RUMINANT GRAZING MONITOR

by

KANCHUAN HUANG

B.S., National Tsing Hua University
Hsinchu, Taiwan, 1981

A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

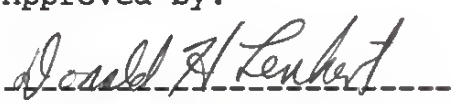
MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by:



Major Professor

ACKNOWLEDGEMENT

I would like to acknowledge my advisor Dr. Lenhert for his patient guidance and unvaluable effort he made on my Master's report. Also, I acknowledge Dr. Haft and Dr. Owensby for their serving of my committee.

Finally, I would like to acknowledge my parents and brothers for their encouragement that I can finish my academic work.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

The goal of this project was to design an electronic recording system to monitor and record grazing events. Currently, the Department of Agronomy is using a mechanical system to record the data of grazing events, but due to the mechanical limitation, the accuracy of the data is not good. In addition, the data can only be read out via human labor, i.e., measure the distance between lines drawn punched by the system to estimate the time period. To improve the reliability and resolution of data, we designed an electronic recording system, which provides better reliability and resolution of the data, and supports RS-232C interface to external computer, that will save time in converting the data to machine readable form.

CHAPTER II

SYSTEM REQUIREMENT

According to Dr. Owensby's letter[1], the following requirements are desired in an electronic recording system:

1. LOW POWER CONSUMPTION

The battery is the power source that we have, to extend the system life time, the lower power consumption, the better. The minimum duration that battery can support system power is nine days to two weeks.

2. TWO WEEKS ON BOARD MEMORY STORAGE CAPACITY

Usually, ruminant grazing events happen on an average of 100 times per day, therefore, two weeks storage capacity is same as 1400 events storage capacity.

3. MINIMUM INSTRUMENT SIZE

To avoid damage by the animal rubbing against objects, and for the possibility of implanting the instrument in ruminants in the future, the instrument should be as small as possible.

4. RS-232C OR HPIL INTERFACE

This interface is used to dump the stored data to a computer or to communicate with a computer.

5. TIME RESOLUTION

To record each grazing event, the resolution of the time will be one second.

CHAPTER III

DESIGN PROCEDURE

To minimize instrument size (unit on ruminant) and reduce power consumption, the prototype was separated into two parts. One is "Unit on Ruminant", the other one is "External Data Dumping Circuitry". Only those components which are necessary in monitoring and recording grazing events were put on "Unit on Ruminant", the rest (data transmission buffers, LED, switches, RS-232C interface connector, etc.) were put on "External Data Dumping Circuitry".

3.1 HARDWARE DESIGN

3.1.1 System Block Diagram

From Fig. 1, we know that this system is composed of three units: Unit on Ruminant, External Data Dumping Circuitry, and Computer. Usually, only "Unit on Ruminant" is fastened on the ruminant's neck. Whenever the stored data is needed, these three units should be connected together to transfer stored data to computer.

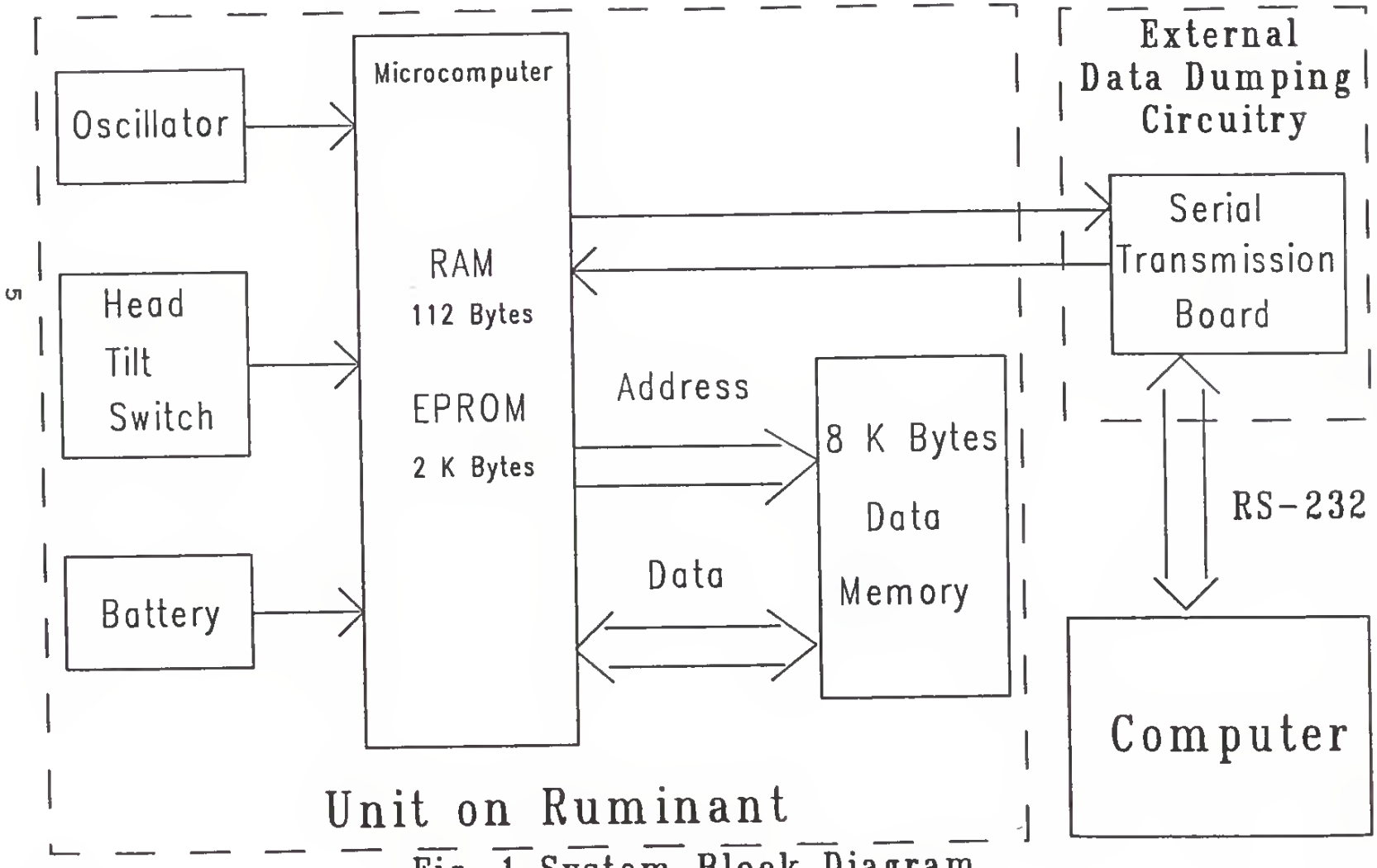


Fig. 1 System Block Diagram

3.1.2 Circuit Description

a. **CPU**

To reduce system power consumption and minimize instrument size, the Motorola's CMOS 8-BIT EPROM MICROCOMPUTER MC1468705G2 (Fig 2.) was chosen as CPU. The following are the major features of the MC1468705G2 EPROM MCU:

1. Typical full speed operating power of 20 mW at 5 V.
2. 112 bytes of on-chip RAM: used for software variable.
3. 2106 bytes of UV erasable, user programmable ROM(EPROM): used to store system firmware (driver).
4. 32 bidirectional I/O lines: used as data lines, address lines, serial I/O lines, control lines, switch status lines, etc. See appendix B.
5. Internal 8-bit timer with software programmable 7-bit prescaler.
6. External and timer interrupts: used for software time clock.
7. Single 3 to 5.5 V supply.
8. Self-programming bootstrap program in ROM simplifies EPROM programming.

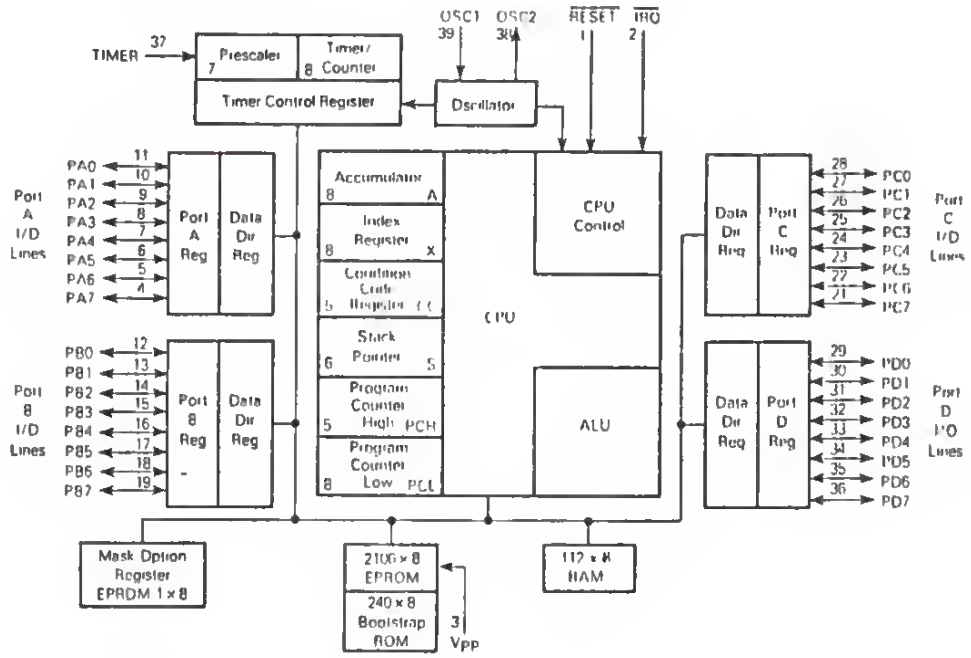


Fig 2. CPU Block Diagram (Courtesy of Motorola[2])

b. **MEMORY**

To meet two-week memory storage capacity, an 8 K bytes static RAM was selected.

The data of each grazing event is composed of two parts, one is the time the head went down, the other one is the event period. Usually, ruminant grazes around 15 minutes each time. To optimize the memory usuage, data are recorded in the following format :

Record-Name	Range	Memory-Byte-Occupied
DAY	1-31	1
HOUR	0-23	1
MIN	0-59	1
SEC	0-59	1
P_MIN	0-59	1
P_SEC	0-59	1

Event Start Time (The time head went down)

DAY: date of month

HOUR: hour

MIN: minute

SEC: second

Event Period

P_MIN: minutes of event period

P_SEC: seconds of event period

Totally, 6 bytes are required for each event. With 8 K memory, the system provides:

8192 / 6 = 1365.3 --> 1365 event storage capacity

1365 / 100 --> Approximately two weeks storage capacity

c. **OSCILLATOR[3]**

The two factors which greatly affect CMOS power consumption are supply voltage and operating frequency. Reducing the supply voltage (Vdd) proportionally reduces power consumption since the EI product is lower.

Each CMOS cell is basically composed of two complementary transistors (a P channel and an N channel), and, in the steady state, only one transistor is turned on. The active P-channel transistor sources current when the output is a logic high, and presents a high impedance when the output is a logic low. Thus, the overall result is extremely low power consumption because there is no power loss through the active P-channel transistor. Since only one transistor is turned on during the steady state, power consumption is determined by leakage currents.

During a transition, both transistors pass through the active regions of their operating characteristics. The actual time spent simultaneously in these active regions directly affects the power consumption. The higher the operating frequency, the more time is spent in

these simultaneous active regions, thus, higher power consumption. By reducing the number of transitions within the CMOS device, power consumption can be reduced.

In order to make operating frequency as low as possible and still get resolution required, we chose a 32.768 kHz oscillator instead of using recommended 1 MHz and 4 MHz high frequency oscillator.

At this frequency, internal oscillator requiring only an external crystal is not reliable, instead, an external oscillator is required. The MC14069 CMOS hex inverter[3] was chosen for this circuit because of its low power consumption, and its ability to operate in the linear region with reasonable stability. As will be seen later, only two resistors are required for the oscillator: bias resistor R1 ensures linear operation and R2 provides current limiting protection for the crystal. Two load capacitors (C1 and C2) ensure proper loading plus correct start-up frequency. Variable capacitor C1 also allows limited tuning of the output frequency.

d. **SWITCHES**

Head Tilt Switch

This mercury bulb switch is used to detect whether ruminant is grazing or not. When the ruminant's head goes down, the mercury will flow to and stay at side B

(Fig 3.(a)). This closed-switch generates a logic low signal to CPU when placed in series with a pull-up resistor. That switch position assumes a grazing activity. When the ruminant's head is raised up, the mercury flows to side A (Fig 3.(b)). This opened-switch results in a logic high signal to the CPU.

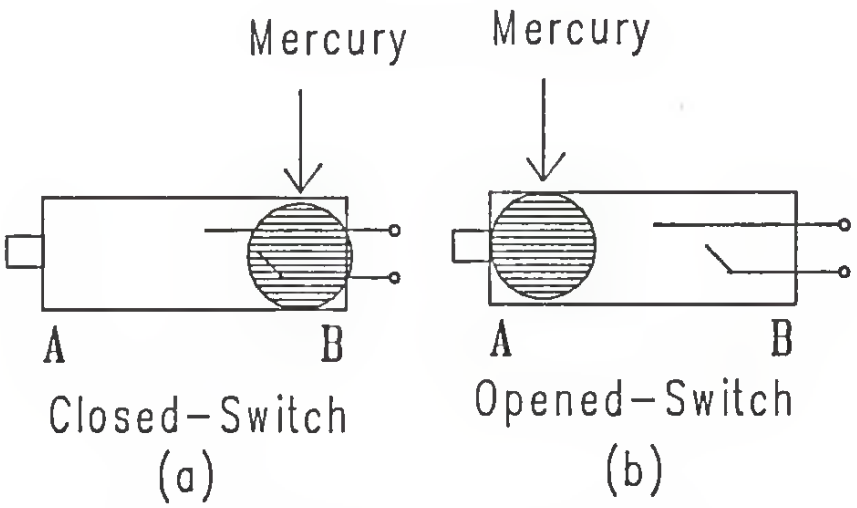


Fig 3. Mercury Bulb Switch

Based on the above assumption, the software program uses this feature to record start time and count the event period. The algorithm is, poll this /FEED signal, store it in memory as OLD_FLAG, poll it again, store this signal as NEW_FLAG, keep polling, updating, and comparing the

NEW_FLAG and OLD_FLAG, we can tell grazing activity as following (Fig 4.):

OLD_FLAG		NEW_FLAG	ACTIVITY
HIGH	-->	LOW	Event Start
LOW	-->	HIGH	Event End
LOW	-->	LOW	Event Undergoing
HIGH	-->	HIGH	No Grazing Event

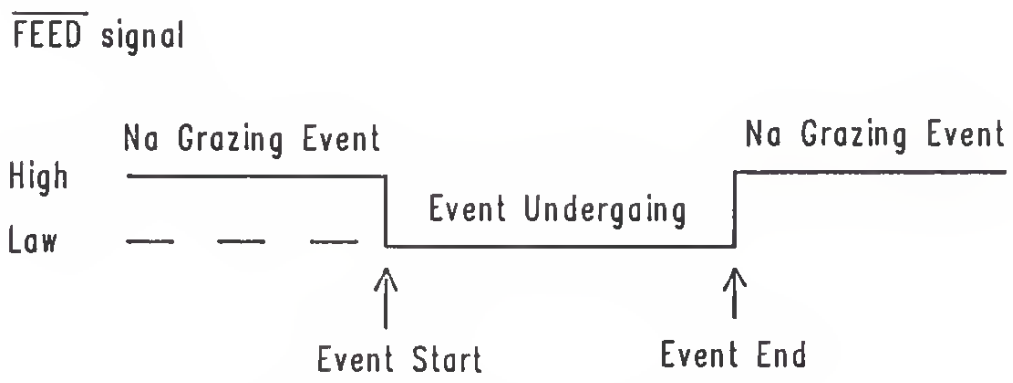


Fig 4. Grazing Event Detection

Dump Switch

This switch provides the user with the opportunity to transfer the stored data to an external computer via an RS-232C interface. Whenever the user transfers the data, or before taking the ruminant grazing monitor away from the animal, the user should set the switch to DUMP MODE. The CPU will record this request, stop monitoring grazing

events (but keep software time clock operating), and wait for a command from the external computer.

Reset Switch

This temporary switch provides the user with the opportunity to restart the system, whenever the user feels something is wrong with the system, or after changing the battery.

e. **RS-232C INTERFACE**

To simplify the hardware circuit, the conversion between parallel data and serial data is implemented via software. Two I/O lines of CPU are used as transmitting and receiving lines. In addition, 1488 and 1489 buffers are used to transform the voltage between TTL logic levels and RS-232C logic levels (+ 12v, - 12v).

f. POWER SOURCE

The power source used initially is three 1.5 volts AA alkaline batteries connected in series. For extended use four Ni-Cad AA batteries may be used since these are rechargable.

g. The schema of the complete ruminant grazing monitor is given in Fig 5.

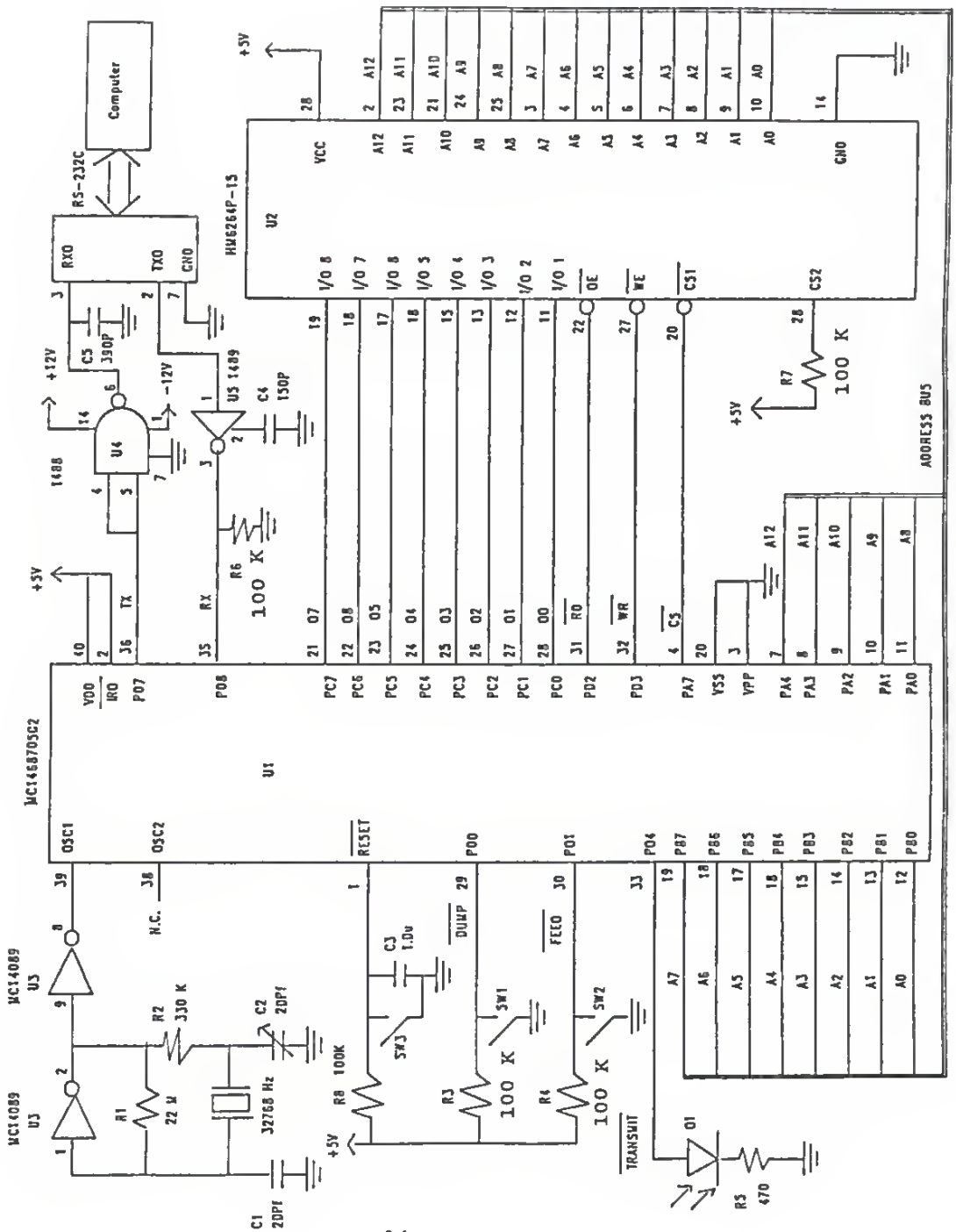


Fig 5. Hardware Circuit

3.1.3 Power Consumption Estimation

To choose an appropriate battery that can support system operation and keep data for two weeks, an estimation of system power consumption was made. From Fig. 5, it can be seen that the battery power is mainly consumed by three components: CPU MC1468705G2, SRAM HM6264P-15 and hex inverter MC14069. The maximum power supply current of these three chips is shown below.

MC1468705G2[2] :

$$I1 = \frac{10\ mA}{30} = 0.333\ mA$$

Where 10 mA: MAX supply current under 1 MHz operation frequency.

30: Due to lower frequency 32768 Hz.

HM6264P-15[10] :

$I2 = 3\ mA$

MC14069[9] :

$I3 = [\ (\ 0.3\ uA/KHz\ )\ f + IDD/6\ ]\ x\ 2\quad uA$

$= [\ (0.3\ x\ 0.001)\ (33) + (1.0\ x\ 0.001)/6\ ]\ x\ 2\quad mA$

$= 0.020\ mA$

Where 33: Due to 32.768 KHz

2: Two gates used.

Total supply current :

$$I = I1 + I2 + I3$$
$$= 0.333 + 3 + 0.020$$
$$= 3.353 \text{ mA}$$

Maximum time duration :

$$24 \times 14 = 336 \text{ hours}$$

Total energy :

$$3.353 \times 336 = 1126.6 \text{ mAmp-hour}$$

So, a battery with 1.2 Amp-hour is what we need for worst case. AA batteries will not supply this amount of current. Since worst case currents were assumed, the currents were measured in the lab. The result was 0.7 mA instead of 3.35 mA. This indicates that the AA batteries should last two weeks.

3.2 SOFTWARE DESIGN

3.2.1 On Chip EPROM Assembly Program

a. System Initialization

To initialize this system, we need to program CPU's Data Direction Register (DDR) for each I/O port, i.e., PORT A, PORT B, PORT C and PORT D. A pin is configured as an output pin if its corresponding DDR bit is set to a

logic one. A pin is configured as an input pin if its corresponding DDR bit is cleared to a logic zero. Also, the Mask Option Register (MOR), Timer Control Register (TCR), and Timer Data Register (TDR) should be appropriately programmed (appendix B).

In order to provide an interrupt at a one hertz rate it is necessary to set the timer prescaler to the appropriate value.

$$\frac{32768\ \text{Hz}}{2 * 128 * 128} = 1\ \text{Hz}$$

Where 32768 Hz: Crystal oscillator frequency
2 : Divide-by-2 oscillator clock
128 : Prescaler
128 : Timer data register value

By programming the timer this way, an interrupt request to the CPU will be generated every second. This is needed to provide a software-handled clock.

b. RS-232C Implementation

The RS-232C implementation uses CPU I/O port D bit 7 as serial output line, and bit 6 as serial input line. To determine the baud rate, the main loop of receiving data program's duration must be considered. The program is

followed by the number of clock cycles in parenthesis after the semicolon.

```
RX_3 BSR  DELAY_RX        ;(6) 33 MPU CYCLES DELAY
     BRCLR  IN,PORT_D,RX_4;(5) TEST SERIAL INPUT LINE
                          ;AND SET C-BIT
     ROR  CHAR            ;(5) ADD THIS BIT TO THE BYTE
     DECX                 ;(3)
     BNE  RX_3            ;(3) MORE BITS TO GET ?
```

Baud Rate = 1 sec / ((6 + 33 + 5 + 5 + 3 + 3) * Tcyc)

Tcyc = 1 / (32768 / 2) sec

==> Baud Rate = 297.9

So, the external computer RS-232C parameters are set up as :

Baud Rate: 300

No. of Stop Bit: 1

Parity: Even

c. Memory (SRAM) Read/Write Sequence

The 8K memory (SRAM) read/write operation is implemented by programming CPU I/O port, i.e., sending the address, data (write only), and control signals to the

corresponding ports and pins in an appropriate sequence. In memory read case, data from data lines are read on appropriate time after the address and control signals are correct. The sequences follow:

Read:

1) Send address A0 - A7
2) Send address A8 - A12 and enable /CS
3) Enable /RD
4) Read in data D0 - D7
5) Disable /RD
6) Disable /CS

Write:

1) Send data D0 - D7
2) Send address A0 - A7
3) Send address A8 - A12 and enable /CS
4) Enable /WR
5) Disable /WR (data was caught by memory)
6) Disable /CS

d. Flowchart

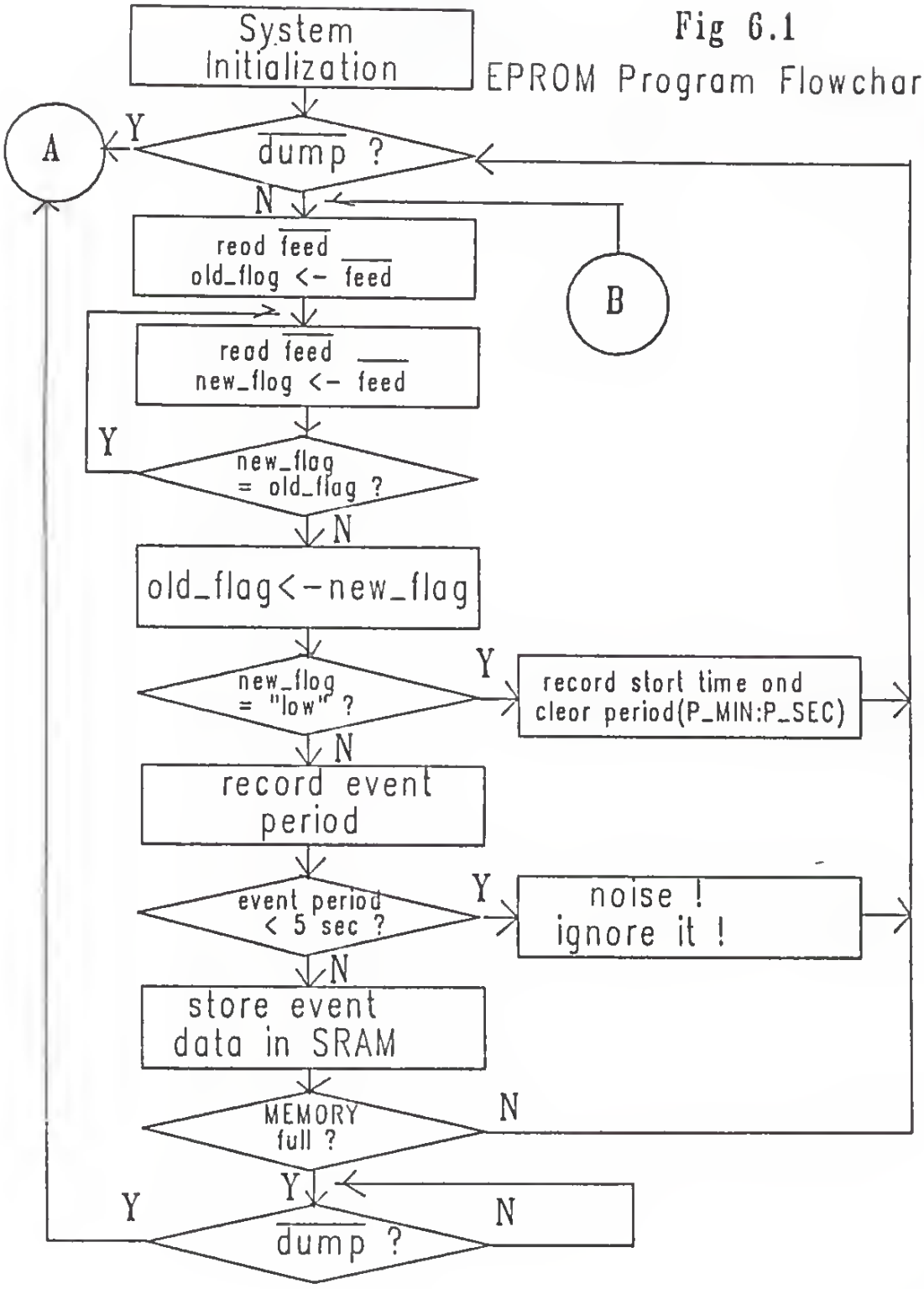
The EPROM program main flow is shown in Fig 6.1 and Fig 6.2. Generally, this program can run in two different modes. One is Normal Mode, the other one is Dump Mode. To switch or change operation mode, just set the Dump

Switch to the mode needed.

In Normal Mode, the program keeps monitoring grazing events and recording data. Also, it keeps polling the /DUMP signal to see if the operating mode has been switched. In Dump Mode, the program stops monitoring the grazing events, and receives a command from the computer and executes it.

Sometimes, the animal may throw its head around and cause the mercury switch to change status. To avoid this incorrect data, any event which has a duration of less than 5 seconds will be ignored by the program. Fig 6.3 and Fig 6.4 show the flowchart of the timer interrupt routine.

Fig 6.1
EPROM Program Flowchart



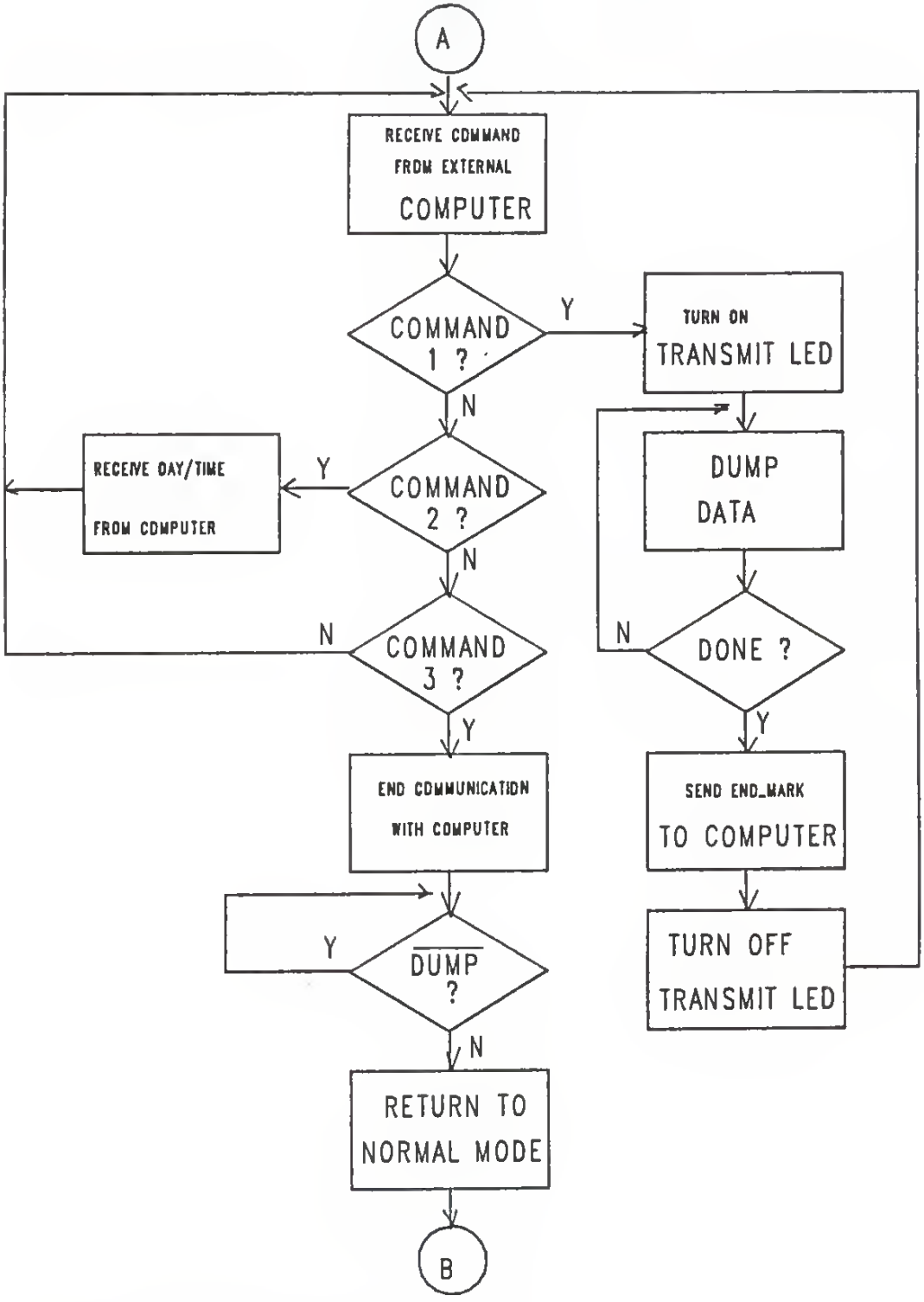


Fig 6.2 EPROM Progrom Flowchort (Cnt'd)

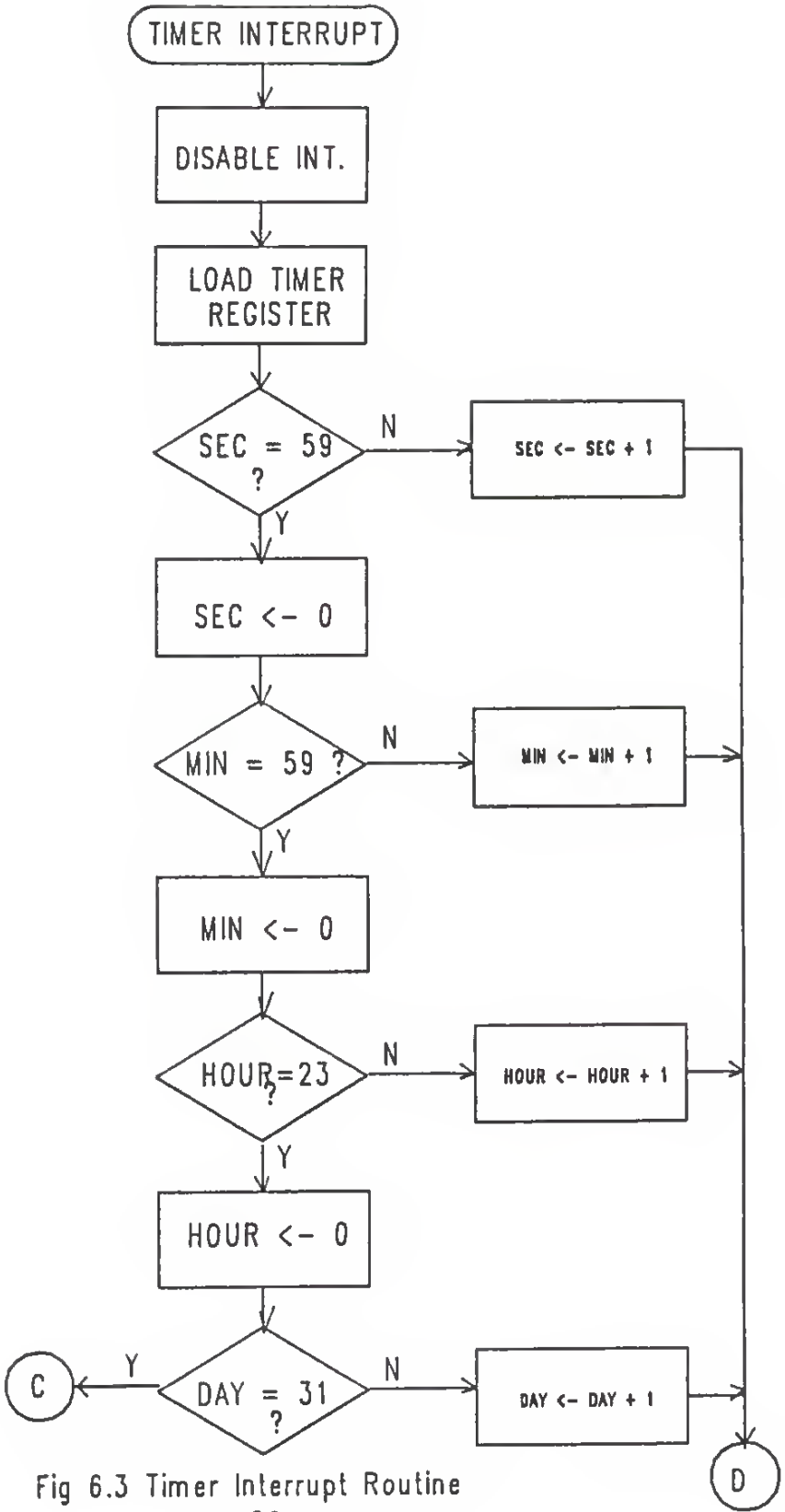


Fig 6.3 Timer Interrupt Routine

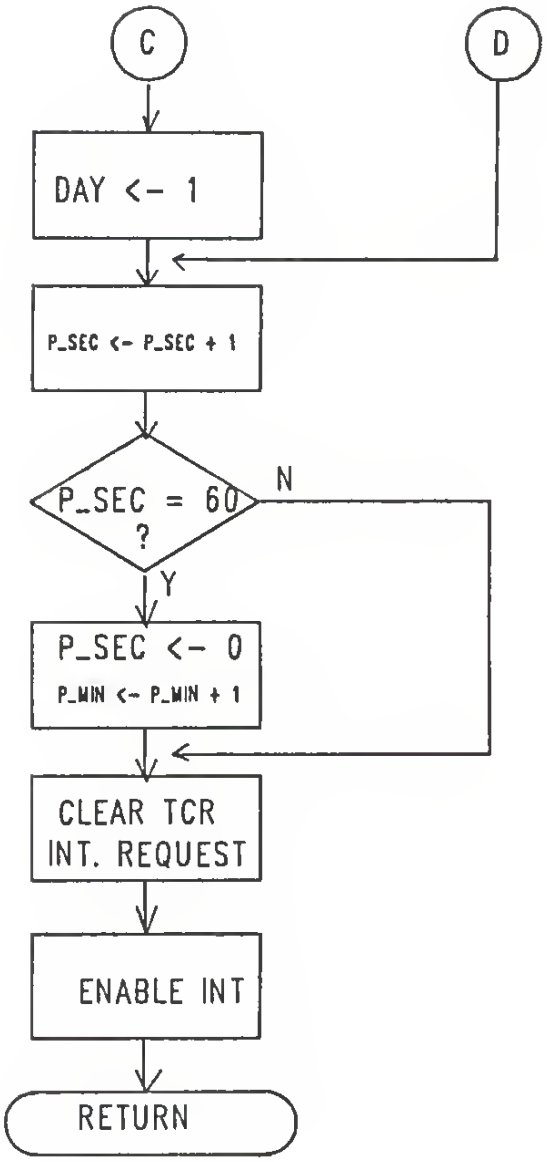


Fig 6.4 Timer Interrupt Routine (Cnt'd)

e. EPROM Programming Procedure

To program the CPU on-chip EPROM, there is a certain procedure that must be followed becaused (Fig 7.):

1. There is no MC1468705G2 cross assembler on the campus main frame IBM 370.
2. The author didn't have a PC available that was connected to digital switch.
3. There is no direct connection between the PC and DATA I/O EPROM Programmer.

3.2.2 External Computer BASIC Program

The purpose of this BASIC program is to test the software and hardware function of prototype. This program provides user functions as follows:

1) Dump data (Ruminant Grazing Monitor --> Computer)
2) Set day/time (Computer --> Ruminant Grazing Monitor)
3) Make a hard copy of obtained data.

Main menu of this program is shown in Fig 8., and the flowchart is given in Fig 9.

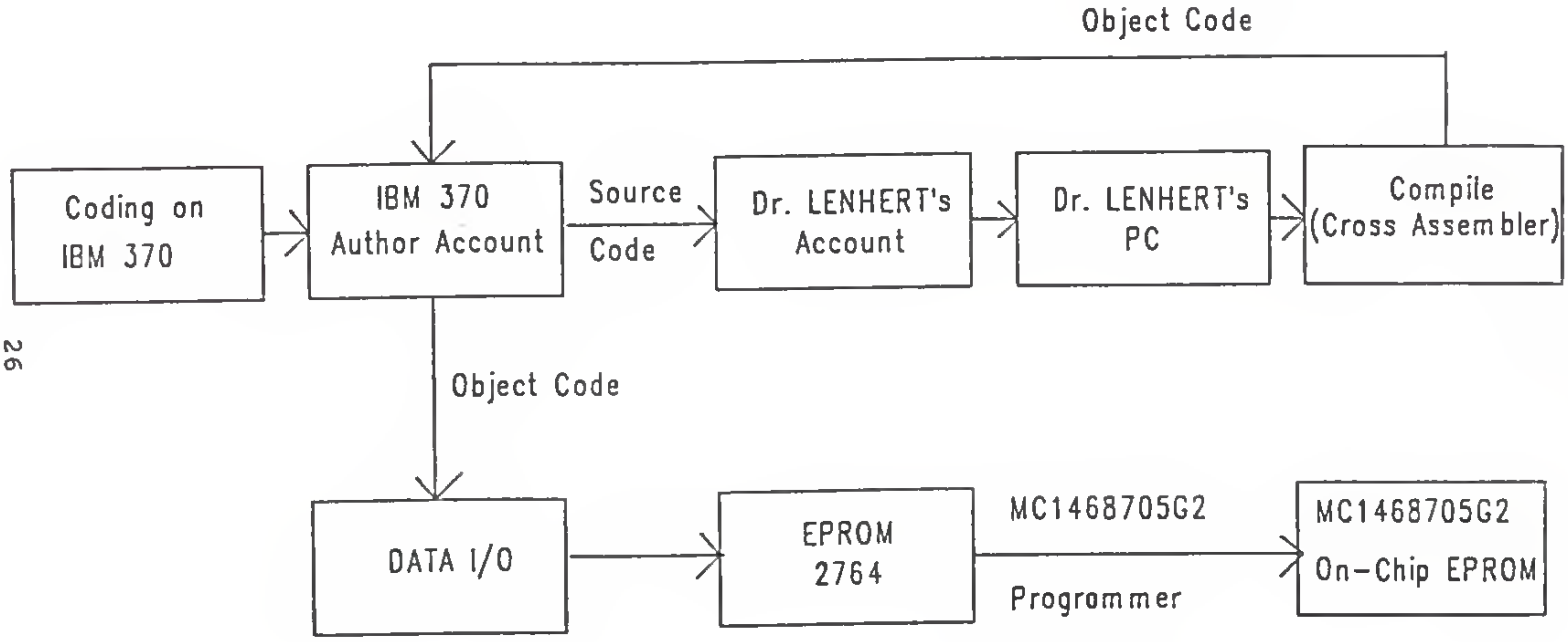


Fig 7. EPROM Programming Procedure

```
****************************************
*        RUMINANT GRAZING MONITOR      *
****************************************


1. DUMP DATA (RUMINANT GRAZING MONITOR --> Z100)
2. SET DAY/TIME
3. END COMMUNICATION
4. PRINT DATA (HARD COPY)
5. EXIT

PLEASE ENTER THE CHOICE (1/2/3/4/5)
```

Fig 8. Main menu of BASIC program

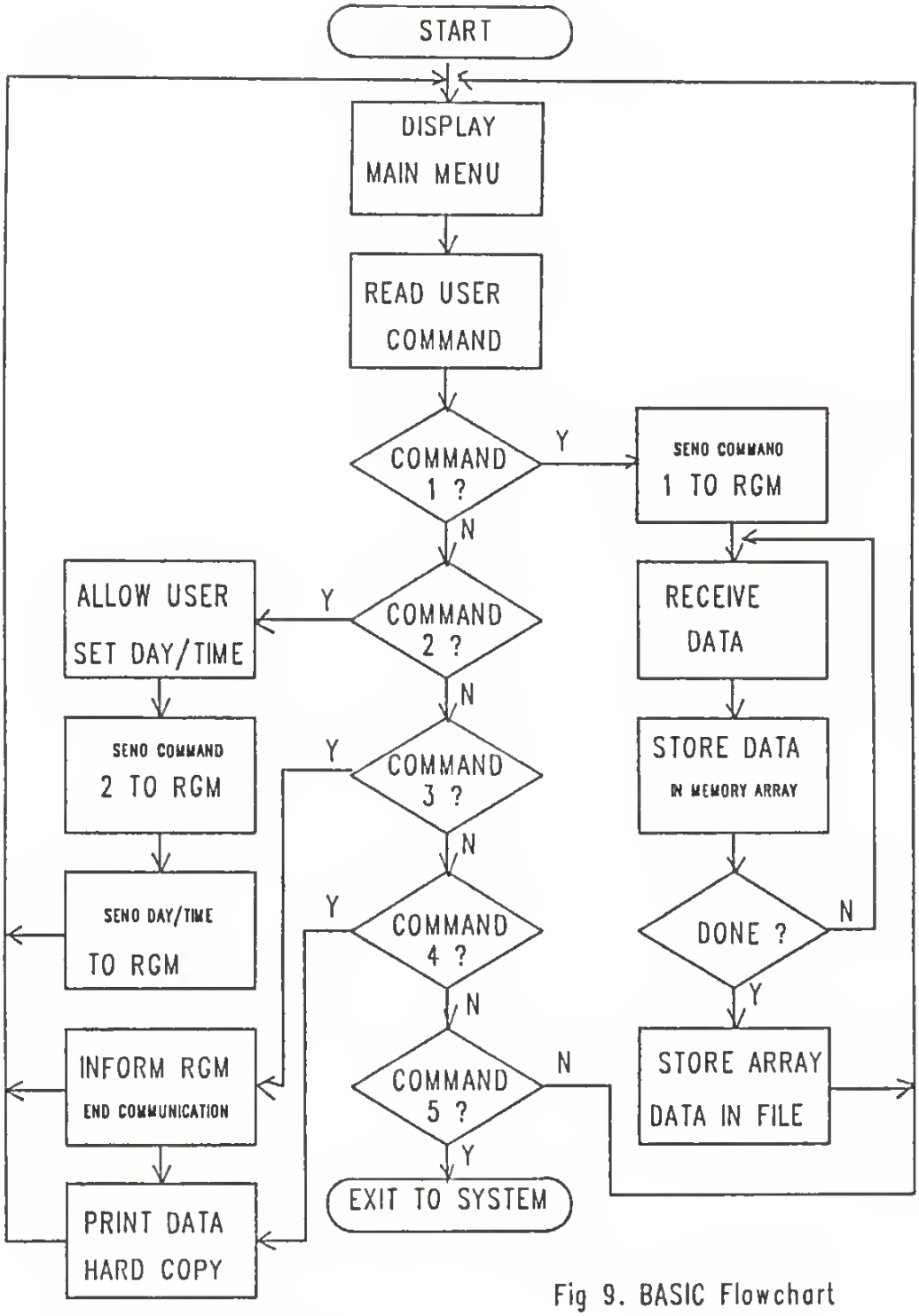


Fig 9. BASIC Flowchart

RGM: Ruminant Grazing Monitor

3.3 SYSTEM SIMULATION TEST

The system simulation test that were run were:

1. Test software time clock.
2. Test function of data storage.
3. Test RS-232C interface.

In the usual case, it takes two weeks to fill up the memory. To test the system software and hardware, we need to go through the program completely. We ran a simulation test, i.e., instead of monitoring the mercury switch indicating a grazing event, we supplied a logic signal from a pulse generator. To measure the rapidly changing signal (1.3 Hz), we also increased up the timer interrupt frequency to 32 Hz. In this simulation test, maximum number of events was set to 1280 which required 7680 bytes of memory.

Theoreotical expectations of the simulation test were:

1) Square wave frequency

   1.3 Hz

2) Time need to fill up 7680 bytes memory

   (1 / 1.3 Hz) * 1280 = 984.6 sec

   = 16 minutes 24.6 sec

Note: 1. 1.3 Hz is square wave frequency generated by the pulse generator.

2. 1280 is the maximum number of recorded event.

Obtained test result:

1) Data Listing

(Event Start Time)		(Event Period)
Date	HH:MM:SS	P_MIN:P_SEC
30	20:30:00	00:08
30	20:30:25	00:08
30	20:30:50	00:08
30	20:31:15	00:07
|	|	|
|	|	|

From above listing, we found the event period was 8 units (1/32 Hz), and events repeated at a frequency of 25 units (Fig 10).

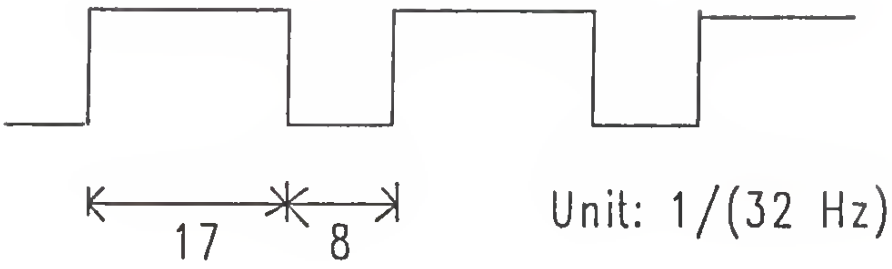


Fig 10. Measured Square Wave for Simulation Test

Measured square wave frequency

= 1 / ((25) (1/32 Hz))

= 1.28 Hz

The difference from the expected 1.3 Hz is due to the low sampling rate (32 Hz) and poor stability of the pulse generator.

2) Time spent to fill up 7680 bytes memory:

16 minutes 29 seconds

The difference from the expected time of 16 minutes 24 seconds was probably caused by inaccuracies in the frequency measurement or stability of the pulse generator, and possibly not accurate by reading the watch.

From the simulation test and obtained data listing, we are confident that the system was running in the way we expected.

CHAPTER IV

USER MANUAL

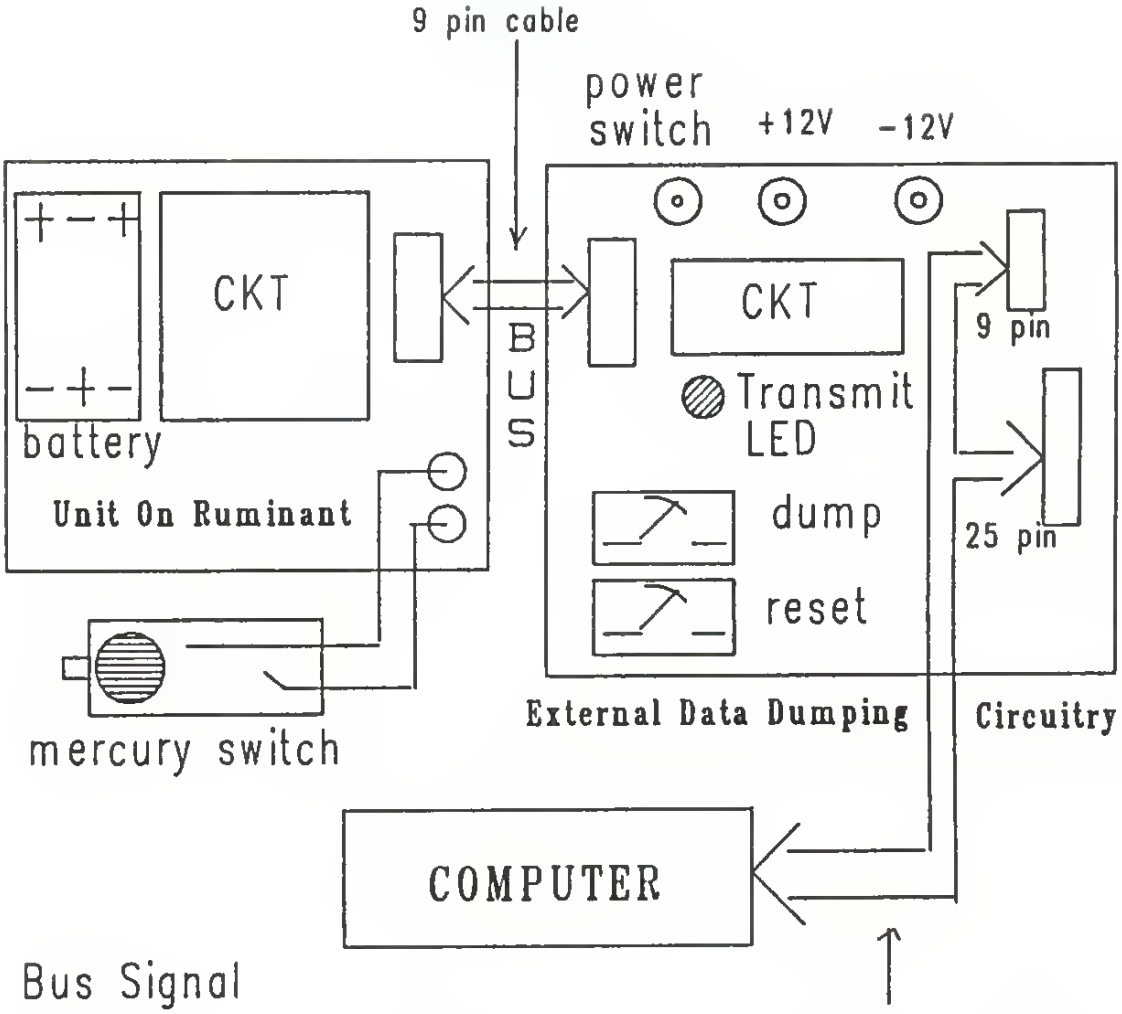
4.1 SYSTEM INSTALLATION

Before starting set up system, the user should get following components ready (Fig 11.):

1. Unit on ruminant.
2. External data dumping circuitry.
3. Three "AA" size, 1.5 volts batteries.
4. Mercury bulb switch with cable.
5. 9 pin bus cable with male connector on both end.
6. 25 pin (or 9 pin) RS-232C interface cable with male (or female) connector on one end, and the other end connector type depends on RS-232C port of computer.
7. Power supply with + 12v, - 12v voltages.
8. Computer with RS-232C port.
9. Grazing monitor program on the computer.

Then, connect the system follow listed procedures:

1. Set "power switch" of External Data Dumping Circuitry off.
2. Set "dump switch" to right side (dump mode).
3. Connect 9 pin bus cable between "Unit on Ruminant" and "External Data Dumping Circuitry".
4. Connect RS-232C interface cable between computer and



Bus Signal

1. TX
2. RX
3. /RESET
4. /DUMP
5. Transmit
6. GND
7. (unused)
8. (unused)
9. + 5 V

Fig 11. Prototype Scheme

"External Data Dumping Circuitry".

5. Plug in + 12v, - 12v, GND power source to the "External Data Dumping Circuitry".
6. Connect the two wires of mercury bulb switch to the "Unit on Ruminant".
7. Put three batteries in holder follow polarity marked on holder.
8. Turn the power switch of "External Data Dumping Circuitry" on.
9. Put the "reset" switch of "External Data Dumping Circuitry" to right side for a second.

At this time the system (Unit on Ruminant) is operating in DUMP MODE, and waiting for command from the external computer.

4.2 OPERATION PROCEDURE

4.2.1 Set System Day/Time

These steps are used to set the correct Day/Time for the system software time clock.

1. Load and execute BASIC program on computer side. The menu is displayed on the computer (Fig 12.).
2. Enter command 2 to set Day/Time.
3. Follow the instruction on screen to enter current day,

hour, minute, second.

4. Computer transfers user entered Day/Time to the Grazing Monitor. At same time, transfer is indicated by the Transmit LED being on.
5. Transmission is completed and Transmit LED is turned off.
6. Enter command 3 to inform the grazing monitor that communication is ended.
7. Set "power switch" of "External Data Dumping Circuitry" off.
8. Remove 9 pin bus cable from "Unit on Ruminant".

The system will keep the software clock operating, until the battery voltage drops below 3V, batteries are removed or system is "reset".

```
****************************************
*        RUMINANT GRAZING MONITOR      *
****************************************


1. DUMP DATA (RUMINANT GRAZING MONITOR --> Z100)
2. SET DAY/TIME
3. END COMMUNICATION
4. PRINT DATA (HARD COPY)
5. EXIT

PLEASE ENTER THE CHOICE (1/2/3/4/5)
```

Fig 12. Main menu of BASIC program

4.2.2 Installation of "Unit on Ruminant"

The user is now ready to mount the unit on the animal.

1. Fasten "Unit On Ruminant" on the animal's neck. Fasten the mercury bulb switch on the side of the ruminant's face and carefully adjust the mercury bulb switch so that when the animal's head is down, the silver blob contacts the two internal wires (Fig 3(a)), and when the animal's head is up the silver blob does not toutch both internal wires (Fig 3(b)).
2. Set "DUMP" switch to left side, then system will start to monitor and record the grazing events.

4.2.3 Transfer of Data to the Computer

Any time within the next two weeks from the time the batteries were connected to the "Unit on Ruminant", the user can dump the stored data to a computer.

1. Disconnect the mercury switch from the "Unit on Ruminant".
2. Set "DUMP" switch to right side (DUMP MODE).
3. Remove unit from animal.
4. Connect the system as 4.2.1 (Fig. 11).
5. Load and execute BASIC program on computer side.
6. Enter command 1 to dump data.

7. Ruminant Grazing Monitor will transfer its stored data to computer (LED is on).
8. Transmission is completed (LED is off).

Now, the obtained data is stored in a computer file called cow.dat . To get a hard copy, enter command 4, then data will be sent to the printer.

Keep in mind, before removing the RS-232C cable or enter command 5 "EXIT" to BASIC system, always enter command 3 (END COMMUNICATION) to inform the grazing monitor that communication is terminated. Once the grazing monitor receives this, it will poll the "DUMP" switch, until this switch is set back to left side (NORMAL MODE), the system will start its monitoring mission again.

4.2.4 Restart System

To reset the system, set "RESET" switch to right side for a second. This action will reset the internal software clock to zero and reinitialize the program. The user should be aware of this. So, generally, after "RESET", entering the dump mode, and setting the day/time is necessary.

4.2.5 Battery Replacement

Battery life time is around two weeks. To guarantee the system's normal operation and to avoid data loss, replace the batteries every two weeks or when data is dumped.

CHAPTER V

CONCLUSIONS & RECOMMENDATIONS

A prototype system was designed and built which met the design requirement. Preliminary tests indicate proper operation of the complete unit. The microcomputer brings far-reaching influence to the world, even a ruminant can't avoid this magic power.

It is recommended that the prototype be tested on a ruminant to verify proper operation in a harsh environment. If any deficiencies are found they should be corrected. Before testing the prototype it would be helpful if a smaller mercury bulb switch could be found and possibly a more rugged one. Before constructing a final production version, miniaturized forms of the CPU and memory chips should be located. Packaging entire unit in pourable plastic should be investigated.

REFERENCE

1. Dr. Owensby's letter, Department of Agronomy, KSU, 1988.
2. Motorola Single-Chip Microcomputer Data, pp. 3-1033 - 3-1058, Motorola Inc., 1984.
3. M6805 HMOS/M146805 CMOS Family User's Manual, pp. 103-106, Motorola Inc., 1983.
4. MC146805G2 8-Bit Microcomputer Programming Guide, Motorola Inc., 1981.
5. RS-232C Interface Technique Application, pp. 57-73, CHUAN-HUA Inc., Taiwan, 1986.
6. Z-Basic, ZENITH DATA SYSTEM CORPORATION, 1982.
7. CMS Survival Kit, Kansas State University, 1988.
8. SERIES 22 PROM PROGRAMMER Operator's Guide, Data I/O Corporation, 1984.
9. Motorola CMOS Logic Data (DL131Rev1), pp. 6-154 - 6-155, Motorola Inc., 1988.
10. IC Memories Data Book, pp. 211-215, Hitachi Ltd.

APPENDIX A

Dr. Owensby's Letter

June 14, 1988

Don Lenhert
Department of Electrical and Computer Engineering
Durland Hall
Kansas State University
CAMPUS

Dear Don:

After reviewing the data for grazing behavior, I have concluded that the following requirements must be met in the electronic recording system:

1. Grazing events per day - 100
2. 9 days storage capacity
3. Each grazing event must have the time the head went down and the time the head was raised recorded.
4. There must be either an RS-232 or HPIL interface connection to dump the data.
5. The instrument should be as small as possible to avoid damage by the animal rubbing against objects.

Hey, I know I am asking a lot, but needs is needs! Thanks for your help with this project.

Yours truly,

Clenton E. Owensby
Professor of Range Management

APPENDIX B

CPU PROGRAMMING & I/O PORT BIT MAPPING

PORT A

Bit	I/O	Signal	Function
7	O	/CS	SRAM Chip Select
6	O	(unused)	
5	O	(unused)	
4	O	A12	Address Line
3	O	A11	Address Line
2	O	A10	Address Line
1	O	A9	Address Line
0	O	A8	Address Line

PORT B

Bit	I/O	Signal	Function
7	O	A7	Address Line
6	O	A6	Address Line
5	O	A5	Address Line
4	O	A4	Address Line
3	O	A3	Address Line
2	O	A2	Address Line
1	O	A1	Address Line
0	O	A0	Address Line

PORT C

Bit	I/O	Signal	Function
7	I/O	D7	Data Line
6	I/O	D6	Data Line
5	I/O	D5	Data Line
4	I/O	D4	Data Line
3	I/O	D3	Data Line
2	I/O	D2	Data Line
1	I/O	D1	Data Line
0	I/O	D0	Data Line

PORT D

Bit	I/O	Signal	Function
7	O	TX	Serial Output Line
6	I	RX	Serial Input Line
5	O	(status of /FEED, system running indicator)	
4	O	Transmit	Transmit LED ON/OFF control
3	O	/WR	SRAM Write
2	O	/RD	SRAM Read
1	I	/FEED	Indicator of grazing
0	I	/DUMP	Status of dump switch

MASK OPTION REGISTER (MOR)

Bit	Setting	Function
7	0	Crystal Oscillator
6	1	Divide-by-2 Oscillator Clock
5	0	
4	0	INT. trigger input type
3	0	
2	0	
1	0	
0	0	

TIMER CONTROL REGISTER (TCR)

Bit	Setting	Function
7	0	Timer interrupt request bit
6	0	Unmask timer interrupt
5	0	Internal clock to timer
4	0	
3	0	
2	1	Prescaler 128
1	1	
0	1	

TIMER DATA REGISTER (TDR)

Load TDR with value 128.

APPENDIX C

GRAZING MONITOR SOURCE CODE

```
                ******************************************
                *                                        *
                *   RUMINANT  GRAZING  MONITOR  PROGRAM  *
                *                                        *
                ******************************************
                *
                          OPT  l
                *
                *
                ORG  $0010      ;ON CHIP RAM
                *
                *
0010            OLD_FLAG  RMB  1
0011            NEW_FLAG  RMB  1
0012            EVENT_256 RMB  1     ;COUNTER OF 256 EVENTS
0013            A_REG     RMB  1     ;A REG. BUFFER
0014            X_REG     RMB  1     ;X REG. BUFFER
0015            X1_REG    RMB  1     ;X REG. BUFFER
0016            CHAR      RMB  1     ;ONE DATA BYTE
0017            PARITY_BIT RMB  1    ;PARITY BIT
0018            COUNT     RMB  1     ;COUNTER
0019            ADDR_H1   RMB  1     ;HIGH ADDRESS BYTE
001a            ADDR_L1   RMB  1     ;LOW ADDRESS BYTE
                *
001b            CLOCK     RMB  4     ;TIME-OF-DAY CLOCK
001b            DAY       EQU  CLOCK
001c            HOUR      EQU  CLOCK+1
001d            MIN       EQU  CLOCK+2
001e            SEC       EQU  CLOCK+3
                *
001f            DAY_S     RMB  1     ;DAY OF EVENT START
0020            HOUR_S    RMB  1     ;HOUR OF EVENT START
0021            MIN_S     RMB  1     ;MIN OF EVENT START
0022            SEC_S     RMB  1     ;SEC OF EVENT START
0023            P_MIN     RMB  1     ;PERIOD OF EVENT
0024            P_SEC     RMB  1
                *
0025            DAY_E     RMB  1     ;DAY OF EVENT END
0026            HOUR_E    RMB  1     ;HOUR OF EVENT END
0027            MIN_E     RMB  1     ;MIN OF EVENT END
0028            SEC_E     RMB  1     ;SEC OF EVENT END
                *
                * ADD (EXTENDED ADDRESSING MODE) SUBROUTINE
```

```
                    *
0029                ADD_OP      RMB  1      ;ADD OPCODE = $CB
002a                ADDR_H      RMB  1      ;HIGH ADDRESS BYTE
002b                ADDR_L      RMB  1      ;LOW ADDRESS BYTE
002c                RTS_OP      RMB  1      ;RTS OPCODE = $81
                    *
002d                BLANK_NO    RMB  1
002e                TIME_ARRAY  RMB  8
0036                BUFFER      RMB  32
                    *
0080                            ORG  $0080      ;PAGE ZERO ROM
                    *
                    *
0080 20             MODULO      FCB  32     ;DAY
0081 18                         FCB  24     ;HOUR
0082 3c                         FCB  60     ;MIN
0083 3c                         FCB  60     ;SEC
0084 20 20 20 20 20 20      TITLE FCC  /          COW GRAZING
20 20 20 43 4f 57                          MONITOR/
20 47 52 41 5a 49
4e 47 20 4d 4f 4e
49 54 4f 52
00a0 0d                         FCB  CR     ;CARRIAGE RETURN
00a1 0a                         FCB  LF     ;LINE FEED
00a2 04                         FCB  EOT
                    *
                    *
0020                BLANK       EQU  $20
000d                CR          EQU  $0D    ;CARRIAGE RETURN
000a                LF          EQU  $0A    ;LINE FEED
0004                EOT         EQU  $04    ;END OF TEXT
0000                ROM_SUM     EQU  $00    ;ROM CHECK_SUM VALUE
0000                PORT_A      EQU  $0000  ;PORT A DATA REG.
0001                PORT_B      EQU  $0001  ;PORT B DATA REG.
0002                PORT_C      EQU  $0002  ;PORT C DATA REG.
0003                PORT_D      EQU  $0003  ;PORT D DATA REG.
0004                A_DDR       EQU  $0004  ;PORT A DDR
0005                B_DDR       EQU  $0005  ;PORT B DDR
0006                C_DDR       EQU  $0006  ;PORT C DDR
0007                D_DDR       EQU  $0007  ;PORT D DDR
0008                TDR         EQU  $0008  ;TIMER DATA REG.
0009                TCR         EQU  $0009  ;TIMER CONTROL REG.
1ff5                MOR         EQU  $1FF5  ;MASK OPTION REG.
0040                T_CONST     EQU  64     ;TIMER DATA CONSTANT
0006                IN          EQU  6      ;SERIAL INPUT LINE
0007                OUT         EQU  7      ;SERIAL OUTPUT LINE
0001                SOME_DAY    EQU  1
0000                SOME_HOUR   EQU  0
0000                SOME_MIN    EQU  0
```

```
0000              SOME_SEC  EQU  0
                  *
                  * -------   PROGRAM START   -------
                  *
0100                        ORG  $0100
                  *
                  * ---    SYSTEM INITIALIZE
                  *
0100 a6 40        RESET LDA #$40 ;CRYSTAL, DIV 2
0102 c7 1f f5          STA  MOR  ;MASK OPTION REG.
0105 a6 ff             LDA  #$FF
0107 b7 04             STA  A_DDR      ;OUTPUT PORT
0109 b7 05             STA  B_DDR      ;OUTPUT PORT
010b b7 06             STA  C_DDR      ;OUTPUT PORT
010d b7 00             STA  PORT_A     ;DISABLE /CS
010f a6 bc             LDA  #$BC
0111 b7 07             STA  D_DDR      ;I/O PORT
0113 a6 8c             LDA  #$8C       ;DISABLE /RD,/WR
0115 b7 03             STA  PORT_D
                  *
                  * ---    SET TIME CLOCK
                  *
0117 a6 01             LDA  #SOME_DAY
0119 b7 1b             STA  DAY
011b a6 00             LDA  #SOME_HOUR
011d b7 1c             STA  HOUR
011f a6 00             LDA  #SOME_MIN
0121 b7 1d             STA  MIN
0123 a6 00             LDA  #SOME_SEC
0125 b7 1e             STA  SEC
0127 a6 40             LDA  #T_CONST
0129 b7 08             STA  TDR        ;TIMER DATA REG.
012b a6 07             LDA  #$07
012d b7 09             STA  TCR        ;TIMER CONTROL REG.
                  *
012f 9a                CLI
0130 a6 ff        RECORD_START  LDA  #$FF
0132 b7 06             STA  C_DDR      ;PORT_C: OUTPUT PORT
0134 3f 12             CLR  EVENT_256
0136 3f 2a             CLR  ADDR_H
0138 3f 2b             CLR  ADDR_L
013a 5f                CLRX            ;CLEAR EVENT COUNT
                  *
                  * --- DETECT COW FEED SWITCH SIGNAL /FEED
                  *
013b b6 03             LDA  PORT_D
013d a4 02             AND  #$02       ;GET /FEED BIT
013f b7 10             STA  OLD_FLAG
0141 01 03 60     FEED_CHECK  BRCLR  0,PORT_D,DUMP_MODE
```

```
                                        ;/DUMP ?
0144 b6 03          LDA  PORT_D
0146 b7 13          STA  A_REG
0148 a4 02          AND  #$02       ;/FEED BIT
014a b7 11          STA  NEW_FLAG
014c b1 10          CMP  OLD_FLAG   ;OLD_FLAG = NEW_FLAG ?
014e 27 f1          BEQ  FEED_CHECK
0150 b7 10          STA  OLD_FLAG   ;NEW_FLAG --> OLD_FLAG
0152 a1 00          CMP  #0         ;NEW_FLAG = 0 ?
0154 26 1c          BNE  EVENT_END
                    *
                    * RECORD START TIME (STORE IN ON-CHIP RAM)
                    *
0156 b6 13          LDA  A_REG
0158 a4 df          AND  #$DF
015a b7 03          STA  PORT_D
015c b6 1e          LDA  SEC        ;SEC
015e b7 22          STA  SEC_S
0160 b6 1d          LDA  MIN        ;MIN
0162 b7 21          STA  MIN_S
0164 b6 1c          LDA  HOUR       ;HOUR
0166 b7 20          STA  HOUR_S
0168 b6 1b          LDA  DAY        ;DAY
016a b7 1f          STA  DAY_S
016c 3f 24          CLR  P_SEC     ;CLEAR EVENT TIME PERIOD
016e 3f 23          CLR  P_MIN
0170 20 cf          BRA  FEED_CHECK
0172 b6 13     EVENT_END LDA  A_REG
0174 aa 20          ORA  #$20
0176 b7 03          STA  PORT_D
                    *
                    * CHECK IF TIME PERIOD LONGER THAN 5 SEC ?
                    *
0178 b6 23          LDA  P_MIN
017a a1 00          CMP  #0
017c 26 06          BNE  FILL_SRAM ;MORE THAN 1 MINUTE ?
017e b6 24          LDA  P_SEC
0180 a1 05          CMP  #5         ;MORE THAN 5 SECS ?
0182 25 bd          BLO  FEED_CHECK
                    *
                    * RECORD EVENT START TIME (4 BYTES) &
                    * PERIOD (2 BYTES) ON SRAM
                    *
0184 bf 18     FILL_SRAM STX  COUNT      ;SAVE X REG.
0186 5f             CLRX                ;CLEAR INDEX X
0187 e6 1f     NEXT_BYTE LDA  DAY_S,X
0189 cd 03 2a       JSR  HEX_BCD
018c cd 02 2e       JSR  W_SRAM     ;WRITE 1 BYTE ON SRAM
018f 5c             INCX            ;NEXT BYTE
```

```
0190 a3 06          CPX  #6          ;6 BYTES DONE ?
0192 26 f3          BNE  NEXT_BYTE
0194 be 18          LDX  COUNT       ;RETRIEVE X REG.
                    *
                    * ---     SEE IF MEMORY FULL ?
                    *
0196 5c             INCX             ;INCREASE NO. OF EVENTS
0197 26 a8          BNE  FEED_CHECK
0199 3c 12          INC  EVENT_256 ;INCREASE NO. OF
                                     ;256-EVENTS
019b b6 12          LDA  EVENT_256
019d a1 05          CMP  #5          ;MEMORY USED =
                                     ; 5 * 256 * 6 BYTES ?
019f 26 a0          BNE  FEED_CHECK
                    *
                    *STAND BY MODE, WAIT FOR /DUMP TO DUMP DATA
                    *
01a1 00 03 fd       STAND_BY  BRSET  0,PORT_D,STAND_BY
                    * ------------------------------------ *
                    *   TRANSMIT DATA TO COMPUTER.         *
                    *   NO. OF DATA BYTES =                *
                    *         ( 256 * EVENT_256 + X ) * 6  *
                    * ------------------------------------ *
01a4 cd 02 cb       DUMP_MODE JSR  RX_BYTE
01a7 a1 31          CMP  #$31
01a9 27 0b          BEQ  DUMP_DATA
01ab a1 32          CMP  #$32
01ad 27 37          BEQ  SET_TIME
01af a1 33          CMP  #$33
01b1 26 f1          BNE  DUMP_MODE
01b3 cc 02 25       JMP  IDLE
01b6 3f 19          DUMP_DATA CLR  ADDR_H1   ;CLEAR POINTER
01b8 3f 1a          CLR  ADDR_L1
                    *
01ba b6 03          LDA  PORT_D
01bc aa 10          ORA  #$10        ;TRANSMIT LED ON
01be b7 03          STA  PORT_D
                    *
                    * ---     TRANSMIT DATA TO COMPUTER
                    *
01c0 a6 00          LDA  #$00
01c2 b7 06          STA  C_DDR       ;PORT_C: INPUT PORT
01c4 b6 2a          D_2  LDA  ADDR_H
01c6 b1 19          CMP  ADDR_H1     ;ADDR_H > ADDR_H1 ?
01c8 22 06          BHI  D_3
01ca b6 2b          LDA  ADDR_L      ;ADDR_L > ADDR_L1 ?
01cc b1 1a          CMP  ADDR_L1
01ce 23 08          BLS  DUMP_END
01d0 cd 02 4f       D_3  JSR  R_SRAM     ;READ SRAM DATA
```

```
01d3 cd 03 17        JSR  SEND_ASCII;SEND TWO ASCII BYTE
                                    ;TO COMPUTER
01d6 20 ec           BRA  D_2       ;LOOP
01d8 a6 7d      DUMP_END  LDA  #$7D ;"}"
01da cd 02 88        JSR  TX_BYTE
01dd b6 03           LDA  PORT_D
01df a4 ef           AND  #$EF      ;TRANSMIT LED OFF
01e1 b7 03           STA  PORT_D
01e3 cc 01 a4        JMP  DUMP_MODE
                *
                * RECEIVE DAY/TIME (DD HH:MM:SS ) 8 DIGIT
                * FROM COMPUTER.
                *
01e6 b6 03      SET_TIME  LDA  PORT_D
01e8 aa 10           ORA  #$10      ;LED ON
01ea b7 03           STA  PORT_D
01ec 5f              CLRX
01ed cd 02 cb   DB_5 JSR  RX_BYTE
01f0 e7 2e           STA  TIME_ARRAY,X
01f2 5c              INCX
01f3 a3 08           CPX  #8
01f5 26 f6           BNE  DB_5
01f7 b6 03           LDA  PORT_D
01f9 a4 ef           AND  #$EF      ;LED OFF
01fb b7 03           STA  PORT_D
01fd 3f 18           CLR  COUNT
01ff be 18      ST_5 LDX  COUNT
0201 58              LSLX           ;X=COUNT * 2
                                    ; DD HH:MM:SS ENTRY
0202 e6 2e           LDA  TIME_ARRAY,X   ;HIGH DIGIT
0204 a0 30           SUB  #$30
0206 48              LSLA
0207 48              LSLA
0208 48              LSLA
0209 48              LSLA
020a b7 13           STA  A_REG
020c 5c              INCX
020d e6 2e           LDA  TIME_ARRAY,X   ;LOW DIGIT
020f a0 30           SUB  #$30
0211 ba 13           ORA  A_REG
0213 cd 03 41        JSR  BCD_HEX
0216 be 18      ST_7 LDX  COUNT
0218 e7 1b           STA  CLOCK,X
021a 3c 18           INC  COUNT
021c b6 18           LDA  COUNT
021e a1 04           CMP  #4
0220 26 dd           BNE  ST_5
0222 cc 01 a4        JMP  DUMP_MODE
0225 01 03 fd   IDLE BRCLR  0,PORT_D,IDLE
```

```
0228 cd 02 7d       JSR  DELAY_2SEC
022b cc 01 30       JMP  RECORD_START


                    * ------------------------------------- *
                    * W_SRAM      WRITE ONE BYTE TO SRAM     *
                    *                                       *
                    * INPUT: A    DATA TO BE WRITTEN INTO SRAM*
                    *        ADDR_H    HIGH ADDRESS BYTE     *
                    *        ADDR_L    LOW ADDRESS BYTE      *
                    *                                       *
                    * OUTPUT:ADDR_H    HIGH ADDRESS BYTE     *
                    *        ADDR_L    LOW ADDRESS BYTE      *
                    * ------------------------------------- *
022e b7 02          W_SRAM    STA  PORT_C    ;DATA
0230 b6 2b              LDA  ADDR_L    ;LOW ADDRESS
0232 b7 01              STA  PORT_B
0234 b6 2a              LDA  ADDR_H    ;HIGH ADDRESS
0236 a4 1f              AND  #$1F      ;ENABLE /CS
0238 b7 00              STA  PORT_A
                    *
023a b6 03              LDA  PORT_D
023c a4 f7              AND  #$F7      ;ENABLE /WR
023e b7 03              STA  PORT_D    ;START WRITE OPERATION
0240 aa 08              ORA  #$08      ;DISABLE /WR
0242 b7 03              STA  PORT_D    ;STOP WRITE OPERATION
0244 a6 80              LDA  #$80
0246 b7 00              STA  PORT_A    ;DISABLE /CS
                    *
0248 3c 2b              INC  ADDR_L    ;NEXT ADDRESS
024a 26 02              BNE  W_END
024c 3c 2a              INC  ADDR_H    ;CARRY FROM LOW ADDRESS
                                       ;BYTE
024e 81             W_END     RTS      ;RETURN
                    * ------------------------------------- *
                    *   R_SRAM    READ ONE BYTE FROM SRAM     *
                    *                                       *
                    *   INPUT:    ADDR_H1                     *
                    *             ADDR_L1                     *
                    *                                       *
                    *   OUTPUT:   A                           *
                    *             ADDR_H1                     *
                    *             ADDR_L1                     *
                    * ------------------------------------- *
024f bf 14          R_SRAM    STX  X_REG     ;SAVE X REG.
0251 b6 1a              LDA  ADDR_L1   ;LOW ADDRESS
0253 b7 01              STA  PORT_B
```

```
0255 b6 19          LDA  ADDR_H1    ;HIGH ADDRESS
0257 a4 1f          AND  #$1F       ;ENABLE /CS
0259 b7 00          STA  PORT_A
                    *
025b b6 03          LDA  PORT_D
025d a4 fb          AND  #$FB       ;ENABLE /RD
025f b7 03          STA  PORT_D     ;START READ OPERATION
0261 9d             NOP             ;WAIT
0262 be 02          LDX  PORT_C     ;READ PORT_C DATA INTO X REG.
0264 aa 04          ORA  #$04       ;DISABLE /RD
0266 b7 03          STA  PORT_D
0268 a6 80          LDA  #$80
026a b7 00          STA  PORT_A     ;DISABLE /CS
                    *
026c 9f             TXA             ;X --> A
026d 3c 1a          INC  ADDR_L1    ;NEXT ADDRESS
026f 26 04          BNE  R_END
0271 3c 19          INC  ADDR_H1    ;CARRY FROM LOW ADDRESS
                                    ;BYTE
0273 be 14          LDX  X_REG      ;RESTORE X
                    *
0275 81             R_END   RTS     ;RETURN
                    * ------------------------------------- *
                    *   DELAY_100MS    DELAY 100 MS         *
                    * ------------------------------------- *
0276 ae c7          DELAY_100MS   LDX  #199      ;(2)
0278 5a             DEC_X    DECX   ;(3)
0279 9d                 NOP         ;(2)
027a 26 fc              BNE  DEC_X  ;(3)
027c 81                 RTS         ;(6)
                    * ------------------------------------- *
                    *   DELAY_2SEC     DELAY 2 SEC          *
                    * ------------------------------------- *
027d 5f             DELAY_2SEC    CLRX ;(3)
027e a6 09          A_9  LDA  #9       ;(2)
0280 4a             DEC_A    DECA      ;(3)
0281 26 fd              BNE  DEC_A     ;(3)
0283 5a                 DECX           ;(3)
0284 9d                 NOP            ;(2)
0285 26 f7              BNE  A_9       ;(3)
0287 81                 RTS            ;(6)
                    * ------------------------------------- *
                    *   SERIAL I/O ROUTINES                 *
                    *        SERIAL INPUT LINE PORT_D BIT 6 *
                    *        SERIAL OUTPUT LINE PORT_D BIT 7*
                    * ------------------------------------- *
                    *
                    * ------------------------------------- *
                    * TX_BYTE   TRANSMIT ONE BYTE TO COMPUTER *
```

```
                    *                                                 *
                    *   INPUT:     A    DATA TO BE TRANSMITTED      *
                    *   OUTPUT:    X & A UNCHANGED                  *
                    * ------------------------------------------- *
0288 9b             TX_BYTE   SEI         ;DISABLE INT.
0289 b7 16                 STA  CHAR      ;DATA TO BE TRANSMITTED
028b b7 13                 STA  A_REG     ;SAVE A
028d bf 14                 STX  X_REG     ;SAVE X
028f ad 27                 BSR  PARITY_GEN;GENERATE PARITY BIT
0291 ae 09                 LDX  #9        ;8 DATA BITS & 1 PARITY
                                          ;BIT
0293 1f 03                 BCLR OUT,PORT_D;SEND START BIT
0295 4d                    TSTA           ;(3)
0296 4d                    TSTA           ;(3)
0297 4d                    TSTA           ;(3) TIMING EQUALIZATION
0298 ad 28                 BSR  DELAY_TX
                    *
                    * ---     MAIN LOOP FOR TX_BYTE
                    *
029a 36 16          TX_1 ROR  CHAR        ;(5) GET NEXT BIT
029c 24 04          TX_2 BCC  TX_3        ;(3) SET OR CLEAR ?
029e 1e 03          BSET OUT,PORT_D       ;(5) BIT DATA "1"
02a0 20 04                 BRA  TX_4      ;(3)
02a2 1f 03          TX_3 BCLR OUT,PORT_D;(5) BIT DATA "0"
02a4 20 00                 BRA  TX_4      ;(3) TIMING EQUALIZATION
02a6 ad 1a          TX_4 BSR  DELAY_TX    ;(6)
02a8 5a                    DECX           ;(3)
02a9 26 ef                 BNE  TX_1      ;(3) 9 BITS DONE ?
                    *
                    * ---     SEND STOP BIT
                    *
02ab 4d                    TSTA           ;(3)
02ac 4d                    TSTA           ;(3)
02ad 9d                    NOP            ;(2) 8 CYCLES DELAY
02ae 1e 03                 BSET OUT,PORT_D;SEND STOP BIT
02b0 ad 10                 BSR  DELAY_TX  ;DELAY FOR THE STOP BIT
02b2 be 14                 LDX  X_REG     ;RESTORE X
02b4 b6 13                 LDA  A_REG     ;RESTORE A
                    *
02b6 9a                    CLI            ;ENABLE INT.
02b7 81                    RTS            ;RETURN
                    * ------------------------------------------- *
                    *   PARITY_GEN   GENERATE EVEN PARITY BIT     *
                    *   INPUT:     A_REG = DATA WHICH PARITY IS*
                    *                      TO BE GENERATED      *
                    *   OUTPUT:    CARRY BIT IN CONDITION CODE  *
                    *              REGISTER = PARITY BIT        *
                    * ------------------------------------------- *
02b8 ae 08          PARITY_GEN     LDX  #8        ;8 BIT COUNT
```

```
02ba 4f           CLRA            ;CLEAR A
02bb b8 13     NEXT_BIT  EOR  A_REG
02bd 48           LSLA            ;SHIFT LEFT ONE BIT
02be 5a           DECX
02bf 26 fa        BNE  NEXT_BIT
02c1 81           RTS
                  * ------------------------------------- *
                  *DELAY_TX  27 MPU CYCLES DELAY FOR TX_BYTE*
                  * ------------------------------------- *
02c2 a6 02     DELAY_TX  LDA  #2    ;(2)
02c4 4a        DEL_TX    DECA       ;(3)
02c5 26 fd        BNE  DEL_TX      ;(3)
02c7 4d           TSTA             ;(3)
02c8 9d           NOP              ;(2)
02c9 9d           NOP              ;(2)
02ca 81           RTS              ;(6)
                  * ------------------------------------- *
                  * RX_BYTE   RECEIVE A BYTE FROM COMPUTER  *
                  *                                       *
                  *   OUTPUT:   CARRY SET    --> ERROR !    *
                  *             CARRY CLEAR  --> OK !       *
                  *             A =   RECEIVED BYTE         *
                  * ------------------------------------- *
02cb 9b        RX_BYTE   SEI          ;DISABLE INT.
02cc bf 14        STX  X_REG       ;SAVE X
02ce ae 09        LDX  #9          ;8 DATA BITS & 1 PARITY
                                   ;BIT.
                  * ---  WAIT FOR HIGH-LOW TRANSITION
02d0 0c 03 fd  START_BIT BRSET  IN,PORT_D,START_BIT
                  *
                  * ---   DELAY 1/2 BIT TIME  (27 MPU CYCLES)
                  *
02d3 a6 03        LDA  #3          ;(2)
02d5 4a        RX_1 DECA           ;(3)
02d6 26 fd        BNE  RX_1        ;(3)
02d8 4d           TSTA             ;(3)
02d9 9d           NOP              ;(2)
02da 9d           NOP              ;(2)
                  *
                  *NOW, WE ARE IN THE MIDDLE OF THE START BIT
                  *FALSE START BIT CHECK
                  *
02db 0c 03 f2     BRSET  IN,PORT_D,START_BIT
02de 4d           TSTA             ;(3)
02df 4d           TSTA             ;(3)
02e0 4d           TSTA             ;(3)
02e1 9d           NOP              ;(2) TIMING EQUALIZATION
                  *
                  * ---    MAIN LOOP OF RECEIVING A BYTE
```

```
                    *
02e2 ad 2a          RX_3 BSR  DELAY_RX  ;(6)
                    * TEST INPUT LINE & SET C-BIT
02e4 0d 03 00            BRCLR IN,PORT_D,RX_4 ;(5)
02e7 36 16          RX_4 ROR  CHAR      ;(5) ADD THIS BIT TO
                                        ;THE BYTE
02e9 5a                  DECX           ;(3)
02ea 26 f6               BNE  RX_3      ;(3) MORE BITS TO GET ?
02ec 39 16               ROL  CHAR      ;ROTATE PARITY BIT BACK
                                        ;TO CARRY BIT.
                    *
                    *    BSR  PARITY_CHK;CHECK PARITY
                    *    BSR  DELAY_RX  ;WAIT OUT THE 10TH BIT
02ee b6 16               LDA  CHAR      ;RECEIVED BYTE
02f0 be 14               LDX  X_REG     ;RESTORE X
                    *
02f2 9a                  CLI
02f3 81                  RTS            ;RETURN
                    * ---------------------------------------- *
                    *   PARITY_CHK     CHECK PARITY BIT       *
                    *   INPUT: CHAR =  DATA WHICH PARITY IS   *
                    *                    TO BE CHECKED        *
                    *          CARRY BIT = RECEIVED PARITY BIT*
                    *   OUTPUT:CARRY SET   --> ERROR !        *
                    *          CARRY CLEAR --> OK !           *
                    * ---------------------------------------- *
02f4 24 06          PARITY_CHK     BCC  PC_1
02f6 a6 01               LDA  #1
02f8 b7 17               STA  PARITY_BIT;PARITY BIT = 1
02fa 20 04               BRA  PC_2
02fc a6 00          PC_1 LDA  #0
02fe b7 17               STA  PARITY_BIT;PARITY BIT = 0
0300 b6 16          PC_2 LDA  CHAR
0302 ad b4               BSR  PARITY_GEN
0304 49                  ROLA           ;SHIFT CARRY(PARITY)
                                        ;BIT INTO A BIT0.
0305 a4 01               AND  #$01      ;EXTRACT PARITY BIT
0307 98                  CLC            ;CLEAR CARRY
0308 b8 17               EOR  PARITY_BIT;COMPARE
030a 27 01               BEQ  PC_3      ;OK !
030c 99                  SEC            ;ERROR !
030d 81                  PC_3          RTS
                    * ---------------------------------------- *
                    *DELAY_RX  33 MPU CYCLES DELAY FOR RX_BYTE*
                    * ---------------------------------------- *
030e a6 03          DELAY_RX  LDA  #3   ;(2)
0310 4a             DEL_RX    DECA      ;(3)
0311 26 fd               BNE  DEL_RX    ;(3)
0313 9d                  NOP            ;(2)
```

```
0314 9d             NOP             ;(2)
0315 4d             TSTA            ;(3) TIMING EQUALIZATION
0316 81             RTS             ;(6)
                    * ------------------------------------- *
                    *   SEND_ASCII   SEND TWO ASCII BYTE    *
                    *                (ONE HEX BYTE) TO RS-232 *
                    *   INPUT:    A    DATA TO BE TRANSMITTED *
                    *   OUTPUT:   NONE                      *
                    * ------------------------------------- *
0317 97         SEND_ASCII      TAX
0318 44             LSRA
0319 44             LSRA
031a 44             LSRA
031b 44             LSRA
031c ab 30          ADD  #$30
031e cd 02 88       JSR  TX_BYTE
0321 9f             TXA
0322 a4 0f          AND  #$0F
0324 ab 30          ADD  #$30
0326 cd 02 88       JSR  TX_BYTE
0329 81             RTS
                    * ------------------------------------- *
                    * HEX_BCD       CONVERT ONE HEX BYTE TO   *
                    *               BCD CODE.               *
                    * INPUT:  A     HEX BYTE TO BE CONVERTED  *
                    * OUTPUT: A     BCD CODE                  *
                    * ------------------------------------- *
032a bf 14      HEX_BCD   STX  X_REG
032c 5f             CLRX
032d a1 0a      H1    CMP  #10
032f 25 05          BCS  H2
0331 a0 0a          SUB  #10
0333 5c             INCX
0334 20 f7          BRA  H1
0336 58         H2    LSLX
0337 58             LSLX
0338 58             LSLX
0339 58             LSLX
033a bf 13          STX  A_REG
033c ba 13          ORA  A_REG
033e be 14          LDX  X_REG
0340 81             RTS
                    * ------------------------------------- *
                    * BCD_HEX    CONVERT ONE BYTE BCD CODE TO *
                    *           HEX CODE.                   *
                    * INPUT: A   BCD CODE TO BE CONVERTED.   *
                    * OUTPUT:A   HEX CODE.                   *
                    * ------------------------------------- *
0341 bf 14      BCD_HEX   STX  X_REG
```

```
0343 97          TAX             ;A -> X
0344 54          LSRX
0345 54          LSRX
0346 54          LSRX
0347 54          LSRX            ;HIGH DIGIT IN X
0348 a4 0f       AND  #$0F       ;LOW DIGIT IN A
034a a3 00   BH_1 CPX  #0
034c 27 05       BEQ  BH_2
034e ab 0a       ADD  #10        ;10 + A -> A
0350 5a          DECX
0351 20 f7       BRA  BH_1
0353 be 14   BH_2 LDX  X_REG
0355 81          RTS
                 *
0356 80          TIMER_WAIT    RTI
0357 80          EXT_INT       RTI
0358 80          SWI_INT       RTI
                 * ------------------------------------- *
                 *   TIMER INTERRUPT ROUTINE             *
                 *   1 HZ INTERRUPT FOR TIME OF DAY CLOCK *
                 * ------------------------------------- *
0359 9b          TIMER_INT SEI       ;DISABLE INT.
035a a6 40       LDA  #T_CONST
035c b7 08       STA  TDR        ;LOAD TIMER CONSTANT
035e ae 03       LDX  #3         ;BEGIN AT LSB OF CLOCK
0360 6c 1b   TICK INC  CLOCK,X   ;BUMP THIS DIGIT
0362 e6 1b       LDA  CLOCK,X
0364 e1 80       CMP  MODULO,X   ;SEE IF IT OVERFLOWED
0366 25 11       BLO  PERIOD     ;DONE IF LOWER THAN
                                 ;MODULO
0368 6f 1b       CLR  CLOCK,X    ;RESET THIS COUNTER
                                 ;AND GO TO NEXT
036a 5a          DECX            ;WITH OVERFLOW
036b 2a f3       BPL  TICK
036d 3c 1b       INC  DAY
036f b6 1b       LDA  DAY
0371 b1 80       CMP  MODULO
0373 25 04       BLO  PERIOD
0375 a6 01       LDA  #1         ;1ST DAY OF MONTH
0377 b7 1b       STA  DAY
0379 3c 24   PERIOD    INC  P_SEC
037b b6 24       LDA  P_SEC
037d a1 3c       CMP  #60
037f 25 04       BLO  CLK_OUT
0381 3f 24       CLR  P_SEC
0383 3c 23       INC  P_MIN      ;CARRY FROM P_SEC
0385 a6 07   CLK_OUT   LDA  #$07
0387 b7 09       STA  TCR        ;CLEAR TCR7 INT.
                                 ;REQUEST BIT
```

```
0389 9a              CLI
038a 80              RTI


                     * ---------------------------------------- *
                     *   INTERRUPT VECTORS TABLE                *
                     * ---------------------------------------- *
1ff6                 ORG  $1FF6
                     *
1ff6 03 56           FDB  TIMER_WAIT
1ff8 03 59           FDB  TIMER_INT
1ffa 03 57           FDB  EXT_INT
1ffc 03 58           FDB  SWI_INT
1ffe 01 00           FDB  RESET
                     *
                     *
                     END
```

APPENDIX D

EXTERNAL COMPUTER BASIC PROGRAM SOURCE CODE

```
10   '*************************************************
20   '* RUMINANT GRAZING MONITOR<-->Z100 RS-232 PROGRAM *
30   '*************************************************
40        DIM A$(16384)
50        DIM B$(12)
60        DIM T$(8)
70        CLS
80        PRINT "   ************************************"
90        PRINT "   *     RUMINANT GRAZING MONITOR      *"
100       PRINT "   ************************************"
110       PRINT
120       PRINT
130       PRINT
140       PRINT
150       PRINT
160       PRINT
170       PRINT "1. DUMP DATA (GRAZING MONITOR --> Z100  "
180       PRINT "2. SET DAY/TIME (Z100-->GRAZING MONITOR)"
190       PRINT "3. END COMMUNICATION                    "
200       PRINT "4. PRINT DATA (HARD COPY)               "
210       PRINT "5. EXIT                                 "
220       PRINT
230       PRINT
240       PRINT
250       PRINT "   PLEASE ENTER THE CHOICE (1/2/3/4/5)";
260       INPUT X
270       IF X = 1 THEN 330
280       IF X = 2 THEN 750
290       IF X = 3 THEN 1270
300       IF X = 4 THEN 1380
310       IF X = 5 THEN 1740
320       GOTO 70
330  '1. DUMP DATA
340       CLS
350       ENDMARK$ = CHR$(&H7D)
360       PRINT "DATA TRANSMITTING --- "
370       RESET
380       OPEN "O", #1, "COW.DAT"
390       OPEN "COM1:150,E,8,1" AS #2
400  'SEND CHOICE NO. 1 TO RUMINANT GRAZING MONITOR
410       PRINT#2, CHR$(&H31);
420  'DATA TRANSMITTION START, STORE RECEIVED DATA INTO
     MEMORY ARRAY
```

```
430       LET N = 0
440       IF LOC(2) = 0 THEN 440
450       A$(N) = INPUT$(1, #2)
460       IF A$(N) = ENDMARK$ THEN 490
470       LET N = N + 1
480       GOTO 440
490       LET N = 0
500       LET M = 0
510       B$(M) = A$(N)
520       IF B$(M) = ENDMARK$ THEN 580
530       M = M + 1
540       IF M = 12 THEN GOSUB 690
550       LET N = N + 1
560       GOTO 510
570   'STORE MEMORY ARRAY DATA INTO FILE COW.DAT
580       FOR Y=0 TO N
590       PRINT#1, A$(Y);
600       NEXT
610       CLOSE#1:CLOSE#2
620       PRINT
630       PRINT "DATA TRANSMITTION COMPLETED ! "
640       PRINT
650       PRINT "PRESS <RETURN> KEY TO CONTINUE --- ";
660       INPUT C$
670       GOTO 70
680   'SUBROUTINE, DISPLAY ONE EVENT ON SCREEN
690       PRINT B$(0);B$(1)" ";
700      PRINT B$(2);B$(3)":"B$(4);B$(5)":"B$(6);B$(7)" ";
710       PRINT B$(8);B$(9)":"B$(10);B$(11)
720       M = 0
730       RETURN
740   '2. SET DAY/TIME
750       OPEN "COM1:150,E,8,1" AS #2
760       CLS
770       PRINT "SET DAY/TIME OF RUMINANT GRAZING MONITOR"
780       PRINT
790       PRINT
800       PRINT
810       PRINT
820       PRINT
830       INPUT "1ST DIGIT OF DAY(0-3)      = "; T$(0)
840       INPUT "2ND DIGIT OF DAY(0-9)      = "; T$(1)
850       INPUT "1ST DIGIT OF HOUR(0-2)     = "; T$(2)
860       INPUT "2ND DIGIT OF HOUR(0-9)     = "; T$(3)
870       INPUT "1ST DIGIT OF MINUTE(0-5)   = "; T$(4)
880       INPUT "2ND DIGIT OF MINUTE(0-9)   = "; T$(5)
890       INPUT "1ST DIGIT OF SECOND(0-5)   = "; T$(6)
900       INPUT "2ND DIGIT OF SECOND(0-9)   = "; T$(7)
910       PRINT
```

```
920       PRINT
930       PRINT "CURRENT DAY/TIME IS:   ";
940       PRINT T$(0);T$(1)" ";
950      PRINT T$(2);T$(3)":"T$(4);T$(5)":"T$(6);T$(7)" ";
960       PRINT "   (Y/N)   ";
970       INPUT C$
980       IF C$ = CHR$(&H59) THEN 1010
990       IF C$ = CHR$(&H79) THEN 1010
1000      GOTO 760
1010      PRINT
1020 'SEND CHOICE NO. 2 TO GRAZING MONITOR
1030      PRINT#2, CHR$(&H32);
1040      GOSUB 1220
1050      PRINT "SENDING DAY/TIME --- "
1060      FOR K = 0 TO 7
1070      PRINT#2, T$(K);
1080      PRINT T$(K);
1090      GOSUB 1220
1100      NEXT
1110      PRINT
1120      CLOSE#2
1130      PRINT
1140      PRINT "PRESS <RETURN> KEY TO CONTINUE --- ";
1150      INPUT C$
1160      GOTO 70
1170 'PRINT ERROR MESSAGE
1180      PRINT
1190      PRINT "INVALID DAY/TIME ! "
1200      PRINT
1210      GOTO 830
1220 'TIME DELAY ROUTINE
1230      FOR J = 0 TO 15
1240      LET J = J
1250      NEXT
1260      RETURN
1270 '3. END COMMUNICATION
1280      CLS
1290      OPEN "COM1:150,E,8,1" AS #2
1300      PRINT "Z100 <--> GRAZING MONITOR DISCONNECTED !"
1310 'SEND CHOICE NO. 3 TO RUMINANT GRAZING MONITOR
1320      PRINT#2, CHR$(&H33)
1330      CLOSE#2
1340      PRINT
1350      PRINT "PRESS <RETURN> KEY TO CONTINUE --- ";
1360      INPUT C$
1370      GOTO 70
1380 '4. PRINT DATA (HARD COPY)
1390      OPEN "I", #1, "COW.DAT"
1400      CLS
```

```
1410      PRINT "PRINT DATA (HARD COPY) --- "
1420      PRINT
1430      LET N = 0
1440      IF LOC(1) = 0 THEN 1440
1450      IF EOF(1) THEN 1500
1460      A$(N) = INPUT$(1, #1)
1470      IF A$(N) = ENDMARK$ THEN 1500
1480      LET N = N + 1
1490      GOTO 1440
1500      LET N = 0
1510      LET M = 0
1520      B$(M) = A$(N)
1530      IF B$(M) = ENDMARK$ THEN 1580
1540      M = M + 1
1550      IF M = 12 THEN GOSUB 1650
1560      LET N = N + 1
1570      GOTO 1520
1580      CLOSE#1
1590      PRINT
1600      PRINT "HARD COPY COMPLETED ! "
1610      PRINT
1620      PRINT "PRESS <RETURN> KEY TO CONTINUE --- ";
1630      INPUT C$
1640      GOTO 70
1650 'SUBROUTINE, PRINT ONE EVENT ON SCREEN AND PRINTER
1660       PRINT B$(0);B$(1)" ";
1670      LPRINT B$(0);B$(1)" ";
1680       PRINT B$(2);B$(3)":"B$(4);B$(5)":"B$(6);B$(7)" ";
1690      LPRINT B$(2);B$(3)":"B$(4);B$(5)":"B$(6);B$(7)" ";
1700       PRINT B$(8);B$(9)":"B$(10);B$(11)
1710      LPRINT B$(8);B$(9)":"B$(10);B$(11)
1720       M = 0
1730       RETURN
1740 '5. EXIT (RETURN TO BASIC SYSTEM)
1750       OPEN "COM1:150,E,8,1" AS #2
1760       PRINT#2, CHR$(&H33)
1770       CLS
1780       CLOSE#2
1790       END
```

APPENDIX E

PARTS LIST

Type	Part No.	Name	Function
IC	U1	MC1468705G2	CPU
"	U2	HM6264P-15	8 K SRAM
"	U3	MC14069	Hex Inverter
"	U4	1488	TTL --> RS-232C
"	U5	1489	RS-232C --> TTL
Resistor	R1	22 M ohm	
"	R2	330 K ohm	
"	R3	100 K ohm	
"	R4	100 K ohm	
"	R5	470 ohm	
"	R6	100 K ohm	
"	R7	100 K ohm	
"	R8	100 K ohm	
Diode	D1	LED	Transmission Indicator
Capacitor	C1	20 pF	
"	C2	20 pF	
"	C3	1.0 uF	
"	C4	150 pF	
"	C5	390 pF	
Crystal		32.768 KHz	
Switch	SW1		:Operation Mode Setting
"	SW2	Mercury Bulb Switch	:Grazing Detect
"	SW3	Temporary Switch	:System Reset
Battery(3)	1.5V	AA Alkaline	:Power Source

RUMINANT GRAZING MONITOR

by

KANCHUAN HUANG

B.S., National Tsing Hua University
Hsinchu, Taiwan, 1981

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

ABSTRACT

This report presents the design of an improved ruminant grazing monitor system. Currently, the Department of Agronomy is using a mechanical system in monitoring grazing events. The system is fastened onto the animal's neck. By measuring the distance between lines drawn by the system, the time of the grazing event can be estimated roughly. The precision of the obtained data is not good, and human labor is required in checking and figuring out the record.

To obtain more reliable data and save man hour, we designed an electronic recording system using a microcomputer to carry out the function. To achieve this goal, we designed a low power microcomputer system, with sufficient memory for two weeks data storage. In order to avoid damage by the animal rubbing against objects, the instrument size was minimized.

The new system is composed of three units: Unit on Ruminant, External Data Dumping Circuitry and Computer. Usually, only "Unit on Ruminant" is fastened on ruminant's neck to monitor and record grazing event data. When stored data is needed, these three units can be connected to transfer data to computer.