    while (!TRIGGER_IN_BOOL);
                } else {
                    if (!TRIGGER_IN_BOOL) {
                        DataTask = MPFD3D_DAUGHTERS_WAIT;
```

```
                break;
            }
        }

    /***** Wait for daughter boards to show they are ready *****/
    case MPFD3D_DAUGHTERS_WAIT:
        if (TRIGGER_IN_BOOL) {
            DataTask = MPFD3D_SETTINGS;
        } else if (!GetSecondTimer3()) {
            T3CONbits.TMR3ON = 0; // Stops Timer3
            TRIGGER_TRIS = 0;
            TW_TRIS = 0;
            DataTask = MPFD3D_DAUGHTERS_TRIGGER;
            if (SHUTDOWN_MODE_SELECTED) {
                if (ShutdownTimeStamp == 0u) {
                    if (XEEReadArray(0x00000000,(unsigned char *)&StoredSettings,5u)   ...
                        == XEE_SUCCESS)
                        ShutdownTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u  ...
                            ]*(0x0Au + ((((unsigned int)StoredSettings[0u]) & 0x00FFu)   ...
                            + (((unsigned int)StoredSettings[1u]) << 8u & 0xFF00u) / 4u)  ...
                            )); // 30 seconds + HVbits/11 (5.477 V/s ramprate * 2.048 bits/ ...
                            V)
                    else
                        ShutdownTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u  ...
                            ]*0xD6u); // 214 seconds (500 V shutdown)
                    ChangeLCDScreen(LCD_NONE);
                    lcdPrint(LCD_ClearScreen);
                    // 01234567890123456789
                    lcdString(" SHUTDOWN INITIATED ");
                    lcdString(" Safely Turning Off ");
                    lcdString(" High Voltage ");
                    lcdString(" Please Wait - ");
                } else if (ShutdownTimeStamp <= GetUTCTimeStamp())
                    DataTask = MPFD3D_SHUTDOWN;
            }
            break;
        } else
            break;

    /***** Determine if updated settings must be sent to the daughter boards *****/
    case MPFD3D_SETTINGS:
        DataSubTask = 0u;
        #if defined(LCD_DEBUG)
            serial_debug(99u,3u);
        #endif
        uc0 = 0u;
        SpeedIndexLast = SpeedIndex;

        if (SpeedEquivalent[SpeedIndex] != SpeedEquivalent[SpeedIndexNew]) {
            //
        } else if (SpeedIndex != SpeedIndexNew) {
```

```
        SpeedIndex = SpeedIndexNew;
        if (GetLCDScreen() == LCD_SPEED)
            LCDForceUpdate();
    }
    if (MPFD_REFRESH_SETTINGS) {
        if (XEEReadArray(0x00000000,(unsigned char *)&StoredSettings,5u) ==   ...
             XEE_SUCCESS) {
            HV_NewSetpoint = (((unsigned int)StoredSettings[0u]) & 0x00FFu) + (((  ...
                unsigned int)StoredSettings[1u]) << 8u & 0xFF00u);
            LLD_NewSetpoint = (((unsigned int)StoredSettings[2u]) & 0x00FFu) + (((  ...
                unsigned int)StoredSettings[3u]) << 8u & 0xFF00u);
            MSV_Res_New = (StoredSettings[4u] >> 4u) & 0b00000011u;
            Curr_Res_New = (StoredSettings[4u] >> 2u) & 0b00000011u;
            Temp_Res_New = StoredSettings[4u] & 0b00000011u;
            MonitorUPS = (StoredSettings[4u] >> 6u) & 0b00000001u;
            MPFD_REFRESH_SETTINGS = FALSE;
        }
    }
    if (LLD_Setpoint != LLD_NewSetpoint) {
        LLD_Setpoint = LLD_NewSetpoint;
        if (GetLCDScreen() == LCD_LLD)
            LCDForceUpdate();
    }
    if (HV_Setpoint != HV_NewSetpoint) {
        HV_Setpoint = HV_NewSetpoint;
        if (GetLCDScreen() == LCD_HIGH_VOLTAGE)
            LCDForceUpdate();
    }
    if (GetLCDScreen() == LCD_BOARDS) {
        for (i = 0x00u; i < 0x0Fu; ++i) {
            if (i >= 0x0Au) {
                if (((ConnectedBoards.Val >> (i + 0x01u)) & 0x0001u) == 0x0000u) {
                    if ((BoardID[i] & 0x0Fu) != ((BoardID[i] >> 4) & 0x0Fu)) {
                        BoardID[i] = ((BoardID[i] << 4) & 0xF0u) | (BoardID[i] & 0  ...
                            x0Fu);
                        LCDForceUpdate();
                    }
                }
            } else {
                if (((ConnectedBoards.Val >> i) & 0x0001u) == 0x0000u) {
                    if ((BoardID[i] & 0x0Fu) != ((BoardID[i] >> 4) & 0x0Fu)) {
                        BoardID[i] = ((BoardID[i] << 4) & 0xF0u) | (BoardID[i] & 0  ...
                            x0Fu);
                        LCDForceUpdate();
                    }
                }
            }
        }
    }
    if (Temp_Res != Temp_Res_New)
        Temp_Res = Temp_Res_New;
```

```
        if (Curr_Res != Curr_Res_New)
            Curr_Res = Curr_Res_New;
        if (MSV_Res != MSV_Res_New)
            MSV_Res = MSV_Res_New;
        DataTask = MPFD3D_COLLECT_DATA;

    /***** Daughter boards are now ready to collect data ****/
    case MPFD3D_COLLECT_DATA:
        #if defined(LCD_DEBUG)
            serial_debug(99u,4u);
        #endif
        if (SHUTDOWN_MODE_SELECTED)
            HV_NewSetpoint = 0x0000u;
        ResetSecondTimer3();
        mpfd3d_collect_data();
        T3CONbits.TMR3ON = 0; // Stops Timer3
        #if defined(LCD_DEBUG)
            serial_debug(99u,5u);
        #endif
        if (SHUTDOWN_MODE_SELECTED) {
            if (ShutdownTimeStamp == 0u) {
                if (XEEReadArray(0x00000000,(unsigned char *)&StoredSettings,5u) ==  ...
                    XEE_SUCCESS)
                    ShutdownTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u]*(0  ...
                        x0Au + ((((unsigned int)StoredSettings[0u]) & 0x00FFu) + ((  ...
                        unsigned int)StoredSettings[1u]) << 8u & 0xFF00u) / 4u))); // 30  ...
                        seconds + HVbits/11 (5.477 V/s ramprate * 2.048 bits/V)
                else
                    ShutdownTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u]*0  ...
                        xD6u); // 214 seconds (500 V shutdown)
                //ShutdownTimeStamp = GetUTCTimeStamp() + (TicksPerSample[2u]* ...
                    MPFD3D_SHUTDOWN_TIMEOUT); // 120 Second delay
                ChangeLCDScreen(LCD_NONE);
                lcdPrint(LCD_ClearScreen);
                // 01234567890123456789
                lcdString(" SHUTDOWN INITIATED ");
                lcdString(" Safely Turning Off ");
                lcdString(" High Voltage ");
                lcdString(" Please Wait - ");
            } else if (ShutdownTimeStamp <= GetUTCTimeStamp())
                DataTask = MPFD3D_SHUTDOWN;
        }
        break;

    case MPFD3D_SHUTDOWN:
        StandbyLED_ON;
        PowerLED_OFF;
        POWER_OFF_OUT = 1;
        break;
    default:
        break;
```

```
        }
    }
    if (SHUTDOWN_MODE_SELECTED) {
        TimeDiff = GetUTCTimeStamp();
        if (TimeDiff < ShutdownTimeStamp) {
            ChangeLCDScreen(LCD_NONE);
            // 12345678901234567890
            lcdCursor(1u,1u);
            lcdString(" SHUTDOWN INITIATED ");
            lcdString(" Safely Turning Off ");
            lcdString(" High Voltage ");
            lcdString(" Please Wait - ");
            uc0 = ((unsigned char)(((ShutdownTimeStamp - TimeDiff) >> 16u)));
            //lcdCursor(16u,4u);
            lcdType((uc0 / 60u) + 0x30u);
            lcdType(':');
            lcdType(((uc0 % 60u) / 10u) + 0x30u);
            lcdType(((uc0 % 60u) % 10u) + 0x30u);
            lcdType(' ');
        }
    }
    if (MPFD_Data_Ready == TRUE) {
        if (SendData == TRUE) {
            if (((MPFD_Data_Array[1] == 0u) && (MPFD_Data_Array[2] == 0u) && ( ...
                MPFD_Data_Array[3] == 0u) && (MPFD_Data_Array[4] == 0u) && ( ...
                MPFD_Data_Array[5] == 0u)) || ((MPFD_Data_Array[7] == 0u) && ( ...
                MPFD_Data_Array[8] == 0u) && (MPFD_Data_Array[9] == 0u) && ( ...
                MPFD_Data_Array[10] == 0u) && (MPFD_Data_Array[11] == 0u)))
                MPFD_Data_Ready = FALSE;
            else {
                MPFD3DRespondSocket = UDPOpen(MPFD3D_RECEIVE_PORT,&connectedNode, ...
                    MPFD3D_RECEIVE_PORT);
                if (UDPIsPutReady(MPFD3DRespondSocket) >= 1327u+4u) {
                    UDPPutROMArray((ROM BYTE*)MPFD3DResponseString[ ...
                        MPFD3D_RESP_DATA],4u);
                    UDPPutArray((BYTE*)MPFD_Data_Array,1327u);
                    UDPFlush();
                    MPFD_Data_Ready = FALSE;
                }
                UDPClose(MPFD3DRespondSocket);
            }
        } else
            MPFD_Data_Ready = FALSE;
    }
    return;
}

MPFD3D_COMMAND ParseMPFD3DCommand(BYTE *cmd) {
    MPFD3D_COMMAND i;
    for ( i = 0; i < (MPFD3D_COMMAND)MPFD3D_COMMAND_TABLE_SIZE; i++ ) {
        if ( !memcmppgm2ram((void*)cmd, (ROM void*)MPFD3DCommandString[i], 3) )
```

```
            return i;
    }
    return MPFD3D_CMD_UNKNOWN;
}

BYTE ParseMPFD3DString(void) {
    BYTE *p;
    BYTE v;
    enum { SM_MPFD3D_PARSE_PARAM, SM_MPFD3D_PARSE_SPACE } smParseMPFD3D;

    smParseMPFD3D = SM_MPFD3D_PARSE_PARAM;
    p = (BYTE*)&MPFD3DString[0];

    // Skip white blanks
    while( *p == ' ' )
        p++;

    MPFD3D_argv[0] = (BYTE*)p;
    MPFD3D_argc = 1;

    while( (v = *p) ) {
        switch(smParseMPFD3D) {
            case SM_MPFD3D_PARSE_PARAM:
                if ((v == ' ') || (v == ',')) {
                    *p = '\0';
                    smParseMPFD3D = SM_MPFD3D_PARSE_SPACE;
                }
                else if ((v == '\r') || (v == '\n'))
                    *p = '\0';
                break;
            case SM_MPFD3D_PARSE_SPACE:
                if ( v != ' ' ) {
                    MPFD3D_argv[MPFD3D_argc++] = (BYTE*)p;
                    smParseMPFD3D = SM_MPFD3D_PARSE_PARAM;
                }
                break;
        }
        p++;
        if(MPFD3D_argc == MAX_MPFD3D_ARGS)
            break;
    }
    return MPFD3D_argc;
}

void ExecuteMPFD3DResponse(void) {
    StringToIPAddress(MPFD3D_argv[1], &connectedNode.IPAddr);
    connectedNode.MACAddr.v[0] = shexatob(*(MPFD3D_argv[2]),*(MPFD3D_argv[2]+1));
    connectedNode.MACAddr.v[1] = shexatob(*(MPFD3D_argv[2]+2),*(MPFD3D_argv[2]+3));
    connectedNode.MACAddr.v[2] = shexatob(*(MPFD3D_argv[2]+4),*(MPFD3D_argv[2]+5));
    connectedNode.MACAddr.v[3] = shexatob(*(MPFD3D_argv[2]+6),*(MPFD3D_argv[2]+7));
    connectedNode.MACAddr.v[4] = shexatob(*(MPFD3D_argv[2]+8),*(MPFD3D_argv[2]+9));
```

```
connectedNode.MACAddr.v[5] = shexatob(*(MPFD3D_argv[2]+10),*(MPFD3D_argv[2]+11));
switch(smMPFD3D) {
    case SM_MPFD3D_WISH:
        MPFD3DExtraSocket = UDPOpen(MPFD3D_RECEIVE_PORT,&takeoverNode,  ...
            MPFD3D_RECEIVE_PORT);
        if(UDPIsPutReady(MPFD3DExtraSocket) < (strlenpgm(MPFD3DResponseString[  ...
            MPFD3D_RESP_WISH])+4u)) {
            UDPClose(MPFD3DExtraSocket);
            return;
        }
        UDPPut(connectedNode.IPAddr.v[0]);
        UDPPut(connectedNode.IPAddr.v[1]);
        UDPPut(connectedNode.IPAddr.v[2]);
        UDPPut(connectedNode.IPAddr.v[3]);
        UDPPutROMString((ROM BYTE*)MPFD3DResponseString[MPFD3D_RESP_WISH]);
        UDPFlush();
        UDPClose(MPFD3DExtraSocket);
        smMPFD3D = SM_MPFD3D_CONNECTED;
        break;
    case SM_MPFD3D_TAKEOVER:
        MPFD3DRespondSocket = UDPOpen(MPFD3D_RECEIVE_PORT,&connectedNode,  ...
            MPFD3D_RECEIVE_PORT);
        if(UDPIsPutReady(MPFD3DRespondSocket) < 4u) {
            UDPClose(MPFD3DRespondSocket);
            return;
        }
        UDPPutROMString((ROM BYTE*)MPFD3DResponseString[MPFD3D_RESP_KILL]);
        UDPFlush();
        UDPClose(MPFD3DRespondSocket);

        MPFD3DResponse = MPFD3D_RESP_CONNECT_OK;
        memcpy((void*)&connectedNode, (const void*)&takeoverNode, sizeof(takeoverNode));
        smMPFD3D = SM_MPFD3D_RESPOND;
    case SM_MPFD3D_RESPOND:
        if (MPFD3DResponse == MPFD3D_RESP_NONE)
            break;
        MPFD3DRespondSocket = UDPOpen(MPFD3D_RECEIVE_PORT,&connectedNode,  ...
            MPFD3D_RECEIVE_PORT);
        if(UDPIsPutReady(MPFD3DRespondSocket) < 4u) {
            UDPClose(MPFD3DRespondSocket);
            return;
        }
        if (MPFD3DResponse == MPFD3D_RESP_CONNECT_OK) {
            ChangeLCDScreen(LCD_CONNECTION);
            LCDForceUpdate();
        }
        UDPPutROMArray((ROM BYTE*)MPFD3DResponseString[MPFD3DResponse],4u);
        UDPFlush();
        UDPClose(MPFD3DRespondSocket);
        smMPFD3D = SM_MPFD3D_CONNECTED;
        if (MPFD3DResponse == MPFD3D_RESP_CHANGING) {
```

```
                smMPFD3D = SM_MPFD3D_RESPOND;
                MPFD3DResponse = MPFD3D_RESP_DONE;
            } else
                MPFD3DResponse = MPFD3D_RESP_NONE;
            break;
        case SM_MPFD3D_KILL:
            MPFD3DRespondSocket = UDPOpen(MPFD3D_RECEIVE_PORT,&connectedNode, ...
                MPFD3D_RECEIVE_PORT);
            if(UDPIsPutReady(MPFD3DRespondSocket) < 4u) {
                UDPClose(MPFD3DRespondSocket);
                return;
            }
            UDPPutROMArray((ROM BYTE*)"KILL",4u);
            UDPFlush();
            UDPClose(MPFD3DRespondSocket);

            smMPFD3D = SM_MPFD3D_NOT_CONNECTED;
            MPFD3DResponse = MPFD3D_RESP_NONE;
            if (GetLCDScreen() == LCD_CONNECTION)
                LCDForceUpdate();
            break;
        case SM_MPFD3D_TIMEOUT:
            MPFD3DRespondSocket = UDPOpen(MPFD3D_RECEIVE_PORT,&connectedNode, ...
                MPFD3D_RECEIVE_PORT);
            if(UDPIsPutReady(MPFD3DRespondSocket) < 4u) {
                UDPClose(MPFD3DRespondSocket);
                return;
            }
            UDPPutROMArray((ROM BYTE*)"TIME",4u);
            UDPFlush();
            UDPClose(MPFD3DRespondSocket);

            smMPFD3D = SM_MPFD3D_CONNECTED;
            MPFD3DResponse = MPFD3D_RESP_NONE;
            break;
        default:
            break;
    }
    return;
}


void ExecuteMPFD3DCommand(MPFD3D_COMMAND cmd) {
    StringToIPAddress(MPFD3D_argv[1], &MPFD3DIPAddress);
    switch(cmd) {
        case MPFD3D_CMD_CONNECT:
            if (smMPFD3D != SM_MPFD3D_NOT_CONNECTED) {
                if (memcmp((void*)&MPFD3DConnIPAddress,(void*)&MPFD3DIPAddress,4) == 0) ...
                    {
                    lastMPFD3DActivity = TickGet();
                    smMPFD3D = SM_MPFD3D_RESPOND;
```

```
                MPFD3DResponse = MPFD3D_RESP_CONNECT_OK;
                StringToIPAddress(MPFD3D_argv[1], &MPFD3DConnIPAddress);
                MPFD3DConnMACAddress.v[0] = shexatob(*(MPFD3D_argv[2]),*( ...
                    MPFD3D_argv[2]+1u));
                MPFD3DConnMACAddress.v[1] = shexatob(*(MPFD3D_argv[2]+2u),*( ...
                    MPFD3D_argv[2]+3u));
                MPFD3DConnMACAddress.v[2] = shexatob(*(MPFD3D_argv[2]+4u),*( ...
                    MPFD3D_argv[2]+5u));
                MPFD3DConnMACAddress.v[3] = shexatob(*(MPFD3D_argv[2]+6u),*( ...
                    MPFD3D_argv[2]+7u));
                MPFD3DConnMACAddress.v[4] = shexatob(*(MPFD3D_argv[2]+8u),*( ...
                    MPFD3D_argv[2]+9u));
                MPFD3DConnMACAddress.v[5] = shexatob(*(MPFD3D_argv[2]+10u),*( ...
                    MPFD3D_argv[2]+11u));
            } else {
                if (REACTOR_PULSE_IN == 1u) {
                    smMPFD3D = SM_MPFD3D_RESPOND;
                    MPFD3DResponse = MPFD3D_RESP_DENY;
                } else {
                    smMPFD3D = SM_MPFD3D_WISH;
                    MPFD3DResponse = MPFD3D_RESP_WISH;
                }
            }
        } else {
            lastMPFD3DActivity = TickGet();
            smMPFD3D = SM_MPFD3D_RESPOND;
            MPFD3DResponse = MPFD3D_RESP_CONNECT_OK;
            StringToIPAddress(MPFD3D_argv[1], &MPFD3DConnIPAddress);
            MPFD3DConnMACAddress.v[0] = shexatob(*(MPFD3D_argv[2]),*(MPFD3D_argv ...
                [2]+1u));
            MPFD3DConnMACAddress.v[1] = shexatob(*(MPFD3D_argv[2]+2u),*( ...
                MPFD3D_argv[2]+3u));
            MPFD3DConnMACAddress.v[2] = shexatob(*(MPFD3D_argv[2]+4u),*( ...
                MPFD3D_argv[2]+5u));
            MPFD3DConnMACAddress.v[3] = shexatob(*(MPFD3D_argv[2]+6u),*( ...
                MPFD3D_argv[2]+7u));
            MPFD3DConnMACAddress.v[4] = shexatob(*(MPFD3D_argv[2]+8u),*( ...
                MPFD3D_argv[2]+9u));
            MPFD3DConnMACAddress.v[5] = shexatob(*(MPFD3D_argv[2]+10u),*( ...
                MPFD3D_argv[2]+11u));
        }
        break;
    case MPFD3D_CMD_TAKEOVER:
        lastMPFD3DActivity = TickGet();
        break;
    case MPFD3D_CMD_DATA_START:
        lastMPFD3DActivity = TickGet();
        SendData = TRUE;
        break;
    case MPFD3D_CMD_DATA_STOP:
        lastMPFD3DActivity = TickGet();
```

```
        SendData = FALSE;
        StringToIPAddress(MPFD3D_argv[1], &MPFD3DConnIPAddress);
        MPFD3DConnMACAddress.v[0] = shexatob(*(MPFD3D_argv[2]),*(MPFD3D_argv[2]+1 ...
            u));
        MPFD3DConnMACAddress.v[1] = shexatob(*(MPFD3D_argv[2]+2u),*(MPFD3D_argv ...
            [2]+3u));
        MPFD3DConnMACAddress.v[2] = shexatob(*(MPFD3D_argv[2]+4u),*(MPFD3D_argv ...
            [2]+5u));
        MPFD3DConnMACAddress.v[3] = shexatob(*(MPFD3D_argv[2]+6u),*(MPFD3D_argv ...
            [2]+7u));
        MPFD3DConnMACAddress.v[4] = shexatob(*(MPFD3D_argv[2]+8u),*(MPFD3D_argv ...
            [2]+9u));
        MPFD3DConnMACAddress.v[5] = shexatob(*(MPFD3D_argv[2]+10u),*(MPFD3D_argv ...
            [2]+11u));
        break;
    case MPFD3D_CMD_QUIT:
        SendData = FALSE;
        smMPFD3D = SM_MPFD3D_NOT_CONNECTED;
        break;
    case MPFD3D_CMD_KILL:
        if (GetLCDScreen() == LCD_CONNECTION)
            LCDForceUpdate();
        SendData = FALSE;
        smMPFD3D = SM_MPFD3D_NOT_CONNECTED;
        MPFD3DResponse = MPFD3D_RESP_NONE;
        break;
    case MPFD3D_CMD_REFRESH:
        lastMPFD3DActivity = TickGet();
        StringToIPAddress(MPFD3D_argv[1], &MPFD3DConnIPAddress);
        MPFD3DConnMACAddress.v[0] = shexatob(*(MPFD3D_argv[2]),*(MPFD3D_argv[2]+1 ...
            u));
        MPFD3DConnMACAddress.v[1] = shexatob(*(MPFD3D_argv[2]+2u),*(MPFD3D_argv ...
            [2]+3u));
        MPFD3DConnMACAddress.v[2] = shexatob(*(MPFD3D_argv[2]+4u),*(MPFD3D_argv ...
            [2]+5u));
        MPFD3DConnMACAddress.v[3] = shexatob(*(MPFD3D_argv[2]+6u),*(MPFD3D_argv ...
            [2]+7u));
        MPFD3DConnMACAddress.v[4] = shexatob(*(MPFD3D_argv[2]+8u),*(MPFD3D_argv ...
            [2]+9u));
        MPFD3DConnMACAddress.v[5] = shexatob(*(MPFD3D_argv[2]+10u),*(MPFD3D_argv ...
            [2]+11u));
        break;
    case MPFD3D_CMD_SETTINGS:
        if (memcmp((void*)&MPFD3DConnIPAddress,(void*)&MPFD3DIPAddress,4) == 0) {
            lastMPFD3DActivity = TickGet();
            StringToIPAddress(MPFD3D_argv[1], &MPFD3DConnIPAddress);
            MPFD3DConnMACAddress.v[0] = shexatob(*(MPFD3D_argv[2]),*(MPFD3D_argv ...
                [2]+1u));
            MPFD3DConnMACAddress.v[1] = shexatob(*(MPFD3D_argv[2]+2u),*( ...
                MPFD3D_argv[2]+3u));
            MPFD3DConnMACAddress.v[2] = shexatob(*(MPFD3D_argv[2]+4u),*( ...
```

```
                    MPFD3D_argv[2]+5u));
                MPFD3DConnMACAddress.v[3] = shexatob(*(MPFD3D_argv[2]+6u),*( ...
                    MPFD3D_argv[2]+7u));
                MPFD3DConnMACAddress.v[4] = shexatob(*(MPFD3D_argv[2]+8u),*( ...
                    MPFD3D_argv[2]+9u));
                MPFD3DConnMACAddress.v[5] = shexatob(*(MPFD3D_argv[2]+10u),*( ...
                    MPFD3D_argv[2]+11u));
                if (((unsigned char)REACTOR_ON_IN) == 1u && ((unsigned char) ...
                    REACTOR_PULSE_IN) == 1u) {
                    smMPFD3D = SM_MPFD3D_RESPOND;
                    MPFD3DResponse = MPFD3D_RESP_DENY;
                } else {
                    smMPFD3D = SM_MPFD3D_RESPOND;
                    MPFD3DResponse = MPFD3D_RESP_CHANGING;
                    HV_NewSetpoint = (((unsigned int)(*(MPFD3D_argv[3]+1u)) << 8u) & 0 ...
                        xFF00u) + (((unsigned int)(*(MPFD3D_argv[3]))) & 0x00FFu);
                    LLD_NewSetpoint = (((unsigned int)(*(MPFD3D_argv[3]+3u)) << 8u) & 0 ...
                        xFF00u) + (((unsigned int)(*(MPFD3D_argv[3]+2u))) & 0x00FFu);
                    MonitorUPS = ((*(MPFD3D_argv[3]+4u)) >> 6u) & 0b00000001u;
                    MSV_Res_New = ((*(MPFD3D_argv[3]+4u)) >> 4u) & 0b00000011u;
                    Curr_Res_New = ((*(MPFD3D_argv[3]+4u)) >> 2u) & 0b00000011u;
                    Temp_Res_New = (*(MPFD3D_argv[3]+4u)) & 0b00000011u;
                    if ((*(MPFD3D_argv[3]+4u)) & 0b01000000u == 0b01000000u)
                        ResetHVFuses = 0x01u;
                }
            }
            break;
        default:
            break;
    }
    return;
}

unsigned char MPFD3DConnected(void) {
    if (smMPFD3D == SM_MPFD3D_NOT_CONNECTED)
        return 0u;
    else
        return 1u;
}

void MPFD3DTypeIPAddress(void) {
    lcdType((MPFD3DConnIPAddress.v[0] / 100u) + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[0] / 10u) % 10u + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[0] % 10u) + 0x30u);
    lcdType('.');
    lcdType((MPFD3DConnIPAddress.v[1] / 100u) + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[1] / 10u) % 10u + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[1] % 10u) + 0x30u);
    lcdType('.');
    lcdType((MPFD3DConnIPAddress.v[2] / 100u) + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[2] / 10u) % 10u + 0x30u);
```

```
    lcdType((MPFD3DConnIPAddress.v[2] % 10u) + 0x30u);
    lcdType('.');
    lcdType((MPFD3DConnIPAddress.v[3] / 100u) + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[3] / 10u) % 10u + 0x30u);
    lcdType((MPFD3DConnIPAddress.v[3] % 10u) + 0x30u);
    return;
}

void MPFD3DTypeMACAddress(void) {
    lcdTypeHex(MPFD3DConnMACAddress.v[0]);
    lcdType(':');
    lcdTypeHex(MPFD3DConnMACAddress.v[1]);
    lcdType(':');
    lcdTypeHex(MPFD3DConnMACAddress.v[2]);
    lcdType(':');
    lcdTypeHex(MPFD3DConnMACAddress.v[3]);
    lcdType(':');
    lcdTypeHex(MPFD3DConnMACAddress.v[4]);
    lcdType(':');
    lcdTypeHex(MPFD3DConnMACAddress.v[5]);
    return;
}

void MPFD3DTypeHVbits(void) {
    lcdTypeHex((HV_Setpoint >> 8) & 0xFFu);
    lcdTypeHex(HV_Setpoint & 0xFFu);
    return;
}

void MPFD3DTypeHVvalue(void) {
    float HVfloat;
    unsigned int HVval;
    char buf[10];

    HVfloat = (((float)(HV_Setpoint)) * 500.0 / 1023.0)*10.0;
    HVval = (unsigned int)(HVfloat);
    if (HVval > 1000u)
        lcdType((HVval / 1000u) + 0x30u);
    if (HVval > 100u)
        lcdType((HVval / 100u) % 10u + 0x30u);
    if (HVval > 10u)
        lcdType((HVval / 10u) % 10u + 0x30u);
    else
        lcdType('0');
    lcdType('.');
    lcdType(HVval % 10u + 0x30u);
    return;
}

void MPFD3DTypeLLDbits(void) {
    lcdTypeHex((LLD_Setpoint >> 8u) & 0xFFu);
```

```
    lcdTypeHex(LLD_Setpoint & 0xFFu);
    return;
}

void MPFD3DTypeLLDvalue(void) {
    float LLDfloat;
    unsigned int LLDval;
    char buf[10];

    LLDfloat = (((float)(LLD_Setpoint)) * 2500.0 / 1023.0)*10.0;
    LLDval = (unsigned int)(LLDfloat);
    if (LLDval > 10000u)
        lcdType((LLDval / 10000u) + 0x30u);
    if (LLDval > 1000u)
        lcdType((LLDval / 1000u) % 10u + 0x30u);
    if (LLDval > 100u)
        lcdType((LLDval / 100u) % 10u + 0x30u);
    if (LLDval > 10u)
        lcdType((LLDval / 10u) % 10u + 0x30u);
    else
        lcdType('0');
    lcdType('.');
    lcdType(LLDval % 10 + 0x30);
    return;
}

void MPFD3DTypeSpeed(void) {
    lcdString(SPSString[SpeedIndex]);
    return;
}

void OFFButtonPressed(void) {
    StandbyLED_ON;
    SHUTDOWN_MODE_SELECTED = TRUE;
    ResetSecondTimer3();
    while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) && GetSecondTimer3());
    //1
    if (OFF_BUTTON_IN_BOOL) {
        ResetSecondTimer3();
        while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) && GetSecondTimer3());
        //2
        if (OFF_BUTTON_IN_BOOL) {
            ResetSecondTimer3();
            while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) && GetSecondTimer3())  ...
                ;
            //3
            if (OFF_BUTTON_IN_BOOL) {
                ResetSecondTimer3();
                while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) &&   ...
                    GetSecondTimer3());
                //4
```

```
                    if (OFF_BUTTON_IN_BOOL) {
                        ResetSecondTimer3();
                        while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) &&   ...
                            GetSecondTimer3());
                        //5
                        if (OFF_BUTTON_IN_BOOL) {
                            ResetSecondTimer3();
                            while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) &&   ...
                                GetSecondTimer3());
                            //6
                            if (OFF_BUTTON_IN_BOOL) {
                                ResetSecondTimer3();
                                while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u) &&   ...
                                    GetSecondTimer3());
                                //7
                                if (OFF_BUTTON_IN_BOOL) {
                                    ResetSecondTimer3();
                                    while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u)   ...
                                        && GetSecondTimer3());
                                    //8
                                    if (OFF_BUTTON_IN_BOOL) {
                                        ResetSecondTimer3();
                                        while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF == 0u  ...
                                            ) && GetSecondTimer3());
                                        //9
                                        if (OFF_BUTTON_IN_BOOL) {
                                            ResetSecondTimer3();
                                            while (OFF_BUTTON_IN_BOOL && (PIR2bits.TMR3IF   ...
                                                == 0u) && GetSecondTimer3());
                                            //10
                                            if ((PIR2bits.TMR3IF == 1u) || GetSecondTimer3()) {
                                                PowerLED_OFF;
                                                POWER_OFF_OUT = 1;
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    //NextTimeStamp = GetUTCTimeStamp() + TicksPerSample[0]*10;
    return;
}

BOOL Shutdown_Mode_Selected(void) {
    return SHUTDOWN_MODE_SELECTED;
}
```

```
WORD GetConnectedBoards(void) {
    return ConnectedBoards.Val;
}

unsigned char GetBoardIDs(unsigned char i) {
    unsigned char uc0;
    uc0 = (0x0Fu & BoardID[i]);
    return uc0;
}

void mpfd3d_daughters_ready(void) {
    unsigned char uc0;

    // Load last saved settings from Flash
    if (XEEReadArray(0x00000000,(unsigned char *)&StoredSettings,5u) == XEE_SUCCESS) {
        HV_NewSetpoint = (((unsigned int)StoredSettings[0u]) & 0x00FFu) + (((unsigned int)  ...
            StoredSettings[1u]) << 8u & 0xFF00u);
        if (HV_NewSetpoint < 0xFFFFu)
            HV_Setpoint = HV_NewSetpoint + 0x0001u;
        else
            HV_Setpoint = HV_NewSetpoint - 0x0001u;

        LLD_NewSetpoint = (((unsigned int)StoredSettings[2u]) & 0x00FFu) + (((unsigned int)  ...
            StoredSettings[3u]) << 8u & 0xFF00u);
        if (LLD_NewSetpoint < 0xFFFFu)
            LLD_Setpoint = LLD_NewSetpoint + 0x0001u;
        else
            LLD_Setpoint = LLD_NewSetpoint - 0x0001u;

        MSV_Res_New = (StoredSettings[4u] >> 4u) & 0b00000011u;
        if (MSV_Res_New > 0x00u)
            MSV_Res = 0x00u;
        else
            MSV_Res = 0x01u;

        Curr_Res_New = (StoredSettings[4u] >> 2u) & 0b00000011u;
        if (Curr_Res_New > 0x00u)
            Curr_Res = 0x00u;
        else
            Curr_Res = 0x01u;

        Temp_Res_New = StoredSettings[4u] & 0b00000011u;
        if (Temp_Res_New > 0x00u)
            Temp_Res = 0x00u;
        else
            Temp_Res = 0x01u;

        if ((StoredSettings[4u] & 0b01000000u) == 0b01000000u)
            ResetHVFuses = 0x01u;
        MonitorUPS = (StoredSettings[4u] >> 6u) & 0b00000001u;
    } else {
```

```
        LLD_NewSetpoint = 0x0096u;
        LLD_Setpoint = LLD_NewSetpoint + 0x0001u;
        HV_NewSetpoint = 0x0000u;
        HV_Setpoint = 0x0001u;
        MSV_Res_New = 0x00u;
        MSV_Res = 0x01u;
        Curr_Res_New = 0x00u;
        Curr_Res = 0x01u;
        Temp_Res_New = 0x00u;
        Temp_Res = 0x01u;
        ResetHVFuses = 0x00u;
    }
    SpeedIndexNew = 1u;
    SpeedIndex = 7u;
    uc0 = 0u;
    if ((TRA1_IN + TRA2_IN + TRA3_IN) % 3u > 0u) {
        ChangeLCDScreen(LCD_NONE);
        uc0++;
        lcdString("Board Init Error: A");
        lcdType(0x30u + (TRA1_IN << 2u) + (TRA2_IN << 1u) + TRA3_IN);
        WaitSeconds(1u);
    } else
        ConnectedBoards.bits.b0 = TRA1_IN;

    if ((TRB1_IN + TRB2_IN + TRB3_IN) % 3u > 0u) {
        ChangeLCDScreen(LCD_NONE);
        uc0++;
        lcdString("Board Init Error: B");
        lcdType(0x30u + (TRB1_IN << 2u) + (TRB2_IN << 1u) + TRB3_IN);
        WaitSeconds(1u);
    } else
        ConnectedBoards.bits.b1 = TRB1_IN;

    if ((TRC1_IN + TRC2_IN + TRC3_IN) % 3u > 0u) {
        ChangeLCDScreen(LCD_NONE);
        uc0++;
        lcdString("Board Init Error: C");
        lcdType(0x30u + (TRC1_IN << 2u) + (TRC2_IN << 1u) + TRC3_IN);
        WaitSeconds(1u);
    } else
        ConnectedBoards.bits.b2 = TRC1_IN;

    if ((TRD1_IN + TRD2_IN + TRD3_IN) % 3u > 0u) {
        ChangeLCDScreen(LCD_NONE);
        uc0++;
        lcdString("Board Init Error: D");
        lcdType(0x30u + (TRD1_IN << 2u) + (TRD2_IN << 1u) + TRD3_IN);
        WaitSeconds(1u);
    } else
        ConnectedBoards.bits.b3 = TRD1_IN;
```

```
if ((TRE1_IN + TRE2_IN + TRE3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: E");
    lcdType(0x30u + (TRE1_IN << 2u) + (TRE2_IN << 1u) + TRE3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b4 = TRE1_IN;

if ((TRF1_IN + TRF2_IN + TRF3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: F");
    lcdType(0x30u + (TRF1_IN << 2u) + (TRF2_IN << 1u) + TRF3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b5 = TRF1_IN;

if ((TRG1_IN + TRG2_IN + TRG3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: G");
    lcdType(0x30u + (TRG1_IN << 2u) + (TRG2_IN << 1u) + TRG3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b6 = TRG1_IN;

if ((TRH1_IN + TRH2_IN + TRH3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: H");
    lcdType(0x30u + (TRH1_IN << 2u) + (TRH2_IN << 1u) + TRH3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b7 = TRH1_IN;

if ((TRI1_IN + TRI2_IN + TRI3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: I");
    lcdType(0x30u + (TRI1_IN << 2u) + (TRI2_IN << 1u) + TRI3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b8 = TRI1_IN;

if ((TRJ1_IN + TRJ2_IN + TRJ3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: J");
    lcdType(0x30u + (TRJ1_IN << 2u) + (TRJ2_IN << 1u) + TRJ3_IN);
    WaitSeconds(1u);
```

```
} else
    ConnectedBoards.bits.b9 = TRJ1_IN;

if ((TRL1_IN + TRL2_IN + TRL3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: L");
    lcdType(0x30u + (TRL1_IN << 2u) + (TRL2_IN << 1u) + TRL3_IN); // L3 = 0000 0011
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b11 = TRL1_IN;

if ((TRM1_IN + TRM2_IN + TRM3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: M");
    lcdType(0x30u + (TRM1_IN << 2u) + (TRM2_IN << 1u) + TRM3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b12 = TRM1_IN;

if ((TRN1_IN + TRN2_IN + TRN3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: N");
    lcdType(0x30u + (TRN1_IN << 2u) + (TRN2_IN << 1u) + TRN3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b13 = TRN1_IN;

if ((TRO1_IN + TRO2_IN + TRO3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: O");
    lcdType(0x30u + (TRO1_IN << 2u) + (TRO2_IN << 1u) + TRO3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b14 = TRO1_IN;

if ((TRP1_IN + TRP2_IN + TRP3_IN) % 3u > 0u) {
    ChangeLCDScreen(LCD_NONE);
    uc0++;
    lcdString("Board Init Error: P");
    lcdType(0x30u + (TRP1_IN << 2u) + (TRP2_IN << 1u) + TRP3_IN);
    WaitSeconds(1u);
} else
    ConnectedBoards.bits.b15 = TRP1_IN;

BoardMask = (~ConnectedBoards.Val) & 0xFBFF;
NumBoards = 15u - (TRA1_IN + TRB1_IN + TRC1_IN + TRD1_IN + TRE1_IN + TRF1_IN +  ...
     TRG1_IN + TRH1_IN + TRI1_IN + TRJ1_IN + TRL1_IN + TRM1_IN + TRN1_IN + TRO1_IN  ...
```

```
            + TRP1_IN);
        #if defined(LCD_DEBUG)
            serial_debug(255,NumBoards);
        #endif

        if (uc0 == 0x00u) {
            ChangeLCDScreen(LCD_BOARDS);
            LCDForceUpdate();
            lcdTask();
        } else {
            lcdDBsync();
            lcdTask();
        }
        return;
}

void mpfd3d_collect_data(void) {
        unsigned char IncreaseSpeedIndex = 0u;
        unsigned char DecreaseSpeedIndex = 1u;
        unsigned char rw;
        unsigned int c;
        WORD_VAL ui1, ui2, ui3;
        unsigned int CollectDataCA = 22u;
        unsigned int CollectDataCB = 109u;
        unsigned int CollectDataCC = 196u;
        unsigned int CollectDataCD = 283u;
        unsigned int CollectDataCE = 370u;
        unsigned int CollectDataCF = 457u;
        unsigned int CollectDataCG = 544u;
        unsigned int CollectDataCH = 631u;
        unsigned int CollectDataCI = 718u;
        unsigned int CollectDataCJ = 805u;
        unsigned int CollectDataCL = 892u;
        unsigned int CollectDataCM = 979u;
        unsigned int CollectDataCN = 1066u;
        unsigned int CollectDataCO = 1153u;
        unsigned int CollectDataCP = 1240u;
        unsigned char CollectDataK = 5u;
        unsigned char CollectDataD;

        // 76543210
        // MPFD_Data_Array[0000] = New Time Stamp[5]
        // MPFD_Data_Array[0001] = New Time Stamp[4]
        // MPFD_Data_Array[0002] = New Time Stamp[3]
        // MPFD_Data_Array[0003] = New Time Stamp[2]
        // MPFD_Data_Array[0004] = New Time Stamp[1]
        // MPFD_Data_Array[0005] = New Time Stamp[0]
        // MPFD_Data_Array[0006] = Last Time Stamp[5]
        // MPFD_Data_Array[0007] = Last Time Stamp[4]
        // MPFD_Data_Array[0008] = Last Time Stamp[3]
        // MPFD_Data_Array[0009] = Last Time Stamp[2]
```

```
// MPFD_Data_Array[0010] = Last Time Stamp[1]
// MPFD_Data_Array[0011] = Last Time Stamp[0]
// MPFD_Data_Array[0012] = 0b01235555 - 1-Reactor ON, 2-Reactor PULSE, 3-AC Good,   ...
      5555-Sample Rate
// MPFD_Data_Array[0013] = Connected Boards _PONMLJI
// MPFD_Data_Array[0014] = Connected Boards HGFEDCBA
// MPFD_Data_Array[0015] = HV_Setpoint
// MPFD_Data_Array[0016] = HV_Setpoint
// MPFD_Data_Array[0017] = LLD_setpoint
// MPFD_Data_Array[0018] = LLD_Setpoint
// MPFD_Data_Array[0019] = Speed
// MPFD_Data_Array[0020] = 0b11223300 - 11-MSV_Res, 22-Curr_Res, 33-Temp_Res
// MPFD_Data_Array[0021] = Motherboard Temperature
// MPFD_Data_Array[0022] - MPFD_Data_Array[0111] = String A Data
// MPFD_Data_Array[0112] - MPFD_Data_Array[0201] = String B Data
// MPFD_Data_Array[0202] - MPFD_Data_Array[0291] = String C Data
// MPFD_Data_Array[0292] - MPFD_Data_Array[0381] = String D Data
// MPFD_Data_Array[0382] - MPFD_Data_Array[0471] = String E Data

// MPFD_Data_Array[0472] - MPFD_Data_Array[0561] = String F Data
// MPFD_Data_Array[0562] - MPFD_Data_Array[0651] = String G Data
// MPFD_Data_Array[0652] - MPFD_Data_Array[0741] = String H Data
// MPFD_Data_Array[0742] - MPFD_Data_Array[0831] = String I Data
// MPFD_Data_Array[0832] - MPFD_Data_Array[0921] = String J Data

// MPFD_Data_Array[0922] - MPFD_Data_Array[1011] = String L Data
// MPFD_Data_Array[1012] - MPFD_Data_Array[1101] = String M Data
// MPFD_Data_Array[1102] - MPFD_Data_Array[1191] = String N Data
// MPFD_Data_Array[1192] - MPFD_Data_Array[1281] = String O Data
// MPFD_Data_Array[1282] - MPFD_Data_Array[1371] = String P Data

LastTriggeredTimeStamp = TriggeredTimeStamp;
TriggeredTimeStamp = GetUTCTimeStampVal();
MPFD_Data_Array[0] = TriggeredTimeStamp.v[5];
MPFD_Data_Array[1] = TriggeredTimeStamp.v[4];
MPFD_Data_Array[2] = TriggeredTimeStamp.v[3];
MPFD_Data_Array[3] = TriggeredTimeStamp.v[2];
MPFD_Data_Array[4] = TriggeredTimeStamp.v[1];
MPFD_Data_Array[5] = TriggeredTimeStamp.v[0];
MPFD_Data_Array[6] = LastTriggeredTimeStamp.v[5];
MPFD_Data_Array[7] = LastTriggeredTimeStamp.v[4];
MPFD_Data_Array[8] = LastTriggeredTimeStamp.v[3];
MPFD_Data_Array[9] = LastTriggeredTimeStamp.v[2];
MPFD_Data_Array[10] = LastTriggeredTimeStamp.v[1];
MPFD_Data_Array[11] = LastTriggeredTimeStamp.v[0];
// 76543210
MPFD_Data_Array[12] = 0x00u | (((unsigned char)REACTOR_ON_IN) << 6u) | (((unsigned char  ...
      )REACTOR_PULSE_IN) << 5u) | (((unsigned char)UPS_AC_GOOD_IN) << 4u) |   ...
      SpeedIndex;
MPFD_Data_Array[13] = 0x00u | ConnectedBoards.byte.HB;
MPFD_Data_Array[14] = 0x00u | ConnectedBoards.byte.LB;
```

```
MPFD_Data_Array[15] = 0x00u | ((HV_Setpoint >> 8u) & 0xFFu);
MPFD_Data_Array[16] = 0x00u | (HV_Setpoint & 0xFFu);
MPFD_Data_Array[17] = 0x00u | ((LLD_Setpoint >> 8u) & 0xFFu);
MPFD_Data_Array[18] = 0x00u | (LLD_Setpoint & 0xFFu);
MPFD_Data_Array[19] = 0x00u | SpeedIndex;
MPFD_Data_Array[20] = 0x00u | (MSV_Res << 6u) | (Curr_Res << 4u) | (Temp_Res << 2u) |   ...
    SpeedIndex;
MPFD_Data_Array[21] = 0x00u | ReadTemp();

MPFD_Data_Array[CollectDataCA] = 0x00u;
MPFD_Data_Array[CollectDataCB] = 0x00u;
MPFD_Data_Array[CollectDataCC] = 0x00u;
MPFD_Data_Array[CollectDataCD] = 0x00u;
MPFD_Data_Array[CollectDataCE] = 0x00u;
MPFD_Data_Array[CollectDataCF] = 0x00u;
MPFD_Data_Array[CollectDataCG] = 0x00u;
MPFD_Data_Array[CollectDataCH] = 0x00u;
MPFD_Data_Array[CollectDataCI] = 0x00u;
MPFD_Data_Array[CollectDataCJ] = 0x00u;
MPFD_Data_Array[CollectDataCL] = 0x00u;
MPFD_Data_Array[CollectDataCM] = 0x00u;
MPFD_Data_Array[CollectDataCN] = 0x00u;
MPFD_Data_Array[CollectDataCO] = 0x00u;
MPFD_Data_Array[CollectDataCP] = 0x00u;

// MPFD_Data_Array[0022] – MPFD_Data_Array[0110] = String A Data
// MPFD_Data_Array[0111] – MPFD_Data_Array[0199] = String B Data
// MPFD_Data_Array[0200] – MPFD_Data_Array[0288] = String C Data
// MPFD_Data_Array[0289] – MPFD_Data_Array[0377] = String D Data
// MPFD_Data_Array[0378] – MPFD_Data_Array[0466] = String E Data

// MPFD_Data_Array[0467] – MPFD_Data_Array[0555] = String F Data
// MPFD_Data_Array[0556] – MPFD_Data_Array[0644] = String G Data
// MPFD_Data_Array[0645] – MPFD_Data_Array[0733] = String H Data
// MPFD_Data_Array[0734] – MPFD_Data_Array[0822] = String I Data
// MPFD_Data_Array[0823] – MPFD_Data_Array[0911] = String J Data

// MPFD_Data_Array[0912] – MPFD_Data_Array[1000] = String L Data
// MPFD_Data_Array[1001] – MPFD_Data_Array[1089] = String M Data
// MPFD_Data_Array[1090] – MPFD_Data_Array[1178] = String N Data
// MPFD_Data_Array[1179] – MPFD_Data_Array[1267] = String O Data
// MPFD_Data_Array[1268] – MPFD_Data_Array[1356] = String P Data

CollectDataD = 0u;

//while ((TRIGGER_IN_BOOL) && (TRA1_IN == 0u || TRB1_IN == 0u || TRC1_IN == 0u ||   ...
    TRD1_IN == 0u || TRE1_IN == 0u || TRF1_IN == 0u || TRG1_IN == 0u || TRH1_IN == 0u ||   ...
    TRI1_IN == 0u || TRJ1_IN == 0u || TRL1_IN == 0u || TRM1_IN == 0u || TRN1_IN == 0u ||   ...
    TRO1_IN == 0u || TRP1_IN == 0u));
while ((TRIGGER_IN_BOOL) && (TRB1_IN == 0u || TRC1_IN == 0u || TRD1_IN == 0u ||   ...
    TRE1_IN == 0u || TRF1_IN == 0u || TRI1_IN == 0u || TRJ1_IN == 0u || TRN1_IN == 0u));
```

```
TW_TRIS = 0;
ResetSyncTimerShort();

while ((CollectDataD <= 230u) && (TRIGGER_IN_BOOL)) {
    switch(CollectDataD) {
        case 1:
            TW_TRIS = (Speed_New >> 1u) & 0x01u;
            break;
        case 3:
            TW_TRIS = Speed_New & 0x01u;
            break;
        case 5:
            TW_TRIS = (MSV_Res_New >> 1u) & 0x01u;
            break;
        case 7:
            TW_TRIS = MSV_Res_New & 0x01u;
            break;
        case 9:
            TW_TRIS = (Curr_Res_New >> 1u) & 0x01u;
            break;
        case 11:
            TW_TRIS = Curr_Res_New & 0x01u;
            break;
        case 13:
            TW_TRIS = (Temp_Res_New >> 1u) & 0x01u;
            break;
        case 15:
            TW_TRIS = Temp_Res_New & 0x01u;
            break;
        case 17:
            TW_TRIS = (LLD_NewSetpoint >> 9u) & 0x0001u;
            break;
        case 19:
            TW_TRIS = (LLD_NewSetpoint >> 8u) & 0x0001u;
            break;
        case 21:
            TW_TRIS = (LLD_NewSetpoint >> 7u) & 0x0001u;
            break;
        case 23:
            TW_TRIS = (LLD_NewSetpoint >> 6u) & 0x0001u;
            break;
        case 25:
            TW_TRIS = (LLD_NewSetpoint >> 5u) & 0x0001u;
            break;
        case 27:
            TW_TRIS = (LLD_NewSetpoint >> 4u) & 0x0001u;
            break;
        case 29:
            TW_TRIS = (LLD_NewSetpoint >> 3u) & 0x0001u;
            break;
        case 31:
```

```
            TW_TRIS = (LLD_NewSetpoint >> 2u) & 0x0001u;
            break;
        case 33:
            TW_TRIS = (LLD_NewSetpoint >> 1u) & 0x0001u;
            break;
        case 35:
            TW_TRIS = LLD_NewSetpoint & 0x0001u;
            break;
        case 37:
            TW_TRIS = (HV_NewSetpoint >> 9u) & 0x0001u;
            break;
        case 39:
            TW_TRIS = (HV_NewSetpoint >> 8u) & 0x0001u;
            break;
        case 41:
            TW_TRIS = (HV_NewSetpoint >> 7u) & 0x0001u;
            break;
        case 43:
            TW_TRIS = (HV_NewSetpoint >> 6u) & 0x0001u;
            break;
        case 45:
            TW_TRIS = (HV_NewSetpoint >> 5u) & 0x0001u;
            break;
        case 47:
            TW_TRIS = (HV_NewSetpoint >> 4u) & 0x0001u;
            break;
        case 49:
            TW_TRIS = (HV_NewSetpoint >> 3u) & 0x0001u;
            break;
        case 51:
            TW_TRIS = (HV_NewSetpoint >> 2u) & 0x0001u;
            break;
        case 53:
            TW_TRIS = (HV_NewSetpoint >> 1u) & 0x0001u;
            break;
        case 55:
            TW_TRIS = HV_NewSetpoint & 0x0001u;
            break;
        default:
            TW_TRIS = 0;
    }

    MPFD_Write_Done();

    switch(CollectDataK) {
        case 7:
            //MPFD_Data_Array[(++CollectDataCA)] = 0x00u | (TRA1_IN << 7) | (TRA2_IN  ...
                << 6) | (TRA3_IN << 5);
            MPFD_Data_Array[(++CollectDataCA)] = 0x00u;
            MPFD_Data_Array[(++CollectDataCB)] = 0x00u | (TRB1_IN << 7) | (TRB2_IN <<  ...
                6) | (TRB3_IN << 5);
```

```
        MPFD_Data_Array[(++CollectDataCC)] = 0x00u | (TRC1_IN << 7) | (TRC2_IN <<  ...
            6) | (TRC3_IN << 5);
        MPFD_Data_Array[(++CollectDataCD)] = 0x00u | (TRD1_IN << 7) | (TRD2_IN <<  ...
            6) | (TRD3_IN << 5);
        MPFD_Data_Array[(++CollectDataCE)] = 0x00u | (TRE1_IN << 7) | (TRE2_IN <<  ...
            6) | (TRE3_IN << 5);
        MPFD_Data_Array[(++CollectDataCF)] = 0x00u | (TRF1_IN << 7) | (TRF2_IN <<  ...
            6) | (TRF3_IN << 5);
        //MPFD_Data_Array[(++CollectDataCG)] = 0x00u | (TRG1_IN << 7) | (TRG2_IN  ...
            << 6) | (TRG3_IN << 5);
        MPFD_Data_Array[(++CollectDataCG)] = 0x00u;
        //MPFD_Data_Array[(++CollectDataCH)] = 0x00u | (TRH1_IN << 7) | (TRH2_IN  ...
            << 6) | (TRH3_IN << 5);
        MPFD_Data_Array[(++CollectDataCH)] = 0x00u;
        MPFD_Data_Array[(++CollectDataCI)] = 0x00u | (TRI1_IN << 7) | (TRI2_IN << 6)  ...
            | (TRI3_IN << 5);
        MPFD_Data_Array[(++CollectDataCJ)] = 0x00u | (TRJ1_IN << 7) | (TRJ2_IN << 6)  ...
            | (TRJ3_IN << 5);
        //MPFD_Data_Array[(++CollectDataCL)] = 0x00u | (TRL1_IN << 7) | (TRL2_IN << ...
             6) | (TRL3_IN << 5);
        MPFD_Data_Array[(++CollectDataCL)] = 0x00u;
        //MPFD_Data_Array[(++CollectDataCM)] = 0x00u | (TRM1_IN << 7) | (TRM2_IN  ...
            << 6) | (TRM3_IN << 5);
        MPFD_Data_Array[(++CollectDataCM)] = 0x00u;
        MPFD_Data_Array[(++CollectDataCN)] = 0x00u | (TRN1_IN << 7) | (TRN2_IN <<  ...
            6) | (TRN3_IN << 5);
        //MPFD_Data_Array[(++CollectDataCO)] = 0x00u | (TRO1_IN << 7) | (TRO2_IN  ...
            << 6) | (TRO3_IN << 5);
        MPFD_Data_Array[(++CollectDataCO)] = 0x00u;
        //MPFD_Data_Array[(++CollectDataCP)] = 0x00u | (TRP1_IN << 7) | (TRP2_IN  ...
            << 6) | (TRP3_IN << 5);
        MPFD_Data_Array[(++CollectDataCP)] = 0x00u;
        CollectDataK = 4u;
        break;
    case 6:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN << 6) | (TRA2_IN << 5) | (  ...
            TRA3_IN << 4);
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN << 6) | (TRB2_IN << 5) | (  ...
            TRB3_IN << 4);
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN << 6) | (TRC2_IN << 5) | (  ...
            TRC3_IN << 4);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN << 6) | (TRD2_IN << 5) | (  ...
            TRD3_IN << 4);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN << 6) | (TRE2_IN << 5) | (  ...
            TRE3_IN << 4);
        MPFD_Data_Array[CollectDataCF] |= (TRF1_IN << 6) | (TRF2_IN << 5) | (  ...
            TRF3_IN << 4);
        //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN << 6) | (TRG2_IN << 5) | (  ...
            TRG3_IN << 4);
        //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN << 6) | (TRH2_IN << 5) | (  ...
            TRH3_IN << 4);
```

```
        MPFD_Data_Array[CollectDataCI] |= (TRI1_IN << 6) | (TRI2_IN << 5) | (TRI3_IN  ...
            << 4);
        MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN << 6) | (TRJ2_IN << 5) | (TRJ3_IN  ...
            << 4);
        //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN << 6) | (TRL2_IN << 5) | (  ...
            TRL3_IN << 4);
        //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN << 6) | (TRM2_IN << 5) | (  ...
            TRM3_IN << 4);
        MPFD_Data_Array[CollectDataCN] |= (TRN1_IN << 6) | (TRN2_IN << 5) | (  ...
            TRN3_IN << 4);
        //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN << 6) | (TRO2_IN << 5) | (  ...
            TRO3_IN << 4);
        //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN << 6) | (TRP2_IN << 5) | (  ...
            TRP3_IN << 4);
        CollectDataK = 3u;
        break;
    case 5:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN << 5) | (TRA2_IN << 4) | (  ...
            TRA3_IN << 3);
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN << 5) | (TRB2_IN << 4) | (  ...
            TRB3_IN << 3);
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN << 5) | (TRC2_IN << 4) | (  ...
            TRC3_IN << 3);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN << 5) | (TRD2_IN << 4) | (  ...
            TRD3_IN << 3);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN << 5) | (TRE2_IN << 4) | (  ...
            TRE3_IN << 3);
        MPFD_Data_Array[CollectDataCF] |= (TRF1_IN << 5) | (TRF2_IN << 4) | (  ...
            TRF3_IN << 3);
        //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN << 5) | (TRG2_IN << 4) | (  ...
            TRG3_IN << 3);
        //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN << 5) | (TRH2_IN << 4) | (  ...
            TRH3_IN << 3);
        MPFD_Data_Array[CollectDataCI] |= (TRI1_IN << 5) | (TRI2_IN << 4) | (TRI3_IN  ...
            << 3);
        MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN << 5) | (TRJ2_IN << 4) | (TRJ3_IN  ...
            << 3);
        //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN << 5) | (TRL2_IN << 4) | (  ...
            TRL3_IN << 3);
        //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN << 5) | (TRM2_IN << 4) | (  ...
            TRM3_IN << 3);
        MPFD_Data_Array[CollectDataCN] |= (TRN1_IN << 5) | (TRN2_IN << 4) | (  ...
            TRN3_IN << 3);
        //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN << 5) | (TRO2_IN << 4) | (  ...
            TRO3_IN << 3);
        //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN << 5) | (TRP2_IN << 4) | (  ...
            TRP3_IN << 3);
        CollectDataK = 2u;
        break;
    case 4:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN << 4) | (TRA2_IN << 3) | (  ...
```

```
            TRA3_IN << 2);
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN << 4) | (TRB2_IN << 3) | ( ...
            TRB3_IN << 2);
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN << 4) | (TRC2_IN << 3) | ( ...
            TRC3_IN << 2);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN << 4) | (TRD2_IN << 3) | ( ...
            TRD3_IN << 2);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN << 4) | (TRE2_IN << 3) | ( ...
            TRE3_IN << 2);
        MPFD_Data_Array[CollectDataCF] |= (TRF1_IN << 4) | (TRF2_IN << 3) | ( ...
            TRF3_IN << 2);
        //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN << 4) | (TRG2_IN << 3) | ( ...
            TRG3_IN << 2);
        //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN << 4) | (TRH2_IN << 3) | ( ...
            TRH3_IN << 2);
        MPFD_Data_Array[CollectDataCI] |= (TRI1_IN << 4) | (TRI2_IN << 3) | (TRI3_IN ...
            << 2);
        MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN << 4) | (TRJ2_IN << 3) | (TRJ3_IN ...
            << 2);
        //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN << 4) | (TRL2_IN << 3) | ( ...
            TRL3_IN << 2);
        //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN << 4) | (TRM2_IN << 3) | ( ...
            TRM3_IN << 2);
        MPFD_Data_Array[CollectDataCN] |= (TRN1_IN << 4) | (TRN2_IN << 3) | ( ...
            TRN3_IN << 2);
        //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN << 4) | (TRO2_IN << 3) | ( ...
            TRO3_IN << 2);
        //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN << 4) | (TRP2_IN << 3) | ( ...
            TRP3_IN << 2);
        CollectDataK = 1u;
        break;
    case 3:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN << 3) | (TRA2_IN << 2) | ( ...
            TRA3_IN << 1);
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN << 3) | (TRB2_IN << 2) | ( ...
            TRB3_IN << 1);
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN << 3) | (TRC2_IN << 2) | ( ...
            TRC3_IN << 1);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN << 3) | (TRD2_IN << 2) | ( ...
            TRD3_IN << 1);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN << 3) | (TRE2_IN << 2) | ( ...
            TRE3_IN << 1);
        MPFD_Data_Array[CollectDataCF] |= (TRF1_IN << 3) | (TRF2_IN << 2) | ( ...
            TRF3_IN << 1);
        //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN << 3) | (TRG2_IN << 2) | ( ...
            TRG3_IN << 1);
        //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN << 3) | (TRH2_IN << 2) | ( ...
            TRH3_IN << 1);
        MPFD_Data_Array[CollectDataCI] |= (TRI1_IN << 3) | (TRI2_IN << 2) | (TRI3_IN ...
            << 1);
        MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN << 3) | (TRJ2_IN << 2) | (TRJ3_IN ...
```

```
            << 1);
        //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN << 3) | (TRL2_IN << 2) | ( ...
            TRL3_IN << 1);
        //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN << 3) | (TRM2_IN << 2) | ( ...
            TRM3_IN << 1);
        MPFD_Data_Array[CollectDataCN] |= (TRN1_IN << 3) | (TRN2_IN << 2) | ( ...
            TRN3_IN << 1);
        //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN << 3) | (TRO2_IN << 2) | ( ...
            TRO3_IN << 1);
        //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN << 3) | (TRP2_IN << 2) | ( ...
            TRP3_IN << 1);
        CollectDataK = 0u;
        break;
    case 2:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN << 2) | (TRA2_IN << 1) | ( ...
            TRA3_IN);
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN << 2) | (TRB2_IN << 1) | ( ...
            TRB3_IN);
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN << 2) | (TRC2_IN << 1) | ( ...
            TRC3_IN);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN << 2) | (TRD2_IN << 1) | ( ...
            TRD3_IN);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN << 2) | (TRE2_IN << 1) | ( ...
            TRE3_IN);
        MPFD_Data_Array[CollectDataCF] |= (TRF1_IN << 2) | (TRF2_IN << 1) | ( ...
            TRF3_IN);
        //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN << 2) | (TRG2_IN << 1) | ( ...
            TRG3_IN);
        //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN << 2) | (TRH2_IN << 1) | ( ...
            TRH3_IN);
        MPFD_Data_Array[CollectDataCI] |= (TRI1_IN << 2) | (TRI2_IN << 1) | (TRI3_IN);
        MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN << 2) | (TRJ2_IN << 1) | (TRJ3_IN) ...
            ;
        //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN << 2) | (TRL2_IN << 1) | ( ...
            TRL3_IN);
        //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN << 2) | (TRM2_IN << 1) | ( ...
            TRM3_IN);
        MPFD_Data_Array[CollectDataCN] |= (TRN1_IN << 2) | (TRN2_IN << 1) | ( ...
            TRN3_IN);
        //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN << 2) | (TRO2_IN << 1) | ( ...
            TRO3_IN);
        //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN << 2) | (TRP2_IN << 1) | ( ...
            TRP3_IN);
        CollectDataK = 7u;
        break;
    case 1:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN << 1) | (TRA2_IN);
        //MPFD_Data_Array[(++CollectDataCA)] = 0x00u | (TRA3_IN << 7);
        MPFD_Data_Array[(++CollectDataCA)] = 0x00u;
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN << 1) | (TRB2_IN);
        MPFD_Data_Array[(++CollectDataCB)] = 0x00u | (TRB3_IN << 7);
```

```
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN << 1) | (TRC2_IN);
        MPFD_Data_Array[(++CollectDataCC)] = 0x00u | (TRC3_IN << 7);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN << 1) | (TRD2_IN);
        MPFD_Data_Array[(++CollectDataCD)] = 0x00u | (TRD3_IN << 7);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN << 1) | (TRE2_IN);
        MPFD_Data_Array[(++CollectDataCE)] = 0x00u | (TRE3_IN << 7);
        MPFD_Data_Array[CollectDataCF] |= (TRF1_IN << 1) | (TRF2_IN);
        MPFD_Data_Array[(++CollectDataCF)] = 0x00u | (TRF3_IN << 7);
        //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN << 1) | (TRG2_IN);
        //MPFD_Data_Array[(++CollectDataCG)] = 0x00u | (TRG3_IN << 7);
        MPFD_Data_Array[(++CollectDataCG)] = 0x00u;
        //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN << 1) | (TRH2_IN);
        //MPFD_Data_Array[(++CollectDataCH)] = 0x00u | (TRH3_IN << 7);
        MPFD_Data_Array[(++CollectDataCH)] = 0x00u;
        MPFD_Data_Array[CollectDataCI] |= (TRI1_IN << 1) | (TRI2_IN);
        MPFD_Data_Array[(++CollectDataCI)] = 0x00u | (TRI3_IN << 7);
        MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN << 1) | (TRJ2_IN);
        MPFD_Data_Array[(++CollectDataCJ)] = 0x00u | (TRJ3_IN << 7);
        //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN << 1) | (TRL2_IN);
        //MPFD_Data_Array[(++CollectDataCL)] = 0x00u | (TRL3_IN << 7);
        MPFD_Data_Array[(++CollectDataCL)] = 0x00u;
        //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN << 1) | (TRM2_IN);
        //MPFD_Data_Array[(++CollectDataCM)] = 0x00u | (TRM3_IN << 7);
        MPFD_Data_Array[(++CollectDataCM)] = 0x00u;
        MPFD_Data_Array[CollectDataCN] |= (TRN1_IN << 1) | (TRN2_IN);
        MPFD_Data_Array[(++CollectDataCN)] = 0x00u | (TRN3_IN << 7);
        //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN << 1) | (TRO2_IN);
        //MPFD_Data_Array[(++CollectDataCO)] = 0x00u | (TRO3_IN << 7);
        MPFD_Data_Array[(++CollectDataCO)] = 0x00u;
        //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN << 1) | (TRP2_IN);
        //MPFD_Data_Array[(++CollectDataCP)] = 0x00u | (TRP3_IN << 7);
        MPFD_Data_Array[(++CollectDataCP)] = 0x00u;
        CollectDataK = 6u;
        break;
    case 0:
        //MPFD_Data_Array[CollectDataCA] |= (TRA1_IN);
        //MPFD_Data_Array[(++CollectDataCA)] = 0x00u | (TRA2_IN << 7) | (TRA3_IN  ...
            << 6);
        MPFD_Data_Array[(++CollectDataCA)] = 0x00u;
        MPFD_Data_Array[CollectDataCB] |= (TRB1_IN);
        MPFD_Data_Array[(++CollectDataCB)] = 0x00u | (TRB2_IN << 7) | (TRB3_IN <<  ...
            6);
        MPFD_Data_Array[CollectDataCC] |= (TRC1_IN);
        MPFD_Data_Array[(++CollectDataCC)] = 0x00u | (TRC2_IN << 7) | (TRC3_IN <<  ...
            6);
        MPFD_Data_Array[CollectDataCD] |= (TRD1_IN);
        MPFD_Data_Array[(++CollectDataCD)] = 0x00u | (TRD2_IN << 7) | (TRD3_IN <<  ...
            6);
        MPFD_Data_Array[CollectDataCE] |= (TRE1_IN);
        MPFD_Data_Array[(++CollectDataCE)] = 0x00u | (TRE2_IN << 7) | (TRE3_IN <<  ...
            6);
```

```
            MPFD_Data_Array[CollectDataCF] |= (TRF1_IN);
            MPFD_Data_Array[(++CollectDataCF)] = 0x00u | (TRF2_IN << 7) | (TRF3_IN <<  ...
                6);
            //MPFD_Data_Array[CollectDataCG] |= (TRG1_IN);
            //MPFD_Data_Array[(++CollectDataCG)] = 0x00u | (TRG2_IN << 7) | (TRG3_IN  ...
                << 6);
            MPFD_Data_Array[(++CollectDataCG)] = 0x00u;
            //MPFD_Data_Array[CollectDataCH] |= (TRH1_IN);
            //MPFD_Data_Array[(++CollectDataCH)] = 0x00u | (TRH2_IN << 7) | (TRH3_IN  ...
                << 6);
            MPFD_Data_Array[(++CollectDataCH)] = 0x00u;
            MPFD_Data_Array[CollectDataCI] |= (TRI1_IN);
            MPFD_Data_Array[(++CollectDataCI)] = 0x00u | (TRI2_IN << 7) | (TRI3_IN << 6);
            MPFD_Data_Array[CollectDataCJ] |= (TRJ1_IN);
            MPFD_Data_Array[(++CollectDataCJ)] = 0x00u | (TRJ2_IN << 7) | (TRJ3_IN << 6);
            //MPFD_Data_Array[CollectDataCL] |= (TRL1_IN);
            //MPFD_Data_Array[(++CollectDataCL)] = 0x00u | (TRL2_IN << 7) | (TRL3_IN <<  ...
                6);
            MPFD_Data_Array[(++CollectDataCL)] = 0x00u;
            //MPFD_Data_Array[CollectDataCM] |= (TRM1_IN);
            //MPFD_Data_Array[(++CollectDataCM)] = 0x00u | (TRM2_IN << 7) | (TRM3_IN  ...
                << 6);
            MPFD_Data_Array[(++CollectDataCM)] = 0x00u;
            MPFD_Data_Array[CollectDataCN] |= (TRN1_IN);
            MPFD_Data_Array[(++CollectDataCN)] = 0x00u | (TRN2_IN << 7) | (TRN3_IN <<  ...
                6);
            //MPFD_Data_Array[CollectDataCO] |= (TRO1_IN);
            //MPFD_Data_Array[(++CollectDataCO)] = 0x00u | (TRO2_IN << 7) | (TRO3_IN  ...
                << 6);
            MPFD_Data_Array[(++CollectDataCO)] = 0x00u;
            //MPFD_Data_Array[CollectDataCP] |= (TRP1_IN);
            //MPFD_Data_Array[(++CollectDataCP)] = 0x00u | (TRP2_IN << 7) | (TRP3_IN  ...
                << 6);
            MPFD_Data_Array[(++CollectDataCP)] = 0x00u;
            CollectDataK = 5u;
        }

        MPFD_Read_Done();
        ++CollectDataD;
    }

    TW_TRIS = 0;

    MPFD_Write_Done();

    //MPFD_Data_Array[108u] |= TRA1_IN;
    MPFD_Data_Array[195u] |= TRB1_IN;
    MPFD_Data_Array[282u] |= TRC1_IN;
    MPFD_Data_Array[369u] |= TRD1_IN;
    MPFD_Data_Array[456u] |= TRE1_IN;
    MPFD_Data_Array[543u] |= TRF1_IN;
```

```
//MPFD_Data_Array[630u] |= TRG1_IN;
//MPFD_Data_Array[717u] |= TRH1_IN;
MPFD_Data_Array[804u] |= TRI1_IN;
MPFD_Data_Array[891u] |= TRJ1_IN;
//MPFD_Data_Array[978u] |= TRL1_IN;
//MPFD_Data_Array[1065u] |= TRM1_IN;
MPFD_Data_Array[1152u] |= TRN1_IN;
//MPFD_Data_Array[1239u] |= TRO1_IN;
//MPFD_Data_Array[1326u] |= TRP1_IN;

//BoardID[0u] = (BoardID[0u] & 0xF0u) | (TRA2_IN << 3) | (TRA3_IN << 2);
BoardID[1u] = (BoardID[1u] & 0xF0u) | (TRB2_IN << 3) | (TRB3_IN << 2);
BoardID[2u] = (BoardID[2u] & 0xF0u) | (TRC2_IN << 3) | (TRC3_IN << 2);
BoardID[3u] = (BoardID[3u] & 0xF0u) | (TRD2_IN << 3) | (TRD3_IN << 2);
BoardID[4u] = (BoardID[4u] & 0xF0u) | (TRE2_IN << 3) | (TRE3_IN << 2);
BoardID[5u] = (BoardID[5u] & 0xF0u) | (TRF2_IN << 3) | (TRF3_IN << 2);
//BoardID[6u] = (BoardID[6u] & 0xF0u) | (TRG2_IN << 3) | (TRG3_IN << 2);
//BoardID[7u] = (BoardID[7u] & 0xF0u) | (TRH2_IN << 3) | (TRH3_IN << 2);
BoardID[8u] = (BoardID[8u] & 0xF0u) | (TRI2_IN << 3) | (TRI3_IN << 2);
BoardID[9u] = (BoardID[9u] & 0xF0u) | (TRJ2_IN << 3) | (TRJ3_IN << 2);
//BoardID[10u] = (BoardID[10u] & 0xF0u) | (TRL2_IN << 3) | (TRL3_IN << 2);
//BoardID[11u] = (BoardID[11u] & 0xF0u) | (TRM2_IN << 3) | (TRM3_IN << 2);
BoardID[12u] = (BoardID[12u] & 0xF0u) | (TRN2_IN << 3) | (TRN3_IN << 2);
//BoardID[13u] = (BoardID[13u] & 0xF0u) | (TRO2_IN << 3) | (TRO3_IN << 2);
//BoardID[14u] = (BoardID[14u] & 0xF0u) | (TRP2_IN << 3) | (TRP3_IN << 2);

MPFD_Read_Done();

// Let Daughter Boards write
TW_TRIS = 0;

MPFD_Write_Done();

//BoardID[0u] |= (TRA1_IN << 1) | TRA2_IN;
BoardID[1u] |= (TRB1_IN << 1) | TRB2_IN;
BoardID[2u] |= (TRC1_IN << 1) | TRC2_IN;
BoardID[3u] |= (TRD1_IN << 1) | TRD2_IN;
BoardID[4u] |= (TRE1_IN << 1) | TRE2_IN;
BoardID[5u] |= (TRF1_IN << 1) | TRF2_IN;
//BoardID[6u] |= (TRG1_IN << 1) | TRG2_IN;
//BoardID[7u] |= (TRH1_IN << 1) | TRH2_IN;
BoardID[8u] |= (TRI1_IN << 1) | TRI2_IN;
BoardID[9u] |= (TRJ1_IN << 1) | TRJ2_IN;
//BoardID[10u] |= (TRL1_IN << 1) | TRL2_IN;
//BoardID[11u] |= (TRM1_IN << 1) | TRM2_IN;
BoardID[12u] |= (TRN1_IN << 1) | TRN2_IN;
//BoardID[13u] |= (TRO1_IN << 1) | TRO2_IN;
//BoardID[14u] |= (TRP1_IN << 1) | TRP2_IN;

//IncreaseSpeedIndex = 0x00u | (TRA3_IN & (ConnectedBoards.Val & 0x01u)) | (TRB3_IN & (( ...
    ConnectedBoards.Val >> 1) & 0x01u)) | (TRC3_IN & ((ConnectedBoards.Val >> 2) & 0x01u) ...
```

```
    ) | (TRD3_IN & ((ConnectedBoards.Val >> 3) & 0x01u)) | (TRE3_IN & ((ConnectedBoards. ...
    Val >> 4) & 0x01u)) | (TRF3_IN & ((ConnectedBoards.Val >> 5) & 0x01u)) | (TRG3_IN & (( ...
    ConnectedBoards.Val >> 6) & 0x01u)) | (TRH3_IN & ((ConnectedBoards.Val >> 7) & 0x01u) ...
    ) | (TRI3_IN & ((ConnectedBoards.Val >> 8) & 0x01u)) | (TRJ3_IN & ((ConnectedBoards.Val ...
    >> 9) & 0x01u)) | (TRL3_IN & ((ConnectedBoards.Val >> 11) & 0x01u)) | (TRM3_IN & (( ...
    ConnectedBoards.Val >> 12) & 0x01u)) | (TRN3_IN & ((ConnectedBoards.Val >> 13) & 0 ...
    x01u)) | (TRO3_IN & ((ConnectedBoards.Val >> 14) & 0x01u)) | (TRP3_IN & (( ...
    ConnectedBoards.Val >> 15) & 0x01u));
IncreaseSpeedIndex = 0x00u | (TRB3_IN & ((ConnectedBoards.Val >> 1) & 0x01u)) | (TRC3_IN ...
    & ((ConnectedBoards.Val >> 2) & 0x01u)) | (TRD3_IN & ((ConnectedBoards.Val >> 3) & 0 ...
    x01u)) | (TRE3_IN & ((ConnectedBoards.Val >> 4) & 0x01u)) | (TRF3_IN & (( ...
    ConnectedBoards.Val >> 5) & 0x01u)) | (TRI3_IN & ((ConnectedBoards.Val >> 8) & 0x01u) ...
    ) | (TRJ3_IN & ((ConnectedBoards.Val >> 9) & 0x01u)) | (TRN3_IN & ((ConnectedBoards. ...
    Val >> 13) & 0x01u));

MPFD_Read_Done();

// Let Daughter Boards write
TW_TRIS = 0;

MPFD_Write_Done();
MPFD_Read_Done();

if (!TRIGGER_IN_BOOL) {
    for (c = 0u; c < 15u; ++c) {
        BoardID[c] = ((BoardID[c] >> 4) & 0x0Fu) | (BoardID[c] & 0xF0u);
    }
    MPFD_Data_Ready = FALSE;
} else
    MPFD_Data_Ready = TRUE;
MPFD_Write_Done();

DataTask = MPFD3D_DAUGHTERS_TRIGGER;
NextTimeStamp = ThisTimeStamp + TicksPerSample[SpeedIndex];
if (SHUTDOWN_MODE_SELECTED) {
    SpeedIndex = 10u;
} else if (SpeedIndexLast == SpeedIndex) {
    DecreaseSpeedIndex = 1u;
    for (c = 0u; c < 15u; ++c) {
        if (c >= 0x0Au) {
            if ((((ConnectedBoards.Val >> (c + 1u)) & 0x0001u) == 0x0000u) { // Board ...
                Connected
                if ((MPFD_Data_Array[23u + (87u * c)] & 0x80u) == 0x80u) { // String ...
                    Connected
                    if ((MPFD_Data_Array[22u + (87u * c)] & 0x20u) == 0x20u) { // Overflow
                        IncreaseSpeedIndex |= 1u;
                    }
                    if ((MPFD_Data_Array[22u + (87u * c)] & 0x10u) != 0x10u) { // Underflow
                        DecreaseSpeedIndex &= 0u;
                    }
                }
```

```
                }
            } else {
                if (((ConnectedBoards.Val >> c) & 0x0001u) == 0x0000u) { // Board Connected
                    if ((MPFD_Data_Array[23u + (87u * c)] & 0x80u) == 0x80u) { // String ...
                        Connected
                        if ((MPFD_Data_Array[22u + (87u * c)] & 0x20u) == 0x20u) { // Overflow
                            IncreaseSpeedIndex |= 1u;
                        }
                        if ((MPFD_Data_Array[22u + (87u * c)] & 0x10u) != 0x10u) { // Underflow
                            DecreaseSpeedIndex &= 0u;
                        }
                    }
                }
            }
        }
        if (IncreaseSpeedIndex >= 1u) {
            if (SpeedIndex < 10u)
                SpeedIndexNew = SpeedIndex + 1u;
        } else if ((DecreaseSpeedIndex == 1u) && (((~ConnectedBoards.Val) & 0xFBFFu) != 0xFBFFu ...
            )) {
            if (SpeedIndex > 0u)
                SpeedIndexNew = SpeedIndex - 1u;
        }
    }
    MPFD_Read_Done();
    while (SyncTimer() && (TRIGGER_IN_BOOL));
    TRIGGER_TRIS = 0;
    READ_TRIS = 0;
    return;
}

void mpfd3d_settings_next_refresh(void) {
    MPFD_REFRESH_SETTINGS = TRUE;
    return;
}
```

End mpfd3d.c

B.7.11 temp.h

This is the header file for the program temp.c (Section §B.7.12) of the mother board's microcontroller.

Begin temp.h

```
#ifndef __TEMP_H
#define __TEMP_H

#include "GenericTypeDefs.h"
```

```
void InitTemp(void);
BYTE ReadTemp(void);
void TempTypeC(void);
void TempTypeF(void);

#endif
```

End temp.h

B.7.12 temp.c

This is the temp program for the mother board's microcontroller.

Begin temp.c

```
#include "temp.h"
#include "config.h"
#include "i2c.h"
#include "SNTP.h"
#include "lcd.h"

static QWORD NextTempUpdateTimeStamp = 0;
BYTE LastTempValue = 0u;
BYTE LCDTempValue = 0u;

void InitTemp(void) {
    EEByteWrite1(TEMP_I2C_ADDR,0x01,0x00);
}

BYTE ReadTemp(void) {
    QWORD CurrTimeStamp;
    CurrTimeStamp = GetUTCTimeStamp();
    if (CurrTimeStamp > NextTempUpdateTimeStamp) {
        NextTempUpdateTimeStamp = CurrTimeStamp + 0x1FFFull;
        LastTempValue = EERandomRead1(TEMP_I2C_ADDR,0x00);
        if (LCDTempValue != LastTempValue) {
            if (GetLCDScreen() == LCD_TEMPERATURE)
                LCDForceUpdate();
            LCDTempValue = LastTempValue;
        }
    }

    return LastTempValue;
}

void TempTypeC(void) {
    BYTE temp;
    temp = ReadTemp();
    if (temp > 100u)
```

```
        lcdType(temp / 100u + 0x30u);
    if (temp > 10u)
        lcdType((temp / 10u) % 10u + 0x30u);
    lcdType(temp % 10u + 0x30u);
}

void TempTypeF(void) {
    BYTE temp;
    temp = (9u*ReadTemp());
    temp = temp/5u;
    temp = temp + 32u;
    if (temp > 100u)
        lcdType(temp / 100u + 0x30u);
    if (temp > 10u)
        lcdType((temp / 10u) % 10u + 0x30u);
    lcdType(temp % 10u + 0x30u);
}
```

End temp.c

B.7.13 TCPIP Stack\FTP.c and TCPIP Stack\FTP.h

FTP.c and FTP.h contain a program module for the file transfer protocol (FTP) which was modified from the original Microchip Technologies, Inc. TCPIP Stack to include passive FTP (PASV) capabilities and to work with the MPFD^3-T 3D System User Interface Application.

B.7.14 TCPIP Stack\Helpers.c and TCPIP Stack\Helpers.h

Helpers.c and Helpers.h contain a set of helper functions modified from the original Microchip Technologies, Inc. TCPIP Stack to include the additional helper functions shexatob (separate hex characters to a single byte) and uctoa (unsigned char to a decimal string).

B.7.15 TCPIP Stack\HTTP.c and TCPIP Stack\HTTP.h

HTTP.c and HTTP.h contain a set of modules for serving HTTP web pages. These programs have been modified from the original Microchip Technologies, Inc. TCPIP Stack to include identifiers for ZIP and EXE type files.

B.7.16 TCPIP Stack\I2CEEPROM.c

I2CEEPROM.c is the driver for communication with the 24FC1025 I^2C EEPROM (Section §5.2.1) for the mother board's microcontroller. This driver is based on the SPIEEPROM.c driver from Microchip Technologies, Inc. TCPIP Stack for communication with a similar series SPI based EEPROM, the 24LC1025.

B.7.17 TCPIP Stack\MPFS.c and TCPIP Stack\MPFS.h

MPFS.c and MPFS.h control the file system (Microchip File System) of the 24FC1025 I^2C EEPROM (Section §5.2.1) for the mother board's microcontroller. This has been modified from the original Microchip Technologies, Inc. TCPIP Stack to include communication options for the 24FC1025.

B.7.18 TCPIP Stack\TCPIP.h

TCPIP.h controls which modules of the Microchip Technologies, Inc. TCPIP Stack are compiled. This header file has been modified from the original file to include communication options for the 24FC1025.

B.7.19 TCPIP Stack\SNTP.c and TCPIP Stack\SNTP.h

SNTP.c and SNTP.h are for the simple network time protocol (SNTP) used for syncronizing the real-time clock on the mother board via the microcontroller. This is modified from the original Microchip Technologies, Inc. TCPIP Stack to include the required functions to work with the real-time clock used in this system and for optimum performance with the rest of the microcontroller program.

B.7.20 TCPIP Stack\Tick.c and TCPIP Stack\Tick.h

Tick.c and Tick.h controls the timing operations of the mother board's microcontroller. This is modified from the original Microchip Technologies, Inc. TCPIP Stack to include

the required functions to work with the real-time clock used in this system and for optimum performance with the rest of the microcontroller program.

B.8 User Interface Application

Visual Basic applications do not lend themselves well for publishing on paper so the code for this application is not included here. In addition, the code contains sensitive material which cannot be freely distributed. A full copy of the program may be requested from the author via email at martin@ohmes.com.

B.8.1 WriteCalibrationCoefficientsFile.m

This MATLAB script generates the Calibration_Coefficients.ccf file necessary for modifying the calibration coefficients (Figure 6.34) used by the User Interface Application.

Begin WriteCalibrationCoefficientsFile.m

```
filepath = 'C:\MPFD Programs\MPFD3-T 3D\MPFD3-T 3D\ ...
    Calibration_Coefficients.ccf';

%% Initialize Variables
HVBoardScale = ones(15,1);
HVBoardOffset = zeros(15,1);

HPBoardScale = ones(15,4);
HPBoardOffset = zeros(15,4);
HPStringScale = ones(15,4);
HPStringOffset = zeros(15,4);
LPBoardScale = ones(15,4);
LPBoardOffset = zeros(15,4);
LPStringScale = ones(15,4);
LPStringOffset = zeros(15,4);
BPBoardScale = ones(15,4);
BPBoardOffset = zeros(15,4);
BPStringScale = ones(15,4);
BPStringOffset = zeros(15,4);

HMBoardScale = ones(15,4);
HMBoardOffset = zeros(15,4);
HMStringScale = ones(15,4);
HMStringOffset = zeros(15,4);
LMBoardScale = ones(15,4);
LMBoardOffset = zeros(15,4);
LMStringScale = ones(15,4);
```

```
LMStringOffset = zeros(15,4);
BMBoardScale = ones(15,4);
BMBoardOffset = zeros(15,4);
BMStringScale = ones(15,4);
BMStringOffset = zeros(15,4);

HCBoardScale = ones(15,4);
HCBoardOffset = zeros(15,4);
HCStringScale = ones(15,4);
HCStringOffset = zeros(15,4);
LCBoardScale = ones(15,4);
LCBoardOffset = zeros(15,4);
LCStringScale = ones(15,4);
LCStringOffset = zeros(15,4);
BCBoardScale = ones(15,4);
BCBoardOffset = zeros(15,4);
BCStringScale = ones(15,4);
BCStringOffset = zeros(15,4);

TBoardScale = ones(15,4);
TBoardOffset = zeros(15,4);
TStringScale = ones(15,4);
TStringOffset = zeros(15,4);

%% Board A
i = 1;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
```

```
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
HPBoardScale(i,j) = 2;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
```

```
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
HPBoardScale(i,j) = 3;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
```

```
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
HPBoardScale(i,j) = 4;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board B
i = 2;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
```

```
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
```

```
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
```

```
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board C
i = 3;
HVBoardScale(i,1) = 0.0253;
HVBoardOffset(i,1) = −172.51;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
```

```
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
```

```
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
```

```
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;
```

```
% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board D
i = 4;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
```

```
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
```

```
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board E
```

```
i = 5;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
```

```
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
```

```
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;
```

```
% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board F
i = 6;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
```

```
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
```

```
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;
```

```
% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
```

```
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board G
i = 7;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
```

```
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
```

```
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board H
i = 8;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
```

```
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
```

```
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;
```

```
% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
```

```
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board I
i = 9;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
```

```
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;
```

```
% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
```

```
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board J
i = 10;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
```

```
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
```

```
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board L
i = 11;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
```

```
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;
```

```
% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
```

```
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;
```

```
% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board M
i = 12;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;
```

```
% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
```

```
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;
```

```
%% Board N
i = 13;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
```

```
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
```

```
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;
```

```
% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board O
i = 14;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;
```

```
% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
```

```
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;
```

```
% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
```

```
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Board P
i = 15;
HVBoardScale(i,1) = 0.0125;
HVBoardOffset(i,1) = 412;

j = 1;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
```

```
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 2;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;
```

```
% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

j = 3;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;
```

```
j = 4;
% HPBoardScale(i,j) = 1;
% HPBoardOffset(i,j) = 0;
% HPStringScale(i,j) = 1;
% HPStringOffset(i,j) = 0;
% LPBoardScale(i,j) = 1;
% LPBoardOffset(i,j) = 0;
% LPStringScale(i,j) = 1;
% LPStringOffset(i,j) = 0;
% BPBoardScale(i,j) = 1;
% BPBoardOffset(i,j) = 0;
% BPStringScale(i,j) = 1;
% BPStringOffset(i,j) = 0;

% HMBoardScale(i,j) = 1;
% HMBoardOffset(i,j) = 0;
% HMStringScale(i,j) = 1;
% HMStringOffset(i,j) = 0;
% LMBoardScale(i,j) = 1;
% LMBoardOffset(i,j) = 0;
% LMStringScale(i,j) = 1;
% LMStringOffset(i,j) = 0;
% BMBoardScale(i,j) = 1;
% BMBoardOffset(i,j) = 0;
% BMStringScale(i,j) = 1;
% BMStringOffset(i,j) = 0;

% HCBoardScale(i,j) = 1;
% HCBoardOffset(i,j) = 0;
% HCStringScale(i,j) = 1;
% HCStringOffset(i,j) = 0;
% LCBoardScale(i,j) = 1;
% LCBoardOffset(i,j) = 0;
% LCStringScale(i,j) = 1;
% LCStringOffset(i,j) = 0;
% BCBoardScale(i,j) = 1;
% BCBoardOffset(i,j) = 0;
% BCStringScale(i,j) = 1;
% BCStringOffset(i,j) = 0;

% TBoardScale(i,j) = 1;
% TBoardOffset(i,j) = 0;
% TStringScale(i,j) = 1;
% TStringOffset(i,j) = 0;

%% Write the values to the file
fid = fopen(filepath,'w','n','US-ASCII');

for i = 1:15
    fprintf(fid,'%e\r\n',HVBoardScale(i,1));
    fprintf(fid,'%e\r\n',HVBoardOffset(i,1));
```

```
    for j = 4:-1:1
        fprintf(fid,'%e\r\n',HPBoardScale(i,j));
        fprintf(fid,'%e\r\n',HPBoardOffset(i,j));
        fprintf(fid,'%e\r\n',HPStringScale(i,j));
        fprintf(fid,'%e\r\n',HPStringOffset(i,j));
        fprintf(fid,'%e\r\n',LPBoardScale(i,j));
        fprintf(fid,'%e\r\n',LPBoardOffset(i,j));
        fprintf(fid,'%e\r\n',LPStringScale(i,j));
        fprintf(fid,'%e\r\n',LPStringOffset(i,j));
        fprintf(fid,'%e\r\n',BPBoardScale(i,j));
        fprintf(fid,'%e\r\n',BPBoardOffset(i,j));
        fprintf(fid,'%e\r\n',BPStringScale(i,j));
        fprintf(fid,'%e\r\n',BPStringOffset(i,j));

        fprintf(fid,'%e\r\n',HMBoardScale(i,j));
        fprintf(fid,'%e\r\n',HMBoardOffset(i,j));
        fprintf(fid,'%e\r\n',HMStringScale(i,j));
        fprintf(fid,'%e\r\n',HMStringOffset(i,j));
        fprintf(fid,'%e\r\n',LMBoardScale(i,j));
        fprintf(fid,'%e\r\n',LMBoardOffset(i,j));
        fprintf(fid,'%e\r\n',LMStringScale(i,j));
        fprintf(fid,'%e\r\n',LMStringOffset(i,j));
        fprintf(fid,'%e\r\n',BMBoardScale(i,j));
        fprintf(fid,'%e\r\n',BMBoardOffset(i,j));
        fprintf(fid,'%e\r\n',BMStringScale(i,j));
        fprintf(fid,'%e\r\n',BMStringOffset(i,j));

        fprintf(fid,'%e\r\n',HCBoardScale(i,j));
        fprintf(fid,'%e\r\n',HCBoardOffset(i,j));
        fprintf(fid,'%e\r\n',HCStringScale(i,j));
        fprintf(fid,'%e\r\n',HCStringOffset(i,j));
        fprintf(fid,'%e\r\n',LCBoardScale(i,j));
        fprintf(fid,'%e\r\n',LCBoardOffset(i,j));
        fprintf(fid,'%e\r\n',LCStringScale(i,j));
        fprintf(fid,'%e\r\n',LCStringOffset(i,j));
        fprintf(fid,'%e\r\n',BCBoardScale(i,j));
        fprintf(fid,'%e\r\n',BCBoardOffset(i,j));
        fprintf(fid,'%e\r\n',BCStringScale(i,j));
        fprintf(fid,'%e\r\n',BCStringOffset(i,j));

        fprintf(fid,'%e\r\n',TBoardScale(i,j));
        fprintf(fid,'%e\r\n',TBoardOffset(i,j));
        fprintf(fid,'%e\r\n',TStringScale(i,j));
        fprintf(fid,'%e\r\n',TStringOffset(i,j));
    end
end
fclose(fid);
disp('Done writing file');
```

End WriteCalibrationCoefficientsFile.m

B.9 Data Analysis

B.9.1 LoadMPFDConverted.m

This MATLAB script converts the ".cal" file into a MATLAB data structure for further analysis.

Begin LoadMPFDConverted.m

```
function data = LoadMPFDConverted(directory,file)

cindex = strfind(file,'_');
filepath = fullfile(directory,file(1,1:cindex(1,1)-1),file(1,cindex(1,1)+1:cindex(1,2)-1),file(1,cindex(1,2) ...
    +1:cindex(1,3)-1),file);
if exist(filepath,'file') == 0
    filepath = fullfile(directory,file);
    if exist(filepath,'file') == 0
        error(['File path "',filepath,'" does not exist']);
    end
end
fid = fopen(filepath);
Boards = {'BoardA','BoardB','BoardC','BoardD','BoardE','BoardF','BoardG',' ...
    BoardH','BoardI','BoardJ','BoardL','BoardM','BoardN','BoardO','BoardP ...
    '};
index = 0;
clear data;
while 1
    tline = fgetl(fid);
    if ~ischar(tline) % ? End of file reached
        break
    end
    token = strtok(tline,',');
    switch token
        case 'Reactor Power'
            [token,data.ReactorPower.Value,data.ReactorPower.Range] = strread(tline,'%s %f %s ...
                ','delimiter',',');
            RangeMultiplier = [0.001,1,1000,1000000];
            RangeString = {'mW','W','kW','MW'};
            data.ReactorPower.Watts = data.ReactorPower.Value*RangeMultiplier(1,strmatch(data. ...
                ReactorPower.Range,RangeString,'exact'));
        case 'Transient Rod Position'
            [token,data.TransientRodPosition] = strread(tline,'%s %f','delimiter',',');
        case 'Safety Rod Position'
            [token,data.SafetyRodPosition] = strread(tline,'%s %f','delimiter',',');
        case 'Shim Rod Position'
            [token,data.ShimRodPosition] = strread(tline,'%s %f','delimiter',',');
        case 'Regulating Rod Position'
            [token,data.RegulatingRodPosition] = strread(tline,'%s %f','delimiter',',');
        case 'Reactor Pool Temperature'
            [token,data.ReactorPoolTemperature.degC,extra] = strread(tline,'%s %f %s',' ...
```

```
        delimiter',',');
case 'Pool Inlet Temperature'
    [token,data.PoolInletTemperature.degC,extra] = strread(tline,'%s %f %s',' ...
        delimiter',',');
case 'Pool Outlet Temperature'
    [token,data.PoolOutletTemperature.degC,extra] = strread(tline,'%s %f %s',' ...
        delimiter',',');
case 'Bulk Water Temperature'
    [token,data.BulkWaterTemperature.degC,extra] = strread(tline,'%s %f %s',' ...
        delimiter',',');
case 'Fuel Temp #1'
    [token,data.FuelTemp1Temperature.degC,extra] = strread(tline,'%s %f %s',' ...
        delimiter',',');
case 'Fuel Temp #2'
    [token,data.FuelTemp2Temperature.degC,extra] = strread(tline,'%s %f %s',' ...
        delimiter',',');
case 'Fuel Temp #3'
    [token,data.FuelTemp3Temperature.degC,extra] = strread(tline,'%s %f %s',' ...
        delimiter',',');
case 'Board A'
    [token,data.BoardA.String,data.BoardA.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board B'
    [token,data.BoardB.String,data.BoardB.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board C'
    [token,data.BoardC.String,data.BoardC.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board D'
    [token,data.BoardD.String,data.BoardD.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board E'
    [token,data.BoardE.String,data.BoardE.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board F'
    [token,data.BoardF.String,data.BoardF.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board G'
    [token,data.BoardG.String,data.BoardG.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board H'
    [token,data.BoardH.String,data.BoardH.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board I'
    [token,data.BoardI.String,data.BoardI.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board J'
    [token,data.BoardJ.String,data.BoardJ.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board L'
    [token,data.BoardL.String,data.BoardL.Box] = strread(tline,'%s %s %s',' ...
```

```
        delimiter',',');
case 'Board M'
    [token,data.BoardM.String,data.BoardM.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board N'
    [token,data.BoardN.String,data.BoardN.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board O'
    [token,data.BoardO.String,data.BoardO.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Board P'
    [token,data.BoardP.String,data.BoardP.Box] = strread(tline,'%s %s %s',' ...
        delimiter',',');
case 'Start Time bits'
    %disp(tline)
case '1/65536 Seconds'
    %disp(tline)
otherwise
    index = index + 1;
    c = 1;
    %disp(tline)
    cindex = strfind(tline,',');
    if isempty(cindex)
        data.RawData(index,:) = tline;
        index = index - 1;
    else
        ei = [cindex - 1 size(tline,1)];
        si = [1 cindex + 1];
        data.StartTime(index).bits = uint64(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
        data.StartTime(index).datenum = datenum(1970,1,1)+double(data.StartTime(index). ...
            bits)/(65536*86400);
        data.StartTime(index).datestr = [datestr(data.StartTime(index).datenum,'mmm dd, ...
             yyyy HH:MM:SS'),'.',num2str(round(rem(data.StartTime(index). ...
            datenum*86400,1)*1000),'%03i')];
        data.EndTime(index).bits = uint64(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
        data.EndTime(index).datenum = datenum(1970,1,1)+double(data.EndTime(index).bits ...
            )/(65536*86400);
        data.EndTime(index).datestr = [datestr(data.EndTime(index).datenum,'mmm dd,  ...
            yyyy HH:MM:SS'),'.',num2str(round(rem(data.EndTime(index).datenum ...
            *86400,1)*1000),'%03i')];
        data.ReactorOn(index) = logical(strmatch(tline(1,si(1,c):ei(1,c)),{'False','True ...
            '})-1); c = c + 1;
        data.ReactorPulse(index) = logical(strmatch(tline(1,si(1,c):ei(1,c)),{'False',' ...
            True'})-1); c = c + 1;
        data.ACGood(index) = logical(strmatch(tline(1,si(1,c):ei(1,c)),{'False','True'}) ...
            -1); c = c + 1;
        data.SampleRate(index).bits = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
        data.SampleRate(index).SPS = single(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
        for i = 1:15
            data.(Boards{1,i})(index).Present = logical(strmatch(tline(1,si(1,c):ei(1,c)),{' ...
                False','True'})-1); c = c + 1;
```

```
end
data.HVSetpoint(index).bits = uint16(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
data.HVSetpoint(index).value = str2double(tline(1,si(1,c):ei(1,c))); c = c + 1;
data.LLDSetpoint(index).bits = uint16(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
data.LLDSetpoint(index).value = str2double(tline(1,si(1,c):ei(1,c))); c = c + 1;
data.LLDSetpoint(index).value = 2500.0 * double(data.LLDSetpoint(index).bits) / ...
    1023.0;
data.ADCSpeed(index) = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
data.MSVResolution(index) = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
data.CurrentResolution(index) = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c = c + 1;
data.TemperatureResolution(index) = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c = c + ...
    1;
data.MotherboardTemperature(index) = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c = c ...
     + 1;
for i = 1:15
    data.(Boards{1,i})(index).PulseOverflow = logical(strmatch(tline(1,si(1,c):ei(1,c)) ...
        ,{'False','True'})-1); c = c + 1;
    data.(Boards{1,i})(index).PulseUnderflow = logical(strmatch(tline(1,si(1,c):ei(1,c) ...
        ),{'False','True'})-1); c = c + 1;
    data.(Boards{1,i})(index).Temperature.Resolution = uint8(str2double(tline(1,si(1,c ...
        ):ei(1,c)))); c = c + 1;
    data.(Boards{1,i})(index).HV.Speed = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c ...
        = c + 1;
    data.(Boards{1,i})(index).StringConnected = logical(strmatch(tline(1,si(1,c):ei(1,c ...
        )),{'False','True'})-1); c = c + 1;
    data.(Boards{1,i})(index).ADCSpeed = uint8(str2double(tline(1,si(1,c):ei(1,c)))); c ...
         = c + 1;
    data.(Boards{1,i})(index).HV.New = logical(strmatch(tline(1,si(1,c):ei(1,c)),{' ...
        False','True'})-1); c = c + 1;
    for j = 1:4
        data.(Boards{1,i})(index).Temperature.New(j) = logical(strmatch(tline(1,si(1, ...
            c):ei(1,c)),{'False','True'})-1); c = c + 1;
        data.(Boards{1,i})(index).MSV.L.New(j) = logical(strmatch(tline(1,si(1,c):ei ...
            (1,c)),{'False','True'})-1); c = c + 1;
        data.(Boards{1,i})(index).MSV.B.New(j) = logical(strmatch(tline(1,si(1,c):ei ...
            (1,c)),{'False','True'})-1); c = c + 1;
        data.(Boards{1,i})(index).MSV.H.New(j) = logical(strmatch(tline(1,si(1,c):ei ...
            (1,c)),{'False','True'})-1); c = c + 1;
        data.(Boards{1,i})(index).Current.L.New(j) = logical(strmatch(tline(1,si(1,c): ...
            ei(1,c)),{'False','True'})-1); c = c + 1;
        data.(Boards{1,i})(index).Current.B.New(j) = logical(strmatch(tline(1,si(1,c): ...
            ei(1,c)),{'False','True'})-1); c = c + 1;
        data.(Boards{1,i})(index).Current.H.New(j) = logical(strmatch(tline(1,si(1,c): ...
            ei(1,c)),{'False','True'})-1); c = c + 1;
    end
    data.(Boards{1,i})(index).HV.bits = int16(str2double(tline(1,si(1,c):ei(1,c)))); c = ...
         c + 1;
    data.(Boards{1,i})(index).HV.voltage = str2double(tline(1,si(1,c):ei(1,c))); c = c + ...
        1;
    for j = 1:4
        data.(Boards{1,i})(index).Temperature.bits(j) = int16(str2double(tline(1,si(1,c ...
```

```
                    ):ei(1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Temperature.mV(j) = str2double(tline(1,si(1,c):ei(1,  ...
                    c))); c = c + 1;
                data.(Boards{1,i})(index).Temperature.degC(j) = str2double(tline(1,si(1,c):ei  ...
                    (1,c))); c = c + 1;
                data.(Boards{1,i})(index).Temperature.degF(j) = str2double(tline(1,si(1,c):ei  ...
                    (1,c))); c = c + 1;
                data.(Boards{1,i})(index).Pulse.L.bits(j) = uint16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Pulse.L.cps(j) = str2double(tline(1,si(1,c):ei(1,c)));   ...
                    c = c + 1;
                data.(Boards{1,i})(index).Pulse.B.bits(j) = uint16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Pulse.B.cps(j) = str2double(tline(1,si(1,c):ei(1,c)));   ...
                    c = c + 1;
                data.(Boards{1,i})(index).Pulse.H.bits(j) = uint16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Pulse.H.cps(j) = str2double(tline(1,si(1,c):ei(1,c)));   ...
                    c = c + 1;
                data.(Boards{1,i})(index).MSV.L.bits(j) = int16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).MSV.L.cps(j) = str2double(tline(1,si(1,c):ei(1,c)));   ...
                    c = c + 1;
                data.(Boards{1,i})(index).MSV.B.bits(j) = int16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).MSV.B.cps(j) = str2double(tline(1,si(1,c):ei(1,c)));   ...
                    c = c + 1;
                data.(Boards{1,i})(index).MSV.H.bits(j) = int16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).MSV.H.cps(j) = str2double(tline(1,si(1,c):ei(1,c)));   ...
                    c = c + 1;
                data.(Boards{1,i})(index).Current.L.bits(j) = int16(str2double(tline(1,si(1,c):ei  ...
                    (1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Current.L.cps(j) = str2double(tline(1,si(1,c):ei(1,c)))  ...
                    ; c = c + 1;
                data.(Boards{1,i})(index).Current.B.bits(j) = int16(str2double(tline(1,si(1,c):  ...
                    ei(1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Current.B.cps(j) = str2double(tline(1,si(1,c):ei(1,c))  ...
                    ); c = c + 1;
                data.(Boards{1,i})(index).Current.H.bits(j) = int16(str2double(tline(1,si(1,c):  ...
                    ei(1,c)))); c = c + 1;
                data.(Boards{1,i})(index).Current.H.cps(j) = str2double(tline(1,si(1,c):ei(1,c))  ...
                    ); c = c + 1;
            end
        end
    end
    % End otherwise
  end
end
fclose(fid);
```

End LoadMPFDConverted.m

B.9.2 ConvertAllFiles.m

This MATLAB script cycles through all ".cal" files in the "Converted" directory structure checking for matching "_cal.mat" and processing remaining files through LoadMPFDConverted.m (Section §B.9.1) for conversion.

Begin ConvertAllFiles.m

```
function [] = ConvertAllFiles(BaseDirectory,RerunAll)

dbstop if error;
FindBaseDirectory = false;
if nargin <= 1
    if nargin == 0
        FindBaseDirectory = true;
        RerunAll = false;
    elseif islogical(BaseDirectory) || isnumeric(BaseDirectory)
        RerunAll = BaseDirectory;
        FindBaseDirectory = true;
    end
    if FindBaseDirectory == true
        BaseDirectory = 'F:\MPFD3T3D';
        if exist(BaseDirectory,'dir') == 0
            BaseDirectory = 'C:\MPFD3T3D';
        end
    end
end

if exist(BaseDirectory,'dir') == 0
    error('BaseDirectory could not be found - rerun and define an  ...
        existing BaseDirectory');
end

Directories = cellstr(BaseDirectory);
DirIndex = 1;
while DirIndex <= size(Directories,1)
    listing = dir(Directories{DirIndex,1});
    NewDirectories = {listing([listing(:).isdir] & ~strcmp('.',{listing(:).name}) & ~strcmp('..',{ ...
        listing(:).name}) & ~strcmp('.hg',{listing(:).name})).name}';
    if ~isempty(NewDirectories)
        Directories = [Directories;strcat(Directories{DirIndex,1},'\',NewDirectories)]; %#ok< ...
            AGROW>
    end
    MatFiles = {listing(~cellfun('isempty',strfind({listing(:).name},'.mat'))').name}';
    CalFiles = {listing(~cellfun('isempty',strfind({listing(:).name},'.cal'))').name}';
    CsvFiles = {listing(~cellfun('isempty',strfind({listing(:).name},'.csv'))').name}';
    if ~isempty(CalFiles)
```

```
    for i = 1:size(CalFiles,1)
        if ~isempty(MatFiles) && RerunAll == false
            if any(strcmp(strrep(CalFiles{i,1},'.cal','_cal.mat'),MatFiles))
                disp([Directories{DirIndex,1},'\',CalFiles{i,1},' has already been ...
                    converted.']);
            else
                fprintf('%s',['Converting ',Directories{DirIndex,1},'\',CalFiles{i,1}, ...
                    ' ...']);
                data = LoadMPFDConverted(Directories{DirIndex,1},CalFiles{i,1}); %#ok< ...
                    NASGU>
                save([Directories{DirIndex,1},'\',strrep(CalFiles{i,1},'.cal','_cal.mat ...
                    ')],'-mat','data');
                clear data;
                fprintf('%s\n',' done.');
            end
        else
            fprintf('%s',['Converting ',Directories{DirIndex,1},'\',CalFiles{i,1},' ...
                ...']);
            data = LoadMPFDConverted(Directories{DirIndex,1},CalFiles{i,1}); %#ok<NASGU>
            save([Directories{DirIndex,1},'\',strrep(CalFiles{i,1},'.cal','_cal.mat')], ...
                '-mat','data');
            clear data;
            fprintf('%s\n',' done.');
        end
    end
end
if ~isempty(CsvFiles)
    for i = 1:size(CsvFiles,1)
        if ~isempty(MatFiles) && RerunAll == false
            if any(strcmp(strrep(CsvFiles{i,1},'.csv','_csv.mat'),MatFiles))
                disp([Directories{DirIndex,1},'\',CsvFiles{i,1},' has already been ...
                    converted.']);
            else
                fprintf('%s',['Converting ',Directories{DirIndex,1},'\',CsvFiles{i,1}, ...
                    ' ...']);
                data = LoadMPFDConverted(Directories{DirIndex,1},CsvFiles{i,1}); %#ok< ...
                    NASGU>
                save([Directories{DirIndex,1},'\',strrep(CsvFiles{i,1},'.csv','_csv. ...
                    mat')],'-mat','data');
                clear data;
                fprintf('%s\n',' done.');
            end
        else
            fprintf('%s',['Converting ',Directories{DirIndex,1},'\',CsvFiles{i,1},' ...
                ...']);
            data = LoadMPFDConverted(Directories{DirIndex,1},CsvFiles{i,1}); %#ok<NASGU ...
                >
            save([Directories{DirIndex,1},'\',strrep(CsvFiles{i,1},'.csv','_csv.mat')], ...
                '-mat','data');
            clear data;
            fprintf('%s\n',' done.');
```

```
            end
        end
    end
    DirIndex = DirIndex + 1;
end
```

End ConvertAllFiles.m

B.9.3 MPFDData.m

This MATLAB script pulls data from the converted data structure produced by LoadMPFD-Converted.m (Section §B.9.1).

Begin MPFDData.m

```
function [returnedData,legendData,returnedCode] = MPFDData(data,dataString)

if ~iscell(dataString)
    dataString = {dataString};
end
dataString = dataString(:);
temp = cell(0);
for i = 1:size(dataString,1);
    temp = [temp;regexp(dataString{i,1},'\.','split')']; %#ok<AGROW>
end
dataString = temp;
clear temp i;

returnedData = [];
legendData = [];
returnedCode = [];

dataStringIndex = dataString{1,1}(1,min([size(dataString{1,1},2)+1,strfind(dataString{1,1},'(')...
    dataString{1,1},'{')]):end);
dataStringCode = strrep(dataString{1,1},dataStringIndex,'');
switch dataStringCode
    case 'StartTime'
        if isfield(data,'StartTime')
            if size(dataString,1) >= 2
                switch dataString{2,1}
                    case {'datestr','string','String'}
                        if isfield(data.StartTime,'datestr')
                            returnedData = [data.StartTime.datestr];
                            legendData = {'Start Time (datestr)'};
                            returnedCode = {'StartTime.datestr'};
                        end
                    case {'bits'}
                        if isfield(data.StartTime,'bits')
                            returnedData = [data.StartTime.bits];
```

```
                    legendData = {'Start Time (bits)'};
                    returnedCode = {'StartTime.bits'};
                end
            case {'datenum','Value','value'}
                if isfield(data.StartTime,'datenum')
                    returnedData = [data.StartTime.datenum];
                    legendData = {'Start Time (datenum)'};
                    returnedCode = {'StartTime.datenum'};
                end
        end
    else
        if isfield(data.StartTime,'datenum')
            returnedData = [data.StartTime.datenum];
            legendData = {'Start Time (datenum)'};
            returnedCode = {'StartTime.datenum'};
        end
    end
end
case 'EndTime'
    if isfield(data,'EndTime')
        if size(dataString,1) >= 2
            switch dataString{2,1}
                case {'datestr','string','String'}
                    if isfield(data.EndTime,'datestr')
                        returnedData = [data.EndTime.datestr];
                        legendData = {'End Time (datestr)'};
                        returnedCode = {'EndTime.datestr'};
                    end
                case {'bits'}
                    if isfield(data.EndTime,'bits')
                        returnedData = [data.EndTime.bits];
                        legendData = {'End Time (bits)'};
                        returnedCode = {'EndTime.bits'};
                    end
                case {'datenum','Value','value'}
                    if isfield(data.EndTime,'datenum')
                        returnedData = [data.EndTime.datenum];
                        legendData = {'End Time (datenum)'};
                        returnedCode = {'EndTime.datenum'};
                    end
            end
        else
            if isfield(data.EndTime,'datenum')
                returnedData = [data.EndTime.datenum];
                legendData = {'End Time (datenum)'};
                returnedCode = {'EndTime.datenum'};
            end
        end
    end
case 'ReactorPower'
    if isfield(data,'ReactorPower')
```

```
        if size(dataString,1) >= 2
            switch dataString{2,1}
                case {'Watts','watts','watt','Watt'}
                    if isfield(data.ReactorPower,'Watts')
                        returnedData = [data.ReactorPower.Watts];
                        legendData = {'Reactor Power (Watts)'};
                        returnedCode = {'ReactorPower.Watts'};
                    end
                case {'Value','value'}
                    if isfield(data.ReactorPower,'Value')
                        returnedData = [data.ReactorPower.Value];
                        legendData = {'Reactor Power (Value)'};
                        returnedCode = {'ReactorPower.Value'};
                    end
                case {'Range','range','Units','units','Unit','unit'}
                    if isfield(data.ReactorPower,'Range')
                        returnedData = [data.ReactorPower.Range];
                        legendData = {'Reactor Power (Units)'};
                        returnedCode = {'ReactorPower.Range'};
                    end
                case {'ValueRange','valuerange','ValueUnits','valueunits', ...
                    'ValueUnit','valueunit'}
                    if isfield(data.ReactorPower,'Range') && isfield(data.ReactorPower,' ...
                        Value')
                        returnedData = [num2str(data.ReactorPower.Value),' ',data. ...
                            ReactorPower.Range];
                        legendData = {'Reactor Power'};
                        returnedCode = {'ReactorPower.ValueRange'};
                    end
            end
        else
            if isfield(data.ReactorPower,'Watts')
                returnedData = [data.ReactorPower.Watts];
                legendData = {'Reactor Power (Watts)'};
                returnedCode = {'ReactorPower.Watts'};
            end
        end
    end
case 'TransientRodPosition'
    if isfield(data,'TransientRodPosition')
        returnedData = data.TransientRodPosition;
        legendData = {'Transient Rod Position'};
        returnedCode = {'TransientRodPosition'};
    end
case 'SafetyRodPosition'
    if isfield(data,'SafetyRodPosition')
        returnedData = data.SafetyRodPosition;
        legendData = {'Safety Rod Position'};
        returnedCode = {'SafetyRodPosition'};
    end
case 'ShimRodPosition'
```

```
    if isfield(data,'ShimRodPosition')
        returnedData = data.ShimRodPosition;
        legendData = {'Shim Rod Position'};
        returnedCode = {'ShimRodPosition'};
    end
case 'RegulatingRodPosition'
    if isfield(data,'RegulatingRodPosition')
        returnedData = data.RegulatingRodPosition;
        legendData = {'Regulating Rod Position'};
        returnedCode = {'RegulatingRodPosition'};
    end
case 'PoolInletTemperature'
    if isfield(data,'PoolInletTemperature')
        returnedData = data.PoolInletTemperature.degC;
        legendData = {'Pool Inlet Temperature (\circC)'};
        returnedCode = {'PoolInletTemperature'};
    end
case 'PoolInletTemperatureF'
    if isfield(data,'PoolInletTemperature')
        returnedData = 9/5*data.PoolInletTemperature.degC + 32;
        legendData = {'Pool Inlet Temperature (\circF)'};
        returnedCode = {'PoolInletTemperatureF'};
    end
case 'PoolOutletTemperature'
    if isfield(data,'PoolOutletTemperature')
        returnedData = data.PoolOutletTemperature.degC;
        legendData = {'Pool Outlet Temperature (\circC)'};
        returnedCode = {'PoolOutletTemperature'};
    end
case 'PoolOutletTemperatureF'
    if isfield(data,'PoolOutletTemperature')
        returnedData = 9/5*data.PoolOutletTemperature.degC + 32;
        legendData = {'Pool Outlet Temperature (\circF)'};
        returnedCode = {'PoolOutletTemperatureF'};
    end
case 'BulkWaterTemperature'
    if isfield(data,'BulkWaterTemperature')
        returnedData = data.BulkWaterTemperature.degC;
        legendData = {'Bulk Water Temperature (\circC)'};
        returnedCode = {'BulkWaterTemperature'};
    end
case 'BulkWaterTemperatureF'
    if isfield(data,'BulkWaterTemperature')
        returnedData = 9/5*data.BulkWaterTemperature.degC + 32;
        legendData = {'Bulk Water Temperature (\circF)'};
        returnedCode = {'BulkWaterTemperatureF'};
    end
case 'FuelTemp1Temperature'
    if isfield(data,'FuelTemp1Temperature')
        returnedData = data.FuelTemp1Temperature.degC;
        legendData = {'Fuel Temp1 Temperature (\circC)'};
```

```
        returnedCode = {'FuelTemp1Temperature'};
    end
case 'FuelTemp1TemperatureF'
    if isfield(data,'FuelTemp1Temperature')
        returnedData = 9/5*data.FuelTemp1Temperature.degC + 32;
        legendData = {'Fuel Temp1 Temperature (\circF)'};
        returnedCode = {'FuelTemp1TemperatureF'};
    end
case 'FuelTemp2Temperature'
    if isfield(data,'FuelTemp2Temperature')
        returnedData = data.FuelTemp2Temperature.degC;
        legendData = {'Fuel Temp2 Temperature (\circC)'};
        returnedCode = {'FuelTemp2Temperature'};
    end
case 'FuelTemp2TemperatureF'
    if isfield(data,'FuelTemp2Temperature')
        returnedData = 9/5*data.FuelTemp2Temperature.degC + 32;
        legendData = {'Fuel Temp2 Temperature (\circF)'};
        returnedCode = {'FuelTemp2TemperatureF'};
    end
case 'FuelTemp3Temperature'
    if isfield(data,'FuelTemp3Temperature')
        returnedData = data.FuelTemp3Temperature.degC;
        legendData = {'Fuel Temp3 Temperature (\circC)'};
        returnedCode = {'FuelTemp3Temperature'};
    end
case 'FuelTemp3TemperatureF'
    if isfield(data,'FuelTemp3Temperature')
        returnedData = 9/5*data.FuelTemp3Temperature.degC + 32;
        legendData = {'Fuel Temp3 Temperature (\circF)'};
        returnedCode = {'FuelTemp3TemperatureF'};
    end
case 'ReactorOn'
    if isfield(data,'ReactorOn')
        returnedData = data.ReactorOn(1,:);
        legendData = {'Reactor On'};
        returnedCode = {'ReactorOn'};
    end
case 'ReactorPulse'
    if isfield(data,'ReactorPulse')
        returnedData = data.ReactorPulse(1,:);
        legendData = {'Reactor Pulse Mode'};
        returnedCode = {'ReactorPulse'};
    end
case 'ACGood'
    if isfield(data,'ACGood')
        returnedData = data.ACGood(1,:);
        legendData = {'AC Power Good'};
        returnedCode = {'ACGood'};
    end
case {'SampleRatebits'}
```

```
    if isfield(data,'SampleRate')
        returnedData = [data.SampleRate(1,:).bits];
        legendData = {'Sample Rate (bits)'};
        returnedCode = {'SampleRate.bits'};
    end
case {'SampleRate','SampleRateSPS'}
    if isfield(data,'SampleRate')
        if size(dataString,1) >= 2
            switch dataString{2,1}
                case {'bits','Bits'}
                    if isfield(data.('SampleRate'),'bits')
                        returnedData = [data.SampleRate(1,:).bits];
                        legendData = {'Sample Rate (bits)'};
                        returnedCode = {'SampleRate.bits'};
                    end
                case 'SPS'
                    if isfield(data.('SampleRate'),'SPS')
                        returnedData = [data.SampleRate(1,:).SPS];
                        legendData = {'Sample Rate (SPS)'};
                        returnedCode = {'SampleRate.SPS'};
                    end
            end
        else
            if isfield(data.('SampleRate'),'SPS')
                returnedData = [data.SampleRate(1,:).SPS];
                legendData = {'Sample Rate (SPS)'};
                returnedCode = {'SampleRate.SPS'};
            end
        end
    end
case {'HVSetpointbits'}
    if isfield(data,'HVSetpoint')
        returnedData = [data.HVSetpoint(1,:).bits];
        legendData = {'High Voltage Setpoint (bits)'};
        returnedCode = {'HVSetpoint.bits'};
    end
case {'HVSetpoint','HVSetpointvalue','HVSetpointV'}
    if isfield(data,'HVSetpoint')
        if size(dataString,1) >= 2
            switch dataString{2,1}
                case {'bits','Bits'}
                    if isfield(data.('HVSetpoint'),'bits')
                        returnedData = [data.HVSetpoint(1,:).bits];
                        legendData = {'High Voltage Setpoint (bits)'};
                        returnedCode = {'HVSetpoint.bits'};
                    end
                case 'value'
                    if isfield(data.('HVSetpoint'),'value')
                        returnedData = [data.HVSetpoint(1,:).value];
                        legendData = {'High Voltage Setpoint (V)'};
                        returnedCode = {'HVSetpoint.value'};
```

```
                end
            end
        else
            if isfield(data.('HVSetpoint'),'value')
                returnedData = [data.HVSetpoint(1,:).value];
                legendData = {'High Voltage Setpoint (V)'};
                returnedCode = {'LLDSetpoint.value'};
            end
        end
    end
case {'LLDSetpointbits'}
    if isfield(data,'LLDSetpoint')
        returnedData = [data.LLDSetpoint(1,:).bits];
        legendData = {'LLD Setpoint (bits)'};
        returnedCode = {'LLDSetpoint.bits'};
    end
case {'LLDSetpoint','LLDSetpointvalue','LLDSetpointV'}
    if isfield(data,'LLDSetpoint')
        if size(dataString,1) >= 2
            switch dataString{2,1}
                case {'bits','Bits'}
                    if isfield(data.('LLDSetpoint'),'bits')
                        returnedData = [data.LLDSetpoint(1,:).bits];
                        legendData = {'LLD Setpoint (bits)'};
                        returnedCode = {'LLDSetpoint.bits'};
                    end
                case 'value'
                    if isfield(data.('LLDSetpoint'),'value')
                        returnedData = [data.LLDSetpoint(1,:).value];
                        legendData = {'LLD Setpoint (V)'};
                        returnedCode = {'LLDSetpoint.value'};
                    end
            end
        else
            if isfield(data.('LLDSetpoint'),'value')
                returnedData = [data.LLDSetpoint(1,:).value];
                legendData = {'LLD Setpoint (V)'};
                returnedCode = {'LLDSetpoint.value'};
            end
        end
    end
case {'ADCSpeed'}
    if isfield(data,'ADCSpeed')
        returnedData = data.ADCSpeed(1,:);
        legendData = {'ADC Speed (bits)'};
        returnedCode = {'ADCSpeed'};
    end
case {'MSVResolution'}
    if isfield(data,'MSVResolution')
        returnedData = data.MSVResolution(1,:);
        legendData = {'MSV Mode ADC Resolution (bits)'};
```

```
            returnedCode = {'MSVResolution'};
        end
    case {'CurrentResolution'}
        if isfield(data,'CurrentResolution')
            returnedData = data.CurrentResolution(1,:);
            legendData = {'Current Mode ADC Resolution (bits)'};
            returnedCode = {'CurrentResolution'};
        end
    case {'TemperatureResolution'}
        if isfield(data,'TemperatureResolution')
            returnedData = data.TemperatureResolution(1,:);
            legendData = {'Temperature ADC Resolution (bits)'};
            returnedCode = {'TemperatureResolution'};
        end
    case {'MotherboardTemperature'}
        if isfield(data,'MotherboardTemperature')
            returnedData = data.MotherboardTemperature(1,:);
            legendData = {'Motherboard Temperature (\circC)'};
            returnedCode = {'MotherboardTemperature'};
        end
    case {'MotherboardTemperatureF'}
        if isfield(data,'MotherboardTemperature')
            returnedData = 9./5.*data.MotherboardTemperature(1,:)+32;
            legendData = {'Motherboard Temperature (\circF)'};
            returnedCode = {'MotherboardTemperatureF'};
        end
    case {'BoardA','BoardB','BoardC','BoardD','BoardE','BoardF','BoardG',' ...
        BoardH','BoardI','BoardJ','BoardL','BoardM','BoardN','BoardO',' ...
        BoardP'}
        [returnedData,legendData,returnedCode] = MPFDBoardData(data,dataString);
        for i = 1:size(returnedCode,1)
            returnedCode{i,1} = [dataString{1,1},'.',returnedCode{i,1}];
            legendData{i,1} = [dataString{1,1},' ',legendData{i,1}];
        end
    case {'StringA','StringB','StringC','StringD','StringE','StringF',' ...
        StringG','StringH','StringI','StringJ','StringL','StringM',' ...
        StringN','StringO','StringP'}
        if isfield(data.BoardA(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardA(1,1).String{1,1})
                [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardA'; ...
                    dataString(2:end,:)]);
            end
        end
        if isfield(data.BoardB(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardB(1,1).String{1,1})
                [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardB'; ...
                    dataString(2:end,:)]);
            end
        end
        if isfield(data.BoardC(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardC(1,1).String{1,1})
```

```
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardC'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardD(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardD(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardD'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardE(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardE(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardE'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardF(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardF(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardF'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardG(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardG(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardG'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardH(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardH(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardH'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardI(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardI(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardI'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardJ(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardJ(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardJ'; ...
                dataString(2:end,:)]);
        end
    end
    if isfield(data.BoardL(1,1),'String')
        if strcmp(strrep(dataString{1,1},'String','String '),data.BoardL(1,1).String{1,1})
            [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardL'; ...
                dataString(2:end,:)]);
        end
```

```
        end
        if isfield(data.BoardM(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardM(1,1).String ...
                {1,1})
                [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardM'; ...
                    dataString(2:end,:)]);
            end
        end
        if isfield(data.BoardN(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardN(1,1).String{1,1})
                [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardN'; ...
                    dataString(2:end,:)]);
            end
        end
        if isfield(data.BoardO(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardO(1,1).String{1,1})
                [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardO'; ...
                    dataString(2:end,:)]);
            end
        end
        if isfield(data.BoardP(1,1),'String')
            if strcmp(strrep(dataString{1,1},'String','String '),data.BoardP(1,1).String{1,1})
                [returnedData,legendData,returnedCode] = MPFDBoardData(data,['BoardP'; ...
                    dataString(2:end,:)]);
            end
        end
        for i = 1:size(returnedCode,1)
            returnedCode{i,1} = [dataString{1,1},'.',returnedCode{i,1}]; %#ok<AGROW>
            legendData{i,1} = [dataString{1,1},'  ',legendData{i,1}]; %#ok<AGROW>
        end
end
end

function [returnedData,legendData,returnedCode] = MPFDBoardData(data,dataString)

returnedData = [];
legendData = [];
returnedCode = [];
switch dataString{2,1}
    case 'String'
        if isfield(data.(dataString{1,1})(1,1),'String')
            returnedData = data.(dataString{1,1})(1,1).String{1,1};
            legendData = {'String Connected'};
            returnedCode = {'String'};
        end
    case 'Box'
        if isfield(data.(dataString{1,1})(1,1),'Box')
            returnedData = data.(dataString{1,1})(1,1).Box{1,1};
            legendData = {'Box Connected'};
            returnedCode = {'Box'};
        end
```

```
case 'Present'
    if isfield(data.(dataString{1,1})(1,1),'Present')
        returnedData = [data.(dataString{1,1})(1,:).Present];
        legendData = {'String Present'};
        returnedCode = {'Present'};
    end
case 'StringConnected'
    if isfield(data.(dataString{1,1})(1,1),'StringConnected')
        returnedData = [data.(dataString{1,1})(1,:).StringConnected];
        legendData = {'String Connected'};
        returnedCode = {'StringConnected'};
    end
case 'PulseOverflow'
    if isfield(data.(dataString{1,1})(1,1),'PulseOverflow')
        returnedData = [data.(dataString{1,1})(1,:).PulseOverflow];
        legendData = {'Pulse Overflow'};
        returnedCode = {'PulseOverflow'};
    end
case 'PulseUnderflow'
    if isfield(data.(dataString{1,1})(1,1),'PulseUnderflow')
        returnedData = [data.(dataString{1,1})(1,:).PulseUnderflow];
        legendData = {'Pulse Underflow'};
        returnedCode = {'PulseUnderflow'};
    end
case 'ADCSpeed'
    if isfield(data.(dataString{1,1})(1,1),'ADCSpeed')
        returnedData = [data.(dataString{1,1})(1,:).ADCSpeed];
        legendData = {'ADC Speed (bits)'};
        returnedCode = {'ADCSpeed'};
    end
case 'Temperature'
    if isfield(data.(dataString{1,1})(1,1),'Temperature') && size(dataString,1) > 2
        dataString{2,1} = 'Temperature';
        [returnedData,legendData,returnedCode] = MPFDTemperatureBoardData(data,dataString ...
            (:));
        for i = 1:size(returnedCode,1)
            returnedCode{i,1} = ['Temperature.',returnedCode{i,1}];
        end
    end
case {'HV','hv'}
    if isfield(data.(dataString{1,1})(1,1),'HV') && size(dataString,1) > 2
        dataString{2,1} = 'HV';
        [returnedData,legendData,returnedCode] = MPFDHVBoardData(data,dataString(:));
        for i = 1:size(returnedCode,1)
            returnedCode{i,1} = ['HV.',returnedCode{i,1}];
        end
    end
case {'Pulse','pulse'}
    if isfield(data.(dataString{1,1})(1,1),'Pulse') && size(dataString,1) > 2
        dataString{2,1} = 'Pulse';
        [returnedData,legendData,returnedCode] = MPFDPulseMSVCurrentBoardData(data, ...
```

```
                dataString(:));
            % for i = 1:size(returnedCode,1)
            % returnedCode{i,1} = ['Pulse.',returnedCode{i,1}];
            % end
        end
    case {'MSV','msv'}
        if isfield(data.(dataString{1,1})(1,1),'MSV') && size(dataString,1) > 2
            dataString{2,1} = 'MSV';
            [returnedData,legendData,returnedCode] = MPFDPulseMSVCurrentBoardData(data, ...
                dataString(:));
            % for i = 1:size(returnedCode,1)
            % returnedCode{i,1} = ['MSV.',returnedCode{i,1}];
            % end
        end
    case {'Current','current','Curr','curr'}
        if isfield(data.(dataString{1,1})(1,1),'Current') && size(dataString,1) > 2
            dataString{2,1} = 'Current';
            [returnedData,legendData,returnedCode] = MPFDPulseMSVCurrentBoardData(data, ...
                dataString(:));
            % for i = 1:size(returnedCode,1)
            % returnedCode{i,1} = ['Current.',returnedCode{i,1}];
            % end
        end
end
end

function [returnedData,legendData,returnedCode] = MPFDTemperatureBoardData(data,dataString)

dataStringIndex = dataString{3,1}(1,min([size(dataString{3,1},2)+1,strfind(dataString{3,1},' ('),strfind( ...
    dataString{3,1},'{')]):end);
dataStringCode = strrep(dataString{3,1},dataStringIndex,'');
switch dataStringCode
    case 'Resolution'
        if isfield(data.(dataString{1,1})(1,1).Temperature,'Resolution')
            returnedData = data.(dataString{1,1})(1,:).Temperature.Resolution(1,1);
        else
            returnedData = zeros(size(data.(dataString{1,1})(1,:),1));
        end
        legendData = {'Temperature ADC Resolution'};
        returnedCode = {'Resolution(1,1)'};
    case 'New'
        returnedData = {false(size(data.(dataString{1,1})(1,:),1));false(size(data.(dataString{1,1})(1,:),1) ...
            );false(size(data.(dataString{1,1})(1,:),1));false(size(data.(dataString{1,1})(1,:),1))};
        legendData = {'Temperature 1 New';'Temperature 2 New';'Temperature ...
            3 New';'Temperature 4 New'};
        returnedCode = {'New(1,1)';'New(1,2)';'New(1,3)';'New(1,4)'};
        if isfield(data.(dataString{1,1})(1,1).Temperature,'New')
            if any(strcmp(dataStringIndex,{' (1,1)';' (:,1)';' (1,:)';' (:)';' (:,:)'})) || ...
                isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
```

```
            temp = [temp; data.(dataString{1,1})(1,i).Temperature.New(1,1)]; %#ok<AGROW ...
                >
        end
        returnedData{1,1} = temp;
        legendData{1,1} = 'Temperature 1 New';
        returnedCode{1,1} = 'New(1,1)';
    end
    if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
        (:,:)'})) || isempty(dataStringIndex)
        temp = [];
        for i = 1:size(data.(dataString{1,1}),2)
            temp = [temp; data.(dataString{1,1})(1,i).Temperature.New(1,2)]; %#ok<AGROW ...
                >
        end
        returnedData{2,1} = temp;
        legendData{2,1} = 'Temperature 2 New';
        returnedCode{2,1} = 'New(1,2)';
    end
    if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
        (:,:)'})) || isempty(dataStringIndex)
        temp = [];
        for i = 1:size(data.(dataString{1,1}),2)
            temp = [temp; data.(dataString{1,1})(1,i).Temperature.New(1,3)]; %#ok<AGROW ...
                >
        end
        returnedData{3,1} = temp;
        legendData{3,1} = 'Temperature 3 New';
        returnedCode{3,1} = 'New(1,3)';
    end
    if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
        (:,:)'})) || isempty(dataStringIndex)
        temp = [];
        for i = 1:size(data.(dataString{1,1}),2)
            temp = [temp; data.(dataString{1,1})(1,i).Temperature.New(1,4)]; %#ok<AGROW ...
                >
        end
        returnedData{4,1} = temp;
        legendData{4,1} = 'Temperature 4 New';
        returnedCode{4,1} = 'New(1,4)';
    end
end
temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
if isempty(dataStringIndex)
    temp = temp(1:4,1);
else
    temp = eval(['temp',dataStringIndex]);
    temp = unique(temp);
end
returnedData = returnedData(temp,1);
legendData = legendData(temp,1);
returnedCode = returnedCode(temp,1);
```

```
case 'bits'
    returnedData = {zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:) ...
        ,1));zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:),1))};
    legendData = {'Temperature 1 bits';'Temperature 2 bits';' ...
        Temperature 3 bits';'Temperature 4 bits'};
    returnedCode = {'bits(1,1)';'bits(1,2)';'bits(1,3)';'bits(1,4)'};
    if isfield(data.(dataString{1,1})(1,1).Temperature,'bits')
        if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
            isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.bits(1,1)]; %#ok<AGROW ...
                    >
            end
            returnedData{1,1} = temp;
            legendData{1,1} = 'Temperature 1 bits';
            returnedCode{1,1} = 'bits(1,1)';
        end
        if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.bits(1,2)]; %#ok<AGROW ...
                    >
            end
            returnedData{2,1} = temp;
            legendData{2,1} = 'Temperature 2 bits';
            returnedCode{2,1} = 'bits(1,2)';
        end
        if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.bits(1,3)]; %#ok<AGROW ...
                    >
            end
            returnedData{3,1} = temp;
            legendData{3,1} = 'Temperature 3 bits';
            returnedCode{3,1} = 'bits(1,3)';
        end
        if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.bits(1,4)]; %#ok<AGROW ...
                    >
            end
            returnedData{4,1} = temp;
            legendData{4,1} = 'Temperature 4 bits';
            returnedCode{4,1} = 'bits(1,4)';
        end
```

```
        end
        temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
        if isempty(dataStringIndex)
            temp = temp(1:4,1);
        else
            temp = eval(['temp',dataStringIndex]);
            temp = unique(temp);
        end
        returnedData = returnedData(temp,1);
        legendData = legendData(temp,1);
        returnedCode = returnedCode(temp,1);
    case 'mV'
        returnedData = {zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:) ...
            ,1));zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:),1))};
        legendData = {'Temperature 1 mV';'Temperature 2 mV';'Temperature 3 ...
            mV';'Temperature 4 mV'};
        returnedCode = {'mV(1,1)';'mV(1,2)';'mV(1,3)';'mV(1,4)'};
        if isfield(data.(dataString{1,1})(1,1).Temperature,'mV')
            returnedData = cell(0);
            legendData = cell(0);
            returnedCode = cell(0);
            if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
                isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.mV(1,1)]; %#ok<AGROW ...
                        >
                end
                returnedData{1,1} = temp;
                legendData{1,1} = 'Temperature 1 mV';
                returnedCode{1,1} = 'mV(1,1)';
            end
            if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.mV(1,2)]; %#ok<AGROW ...
                        >
                end
                returnedData{2,1} = temp;
                legendData{2,1} = 'Temperature 2 mV';
                returnedCode{2,1} = 'mV(1,2)';
            end
            if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.mV(1,3)]; %#ok<AGROW ...
                        >
                end
                returnedData{3,1} = temp;
```

```
            legendData{3,1} = 'Temperature 3 mV';
            returnedCode{3,1} = 'mV(1,3)';
        end
        if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.mV(1,4)]; %#ok<AGROW ...
                    >
            end
            returnedData{4,1} = temp;
            legendData{4,1} = 'Temperature 4 mV';
            returnedCode{4,1} = 'mV(1,4)';
        end
    end
    temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
    if isempty(dataStringIndex)
        temp = temp(1:4,1);
    else
        temp = eval(['temp',dataStringIndex]);
        temp = unique(temp);
    end
    returnedData = returnedData(temp,1);
    legendData = legendData(temp,1);
    returnedCode = returnedCode(temp,1);
case 'degC'
    returnedData = {zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:) ...
        ,1));zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:),1))};
    legendData = {'Temperature 1 degC';'Temperature 2 degC';' ...
        Temperature 3 degC';'Temperature 4 degC'};
    returnedCode = {'degC(1,1)';'degC(1,2)';'degC(1,3)';'degC(1,4)'};
    if isfield(data.(dataString{1,1})(1,1).Temperature,'degC')
        returnedData = cell(0);
        legendData = cell(0);
        returnedCode = cell(0);
        if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
            isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.degC(1,1)]; %#ok< ...
                    AGROW>
            end
            returnedData{1,1} = temp;
            legendData{1,1} = 'Temperature 1 degC';
            returnedCode{1,1} = 'degC(1,1)';
        end
        if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).Temperature.degC(1,2)]; %#ok< ...
```

```
                    AGROW>
                end
                returnedData{2,1} = temp;
                legendData{2,1} = 'Temperature 2 degC';
                returnedCode{2,1} = 'degC(1,2)';
            end
            if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.degC(1,3)]; %#ok< ...
                        AGROW>
                end
                returnedData{3,1} = temp;
                legendData{3,1} = 'Temperature 3 degC';
                returnedCode{3,1} = 'degC(1,3)';
            end
            if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.degC(1,4)]; %#ok< ...
                        AGROW>
                end
                returnedData{4,1} = temp;
                legendData{4,1} = 'Temperature 4 degC';
                returnedCode{4,1} = 'degC(1,4)';
            end
        end
        temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
        if isempty(dataStringIndex)
            temp = temp(1:4,1);
        else
            temp = eval(['temp',dataStringIndex]);
            temp = unique(temp);
        end
        returnedData = returnedData(temp,1);
        legendData = legendData(temp,1);
        returnedCode = returnedCode(temp,1);
    case 'degF'
        returnedData = {zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:) ...
            ,1));zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:),1))};
        legendData = {'Temperature 1 degF';'Temperature 2 degF';' ...
            Temperature 3 degF';'Temperature 4 degF'};
        returnedCode = {'degF(1,1)';'degF(1,2)';'degF(1,3)';'degF(1,4)'};
        if isfield(data.(dataString{1,1})(1,1).Temperature,'degF')
            if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
                isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.degF(1,1)]; %#ok<AGROW ...
```

```
                    >
                end
                returnedData{1,1} = temp;
                legendData{1,1} = 'Temperature 1 degF';
                returnedCode{1,1} = 'degF(1,1)';
            end
            if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.degF(1,2)]; %#ok<AGROW ...
                        >
                end
                returnedData{2,1} = temp;
                legendData{2,1} = 'Temperature 2 degF';
                returnedCode{2,1} = degFmV(1,2)';
            end
            if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.degF(1,3)]; %#ok<AGROW ...
                        >
                end
                returnedData{3,1} = temp;
                legendData{3,1} = 'Temperature 3 degF';
                returnedCode{3,1} = 'degF(1,3)';
            end
            if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).Temperature.degF(1,4)]; %#ok<AGROW ...
                        >
                end
                returnedData{4,1} = temp;
                legendData{4,1} = 'Temperature 4 degF';
                returnedCode{4,1} = 'degF(1,4)';
            end
        end
        temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
        if isempty(dataStringIndex)
            temp = temp(1:4,1);
        else
            temp = eval(['temp',dataStringIndex]);
            temp = unique(temp);
        end
        returnedData = returnedData(temp,1);
        legendData = legendData(temp,1);
        returnedCode = returnedCode(temp,1);
    otherwise
```

```
        returnedData = 0;
        legendData = {'Case Not Found'};
end
end

function [returnedData,legendData,returnedCode] = MPFDHVBoardData(data,dataString)

returnedData = [];
legendData = [];
returnedCode = [];

dataStringIndex = dataString{3,1}(1,min([size(dataString{3,1},2)+1,strfind(dataString{3,1},'('),strfind( ...
    dataString{3,1},'{')]):end);
dataStringCode = strrep(dataString{3,1},dataStringIndex,'');
switch dataStringCode
    case 'Speed'
        if isfield(data.(dataString{1,1})(1,1).HV,'Speed')
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).HV.Speed(1,1)]; %#ok<AGROW>
            end
            returnedData{1,1} = temp;
            legendData = {'HV ADC Speed (bits)'};
            returnedCode = {'Speed(1,1)'};
        end
    case 'New'
        if isfield(data.(dataString{1,1})(1,1).HV,'New')
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).HV.New(1,1)]; %#ok<AGROW>
            end
            returnedData{1,1} = temp;
            legendData = {'HV New Reading'};
            returnedCode = {'New(1,1)'};
        end
    case {'bits','Bits','raw','Raw'}
        if isfield(data.(dataString{1,1})(1,1).HV,'bits')
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).HV.bits(1,1)]; %#ok<AGROW>
            end
            returnedData{1,1} = temp;
            legendData = {'HV (bits)'};
            returnedCode = {'bits(1,1)'};
        end
    case {'voltage','Voltage','V'}
        if isfield(data.(dataString{1,1})(1,1).HV,'voltage')
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).HV.voltage(1,1)]; %#ok<AGROW>
            end
```

```
            returnedData{1,1} = temp;
            legendData = {'HV (V)'};
            returnedCode = {'voltage(1,1)'};
        end
end
end

function [returnedData,legendData,returnedCode] = MPFDPulseMSVCurrentBoardData(data,dataString ...
    )

if size(dataString,1) == 3
    [returnedDataL,legendDataL,returnedCodeL] = MPFDPulseLBHData(data,[dataString;'L']);
    [returnedDataB,legendDataB,returnedCodeB] = MPFDPulseLBHData(data,[dataString;'B']);
    [returnedDataH,legendDataH,returnedCodeH] = MPFDPulseLBHData(data,[dataString;'H']);
    if ~iscell(returnedDataL)
        returnedDataL = {returnedDataL};
    end
    if ~iscell(legendDataL)
        legendDataL = {legendDataL};
    end
    if ~iscell(returnedCodeL)
        returnedCodeL = {returnedCodeL};
    end
    if ~iscell(returnedDataB)
        returnedDataB = {returnedDataB};
    end
    if ~iscell(legendDataB)
        legendDataB = {legendDataB};
    end
    if ~iscell(returnedCodeB)
        returnedCodeB = {returnedCodeB};
    end
    if ~iscell(returnedDataH)
        returnedDataH = {returnedDataH};
    end
    if ~iscell(legendDataH)
        legendDataH = {legendDataH};
    end
    if ~iscell(returnedCodeH)
        returnedCodeH = {returnedCodeH};
    end
    returnedData = [returnedDataL(:); returnedDataB(:); returnedDataH(:)];
    legendData = [legendDataL(:); legendDataB(:); legendDataH(:)];
    returnedCode = [returnedCodeL(:); returnedCodeB(:); returnedCodeH(:)];
else
    [returnedData,legendData,returnedCode] = MPFDPulseLBHData(data,dataString);
    if ~iscell(returnedData)
        returnedData = {returnedData};
    end
    if ~iscell(legendData)
        legendData = {legendData};
```

```
    end
    if ~iscell(returnedCode)
        returnedCode = {returnedCode};
    end
end
end

function [returnedData,legendData,returnedCode] = MPFDPulseLBHData(data,dataString)

dataStringIndex = dataString{4,1}(1,min([size(dataString{4,1},2)+1,strfind(dataString{4,1},'(' ),strfind( ...
    dataString{4,1},'{')]):end);
dataStringCode = strrep(dataString{4,1},dataStringIndex,'');
switch dataStringCode
    case 'New'
        returnedData = {false(size(data.(dataString{1,1})(1,:),1));false(size(data.(dataString{1,1})(1,:),1) ...
            );false(size(data.(dataString{1,1})(1,:),1));false(size(data.(dataString{1,1})(1,:),1))};
        legendData = {'1 New';'2 New';'3 New';'4 New'};
        returnedCode = {'New(1,1)';'New(1,2)';'New(1,3)';'New(1,4)'};
        if isfield(data.(dataString{1,1})(1,1).(dataString{2,1}).(dataString{3,1}),'New')
            if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
                isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).New ...
                        (1,1)]; %#ok<AGROW>
                end
                returnedData{1,1} = temp;
                legendData{1,1} = '1 New';
                returnedCode{1,1} = 'New(1,1)';
            end
            if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).New ...
                        (1,2)]; %#ok<AGROW>
                end
                returnedData{2,1} = temp;
                legendData{2,1} = '2 New';
                returnedCode{2,1} = 'New(1,2)';
            end
            if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).New ...
                        (1,3)]; %#ok<AGROW>
                end
                returnedData{3,1} = temp;
                legendData{3,1} = '3 New';
                returnedCode{3,1} = 'New(1,3)';
```

```
        end
        if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).New ...
                    (1,4)]; %#ok<AGROW>
            end
            returnedData{4,1} = temp;
            legendData{4,1} = '4 New';
            returnedCode{4,1} = 'New(1,4)';
        end
    end
    temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
    if isempty(dataStringIndex)
        temp = temp(1:4,1);
    else
        temp = eval(['temp',dataStringIndex]);
        temp = unique(temp);
    end
    returnedData = returnedData(temp,1);
    legendData = legendData(temp,1);
    returnedCode = returnedCode(temp,1);
case 'bits'
    returnedData = {zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:) ...
        ,1));zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:),1))};
    legendData = {'1 bits';'2 bits';'3 bits';'4 bits'};
    returnedCode = {'bits(1,1)';'bits(1,2)';'bits(1,3)';'bits(1,4)'};
    if isfield(data.(dataString{1,1})(1,1).(dataString{2,1}).(dataString{3,1}),'bits')
        if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
            isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).bits ...
                    (1,1)]; %#ok<AGROW>
            end
            returnedData{1,1} = temp;
            legendData{1,1} = '1 bits';
            returnedCode{1,1} = 'bits(1,1)';
        end
        if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).bits ...
                    (1,2)]; %#ok<AGROW>
            end
            returnedData{2,1} = temp;
            legendData{2,1} = '2 bits';
            returnedCode{2,1} = 'bits(1,2)';
        end
```

```
        if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).bits ...
                    (1,3)]; %#ok<AGROW>
            end
            returnedData{3,1} = temp;
            legendData{3,1} = '3 bits';
            returnedCode{3,1} = 'bits(1,3)';
        end
        if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
            (:,:)'})) || isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).bits ...
                    (1,4)]; %#ok<AGROW>
            end
            returnedData{4,1} = temp;
            legendData{4,1} = '4 bits';
            returnedCode{4,1} = 'bits(1,4)';
        end
    end
    temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
    if isempty(dataStringIndex)
        temp = temp(1:4,1);
    else
        temp = eval(['temp',dataStringIndex]);
        temp = unique(temp);
    end
    returnedData = returnedData(temp,1);
    legendData = legendData(temp,1);
    returnedCode = returnedCode(temp,1);
case 'cps'
    returnedData = {zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:) ...
        ,1));zeros(size(data.(dataString{1,1})(1,:),1));zeros(size(data.(dataString{1,1})(1,:),1))};
    legendData = {'1 cps';'2 cps';'3 cps';'4 cps'};
    returnedCode = {'cps(1,1)';'cps(1,2)';'cps(1,3)';'cps(1,4)'};
    if isfield(data.(dataString{1,1})(1,1).(dataString{2,1}).(dataString{3,1}),'cps')
        returnedData = cell(0);
        legendData = cell(0);
        returnedCode = cell(0);
        if any(strcmp(dataStringIndex,{'(1,1)';'(:,1)';'(1,:)';'(:)';'(:,:)'})) || ...
            isempty(dataStringIndex)
            temp = [];
            for i = 1:size(data.(dataString{1,1}),2)
                temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).cps ...
                    (1,1)]; %#ok<AGROW>
            end
            returnedData{1,1} = temp;
            legendData{1,1} = '1 cps';
```

```
                returnedCode{1,1} = 'cps(1,1)';
            end
            if any(strcmp(dataStringIndex,{'(1,2)';'(2,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).cps  ...
                        (1,2)]; %#ok<AGROW>
                end
                returnedData{2,1} = temp;
                legendData{2,1} = '2 cps';
                returnedCode{2,1} = 'cps(1,2)';
            end
            if any(strcmp(dataStringIndex,{'(1,3)';'(3,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).cps  ...
                        (1,3)]; %#ok<AGROW>
                end
                returnedData{3,1} = temp;
                legendData{3,1} = '3 cps';
                returnedCode{3,1} = 'cps(1,3)';
            end
            if any(strcmp(dataStringIndex,{'(1,4)';'(4,1)';'(:,1)';'(1,:)';'(:)';' ...
                (:,:)'})) || isempty(dataStringIndex)
                temp = [];
                for i = 1:size(data.(dataString{1,1}),2)
                    temp = [temp; data.(dataString{1,1})(1,i).(dataString{2,1}).(dataString{3,1}).cps  ...
                        (1,4)]; %#ok<AGROW>
                end
                returnedData{4,1} = temp;
                legendData{4,1} = '4 cps';
                returnedCode{4,1} = 'cps(1,4)';
            end
        end
        temp = [1 2 3 4; 2 2 3 4; 3 3 3 4; 4 4 4 4];
        if isempty(dataStringIndex)
            temp = temp(1:4,1);
        else
            temp = eval(['temp',dataStringIndex]);
            temp = unique(temp);
        end
        returnedData = returnedData(temp,1);
        legendData = legendData(temp,1);
        returnedCode = returnedCode(temp,1);
    otherwise
        returnedData = [];
        legendData = [];
        returnedCode = [];
end
```

```
if ~iscell(returnedData)
    returnedData = {returnedData};
end
if ~iscell(legendData)
    legendData = {legendData};
end
if ~iscell(returnedCode)
    returnedCode = {returnedCode};
end
for i = 1:size(returnedData,1)
    legendData{i,1} = [dataString{2,1},'  ',dataString{3,1},legendData{i,1}];
    returnedCode{i,1} = [dataString{2,1},'.',dataString{3,1},'.',returnedCode{i,1}];
end
end
```

End MPFDData.m

Appendix C

Dimensional Drawings

C.1 MPFD^3-T Cavity Substrate

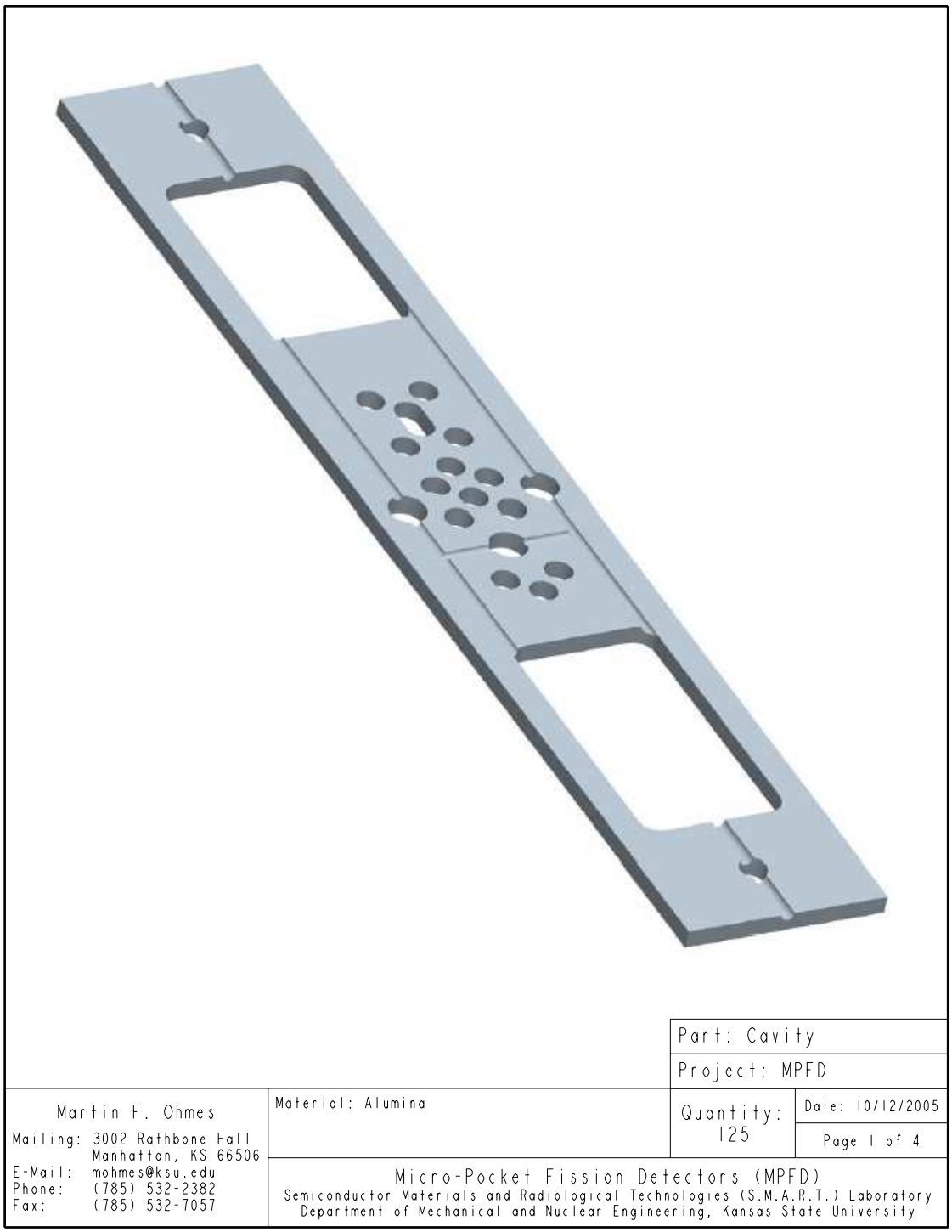


Figure C.1: MPFD^3-T substrate cavity drawing, page 1.

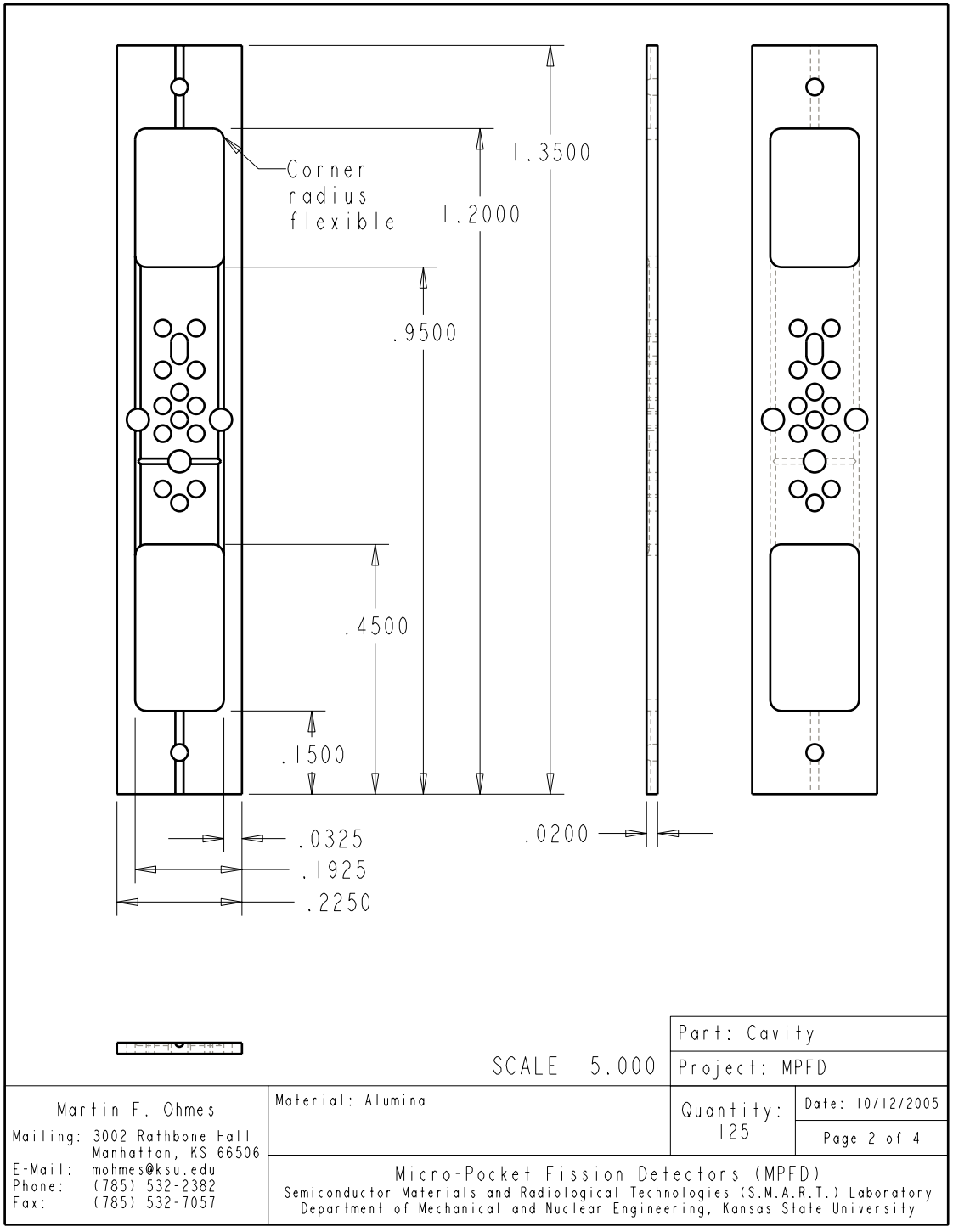
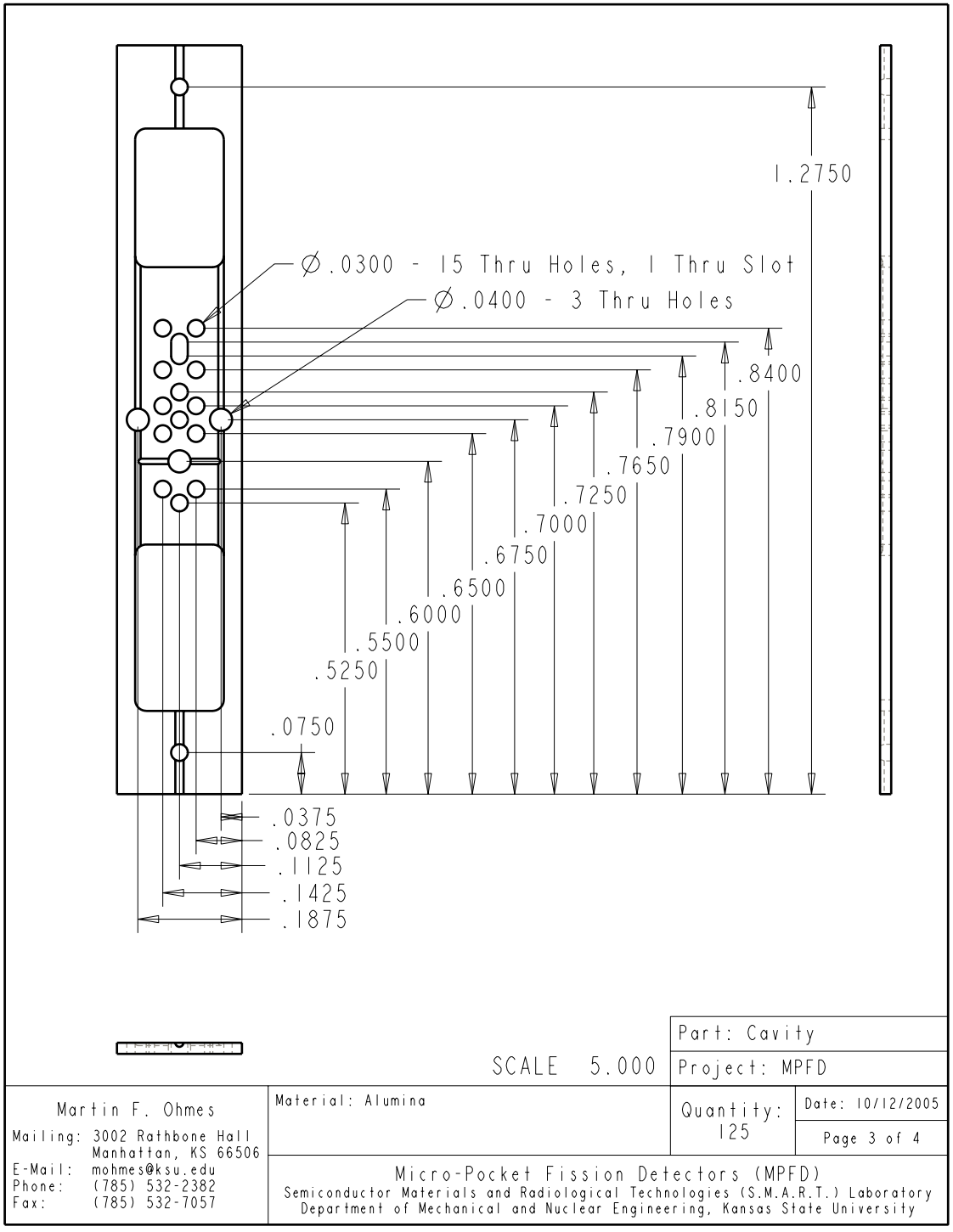
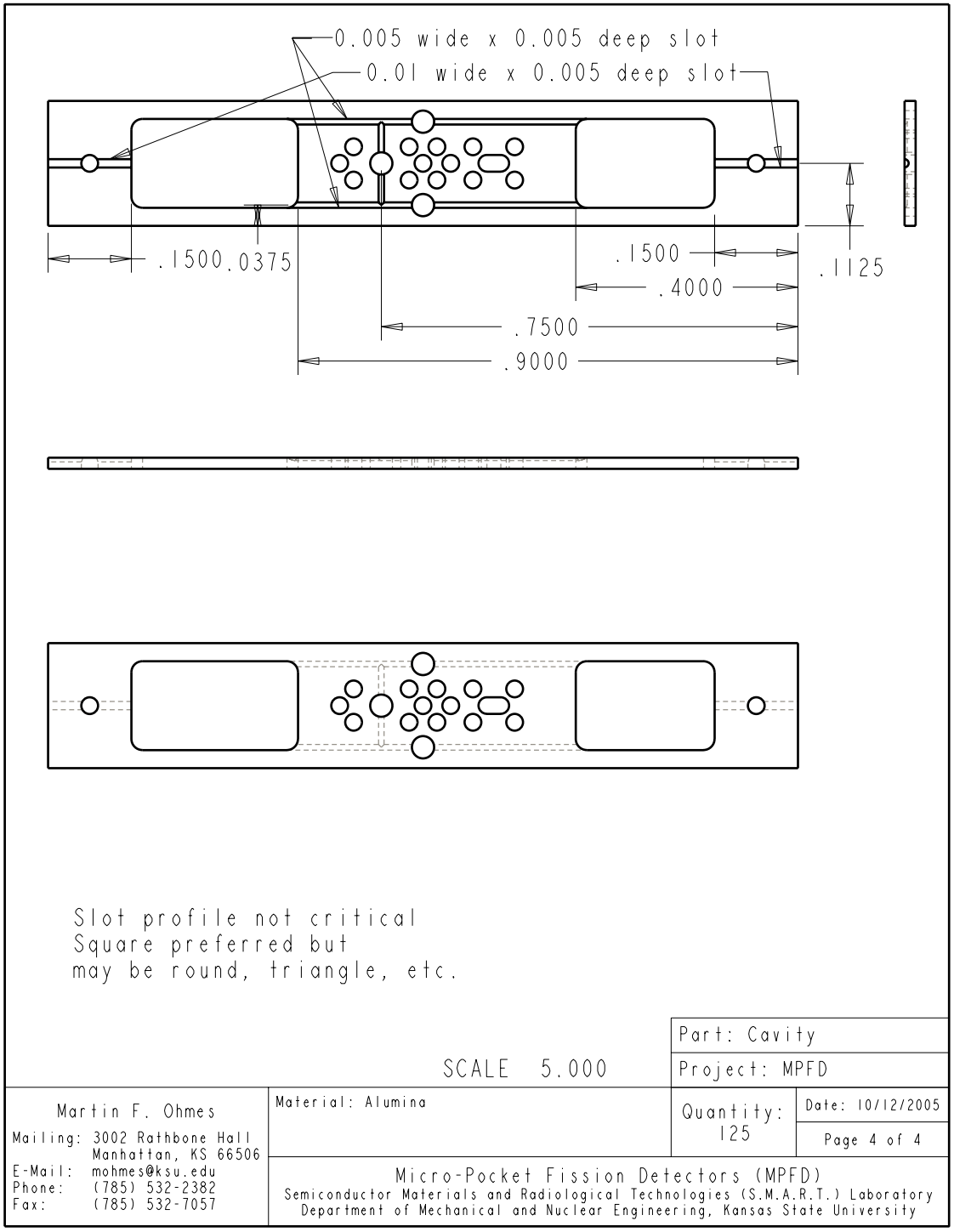


Figure C.2: MPFD^3-T substrate cavity drawing, page 2.



Martin F. Ohmes
Mailing: 3002 Rathbone Hall
Manhattan, KS 66506
E-Mail: mohmes@ksu.edu
Phone: (785) 532-2382
Fax: (785) 532-7057

Figure C.3: MPFD^3-T substrate cavity drawing, page 3.



Martin F. Ohmes
Mailing: 3002 Rathbone Hall
Manhattan, KS 66506
E-Mail: mohmes@ksu.edu
Phone: (785) 532-2382
Fax: (785) 532-7057

Figure C.4: MPFD^3-T substrate cavity drawing, page 4.

C.2 MPFD^3-T Base Substrate

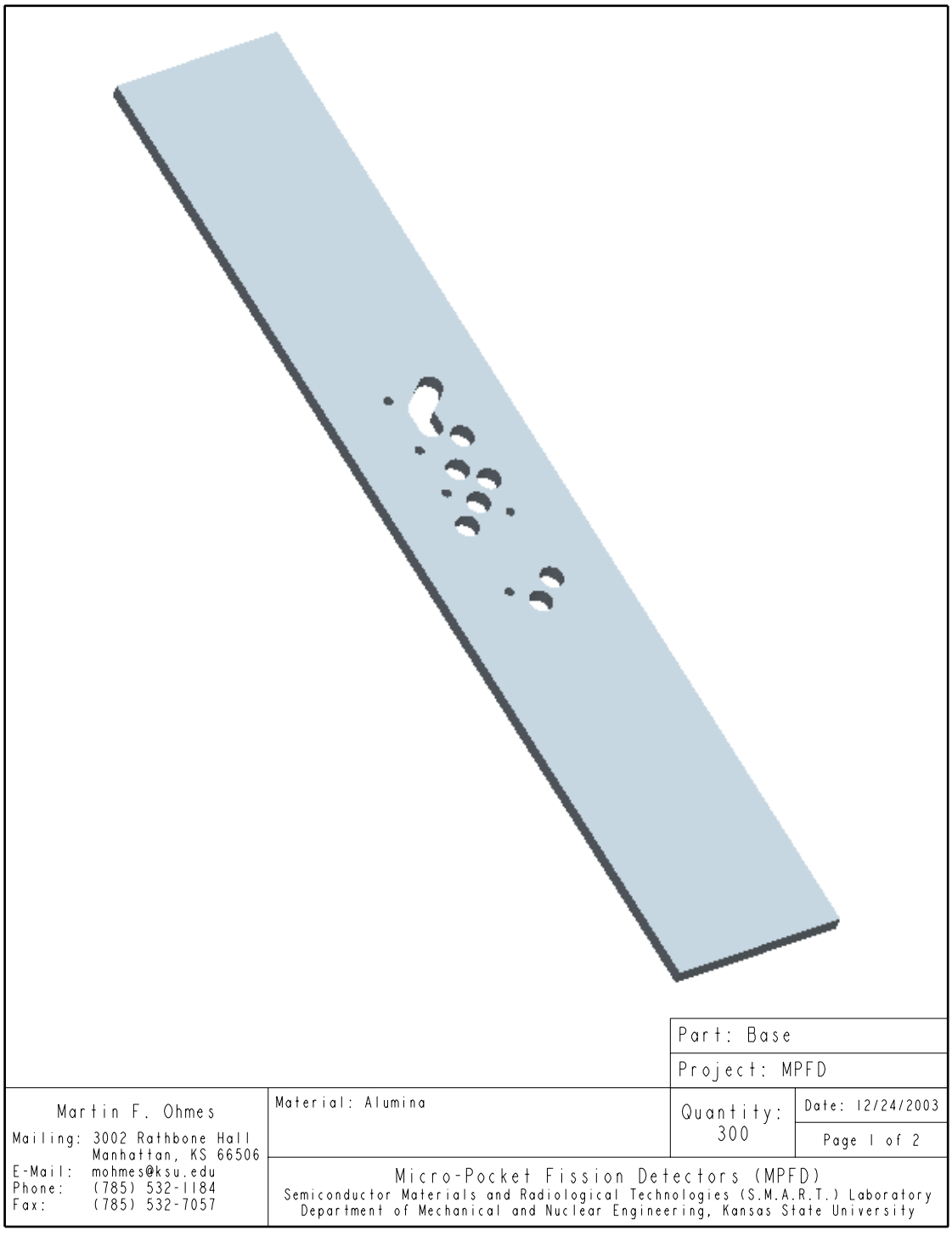


Figure C.5: MPFD^3-T substrate base drawing, page 1.

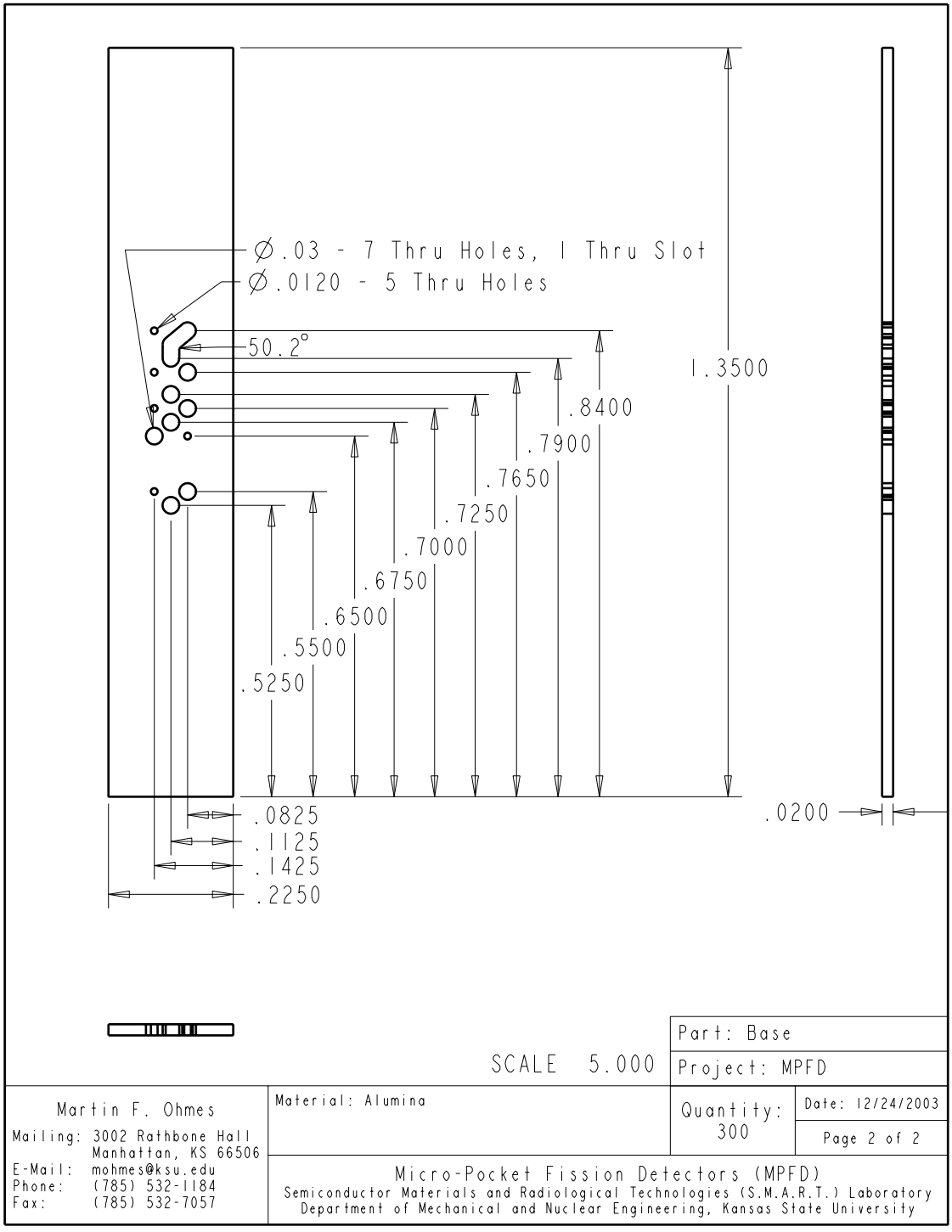


Figure C.6: MPFD^3-T substrate base drawing, page 2.

C.3 Contact Evaporation Mask

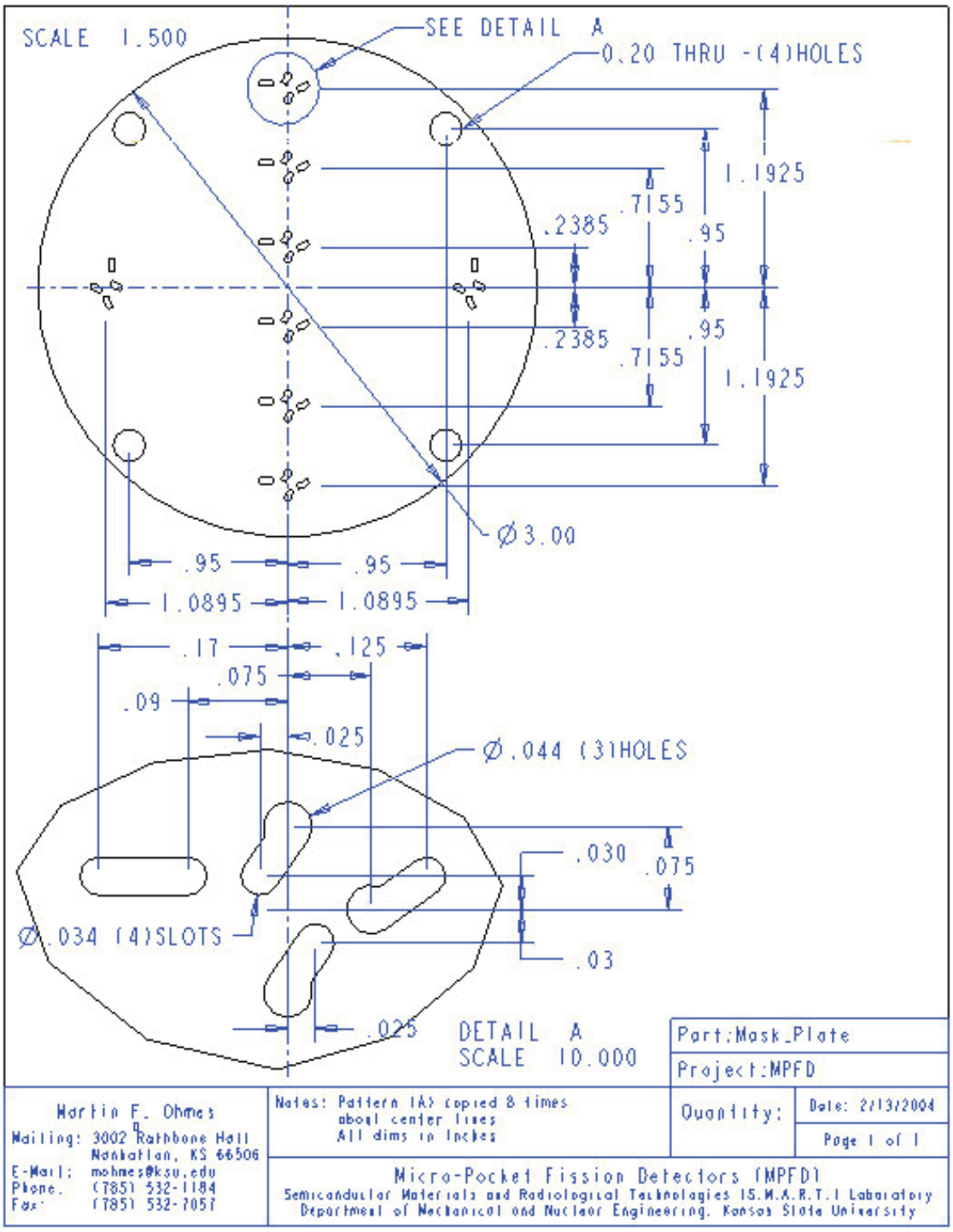


Figure C.7: Contact evaporation mask drawing.

C.4 Ion Mill Substrate Holder

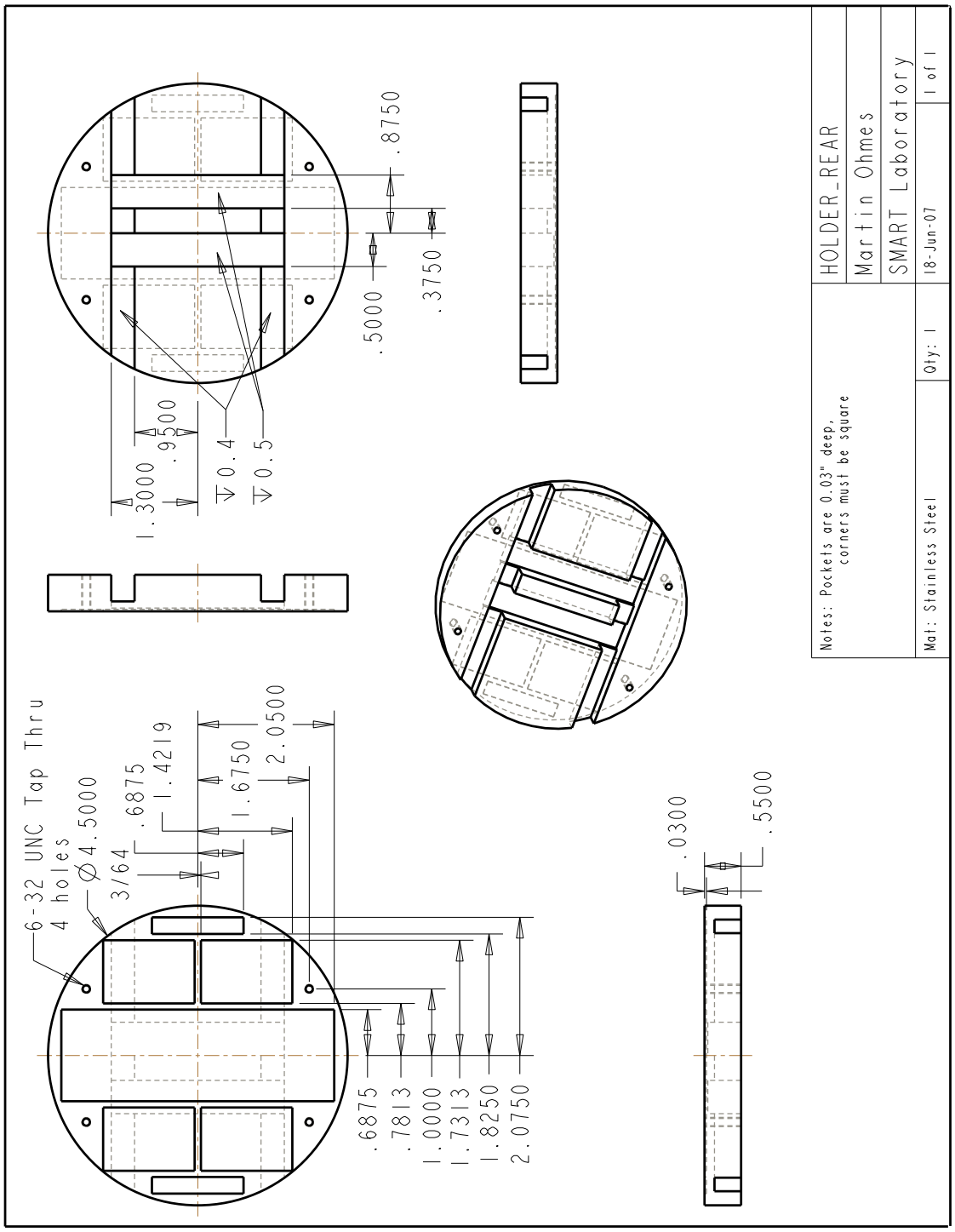


Figure C.8: Ion mill substrate holder back view.

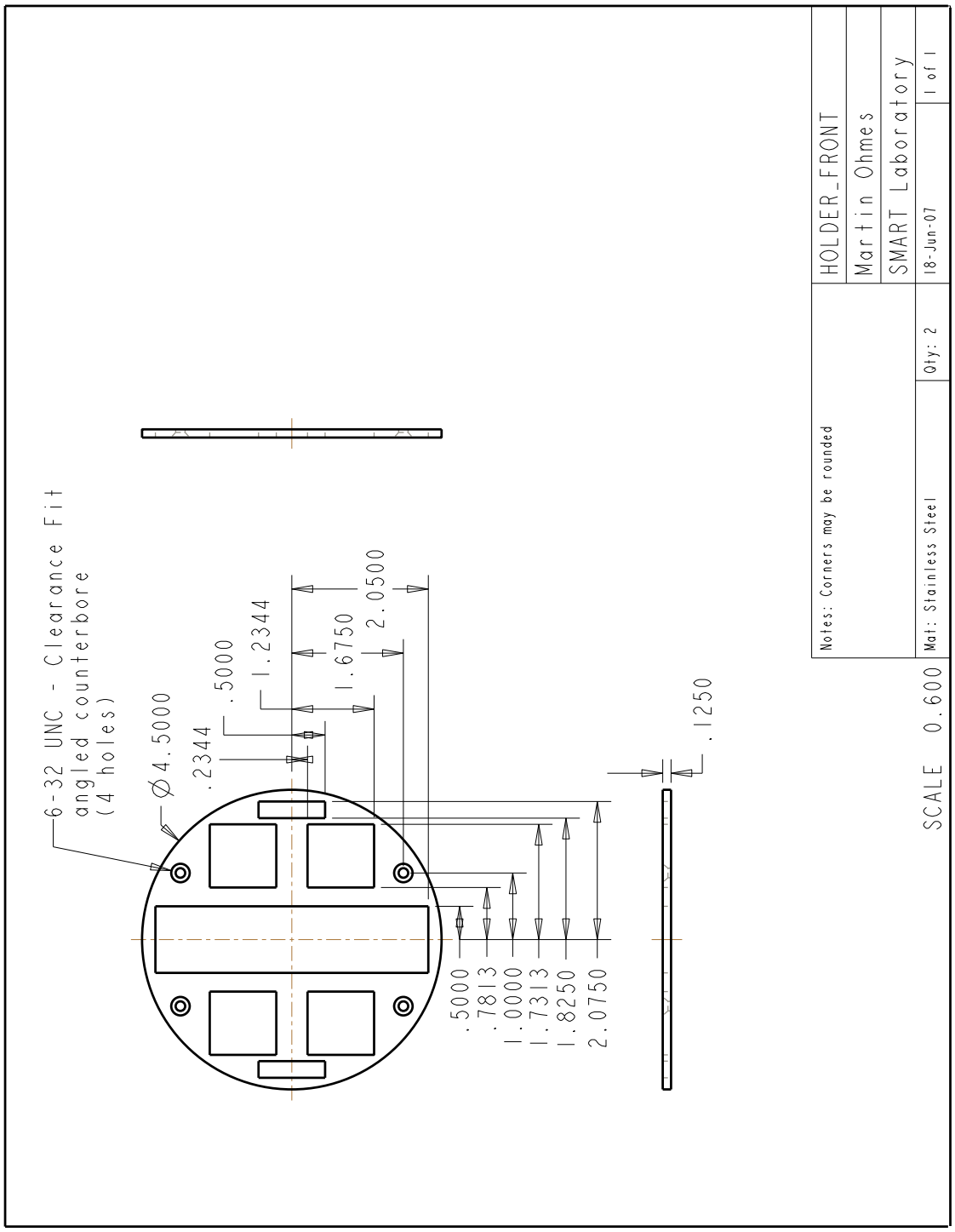


Figure C.9: Ion mill substrate holder front view.

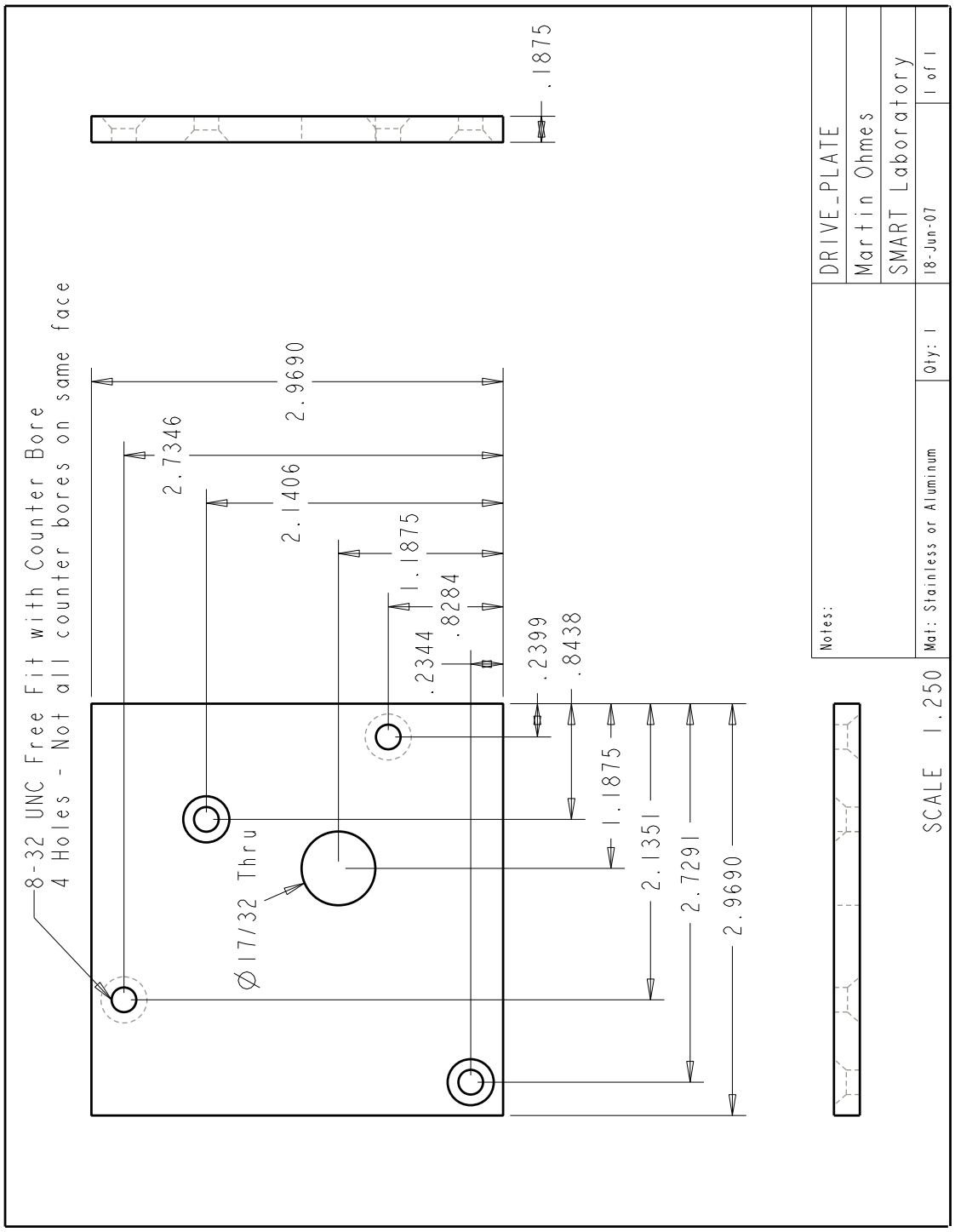


Figure C.10: Ion mill sample rotation drive motor mounting plate.

C.5 In-Core Housing Tube

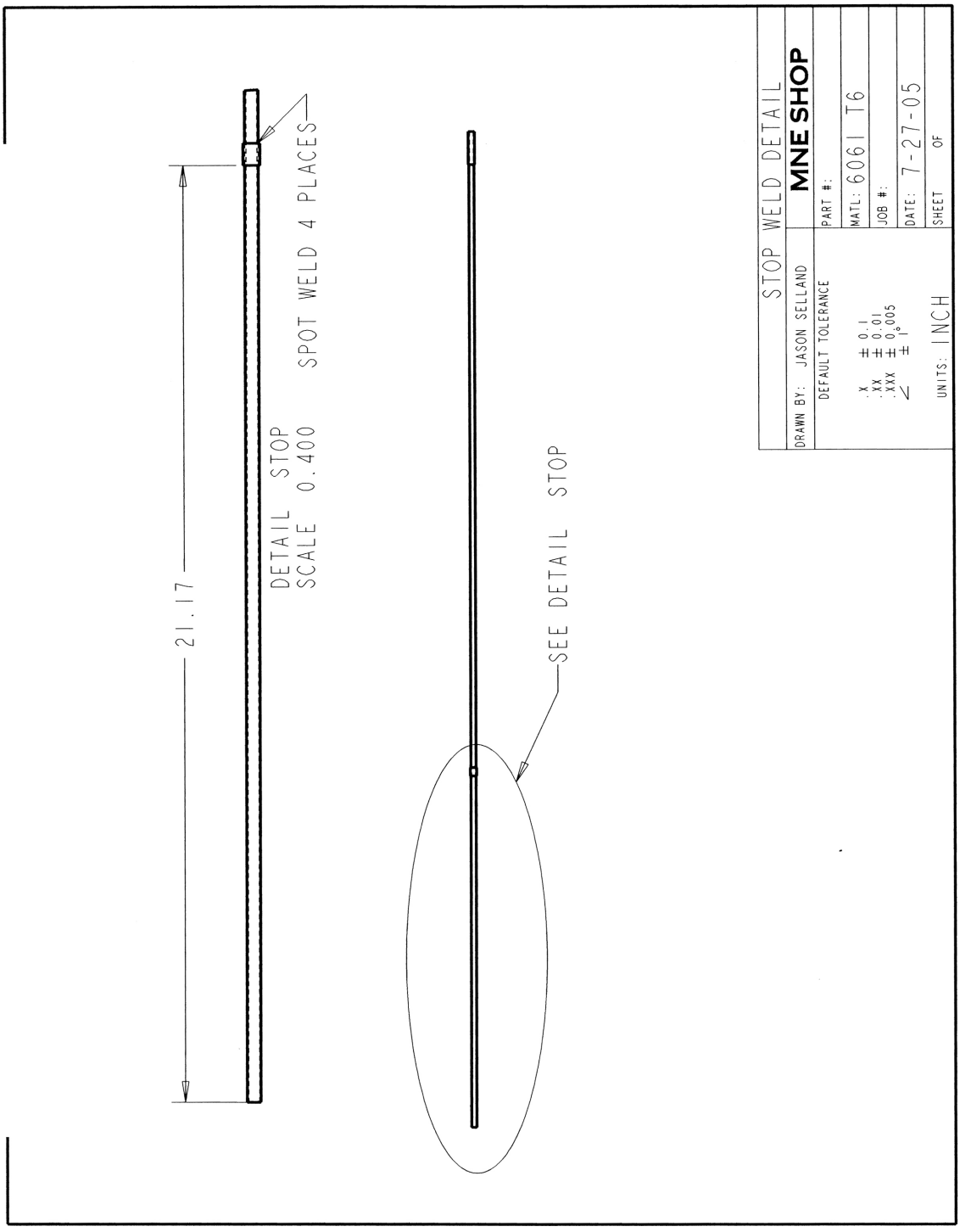


Figure C.11: Flux probe stop weld detail, page 1.

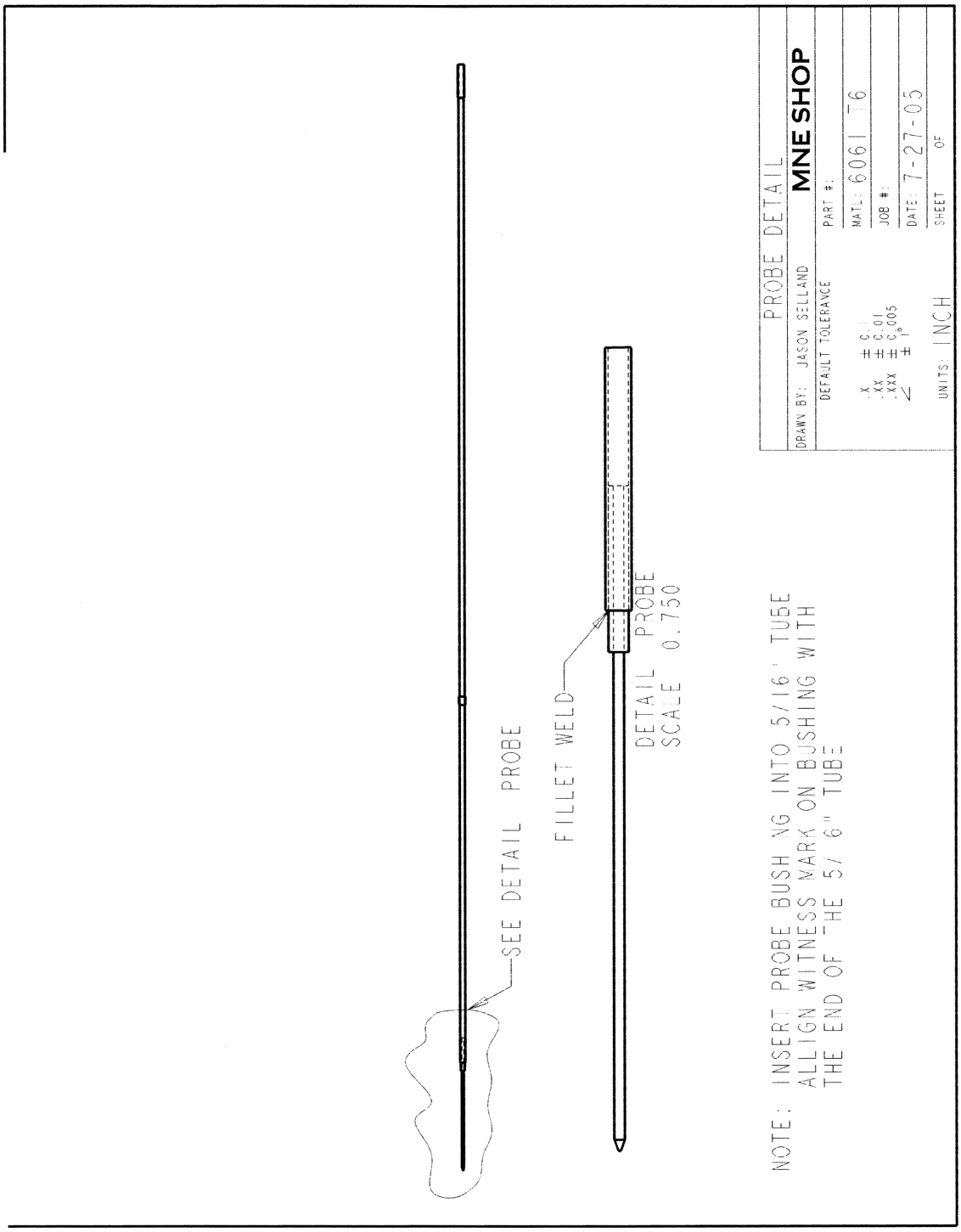


Figure C.12: Flux probe tip detail, page 2.

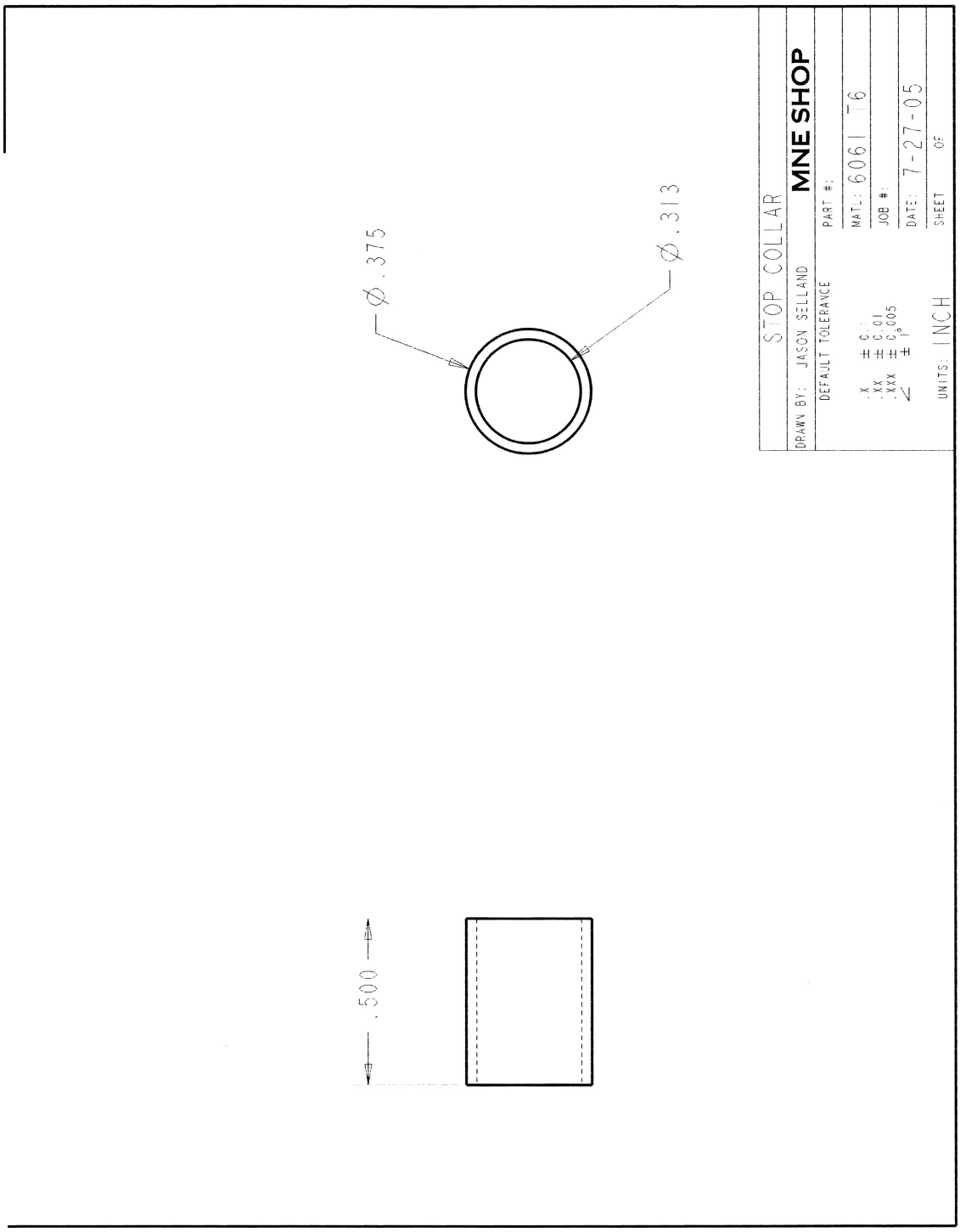


Figure C.13: Flux probe stop collar drawing, page 3.

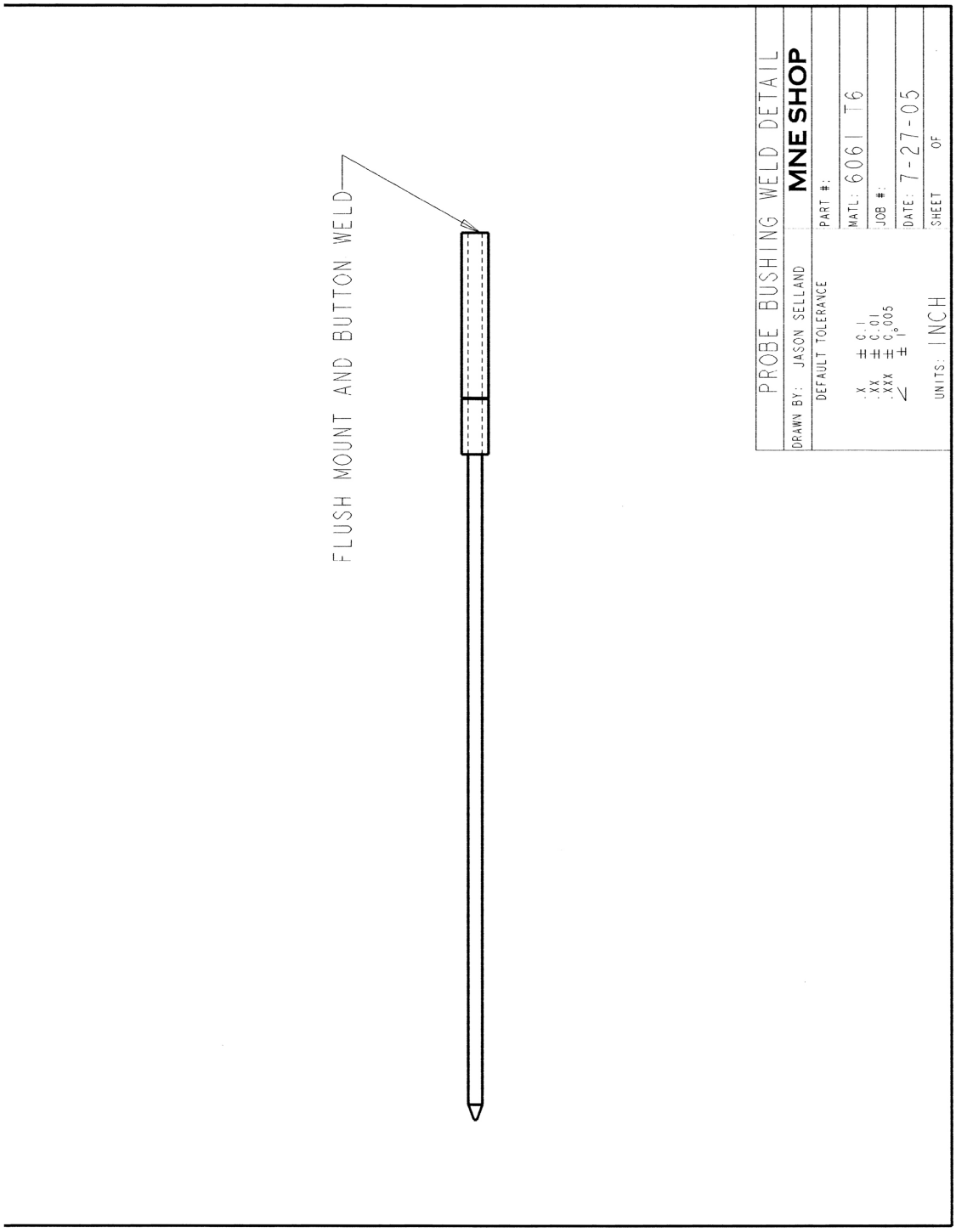


Figure C.14: Flux probe bushing weld detail, page 4.

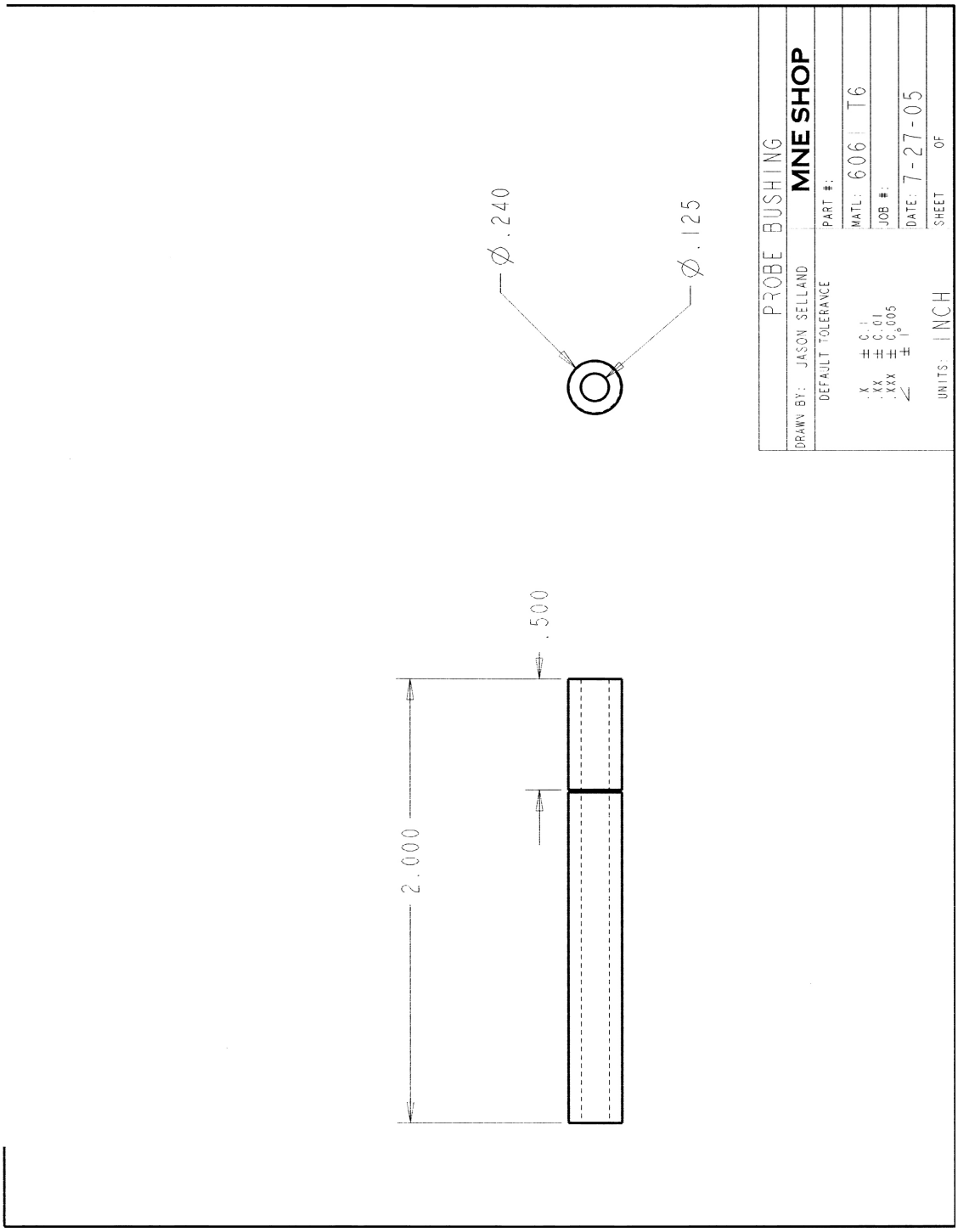


Figure C.15: Flux probe bushing drawing, page 5.

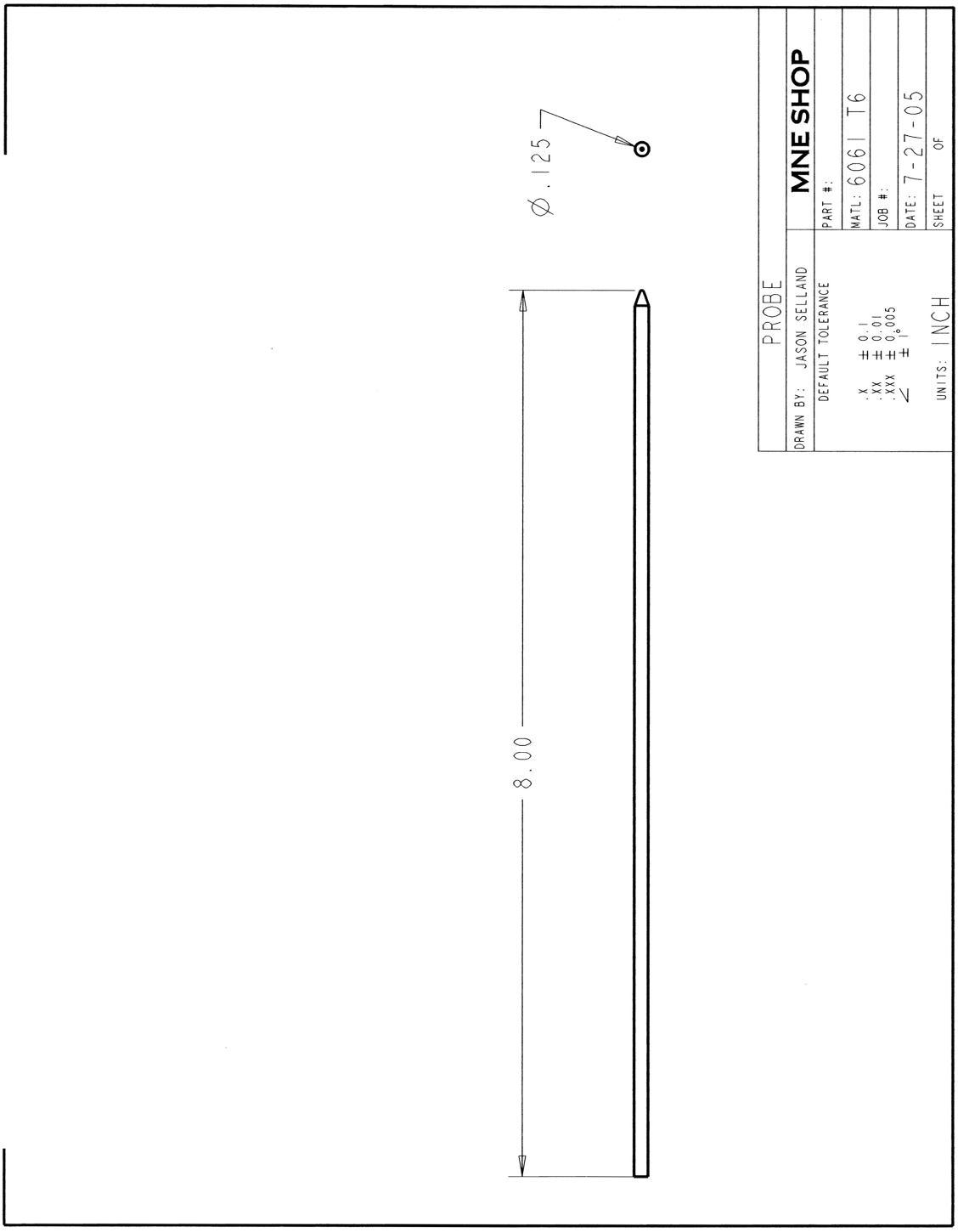


Figure C.16: Flux probe tip drawing, page 6.

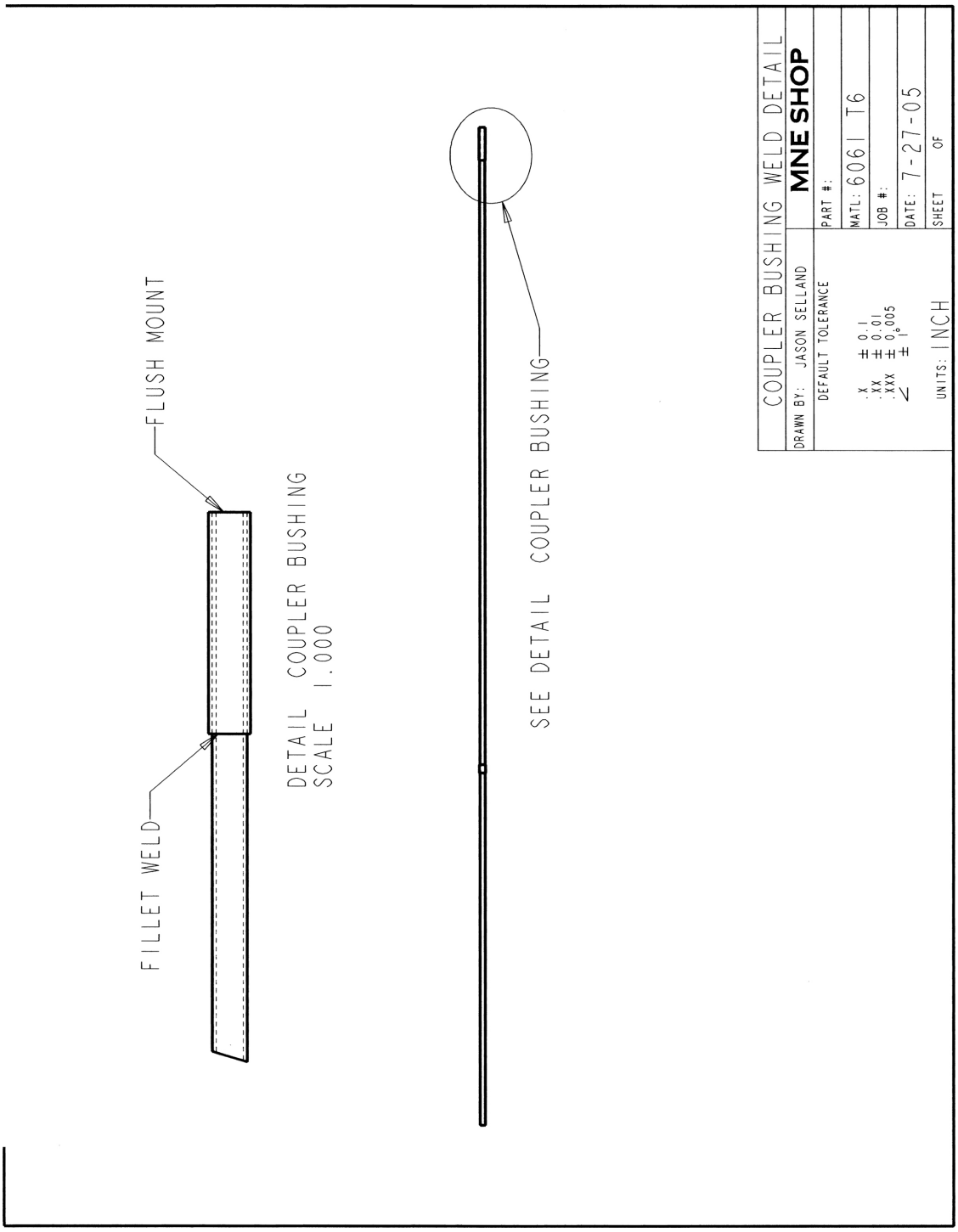


Figure C.17: Flux probe coupler bushing weld detail, page 7.

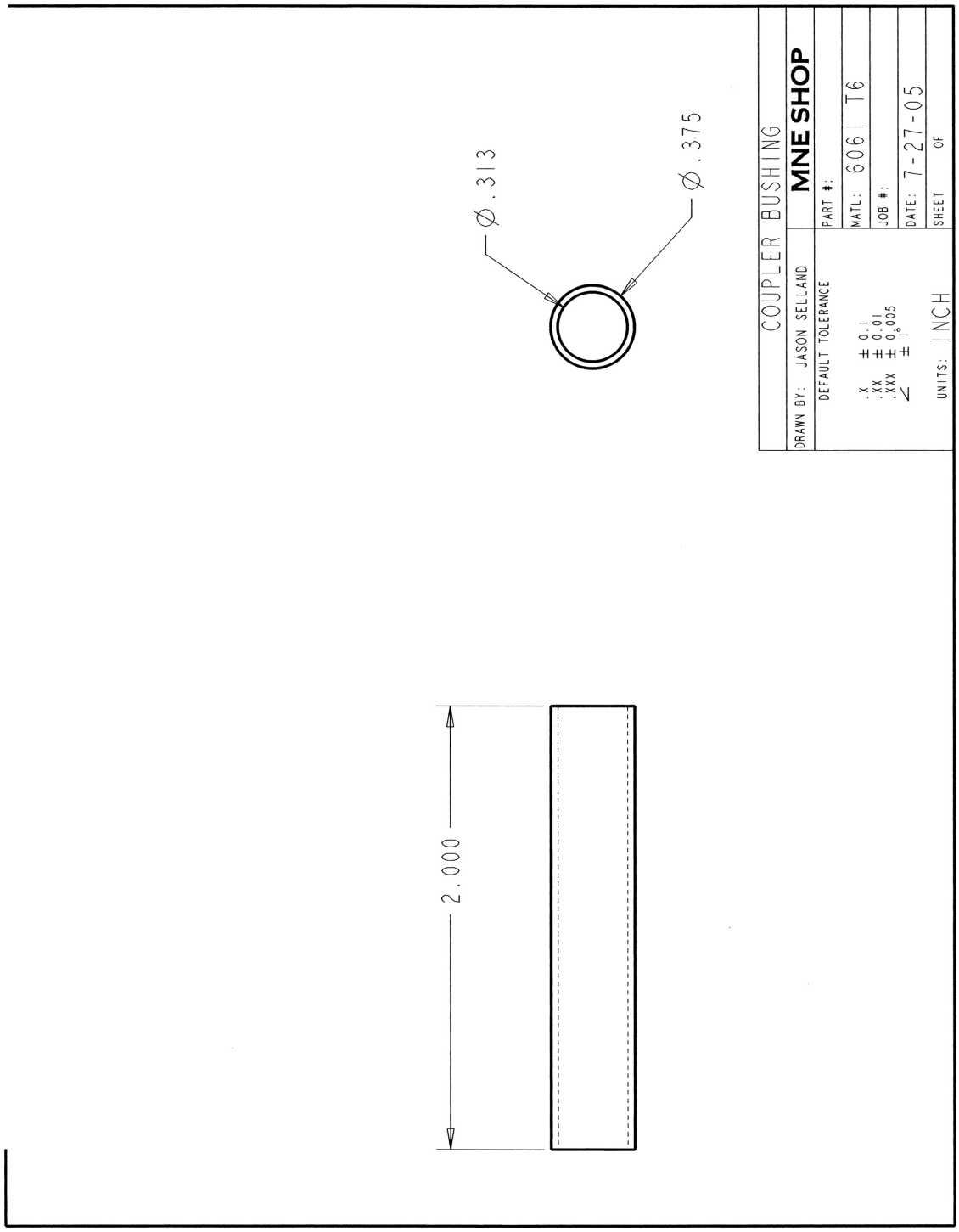


Figure C.18: Flux probe coupler bushing detail, page 8.

C.6 Drilling Rig

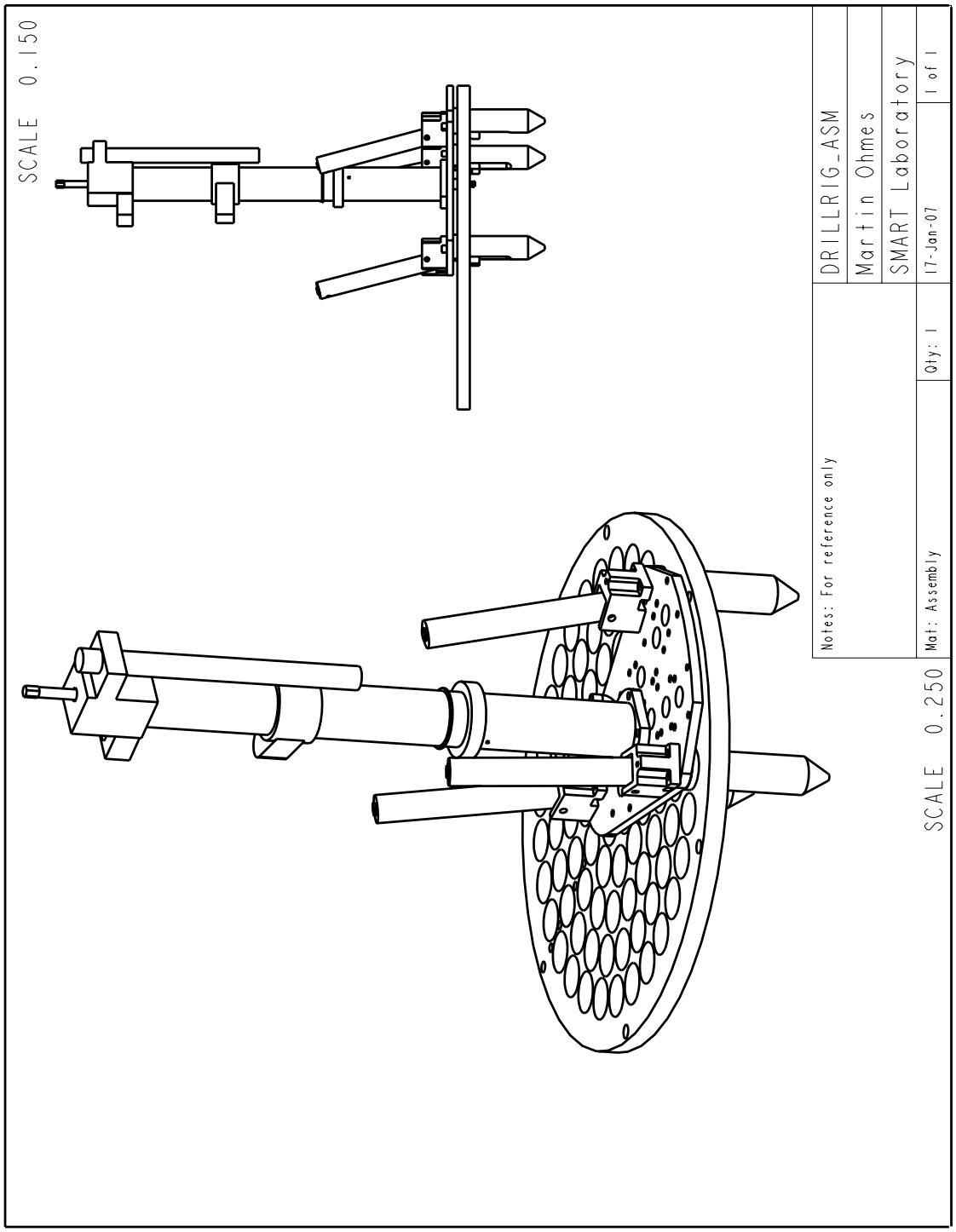


Figure C.19: Drilling rig assembly drawing.

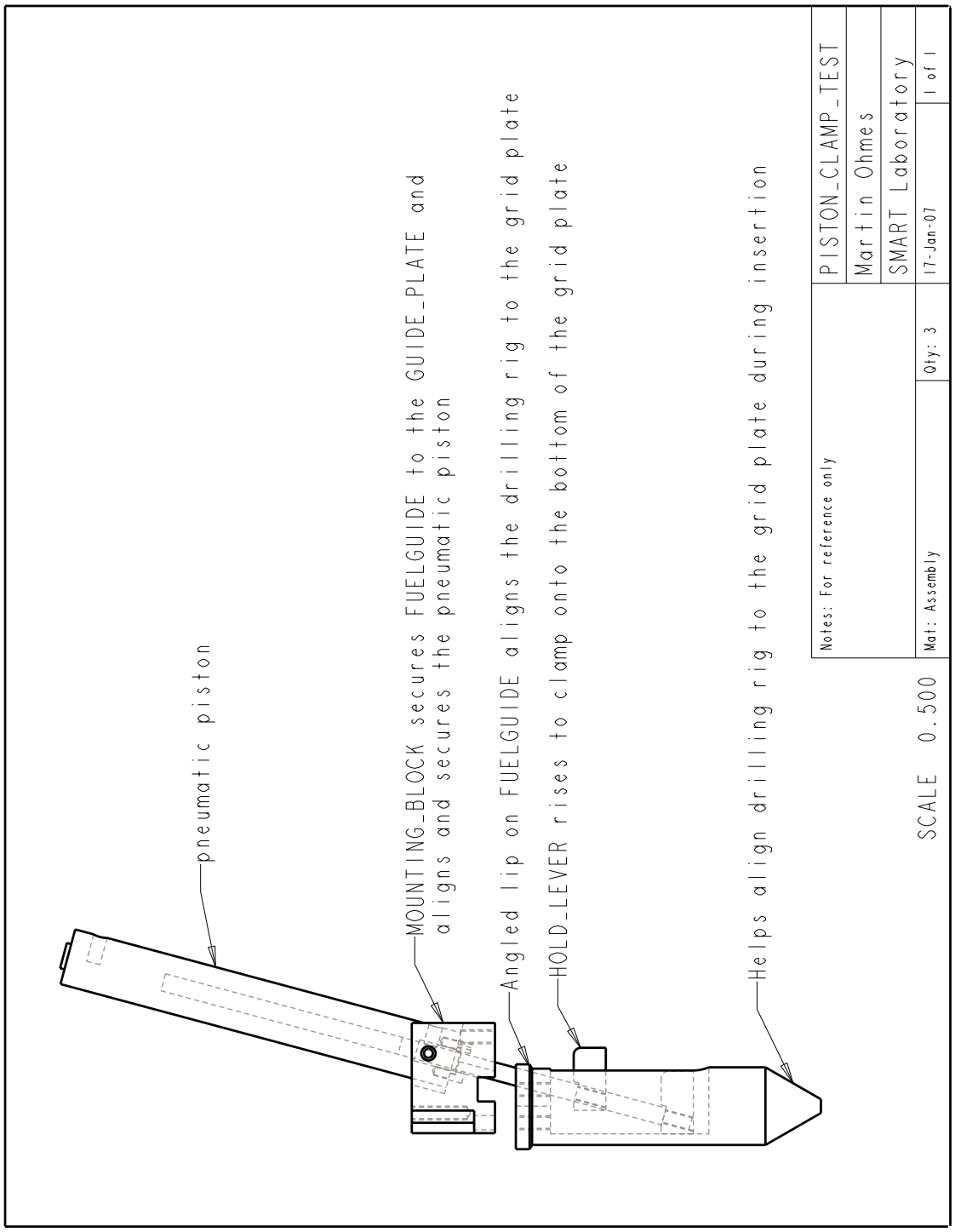


Figure C.20: Drilling rig piston clamp assembly drawing.

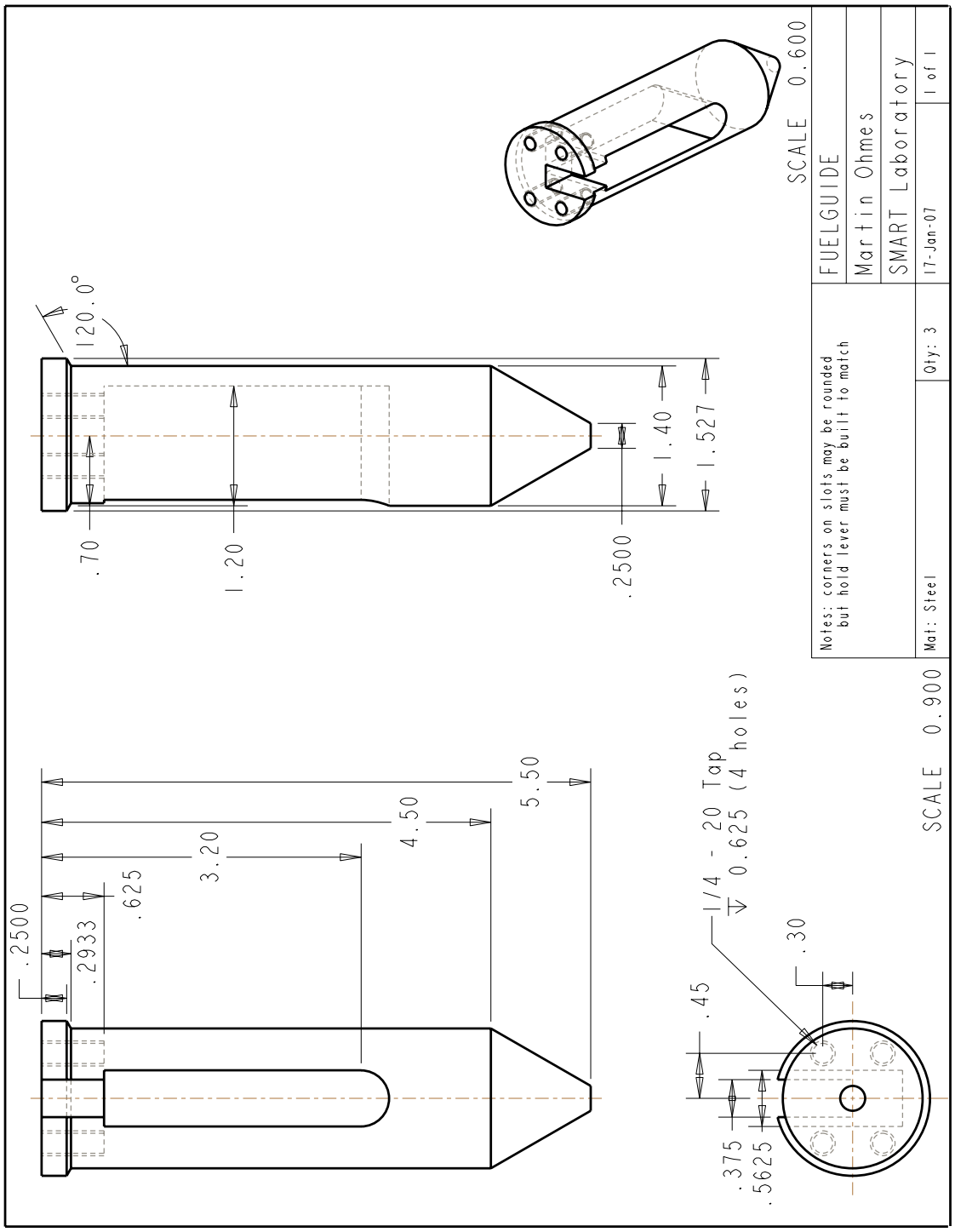


Figure C.21: Drilling rig fuel guide pin drawing.

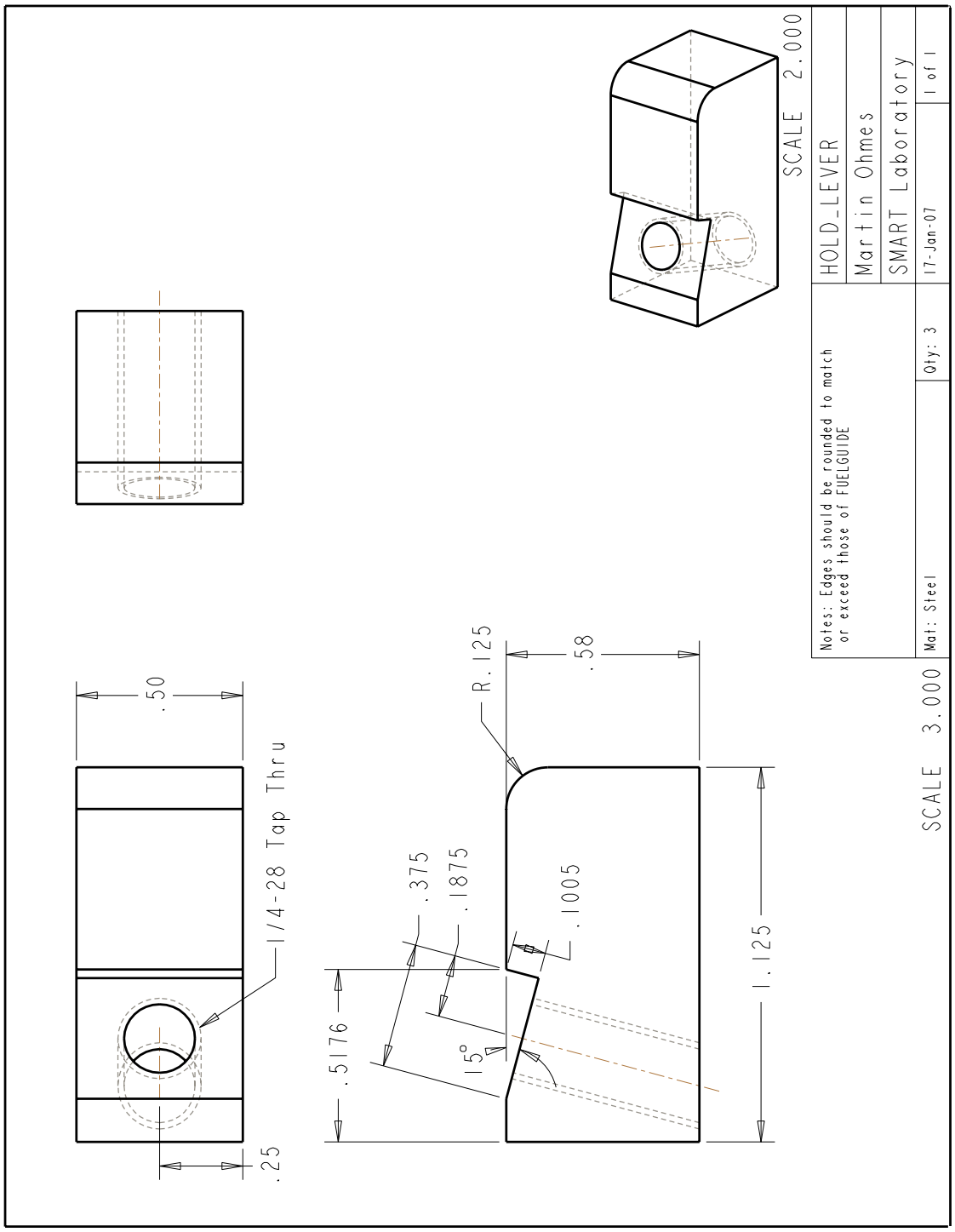


Figure C.22: Drilling rig hold lever drawing.

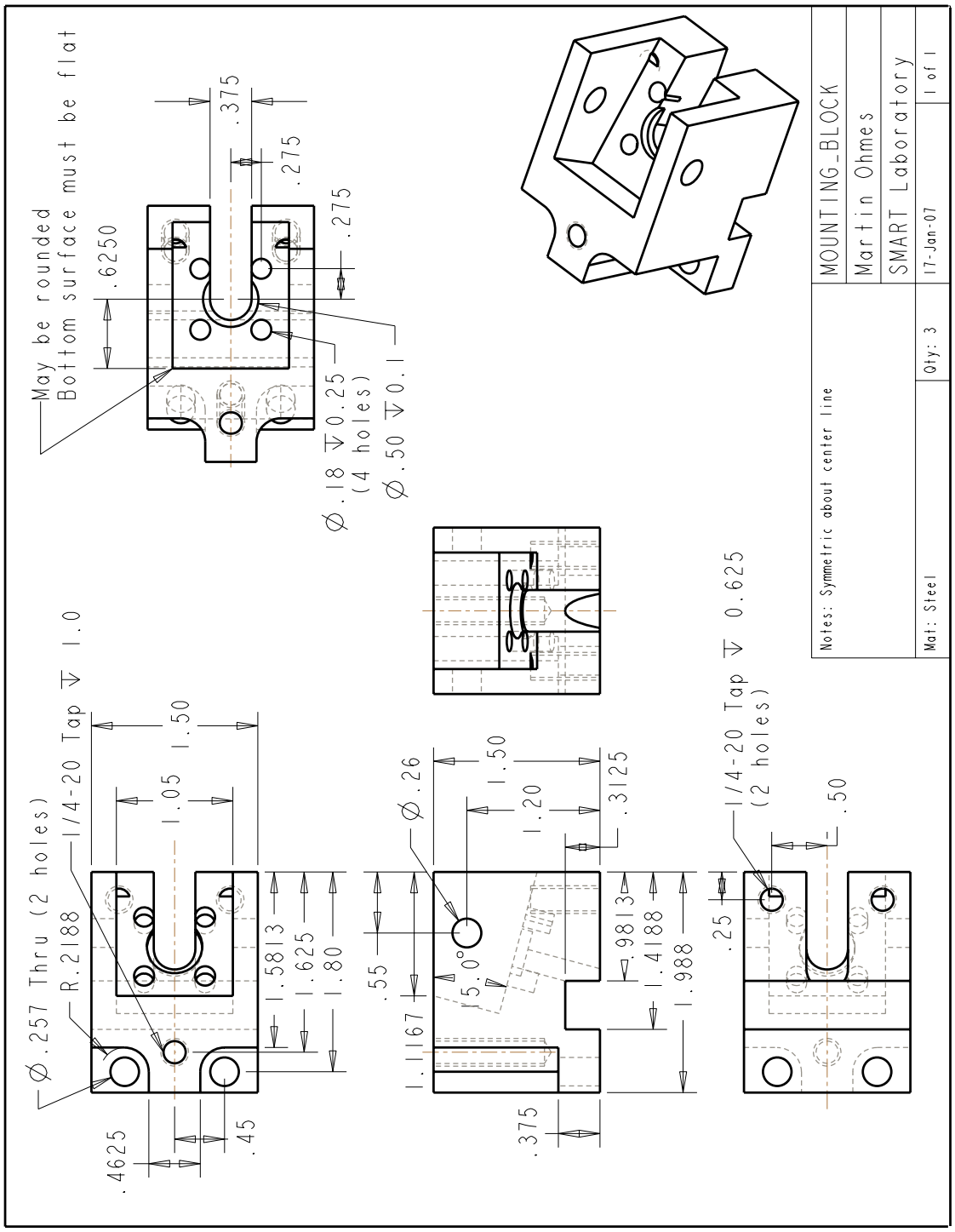


Figure C.23: Drilling rig piston mounting block drawing.

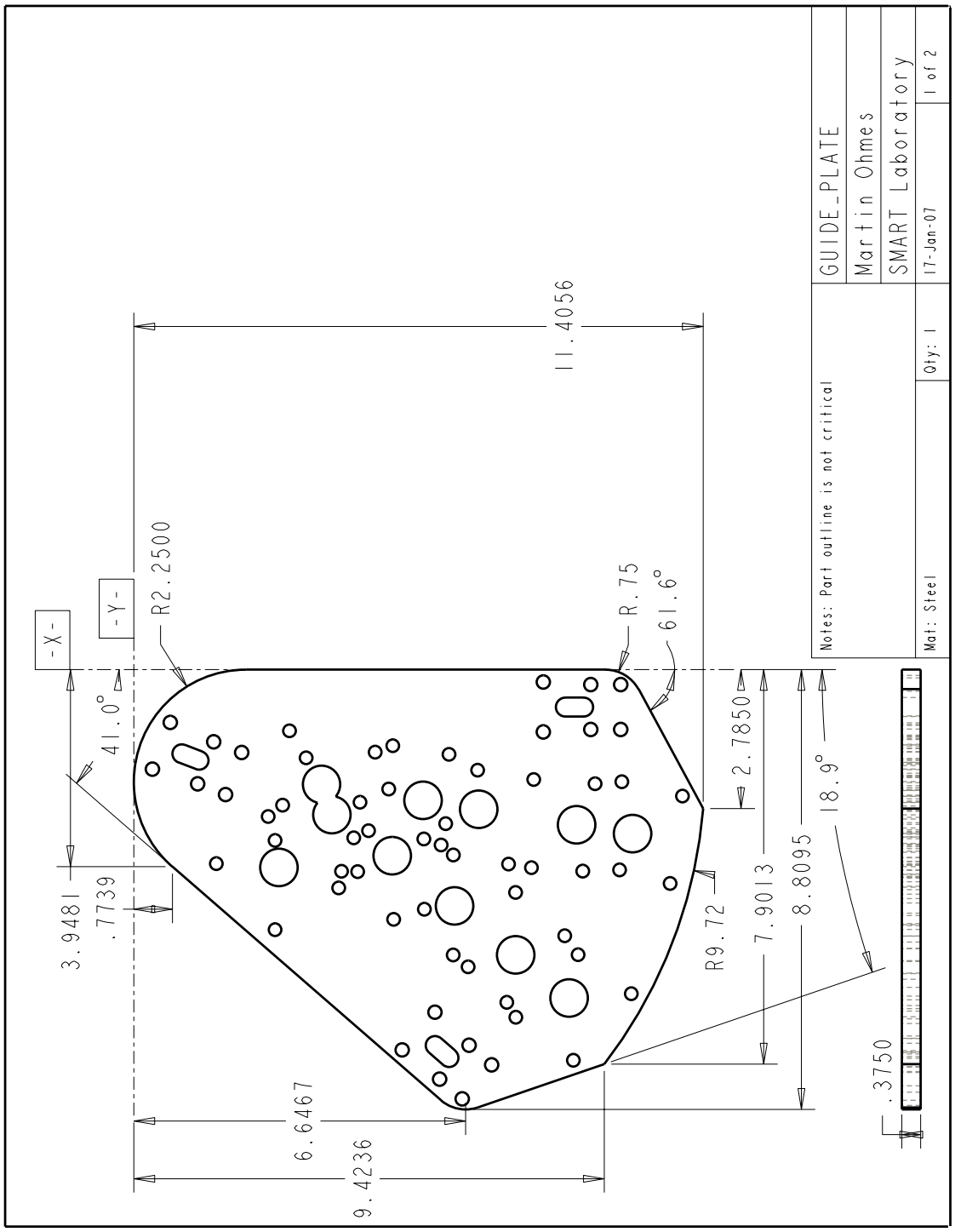


Figure C.24: Drilling rig guide plate drawing, page 1.

3 Slots - Ø3/8 - from hole centers

-X-	-Y-	-X-	-Y-
7.7725	6.3166	7.5265	6.0335
0.7500	9.0134	0.7500	8.6434
1.8179	1.3110	1.6835	0.9609

18 holes - 0.266 Thru
(1/4-20 close fit)

-X-	-Y-
8.5960	6.5779
8.2023	6.1251
7.6168	5.3753
7.9167	7.1684
7.5231	6.7156
6.8621	6.0314
1.2000	9.7534
1.2000	9.1534
1.2500	8.2034
0.3000	9.7534
0.3000	9.1534
0.2500	8.2034
2.5024	1.8382
2.2873	1.2781
1.9934	0.3733
1.6622	2.1608
1.4471	1.6007
1.0599	0.7317

11 holes - Ø3/4 thru

-X-	-Y-
6.5773	8.7186
5.7133	7.6486
4.7329	6.4259
3.7290	5.1756
2.9070	3.9683
3.9628	2.9061
2.2982	3.7620
2.6200	5.7937
2.7968	6.9098
3.1073	8.8703
3.2849	9.9913

42 holes - 1/4-20 Tap Thru

-X-	-Y-	-X-	-Y-	-X-	-Y-
7.8243	8.8058	2.7151	2.9815	5.3304	8.6314
6.9633	7.6507	3.3697	4.4058	5.7112	8.8986
6.6645	7.4717	3.2260	4.7004	6.4901	9.9656
5.9535	6.6956	2.6575	4.5318	4.2832	10.7436
5.7154	6.3986	2.3912	5.1069	2.5326	10.9896
4.8004	5.8194	3.3921	5.8106	2.2459	9.7761
5.0026	5.2053	3.5124	6.1563	2.2866	9.2391
4.0456	4.4841	3.0852	6.2470	2.2015	8.0089
4.0386	4.1596	3.7133	6.3997	2.0140	6.8869
4.3728	4.1041	3.8960	7.5050	1.6976	6.3145
5.2106	2.8306	4.4633	7.6465	1.5268	5.1877
3.8870	1.6525	3.9687	7.9645	1.6544	4.8334
3.4228	2.8297	4.0371	8.9930	1.7684	3.4525
2.9420	2.6905	4.0131	9.7317	1.2268	3.1182

GUIDE_PLATE	
Martin Ohmes	
SMART Laboratory	
17-Jan-07	2 of 2

Figure C.25: Drilling rig guide plate dimensions table, page 2.

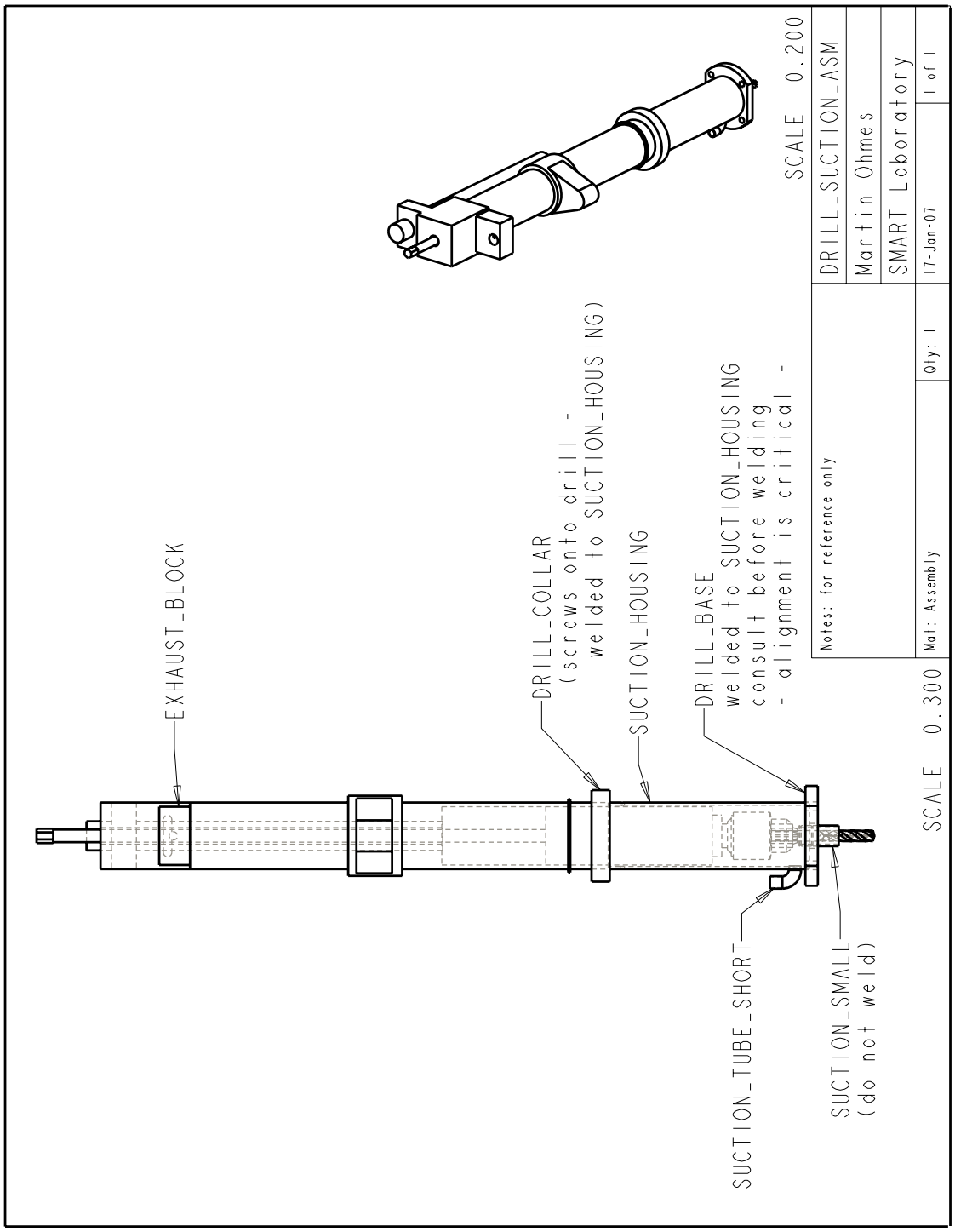


Figure C.26: Drilling rig drill suction assembly drawing.

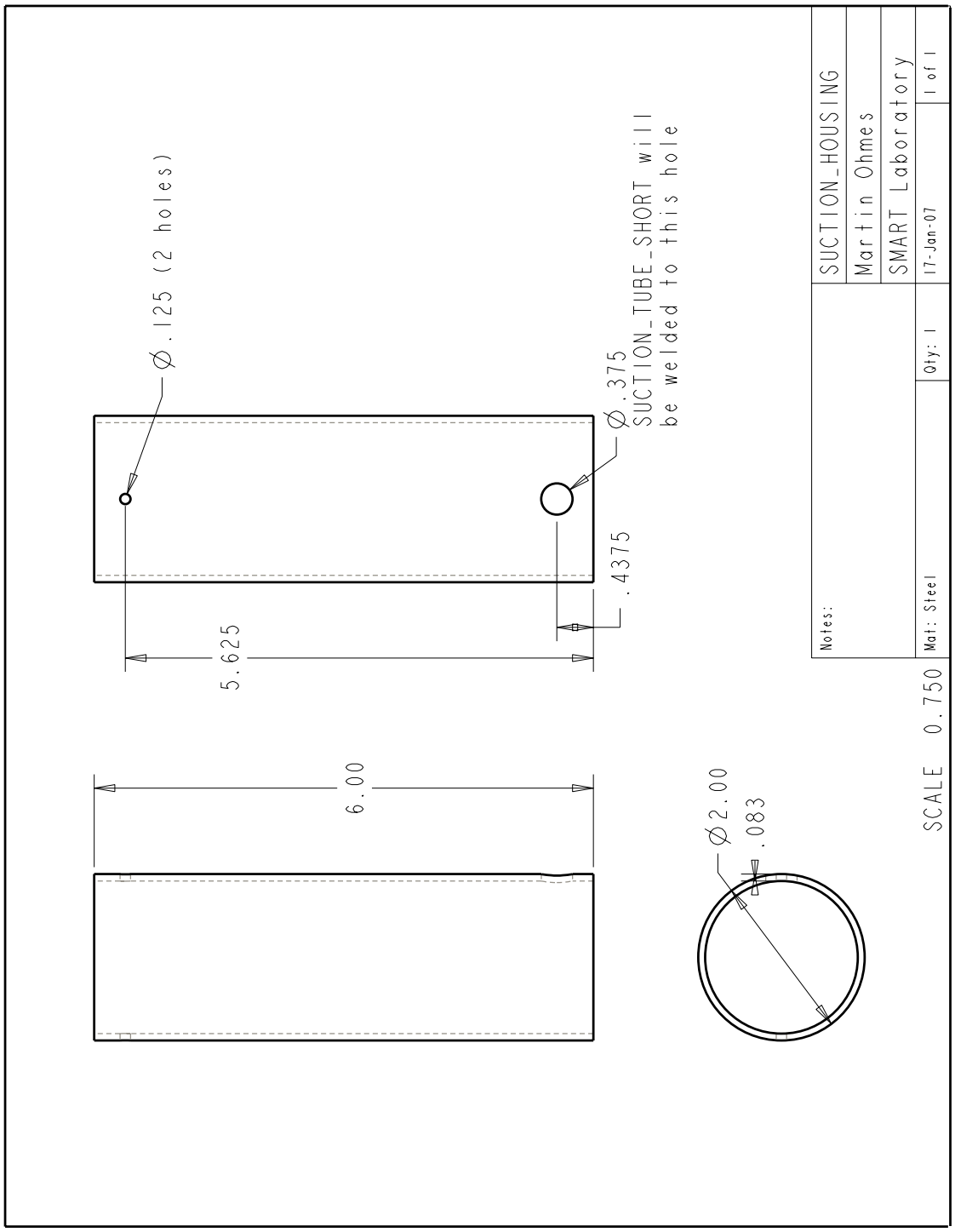


Figure C.27: Drilling rig suction housing drawing.

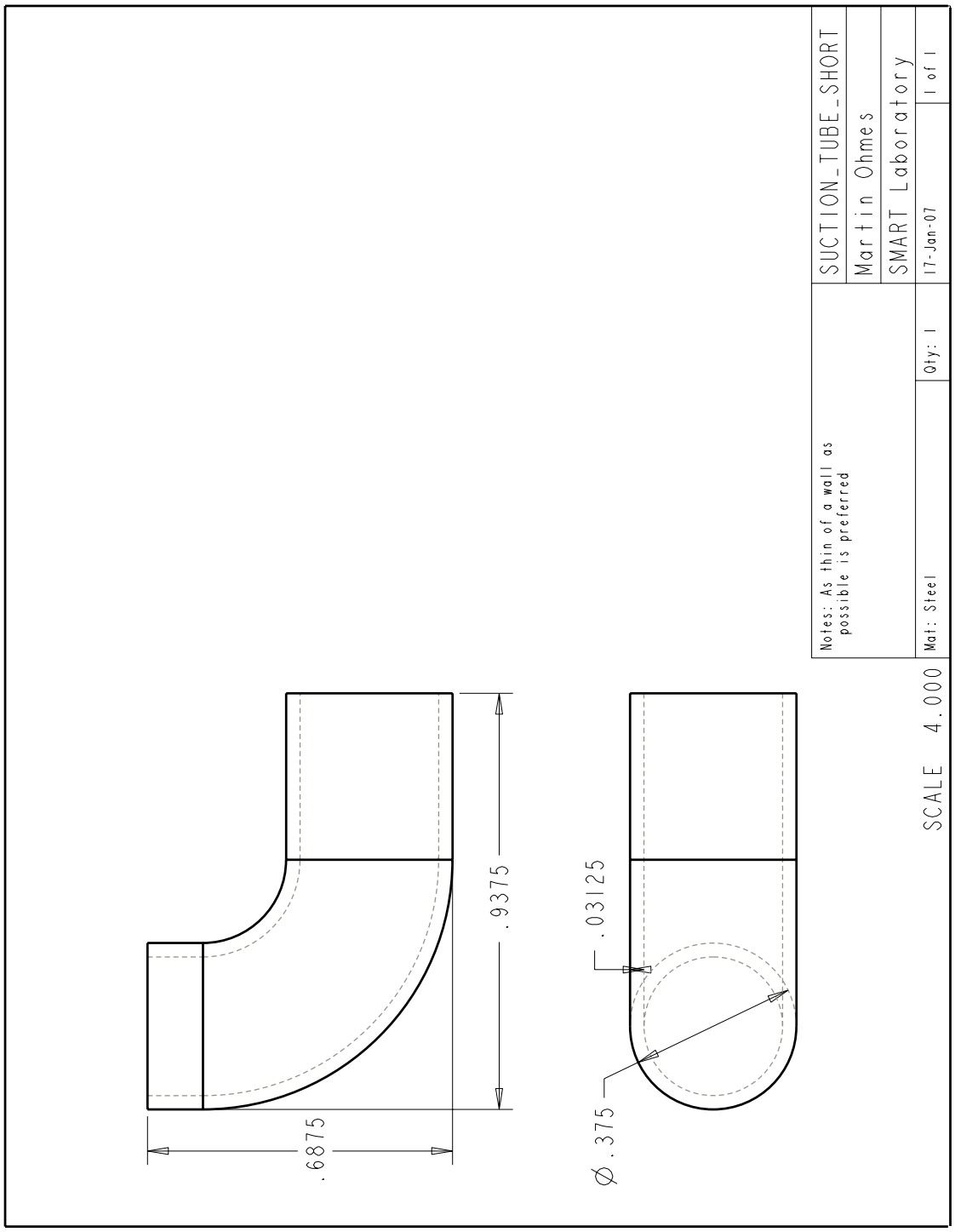


Figure C.28: Drilling rig suction tube short elbow drawing.

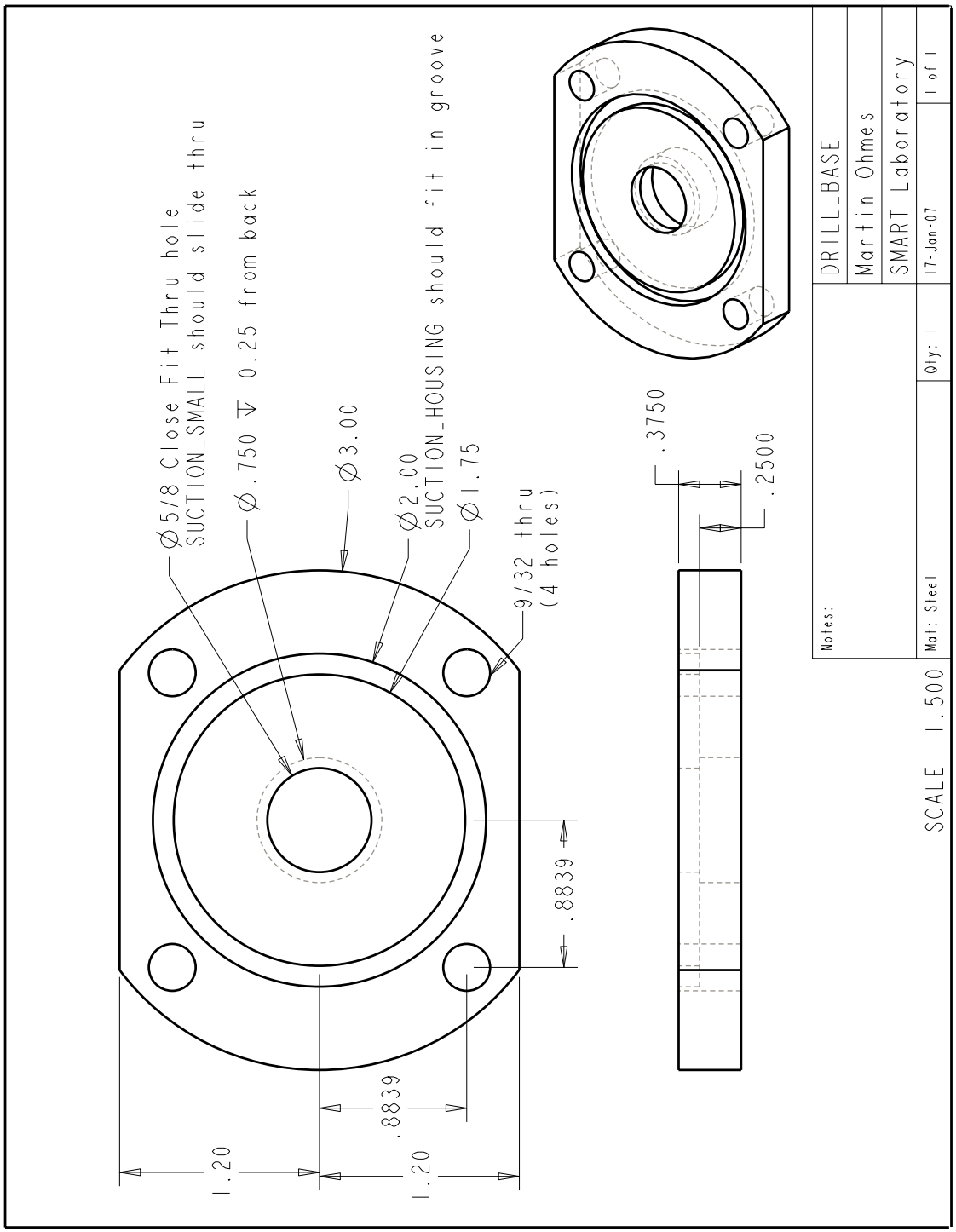


Figure C.29: Drilling rig drill base drawing.

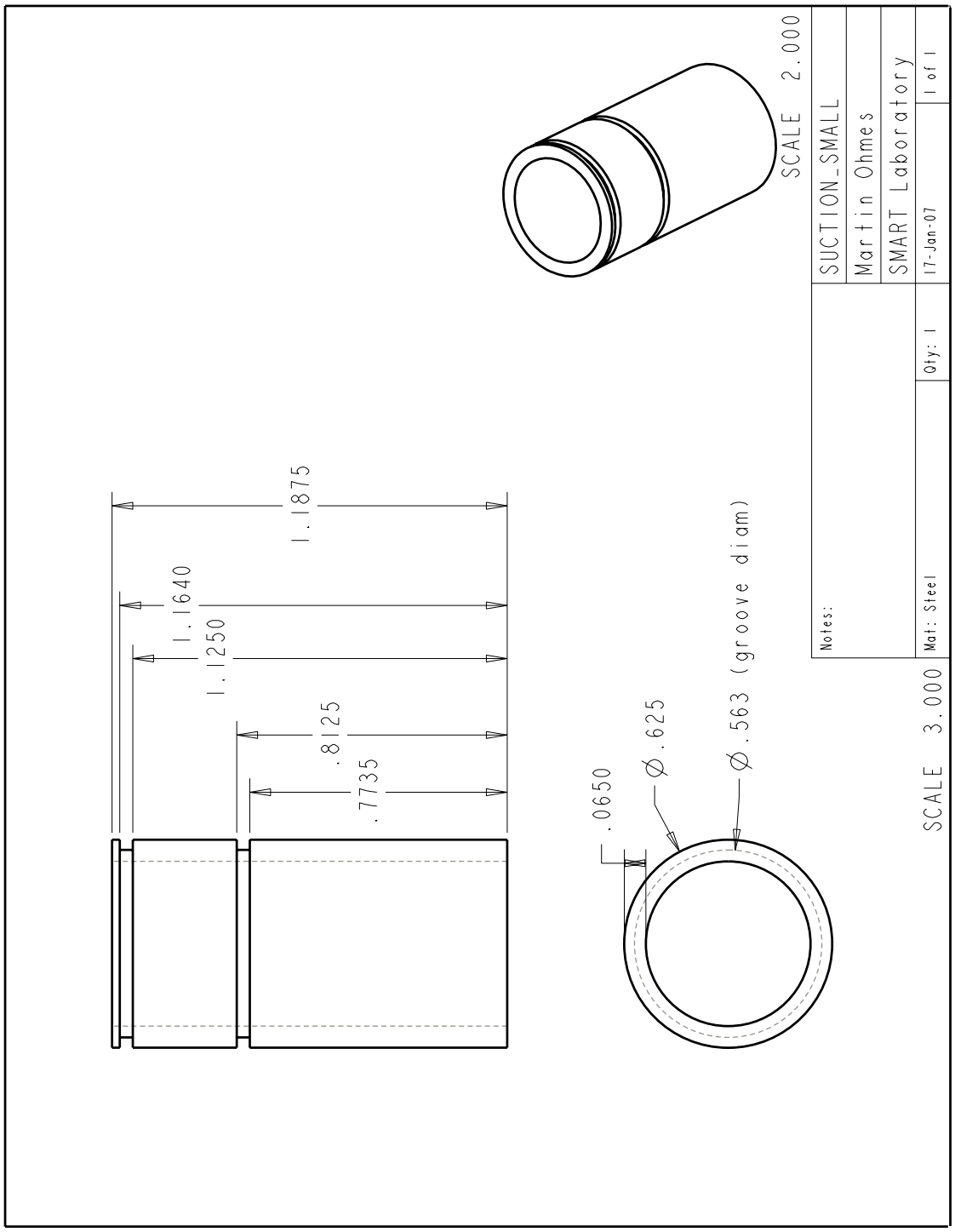


Figure C.30: Drilling rig floating suction tube drawing.

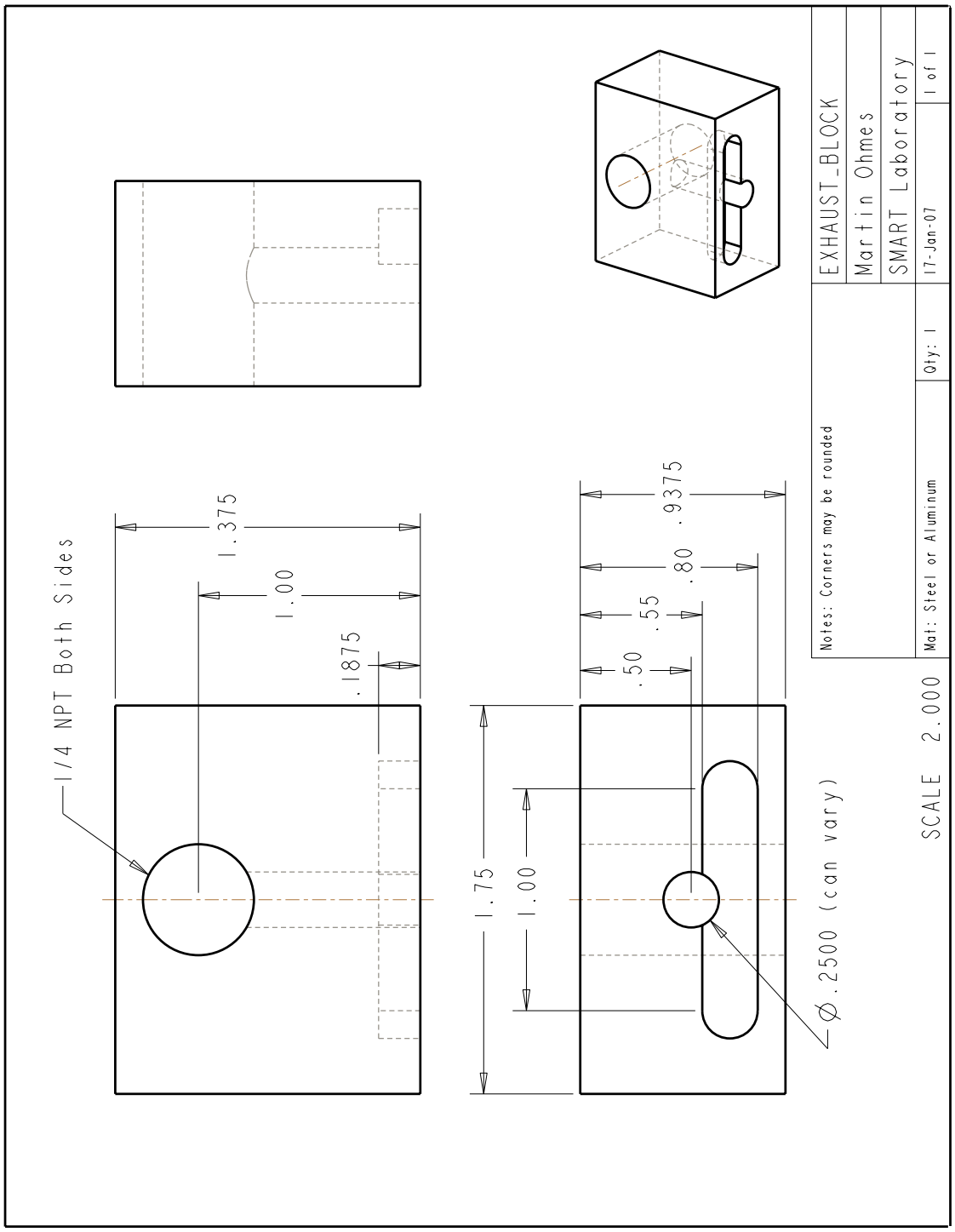


Figure C.31: Drilling rig drill exhaust capture block drawing.

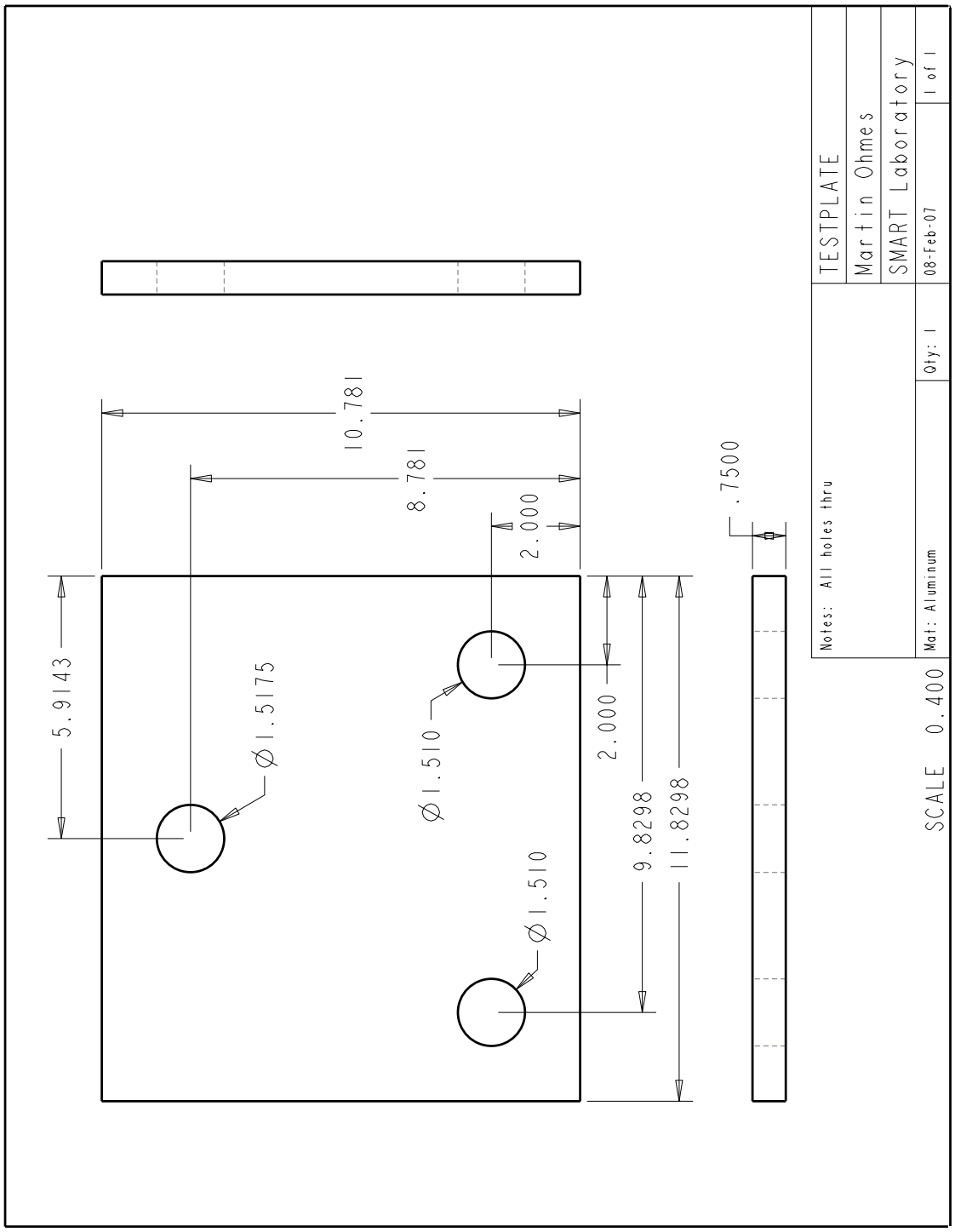


Figure C.32: Drilling rig test plate (mock reactor grid plate) drawing.

C.7 Reactor Deck Plate

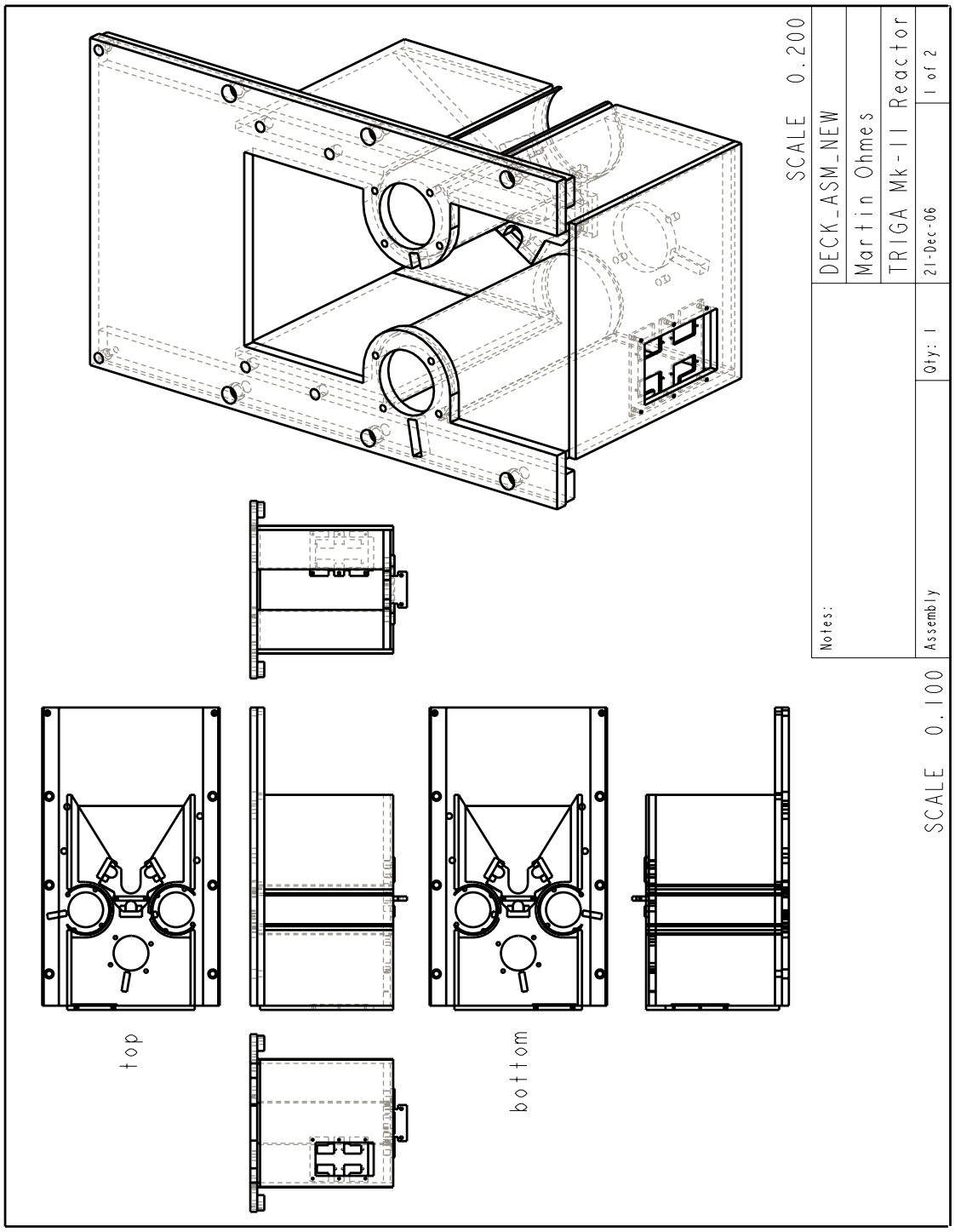


Figure C.33: Reactor deck plate assembly drawing, page 1.

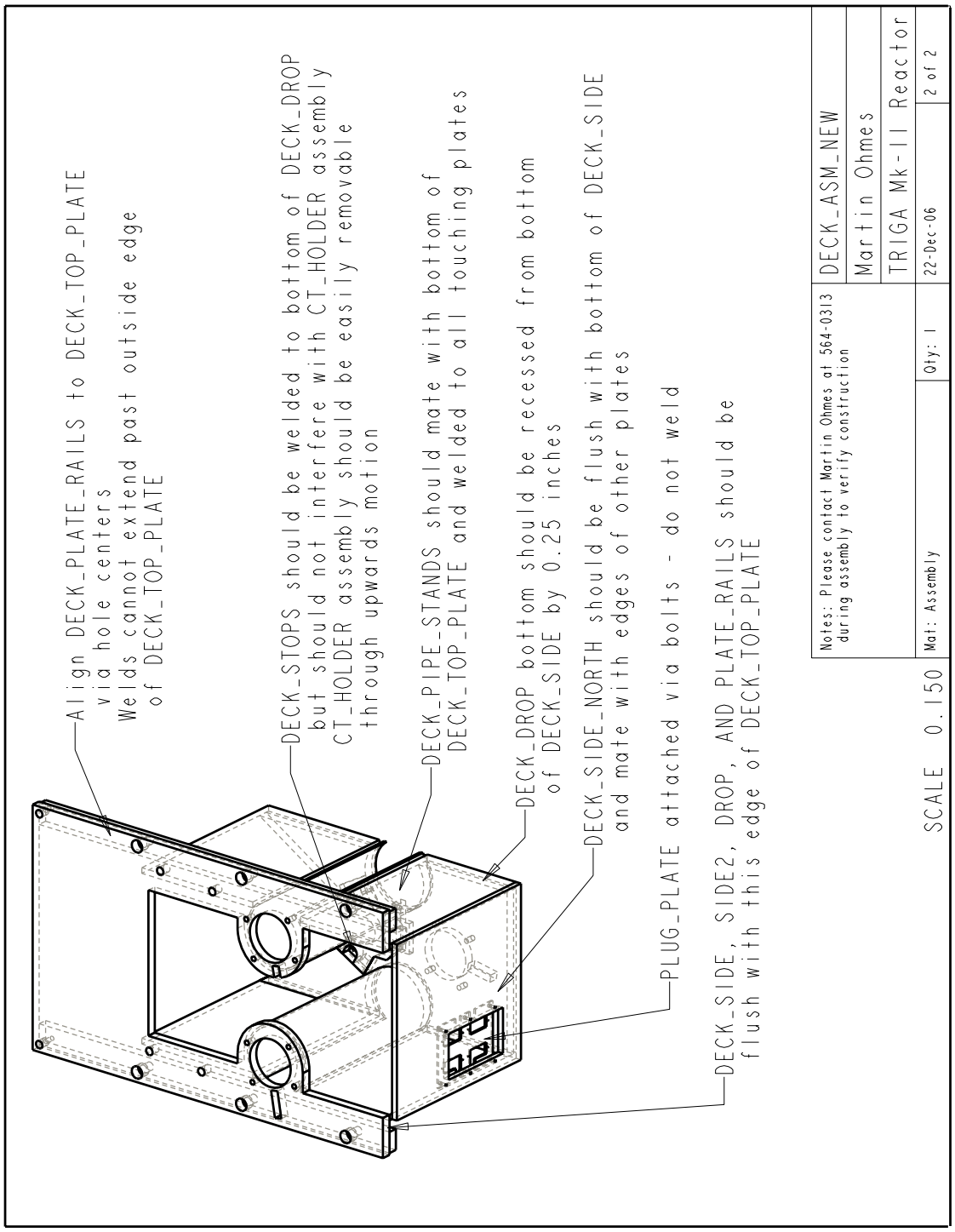


Figure C.34: Reactor deck plate assembly drawing, page 2.

C.8 Control Rod Guide Tube Tool

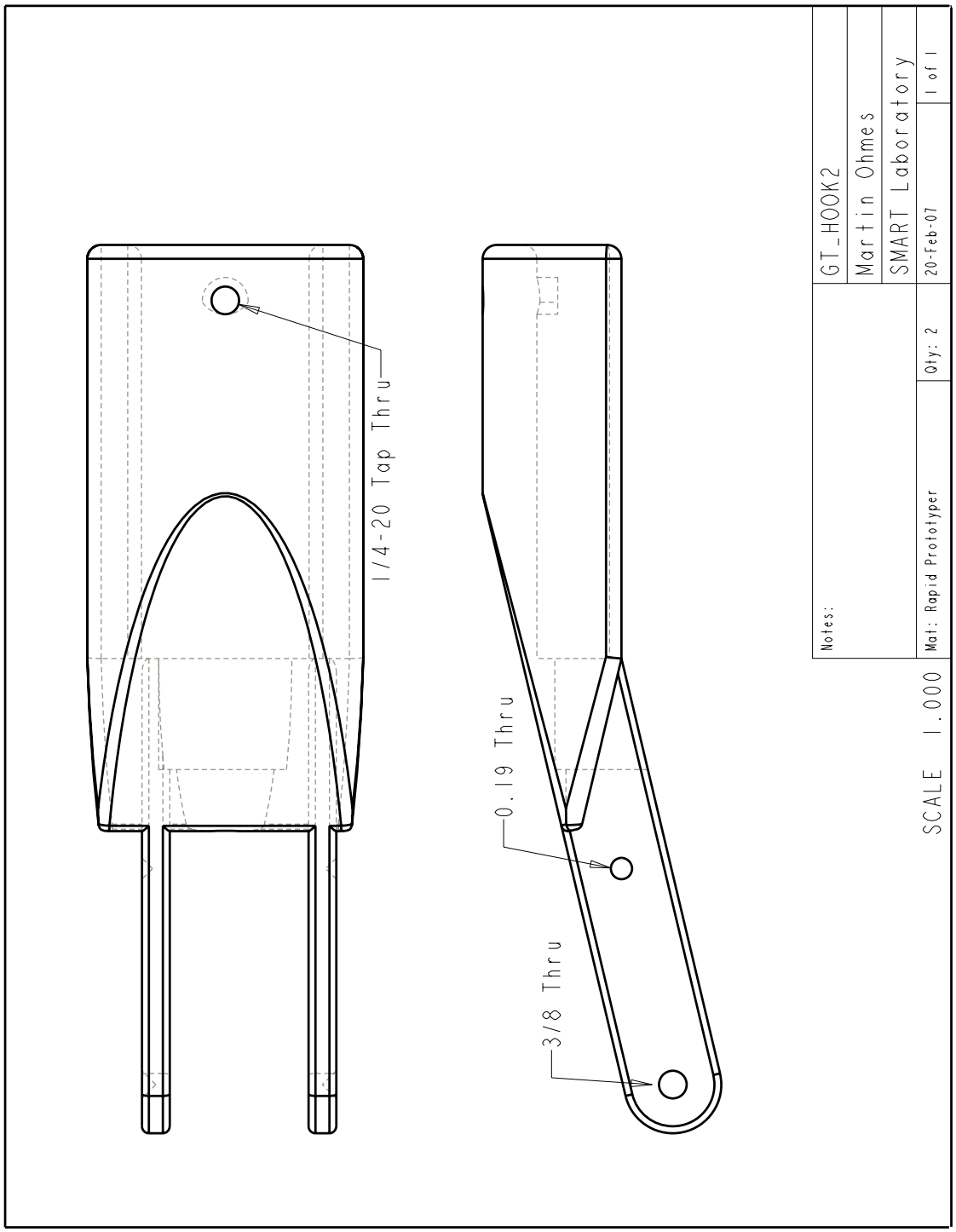


Figure C.35: Control rod guide tube tool drawing, page 1.

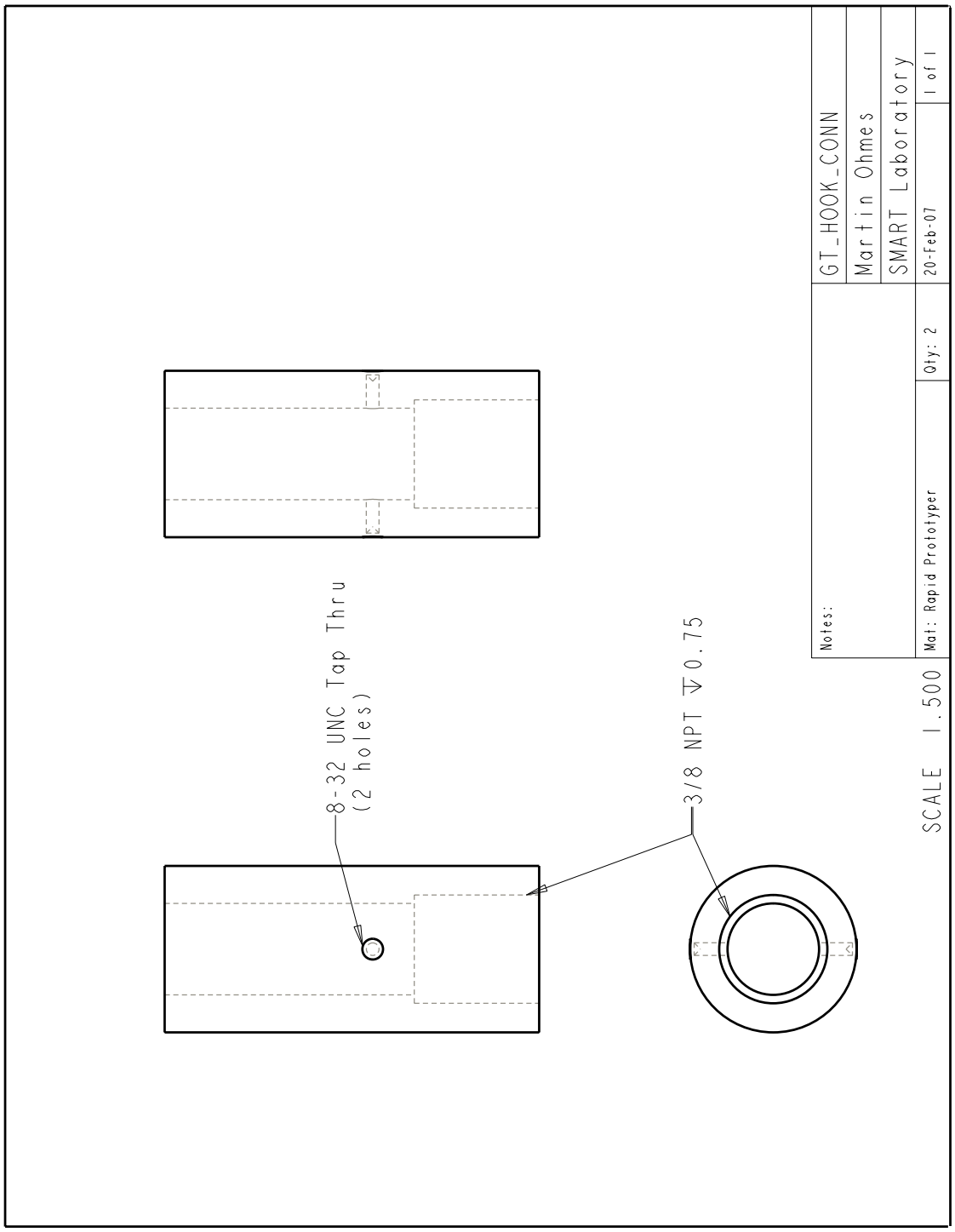


Figure C.36: Control rod guide tube tool drawing, page 2.

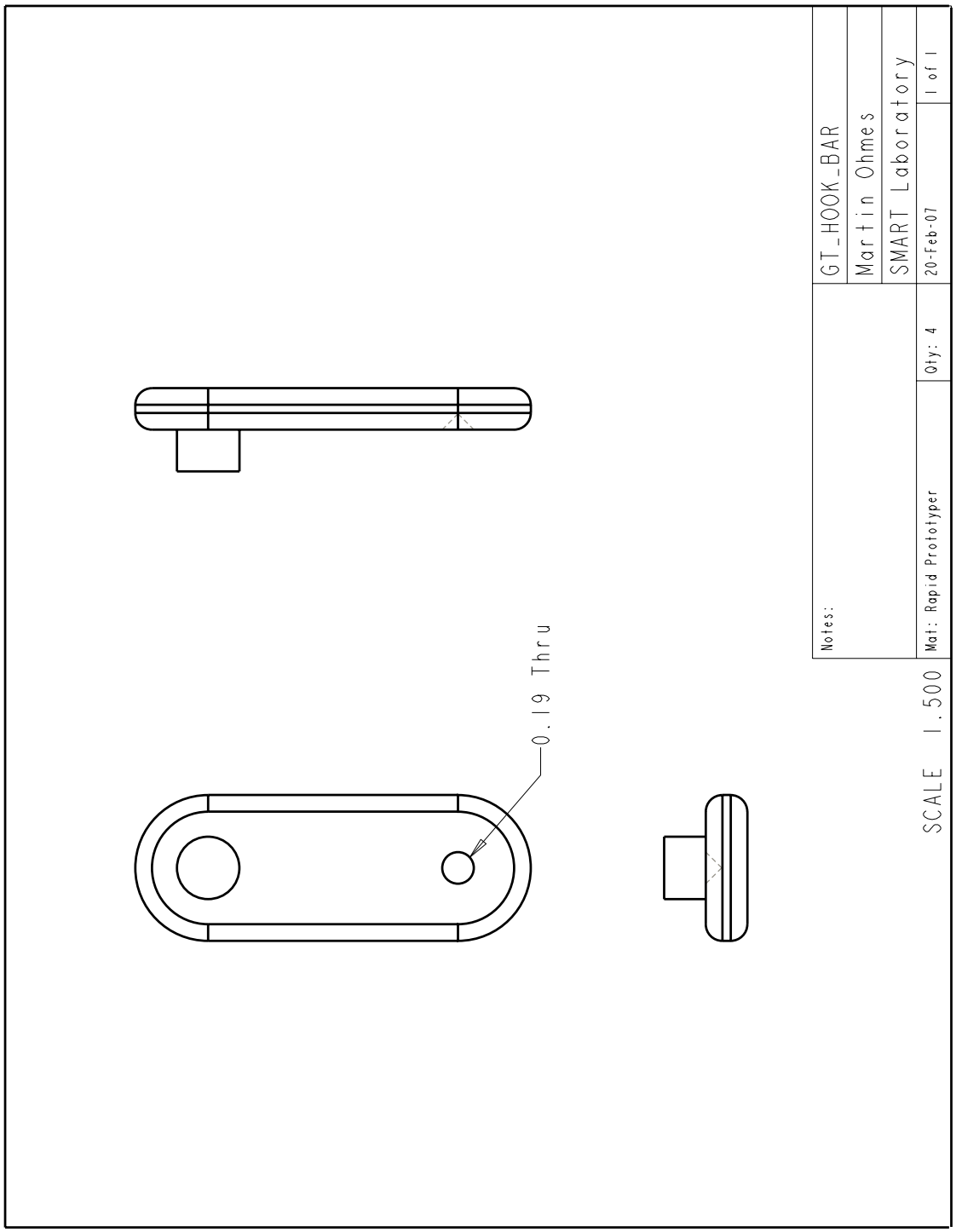


Figure C.37: Control rod guide tube tool drawing, page 3.

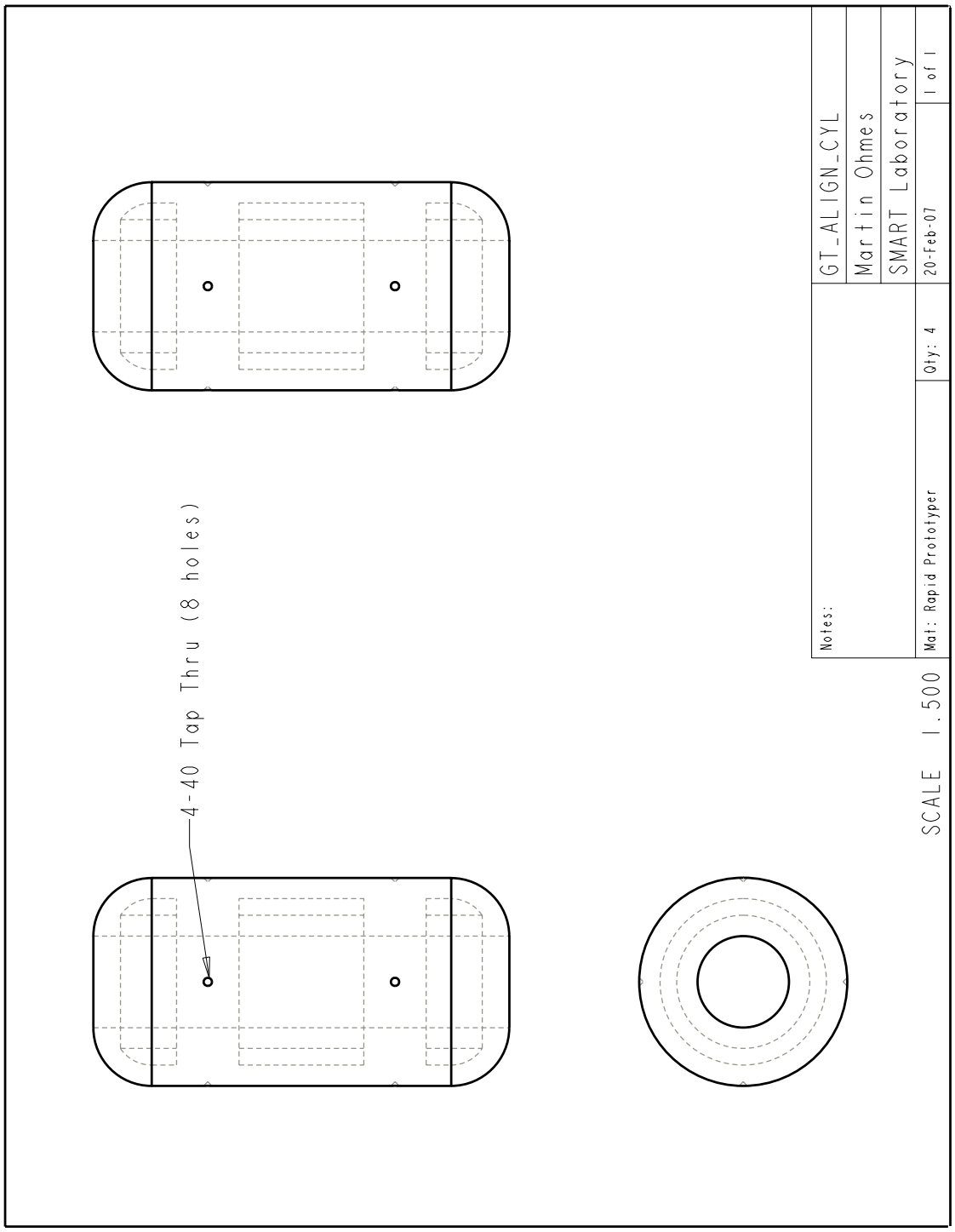


Figure C.38: Control rod guide tube tool drawing, page 4.

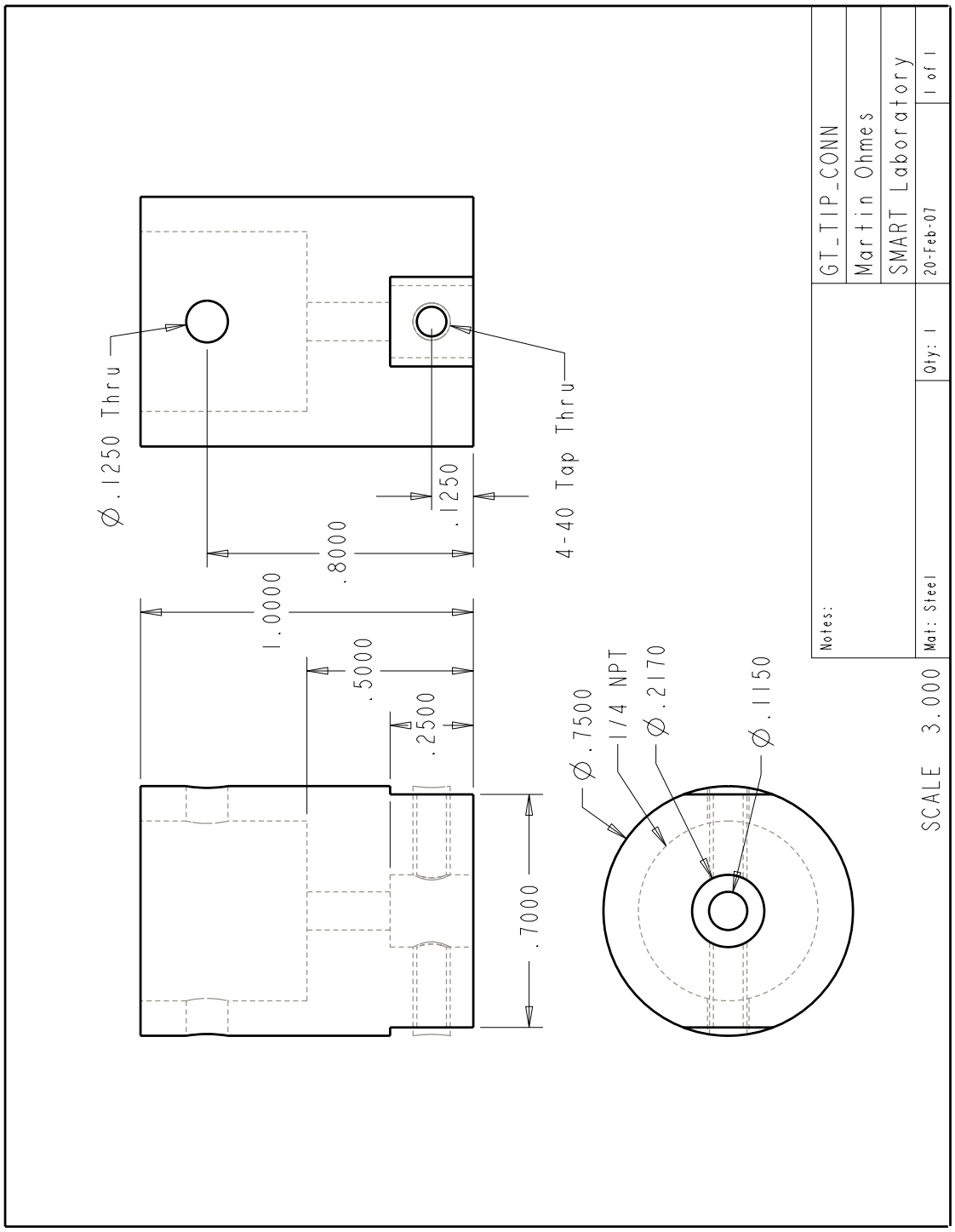


Figure C.39: Control rod guide tube tool drawing, page 5.

Appendix D

Electronic's Schematics and Layouts

Schematics and layouts were made using EAGLE (Easily Applicable Graphical Layout Editor) Professional Edition version 4.16. A custom shorthand notation is used throughout these schematics to help quickly identify part types. For capacitors a value of 1uC would indicate a 1 μF ceramic capacitor, while a value of 1pT would represent a 1 pF tantalum capacitor. Likewise, a suffix of P indicates a poly capacitor and E represents an electrolytic capacitor. Resistors also have a shorthand notation of 2.7k.1 which indicates a 2.7 kΩ 0.1% resistor.

D.1 MPFD^3-T Daughter Board Schematic and Layout

For descriptions of the circuits see Section §5.1.

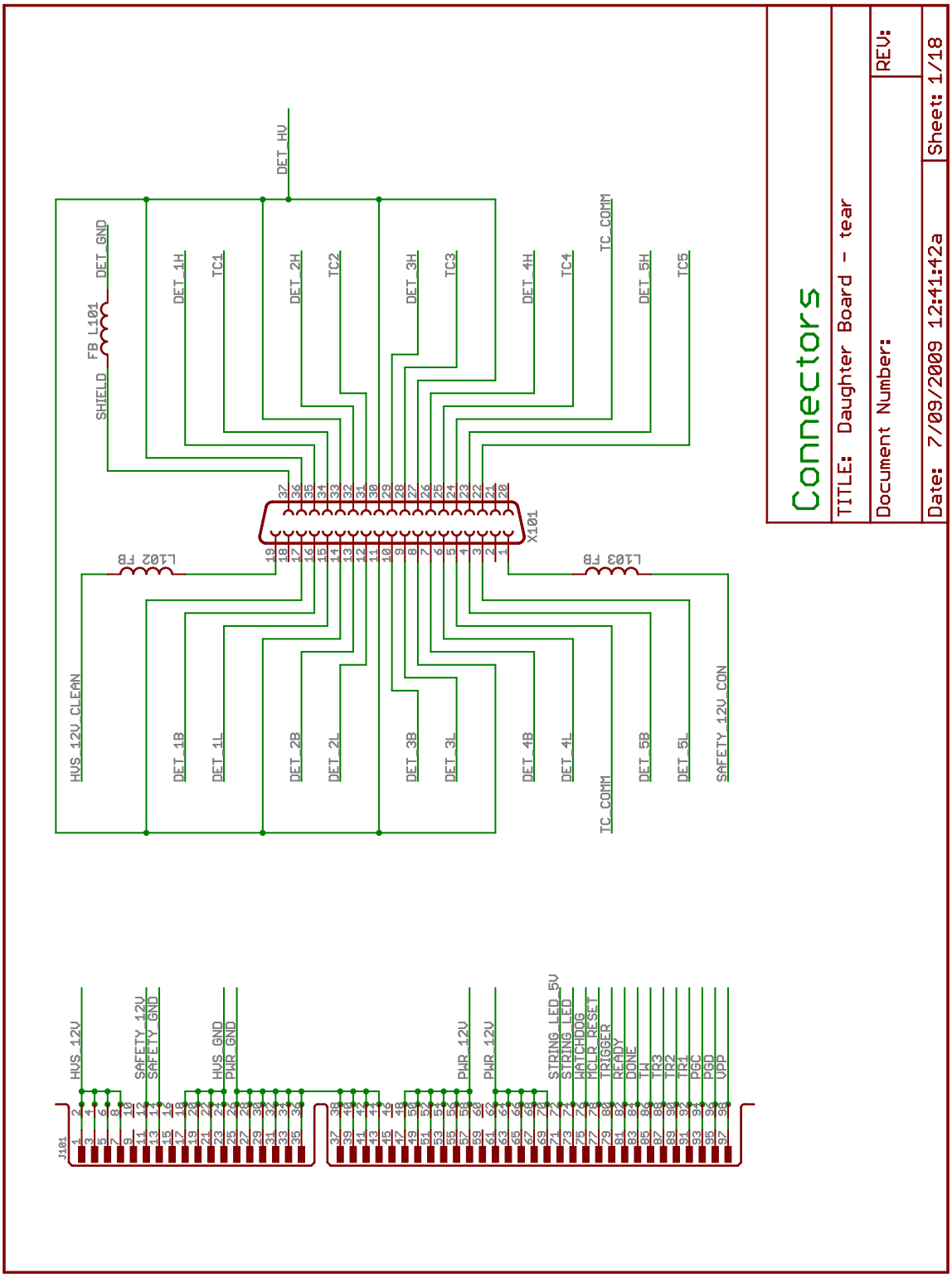


Figure D.1: Schematic for the daughter boards, page 1.

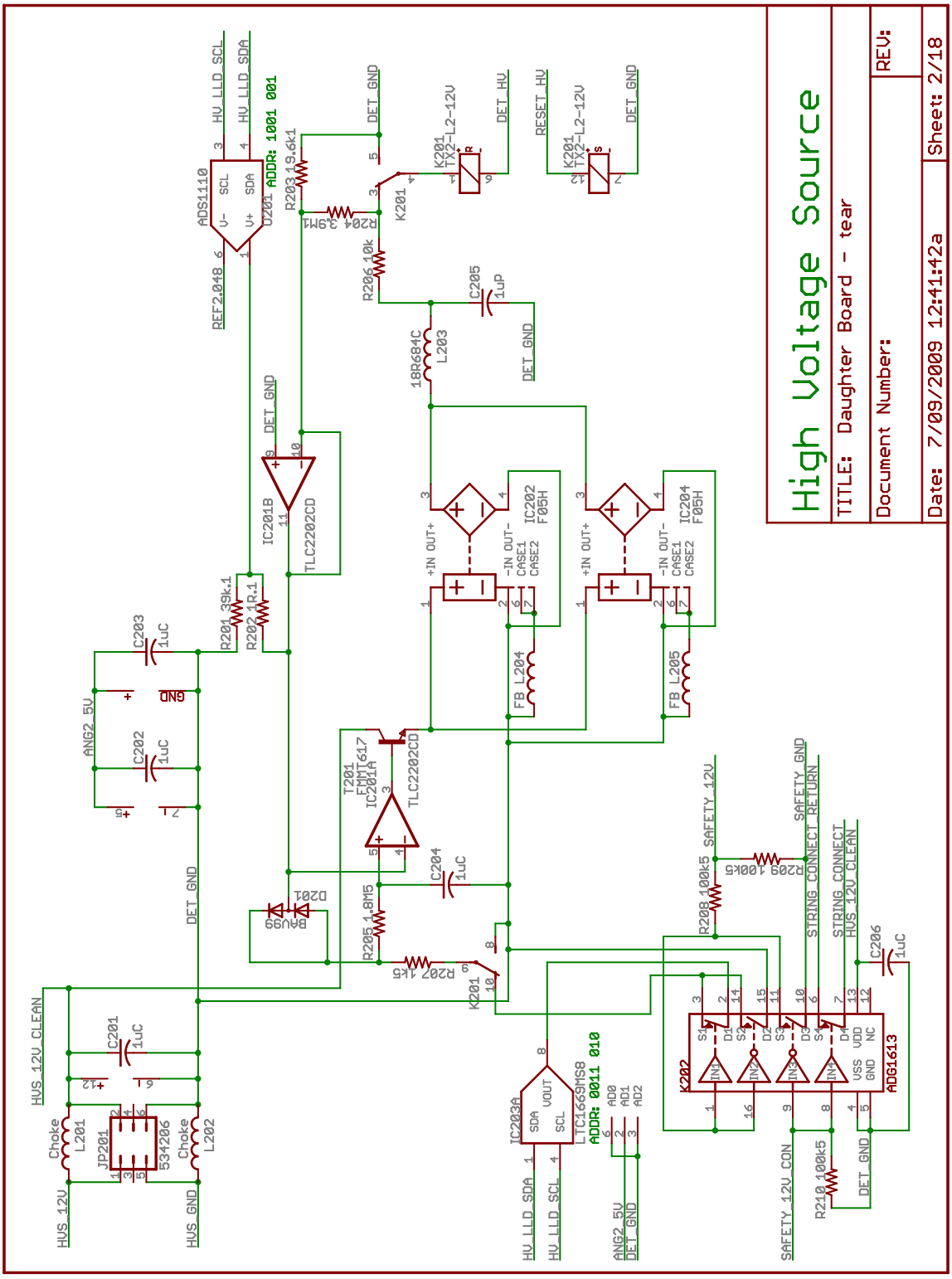


Figure D.2: Schematic for the daughter boards, page 2.

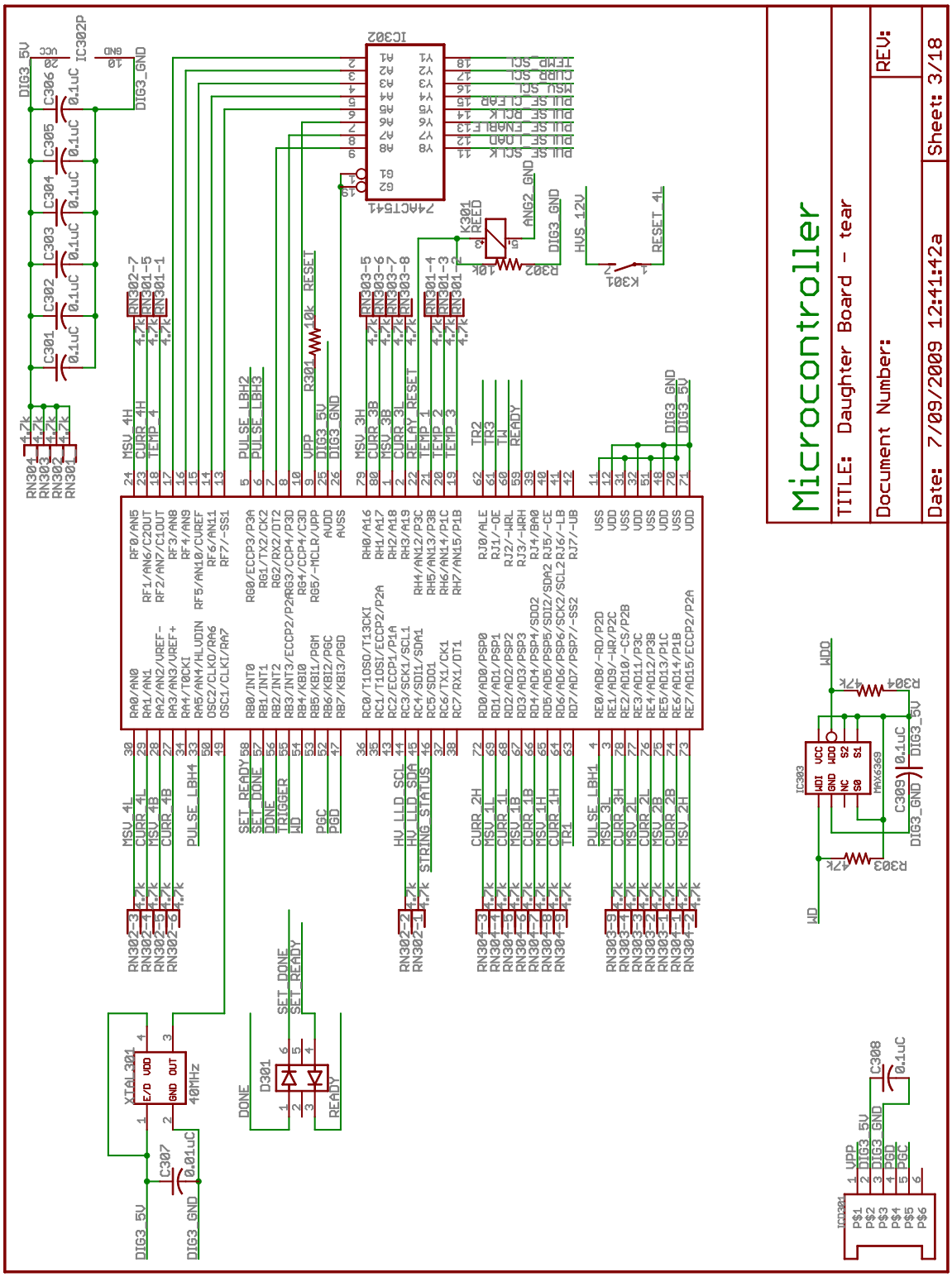


Figure D.3: Schematic for the daughter boards, page 3.

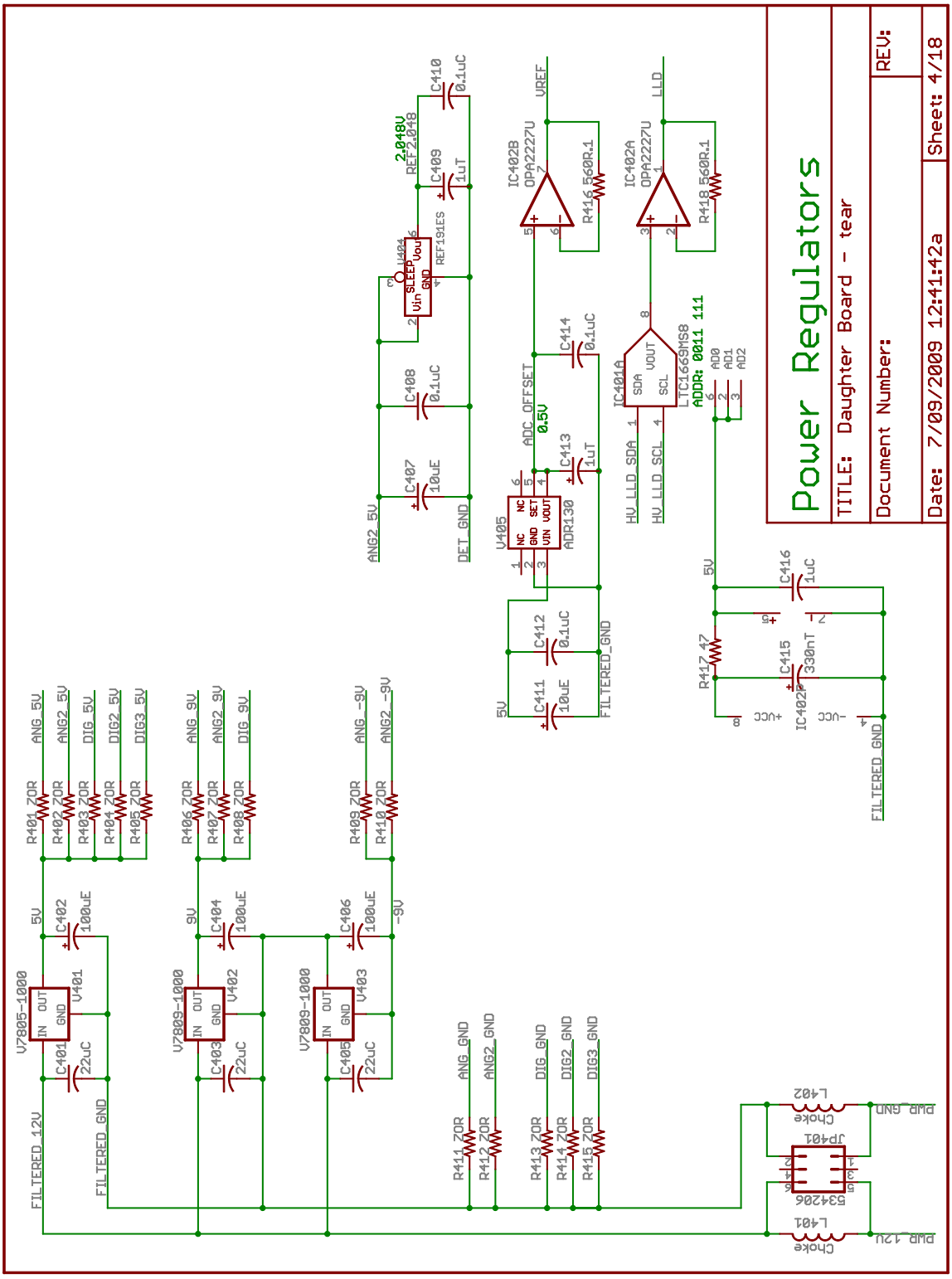


Figure D.4: Schematic for the daughter boards, page 4.

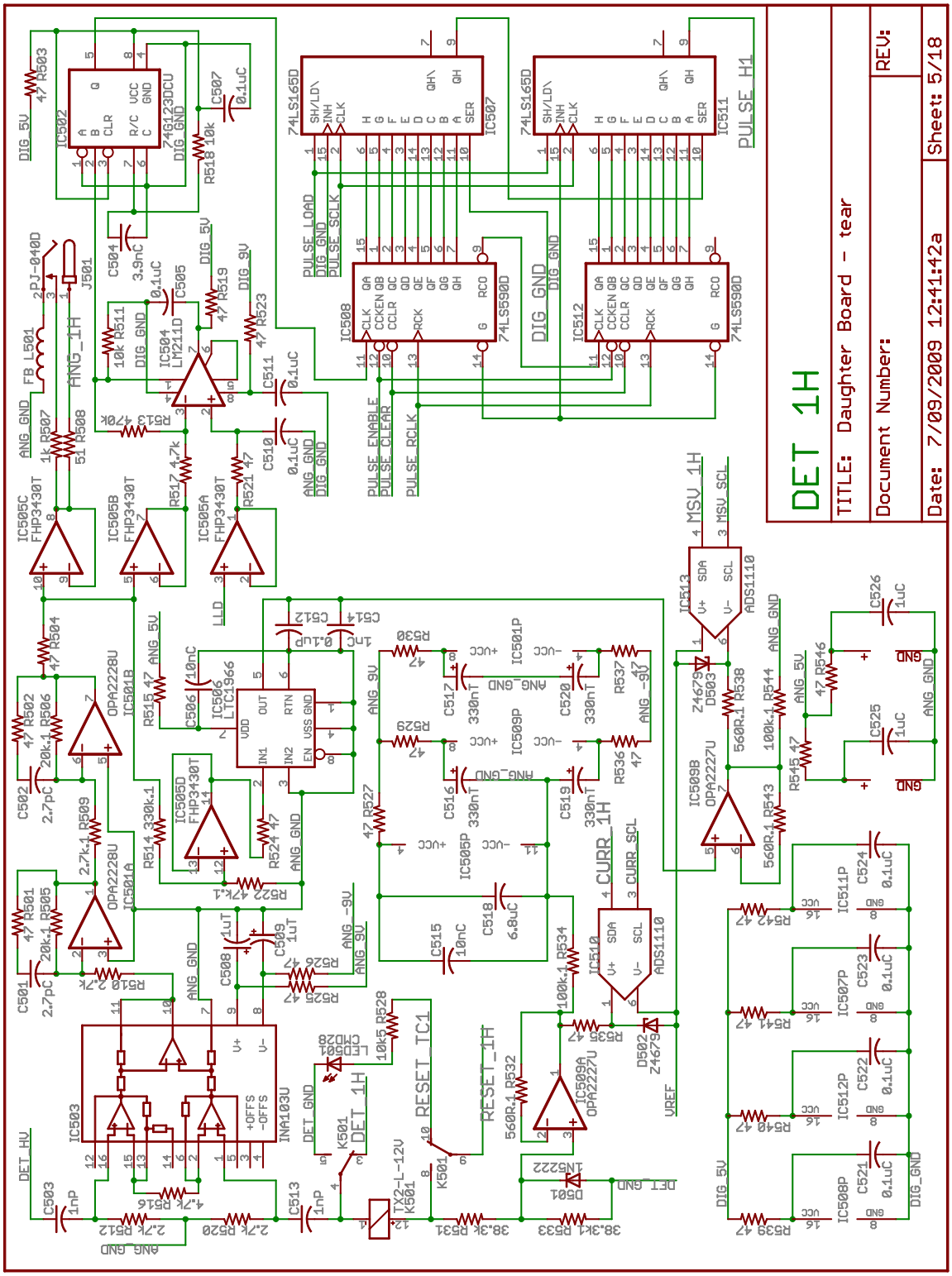


Figure D.5: Schematic for the daughter boards, page 5.

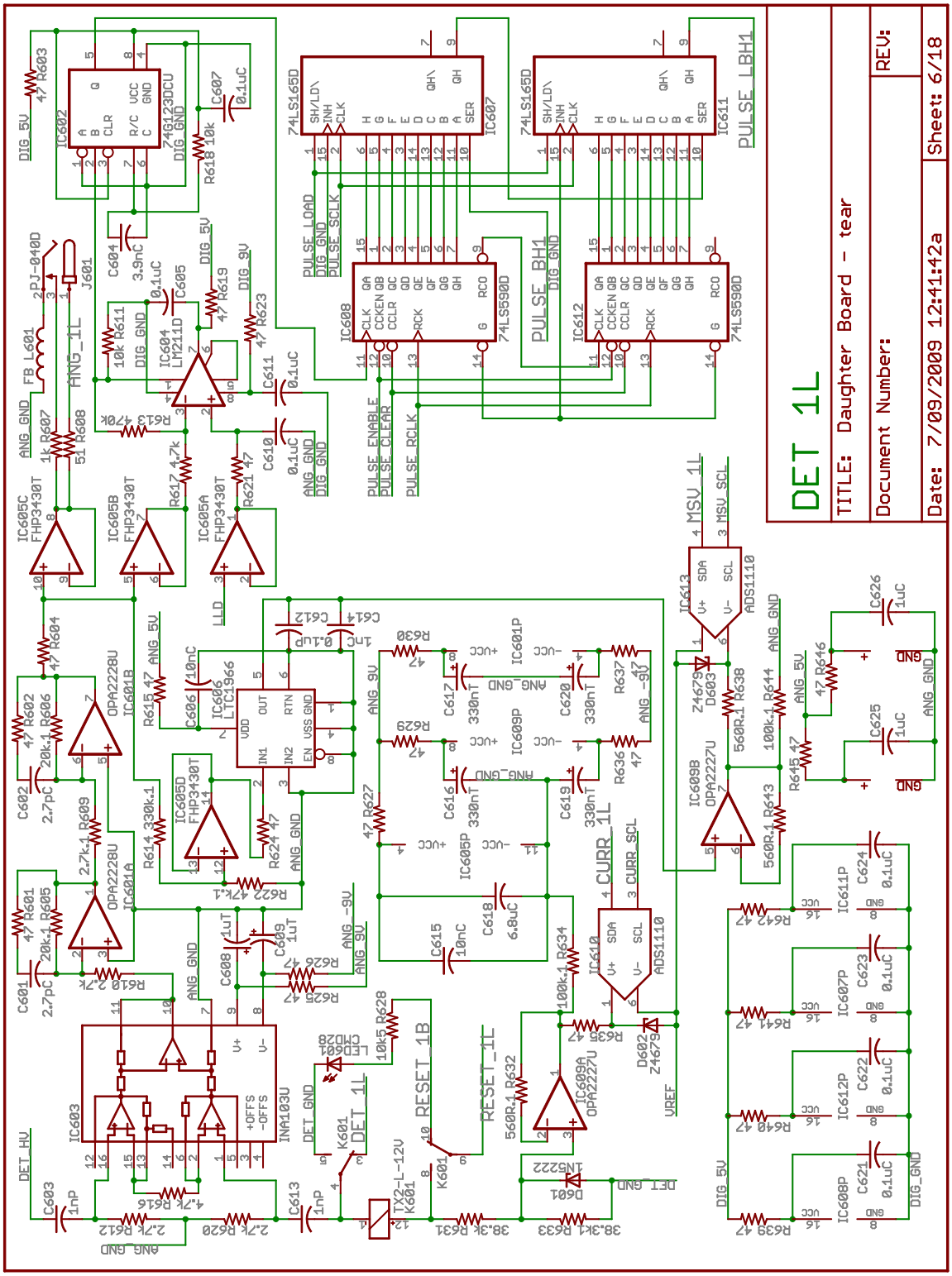


Figure D.6: Schematic for the daughter boards, page 6.

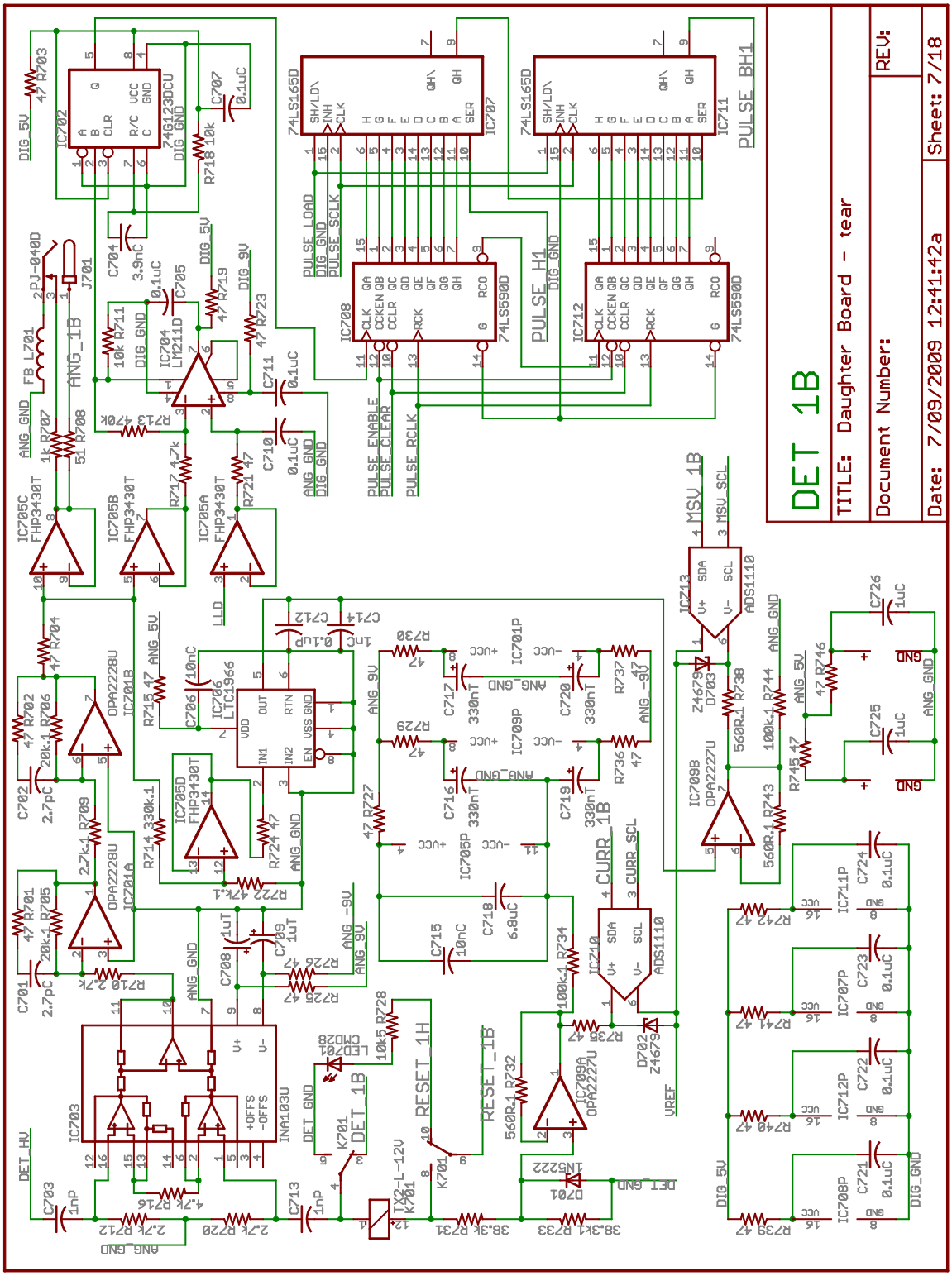


Figure D.7: Schematic for the daughter boards, page 7.

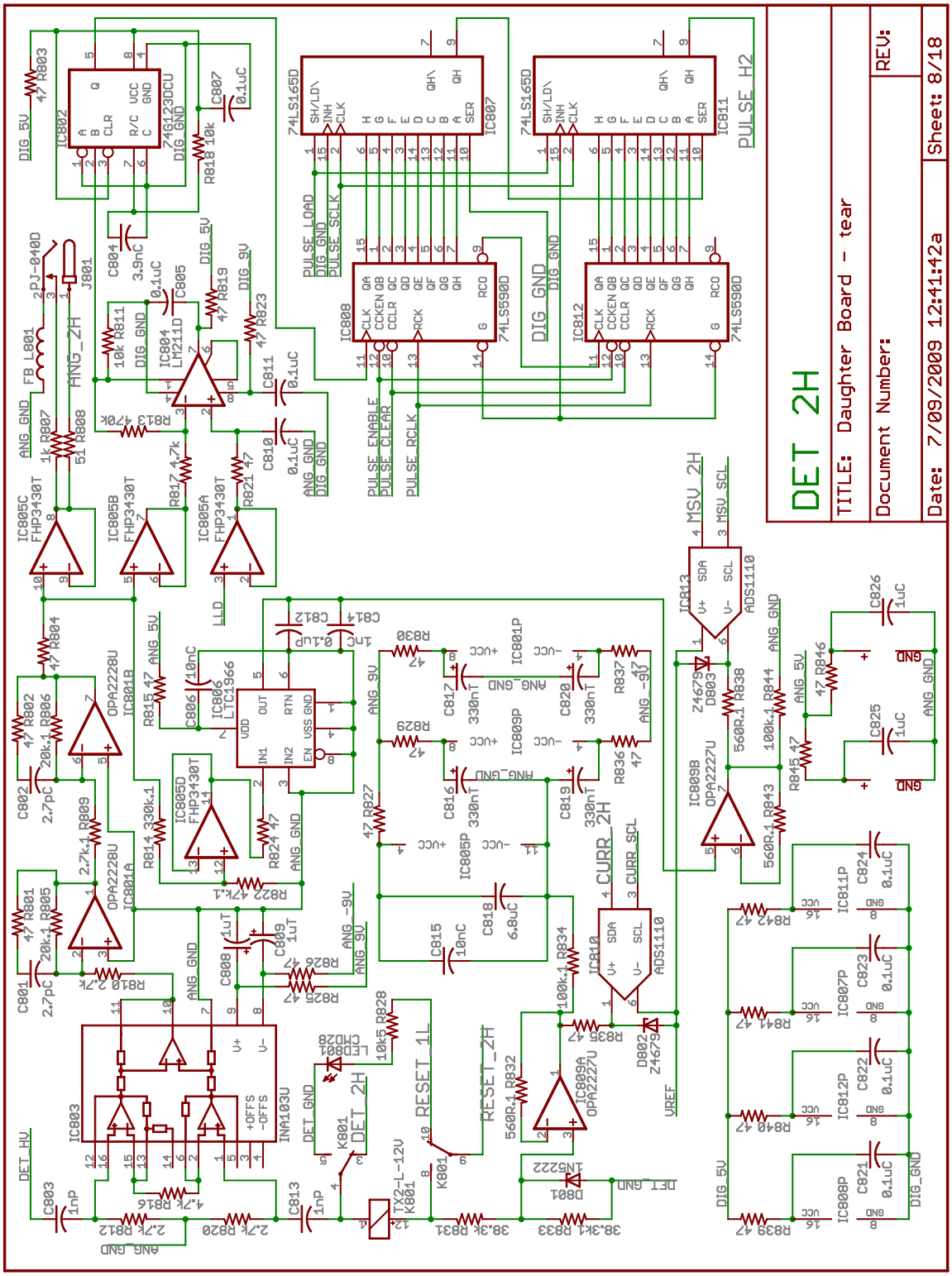


Figure D.8: Schematic for the daughter boards, page 8.

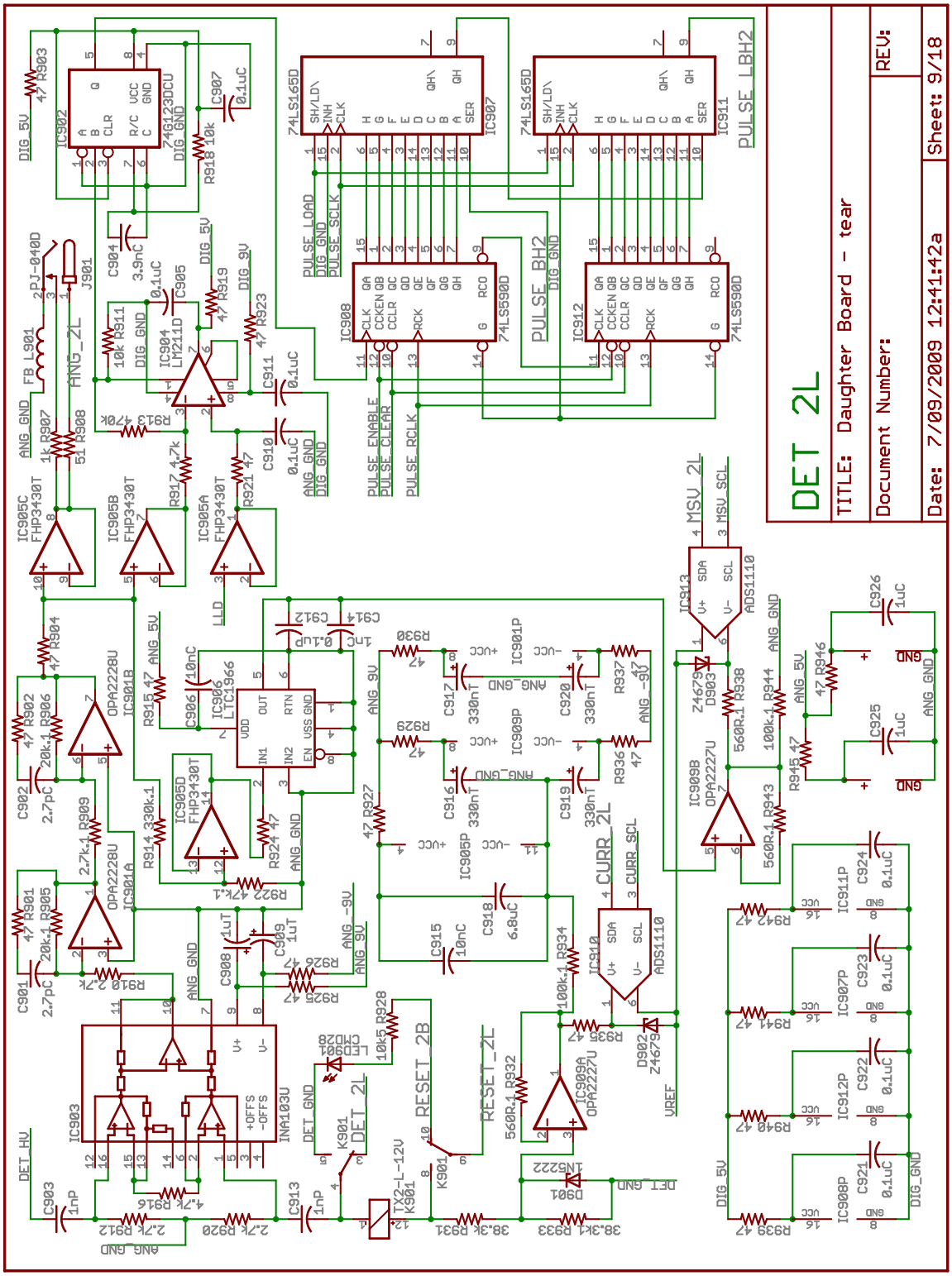


Figure D.9: Schematic for the daughter boards, page 9.

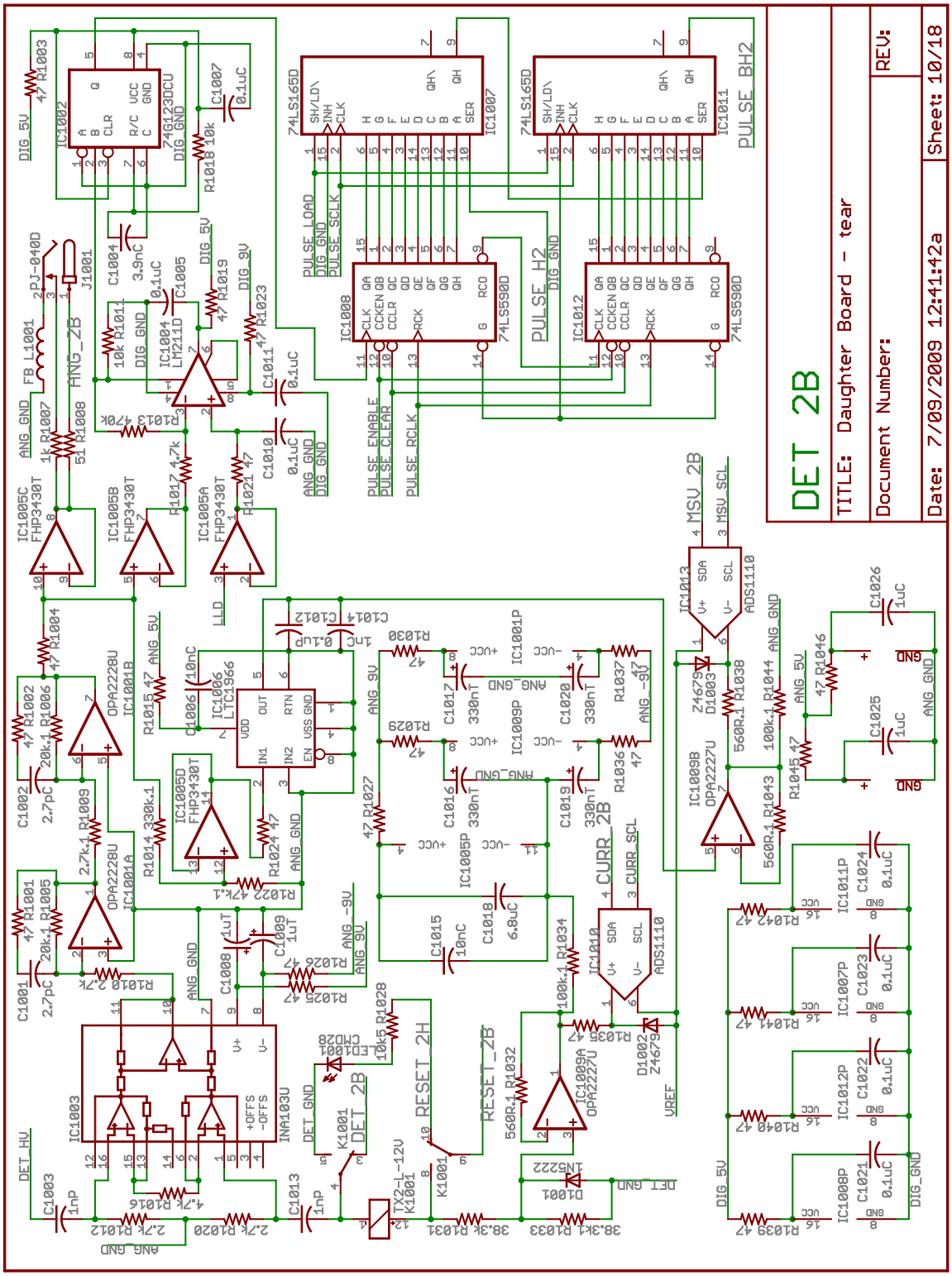


Figure D.10: Schematic for the daughter boards, page 10.

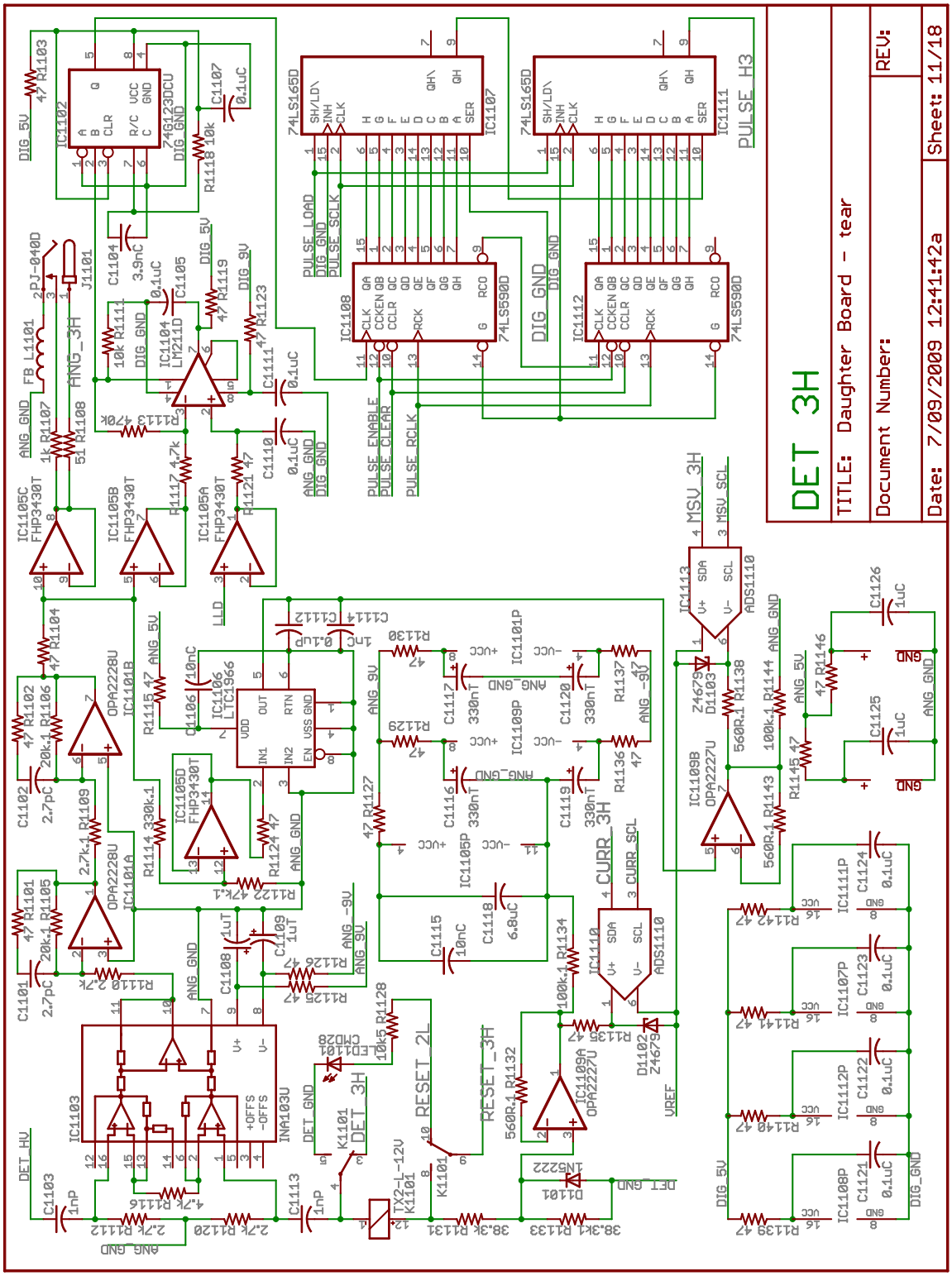


Figure D.11: Schematic for the daughter boards, page 11.

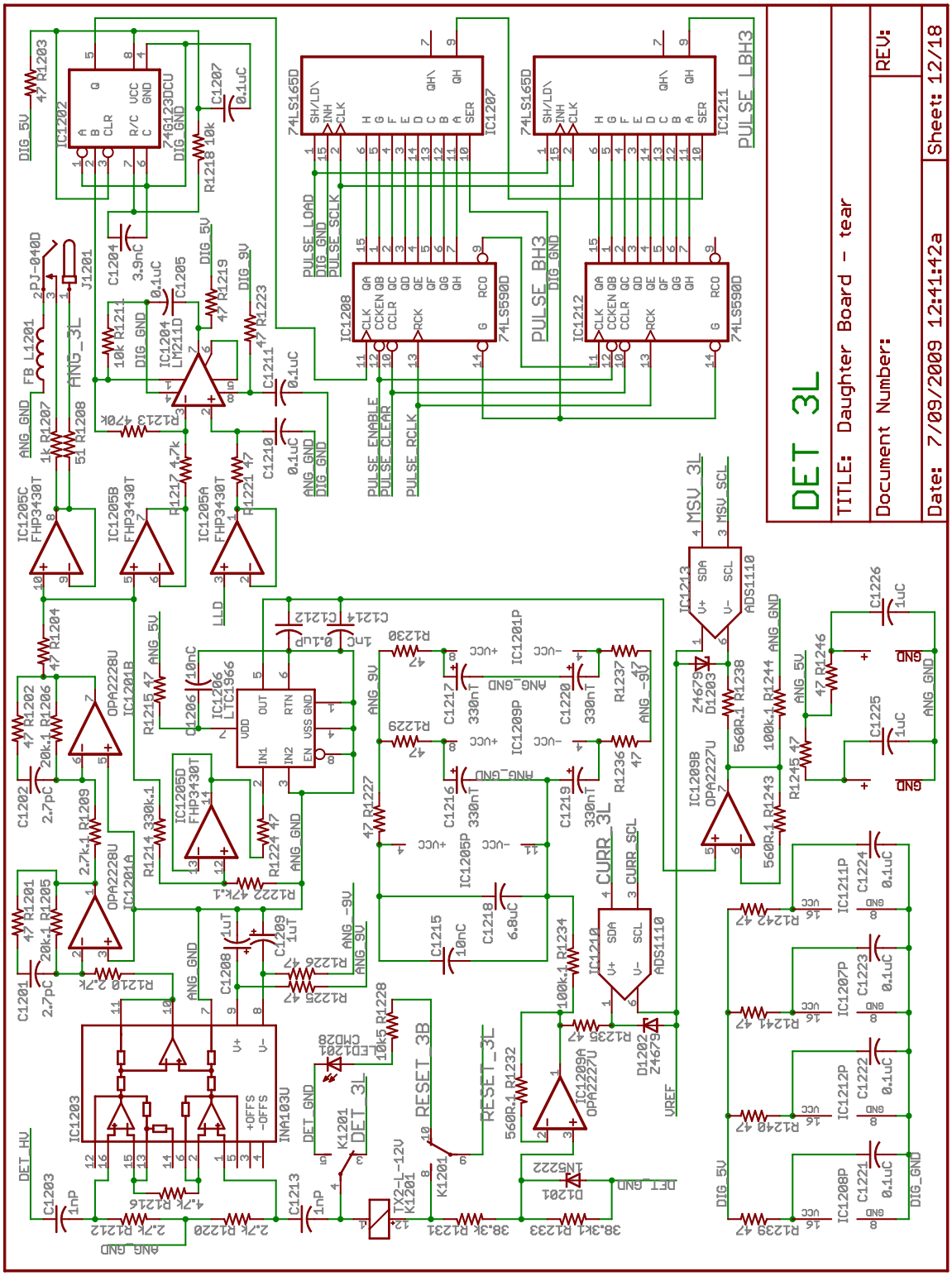


Figure D.12: Schematic for the daughter boards, page 12.

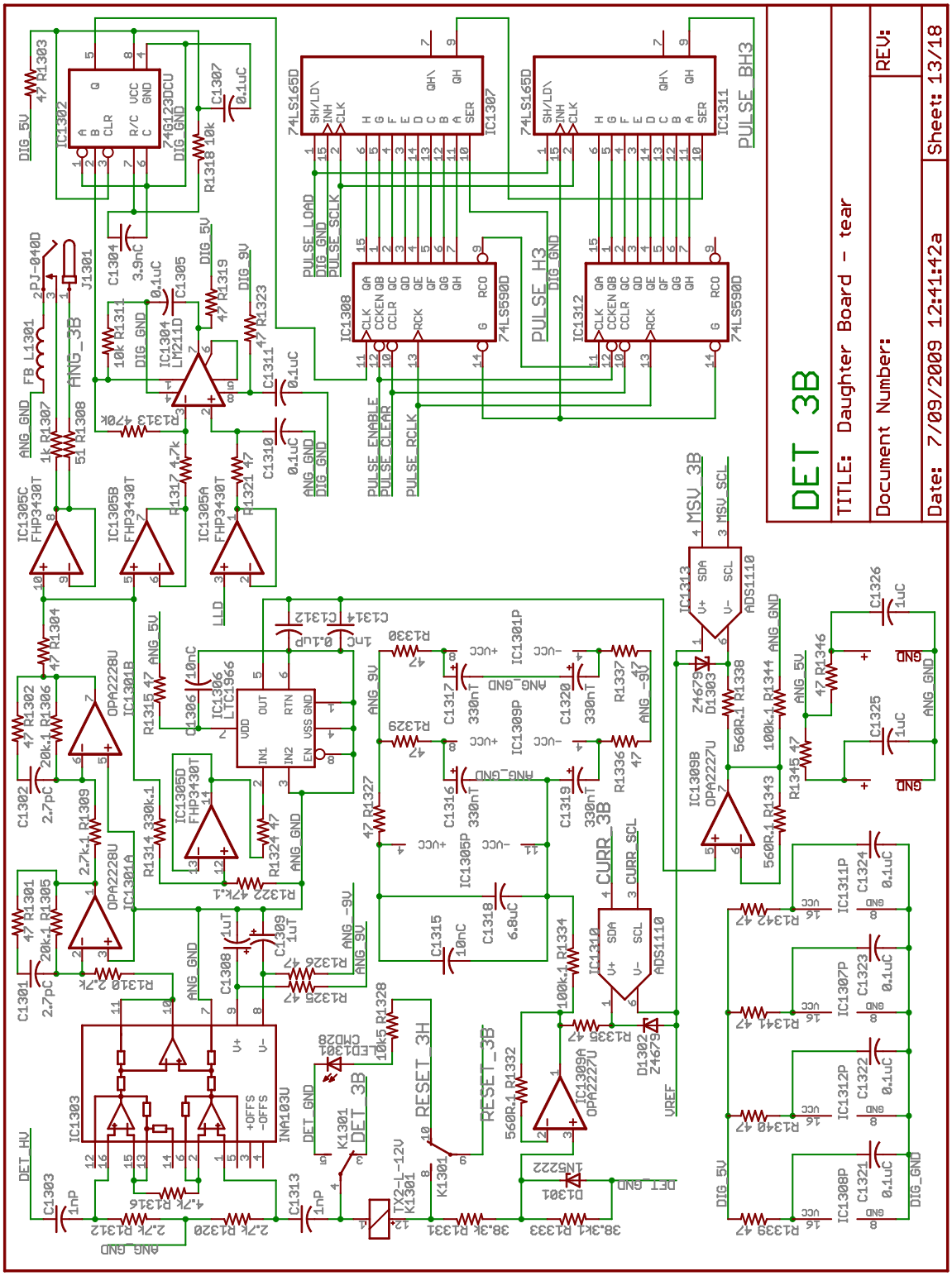


Figure D.13: Schematic for the daughter boards, page 13.

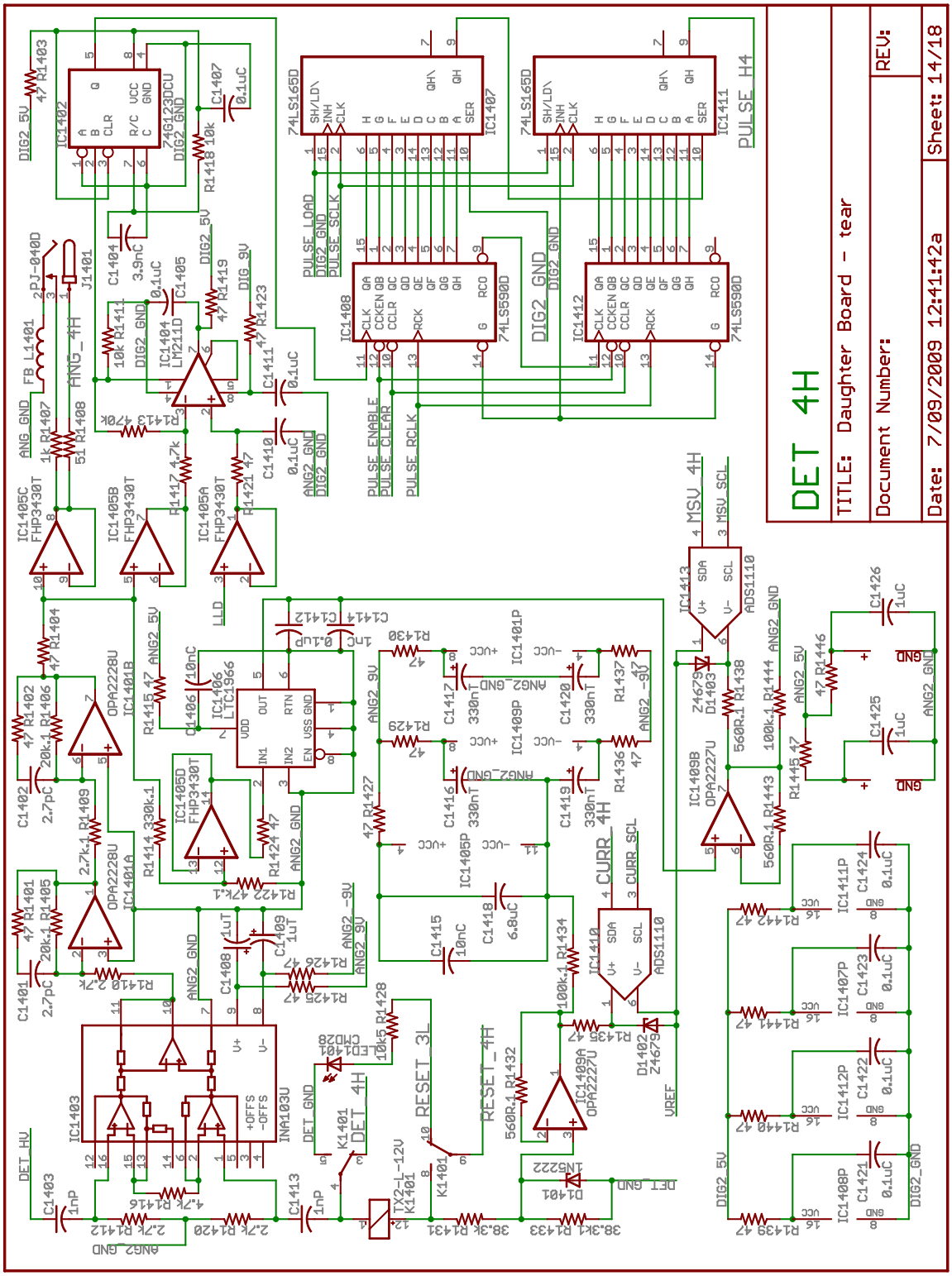


Figure D.14: Schematic for the daughter boards, page 14.

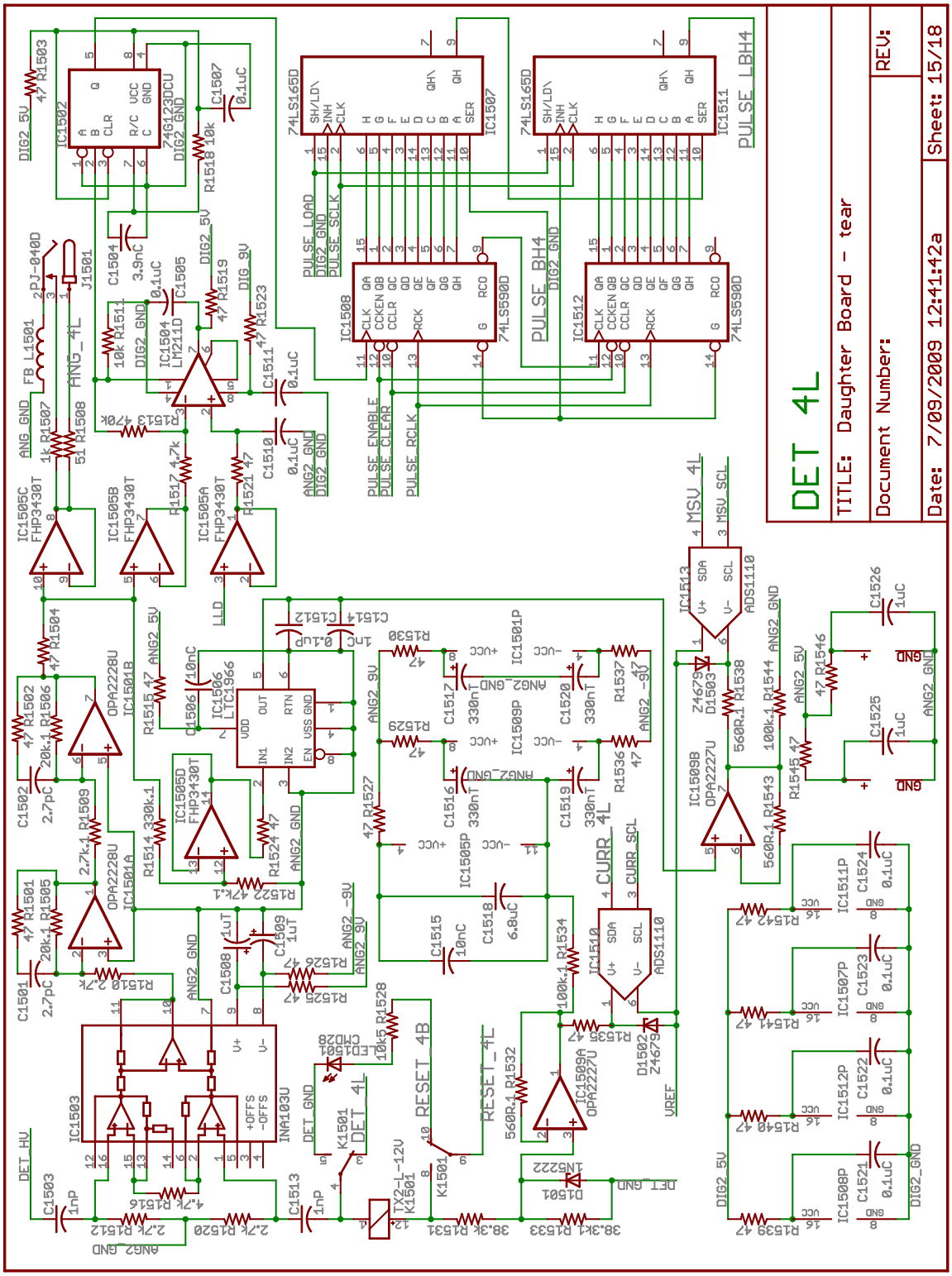


Figure D.15: Schematic for the daughter boards, page 15.

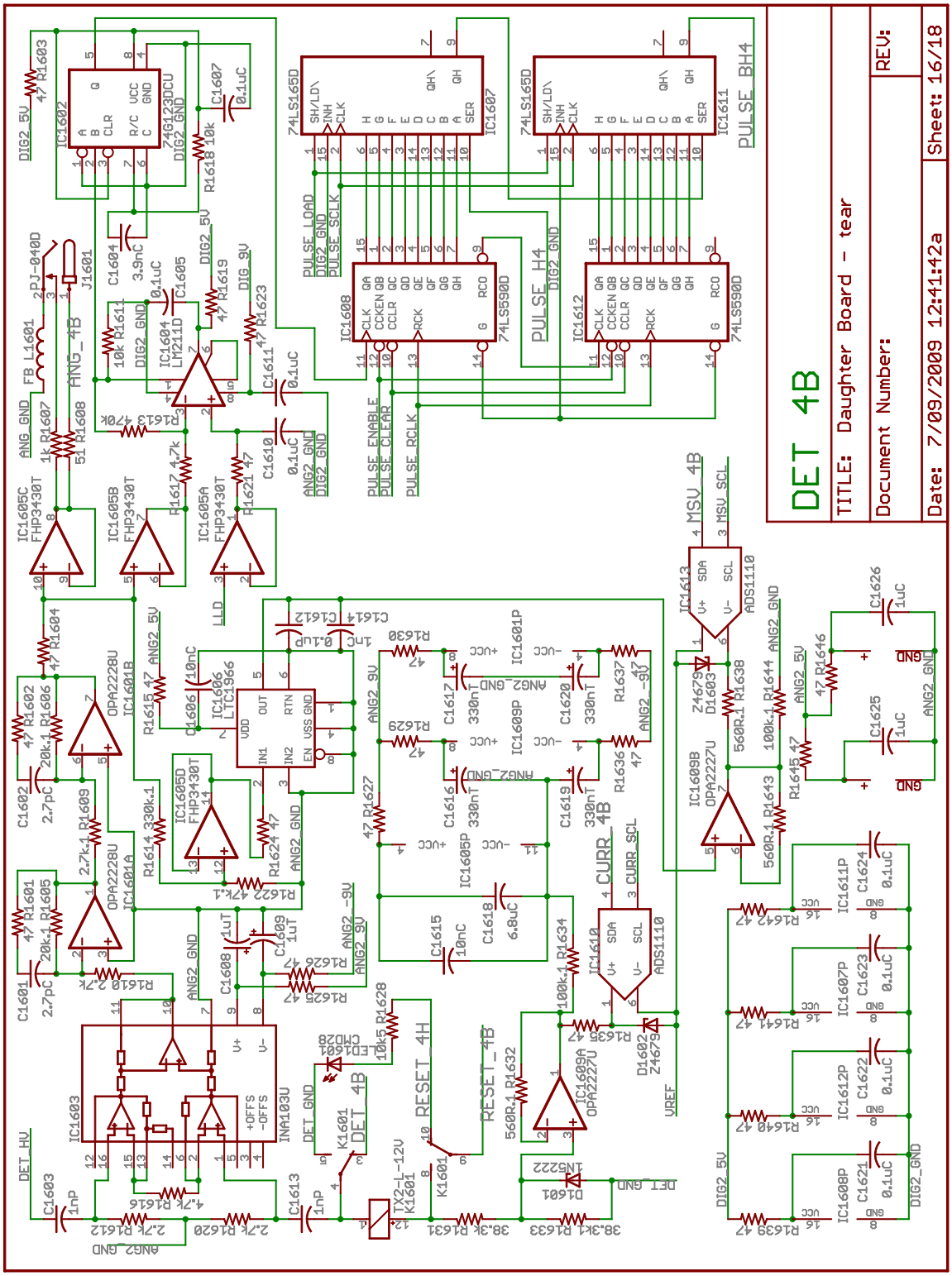


Figure D.16: Schematic for the daughter boards, page 16.

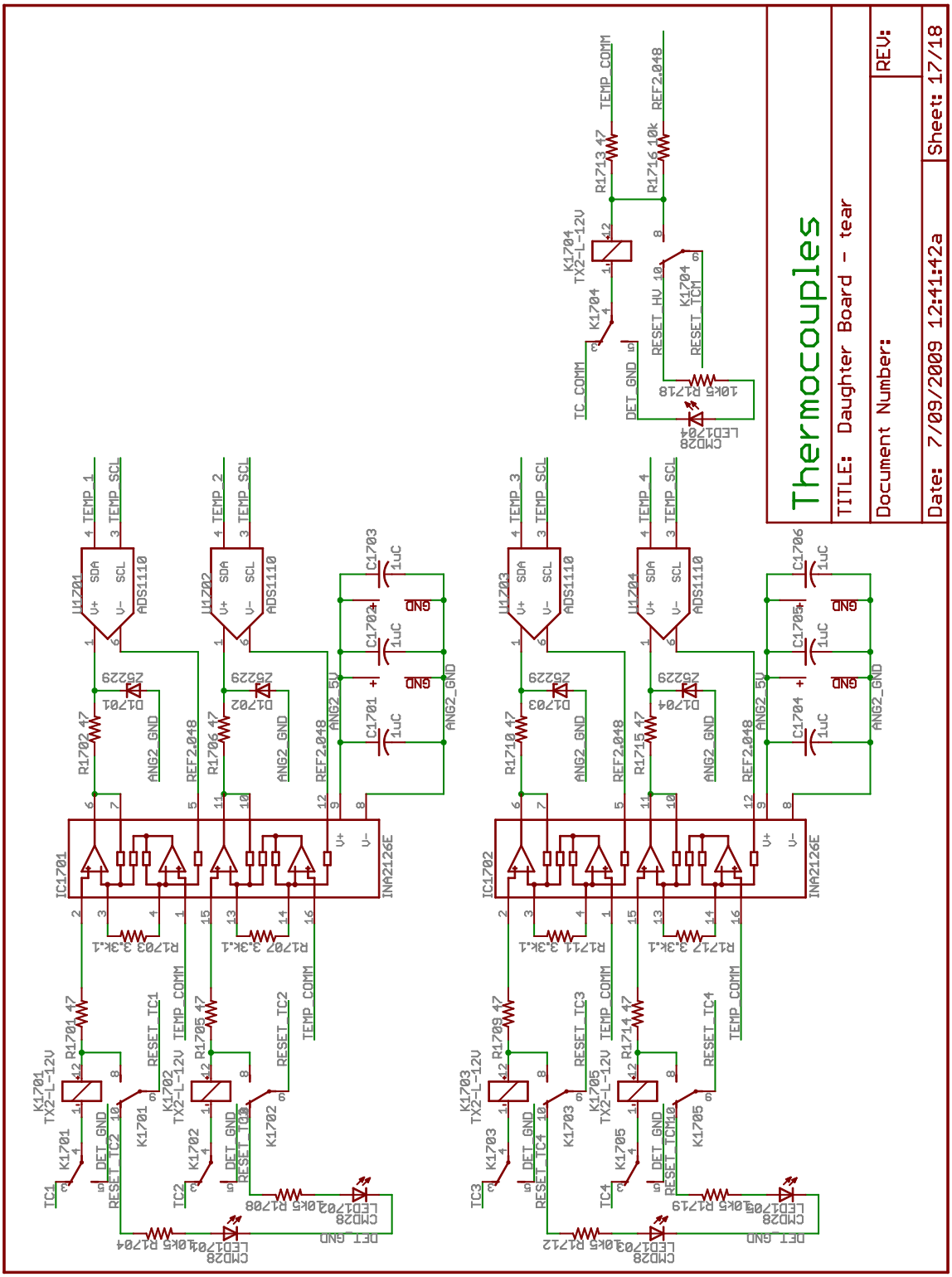


Figure D.17: Schematic for the daughter boards, page 17.

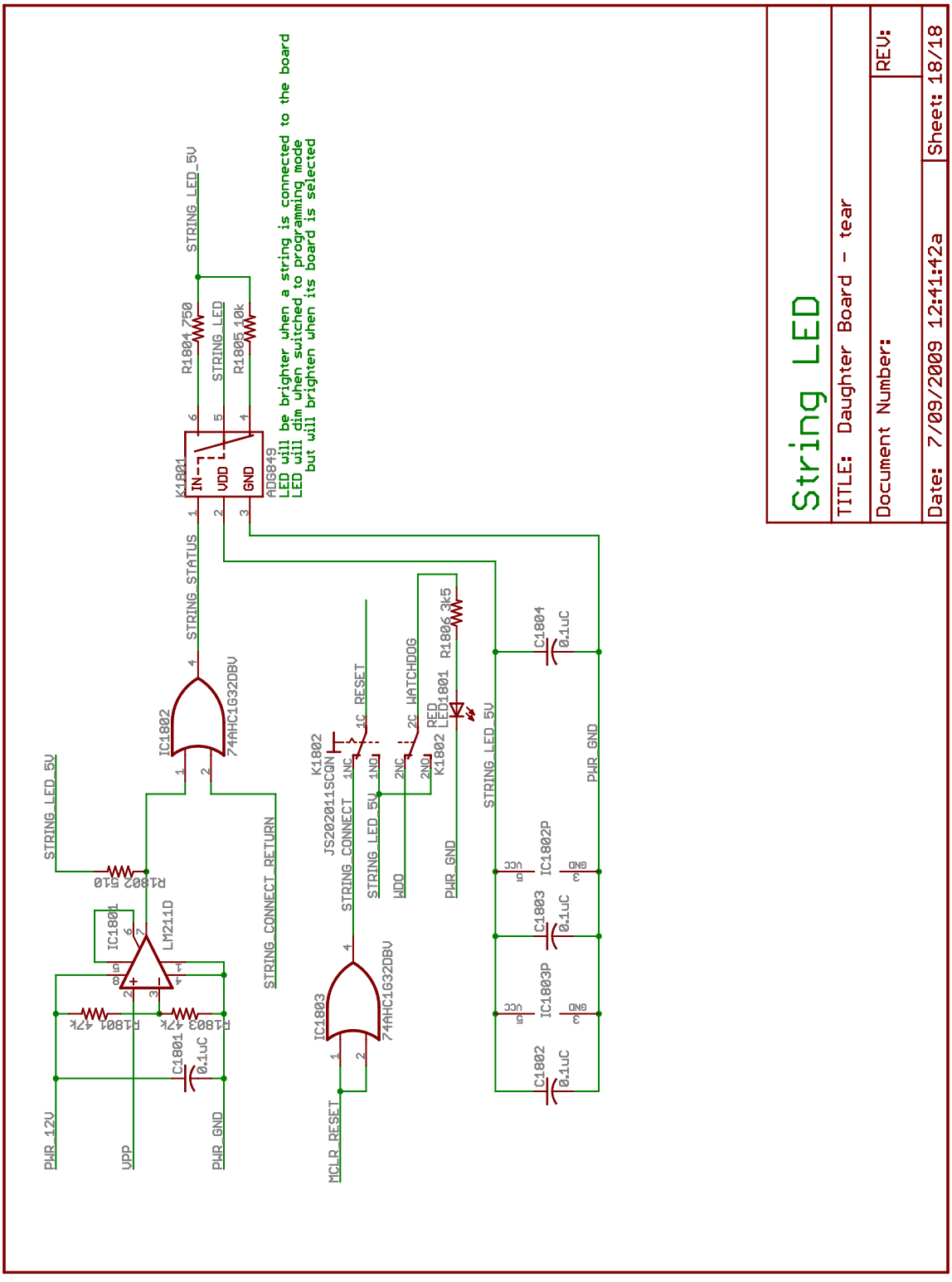


Figure D.18: Schematic for the daughter boards, page 18.

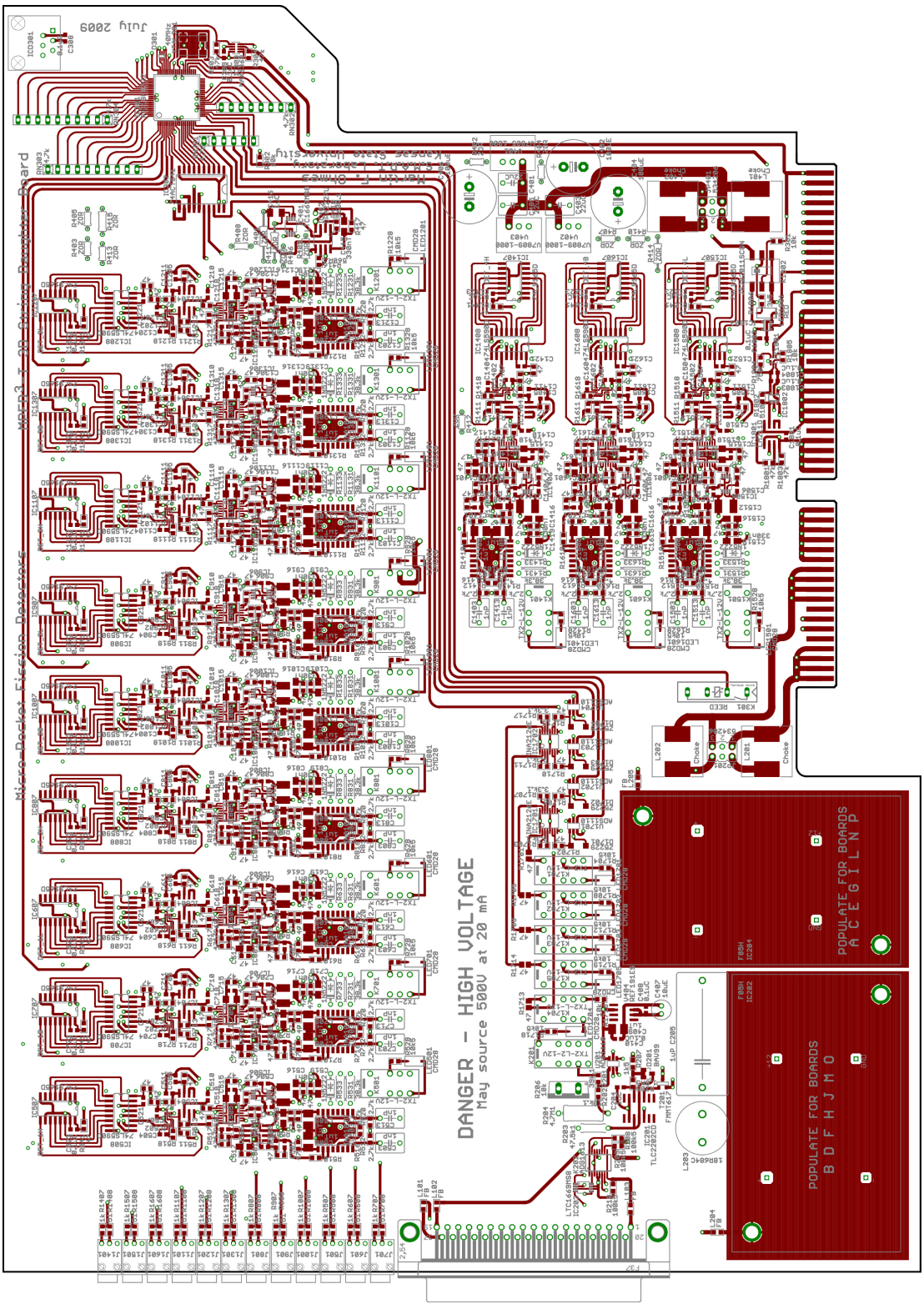


Figure D.19: Top side data and signal plane, layer 1, layout for the daughter boards.

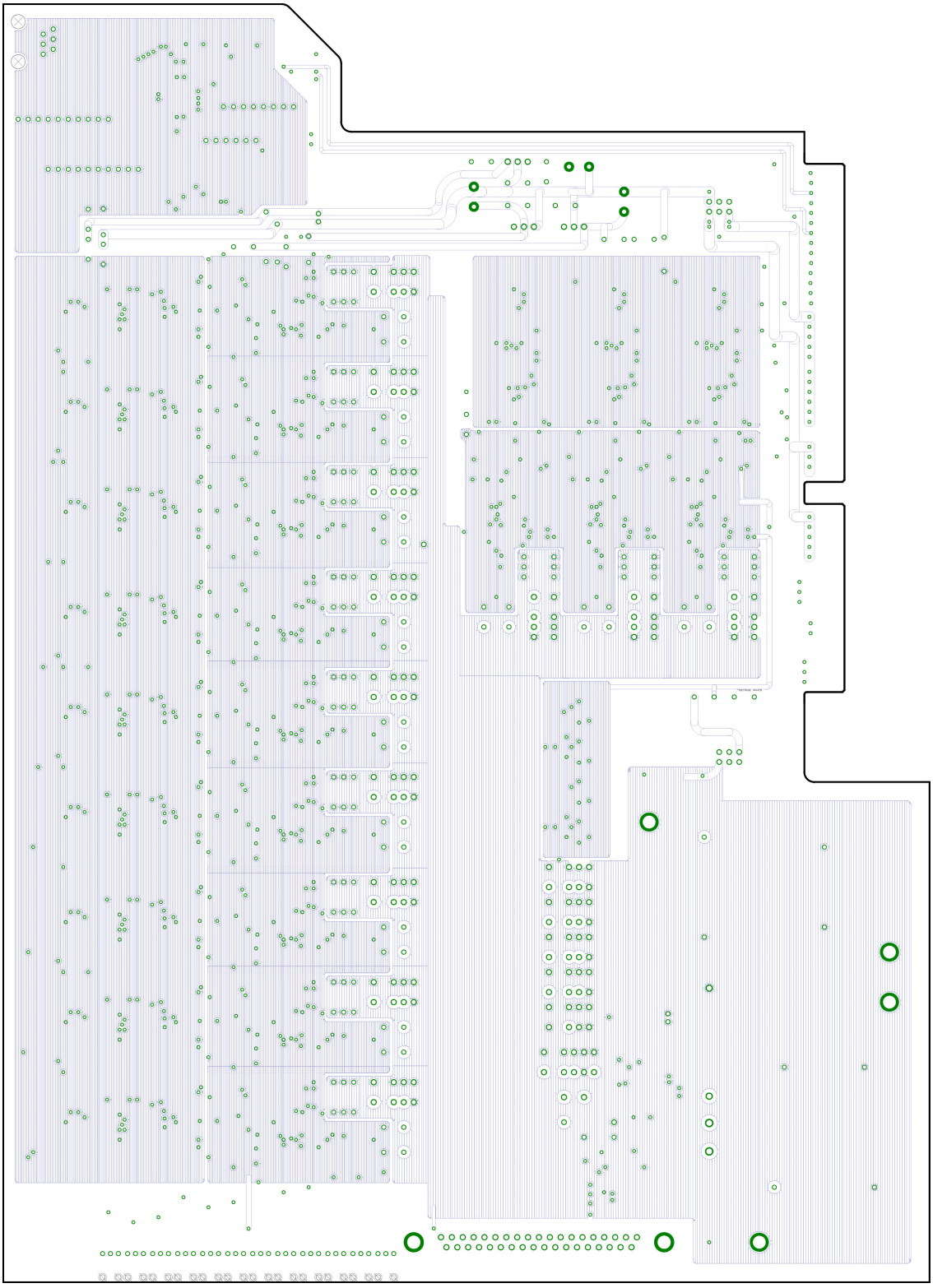


Figure D.20: Interior ground plane, layer 2, layout for the daughter boards.

Figure D.21: Interior data and signal plane, layer 3, layout for the daughter boards.

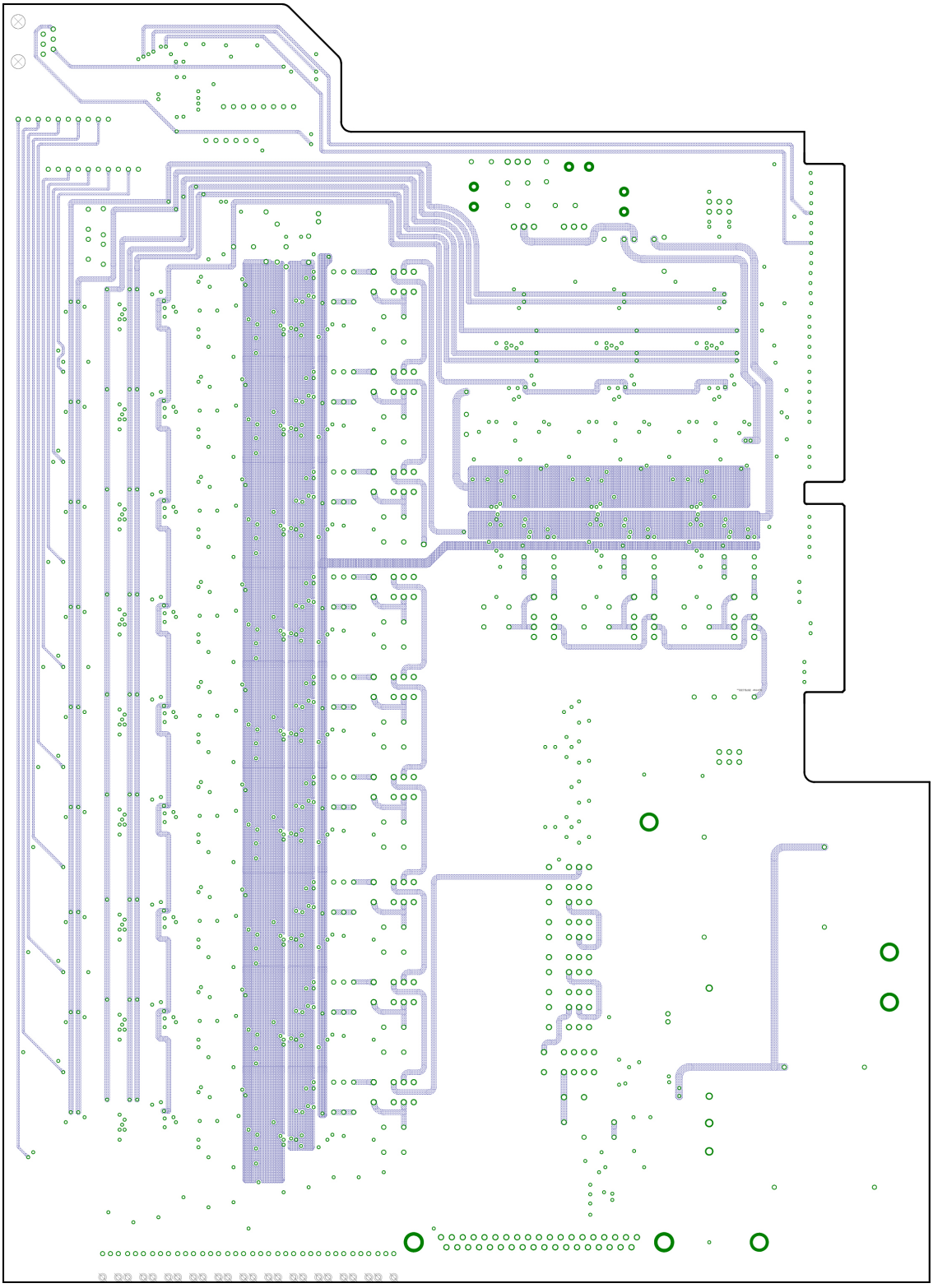


Figure D.22: Interior data and signal plane, layer 4, layout for the daughter boards.

Figure D.23: Interior power and high voltage plane, layer 5, layout for the daughter boards.

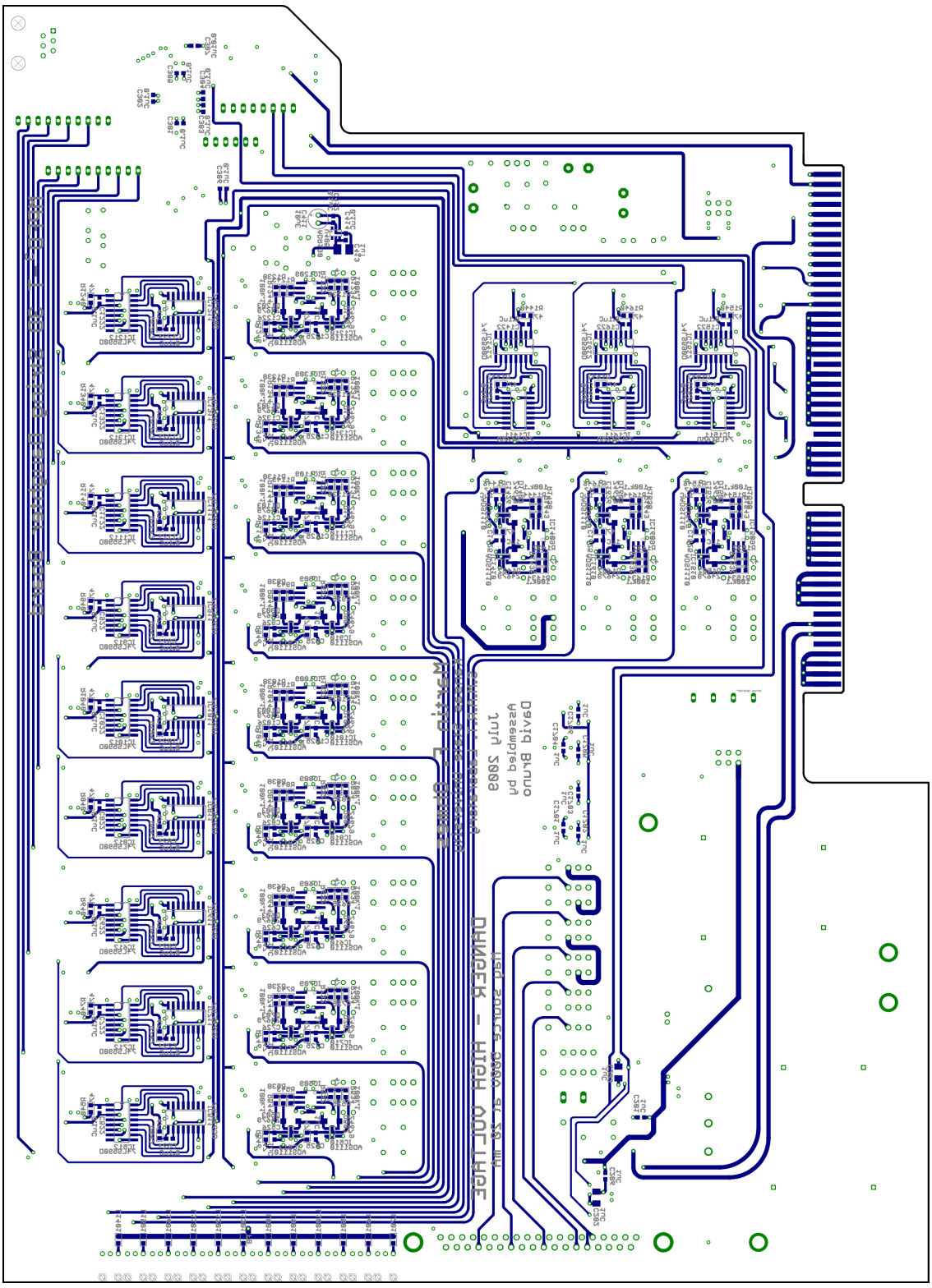


Figure D.24: Bottom side data and signal plane, layer 6, layout for the daughter boards.

Table D.1: Bill of materials for the daughter boards, page 1.

Qty	Value	Digikey (Mouser) Part Number	Part Identification on Schematic
33	1uC	587-1242-1-ND	C201, C204, C206, C525, C526, C625, C626, C725, C726, C825, C826, C925, C926, C1025, C1026, C1125, C1126, C1225, C1226, C1325, C1326, C1425, C1426, C1525, C1526, C1625, C1626, C1701, C1702, C1703, C1704, C1705, C1706
3	1uC	445-1592-1-ND	C202, C203, C416
1	1uP	495-3632-ND	C205
112	0.1uC	399-1096-1-ND (80-C0603C104K4R)	C301, C302, C303, C304, C305, C306, C308, C309, C408, C410, C412, C414, C505, C507, C510, C511, C521, C522, C523, C524, C605, C607, C610, C611, C621, C622, C623, C624, C705, C707, C710, C711, C721, C722, C723, C724, C805, C807, C810, C811, C821, C822, C823, C824, C905, C907, C910, C911, C921, C922, C923, C924, C1005, C1007, C1010, C1011, C1021, C1022, C1023, C1024, C1105, C1107, C1110, C1111, C1121, C1122, C1123, C1124, C1205, C1207, C1210, C1211, C1221, C1222, C1223, C1224, C1305, C1307, C1310, C1311, C1321, C1322, C1323, C1324, C1405, C1407, C1410, C1411, C1421, C1422, C1423, C1424, C1505, C1507, C1510, C1511, C1521, C1522, C1523, C1524, C1605, C1607, C1610, C1611, C1621, C1622, C1623, C1624, C1801, C1802, C1803, C1804
25	10nC	399-1091-1-ND (80-C0603C103K5R)	C307, C506, C515, C606, C615, C706, C715, C806, C815, C906, C915, C1006, C1015, C1106, C1115, C1206, C1215, C1306, C1315, C1406, C1415, C1506, C1515, C1606, C1615
3	22uC	445-2904-ND (810-FK22Y5V1E226Z)	C401, C403, C405
3	100uE (470uE)	493-3017-ND (647-PLF1A471MDO1)	C402, C404, C406 (Changed to 470uE due to availability)
2	10uE	P13138-ND	C407, C411
26	1uT	718-1309-1-ND	C409, C413, C508, C509, C608, C609, C708, C709, C808, C809, C908, C909, C1008, C1009, C1108, C1109, C1208, C1209, C1308, C1309, C1408, C1409, C1508, C1509, C1608, C1609
49	330nT	718-1141-1-ND or 399-1645-1-ND	C415, C516, C517, C519, C520, C616, C617, C619, C620, C716, C717, C719, C720, C816, C817, C819, C820, C916, C917, C919, C920, C1016, C1017, C1019, C1020, C1116, C1117, C1119, C1120, C1216, C1217, C1219, C1220, C1316, C1317, C1319, C1320, C1416, C1417, C1419, C1420, C1516, C1517, C1519, C1520, C1616, C1617, C1619, C1620
24	2.7pC	490-1382-1-ND (81-GRM39C2R7C50)	C501, C502, C601, C602, C701, C702, C801, C802, C901, C902, C1001, C1002, C1101, C1102, C1201, C1202, C1301, C1302, C1401, C1402, C1501, C1502, C1601, C1602
24	1nP	495-1255-ND	C503, C513, C603, C613, C703, C713, C803, C813, C903, C913, C1003, C1013, C1103, C1113, C1203, C1213, C1303, C1313, C1403, C1413, C1503, C1513, C1603, C1613

Table D.2: Bill of materials for the daughter boards, page 2.

Qty	Value	Digikey (Mouser) Part Number	Part Identification on Schematic
12	3.9nC	445-2665-1-ND	C504, C604, C704, C804, C904, C1004, C1104, C1204, C1304, C1404, C1504, C1604
12	0.1uP	478-4535-1-ND	C512, C612, C712, C812, C912, C1012, C1112, C1212, C1312, C1412, C1512, C1612
12	1nC	399-1083-1-ND (80-C0603C102J5R)	C514, C614, C714, C814, C914, C1014, C1114, C1214, C1314, C1414, C1514, C1614
12	6.8uC	445-4040-1-ND	C518, C618, C718, C818, C918, C1018, C1118, C1218, C1318, C1418, C1518, C1618
4	Z5229	MMSZ5229BFSCT-ND (863-MMSZ5229BT1G)	D1701, D1702, D1703, D1704
1	BAV99	BAV99TPMSCT-ND	D201
1	BAT54-09P6FILM	497-5713-1-ND	D301
12	1N5222	IN5222B-TPCT-ND	D501, D601, D701, D801, D901, D1001, D1101, D1201, D1301, D1401, D1501, D1601
24	Z4679	MMSZ4679T1GOSCT-ND (863-MMSZ4679T1G)	D502, D503, D602, D603, D702, D703, D802, D803, D902, D903, D1002, D1003, D1102, D1103, D1202, D1203, D1302, D1303, D1402, D1403, D1502, D1503, D1602, D1603
2	INA2126E	INA2126ECT-ND	IC1701, IC1702
2	74AHC1G32DBV	296-1093-1-ND	IC1802, IC1803
1	TLC2202CD	296-1807-5-ND	IC201
2	F05H	EMCO F05H	IC202, IC204
2	LTC1669MS8	LTC1669IMS8#PBF-ND	IC203, IC401
1	PIC18F8722	PIC18F8722-I/PT-ND	IC301
1	74ACT541	296-4280-5-ND (595-CD74ACT541M)	IC302
1	MAX6369	MAX6369KA+TCT-ND	IC303
13	OPA2227U	OPA2227UA-ND (595-OPA2227UAE4)	IC402, IC509, IC609, IC709, IC809, IC909, IC1009, IC1109, IC1209, IC1309, IC1409, IC1509, IC1609
12	OPA2228U	OPA2228U-ND	IC501, IC601, IC701, IC801, IC901, IC1001, IC1101, IC1201, IC1301, IC1401, IC1501, IC1601
12	74G123DCU	296-18578-1-ND (595-SN74LVC1G123DCUR)	IC502, IC602, IC702, IC802, IC902, IC1002, IC1102, IC1202, IC1302, IC1402, IC1502, IC1602
12	INA103U	INA103KU-ND (595-INA103KU)	IC503, IC603, IC703, IC803, IC903, IC1003, IC1103, IC1203, IC1303, IC1403, IC1503, IC1603
13	LM211D	296-14589-1-ND	IC504, IC604, IC704, IC804, IC904, IC1004, IC1104, IC1204, IC1304, IC1404, IC1504, IC1604, IC1801
12	FHP3430T	FHP3430IMTC14XCT-ND (512-FHP3430IMTC14X)	IC505, IC605, IC705, IC805, IC905, IC1005, IC1105, IC1205, IC1305, IC1405, IC1505, IC1605
12	LTC1966	LTC1966CMS8-ND	IC506, IC606, IC706, IC806, IC906, IC1006, IC1106, IC1206, IC1306, IC1406, IC1506, IC1606

Table D.3: Bill of materials for the daughter boards, page 3.

Qty	Value	Digikey (Mouser) Part Number	Part Identification on Schematic
24	74LS165D	296-8249-5-ND	IC507, IC511, IC607, IC611, IC707, IC711, IC807, IC811, IC907, IC911, IC1007, IC1011, IC1107, IC1111, IC1207, IC1211, IC1307, IC1311, IC1407, IC1411, IC1507, IC1511, IC1607, IC1611
24	74LS590D	296-17132-1-ND	IC508, IC512, IC608, IC612, IC708, IC712, IC808, IC812, IC908, IC912, IC1008, IC1012, IC1108, IC1112, IC1208, IC1212, IC1308, IC1312, IC1408, IC1412, IC1508, IC1512, IC1608, IC1612
29	ADS1110	296-14462-1-ND	IC510, IC513, IC610, IC613, IC710, IC713, IC810, IC813, IC910, IC913, IC1010, IC1013, IC1110, IC1113, IC1210, IC1213, IC1310, IC1313, IC1410, IC1413, IC1510, IC1513, IC1610, IC1613, U201, U1701, U1702, U1703, U1704
1	ICD	A31406-ND or A31401-ND (571-5520242-3)	ICD301
12	PJ-040D	CP-040D-ND	J501, J601, J701, J801, J901, J1001, J1101, J1201, J1301, J1401, J1501, J1601
2	534206	A32934-ND (571-5-534206-3)	JP201, JP401
4		923345-01-ND (517-923345-01)	Jumper pins if needed
1	ADG849	ADG849YKSZ500RLCT	K1801
1	JS202011SCQN	401-2002-ND	K1802
1	TX2-L2-12V	255-1782-5-ND	K201
1	ADG1613	ADG1613BRUZ-ND	K202
1	REED	CLA274-ND	K301
17	TX2-L-12V	255-1898-ND	K501, K601, K701, K801, K901, K1001, K1101, K1201, K1301, K1401, K1501, K1601, K1701, K1702, K1703, K1704, K1705
17	FB	445-2179-1-ND	L101, L102, L103, L204, L205, L501, L601, L701, L801, L901, L1001, L1101, L1201, L1301, L1401, L1501, L1601
4	Choke	M8105CT-ND	L201, L202, L401, L402
1	18R684C	811-1310-ND (580-18R684C)	L203
1	RED	L62701CT-ND	LED1801
17	CMD28	L62705CT-ND (606-CMD28-21VGCT1)	LED501, LED601, LED701, LED801, LED901, LED1001, LED1101, LED1201, LED1301, LED1401, LED1501, LED1601, LED1701, LED1702, LED1703, LED1704, LED1705
5	47	RP20S47FCT-ND	R1701, R1705, R1709, R1713, R1714
4	3.3k.1	P3.3KDBCT-ND	R1703, R1707, R1711, R1717
1	510	311-510GRCT-ND	R1802
1	750	311-750GRCT-ND	R1804
1	3k5	311-3.0KGRCT-ND	R1806
1	39k.1	P39KDBCT-ND	R201
1	10k.1	P10KDBCT-ND	R202

Table D.4: Bill of materials for the daughter boards, page 4.

Qty	Value	Digikey (Mouser) Part Number	Part Identification on Schematic
1	47.5k1	47.5KXBK-ND	R203
1	4.7M1	PPCHF4.70MCT-ND	R204
1	1.8M5	541-1.80MHCT-ND	R205
1	10k	MP925-10.0KF-ND	R206
13	1k	P1.0KDBCT-ND	R207, R507, R607, R707, R807, R907, R1007, R1107, R1207, R1307, R1407, R1507, R1607
3	100k5	311-100KGRCT-ND	R208, R209, R210
45	10k5	311-10KGRTR-ND	R301, R302, R511, R518, R528, R611, R618, R628, R711, R718, R728, R811, R818, R828, R911, R918, R928, R1011, R1018, R1028, R1111, R1118, R1128, R1211, R1218, R1228, R1311, R1318, R1328, R1411, R1418, R1428, R1511, R1518, R1528, R1611, R1618, R1628, R1704, R1708, R1712, R1716, R1718, R1719, R1805
4	47k	311-47KGRCT-ND	R303, R304, R1801, R1803
15	ZOR	0.0EBK-ND	R401, R402, R403, R404, R405, R406, R407, R408, R409, R410, R411, R412, R413, R414, R415
38	560R.1	P560DBCT-ND	R416, R418, R532, R538, R543, R632, R638, R643, R732, R738, R743, R832, R838, R843, R932, R938, R943, R1032, R1038, R1043, R1132, R1138, R1143, R1232, R1238, R1243, R1332, R1338, R1343, R1432, R1438, R1443, R1532, R1538, R1543, R1632, R1638, R1643
281	47	311-47GRCT-ND	R417, R501, R502, R503, R504, R515, R519, R521, R523, R524, R525, R526, R527, R529, R530, R535, R536, R537, R539, R540, R541, R542, R545, R546, R601, R602, R603, R604, R615, R619, R621, R623, R624, R625, R626, R627, R629, R630, R635, R636, R637, R639, R640, R641, R642, R645, R646, R701, R702, R703, R704, R715, R719, R721, R723, R724, R725, R726, R727, R729, R730, R735, R736, R737, R739, R740, R741, R742, R745, R746, R801, R802, R803, R804, R815, R819, R821, R823, R824, R825, R826, R827, R829, R830, R835, R836, R837, R839, R840, R841, R842, R845, R846, R901, R902, R903, R904, R915, R919, R921, R923, R924, R925, R926, R927, R929, R930, R935, R936, R937, R939, R940, R941, R942, R945, R946, R1001, R1002, R1003, R1004, R1015, R1019, R1021, R1023, R1024, R1025, R1026, R1027, R1029, R1030, R1035, R1036, R1037, R1039, R1040, R1041, R1042, R1045, R1046, R1101, R1102, R1103, R1104, R1115, R1119, R1121, R1123, R1124, R1125, R1126, R1127, R1129, R1130, R1135, R1136, R1137, R1139, R1140, R1141, R1142, R1145, R1146, R1201, R1202, R1203, R1204, R1215, R1219, R1221, R1223, R1224, R1225, R1226, R1227, R1229, R1230, R1235, R1236, R1237, R1239, R1240, R1241, R1242,
24	20k.1	P20KYCT-ND	R505, R506, R605, R606, R705, R706, R805, R806, R905, R906, R1005, R1006, R1105, R1106, R1205, R1206, R1305, R1306, R1405, R1406, R1505, R1506, R1605, R1606

Table D.5: Bill of materials for the daughter boards, page 5.

Qty	Value	Digikey (Mouser) Part Number	Part Identification on Schematic
12	51	P51DACT-ND	R508, R608, R708, R808, R908, R1008, R1108, R1208, R1308, R1408, R1508, R1608
48	2.7k.1	P2.7KDBCT-ND	R509, R510, R512, R520, R609, R610, R612, R620, R709, R710, R712, R720, R809, R810, R812, R820, R909, R910, R912, R920, R1009, R1010, R1012, R1020, R1109, R1110, R1112, R1120, R1209, R1210, R1212, R1220, R1309, R1310, R1312, R1320, R1409, R1410, R1412, R1420, R1509, R1510, R1512, R1520, R1609, R1610, R1612, R1620
12	470k	RMCF1/16470KFRTR-ND	R513, R613, R713, R813, R913, R1013, R1113, R1213, R1313, R1413, R1513, R1613
12	330k.1	P330KDBCT-ND	R514, R614, R714, R814, R914, R1014, R1114, R1214, R1314, R1414, R1514, R1614
24	4.7k	P4.7KDBCT-ND	R516, R517, R616, R617, R716, R717, R816, R817, R916, R917, R1016, R1017, R1116, R1117, R1216, R1217, R1316, R1317, R1416, R1417, R1516, R1517, R1616, R1617
12	47k.1	P47KDBCT-ND	R522, R622, R722, R822, R922, R1022, R1122, R1222, R1322, R1422, R1522, R1622
24	38.3k1	CMF38.3KHFCT-ND	R531, R533, R631, R633, R731, R733, R831, R833, R931, R933, R1031, R1033, R1131, R1133, R1231, R1233, R1331, R1333, R1431, R1433, R1531, R1533, R1631, R1633
24	100k.1	P100KDBCT-ND	R534, R544, R634, R644, R734, R744, R834, R844, R934, R944, R1034, R1044, R1134, R1144, R1234, R1244, R1334, R1344, R1434, R1444, R1534, R1544, R1634, R1644
1	4.7k	4606X-1-472LF-ND	RN301
1	4.7k	4608X-1-472LF-ND	RN302
2	4.7k	4610X-1-472LF-ND	RN303, RN304
1	FMMT617	FMMT617CT-ND	T201
1	V7805-1000	102-1715-ND	V401
2	V7809-1000	102-1717-ND	V402, V403
1	REF191ES	REF191ESZ-ND	V404
1	ADR130	ADR130BUJZ-REEL7CT-ND	V405
1		182-937FE-ND	X101
1	40MHz	478-2079-1-ND	XTAL301

D.2 MPFD^3-T Mother Board Schematic and Layout

For descriptions of the circuits see Section §5.2.

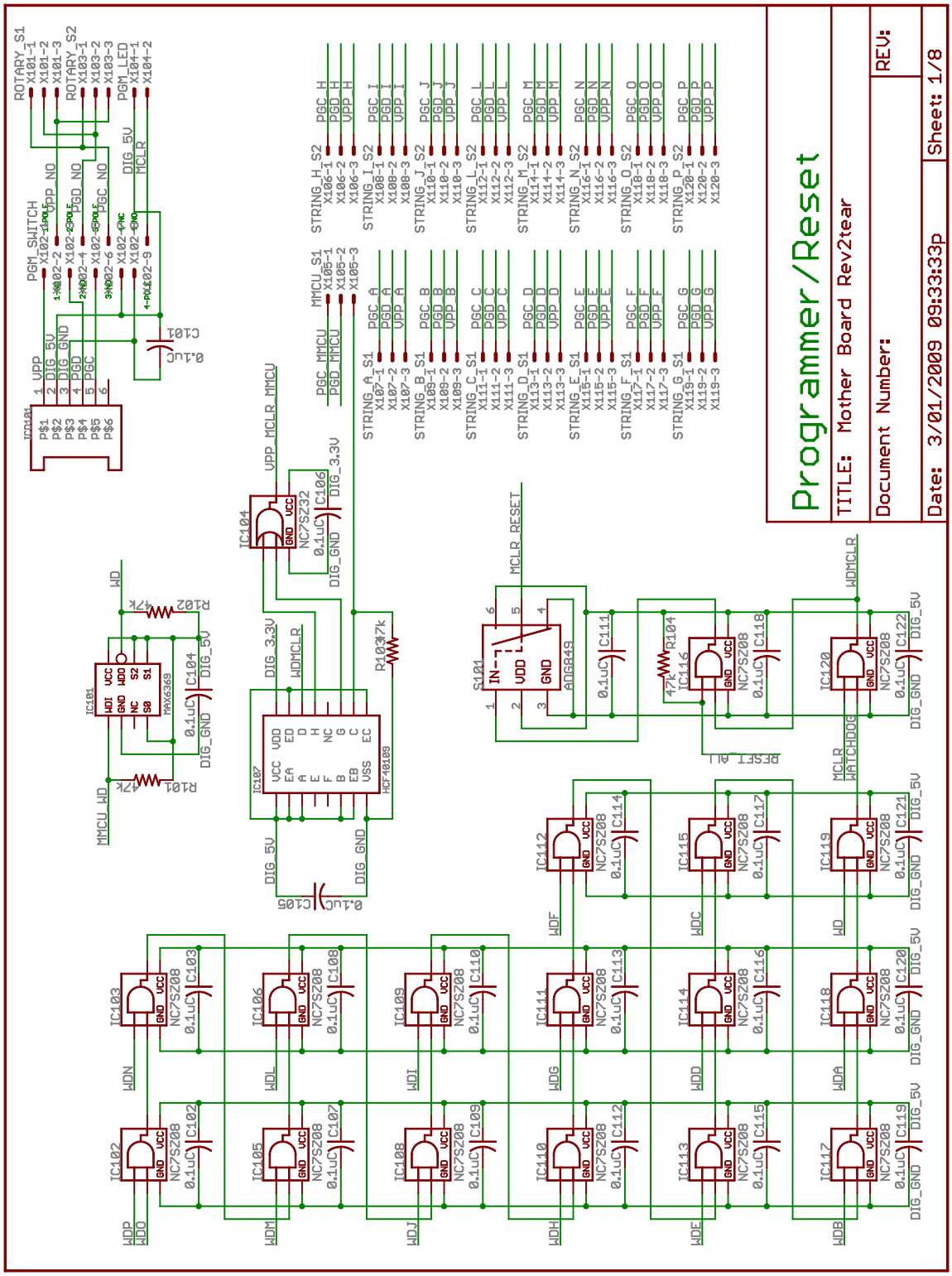


Figure D.25: Schematic for the mother board, page 1.

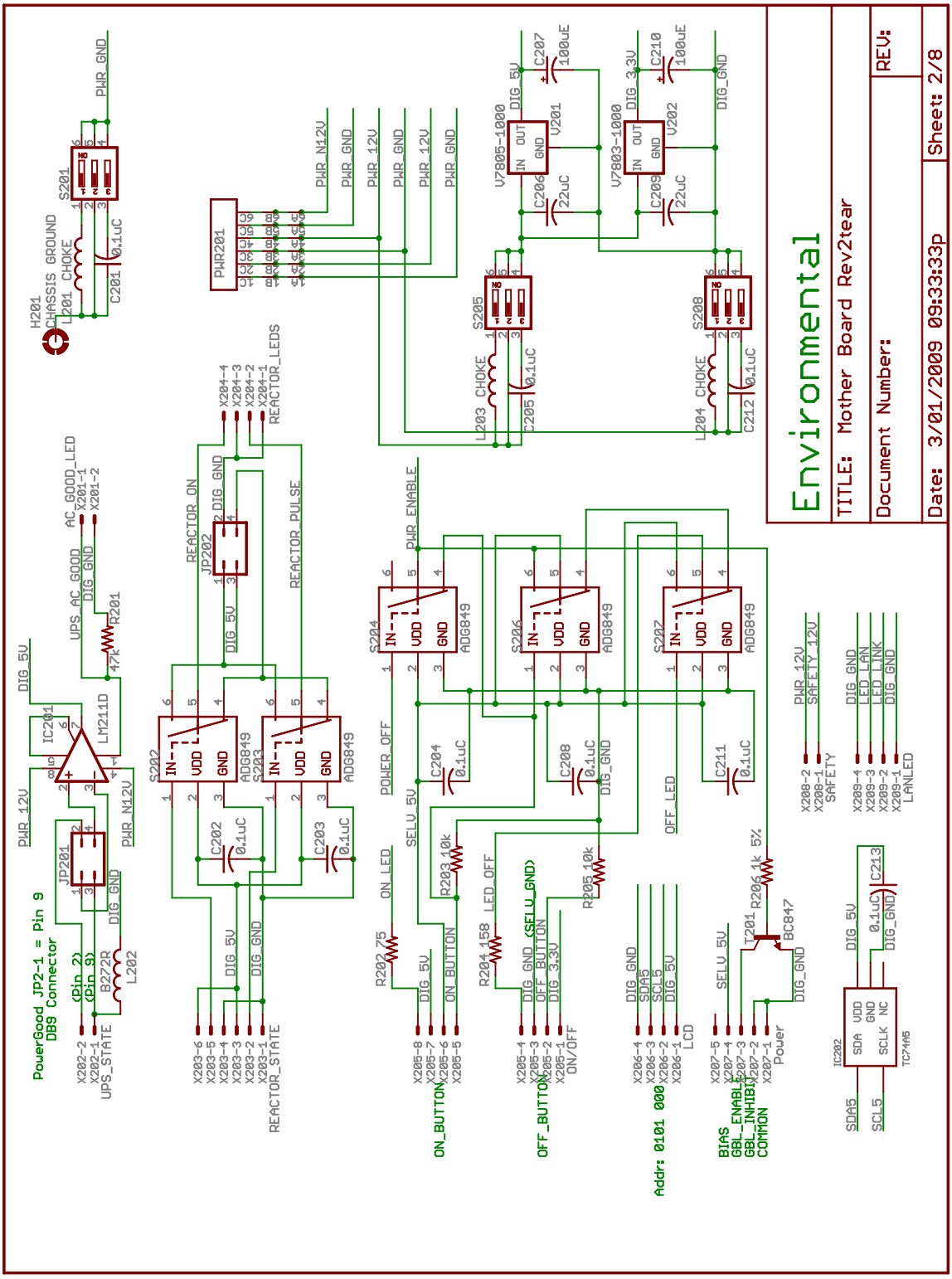


Figure D.26: Schematic for the mother board, page 2.

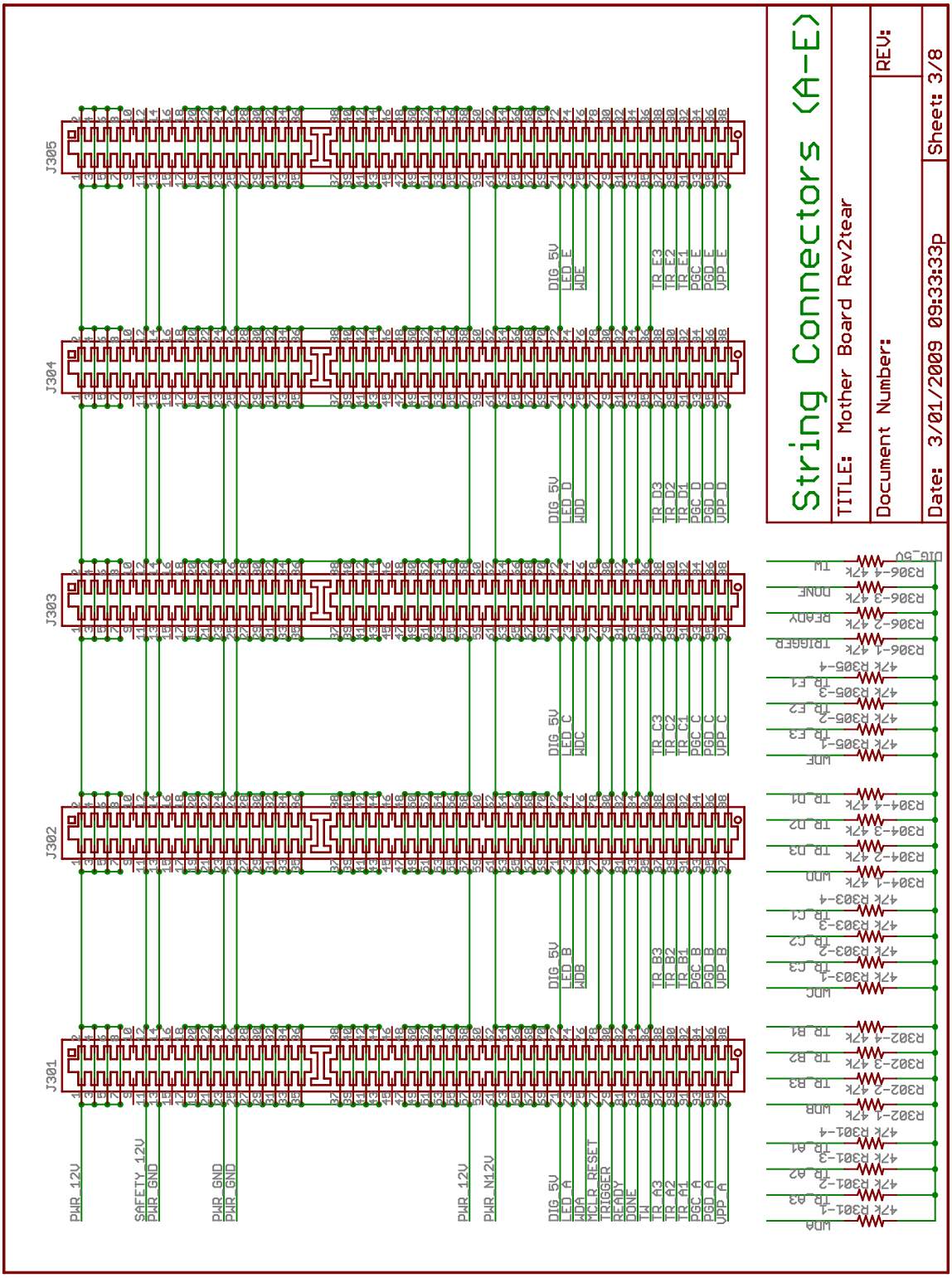


Figure D.27: Schematic for the mother board, page 3.

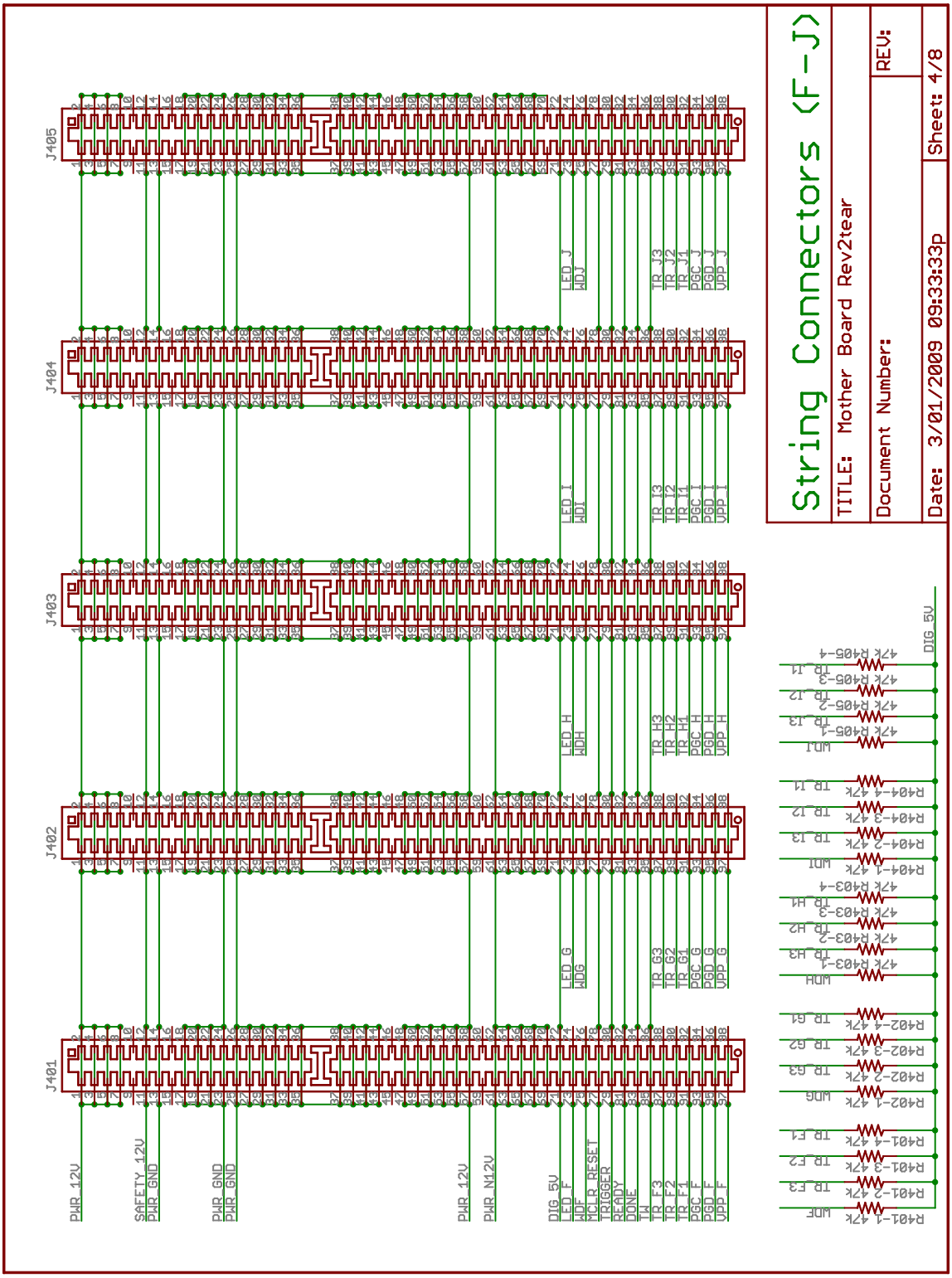


Figure D.28: Schematic for the mother board, page 4.

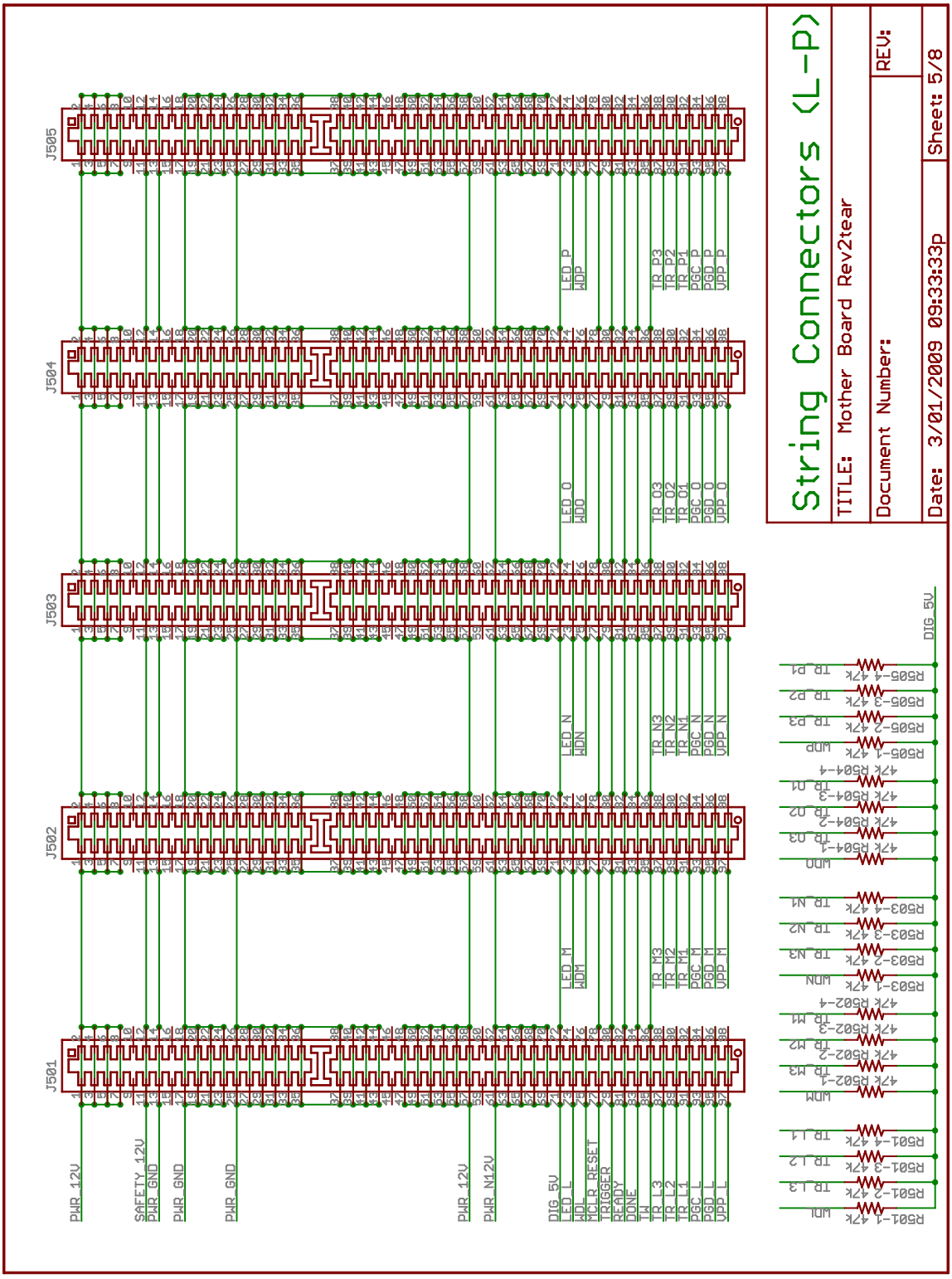


Figure D.29: Schematic for the mother board, page 5.

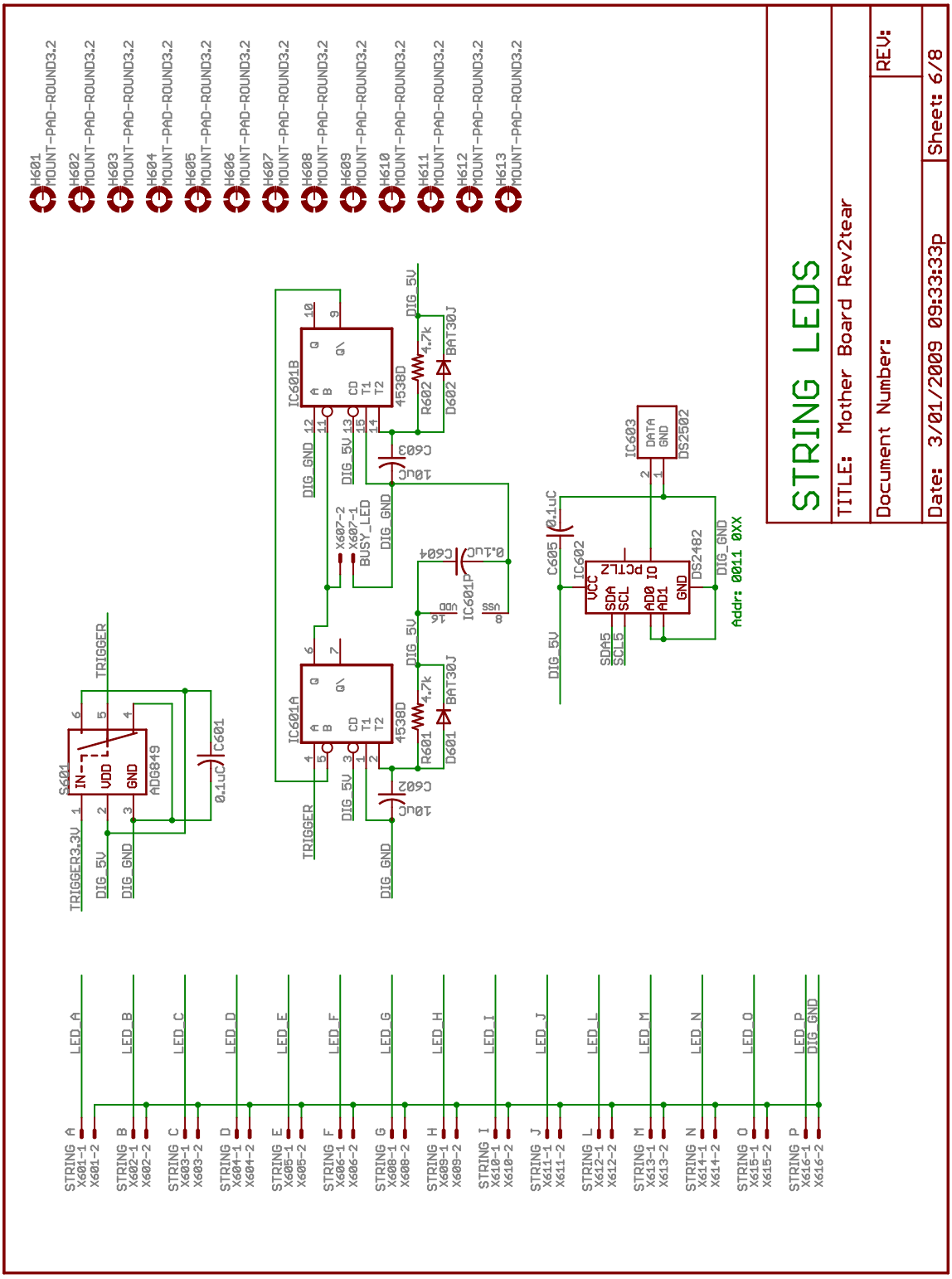


Figure D.30: Schematic for the mother board, page 6.

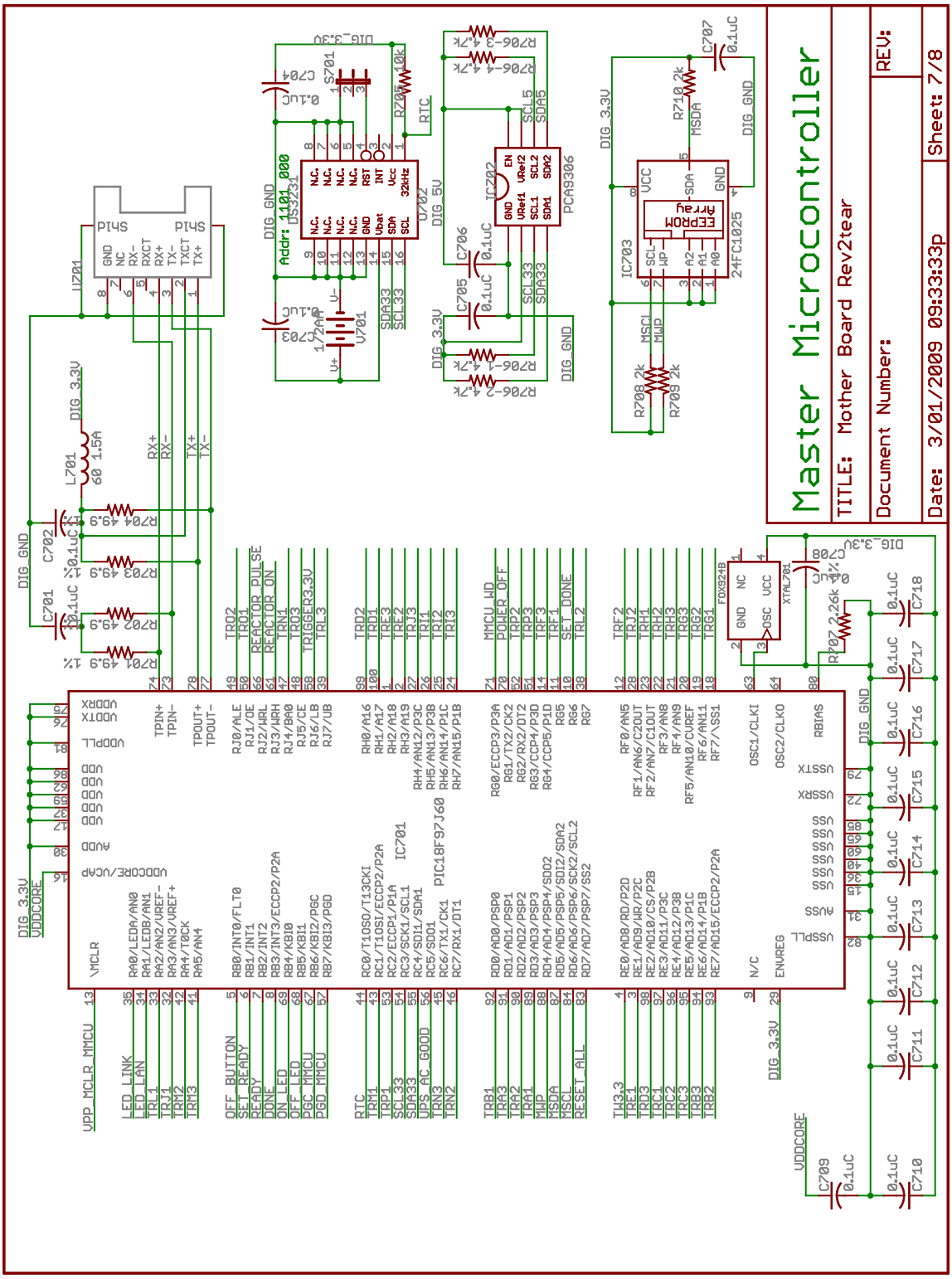


Figure D.31: Schematic for the mother board, page 7.

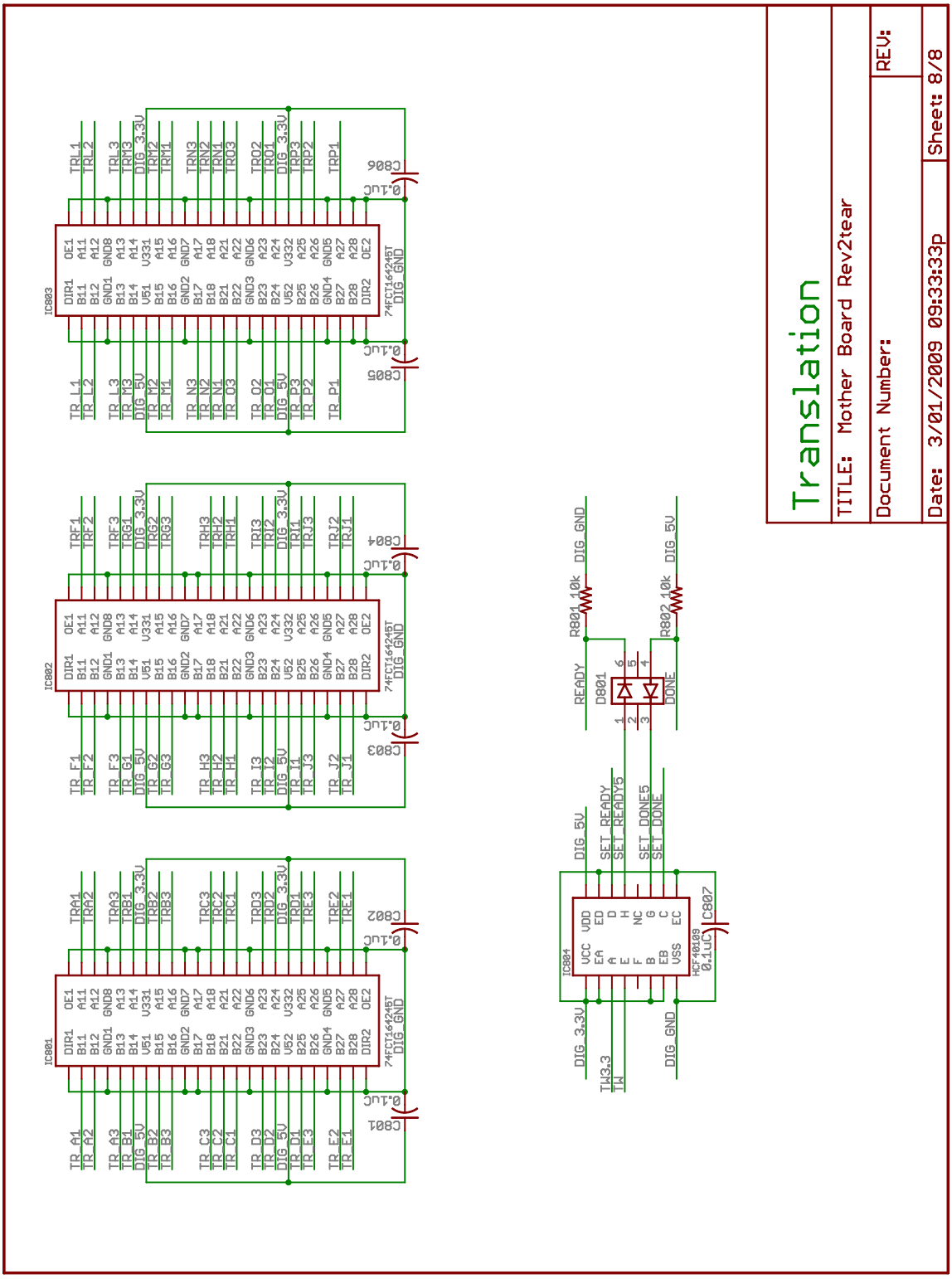


Figure D.32: Schematic for the mother board, page 8.

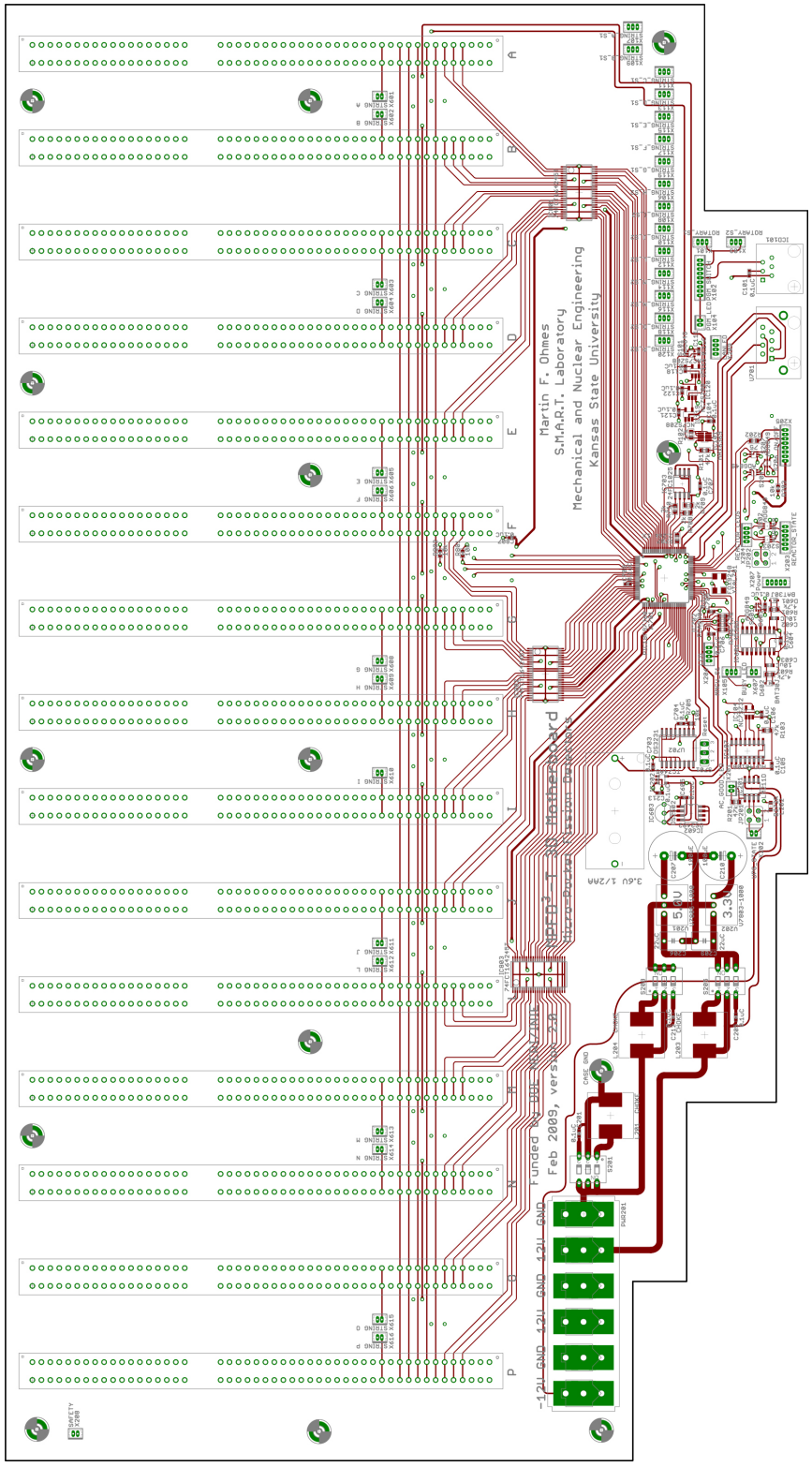


Figure D.33: Top side data plane, layer 1, layout for the mother board.

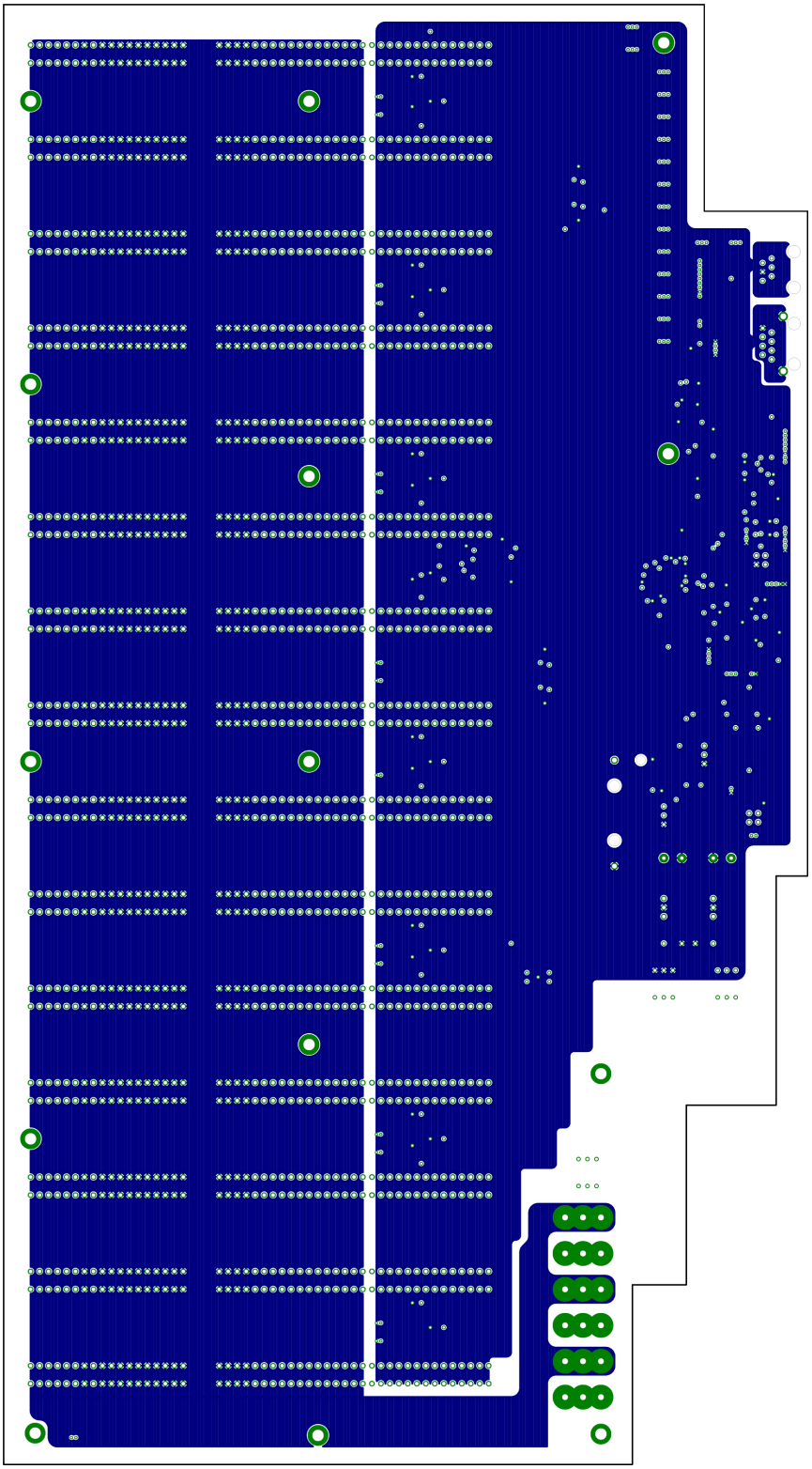


Figure D.34: Interior ground plane, layer 2, layout for the mother board.

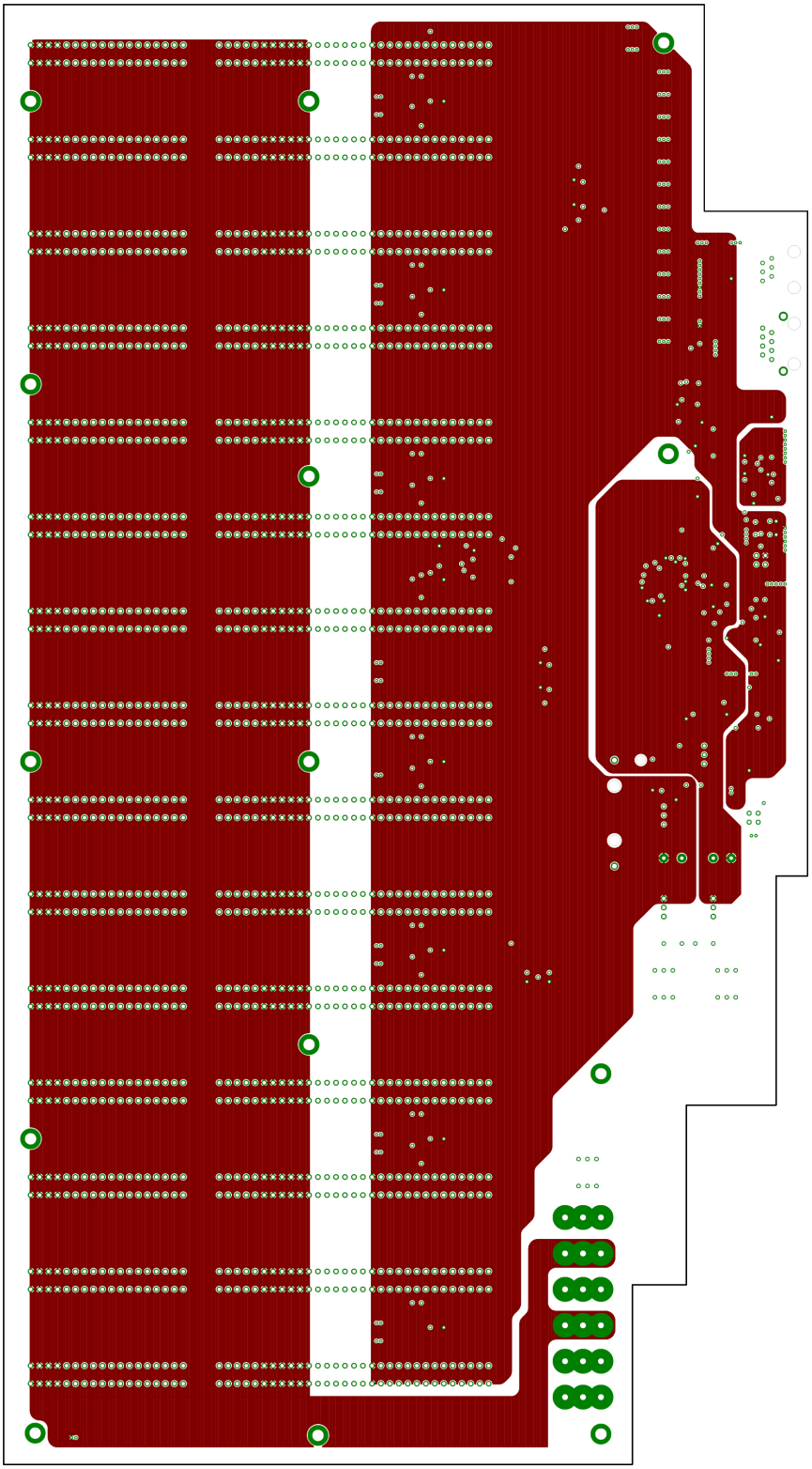


Figure D.35: Interior power plane, layer 3, layout for the mother board.

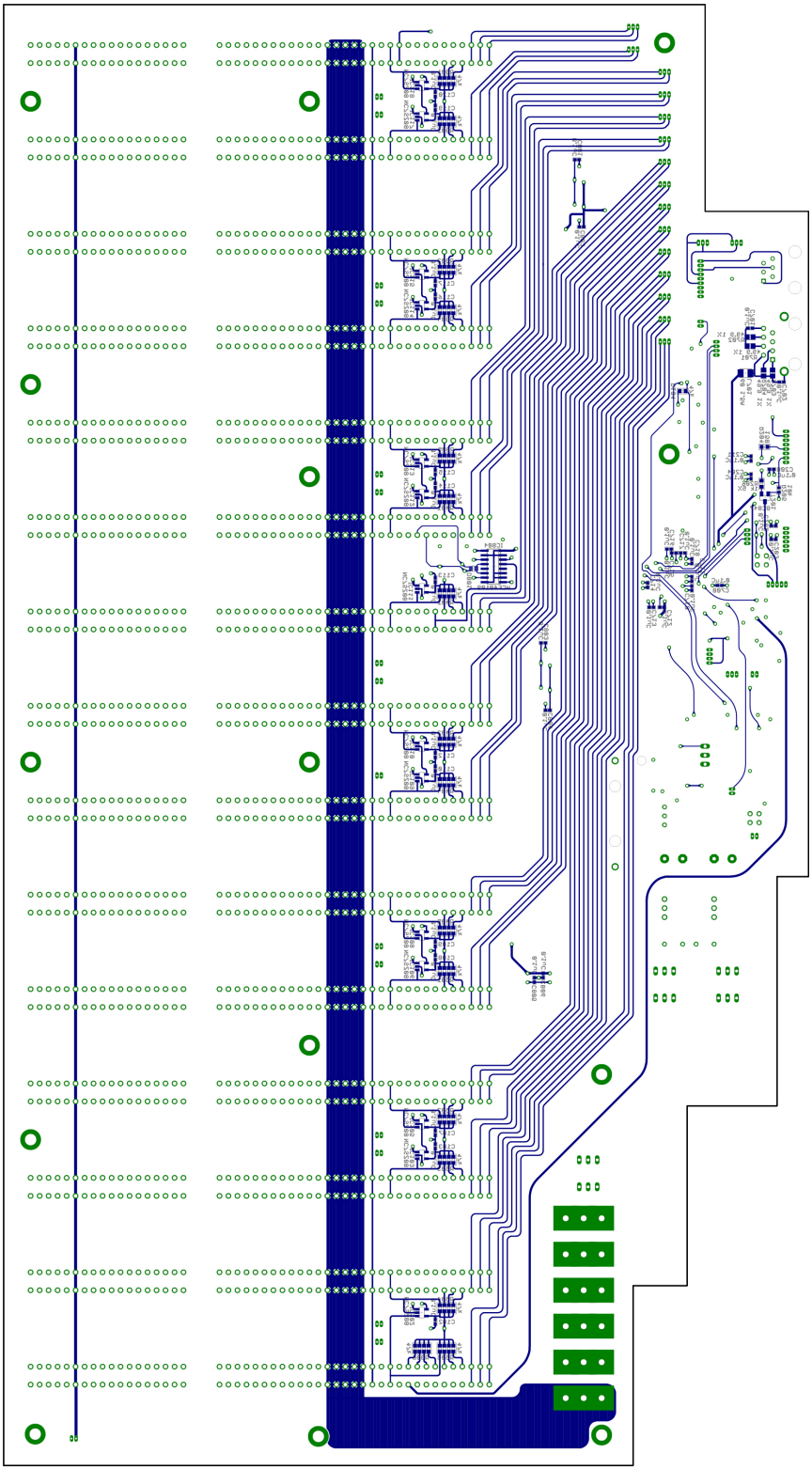


Figure D.36: Bottom side data plane, layer 4, layout for the mother board.

Table D.6: Bill of materials for the mother board, page 1.

Qty	Value	Device	Parts
3	DIP EXT RCKR	DIP03YL	S201, S205, S208
1	CKN1702-ND	JP2E	S701
2	JP2Q	JP2Q	JP201, JP202
59	0.1uC	C-USC0402K	C101, C102, C103, C104, C105, C106, C107, C108, C109, C110, C111, C112, C113, C114, C115, C116, C117, C118, C119, C120, C121, C122, C201, C202, C203, C204, C205, C208, C211, C212, C213, C601, C604, C605, C701, C702, C703, C704, C705, C706, C707, C708, C709, C710, C711, C712, C713, C714, C715, C716, C717, C718, C801, C802, C803, C804, C805, C806, C807
1	1/2AA	1/2AA	V701
1	1k 5%	R-US_R0603	R206
1	2.26k 1%	R-US_R0603	R707
3	2k	R-US_R0603	R708, R709, R710
1	4.7k	R-US4_R0603	R706
2	4.7k	R-US_R0603	R601, R602
5	10k	R-US_R0402	R203, R205, R705, R801, R802
2	10uC	C-USC0603	C602, C603
2	22uC	C-US050-050X075	C206, C209
1	24FC1025	24C04ASM	IC703
16	47k	R-US4_R0603	R301, R302, R303, R304, R305, R306, R401, R402, R403, R404, R405, R501, R502, R503, R504, R505
5	47k	R-US_R0603	R101, R102, R103, R104, R201
4	49.9 1%	R-US_R0805	R701, R702, R703, R704
1	60 1.5A	L-USL3225M	L701
3	74FCT164245T	74FCT164245T	IC801, IC802, IC803
1	75	R-US_R0603	R202
2	100uE	CPOL-USE5-13	C207, C210
1	158	R-US_R0603	R204
1	4538D	4538D	IC601
1	AC_GOOD_LED	53047-02	X201
7	ADG849	ADG849	S101, S202, S203, S204, S206, S207, S601
1	B272R	L-USL2012C	L202
2	BAT30J	DIODE-SOD323-R	D601, D602
1	BAT54-09P6FILM	BAT54-09P6FILM	D801
1	BC847	BC847	T201
15	BOARD-EDGE-CON-49	BOARD-EDGE-CON-49	J301, J302, J303, J304, J305, J401, J402, J403, J404, J405, J501, J502, J503, J504, J505
1	BUSY_LED	53047-02	X607
1	CHASSIS GROUND	MOUNT-PAD-ROUND3.2	H201
3	CHOKE	L-USPM12639S	L201, L203, L204
1	DS2482	DS2482	IC602
1	DS2502	DS2502	IC603
1	DS3231	DS3231	U702
1	FOX924B	FOX924B	XTAL701
2	HCF40109	HFC40109	IC107, IC804
1	ICD	ICD	ICD101
1	LANLED	53047-04	X209
1	LCD	53047-04	X206
1	LM211D	LM211D	IC201
1	MAX6369	MAX6369	IC101
1	MMCU_S1	53047-03	X105
13	MOUNT-PAD-ROUND3.2	MOUNT-PAD-ROUND3.2	H601, H602, H603, H604, H605, H606, H607, H608, H609, H610, H611, H612, H613
17	NC7SZ08	NC7SZ08	IC102, IC103, IC105, IC106, IC108, IC109, IC110, IC111, IC112, IC113, IC114, IC115, IC116, IC117, IC118, IC119, IC120
1	NC7SZ32	NC7SZ32	IC104

Table D.7: Bill of materials for the mother board, page 2.

Qty	Value	Device	Parts
1	ON/OFF	53047-08	X205
1	PCA9306	PCA9306	IC702
1	PGM_LED	53047-02	X104
1	PGM_SWITCH	53047-09	X102
1	PIC18F97J60	18F9XJ6X	IC701
1	PWR-6	PWR-6	PWR201
1	Power	53047-05	X207
1	REACTOR_LEDS	53047-04	X204
1	REACTOR_STATE	53047-06	X203
1	ROTARY_S1	53047-03	X101
1	ROTARY_S2	53047-03	X103
1	SAFETY	53047-02	X208
1	SI-60002	SI-10021	U701
1	STRING A	53047-02	X601
1	STRING B	53047-02	X602
1	STRING C	53047-02	X603
1	STRING D	53047-02	X604
1	STRING E	53047-02	X605
1	STRING F	53047-02	X606
1	STRING G	53047-02	X608
1	STRING H	53047-02	X609
1	STRING I	53047-02	X610
1	STRING J	53047-02	X611
1	STRING L	53047-02	X612
1	STRING M	53047-02	X613
1	STRING N	53047-02	X614
1	STRING O	53047-02	X615
1	STRING P	53047-02	X616
1	STRING_A_S1	53047-03	X107
1	STRING_B_S1	53047-03	X109
1	STRING_C_S1	53047-03	X111
1	STRING_D_S1	53047-03	X113
1	STRING_E_S1	53047-03	X115
1	STRING_F_S1	53047-03	X117
1	STRING_G_S1	53047-03	X119
1	STRING_H_S2	53047-03	X106
1	STRING_I_S2	53047-03	X108
1	STRING_J_S2	53047-03	X110
1	STRING_L_S2	53047-03	X112
1	STRING_M_S2	53047-03	X114
1	STRING_N_S2	53047-03	X116
1	STRING_O_S2	53047-03	X118
1	STRING_P_S2	53047-03	X120
1	TC74A5	TC74A5	IC202
1	UPS_STATE	53047-02	X202
1	V7803-1000	V7803-1000	V202
1	V7805-1000	V7805-1000	V201