CONTROL COMPUTER LOCAL DRIVER ROUTINES IN A FUNCTIONALLY DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

by

EUGENE KENNETH GOODELL

B.S., United States Military Academy, 1961

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1977

Approved by:

Major Professor

ACKNOWLEDGEMENT

This project discusses hardware devices and techniques which are currently being researched and developed by the Department of Computer Science, Kansas State University. Consequently, there are few available references in published form. The majority of my information, therefore, was obtained through personal conversation with and access to the notes of Dr. Myron A. Calhoun of the Department of Computer Science. His patience and time in providing guidance and support to me during the preparation of this report is gratefully acknowledged.

TABLE OF CONTENTS

ILLUSTRATIONS

FIGURE

PART I: Background

1. Physical Design of the Network

Figure 1 represents an overview of the Functionally Distributed Data Base System. Each cluster may have any number of main computers and each of the main computers may or may not contain a data base of the network. The only impact of a main computer having a data base is the temporary loss of that data base to the system in the event of that computer's breakdown. Backup has been built into the network to allow for minor failures and the initiation of appropriate messages to the remainder of the system to permit continued operation without the disabled data base. One cluster communicates with another via one of the control computers allied with each main computer. These are discussed below

2. The Cluster

The cluster is basically a miniature of the network except that data movement within the cluster is over high speed data lines. Each node of the cluster contains a host or main computer, a control computer, and a number of autonomous functional units (AFU's) which perform the physical movement of data within the cluster. The purpose of using control computers is to relieve the main computer of the relatively menial task of moving data from one user to another and permit it to concentrate solely on its primary function of processing. In this light, a micro computer, due to its low cost, is considered to be the best type of computer for the control role. Within the cluster, host computers may communicate with each other and control computers may communicate with other control computers over hardwired links. Data movement is conducted under the command of the control computers and over a separate set of lines: the KSUBUS.

3. Main Computer/Control Computer Relationship

Each main computer is related to its associated control computer and allied AFU's via 1) the KSUBUS for data movement and some control commands and 2) control lines for the relay of commands. The KSUBUS is a multi-wire bus which provides a hardwired, high speed path for the transfer of data in and out of

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

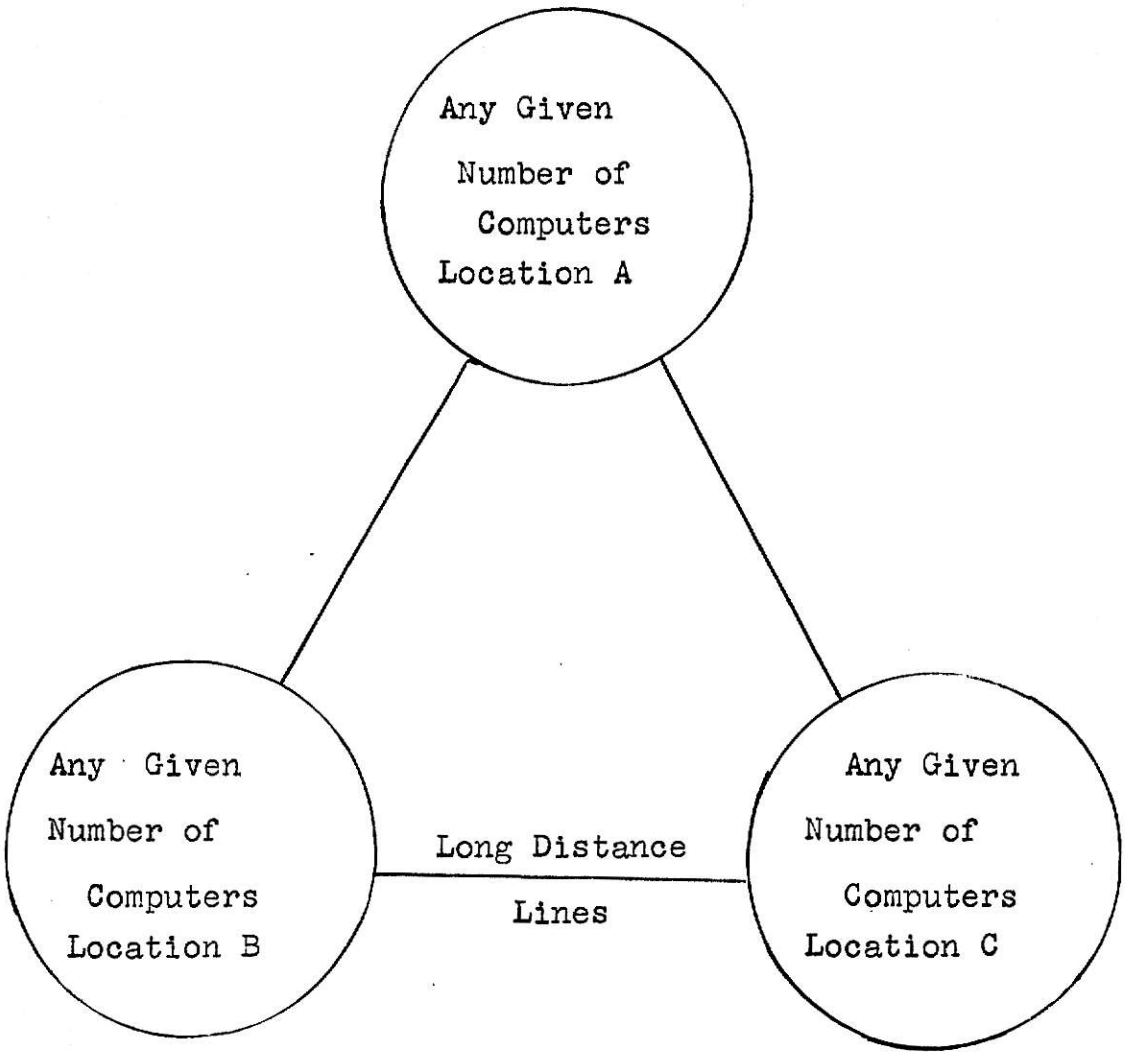


Figure 1

Overview of a Functionally Distributed
Data Base System

the host and, if the data is destined for long distance transmission to a remote receiver, to control memory. Access to the memory of either the host or control computer is by a Direct Memory Access (DMA) device which permits access to the computer's memory without interrupting any main process which may be running. DMA's are passive devices, as far as control language is concerned, and simply respond to commands from an AFU. Host and control computers communicate via a direct communication line and the control computer communicates through a Universal Logic Interface (ULI) to the appropriate AFU by hardwired means. Transmitter and receiver AFU's are hardwired by a multi-conductor cable (on the order of 50 lines) to matching AFU's in other nodes of the cluster. Figure 2 best summarizes this relationship. The transmitter/receiver AFU's connect to the KSUBUS only for the purpose of DMA to DMA movement and transmission to multiple receivers.

PART II: Software Requirements of the System

1. The purpose of the Functionally Distributed Data Base System is to expand the data base available to one main computer by enabling it to access the data base of other computers and, in turn, share its data base with them.

2. When the host computer operates using the data base contained within its own memory, the normal operating system interface maintains the data flow required by the local user. When the user desires to access a data base not in the memory of the host computer, however, a sequence of events specified by the Functionally Distributed Data Base System is invoked which will transfer the requested data base into the user memory space on the host computer. While this sequence may be transparent to the user when performed at the local level, loading time and the system load could cause a noticable delay when accessing the data base on a remote machine.

3. In the case of a host computer's data base being accessed by another machine, the reading of the data base by the Direct Memory Access device should be totally transparent to the local user. AFU's will read the requested data directly through the

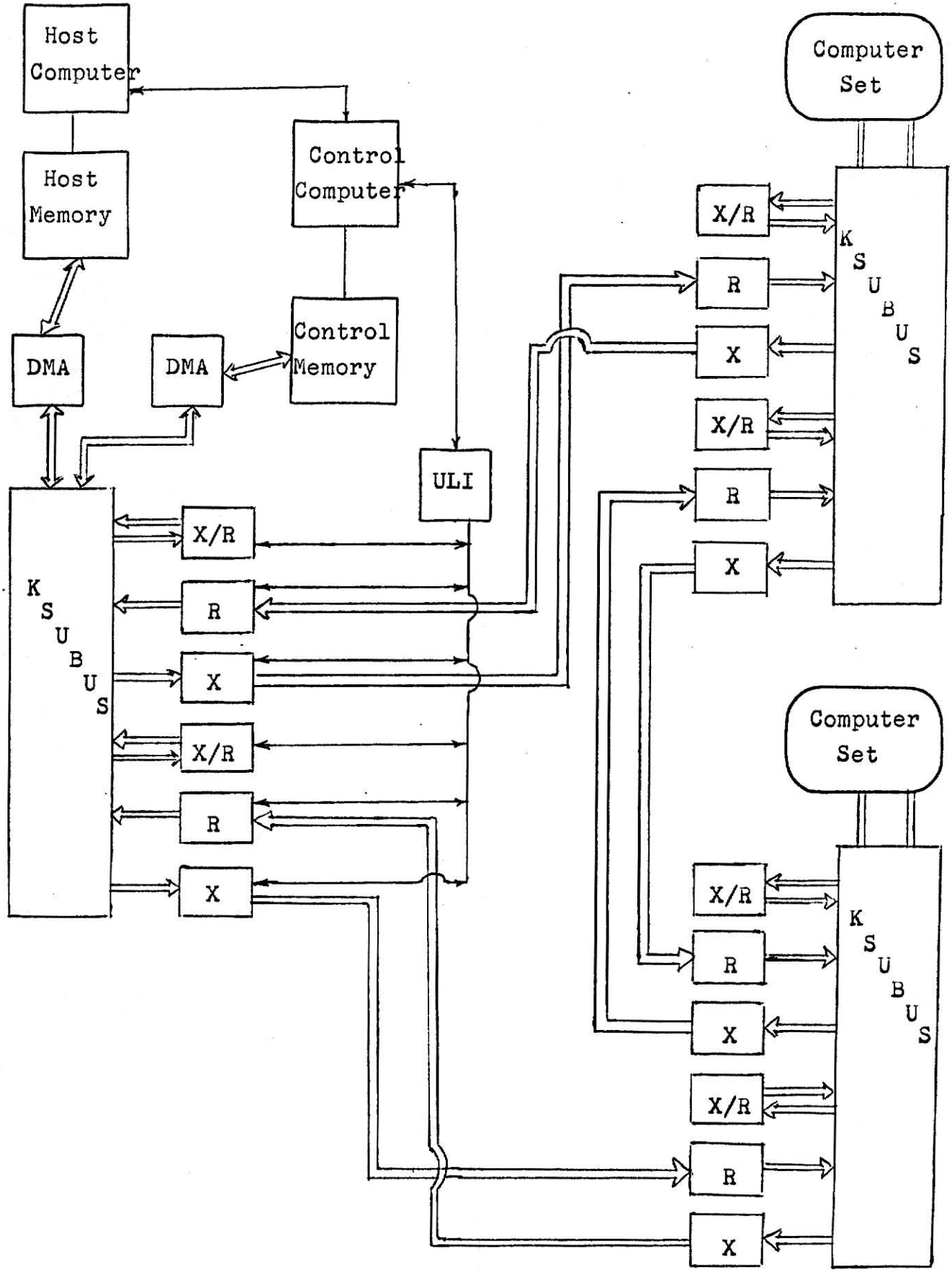


Figure 2
Local Data Movement in the Cluster

DMA without interfering with the operating system controlling local memory or, if the local machine is currently using the data, the AFU will either "steal" the data on alternate cycles with the host machine or be locked out by the host machine's software until the desired data base is free. Two situations will arise, then: 10 a host machine requires a data base it does not have or 2) an outside machine requires a data base that the host has. Either situation triggers action by the control computers associated with both the requesting and data owning computers.

4. The control computer exchanges significant information about the subject data base with its host. Either the host requests the control computer to find a needed data base, if it needs data, or the control computer requests the memory address of a data base within the host's memory if data is requested by another machine. If requested by a machine within the cluster, the request could have been made between control computers, between host computers or via a host 1-control 1-control 2-host 2 path. Regardless of how the request is made known in the system, the control computer of the host machine possessing the data base and the control computer of the machine wanting the data base must coordinate the transfer of data.

5. Should a requirement for a data base come from outside the cluster, (external demand) it will be made through one of the control computers as host computers can communicate with one another only within the same cluster. If necessary, the request will be routed about the cluster to the control computer associated with the host computer maintaining the requested data base. Transfer of this externally required data will be made through the transmitter/receiver AFU which accesses the host memory and sends the data to the control computer which, in turn, transmits the data to the requesting remote site. The reverse of this procedure is used when receiving data into the cluster from a remote site. Figure 3 summarizes this case.

6. This project considers the case of a data base transfer within the same cluster. This involves a requesting set of

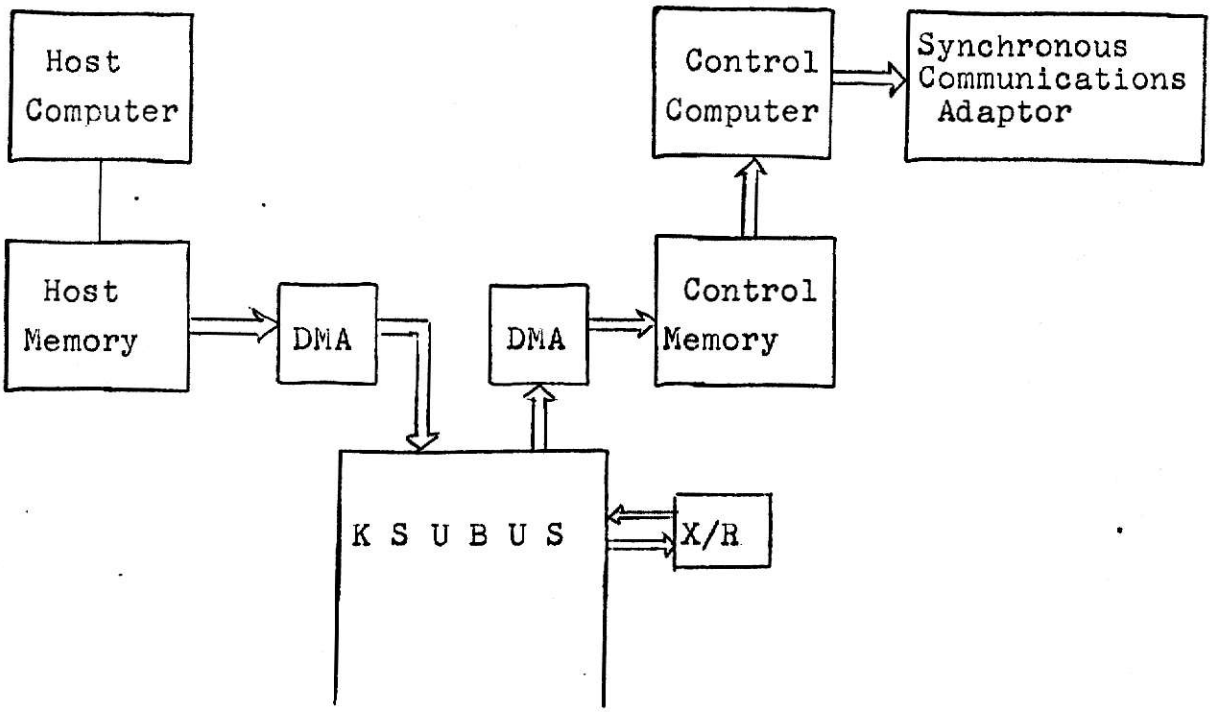


Figure 3a

Data Flow for Long Range Request

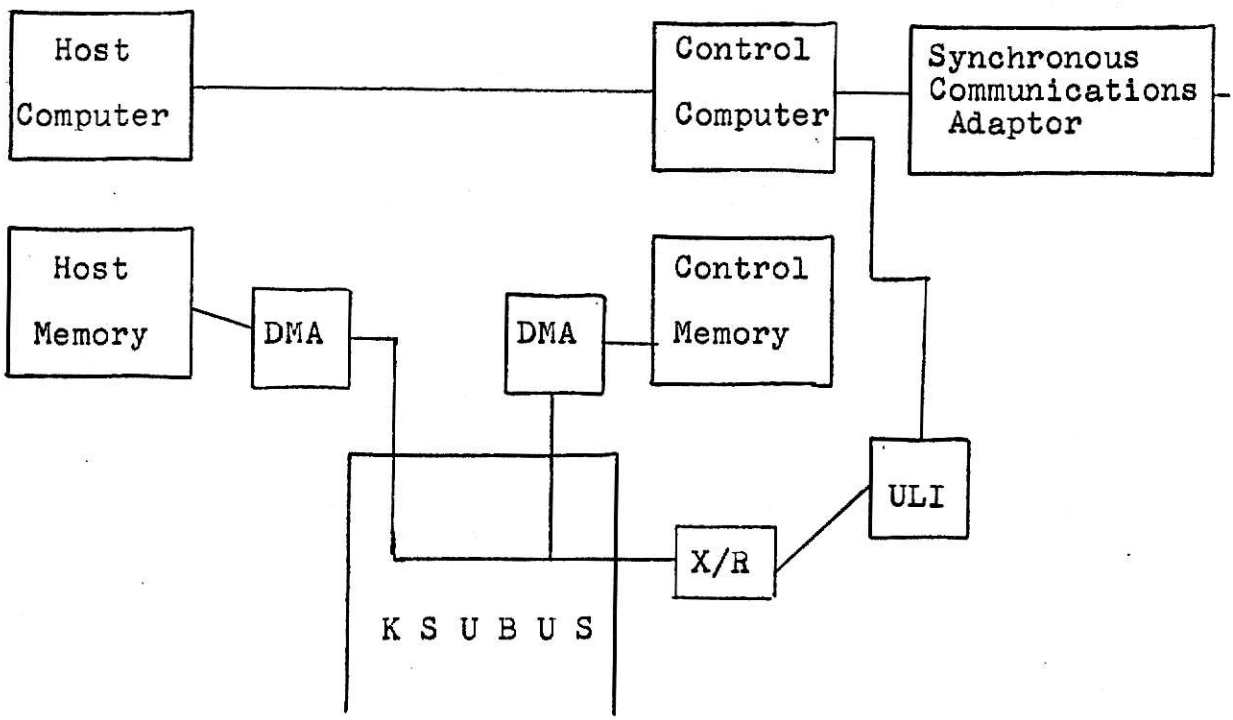


Figure 3b

Control Communications for Long Range Request

computers, one host and one control computer per set, directing their receiving AFU and a sending set of computers directing their transmitting AFU. As the two AFU's are hardwired together, the problem narrows to formulating appropriately coded commands to be issued from the control computers to the AFU's.

PART III: The Data Transfer Sequence

1. This project considers 6 cases which will arise within a cluster to create interaction between the control computer and a particular AFU or set of AFU's. Before detailing the actual handling of the different type tasks by the AFU's, however, a description of the KSUBUS and some aspects of the internal design of the AFU's is in order.

2. The KSUBUS

a. An AFU's position on its KSUBUS is based on a priority scheme where the AFU, in obtaining or storing a halfword of data, must share the DATABUS section of the KSUBUS. Only one AFU may use the KSUBUS at any one time. It is this delay on the KSUBUS which necessitates control signals to be passed over an AFU connection (described below) to control the passage of data. It should be recognized, however, that this multiple use of the KSUBUS is one of the keys to a functionally distributed data base system.

b. The KSUBUS is a hardwired data and control message path (series of buses) which permits the accomplishment of a sequence of actions based on the control computer instructions for moving the subject block of data.

c. Each device, AFU or DMA, has access to the 4 control buses and 2 data moving buses on the KSUBUS. Any one device, however, only connects to the buses it needs to function. By setting an assigned bit on the REQUESTBUS, a device indicates its need to use the DATABUS. A READYBUS bit indicates the availability of a device as a receptor for data. A set TOBUS bit indicates the device destination of the current contents of the DATABUS. A bit is set on the FROMBUS by a transmitter or transmitter/receiver device. When read by a DMA, a set bit on the FROMBUS tells the DMA to return data and the absence of a set bit tells the DMA to store data (a receiver is sending the data) A more detailed description of the

TYPICAL KSUBUS ARRANGEMENT

BIT NUMBER →	0	1	2	3	4	5	6	7
READY BUS	DMXR No 7	DMXR No 6	DMXC No 5	DMXC No 4			HOST DMA	CONTROL DMA
REQUEST BUS	DMXR No 7	DMXR No 6	DMXC No 5	DMXC No 4			DMRC No 1	DMRC No 0
TOBUS	DMXR No 7	DMXR No 6	DMXC No 5	DMXC No 4			HOST DMA	CONTROL DMA
FROM BUS	DMXR No 7	DMXR No 6	DMXC No 5	DMXC No 4				

DATABUS	16 bits
ADDRESS BUS	24 bits

SUMMARY:
1) All devices sense the pertinent bits of the six buses.
2) The transmitters and transmitter/receivers have a bit assigned to them on each of the four control buses.
3) Receivers never receive data from the KSUBUS, but only send incoming data to one of the other devices. They therefore need only a bit on the REQUESTBUS.
4) DMA's must return data to the requesting transmitter so they have a TOBUS bit as well as a READYBUS bit. They are passive devices, acting only on request from another AFU so they use the REQUESTBUS bit of that AFU.

Figure 4

KSUBUS and its organizatio is at Appendix D.

d. Once an AFU has its instructions from the control computer, it sets a bit on the REQUESTBUS to indicate that it needs to use the DATABUS. A receiver AFU would want to move incoming data to memory storage and a transmitter would want to move data out of a memory to be transmitted over one of the direct AFU connections.

3. The AFU Registers

Each AFU has three 16 bit registers which are used to monitor data movement. A Memory Data Register (MDR) holds each halfword until it can be moved to its designated destination by the AFU. A Current Counter Register (CCR) decrements the value of the initial count of halfwords in the data block being moved and serves as an indicator of completion. The Current Address Register (CAR) maintains the value of the memory address where data is coming from or going to, as the case may be.

4. Situation 0

a. The first of the six cases to be discussed is the situation where data is to be moved from the memory of one computer of a set (either the host or the control computer) to the memory of the other.

b. This example might be termed an "internal" transfer. It is performed by the paths used in the remote transfer portrayed in figure 3 except that, once moved, the data does not get transmitted beyond the receiving memory.

c. Initialization of AFU's

1) In this case, the transmitter/receiver AFU is called upon to perform the data movement. The X/R AFU is no more than a transmitter AFU and a receiver AFU put together in one unit. Through this arrangement, the KSUBUS only needs to be used when data goes from one memory to the transmitter side of the AFU and when data goes from the receiver side of the X/R to the receiving memory.

2) The control computer instructs the X/R AFU to move a given size block of data from the DMA servicing the originating memory to the DMA servicing the memory which is to receive the data. The transmitter side of the AFU will then address the

sending memory and the receiver side will address the receiving memory over the KSUBUS through each respective DMA.

3) The transmitter side of the AFU needs its own CAR to keep track of the sending memory and the receiver similarly has a CAR to monitor the addresses in the receiving memory. One Current Counter Register suffices for both sides of the AFU as does one Memory Data Register.

4) The control computer must issue the following instructions:

a) Move a specific block size of data (in halfwords).

b) Get the data through a specified DMA.

c) Start getting the data at a specified address.

d) Store the data through a specified DMA.

e) Store the data starting at a specified memory address.

5) On transmission of the 5th instruction, the AFU is so designed that it commences the hardware controlled transfer of data.

d. Transfer of data (Flowchart, page A-1)

1) The transmitter/receiver first gains access to the DMA which serves the memory where the data is stored. Once this device is available, the AFU requests use of the DATABUS by setting a bit on the REQUEST BUS. If the DATABUS is in use by another AFU, the TOBUS is checked by the X/R AFU to determine if the DMA just checked and found ready will be used by the current DATABUS user. If this is the case, the X/R AFU must wait for this DMA to again be available before making another request for the DATABUS. If the DMA is not being currently addressed and is still ready, the AFU continues in a request DATABUS - chack TOBUS - check DMA/request DATABUS mode until the DATABUS is given to this AFU.

2) One halfword of data is then moved into the MDR of the X/R AFU and the DATABUS is released for the next priority requesting AFU, if any.

3) The AFU now checks the DMA serving the memory which is to receive the data and, when that device is ready, requests the DATABUS. The same sequence of checking and requesting described above is again followed.

4) When the DATABUS is again given to this AFU, the halfword

of data is moved out of the MDR of the AFU and into the current address in memory as indicated by the value in the CAR of the receiver side of the AFU.

5) After each cycle, the AFU decrements its CCR, and increments the CAR's of both the transmitter and receiver side of the AFU. If the value of the CCR is zero, the AFU sends its identity code to the control computer, through the Universal Logic Interface. If the value is non-zero, the AFU processes the next halfword of data.

5. Situation 1

a. One host computer (A) needs data from a memory other than its own or that of its control computer. This situation actually presents two cases as it examines both the receiver and transmitter AFU conditions and forms the basis for the discussions of the following situations.

b. The initial communication between host and control computers is outside the scope of this particular project. Through a series of messages, the location and other particulars (such as size, etc) are established such that control computer A requests control computer B to send the data from computer set B to computer set A.

c. Initialization of AFU's (Figure 5)

1) Control computer A instructs, through a Universal Logic Interface (ULI), its receiver AFU (which is connected to the transmitter at computer set B) to:

a) be ready for receipt of a specific size of data (number of halfwords),

b) store the data in either control memory or host memory (DMA identification), and

c) start the storage at a specific address.

2) At the same time, control computer B, through a ULI, instructs its transmitter AFU (which is connected to computer A's receiver) to:

a) get a block of data of a specified size (the same size as that given to receiver A by control computer A),

b) get the data from host or control memory (DMA identification), and

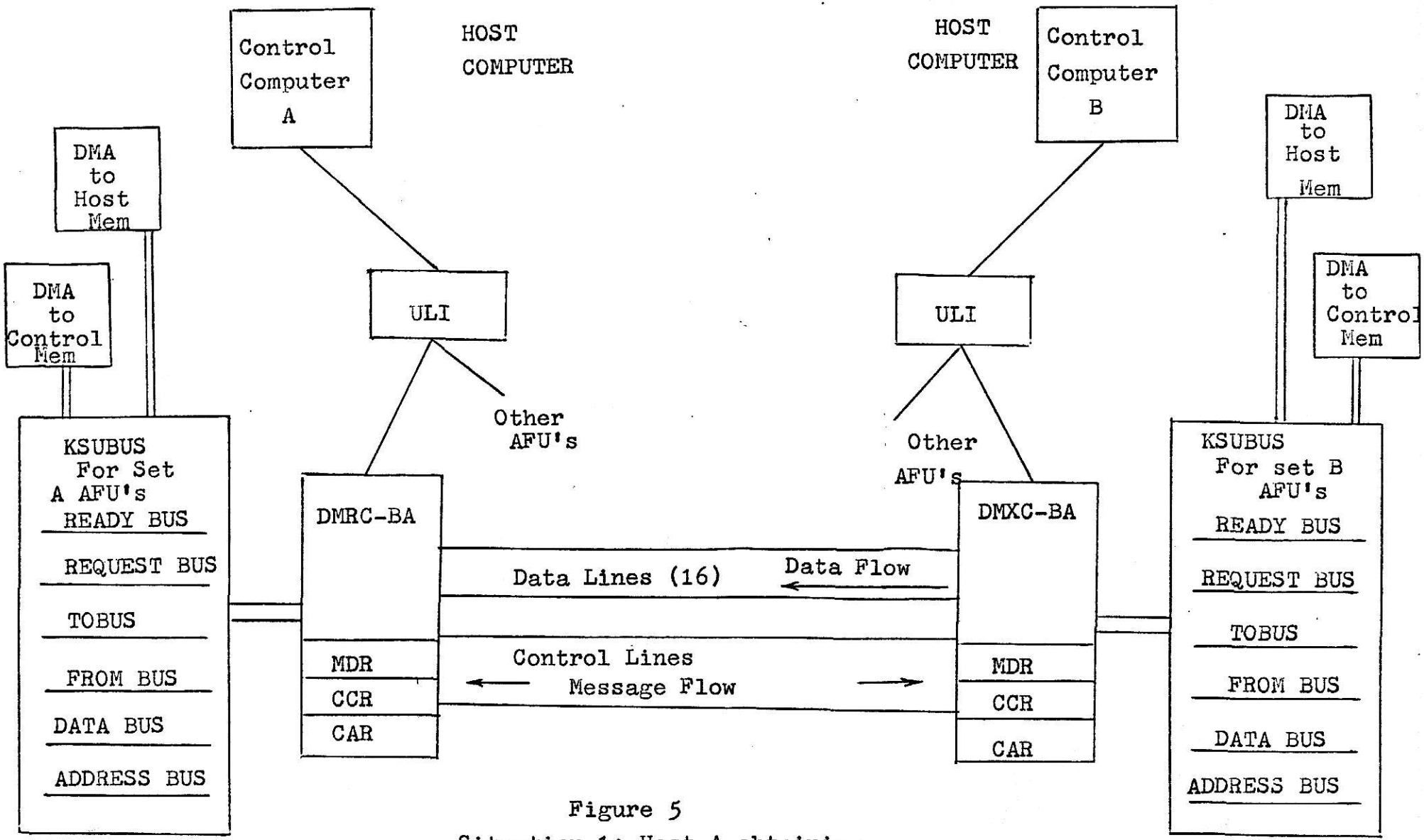


Figure 5
Situation 1: Host A obtaining a data block from Host B

c) start getting this data at a specific memory address.

3) On each end of the AFU connection, at the issuance of the third command, the AFU's are free (and directed) to commence the actual transmission of the data.

d. Transfer of Data (Flowchart, pages A-3 and A-5)

1) Once the transfers of instructions and information are complete to the AFU's, hardwired logic circuits within the DMA's and AFU's accomplish the physical relocation of the requested block of data.

2) Each AFU, the transmitter for data from computer set B to computer set A (DMXC-BA) and the receiver for data from computer set B to computer set A (DMRC-BA), must "look" two ways: toward its own KSUBUS that interfaces between the AFU and its parent control and host memories, and toward the physical wires connecting it to its associated AFU in the other computer set (here called the AFU connection).

3) AFU connection

a) Data is transferred one halfword (16 bits) at a time. The interconnection between the AFU's has 16 wires for this data transfer, so that one bit may move over each wire simultaneously. This provides the high speed characteristic of the system.

b) Also included in this connection are several control wires which coordinate the passage of the data between AFU's so that another halfword is not sent by DMXC-BA until the previous halfword has been stored by DMRC-BA. This communication actually consists of a series of flags or bit conditions set on a specific control wire to indicate an AFU's status. For example:

1. Line 1 might be used to set a condition of the sending AFU which, in effect, asks the receiver if it is ready for data.

2. Line 2 could then be sensed by the sending AFU for the presence of an electrical condition (bit) indicating that the receiver is ready to receive the data.

3. The sender then sends a condition message over a third control line to indicate that the halfword is being sent.

4. The receiver responds with an acknowledgement - of - receipt condition.

c) Other yes and no information can be passed over

additional control lines.

e. In summary, the sequence of action for Situation 1 is:

1) Transmitting AFU (B) gains access to its KSUBUS, fetches one halfword of data from the proper memory via the attached DMA into its MDR, sends this halfword to the receiving AFU (A) and, once AFU (A) acknowledges receipt, gets the next halfword of data.

2) After obtaining the first halfword of data in its MDR, receiver (A) moves over its KSUBUS via the proper DMA to store the halfword in its assigned memory address and then sets its ready flag on the proper control line in the AFU connection.

3) After the entire block of data has been transferred, each AFU notifies its respective control computer, (through the ULI) that the transfer is complete. These messages are triggered by the CCR reaching a zero value. Software in the control computer can then verify the transfer by comparison of the transmitter and receiver CCR's or by exception, where no message is sent unless one control computer or the other notes a non-zero condition in the CCR of an AFU which has indicated condition.

6. Situation 2 (Figure 6)

a. One machine set in the cluster (assume set A) is either not connected or has lost contact with another (assume set C) so that data must be relayed through set B. This situatio might exist when no original link between set A and set C was provided because there was very little data transfer expected between them or several other reasons.

b. This case differs from the previous case in that once a halfword of data is received by the receiver at set B (DMRC-CB) from the transmitter at set C (DMXC-CB) it is not relayed to either memory in the set but, rather, to the transmitter which connects set B to set A (DMXC-BA).

c. This task is established in the cluster by coordination between the three concerned control computers which convey the request that set C has a particular size of data which must go via set B to set A.

d. AFU Initialization

1) The same three categories of information are sent by

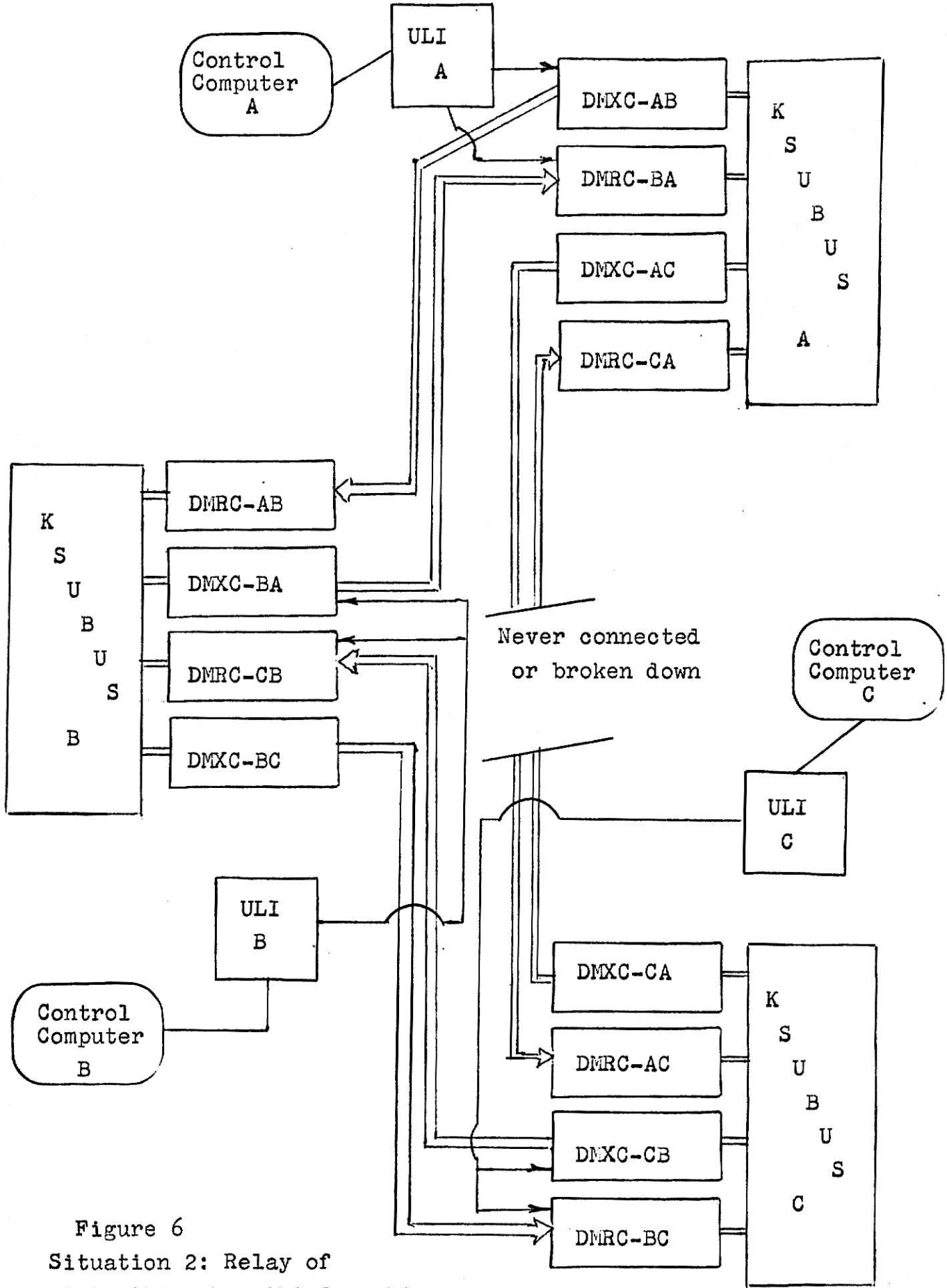


Figure 6
Situation 2: Relay of data through a third machine

each control computer to its applicable AFU's as was sent in Situation 1. Control computer B, however, must be alerted to arrange a relay task as opposed to simply coordinating a receive and transmit job. This will invoke a different program in the control computer which will address two AFU's with appropriate instructions.

2) Control computers C and A will establish their sending and receiving AFU's, respectively, much the same as in the direct transfer of data except that different AFU's will be utilized (those connecting to set B instead of to each other).

3) Control computer B will initialize DMRC-CB to the size of the incoming data and instruct that device to foward the data to transmitter DMXC-BA (Flowchart, page A-7). DMXC-BA will be advised of the size of data and told to wait passively until loaded from the KSUBUS. The Current Address Registers of DMRC-CB and DMXC-BA are not used in this case.

4) One impact of a relay situation is the delays in each of the three KSUBUS's involved. If any AFU is bottom priority (highest order bit) on its REQUEST BUS and a particularly busy period is realized by that set during the relay mission, a bottleneck, although not a deadlock, would occur.

e. On completion of this task, each AFU will notify its control computer of completion and verification or exception messages can be exchanged between control units.

7. Situation 3

a. A receiving computer set desires to retain a copy of data that it is fowarding from one computer to another.

b. This situation is an extension of Situation 2 and Figure 6 still applies. Rather than the relay receiving AFU simply fowarding the data received to a transmitting AFU, it must also send the data to a memory address in its own computer set. This is accomplished by instructions to the relay receiver AFU to send each halfword to both the relay transmitter and a designated DMA. The receiver cannot send the halfword until both the DMA and the transmitter are ready.

c. Initialization of AFU's

1) Sets A and C initialize their AFU's in the same manner

as in the previous situations. Set B, however, must instruct its receiver to address two destinations over the TOBUS as each word is placed on the KSUBUS. The set B transmitter is set to be ready for data from the receiver, DMRC-CB, so there is no starting address for it to look for.

2) The receiver in set B must now watch for two flags to be set on the READY BUS: the DMA and the transmitter to set A. The receiver's MDR, CCR, and CAR are set in the normal manner. A command is also given, however, to watch for the appropriate bits on the READY BUS which designate the selected DMA and the transmitter. Hardware logic will then accomplish the data movement.

3) The transmitter of set B is simply a relay mechanism. It need only be instructed as to the size of the data as in Situation 2, and that this is a fowarding tas. The transmitter has no concern or appreciation that the data is being copied - only that it is fowarding data.

4) The key to fowarding data with copy, therefore, is that the middle AFU, at set B, is instructed to insure that both the selected DMA and the transmitter are ready before sending the halfword of data (Flowchart, page A-9).

5) After the last halfword is sent, all CCR's in the chain of data movement should be at zero value and each AFU used in this task will independently signal its control computer that it is finished.

8. Situation 4

a. It is desired to send a given block of data to multiple receivers.

b. This offers the extreme situation in moving data within a cluster. It can combine all the previously cited cases into one task and, while this would be an exceptional case, it illustrates how the hardware interfaces can efficiently move a block of data by minimizing the number of actual data transfers.

c. To illustrate this case, the basic mode, that of using the transmitter/receiver AFU, will be described. To move the data to sets not directly connected to the sending set, however, an expansion of the foward or foward with copy cases described in

Situations 2 and 3 could be done to extend the data to all computer sets in the cluster in 2 or 3 actual moves (Flowchart, page A-11).

d. Initialization of the X/R AFU (Flowchart, page A-11)

1) The same basic instructions are sent to the X/R AFU as are sent to perform a DMA to DMA transfer. The receiver side of the AFU, however, is instructed to place the halfword of data on the DATABUS only after the transmitters which are specified to foward the data are ready. Conceivably, in a case where a new data base is being distributed onto the system, the X/R AFU might be required to send the data to all the transmitters on the KSUBUS.

2) The following instructions are typical of the type which would be sent to the X/R AFU for this task:

a) Send a block of data of a specific size (load the CCR)

b) Get the data from a specific DMA

c) Send the data to transmitter 1

d) Send the data to transmitter 2

e) Start getting the data at a specified address (load the CAR of the receiver side -(the CAR on the transmitter side is left indeterminate).

3) The X/R AFU, on receipt of the starting memory address, commences the transfer of data out of memory (via the DMA) to the designated transmitters. Once a transmitter AFU is ready for the X/R AFU to send the data, however, it will not be used by another device so once all transmitters are ready, the X/R AFU only needs to request and get the DATABUS.

4) The matching receivers for each transmitter are initialized in the same manner as in situations 1, 2, or 3 (Flowchart, page A-5).

e. As in the previous cases, when the CCR's of all AFU's go to zero, each AFU signals its control computer that the task is completed.

PART IV: Driver Routines

1. In each situation, before issuing instructions to any Autonomous Functional Unit, the control computer must, through

a higher level software program, determine the size of the data block and where it is in memory, if the task is to send it, or where it is to go in memory, if the task is to receive it. The type task to be performed must normally be known to the control computer. In a relay task, for instance, the initial sender set and terminal receiver set of computers ast much as they do in a simple transfer and will be initialzed as though that were the task. The middle, or relay set, however, must know that the data is not to be copied but, rather, sent to the appropriate transmitter on the KSUBUS. The transmitter will not be requesting data as it usually does in a simple transfer, but accepting it as it is "automatically" sent. This state of "knowing" the task is conveyed by appropriate control computer driver routines.

2. Any set of computers in the Functionally Distributed Data Base System must be able to perform any of the tasks depicted in the six cases discussed in Part III. When invoking any of the routines, the control computer will communicate with the required AFU's to initialize them with the appropriate data. In the event that numerous AFU's need to be initialized, (such as in the case of multiple fowarding) no special sequence is necessary as no AFU can start moving data until all of its counterparts are ready.

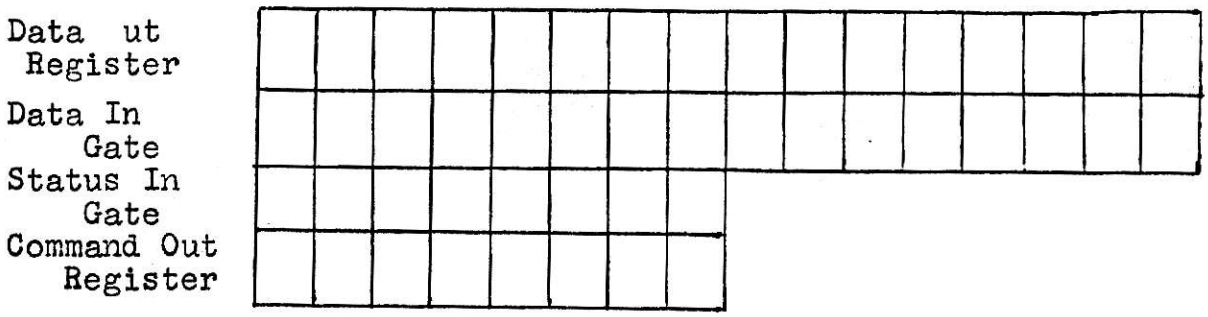
3. To communicate with its AFU's, a control computer uses a Universal Logic Interface (ULI). This is a master-to-slave relationship in that the control computer can always interrupt the AFU but the AFU can only request, through the ULI, to interrupt the control computer. This arrangement enables the control computer to complete a communication to other AFU's, the host computer, or any of the other control computers without interruption. It can perform processing, if desired, and manage AFU's at the same time.

4. The ULI

a. A ULI provides a set number of communication channels by acting as a relay between the control computer and the AFU. In this sense, the ULI can be viewed as a buffer for the control computer.

b. (Figure 7) The ULI has two registers and two sets of gates

UNIVERSAL LOGIC INTERFACE (ULI)



Type commands: (Interrupt and mode bits are constant as shown)

I	I	HW	C	C	A	A	A	
0	1	1	0	0				Select an AFU
0	1	1	0	1				Stop AFU
0	1	1	1	0				Set Read CCR of AFU
0	1	1	1	1				Spare

Type address bits:

1	1	1	XRCC No. 7
1	1	0	XRCC No. 6
1	0	1	DMXC No. 5
1	0	0	DMXC No. 4
0	0	1	DMRC No. 1
0	0	0	DMRC No. 0
0	1	1	not used
0	1	0	not used

Source: Interdata User's Manual and notes of Dr. M.A. Calhoun

Figure 7

which are used to relay information to and from the registers of the AFU's. The DATA OUT register can pass up to 16 bits of data from the control computer to the AFU. The DATA IN gates pass up to 16 bits of data from the AFU to the control computer. The STATUS IN gate is used to pass an AFU's identification code when the AFU requests to contact the control computer (principally on completion of the AFU's task). The COMMAND OUT register is used to relay a type command, identified by 2 control bits, to a particular AFU, identified by 3 control bits. The 4 combinations of 2 bits can then provide 4 type commands to the AFU's addressed through a combination of 3 identification bits.

Control bits	Command
0 0	Select AFU AAA (alert the AFU)
0 1	Stop AFU AAA (accept a "done" report)
1 0	Read CCR of AFU AAA
1 1	not used

AAA is a combination of bits which uniquely identifies one of the AFU's serviced by this ULI.

c. In addition to the 5 user bits, described above, there are 3 control bits, 2 of which set the interrupt code for the control computer and 1 bit which declares the mode, halfword or byte, for the system. As the halfword mode is habitually used, the mode bit will always be the same and, while the ability to disable or disarm the interrupts will remain, it is a separate discussion to review the impact of interrupt disabling so these 2 bits will always be in the enable mode for the purpose of this report.

d. The ULI is hardwired. The above commands are produced by the applicable driver program in the control computer software and relayed through the gates and registers.

5. Driver Routine Coding

a. The following driver routines are called as sub routines of an initializing program which obtains the critical data such as data size, location, etc.

b. If it is desired to move the data from the present memory location to the control computer's memory (Situation 0), the following routine will be called in the control computer:

```
ROUTINE_INTERNAL (A, B, C, X, E)
   Load the CCR with A
   Get the data through DMA B
   Set the CAR(in) to address C
   Send the data to DMA X
   Set the CAR(out) to address E
END ROUTINE_INTERNAL
```

The act of calling this particular routine will alert one of the X/R AFU's on the KSUBUS. The value of A will equal the size of the data in halfwords, B will equal the device identification code of the sending DMA, and C will equal the starting memory address in the sending memory. The instruction of where to send the data will come to the routine as a compacted instruction whose bit values will depict which devices on the KSUBUS will be recipients of the data. This instruction will be calculated by hashing all input parameters representing device identification codes (one for each eligible device) and assigning the resulant value to X. In this first case, only the receiving DMA will be contacted. The next case will use a different instruction.

c. Another routine using the X/R AFU is one to move data to multiple receivers (Situation 4). In this case, the above routine is again called but, this time, with a different compacted value for the fourth parameter.

```
ROUTINE_INTERNAL (A, B, C, Y, F)
   Load the CCR with A
   Get the data through DMA B
   Set the CAR(in) to address C
   Send the data to devices Y
   Set the CAR(out) to address F
END ROUTINE_INTERNAL
```

A, B, and C have the same values as before. Y represents the compacted value of the transmitters which are to receive this data and the value of F is ignored (data is coming from a receiver, not a memory).

d. Data movement in situations 1, 2, and 3 is accomodated by one other routine. this same routine is called for each type task but the AFU which is called and the calling parameters will

be passed by the main routine so that the effect of the routine will vary as the values of the parameters vary.

1) In a simple set to set transfer (Situation 1), the routine for the transmitting set will have the form:

```
ROUTINE_SEND (A, X, C)
   Load the CCR with value A
   Get the data from DMA X
   Set the CAR to address C
END ROUTINE_SEND
```

In this case, the routine is addressing a transmitting AFU which is hardwired to a receiver and can send the data nowhere else so there is no requirement to tell it where to send the data.

The receiver set form will be:

```
ROUTINE_SEND (A, Y, D)
   Load CCR with value A
   Send the data to device Y
   Set the CAR to address D
```

The receiver could send the data to any other eligible device on its DMA so the value of Y could be a DMA identification or a compacted value of several transmitters. In the case of just a DMA, the value of D is the starting address in memory. The compacted case is discussed next.

2) Situation 2, relaying data for a third set, and Situation 3, copying the data as it is relayed, will also use the above routines. In Situation 2, the original and terminal sets will initialize their AFU's exactly as above. The middle set AFU's will be initialized with two calls of the routine: one for the appropriate receiver in the form:

```
ROUTINE_SEND (A, U, C)
   Load the CCR with A
   Send data to device U
   Load the CAR with value C
```

and one for the appropriate transmitter in the form:

```
ROUTINE_SEND (A, V, C)
   Load the CCR with the value A
```

Get the data from device V
Set the CAR with address C

U and V will be the compaction of bit values which will tell each AFU whxt devices to transfer data to or from. In Situation 2, the device for each AFU to exchange data with will be the other AFU and the CAR value (C) will be ignored by both AFU's. In Situation 3, U will represent both the designated DMA and the designated transmitter. The CAR value will be taken by the receiver as the starting memory address but will be ignored by the transmitter.

e. The above routines will be called by a higher program which will have determined which AFU is to be called and who that AFU will send data to. Such calls will effectively say:

Call	Driver routine action
"Send data to set B"	/* Invoke ROUTINE_SEND to the transmitter AFU which is connected to the receiver in set B.
"Recieve data from set C"	/* Invoke ROUTINE_SEND to the receiver AFU which is connected to the transmitter AFU at set C.
"Relay data from set B to set C"	/* Invoke ROUTINE_SEND to the receiver AFU connected to the transmitter AFU at set B. Invoke ROUTINE_SEND to the transmitter AFU connected to the receiver AFU at set C.
"Send data to computer sets D, E, and F"	/* Invoke ROUTINE_INTERNAL to an available X/R AFU. A compacted value will be passed to direct it to the appropriate transmitter AFU's. /* Invoke ROUTINE_SEND to the transmitter AFU connected to set D, again to the transmitter AFU connected to set E, and again to the AFU connected to set F.

PART V: Conclusion

1. This report has addressed one of the lowest software levels in the Functionally Distributed Data Base Management System. The local driver routines are the last software commodities before the logic gates and electronic circuits of the Autonomous Functional Units take over. For this reason, the report has dwelt on the hardware/software interaction.

2. From a simple movement of one block of data between a host's memory and its control computer's memory, to the extensive activity in multiple fowarding of data, two basic driver routines suffice. The control computer, when given the task of arranging the movement of data, is effectively told which AFU to initialize. By its design, the AFU is going to react with a predictable sequence of events which will be dictated by the parameters of the calling program.

3. Additional programs which will return error conditions and diagnoses of hardware failures will be necessary to the control software package for dealing with AFU's. While these are not treated in this report, their importance has been recognized throughout the development of the project.

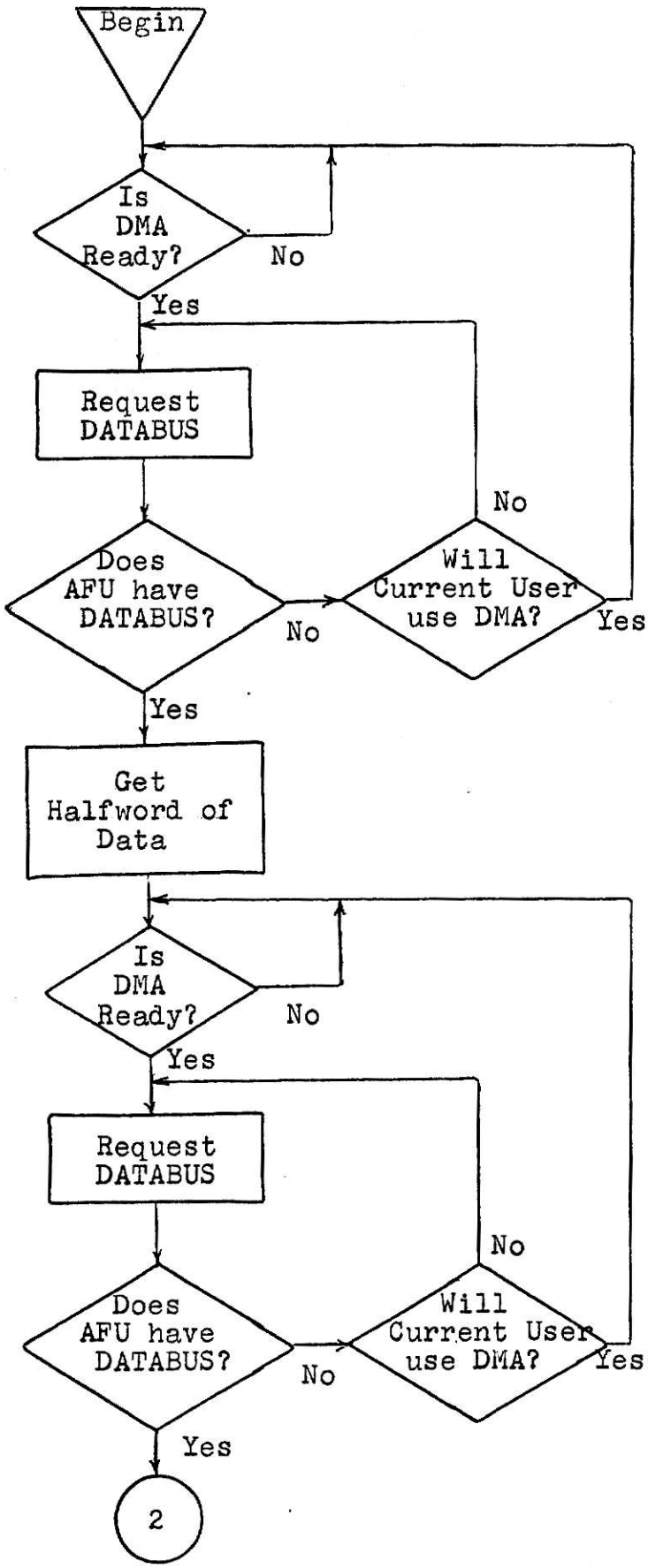
4. As of the writing of this paper, a working prototype of the computer set cluster has not yet been completed. For this reason, the application of the final form of the routines was not possible. In lieu of that, however, simulated routines have been prepared and assembled on the Interdata 7/16 computer and are enclosed at Appendix C.

5. While the successful application of the data moving apparatus is still in the future, this report is an important, if small, part of that structure. The basic drivers presented here should serve as part of the initial software package in control computers of the Functionally Distributed Data Base Management System.

APPENDIX A

DATA TRANSFER FLOWCHARTS

Transmitter/Receiver AFU During an "internal" transfer, Situation 0



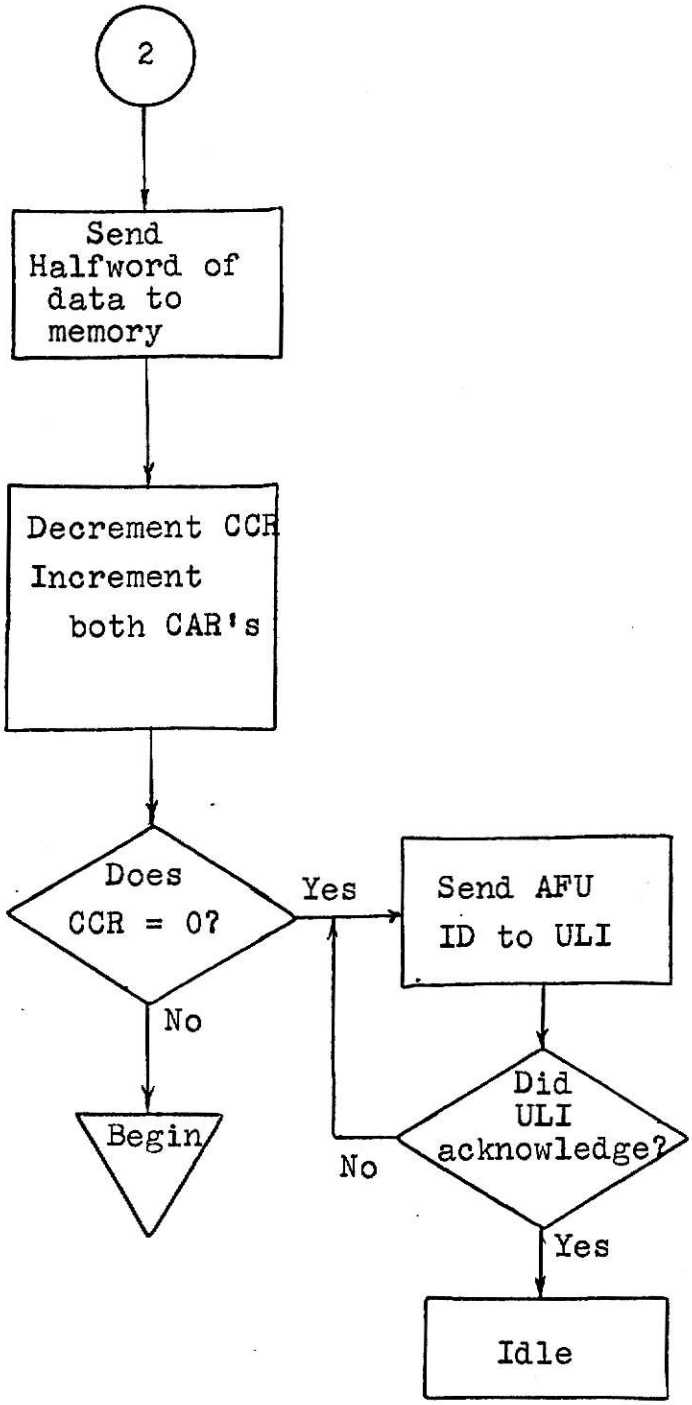
/* Check to see if the "sending" DMA is ready.

/* If DATABUS is in use, see if the current user is accessing the desired DMA (check the TOBUS). If so, the DMA must be requested again.

/* Receive halfword of data into MDR from sending memory.

/* Check to see if "receiving" DMA is ready.

/* Check DATABUS in the same manner as above.

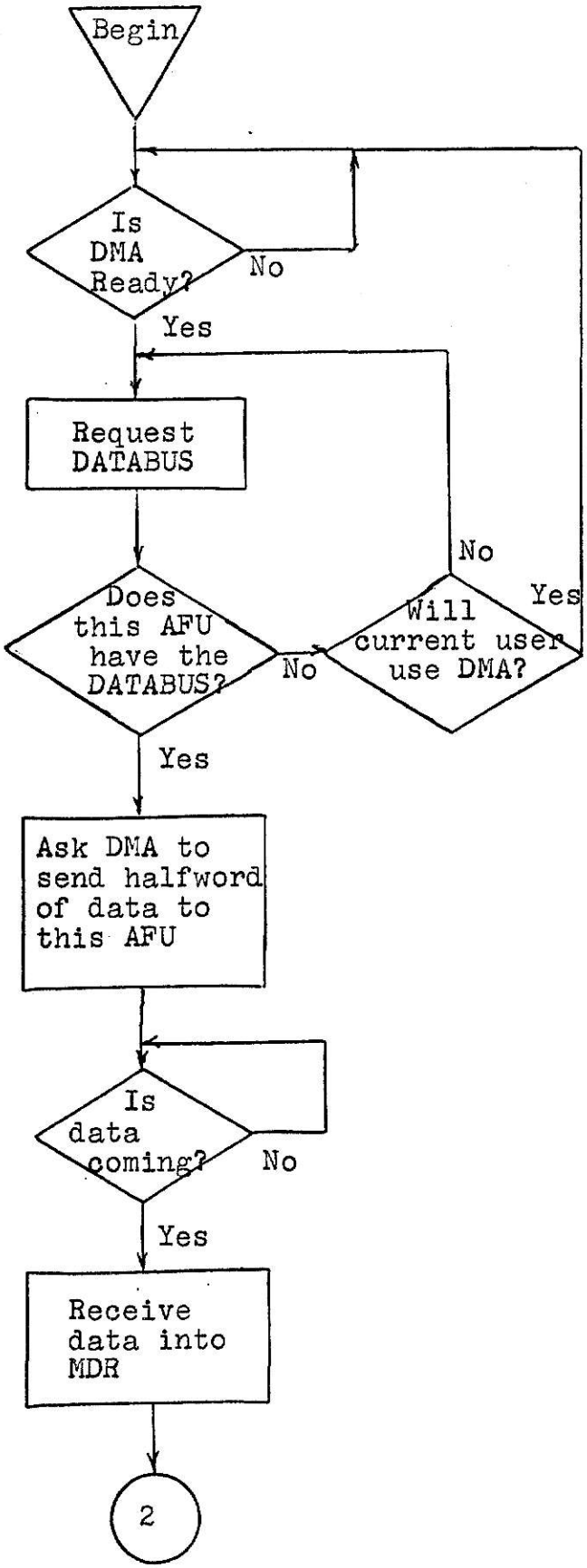


/* Send the halfword of data to the receiving memory.

/* Counting registers are set to the next halfword of data to be moved.

/* If the value of the CCR is zero, notify the ULI that the task is finished (send AFU ID) and, after acknowledgement, go to idle. Otherwise return for the next halfword.

Transmitter for Situation 1 or initial transmitter in Situations 2 and 3.



/* Check READYBUS bit of desired DMA. Loop until it is ready.

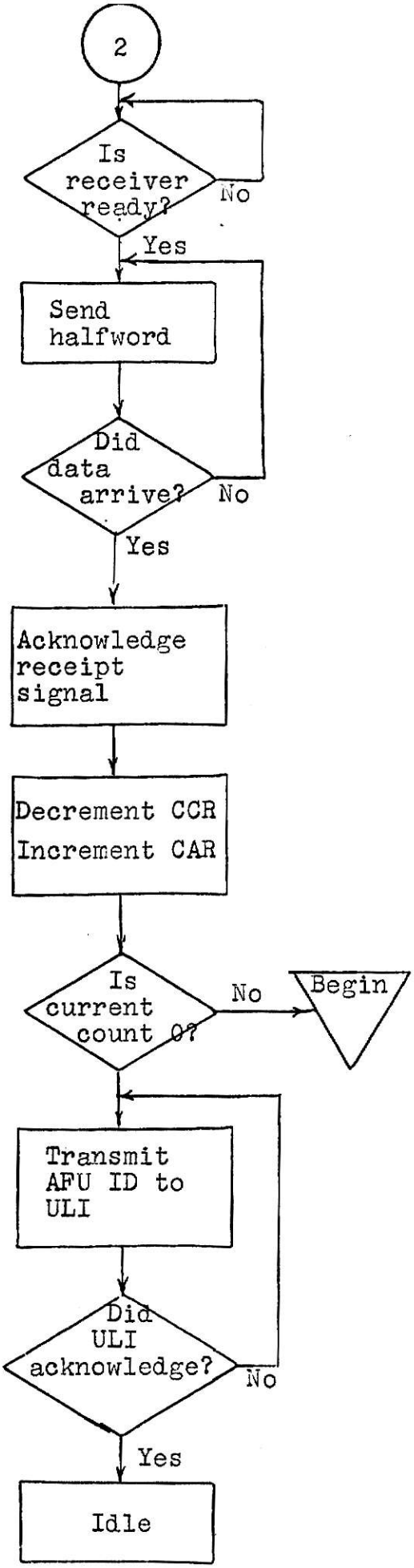
/* Set this transmitter's bit on the REQUEST BUS.

/* If any bits ahead of the one of this AFU are set on the REQUEST BUS, it must be requested again. If the using AFU will use the ready DMA, the DMA must be requested again.

/* This command sets the AFU's bit on the FROMBUS so the DMA can return data to this AFU. It also resets the DMA bit on the READYBUS.

/* Check the TOBUS. The bit of this transmitter will be set when data is coming.

/* The DMA is through for this cycle so its READYBUS bit is set again for any waiting AFU.



/* Query the appropriate control line for a condition code equal to ready. An affirmative answer equates to a "send it" instruction. Otherwise, loop.

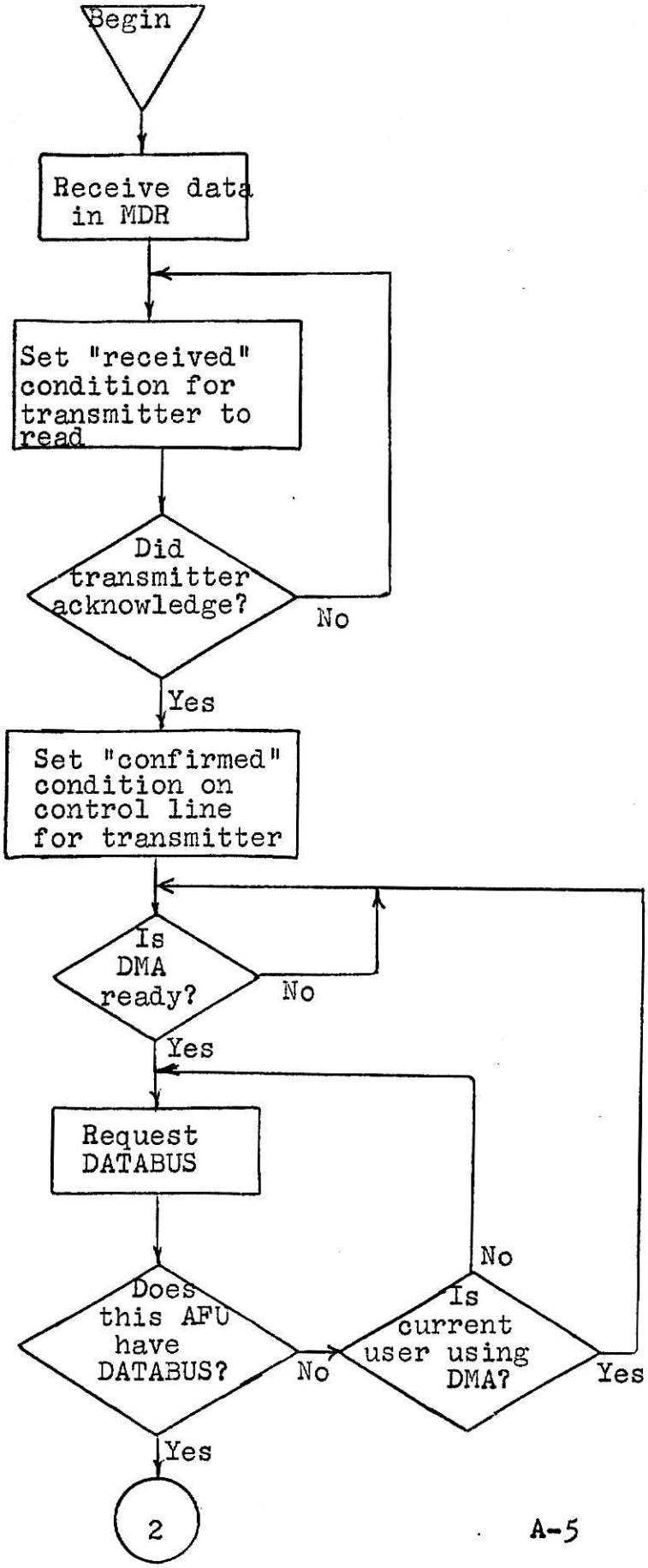
/* The halfword of data is sent over the 16 data wires connecting to the receiver.

/* Transmitter is looking for a condition code on a control line. A "yes" will be indicated by a prescribed condition. The transmitter can acknowledge the acknowledgement, although not required to do so.

/* The AFU counts down the halfword count and increments the address register 16 bits to the next memory address.

/* If there is more data remaining, return for another cycle. If the CCR count is zero, the AFU signals the ULI that it is done by sending its identification code to the ULI. This signal is looped until acknowledged and then the AFU is placed in an idle mode.

Receiver for Situation 1 or any terminal addressee in Situations 2, 3, and 4



/* Ready flag on AFU connection is set.

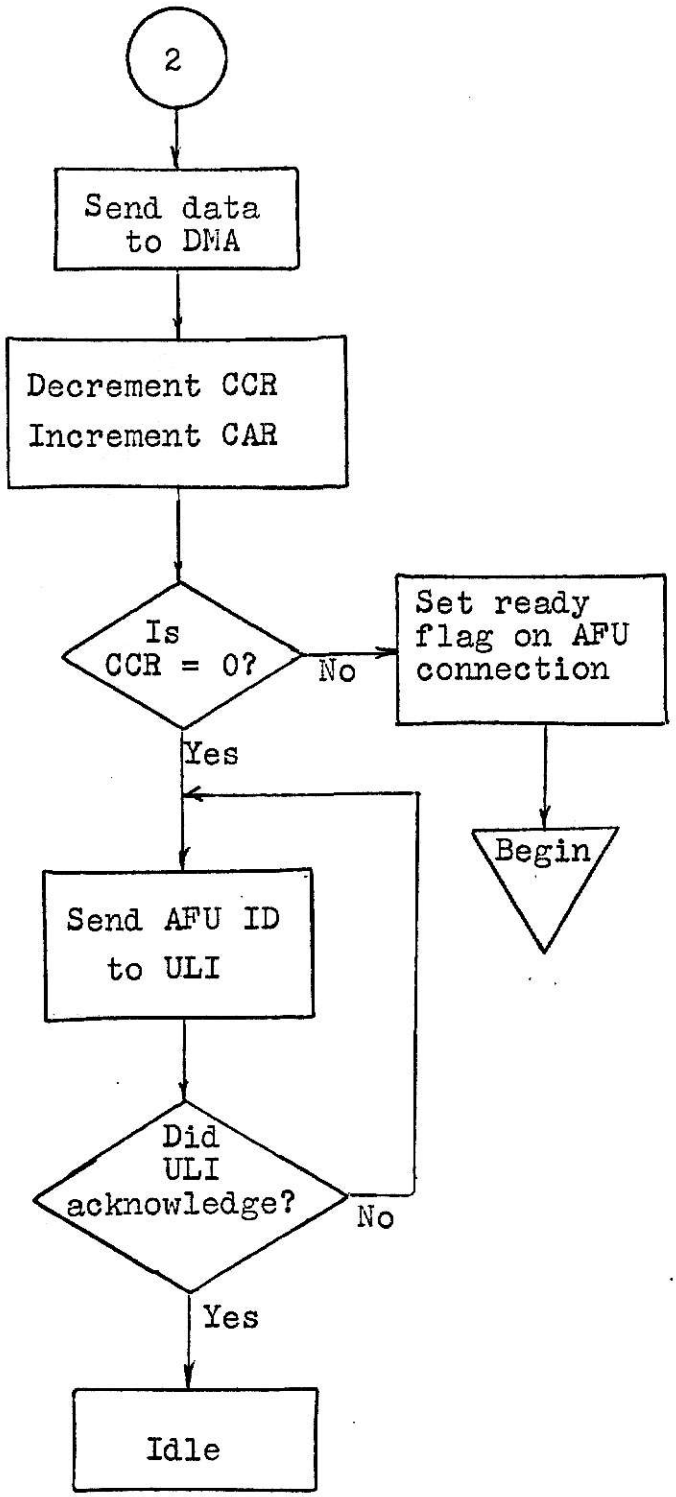
/* 16 bits of data are sent by the associated transmitter.

/* Set a bit on the appropriate control line for the associated transmitter to sense that the halfword was received.

/* Transmitter will respond by setting a bit on a control line. Receiver will confirm the exchange.

/* Check READYBUS for appropriate DMA

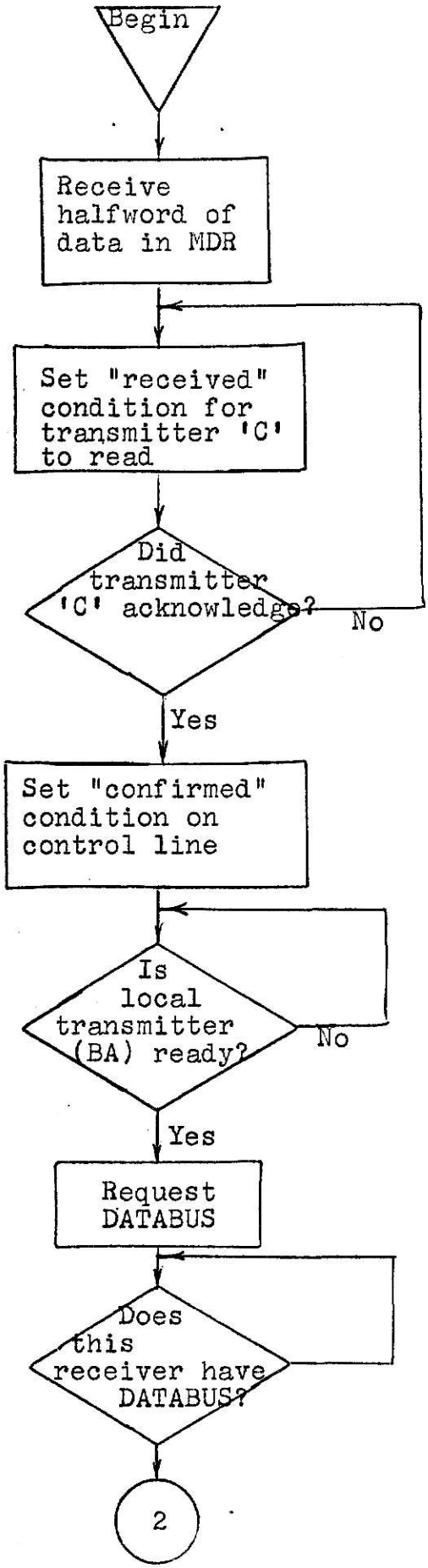
/* Set bit on the REQUESTBUS and check the bits ahead of the one for this AFU. If there is a user, check the TOBUS to see if the desired DMA is being used. If so, the DMA must be rechecked again until it is ready. Then the DATABUS must be rerequested.



/* Send the halfword to the memory address in the CAR through the DMA.

/* The standard "clean-up" sequence of events to either return for the next halfword or go into an idle mode is run.

Middle Receiver in foward mode, Situation 2



/* Get data from transmitter at set C.

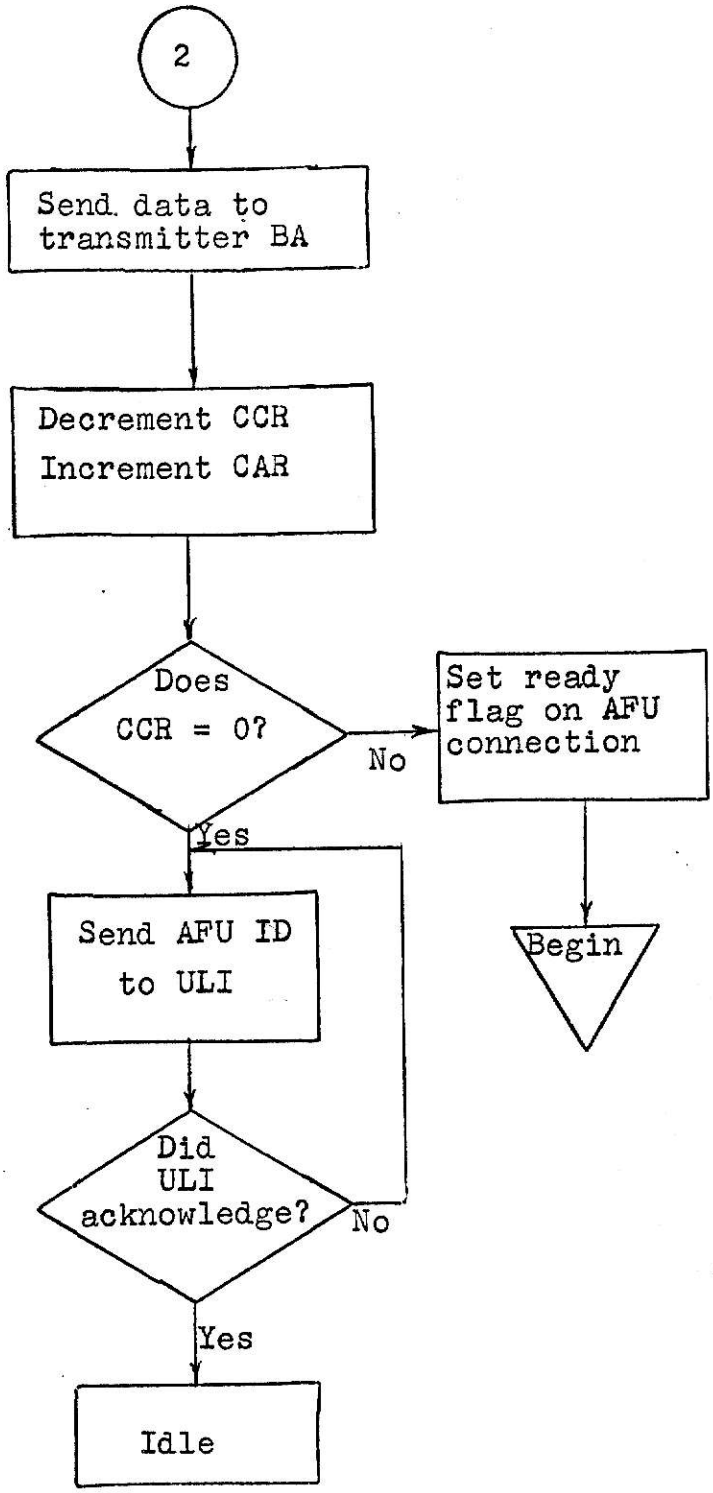
/* Acknowledge receipt of halfword of data.

/* If there is no response to the acknowledgement of receipt message, set the bit again.

/* If there was a response, confirm this communication with another set bit on a control line.

/* Loop until transmitter to set A is ready. Once it is ready, set the request bit for the DATABUS.

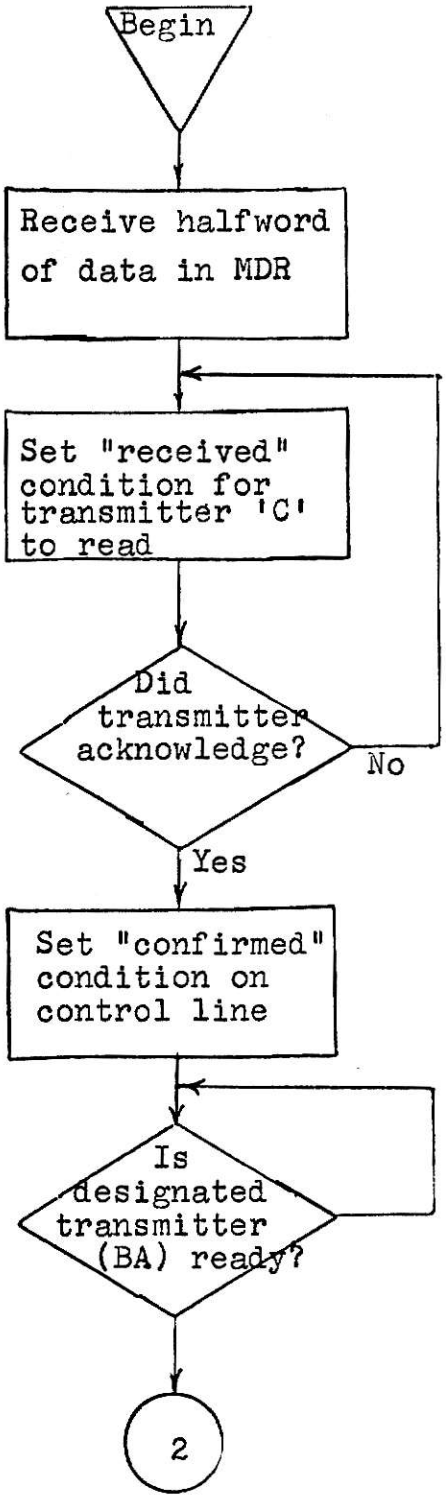
/* Once the B to A transmitter is ready, it will only be used by this receiver so all that is needed is the DATABUS.



/* The transmitter monitors the TOBUS and will pick off data when its bit is set on the TOBUS.

/* The AFU then does the normal "clean up" functions after completing the cycle.

Middle Receiver in a copy and foward mode, Situation 3



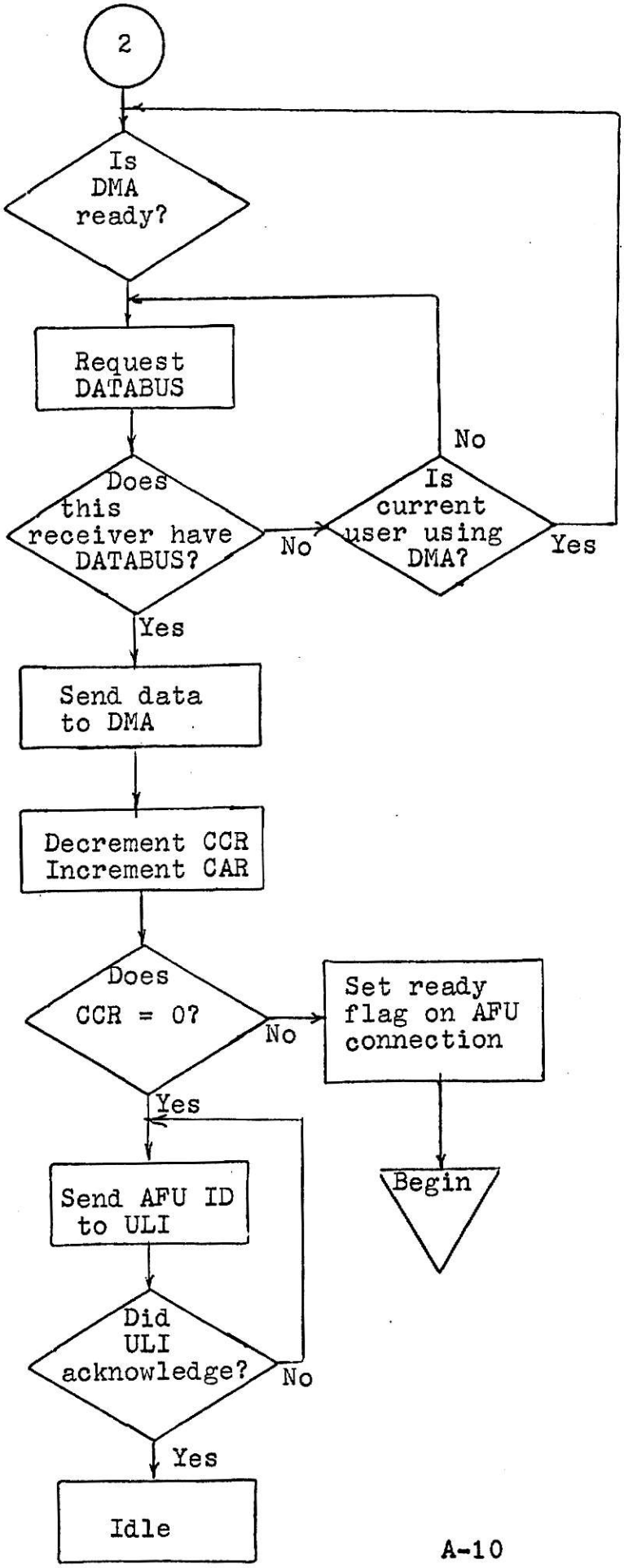
/* Get data from transmitter at set C.

/* Acknowledge receipt of halfword of data by setting a bit on the control line.

/* If no response from transmitter, set acknowledge bit again.

/* If there is a response, set a bit on the control line to confirm this communication.

/* Once the transmitter to A is ready, it will stay ready until the receiver sends a halfword of data.

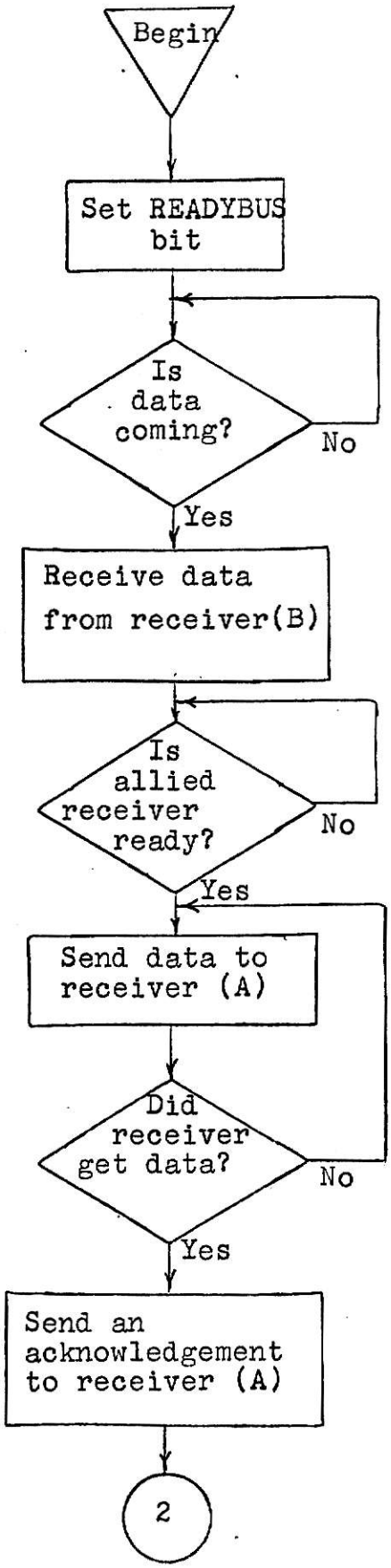


/* If DATABUS is in use, the requested DMA must be checked (by looking at that DMA's bit on the TOBUS) to see if it is in use. If so, the DMA must be rechecked until it is free.

/* The data is sent to the designated DMA. The transmitter, having been instructed, copies and sends the data.

/* AFU then does the normal "clean up" sequence for this cycle.

Middle Transmitter in a foward, foward with copy, and multiple foward mode; Situations 2, 3, and 4



/* Once initialized, this AFU indicates it is ready by setting its bit on the READYBUS.

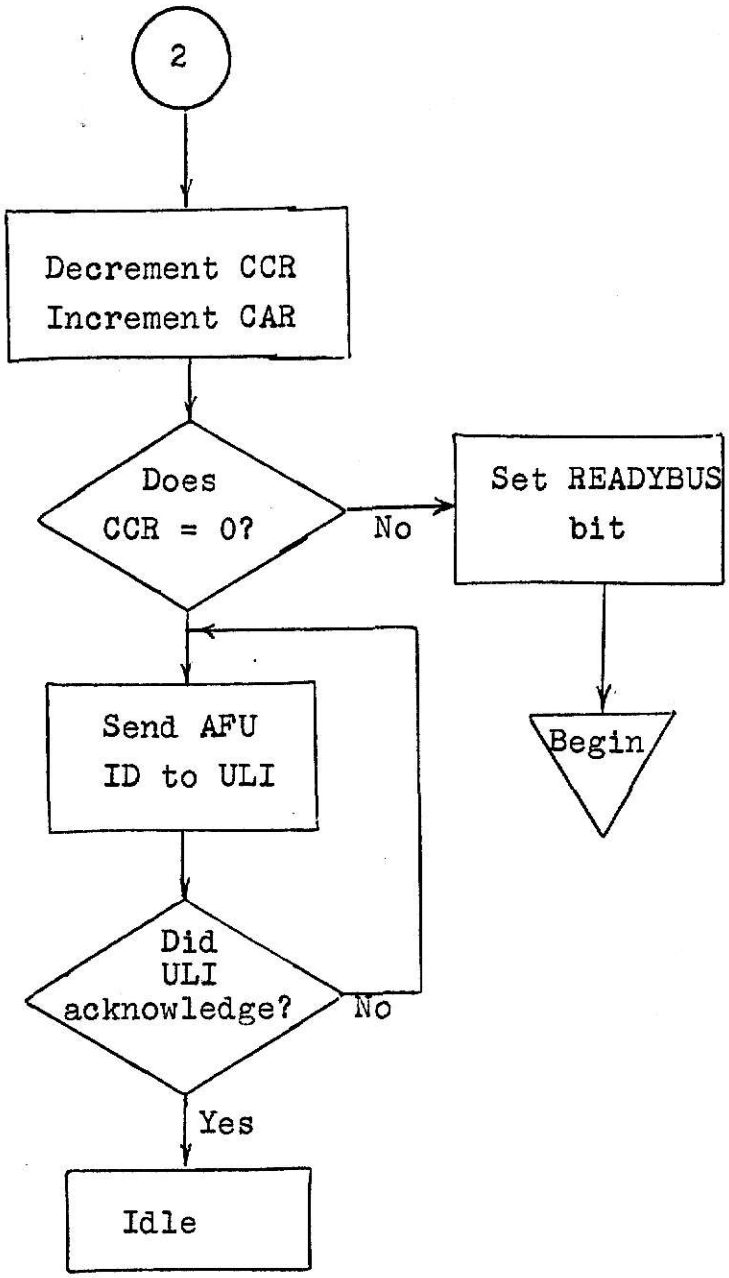
/* Watch TOBUS for this information. This AFU's bit will be set when data is coming.

/* Get data from receiver and reset the bit on the READYBUS.

/* Check over the control lines.

/* Look for a bit being set on the control wire to indicate reception.

/* Let the receiver know that this AFU is aware of the successful transmission.

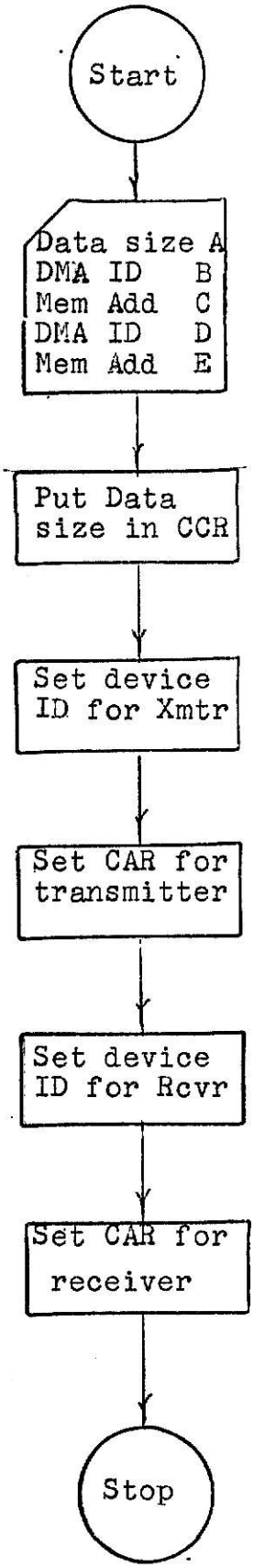


/* Do the normal AFU "clean up" for this cycle.

APPENDIX B

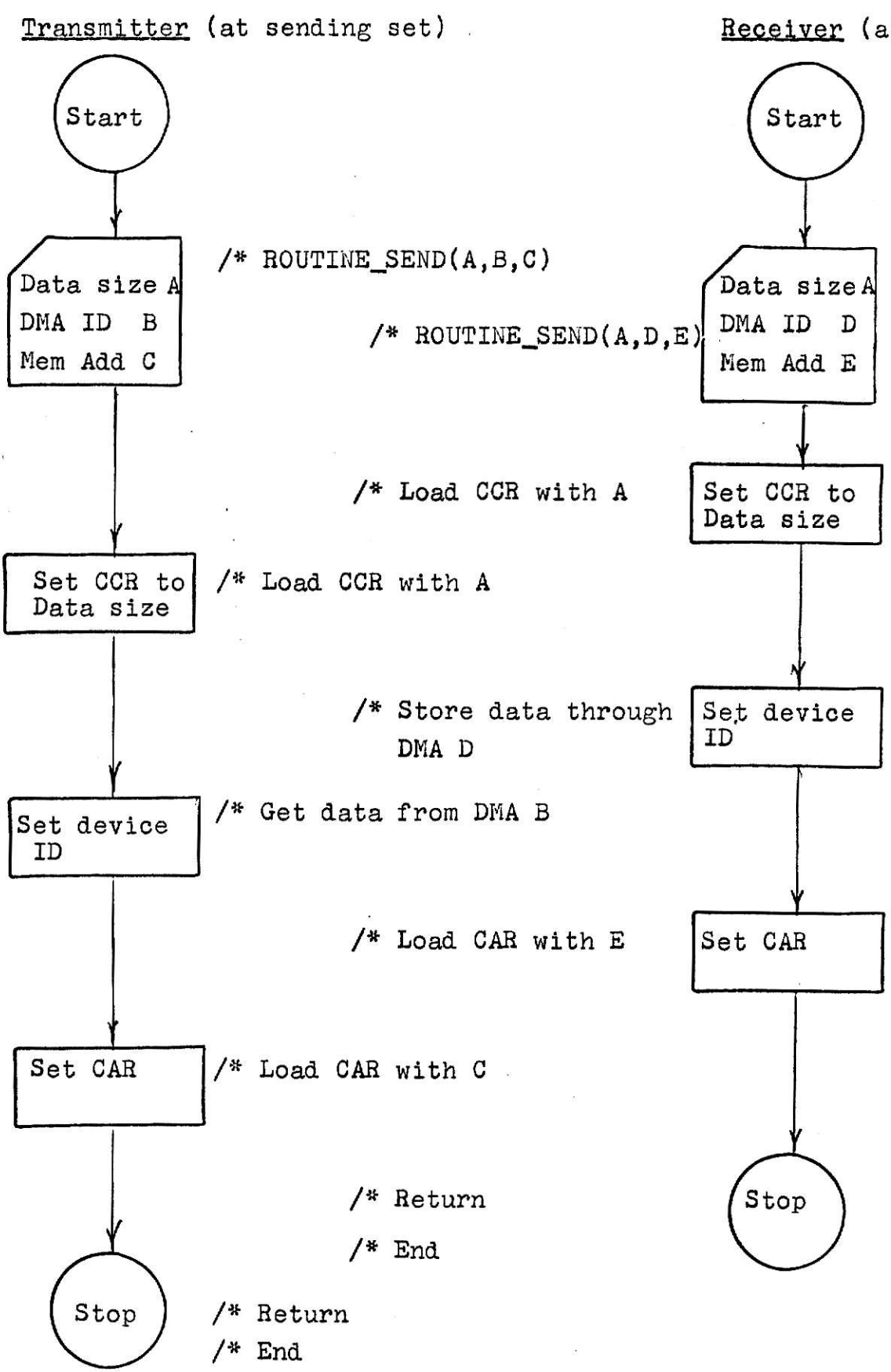
DRIVER ROUTINE FLOWCHARTS

Flowchart and code for Case 0, memory to memory transfer



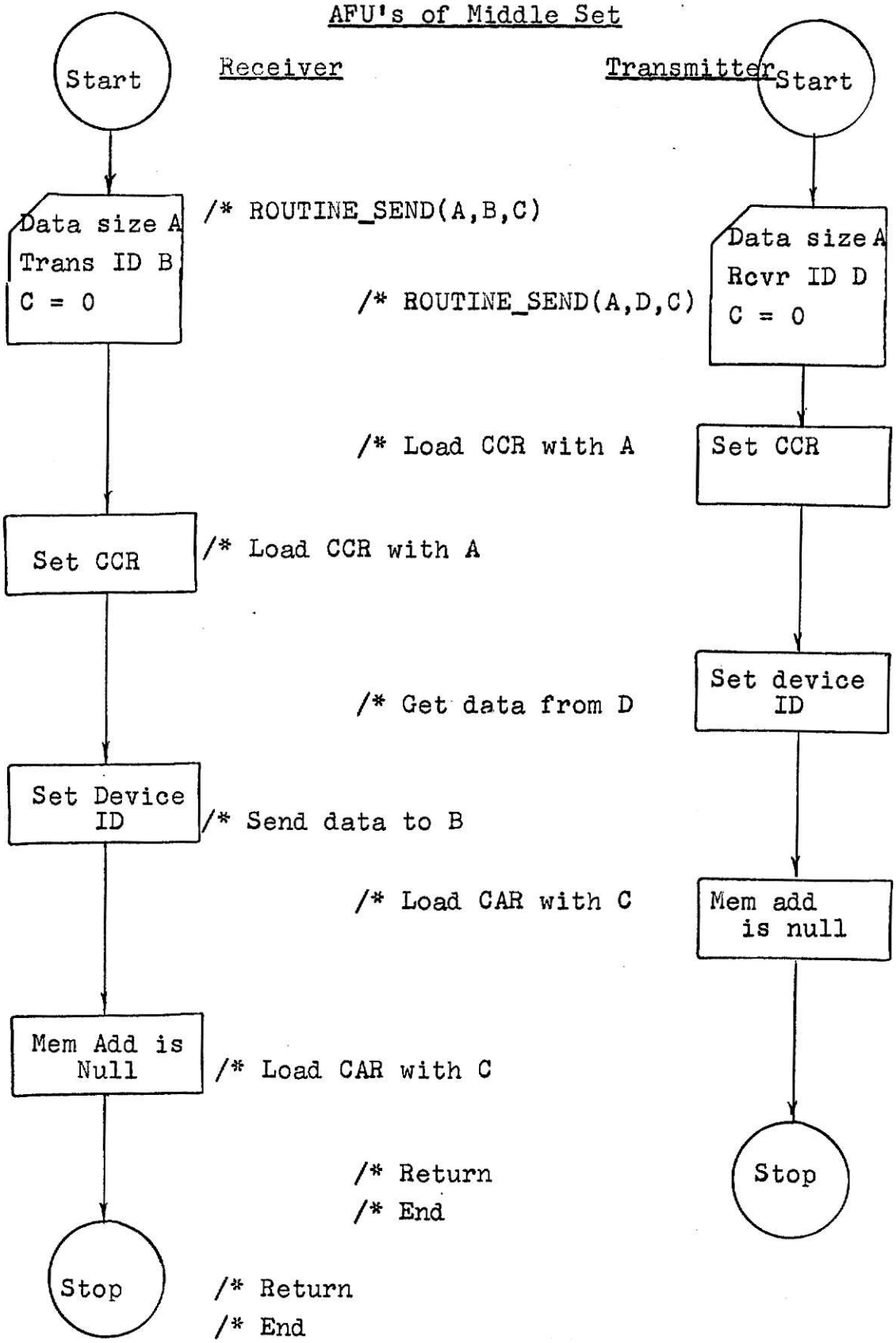

```
/*  ROUTINE_INTERNAL (A,B,C,D,E)

/* Load CCR with A

/* Get data from DMA B

/* Load CAR with C

/* Store data through DMA D

/* Load CAR with E

/* Return

/* End
```

Flowchart and codes for Case 1, set to set transfer

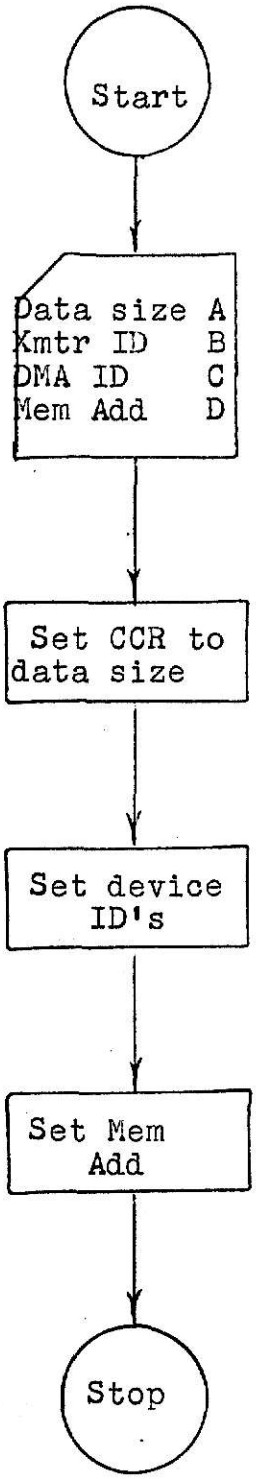


Flowchart and codes for Case 2, fowarding data from one set to another

The originating transmitter and the terminal receiver are coded as for Case 1.

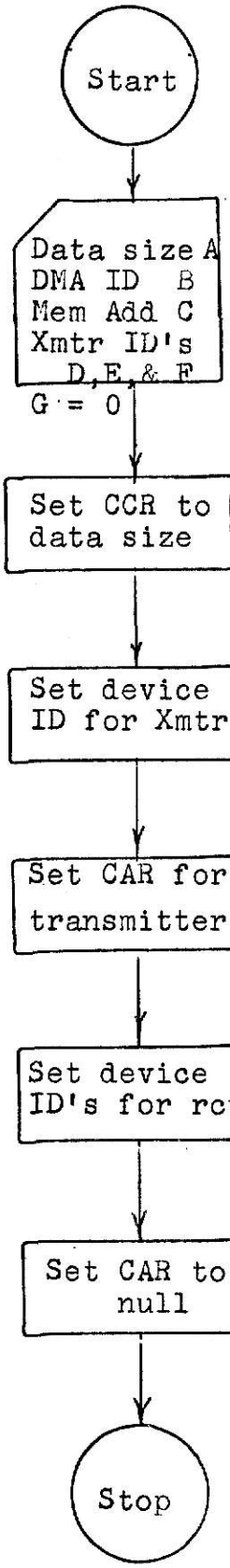


Flowchart and code for Case 3, foward data with a copy retained. The only difference from Case 2 is in the code for the middle receiver.




```
/* ROUTINE_SEND(A,BC,D)
/* Load CCR with A
/* Send data to devices B and C
/* Load CAR with D
/* Return
/* End
```

Flowchart and code for Case 4, sending to multiple receivers




```
/* ROUTINE_INTERNAL (A,B,C,DEF,G)

/* Load CCR with A

/* Get data from DMA B

/* Load CAR with C

/* Send the data to DEF

/* Load CAR with G

/* Return

/* End
```

APPENDIX C

ROUTINE CODES
and
SOFTWARE INTERFACE

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

NONE ASSEMBLED BY CAL/16 00-00

```
                                1  *     PROGRAM TRANSFER
                                2  *
0000R  D000 00ACR               3  INTERNAL STM     0,RSAVE
0004R  C820 009CR               4           LHI     2,DATA          ENTER DATA PARAMETERS
0008R  C830 00A2R               5           LHI     3,DEVICE        ENTER DEVICE PARAMETERS
000CR  C840 00AAR               6           LHI     4,TARGET        ENTER TARGET PARAMETER
0010R  D812 0000                7           WH      1,0(2)          LOAD CCR WITH A
0014R  D812 0002                8           WH      1,2(2)          OBTAIN DATA THROUGH DMA B
0018R  D812 0004                9           WH      1,4(2)          START DATA TRANSFER AT ADDRESS C
001CR  C860 0000               10           LHI     6,0             LOAD COMPAC REGISTER WITH ZEROES
0020R  41F0 0078R              11           BAL     15,COMPAC       CALL COMPAC
0024R  9816                    12           WHR     1,6             SEND DATA TO DEVICES
0026R  D814 0000               13           WH      1,0(4)          START STORING DATA AT ADDRESS Y
002AR  D100 00ACR              14           LM      0,RSAVE         EXIT THIS ROUTINE
002ER  430F 0000               15           B       0(15)
                               16  *
0032R  D000 00ACR              17  SENDT    STM     0,RSAVE         STORE CURRENT REGISTERS
0036R  C820 009CR              18           LHI     2,DATA          ENTER DATA PARAMETERS
003AR  D812 0000               19           WH      1,0(2)          LOAD CCR WITH A
003ER  D812 0004               20           WH      1,4(2)          OBTAIN DATA THROUGH DEVICE B
0042R  D812 0008               21           WH      1,8(2)          START GETTING DATA AT ADDRESS C
0046R  D100 00ACR              22           LM      0,RSAVE         EXIT TIHIS ROUTINE
004AR  430F 0000               23           B       0(15)
                               24  *
004ER  D000 00ACR              25  SENDR    STM     0,RSAVE         STORE CURRENT REGISTERS
0052R  C820 009CR              26           LHI     2,DATA          ENTER DATA PARAMETERS
0056R  C830 00A2R              27           LHI     3,DEVICE        ENTER DEVICE PARAMETERS
005AR  C840 00AAR              28           LHI     4,TARGET        ENTER TARGET PARAMETER
005ER  D812 0000               29           WH      1,0(2)          LOAD CCR WITH A
0062R  C860 0000               30           LHI     6,0             LOAD COMPAC REGISTER WITH ZEROES
0066R  41F0 0078R              31           BAL     15,COMPAC       CALL COMPAC
006AR  9816                    32           WHR     1,6             SEND DATA TO DEVICES
006CR  D810 0004               33           WH      1,4             START STORING DATA AT ADDRESS Y
0070R  D100 00ACR              34           LM      0,RSAVE         EXIT THIS ROUTINE
0074R  430F 0000               35           B       0(15)
                               36  *
```

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

```
0078R  24C4                37  COMPAC   LIS     12,4              SET COUNTER TO 4
007AR  2460                38           LIS     6,0               SET L EQUAL TO ZERO
007CR  24D1                39           LIS     13,1              SET K EQUAL TO ONE
007ER  24E0                40           LIS     14,0              PUT ZERO IN REGISTER 14
0080R  45E3 0000           41  LOOP     CLH     14,0(3)           SEE IF THE PARAMETER EQUALS ZERO
0084R  4330 008AR          42           BE      NONE              IF PARAMETER EQUALS ZERO, GO FOR
                           43  *                                 NEXT PARAMETER
0088R  0A6D                44           AHR     6,13              OTHERWISE ADD K TO L
008AR  0ADD                45  NONE     AHR     13,13             INCREMENT K TO NEXT BINARY VALUE
008CR  2634                46           AIS     3,4               MOVE REGISTER THREE TO NEXT PARAME
008ER  27C1                47           SIS     12,1              DECREMENT COUNTER
0090R  C5C0 0000           48           CLHI    12,0              IF THERE IS STILL A VALUE IN 12,
                           49  *                                 BRANCH BACK
0094R  4230 0080R          50           BNE     LOOP
0098R  430F 0000           51           B       0(15)             EXIT ROUTINE COMPAC
                           52  *
009CR                      53  DATA     DS      6                 ALLOW FOR THREE VALUES
00A2R                      54  DEVICE   DS      8                 ALLOW FOR 4 DEVICES
```



```
00AAR                      55  TARGET   DS      2                 ONE HALFWORD FOR THE STARTING ADDR
00ACR                      56  RSAVE    DSH     16                SPACE TO STORE REGISTERS.
00CCR                      57           END
```

```
NO ERRORS

CAL/16 00-00

   ABSTOP          0000
   ADC             0002
   COMPAC          0078R
   DATA            009CR
   DEVICE          00A2R
   IMPTOP          00CCR
   INTERNAL        0000R
   LADC            0001
   LOOP            0080R
   NONE            008AR
   RSAVE           00ACR
   SENDR           004ER
   SENDT           0032R
   TARGET          00AAR
END OF TASK    0
```

Software Interface

1. To employ the driver routines, an interface routine between the control computer and each of the driver routines is needed. This enables the software of the control computer to call a routine which will arrange the transfer of data and report back to the control computer either when the task is completed or when an error condition arises.

2. The following calling formats are used to call the interface routines:

a. CALL XMITDATA (XMIT_DEVICE_NO, SIZE, SOURCE_MODE, SOURCE_DEVICE_ID, STARTING_ADDRESS, RETURN_CODE)

Where:

XMIT_DEVICE_NO is the identification code of the transmitter called; either 4 or 5

SIZE is the size of the data to be transferred, expressed in number of bytes. Values from 1 to 64K 16-bit words can be handled.

SOURCE_MODE is a binary condition indicating foward or transmit.

0 = foward (source is one of the receivers)

1 = transmit (source is a DMA)

SOURCE_DEVICE_ID is the unique (8 bit) identification of the device from which the data will come.

If SOURCE_MODE = 0

DEVICE_ID = 0 (transmitter will await data)

If SOURCE_MODE = 1

DEVICE_ID = 1 or 2 (transmitter will actively get data from DMA)

/* These ID numbers can be expanded to 255 if needed.

STARTING_ADDRESS is the address of the first unit of data expressed in whatever address scheme the host machine uses.

If SOURCE_MODE = 0

STARTING_ADDRESS is irrelevant

If SOURCE_MODE = 1
STARTING_ADDRESS must have a value, including 0.
/* Warning: MAX_ADD - DATA_SIZE should be less or equal to the STARTING_ADDRESS to prevent fold over in memory.

RETURN_CODE is passed as zero and returned on completion of the transfer.

If execution of transfer is successful
R = 0
If a time out occurs
R = 1

b. CALL RCVDATA (RCV_DEVICE_NO, SIZE, TARGET_MODE, TARGET_DEVICE_ID, STARTING_ADDRESS, RETURN_CODE)

Where:

RCV_DEVICE_NO is the identification code of the receiver called; either 0 or 1

SIZE is the number of bytes to be transferred. Values from 1 to 64K 16 bit words can be handled.

TARGET_MODE is the binary condition indicating foward or store.

0 = foward (target is a transmitter)
1 = store (target is a DMA)

TARGET_DEVICE_ID is the unique (8 bit) identification of the device to which the data is to be sent.

If TARGET_MODE = 0
TARGET_DEVICE_ID = 32 or 16 (transmitter)
If TARGET_MODE = 1
TARGET_DEVICE_ID = 1 or 2 (DMA)

STARTING_ADDRESS is the first address in memory to start storing the data.

If TARGET_MODE = 0
STARTING_ADDRESS is irrelevant
If TARGET_MODE = 1
STARTING_ADDRESS must have a value, including 0.
/* Warning: MAX_ADD - DATA_SIZE should be less or equal to STARTING_ADDRESS to

prevent fold over in memory.

RETURN_CODE is passed as zero and returned on completion of the transfer.

If execution is successful
R = 0
If a time out occurs
R = 1

c. CALL XRDATA (XR_DEVICE_ID, SIZE, MODE, SOURCE_DEVICE_ID, SOURCE_START_ADDRESS, TARGET_DEVICE_ID, TARGET_START_ADDRESS, RETURN_CODE)

Where:

XR_DEVICE_ID is the identification code of the X/R AFU; either 6 or 7

SIZE is the size of the data to be transferred, expressed in number of bytes. Values from 1 to 64K 16 bit words can be handled.

MODE is a binary condition indicating an internal mode (DMA to DMA) or external mode (DMA to transmitter). Only the internal mode is being considered for this software interface.

SOURCE_DEVICE_ID is the unique (8 bit) identification of the device from which the data will come. This must be a 1 or a 2 (DMA).

SOURCE_START_ADDRESS is the address of the first unit of data in whatever address scheme the host machine uses.

/* Warning: MAX_ADD - DATA_SIZE should be less than or equal to START_ADDRESS to prevent fold over in memory.

TARGET_DEVICE_ID is the unique (8 bit) identification of the device to which the data will go. This should be 1 or 2.

TARGET_START_ADDRESS is the memory address where storage of the data will start.

/* Warning: MAX_ADD - DATA_SIZE should be less than or equal to TARGET_START_ADDRESS to prevent fold over in memory.

RETURN_CODE is passed as zero and returned on completion of the transfer.

If execution is successful

R = 0

If a time out occurs

R = 1

3. Calling convention for AFU drivers

a. The interface routine determines the availability of the desired AFU by inspecting its CCR or status of its return code or the like.

b. Alert the AFU and set its identification code in the output device address register. (R1 in the program code on page C-1).

c. Call the appropriate initialization routine with parameters as indicated:

1) Parameter A - SIZE of the data expressed in 16 bit units.
2) Parameter B - the SOURCE_DEVICE_ID "AND'd" with MODE.
3) Parameter C - the STARTING_ADDRESS for the data source.
4) A parameter indicating the desired destination for the data.
   a) The coded routine (page C-1) allows for multiple target devices which are compacted into one target instruction. This case is not being considered by the software interface at this time.
   b) Parameter X is the DMA to receive the data or a transmitter AFU to foward the data.
5) Parameter Y - the STARTING_ADDRESS for storage in the target DMA.
6) Parameter R - the value to return on completion of the data transfer.

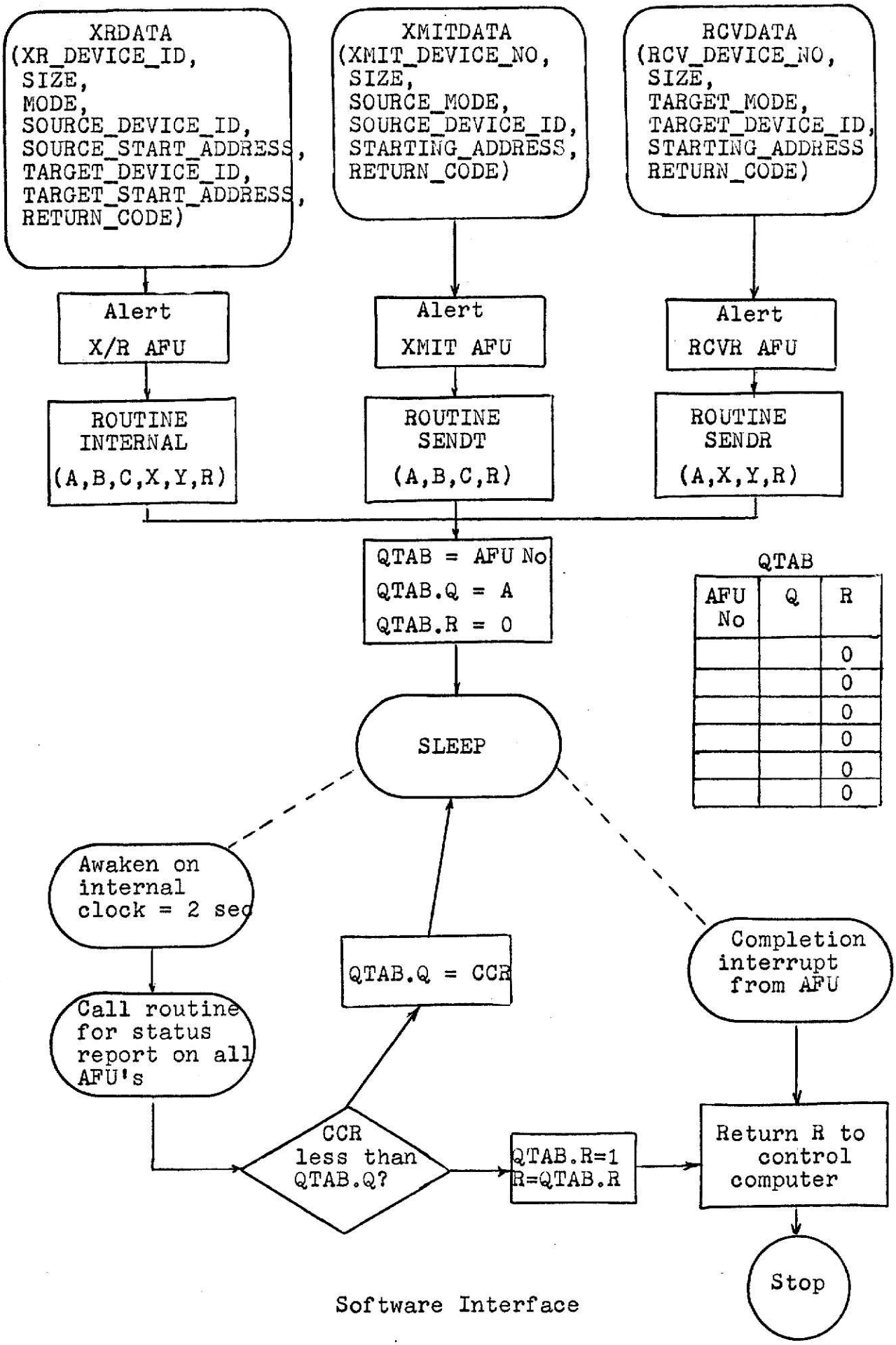
4. Monitoring the transfer routine

a. After initializing the driver routines, the CCR initial value, Parameter A, is put into a reference location in a table which holds these value for all AFU's. The interface routine then goes to a sleep or wait state.

b. The routine is awoken either by an interrupt from the AFU when it completes the task or by the internal clock system.

c. The clock interrupt requires a routine to be periodically run which checks on the progress or status of all AFU's. This check is conducted by requesting the AFU to send the value in its current counter register. If this value is less than that in the reference location in the table, the CCR value replaces the reference value and the routine returns to the wait state. If the CCR value is the same as the reference value (indicating no movement of data), an error code is returned to the control computer and the program stops.

d. On completion of the task, the AFU sends its AFU ID to the routine via an interrupt. On receipt of this interrupt, the routine returns a completion code to the control computer and stops.



Software Interface

APPENDIX D

KSUBUS ORGANIZATION

1. The general capabilities needed by each type device, AFU or DMA, on the KSUBUS is as follows:

	X/R	XMIT	RCVR	DMA
READYBUS	R/W	R/W	R	W
REQUESTBUS	R/W	R/W	R/W	R/W
TOBUS	R/W	R/W	R/W	R/W
FROMBUS	W	W		R

Legend: R = Read. The device must be able to sense the bit condition.
W = Write. The device must be able to set the bit condition.

2. In the typical KSUBUS arrangement shown in figure 4, eight devices are represented. Their hardwired connections with respect to each bus and bit are shown in the following matrices; one for each device. Bits 4 and 5 are not used by any device in this example and are spares.

a. AFU No 7 (X/R):

	0	1	2	3	4	5	6	7
READY BUS	W	R	R	R			R	R
REQUEST BUS	W							
TOBUS	R	W	W	W			R/W	R/W
FROM BUS	W							

Comments: 1) AFU No 7 is the top priority AFU as it sets its request on the most significant bit (0). It also sets bits on the READYBUS and the FROMBUS which are assigned to it.
2) It only needs to read the ready status of DMA's and transmitters. It sends data to

no one else. It sets the TOBUS bits of these devices when it sends data to them.

3) It reads the TOBUS condition of the DMA's to see if the DMA will be used by another device. It also reads its own TOBUS bit to determine when data is coming.

b. AFU No 6 (X/R)

	0	1	2	3	4	5	6	7
READY BUS	R	W	R	R			R	R
REQUEST BUS	R	W						
TOBUS	W	R	W	W			R/W	R/W
FROM BUS		W						

Comments: 1) AFU No 6 is second priority on the REQUEST BUS (bit 1). It sets its assigned bits on the READYBUS, REQUESTBUS, and FROMBUS.

2) It reads the ready condition of potential data receptors (the DMA's and other transmitters) and its own TOBUS bit.

3) It sets the TOBUS bit of a device to which it sends data.

4) It must read the REQUESTBUS bit of AFU No 7 as it cannot run until that device is through with the KSUBUS.

5) It reads the TOBUS bit for a DMA to find out if it is being used by another device.

Continued on next page

c. AFU No 5 (XMITR)

	0	1	2	3	4	5	6	7
READY BUS			W				R	R
REQUEST BUS	R	R	W					
TOBUS			R				R/W	R/W
FROM BUS			W					

Comments: 1) On the KSUBUS, a transmitter actively deals with only a DMA. It need only set its bit on its READYBUS, REQUESTBUS, and FROMBUS.

2) It reads the ready condition of the two DMA's and the TOBUS bits of those devices.

3) It must read the REQUESTBUS bits of the two devices ahead of it.

d. AFU No 4 (XMITR)

	0	1	2	3	4	5	6	7
READY BUS				W			R	R
REQUEST BUS	R	R	R	W				
TOBUS				R			R/W	R/W
FROM BUS				W				

Comments: 1) Like AFU No 5, this transmitter only sends to DMA's over the KSUBUS. It needs only to read the DMA READYBUS bits and the TOBUS bits of the DMA's and itself.

2) It sets its bits (3) on the READYBUS, REQUESTBUS, and FROMBUS and sets the

TOBUS bits of the DMA's.

3) It must read the REQUESTBUS bits of the three devices ahead of its REQUESTBUS bit.

e. DMA No 1

	0	1	2	3	4	5	6	7
READY BUS							W	
REQUEST BUS	R/W	R/W	R/W	R/W				
TOBUS	W	W	W	W			R	
FROM BUS	R	R	R	R				

Comments: 1) DMA's are passive devices, with some exceptions. DMA No 1 sets its READYBUS bit and reads its TOBUS bit.

2) It reads the FROMBUS bits to determine whether to return or store data. If returning data, (FROMBUS bit set) it copies the FROMBUS bit onto the corresponding REQUESTBUS bit for use of the DATABUS on return of the data. It thus has a "floating" read/write action depending on to which transmitting device it is to return data.

3) It sets the TOBUS bit of any device (transmitter) to which it sends data.

4) Once a transmitter requests data from a DMA, the transmitter remains ready so there is no need for the DMA to check the READYBUS.

Continued on next page

f. DMA No 2

	0	1	2	3	4	5	6	7
READY BUS								W
REQUEST BUS	R/W	R/W	R/W	R/W				
TOBUS	W	W	W	W				R
FROM BUS	R	R	R	R				

Comments: This DMA is identical to the first except that its assigned bit is bit 7. It reads and writes to the bits of the same transmitter devices as DMA No 1.

g. AFU No 1 (RCVR)

	0	1	2	3	4	5	6	7
READY BUS	R	R	R	R			R	R
REQUEST BUS	R	R	R	R			W	
TOBUS	W	W	W	W			R/W	R/W
FROM BUS								

Comments: 1) A receiver only puts data onto the KSUBUS. It is never a target so it never needs to set a READYBUS bit nor have a TOBUS bit. It needs only to set a REQUESTBUS bit and read the higher priority bits.

2) It reads the ready bits of all potential receptors (all transmitters and DMA's) and sets the TOBUS bits of these devices.

3) It must read the TOBUS bits of the DMA's to keep track of their availability.

h. AFU No 0 (RCVR)

	0	1	2	3	4	5	6	7
READY BUS	R	R	R	R			R	R
REQUEST BUS	R	R	R	R			R	W
TOBUS	W	W	W	W			R/W	R/W
FROM BUS								

Comment: AFU No 0 is identical to AFU No 1 except that it is one bit lower in priority and must, therefore, read the REQUESTBUS bit of AFU No 1.

3. To summarize the generic forms of a device's use of the control buses, the following "rules of thumb" can be stated:

a. READYBUS

1) An AFU or DMA sets its own bit (except a receiver AFU - it is never sent to, so needs no ready bit).
2) An AFU reads the bits of all eligible target devices.
3) A DMA does not need to read as its target (a transmitter) requested the data and is ready by default.

b. REQUESTBUS

1) An AFU sets its own bit and reads those bits above its own in priority.
2) A DMA sets the bit of the device which requested its data and reads the higher priority bits. The actual bits set and read depend on the AFU in point. Hence, a floating "own" bit for a DMA.

c. TOBUS

1) AFU's and DMA's read their own bits (except a receiver AFU is never sent to so it has no TOBUS bit).
2) AFU's and DMA's set the bits of targets to which data is sent.

d. FROMBUS

1) Bits on the FROMBUS are assigned only to X/R and

transmitter AFU's and are set by them.

2) The bits are read by DMA's to indicate whether to return or store data and, if return, to whom (sets the appropriate bit on the REQUESTBUS).

BIBLIOGRAPHY

Calhoun, Dr. Myron A. Department of Computer Science, Kansas State University, Manhattan, Kansas. Conversations and notes, August through December 1976.

Interdata, Inc. User's Manual. Publication Number 29-261R02. Oceanport, New Jersey, March 1974.

Kansas State University, Department of Computer Science. Progress Report on Functionally Distributed Computer Systems Development: Software and System Structure. Manhattan, Kansas, 20 May 1976.

CONTROL COMPUTER LOCAL DRIVER ROUTINES IN A FUNCTIONALLY DISTRIBUTED DATA BASE MANAGEMENT SYSTEM

by

EUGENE KENNETH GOODELL

B.S., United States Military Academy, 1961

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1977

ABSTRACT

The Functionally Distributed Data Base Management System links computers in one geographic location together into a cluster and then forms a network with remote (distant) clusters, providing a system where each machine in the network operates in a specific computer area and each data base in the system is managed by one specific machine. To control this network, a second, smaller computer (ultimately a micro-computer) is allied with each main or host computer in the system. This control computer receives and issues instructions from and to the host computer or other control computers to arrange the movement of data from the memory of one computer to the memory of any other computer in the network.

This project describes local driver routines which direct the hardwired logic of local data moving mechanisms. Included are detailed descriptions of the actions required by each request and an explanation of the software-hardware relationship.