To the Graduate Council:

I am submitting herewith a thesis written by Wilson Elijah McCreary entitled "A Pulse-Code-Modulated Telemetry Decommutator-Computer Interface." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Eugene C. Huebschmann

Eugene C. Huebschmann, Major Professor

We have read this thesis
and recommend its acceptance:

Roy L Joseph

Fletcher W. Donaldson

Accepted for the Council:

L Evans Peck

Vice Chancellor
Graduate Studies and Research

A PULSE-CODE-MODULATED TELEMETRY DECOMMUTATOR-COMPUTER INTERFACE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Wilson Elijah McCreary

March 1981

ACKNOWLEDGMENTS

The author wishes to express his appreciation to Dr. E. C. Huebschmann for his valuable advice and assistance during the course of this work, and to Mrs. Jeannie Harper for her competent preparation of the manuscript. Appreciation is also expressed to the management of ARO, Inc., for the opportunity to attend The University of Tennessee Space Institute.

The research reported herein was sponsored by Headquarters, Arnold Engineering Development Center, Arnold Air Force Station, Tennessee, within the terms of Contract Number F40600-77-C-003 with ARO, Inc., a Sverdrup Corporation Company. Further reproduction is authorized to satisfy the needs of the United States Government.

ABSTRACT

A hardware interface was developed which connects an existing pulse-code-modulation (PCM) decommutator with computer input/output (I/O) channels. The interface was designed to operate with a number of existing computers with the proper interconnecting cable configuration.

In addition to electronic interface functions, the interface also provides a means for editing incoming telemetry data and a first-in-first-out (FIFO) buffer memory. Editing makes it possible to pass only selected PCM data values from the PCM decommutator output data. Editing in conjunction with FIFO memory allows the processing of data at telemetry data rates higher than otherwise possible. The FIFO memory minimizes data rate constraints imposed by computer processing time overhead and synchronizes telemetry data flow into the computer.

A prototype was fabricated using wire-wrap modules and was checked out on a Modcomp Model MCII computer. The prototype was later connected to a Digital Equipment Corporation Model PDP 11/40 computer. Printed circuit modules were fabricated and installed in the final operational system. The system was put into use in test data acquisition.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

Numerous systems tested in the aerospace environmental test facilities at Arnold Engineering Development Center (AEDC) produce serial pulse-code-modulated (PCM) telemetry output data streams. These serial data are decommutated (i.e., synchronized and converted to parallel) with decommutation equipment and input to a general purpose digital computer for analysis. This investigation describes an interface designed to connect a PCM decommutator and a computer and to provide certain features which enhance system speed and flexibility.

Many systems tested produce PCM data streams at rates sufficiently high that it becomes impossible to process or even retain and store real-time data on digital mass storage media. Two features of the interface described herein minimize problems caused by high data rates. One feature is an edit memory which allows selecting only a portion of the data stream values to pass to the computer. The second feature is a "first-in-first-out" (FIFO) buffer memory which enables minimizing the effects of computer time overhead on PCM data rate capacity.

Figure 1.1, derived from the Interrange Information Group (IRIG) standard [1],[1] illustrates PCM telemetry data formatting. Figure 1.2 illustrates a complete PCM telemetry data link. During

---

[1]Numbers in brackets refer to similarly numbered references in the Bibliography.

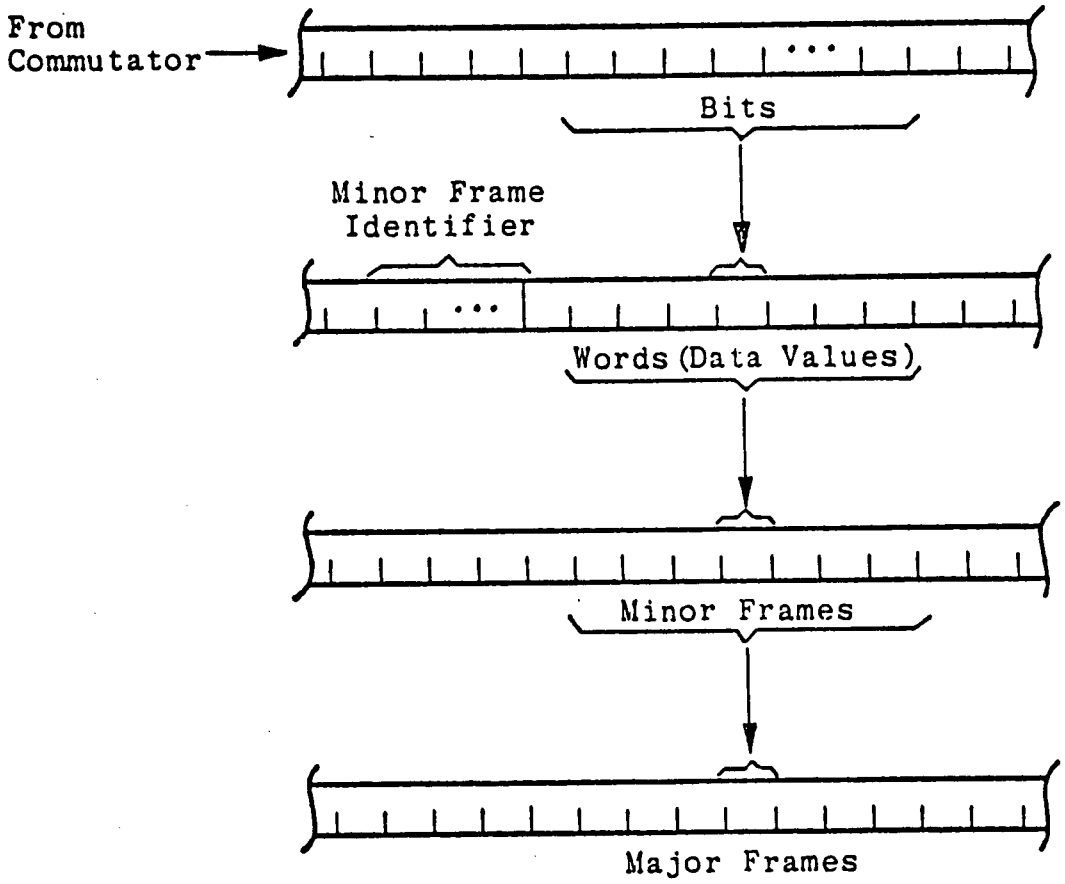


Figure 1.1. Serial telemetry data stream.

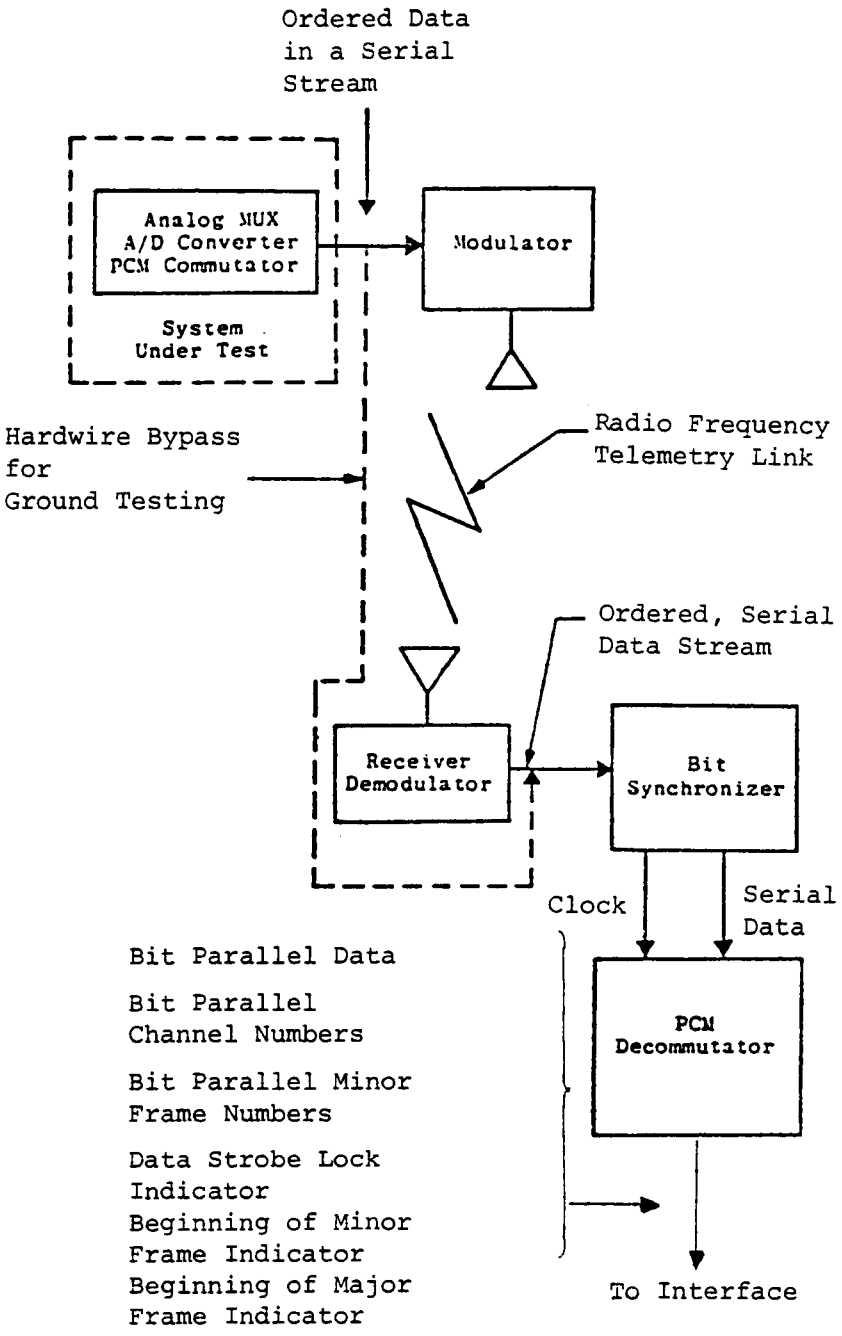


Figure 1.2. Pulse-code-modulated telemetry data link.

most tests at AEDC the radio-frequency system elements are not used and serial data are transferred over a "hard" line. The interface described in this study operates on parallel data, parallel channel number, status, and control information produced by the PCM decommutator. Figure 1.3 illustrates the interface's functional operation. The "continuous PCM data stream" highlighted portions represent data with corresponding channel numbers specified in the edit memory. Buffered, edited data pass to the computer at rates as required by the computer (or computer input port) allowing interruptions in data flow for input processing overhead (block transfer reinitialization, and so forth).

Chapter II of this investigation presents the design methods used. Chapter III describes system operation from the programmer's (user's) point of view to give the reader a more detailed perspective of system requirements. Chapter IV presents the application of the design methods used for one of the system's modules. Finally, Chapter V describes the operation achieved with the current system and possible future extensions.

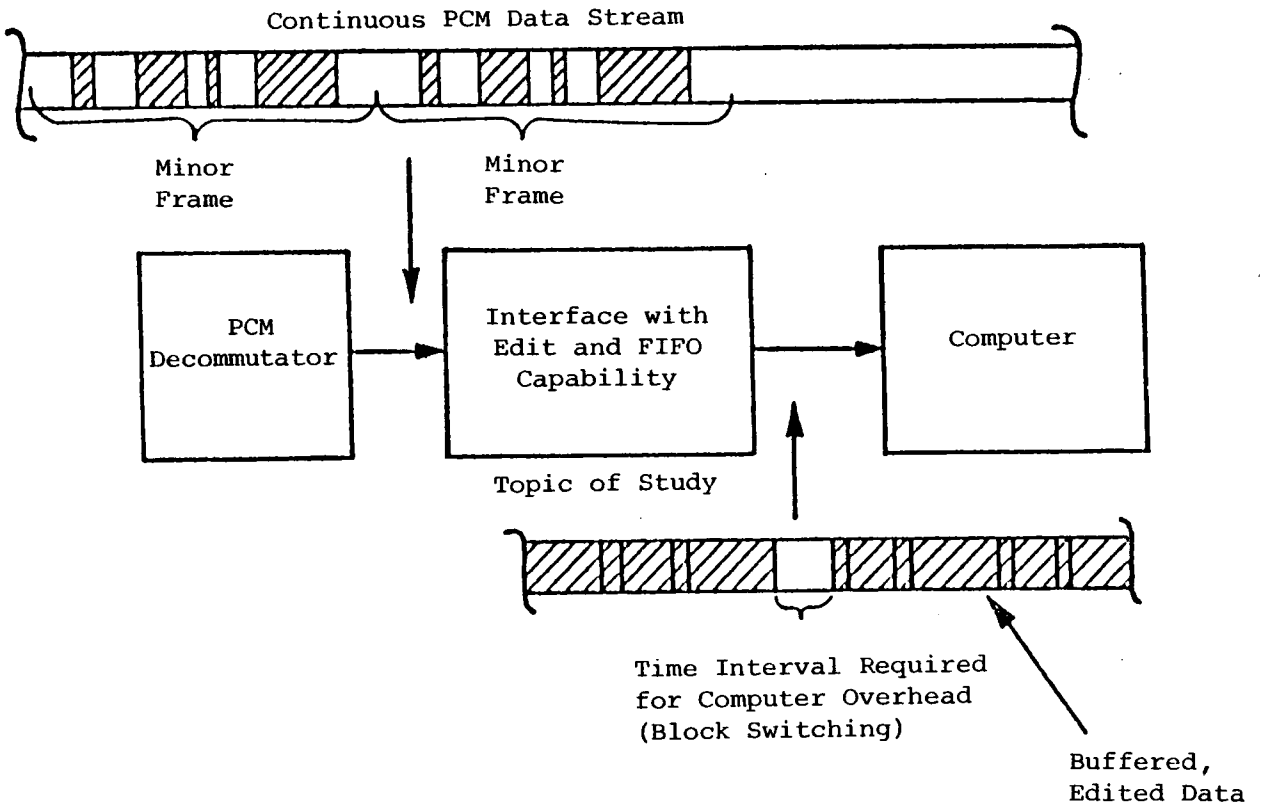


Figure 1.3. Interface functional illustration.

CHAPTER II

DESIGN METHODS

2.1 INTRODUCTION

One consideration generally observed in designing digital systems of any size is that of breaking or decomposing the problem into smaller, more manageable blocks [2]. Each block can then be treated individually. A number of tools are available for the designer in arriving at a suitable hardware configuration.

Among the tools available to the digital designer are "cut-and-try" methods [3] of Boolean equation manipulation, state diagram [4], and state table [2, 4] descriptions of sequential machines [2, 3, 4, 5], Karnaugh map representations [2, 4, 5] of Boolean functions, and procedures for determining minimal Boolean functions from Karnaugh maps. Another type of resource available to the designer is the medium-scale and large-scale integrated circuit package [2]. Functions utilized in the PCM computer interface include the first-in-first-out (FIFO) buffer function [6, 7]; random access memory (RAM) arrays [2, 8]; and multiplexer, demultiplexer gate arrays [2, 9]. All of these were the tools used in designing and fabricating the PCM computer interface.

2.2 DECOMPOSING THE PROBLEM BY FUNCTION AND STRUCTURE

Certain functional characteristics included in the interface requirements imply functional entities that lead to modularity. These

are the "edit" function and the FIFO buffer function. Not specifically called out, but implied in the requirements, is a block to translate computer control information into interface control signals. A second type of constraint that contains implications of modularity is the physical structure containing the system. The physical structure ultimately chosen for the PCM computer interface contains a "motherboard" into which plug individual printed circuit (PC) boards containing the logic circuit elements. The motherboard is itself a PC board with a 100-signal line "bus" on its surface. The bus concept introduces still another constraint on modularity--desired module-to-module signal flow.

The functional and physical constraints described above provide the basis for the first level of system decomposition. Estimating the number of circuit elements required for the functional elements indicated that the FIFO, edit memory, and computer control functions would each fit onto an individual PC board. Decommutator parallel output data (Fig. 1.2) include 12 data and 12 channel number signal lines and five status and control signal lines. Printed circuit board space required for terminating the signal lines is not included on the PC boards processing these signals. In addition, placing the signals on the bus facilitates provisions for future system expansion. As a consequence, an additional PC board was designed to terminate the decommutator signals and impress them on the bus. A clock source and certain other functions were also placed on the board. Figure 2.1 illustrates the interface functional breakdown with the constraints considered thus far.

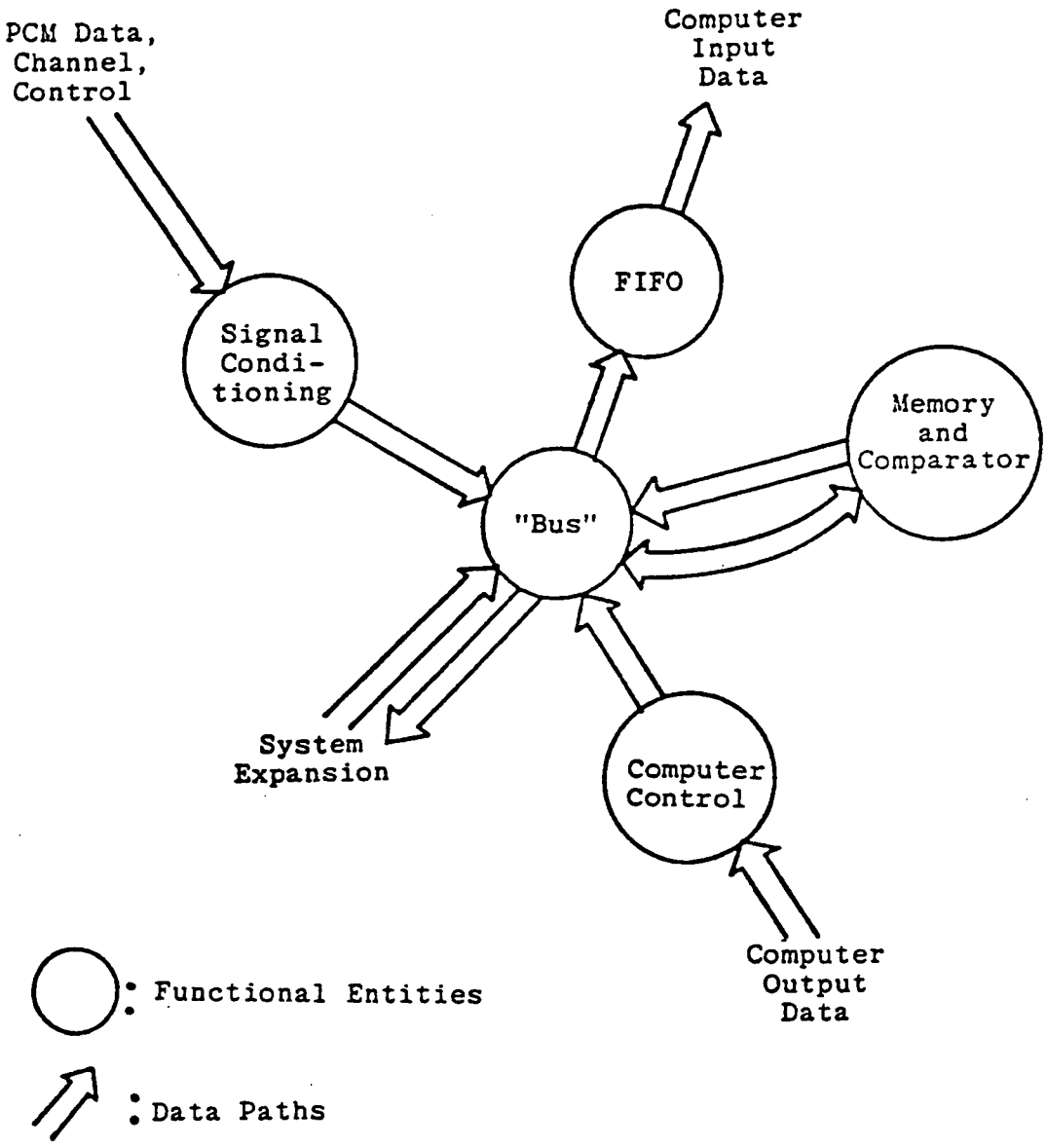


Figure 2.1. Initial system decomposition.

Subsequent functional breakdown was essentially a separation of each board's basic function and its secondary control functions. For example, the FIFO board circuitry consists of:

1. Basic function--FIFO memory proper.
2. Secondary functions--circuitry to synchronize and strobe FIFO input data, circuitry to synchronize and strobe FIFO output data into the computer data channel, and so forth.

CHAPTER III

TELEMETRY DECOMMUTATOR INTERFACE PROGRAMMING CONSIDERATIONS

3.1 INTRODUCTION

This chapter describes the information transfer necessary to control and receive data from the pulse-code-modulated (PCM) interface and includes descriptions of:

1. Control word formats.
2. Control bit functions.
3. Control word sequences.
4. PCM output data sequences.
5. PCM input data format and status bit functions.
6. Input data validation and error recovery.

3.2 DATA FORMATS

Four types of functions must be implemented in receiving PCM data:

1. PCM interface must be initialized.
2. Edit memory must be loaded with desired PCM channel numbers if editing is required.
3. Run mode and run conditions (edit or not edit, starting point in the PCM data structure, etc.) must be set.
4. PCM data must be input through the computer data channel.

3.2.1 Control Word Formats and Control Bit Functions

Two types of words are output to the PCM interface. "Control" words are distinguished by a "1" bit in the most significant bit position (PDP11/40 bit position 15). "Data" words (edit memory data words) are distinguished by a "0" bit in the most significant bit position.

Control words are formatted as follows:

15	14 13	12	11	10	9	8	7	6	5 4 3 2 1 0	Bit Number
1	SEL	LOAD	RUN	SO MF	SO SF	ED IT	MEM CLR	FIFO CLR	X	Control Word

Bit fields are defined in Table 3.1. Sequences and constraints on ordering of control words will be described later.

Edit memory "data" words are formatted as follows:

15	14 13	12	11	10 9 8 7 6 5 4 3 2 1 0	Bit Number
0	SEL	X	TRUN	CHNU (Channel Number)	Data Word

Bit fields are defined in Table 3.2.

3.3 CONTROL WORD SEQUENCES

3.3.1 Initial Sequences

To initially prepare the interface for data transfer the user (computer system) must:

Table 3.1. Definitions of Bit Fields for Control Words

Bit(s)	Mnemonic	Function
13, 14	SEL	Channel select. The current configuration is 0,0. Other configurations may be used in future system expansion.
12	LOAD	A "1" sets up the edit memory to store channel numbers which follow until a channel number with bit 11 = "1" is encountered. The load operation is then terminated.
11	RUN	A "1" places the interface in the run mode.
10	SOMF	A "1" specifies that the interface start passing data with the beginning of a "main frame." (Appendix A describes the main frame.)
9	SOSF	A "1" specifies that the interface start passing data with the beginning of a "sub-frame" (often referred to as a "master frame," see Appendix A); "1" only when bit 11-RUN is a "1."
8	EDIT	A "1" specifies that only data with channel numbers specified in the edit memory are to be passed to the DR11-B.
7	MEMCLR	A "1" causes the edit memory to be cleared to all zeros; "1" only when bits 8 through 12 are "0."
6	FIFOCLR	A "1" causes the "FIFO" memory to be cleared of data.
0 to 5	X	These bits are not used. Any bit configuration is allowed.

Table 3.2. Definitions of Bit Fields for Edit Memory Data Words

Bit(s)	Mnemonic	Function
13, 14	SEL	Channel select. Same as for control words.
12	X	This bit is not used; it can be "0" or "1."
11	TRUN	A "1" specifies the last channel number to be used before going back to the first channel number specified.
0 to 10	CHNU	A PCM channel number of data to be passed.

1. Clear the FIFO memory.
2. Clear the edit memory address counter.[1]
3. Load the edit memory.[1]
4. Again clear the edit memory address counter.[1]
5. Set up the run mode.
6. Input data.

Input data format depends on whether editing is used and the first PCM data value transferred can be: arbitrary (first data available after the run mode is specified) when RUN = 1, SOMF = 0, and SOSF = 0; the first data value of a "minor frame" when SOMF = 1; or the first data value of a "master frame" when SOSF = 1. The terms RUN, SOMF, and SOSF are normally never simultaneously specified as 1's. When more than one is a 1, RUN overrides SOMF and SOSF, and SOMF overrides SOMF.

Running without editing requires the simplest output sequence (Sequence 1) as shown in Fig. 3.1. Editing requires the output sequence (Sequence 2) shown in Fig. 3.2. After one of the above sequences is output, PCM data starts entering the FIFO memory. The computer must begin data input before the FIFO memory is filled. The time available between setting the run mode and starting data input can be found from the equation:

$$TA = \frac{1}{PWR}\left(\frac{CF}{CP}\right) * 128 \ , \qquad (3.1)$$

where,

---

[1]Not required if editing is not required.

Sequence 1

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Clear the FIFO Memory															
Word 1	1	0	0	0	0	0	0	0	0	1	X					
	Set Run Mode															
Word 2 (Last Word)	1	0	0	0	RUN	SO MF	SO SF	0	0	0	0	X				

Figure 3.1. Running without editing (Sequence 1).

Note: There must be an interval of at least 0.5 microsecond between the "clear FIFO" word and the subsequent "run mode" word. The interval can be achieved by generating a delay between word transfers during a "test and transfer" type operation or by outputting dummy words (all zeros) during DMA type operation.

Figure 3.2. Running with editing (Sequence 2).

Note: There must be an interval of at least 0.5 microsecond between the "clear" word and the "run mode" word. The interval can be achieved by generating a delay between word transfers during a "test and transfer" type operation or by outputting dummy words (all zeros) during DMA type operation.

Sequence 2

Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Clear the Edit Memory Address Counter															
Word 1	1	0	0	0	0	0	0	0	1	X						
	Prepare to Load Edit Memory															
Word 2	1	0	0	1	0	0	0	0	0	0	X					
	Followed by															
Word 3	0	0	0	X	0	First PCM Channel Number										
Word 4	0	0	0	X	0	Second PCM Channel Number										
Word N-2	0	0	0	X	1	Last PCM Channel Number										
	Clear the Edit Memory Address Counter and FIFO Memory															
Word N-1	1	0	0	0	0	0	0	0	1	1	X					
	Set Run Mode (Include Edit)															
Word N (Last Word)	1	0	0	0	RUN	SO MF	SO SF	1	0	0	X					

Figure 3.2.

TA = time available,

PWR = PCM word rate ,

CP = channels passed ,

CF = channels per frame .

It is assumed here that if editing is used, the edit memory values (PCM channel numbers) increase monotonically and no values are repeated. If editing is not used, CP/CF = 1, and

128 = number of FIFO memory locations .

The two sequences above are sufficient to run most ordinary tests. Control sequence manipulation can be used to achieve unusual operating conditions. PCM channel numbers stored in the edit memory are subject to certain constraints. For most operations, channel number values must increase monotonically. Extended editing functions can be implemented by judicious edit memory channel number sequence selection.

3.3.2 Noninitial Sequences

It is convenient to characterize two types of PCM data acquisition--continuous and noncontinuous. All valid incoming PCM data enter the FIFO memory (after editing, if editing is used). The number of empty FIFO memory locations is a critical consideration for continuous data acquisition and, in some cases, for noncontinuous acquisition.

Continuous data acquisition. PCM data input is by block transfer in most cases. A new block transfer must be initiated after

each current block transfer has completed. The time available for initiating a new transfer can be found from the equation:

$$TA = \frac{N}{PWR}\left(\frac{CF}{CP}\right) - \frac{N}{IWR} , \qquad (3.2)$$

where,

N = number of words per block ,

PWR, CP, and CF are defined as for Eq. (3.1) ,

and

IWR = computer input data rate .

The term TA is not continuous over the entire range of the independent variables of Eq. (3.2); TA can be no greater than the value given in Eq. (3.1) and is invalid when negative. An overflow condition exists when Eq. (3.2) gives a negative value. Later paragraphs describe overflow in more detail.

Noncontinuous data acquisition. During continuous acquisition, no control output is required after the initial output (Sequence 1 or Sequence 2), unless an overflow or other error occurs, until data transfer is terminated. Noncontinuous input can be accomplished with no additional control output if:

$$TBB < TA ,$$

where,

TBB = time between input data blocks ,

and

TA = the value obtained from Eq. (3.1) .

If,

$$TBB > TA ,$$

then a control sequence must be output for each new input data block. Edit memory values must be output (after the initial output) only if channel numbers must be deleted, added, or replaced.

3.4 COMPUTER INPUT DATA FORMAT AND STATUS BIT FUNCTIONS

The PCM decommutator and PCM data source determine two input data format parameters. These parameters are data value justification and data bit significance. The two parameters cannot be determined until data source characteristics are established.

Computer input data are formatted as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit Number
PCM Data												MFL	SFL	OV FL	FF	Data Word

Bit fields are defined in Table 3.3.

3.5 DATA VALIDATION AND ERROR RECOVERY

The status bits described above provide the means of verifying error-free data. When a "sub frame" and/or "main frame" out-of-lock condition occurs (SFL and/or MFL = "0") or the overflow condition occurs (OVFL = "1") the probability is high that the frame flag (FF) will be displaced from its expected position in the data block. Thus,

Table 3.3. Definitions of Bit Fields for Pulse-Code-Modulated Telemetry Input Data Words

Bit(s)	Mnemonic	Function
4 to 15	PCM DATA	PCM input data appears in this field. Bit 15 is the leftmost data bit and data extend contiguously to the right three to 12 bits.
3	MFL	A "1" implies PCM minor frame is in lock (no error).
2	SFL	A "1" implies PCM master frame is in lock (no error).
1	OVFL	A "1" implies an overflow condition (error (error condition).
0	FF	"Frame flag"; "1" implies the corresponding data value is the first of a frame.

Note: MFL and SFL must be "1" and OVFL must be "0" for valid data. Other states exist only when an error has occurred.

checking the frame flag position provides an effective means of checking for most errors (excluding hardware failures). Where data validity is very critical, an exhaustive scan of all data words and testing all status flags should be undertaken.

When an error is detected, return to proper operation should include outputting a new control sequence in most cases. MFL and SFL will return to normal automatically when data source and/or PCM decommutator operation return to normal. The PCM interface forces OVFL to "1" when overflow occurs. OVFL remains "1" after overflow until a subsequent output control sequence clears it. A control word with bits 15 and 6 (FIFO CLR) clears OVFL to zero (see Fig. 3.1, page 15, Sequence 1, Word 1; and Fig. 3.2, page 16, Sequence 2, Word N-1).

CHAPTER IV

DESIGN APPLICATION TO FIRST-IN-FIRST-OUT MODULE

The PCM decommutator-computer interface first-in-first-out (FIFO) module consists of FIFO memory circuitry and peripheral circuitry which synchronize FIFO memory input; synchronize FIFO memory output; and clear the FIFO memory of residual data prior to starting a data transfer operation. The Fairchild Data Book [7] contains an FIFO memory circuit structured nine bits wide by 192 words long. The Fairchild circuit was restructured to 16 bits wide by 128 words long for the FIFO module.

Figure 4.1 depicts the FIFO memory in block diagram form. Signal CIR ("composite input ready"), when high, indicates that the memory is ready for data input. When low, CIR indicates that the memory is full or that data are being input. Signal SFFI ("strobe FIFO in") is placed high to input data and subsequently placed low when CIR responds by going low, indicating the data are accepted. Signal FCLR ("FIFO clear"), which is low-true in the actual circuitry, is used to clear the FIFO IC's of all data and initialize their internal control circuitry. Signal COR ("composite output ready"), when high, indicates that data are present and can be output from the FIFO memory. When low, COR indicates that data are not present or are being output. Signal SFFO ("strobe FIFO out") is placed high to cause new data to shift out, and is subsequently placed low when COR responds by going low. New data are present and stable when COR returns to high.

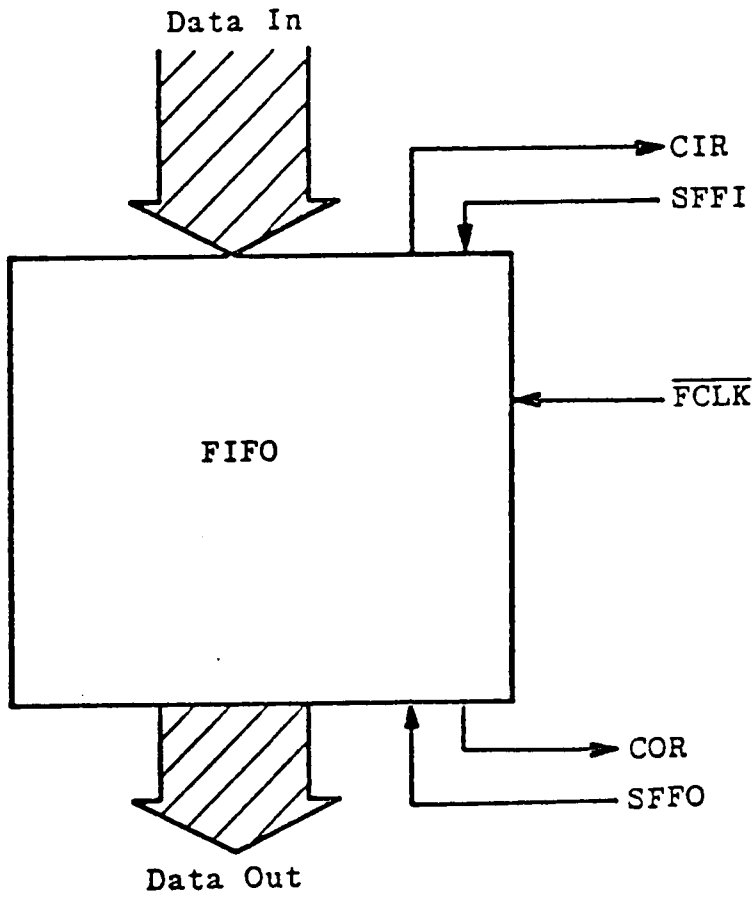


Figure 4.1. Composite first-in-first-out memory circuit.

Conditions external to the FIFO memory also determine when data can be transferred. Data are transferred only when the interface is in a run mode. The run mode is indicated on the interface bus by the signals RUN0B, RUN1B, RUN2B, and RUN3B. Only RUN0B is used in the current interface configuration. However, to ease system expansion when it becomes desirable, the FIFO module should operate selectively with any of the four signals.

Similarly, data can be transferred only when one of the interface bus signals EQU0B, EQU1B, EQU2B, or EQU3B is true. The signal EQU--, when true, indicates that editing was not specified when the current run mode was specified or that editing was specified and the current PCM decommutator channel number corresponds to the current edit memory output. Again, the module should operate selectively with any of the EQU-- signals.

Bus signals BD13B and BD14B are computer control word unit selection bits. Bus signal ADRGB indicates that a computer control word including the unit selection bits is being sent to the interface. Bus signal FCLRB indicates that the current computer control word includes a specification that the FIFO memory be cleared. Now all signals used in the unit selection process have been described. The first of two design examples follows.

Figure 4.2 illustrates a functional block including the input signals processed by, and the output signals produced by, the selection circuitry. Input signals are to the left and output signals are to the right. A description of the design of this function block follows.

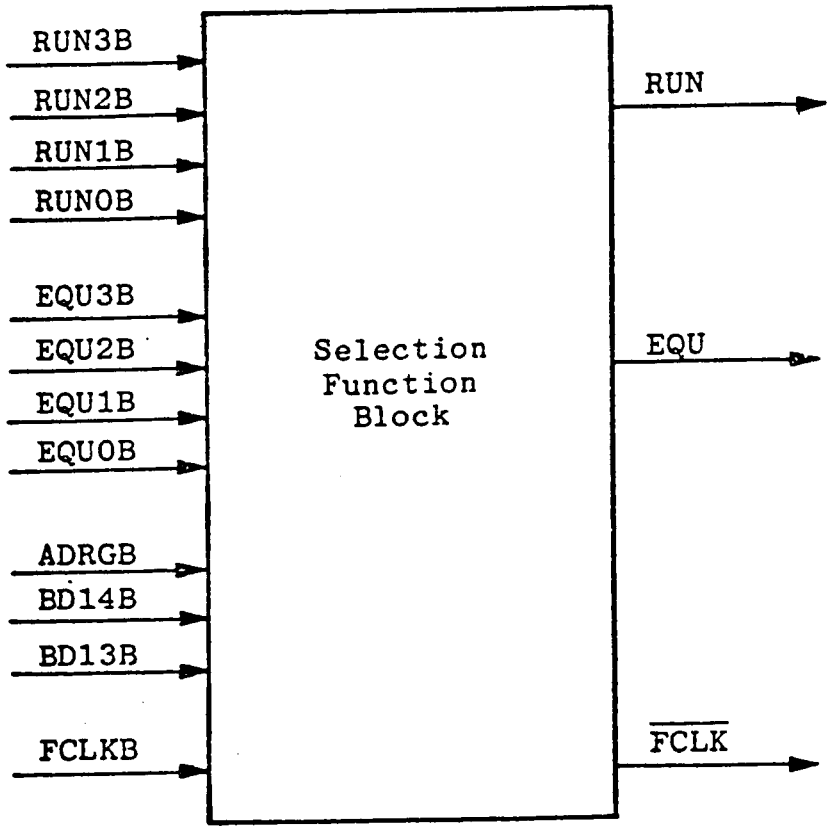


Figure 4.2. Functional block designed with "cut-and-try" methods.

Signals BD14B and BD13B unit selection configurations are defined as:

$$\text{Unit 0} = \overline{\text{BD14B}} \cdot \overline{\text{BD13B}}$$

$$\text{Unit 1} = \overline{\text{BD14B}} \cdot \text{BD13B}$$

$$\text{Unit 2} = \text{BD14B} \cdot \overline{\text{BD13B}}$$

$$\text{Unit 3} = \text{BD14B} \cdot \text{BD13B} \ .$$

In order to select one of the configurations, one may provide a circuit on the module which generates four states and compare the module state with the incoming signal state. The switch circuit of Fig. 4.3a generates the four states:

$$\text{State 0} = \overline{A} \cdot \overline{B}$$

$$\text{State 1} = \overline{A} \cdot B$$

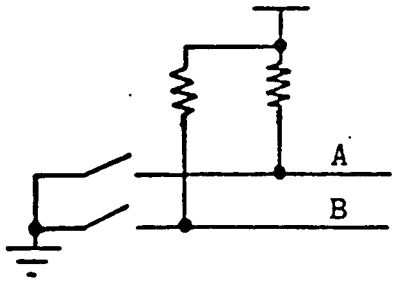
$$\text{State 2} = A \cdot \overline{B}$$

$$\text{State 3} = A \cdot B \ .$$

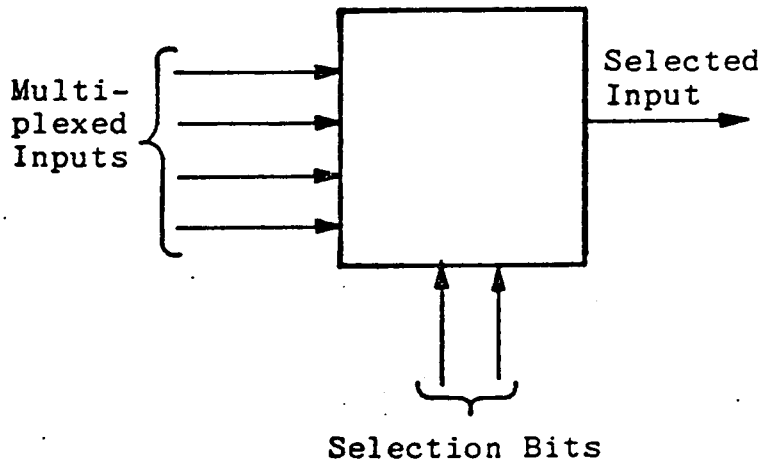
Selection can be expressed algebraically as,

$$\text{Unit Select} = \overline{\text{BD14B}} \cdot \overline{\text{BD13B}} \cdot \overline{A} \cdot \overline{B} + \overline{\text{BD14B}} \cdot \text{BD13B} \cdot \overline{A} \cdot B + \text{BD14B} \cdot \overline{\text{BD1B}} \cdot A \cdot \overline{B} + \text{BD14B} \cdot \text{BD138} \cdot \cdot A \cdot B \ . \quad (4.1)$$

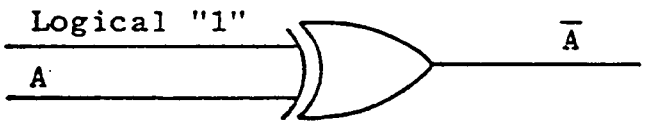
This equation could be implemented directly. However, by applying theorems and postulates of Boolean algebra [2, 3, 4, 5, 6], an equation form requiring less hardware may be determined. Gathering terms,



a. Unit select switches



b. Four-to-one multiplexer



c. Exclusive or gate configured as an inverter

Figure 4.3. Logic function elements.

$$\text{Unit Select} = \overline{BD14B} \cdot \overline{B} \cdot (\overline{BD13B} \cdot \overline{A} + BD13B \cdot A)$$

$$+ BD14B \cdot B \cdot (\overline{BD13B} \cdot \overline{A} + BD1B \cdot A)$$

$$= (\overline{BD14B} \cdot \overline{B} + BD14B \cdot B) \cdot$$

$$\cdot (\overline{BD1B} \cdot \overline{A} + BD13B \cdot A) . \qquad (4.2)$$

Observing that,

$$\overline{X} \cdot \overline{Y} + X \cdot Y = X' \cdot \overline{Y} + \overline{X'} \cdot Y ,$$

where,

$$X' = \overline{X} ,$$

Eq. (4.2) may be expressed as,

$$\text{Unit Select} = (BD14B \cdot \overline{B'} + \overline{BD14B} \cdot B') \cdot$$

$$\cdot (BD13B \cdot \overline{A'} + \overline{BD13B} \cdot A') , \qquad (4.3)$$

where,

$$A' = \overline{A} , \; B' = \overline{B} .$$

The form,

$$X \cdot \overline{Y} + \overline{X} \cdot Y ,$$

is the "exclusive or" [3, 4, 5] written as,

$$X \oplus Y ,$$

and Eq. (4.3) can be written as,

$$\text{Unit Select} = (BD14B \oplus B') \cdot (BD13B \oplus A') . \qquad (4.4)$$

The "exclusive or" function is implemented in standard transistor-transistor-logic (TTL) [9]. The logic element is shown in Fig. 4.3c.

Signal ADRGB must be considered to insure that unit selection occurs only when a control word is being sent. The selection process can also be inhibited when a given unit is running to minimize the possibility of interference by users of other units. With these considerations, Eq. (4.4) becomes,

$$\text{Unit Select} = (\text{BD14B} \oplus \text{B}') \cdot (\text{BD13B} \oplus \text{A}') \cdot \text{ARDGB} \cdot \overline{\text{RUN}} \ . \qquad (4.5)$$

Selecting the appropriate RUN--- signal and EQU--- signal can be approached as a multiplexing problem. Expressed algebraically the functions are,

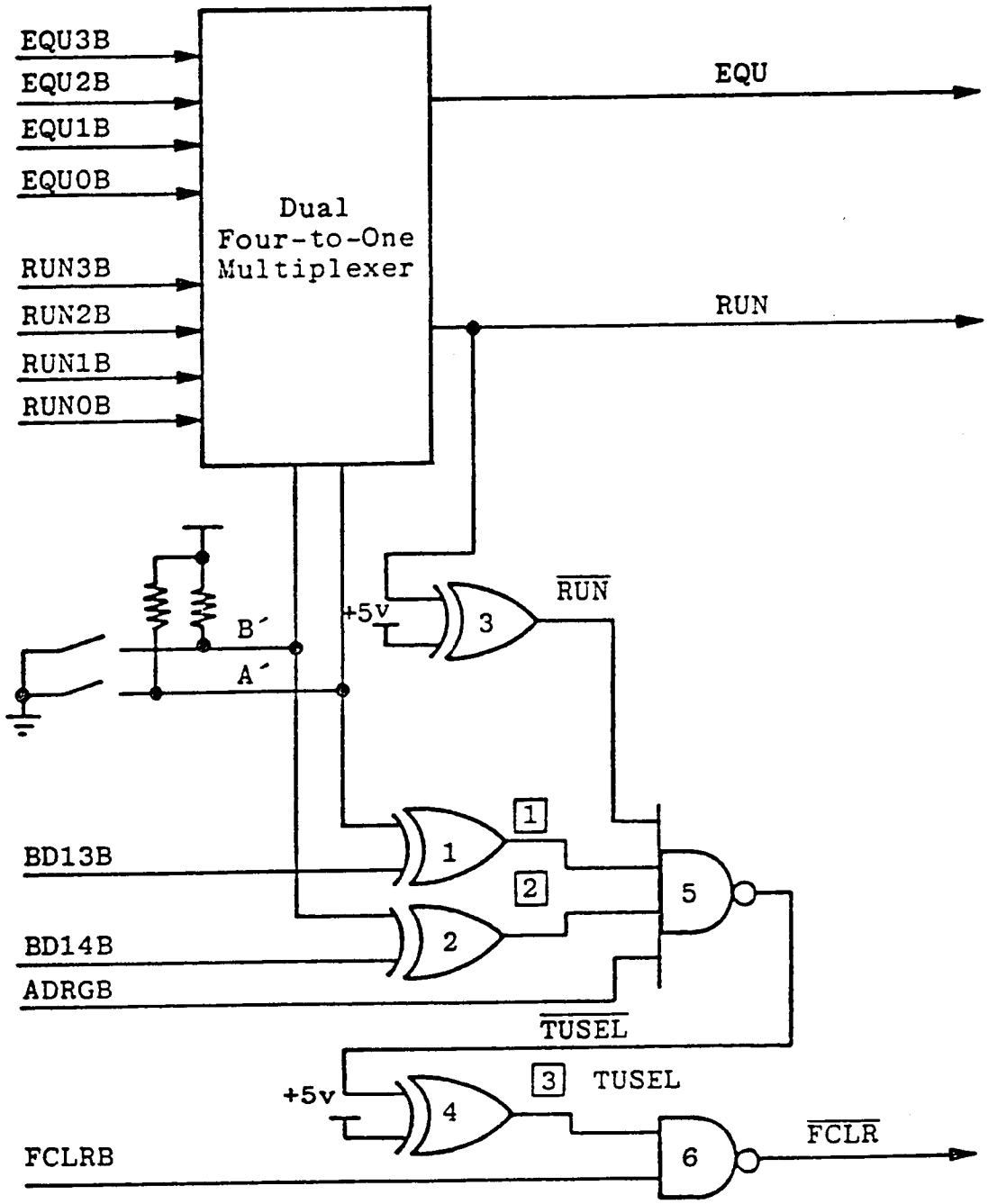
$$\text{RUN} = \text{RUN0B} \cdot \text{A}' \cdot \text{B}' + \text{RUN1B} \cdot \text{A}' \cdot \overline{\text{B}'} + \text{RUN2B} \cdot \overline{\text{A}'} \cdot \text{B}' + \text{Run} \cdot \overline{\text{A}'} \cdot \overline{\text{B}'} \qquad (4.6)$$

and,

$$\text{EQU} = \text{EQU0B} \cdot \text{A}' \cdot \text{B}' + \text{EQU1B} \cdot \text{A}' \cdot \overline{\text{B}'} + \text{EQU2B} \cdot \overline{\text{A}'} \cdot \overline{\text{B}'} + \text{EQU3B} \cdot \overline{\text{A}'} \cdot \text{B}' \ , \qquad (4.7)$$

where A' and B' are the same as A' and B' in Eq. (4.5). Equations (4.6) and (4.7) are equations for "four-to-one" multiplexers [7, 9]. Figure 4.3b illustrates the multiplexer. A standard TTL device exists for implementing dual four-to-one multiplexers [9].

Logic circuitry can now be implemented from Eqs. (4.5), (4.6), and (4.7). Figure 4.4 illustrates the functional block using physical circuit element representations. The "exclusive or" gates labeled "3" and "4" are configured as the logical inverter of Fig. 4.3c. The



[1] BD13B + A´

[2] BD14B + B´

[3] (BD13B + A´) · (BD14D + B´) · ADRGB · $\overline{\text{RUN}}$

Figure 4.4. Unit selection functional block circuit implementation.

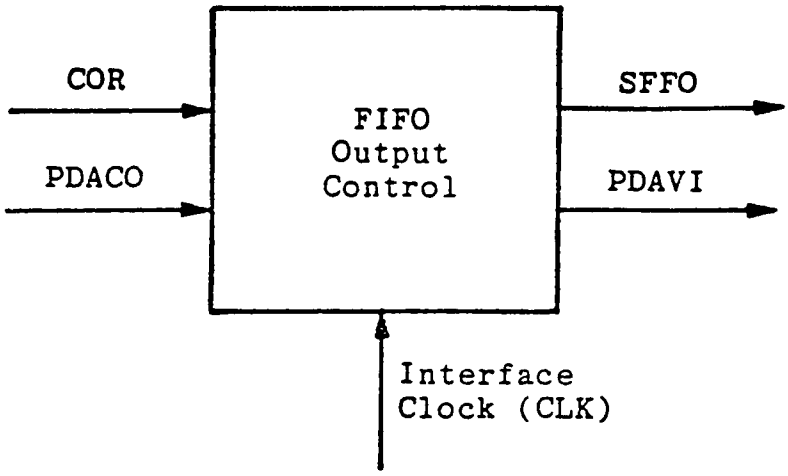
"exclusive or" function was not required and use in this manner utilizes circuitry which would otherwise go unused. (The standard TTL "exclusive or" IC package includes four gates.)

Nand gate 5 and "exclusive or" gate 4 form a four-input "and" function [2, 3, 5]. This was done because the four-input "and" circuit package was not a local stock item and is not used as the four-input nand. Nand gate 6 provides a convenient means of generating the low-true "clear" signal required by the FIFO memory circuits.

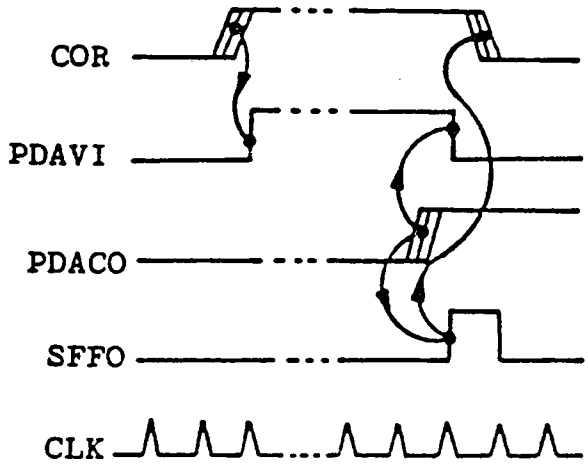
The unit selection function block consists of combinational logic [2, 3, 4, 5]. The second design example is the FIFO output synchronization circuitry. This functional block is a synchronous sequential circuit [2, 3, 5]. Figure 4.5a is a block diagram which illustrates the functional block. Figure 4.5b illustrates desired circuit timing.

Three signals not described previously are shown in Fig. 4.5. Signal PDAVI, "data available in," is a computer input/output channel input signal which indicates that valid input data are available. Signal PDACO, "data accepted out," is a computer input/output channel output signal which indicates that the current input data have been accepted. The interface clock (CLK) is a 4.6 MHz clock used for the synchronous circuit's memory elements.

Figure 4.6 is a state diagram [4] which characterizes desired circuit operation. The vertices correspond to the circuit memory's state and the directed arcs correspond to state transitions. Signals listed adjacent to the vertices are output signals required to be "true" during that state. Output signals not listed are required to



a. FIFO output control circuit block



b. FIFO output timing

Figure 4.5. First-in-first-out output control circuit and timing.

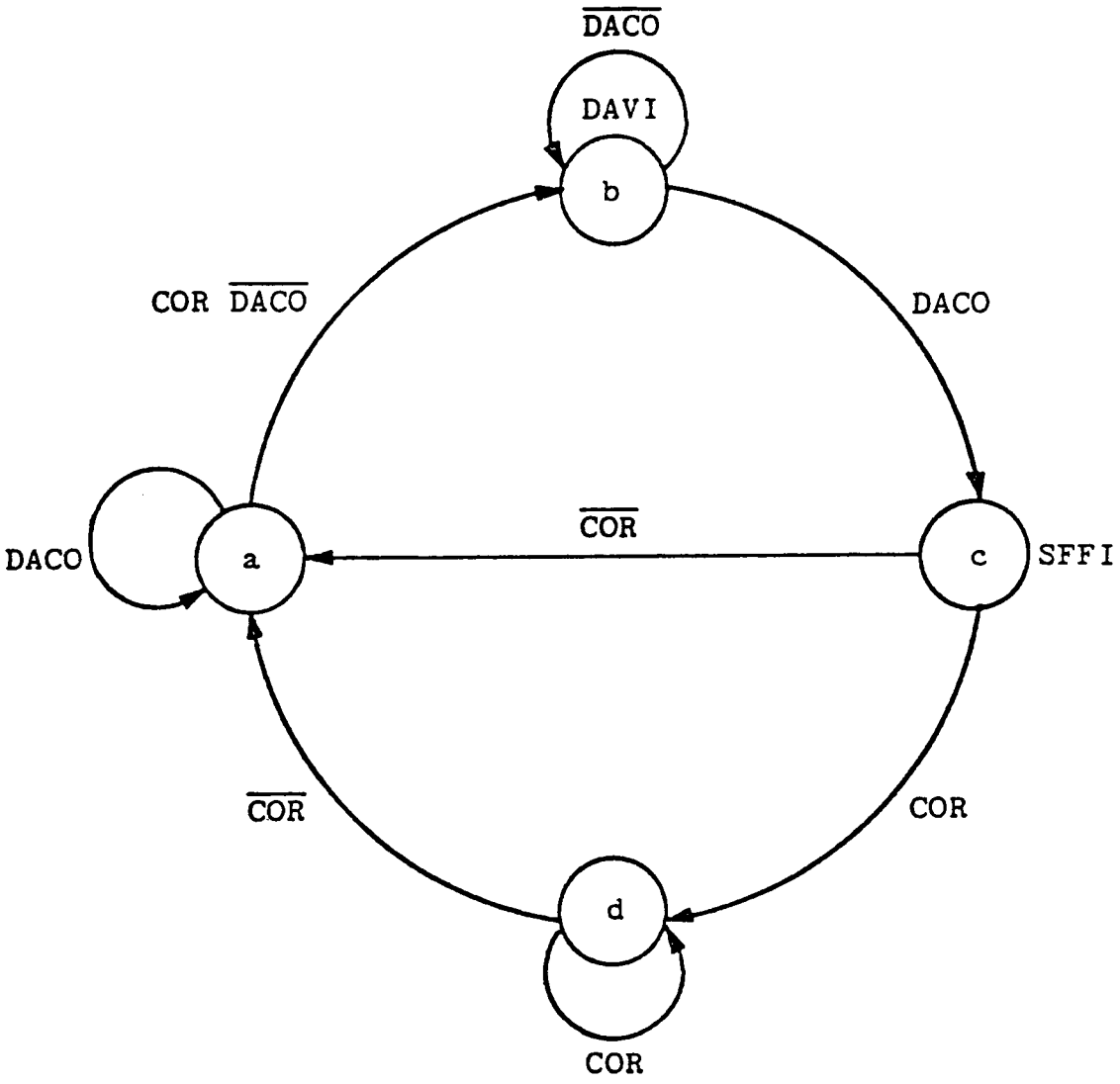


Figure 4.6. First-in-first-out output control circuit state diagram.

be "false." Signals or expressions listed adjacent to directed arcs are input conditions which, when "true," cause the indicated state transition.

From the timing diagram, or state diagram, or both, one can generate a state table [2, 3, 4, 5]. Table 4.1 is a state table for the output control circuit.

The symbol "0" occurring in the fourth column of Table 4.1 indicates a "don't care" state. The "don't care" state appears because of the combination of present input (input at time = n) and present state that will never occur unless a hardware failure occurs.

The number of distinct states in the state table determines the number of memory elements required to implement the circuit. Kohavi [4] provides an equation which yields the required number of memory elements, k:

$$k = [\log_2 n],$$

where,

$$n = \text{the required number of memory elements},$$

and "[g] is the smallest integer larger than or equal to g" for the output control circuit n = 4 and k = 2.

Once the number of memory elements is known, state assignment [2, 3, 4] can be undertaken. That is, memory element states can be assigned to each distinct state found in the state table. The way the memory element states are assigned can have considerable effect on the resulting implementation's complexity and resulting cost. Peatman [2] gives three "essentially different" state assignments for

Table 4.1. First-In-First-Out Output Control Circuit State Diagram

Time = n		Time = n	Time = n+1
DACO	COR	Present State	Next State
0	0	a	a
0	0	b	0
0	0	c	a
0	0	d	a
0	1	a	b
0	1	b	b
0	1	c	d
0	1	d	d
1	0	a	a
1	0	b	0
1	0	c	a
1	0	d	a
0	1	a	a
0	1	b	c
0	1	c	d
0	1	d	d

the case where two memory elements are required. These are shown in Table 4.2. The three state assignments were all considered in the output control circuit.

With a state assignment, one can derive a transition table by replacing each state table state with its corresponding assigned memory state. Choosing a memory element type (the "J-K" flip-flop was used almost exclusively in the telemetry decommutator interface) one can also derive a memory element excitation table [4]. Figure 4.7 and Table 4.3 illustrate the J-K flip-flop block diagram and excitation requirements, respectively. Table 4.4 illustrates a combined transition, excitation, and output table for the output control circuit.

The problem has now been reduced to a combinational circuit problem. The excitation functions and output functions can be generated from Table 4.4. Table 4.4 is derived from state assignment Case 2, Table 4.2, which yields the simplest circuit implementation. From Table 4.4, Karnaugh maps were generated. Figure 4.8 illustrates the maps for state assignment Case 2. From the maps, minimal logic equations were derived and the set of equations yielding the simplest (least costly) implementation was determined. Table 4.5 contains logic equations from all three state assignment cases, illustrating the difference in complexity (and cost) resulting from different state assignments. Finally, Fig. 4.9 illustrates the output control circuit implementation resulting from state assignment Case 2.

Table 4.2. "Essentially Different" State Assignments for Two Memory Elements

Case	Circuit State	Memory Element State HS0	HS1
1	a	0	0
	b	1	1
	c	0	1
	d	1	0
2	a	0	0
	b	0	1
	c	1	1
	d	1	0
3	a	0	0
	b	0	1
	c	1	0
	d	1	1

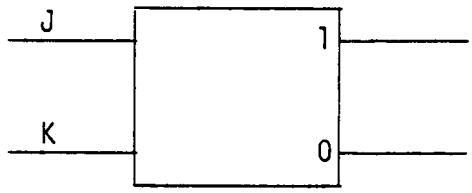


Figure 4.7. J-K flip-flop block diagram.

Table 4.3. J-K Flip-Flop Excitation Requirements

Circuit Change		Required Input	
From	To	J	K
0	0	0	0
0	1	1	0
1	0	0	1
1	1	0	0

Table 4.4. Composite First-In-First-Out Table for State Assignment Case 2

Circuit Inputs (time=n)		Present State (time=n)		Next State (time=n+1)		Memory Element Excitation (time=n)				Circuit Outputs (time=n)	
DACO	COR	HSO	HS1	HSO	HS1	JHSO	KHSO	JHS1	KHS1	DAVO	SFFO
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	1	0	1
0	0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0
0	1	0	1	0	1	0	0	0	0	1	0
0	1	1	1	1	0	0	0	0	1	0	1
0	1	1	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	1	0	1	0	1
1	0	1	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	1	0	0	0	0	1	0	1
1	1	1	0	1	0	0	0	0	0	0	0

HSO HS1 \ DACO COR	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	0	0	0
10	0	0	0	0

a. JHSO excitation map

HSO HS1 \ DACO COR	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	1

b. KHSO excitation map

HSO HS1 \ DACO COR	00	01	11	10
00	0	1	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

c. JHS1 excitation map

HSO HS1 \ DACO COR	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

d. KHS1 excitation map

HSO HS1 \ DACO COR	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	0	0	0
10	0	0	0	0

e. Output DAVI map

HSO HS1 \ DACO COR	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

f. Output SFFO map

Figure 4.8. Maps for state assignment Case 2.

Table 4.5. First-In-First-Out Output Control Logic Equations

Case	
1	$JHS0 = \overline{DAC0} \cdot COR + COR \cdot HS1$
	$KHS0 = \overline{COR} + DAC0 \cdot HS1$
	$JHS1 = \overline{DAC0} \cdot COR \cdot \overline{HS1}$
	$KHS1 = DAC0 + HS0$
	$DAVI = HS0 \cdot HS1$
	$SFF0 = HS0 \cdot FFB$
2	$JHS0 = DAC0 \cdot HS1$
	$KHS0 = \overline{COR}$
	$JHS1 = \overline{DAC0} \cdot COR \cdot \overline{HS0}$
	$KHS1 = HS0$
	$DAVI = HS0 \cdot HS1$
	$SFF0 = HS0 \cdot HS1$
3	$JHS0 = DAC0 \cdot HS1$
	$KHS0 = \overline{COR}$
	$JHS1 = \overline{DAC0} \cdot COR + HS0 \cdot COR$
	$KHS1 = \overline{COR} + DAC0 \cdot \overline{HS0}$
	$DAVI = \overline{HS0} \cdot HS1$
	$SFF0 = HS0 \cdot \overline{HS1}$

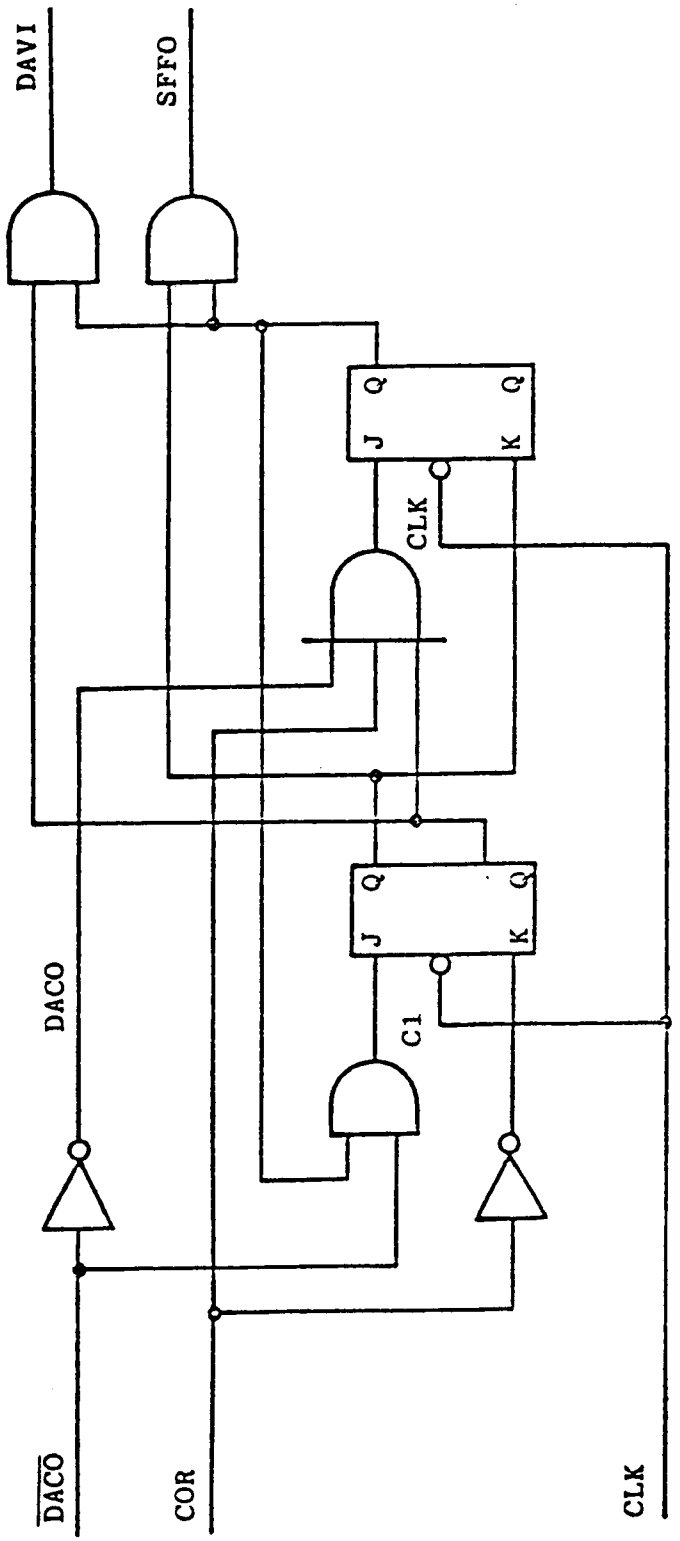


Figure 4.9. First-in-first-out output control circuit implementation.

CHAPTER V

CONCLUSIONS

A pulse-code-modulated (PCM) computer interface prototype was designed and fabricated as planned. Initial checkout was accomplished using a Modular Computer Corporation Model MCII computer. Input/output (I/O) data channels were simulated using the MCII's digital I/O ports. Subsequently, the prototype was connected to a Digital Equipment Corporation (DEC) Model PDP11/40 computer through a DEC Model DR11-B-type direct-memory-access (DMA) controller. Print circuit modules were fabricated and substituted for the prototype modules. The interface is being used on a daily basis along with the other system components in checkout for an upcoming test.

Appendix A is a detailed system operation description. Appendix B contains detailed system logic diagrams and cable drawings. Appendix C contains system bus signal names and descriptors.

The interface provides an effective means of editing, buffering, and passing PCM-decommutator data into a computer. Expansion capability provides several attractive potential capabilities. These include: multiple paths to multiple computers, further reducing computer speed constraints on maximum PCM data rates; and the capability to send different "groupings" of data to different computers, providing appropriate processing functions for each data grouping.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. IRIG Standard 106-77. Telemetry Group, Range Commanders Council. White Sands Missile Range, New Mexico: Secretariat, Range Commanders Council, 1977.

2. Peatman, John B. The Design of Digital Systems. New York: McGraw-Hill Book Company, Inc., 1972.

3. Phister, Montgomery. Logical Design of Digital Computers. New York: John Wiley and Sons, Inc., 1958.

4. Kohavi, Zvi. Switching and Finite Automata Theory. New York: McGraw-Hill Book Company, Inc., 1970.

5. Maley, Gerald A., and John Earle. The Logic Design of Transistor Digital Computers. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1963.

6. Knuth, Donald E. The Art of Computer Programming. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1968.

7. MOS/CCD Data Book. Mountain View California: Fairchild Camera and Instrument Corporation, 1975.

8. Component Data Catalog. Santa Clara, California: Intel Corporation, 1978.

9. TTL Data Book. Santa Clara, California: National Semiconductor Corporation, 1976.

10. Instruction Manual, Models 4003, 4003A, 4004, and 4004A, PCM System. Danbury, Connecticut: Data Control Systems, Inc., 1973.

11. DR11-B Assembly and Installation Manual. Marlborough, Massachusetts: Digital Equipment Corporation, 1975.

12. TTL Applications Handbook. Mountain View, California: Fairchild Semiconductor Corporation, 1973.

APPENDICES

APPENDIX A

PULSE-CODE-MODULATED TELEMETRY DECOMMUTATOR-COMPUTER INTERFACE OPERATIONAL DESCRIPTION

A.1 INTRODUCTION

This appendix relies heavily on logic equations. The logic equations follow conventional rules for Boolean expressions. The symbols (=), (•), (+), and ($\oplus$) represent "equate," logical "and," logical "or," and logical "exclusive or," respectively. The barred term ($\overline{\quad\quad}$) represents the logical inverse of the unbarred term. Certain additional conventions are included as follows:

1. Terms to the left of (=) that set or reset memory elements are prefaced by a lower case "s" or "r," respectively.
2. Clock signals to memory elements are underlined.

All equations are written for positive voltage logic.

Some telemetry equipment manufacturers use terminology for telemetry data formats different from that of the Interrange Information Group (IRIG) telemetry standard [1]. In particular, Data Control Systems, Inc. (DCS), manufacturer of the decommutator being interfaced, uses different terminology to refer to minor frame and major frame. Data Control Systems uses the term main frame for minor frame, and the term sub-frame for major frame. The DCS terminology and signal mnemonics derived from the DCS terminology are used in this chapter and in the logic diagrams (Appendix B).

A.2 GENERAL DESCRIPTION

The pulse-code-modulated (PCM) interface provides three functions for PCM data acquisition:

1. Electrical/logical interface between the PCM decommutator and computer data channel.
2. Edit function to pass only data from selected channels within a PCM main-frame.
3. Data buffering to reduce data rate limitations imposed by computer time overhead during block switching operations.

Figure A.1 illustrates the interface's functional concepts. PCM data and channel numbers enter the interface through the signal paths on the left. When editing is not required channel numbers are ignored, the data switch is left open, and data flow through the first-in-first-out (FIFO) memory to the computer. When editing is in effect, data are passed only when there is a corresponding channel number stored in the edit memory.

Figure A.2 illustrates a concept of the interface's physical structure. The hardware is structured physically and functionally on a modular concept. The structure allows future expansion by adding one control PC board and adding two additional PC boards (EDIT MEMORY and FIFO) per computer input data stream.

A.3 CONTROL MODULE

Figure A.3 illustrates the control module. Appendix B, pages 75 and 76, contains the control module logic diagram.

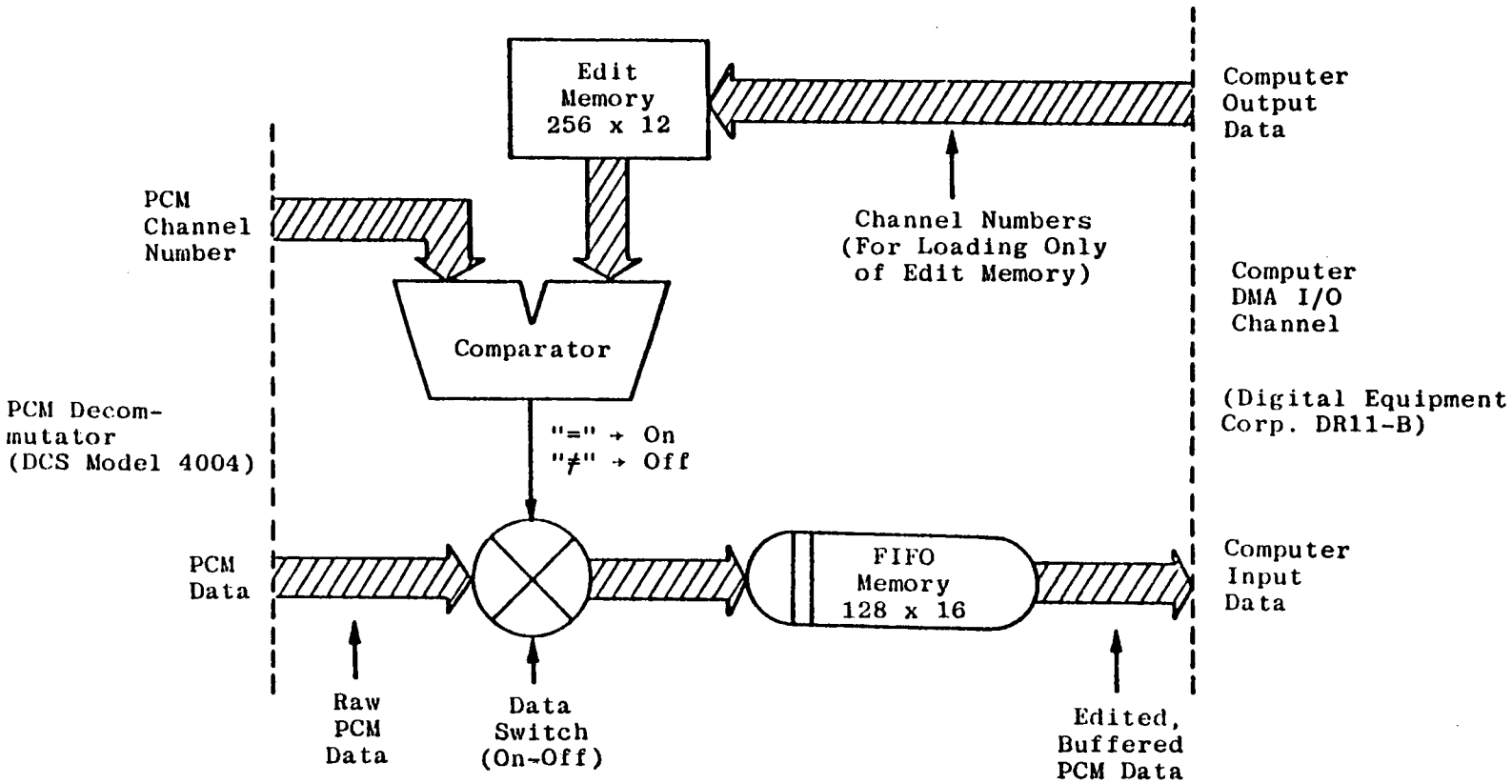


Figure A.1. Decommutator interface functional concept.

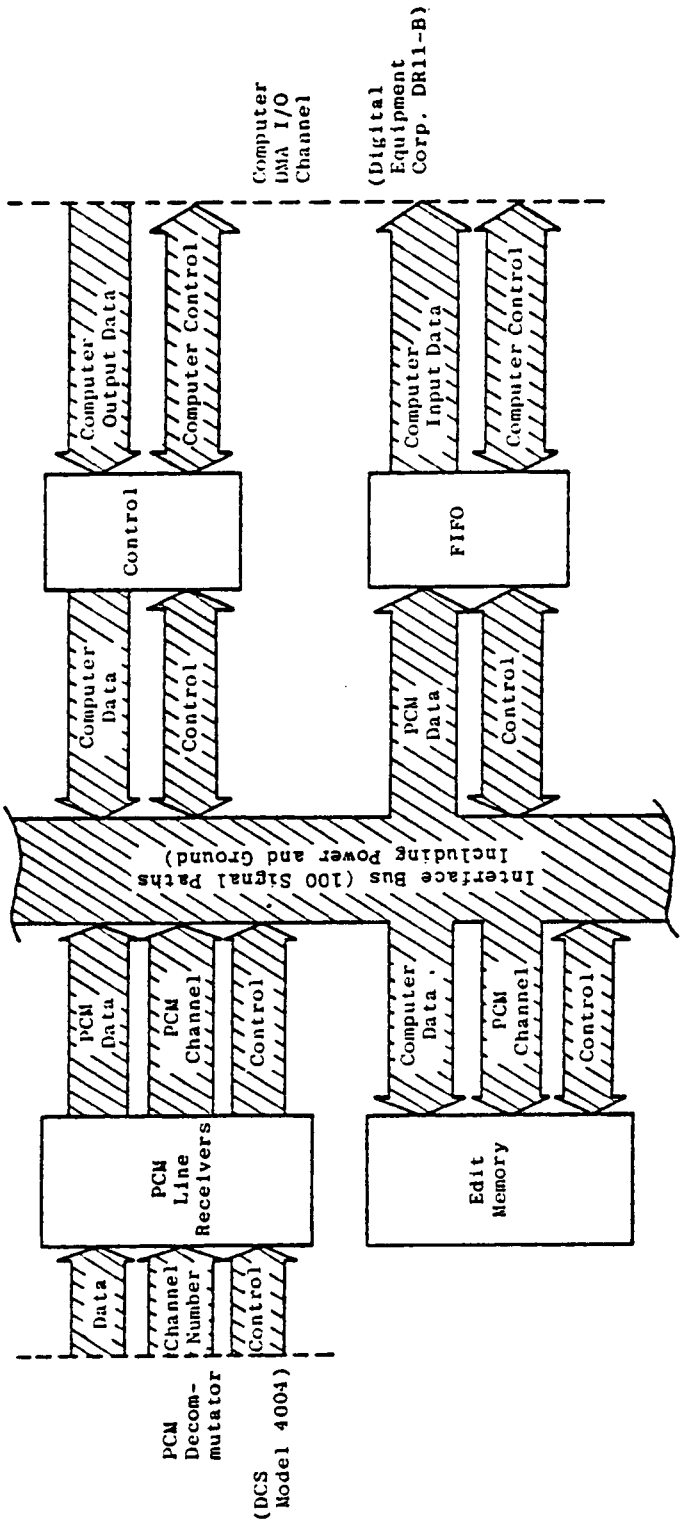


Figure A.2. Decommutator interface physical concept.

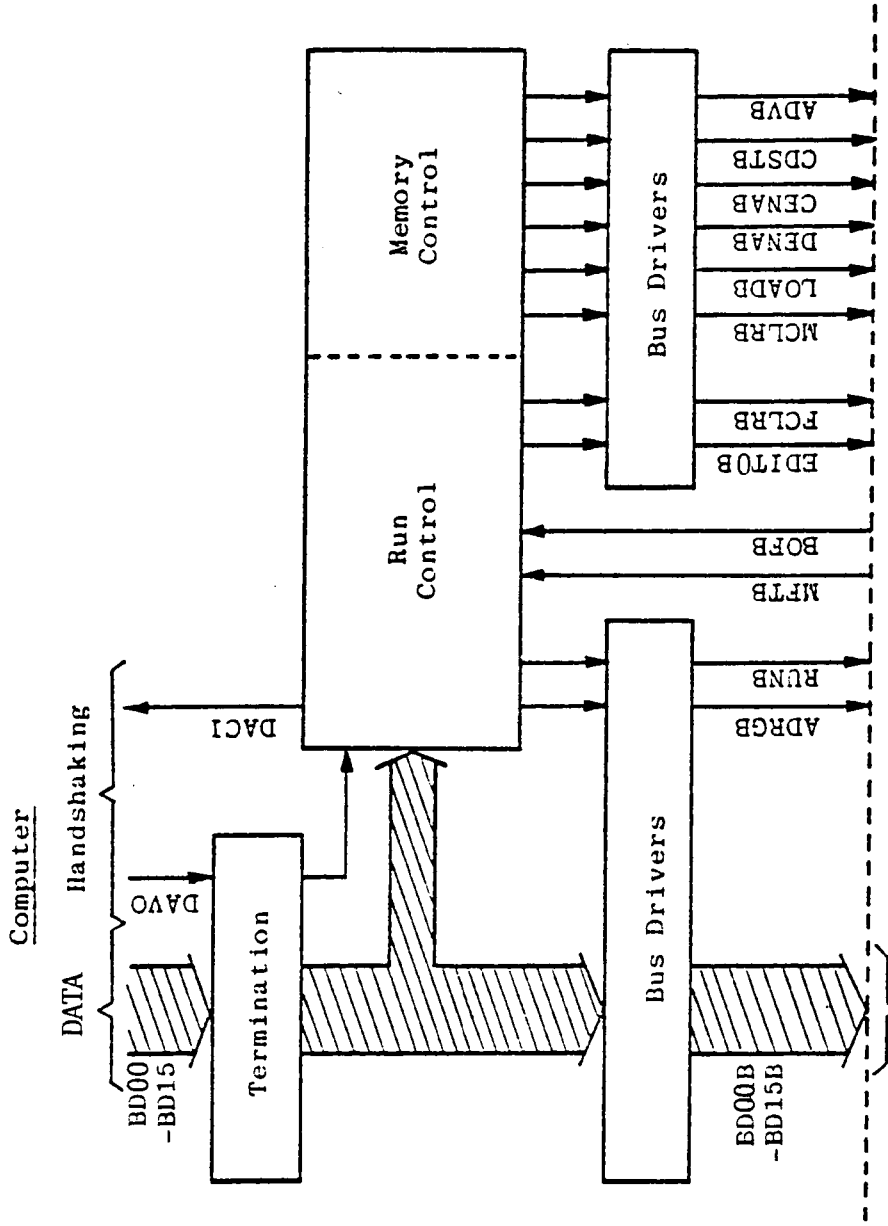


Figure A.3. Control module functional block diagram.

Computer data enter the module through dual inline package (DIP) sockets 30I, 31I, and 32I (Appendix B, page 75). The data signals and handshaking signal, DA OUT, are terminated electrically with components in DIP sockets 18I through 23I and 33I through 35I (Appendix B, page 75). The data signals pass through inverters 9I through 13I and are placed on the interface bus through driver IC's 1I through 3I. Appropriate data signals and DA OUT (DAVO) are taken at the inverter outputs and impressed on run and control logic (Appendix B, page 76).

Run mode conditions are set into IC 5I (Appendix B, page 76),

$$sRUN = BD11 \cdot GATE \cdot \underline{CLK} , \tag{A.1}$$

$$rRUN = \overline{BD11} \cdot GATE \cdot \underline{CLK} , \tag{A.2}$$

$$sSOS = BD10 \cdot GATE \cdot \underline{CLK} , \tag{A.3}$$

$$rSOS = \overline{BD10} \cdot GATE \cdot \underline{CLK} , \tag{A.4}$$

$$sSOM = BD09 \cdot GATE \cdot \underline{CLK} , \tag{A.5}$$

$$rSOM = \overline{BD09} \cdot GATE \cdot \underline{CLK} , \tag{A.6}$$

$$sEDITO = BD08 \cdot GATE \cdot \underline{CLK} , \tag{A.7}$$

$$rEDITO = \overline{BD08} \cdot GATE \cdot \underline{CLK} , \tag{A.8}$$

where,

$$GATE = \overline{BD13} \cdot \overline{BD14} \cdot BD15 \cdot DSTB , \tag{A.9}$$

and DSTB (25I6) (Appendix B, page 76) is essentially a strobe derived from the computer data strobe, DAVO, described further below. The signal RUNO (14I12) (Appendix B, page 76) establishes the run

condition. The run condition occurs when the RUN signal (14I12) (Appendix B, page 76) is true,

$$RUNO = RUN + \ldots ; \tag{A.10}$$

the "start at beginning of frame" signal, SOF (5I12) (Appendix B, page 76), is true and the "beginning of frame" signal, BOF (29I2) (Appendix B, page 76), has occurred,

$$sSOFF = SOS \cdot BOF \cdot \underline{CLK}' , \tag{A.11}$$

$$RUNO = SOFF + \ldots , \tag{A.12}$$

or the "start at beginning of sub-frame" signal, SOS (5I11) (Appendix B, page 76), is true and the "sub-frame tick" signal, SFT (9I12) (Appendix B, page 76), has occurred,

$$sSOMF = SOM \quad SFT \quad \underline{CLK}' , \tag{A.13}$$

$$RUNO = SOMF + \ldots , \tag{A.14}$$

IC 1I (Appendix B, page 75) places the edit signal, EDITOB, and the run signal, RUNOB, on the interface bus.

The bus signals, MCLRB AND FCLRB (6I11 and 13, respectively) (Appendix B, page 76), provide first-in-first-out buffer (Section A.5) and memory address counter (Section A.6) initialization. MCLRB clears the edit memory address counter and is generated by a control word with bit BD08 true,

$$MCLRB = ADRG \cdot BD08 , \tag{A.15}$$

where,

$$ADRG = BD15 \cdot DSTB \ .$$

FCLRB clears data from the first-in-first-out buffer and is generated by a control word with bit BD07 set,

$$FCLRB = ADRG \cdot BD07 \ . \qquad (A.16)$$

Flip-flops HS0, HS1, DACX, and DACY are used in handshaking during computer data output. When computer data are available, DAV0 (13I12 (Appendix B, page 75) and 26I1 (Appendix B, page 76)) goes true and HS0 is set,

$$sHS0 = DAV0 \cdot \underline{CLK} \ . \qquad (A.17)$$

Subsequently, if the data is a control word, run mode conditions can be established (Eqs. (A.1) through (A.8)) or the load control flip-flop, LOAD, can be set or reset,

$$sLOAD = BD12 \cdot ADRG \cdot \underline{CLK} \ , \qquad (A.18)$$

$$rLOAD = (\overline{BD12} \cdot ADRG + \ldots) \cdot \underline{CLK} \ , \qquad (A.19)$$

and flip-flop DACX is set,

$$sDACX = HS0 \cdot \overline{HS1} \cdot \overline{LOAD} \cdot CLK \ .$$

If the data is not a control word (i.e., is an edit memory data word) and flip-flop LOAD had been set previously, an edit memory cycle is initiated by the signal LDI (17I3) (Appendix B, page 76),

$$\overline{LDI} = LOAD \cdot LSTB \ , \qquad (A.20)$$

$$SERI = LDI + \ldots \ . \qquad (A.21)$$

IC 7I and gates 17I8 and 17I6 (Appendix B, page 76) are configured as a four-bit, self-initializing (to state 0000), feed-back shift register. The shift register outputs are combined, producing edit memory control signals,

$$\text{DENAB} = \text{SR0} + \text{SR2} \ , \tag{A.22}$$

$$\text{CENAB} = \text{SR1} \ , \tag{A.23}$$

$$\text{CDSTB} = \text{SR2} \cdot \text{SR3} \ , \tag{A.24}$$

$$\text{ADVB} = \text{SR2} \cdot \text{SR3} \ . \tag{A.25}$$

When an edit memory cycle is initiated (Eqs. (A.20) and (A.21),

$$\text{SERI} = \text{LOAD} \cdot \text{LSTB} + \ldots \ , \tag{A.26}$$

the shift register takes on the sequence of states as shown in Table A.1.

Signals DENAB, CENAB, and CDSTB provide edit memory data and strobe functions. ADVB advances the edit memory address counter preparing for the next data value (PCM channel number) input. ADVB also sets flip-flop, DACY.

Flip-flops DACX and DACY signify that computer data have been accepted. Signal DACI (28I8) (Appendix B, page 76),

$$\begin{aligned} \text{DACI} &= \overline{\overline{\text{DACX}} \cdot \overline{\text{DACY}}} \\ &= \text{DACX} + \text{DACY} \ , \end{aligned} \tag{A.27}$$

is sent to the computer to indicate data acceptance. The computer responds by making DACO false. Subsequently, DACX or DACY is reset

Table A.1. Sequence of States for Shift Register

SERI	SR0	SR1	SR2	SR3	DENAB	CENAB	CDSTB	ADVB
1	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0
1	1	1	0	0	1	1	0	0
0	1	1	1	0	1	1	1	0
0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	0	0

(they are never simultaneously true),

$$rDACX = \overline{DAVO}\ dc\ , \tag{A.28}$$

$$rDACY = \overline{DAVO}\ dc\ , \tag{A.29}$$

HSO is reset,

$$rHSO = \overline{DAVO} \cdot \underline{CLK}\ , \tag{A.30}$$

HS1 is reset,

$$rHS1 = \overline{HSO} \cdot \underline{CLK}\ , \tag{A.31}$$

and the control module is ready for the next computer input.

A.4 TELEMETRY DECOMMUTATOR LINE RECEIVER MODULE

The line receiver module electrically terminates PCM data and channel number signal lines; generates the interface system clock; conditions certain PCM control signals; and places clock, PCM data, PCM channel number, and control signals on the interface bus. Figure A.4 illustrates the module. Appendix B, pages 77 and 78, contains the line receiver module logic diagram.

PCM data, channel number, and control signals enter the module through DIP sockets 21I, 23I, 25I, 27I, and 29I (Appendix B, pages 77 and 78). The signals are terminated electrically with components in DIP sockets 22I, 24I, 26I, 28I, 30I, and 11I through 20I (Appendix B, pages 77 and 78). Data and channel number signals pass through inverters 6I, 7I, 9I, and 10I and onto the interface bus through bus drivers 1I, 2I, 4I, and 6I (Appendix B, pages 77 and 78). Control

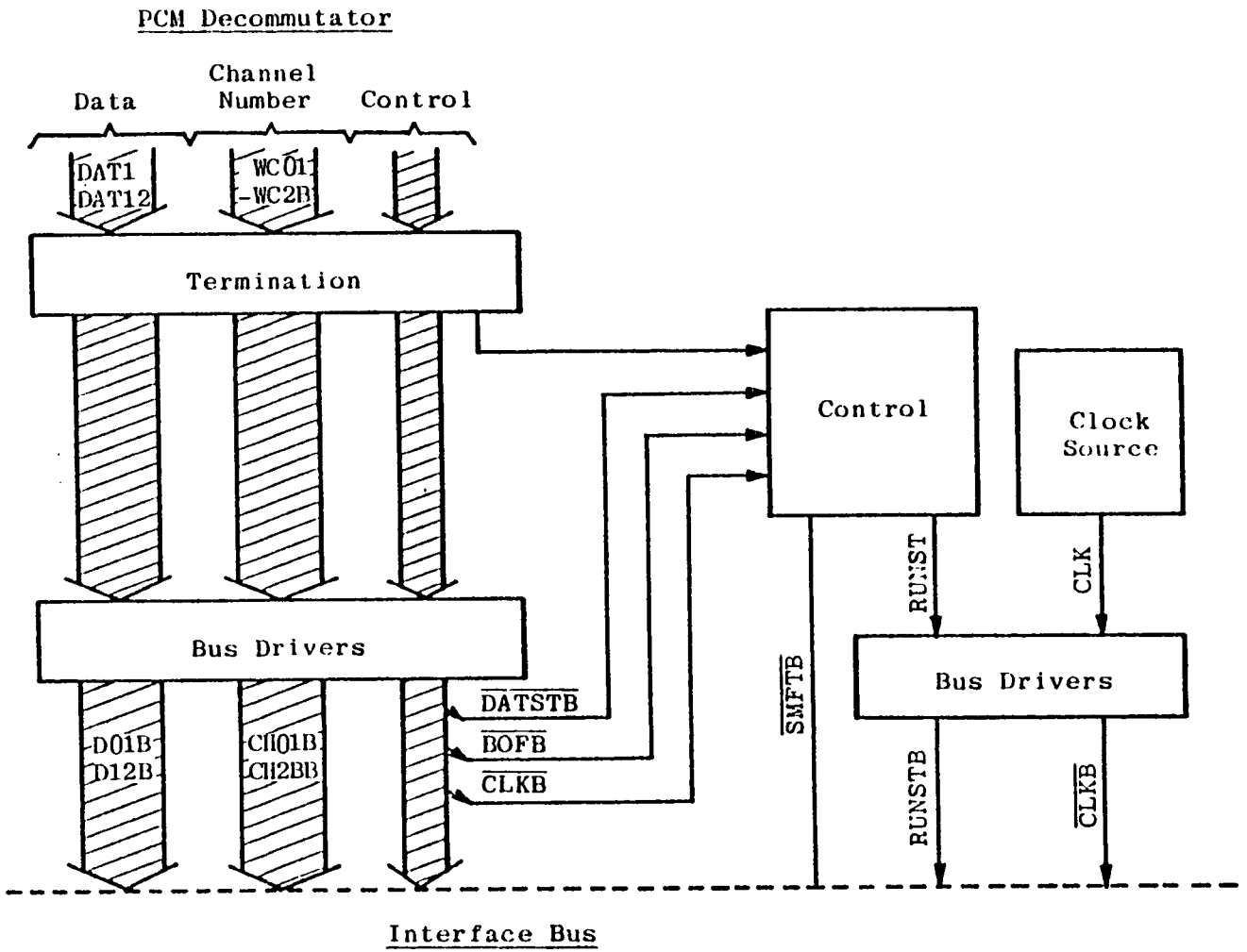


Figure A.4. Decommutator line receiver module.

signals $\overline{\text{MF LOCK}}$, SF LOCK, $\overline{\text{DATA STROBE}}$, SYNC DET, and SF MARK (25I1, 25I2, 25I3, 25I4, and 25I5, respectively) (Appendix B, page 77) are passed through inverter 8I and bus driver 3I (Appendix B, page 75) to the interface bus.

Gate 34I and discrete components on 33I (Appendix B, page 77) are configured as a crystal oscillator producing an 18.432 MHz TTL signal, CLK18, at 34I8. Signal CLK18 drives IC 32I, configured as a divide-by-four counter, and gate 34I8, producing a 4.608 MHz, 25 percent duty cycle clock at 34I8. The clock passes to the interface bus at 3II1 (signal $\overline{\text{CLKB}}$).

IC 35I (configured as a two-bit, synchronous shift register), gate 3II6, and inverter 36I6 (Appendix B, page 77) produce a pulse one interface clock (CLK) cycle wide. The pulse follows the PCM data strobe's leading edge. The pulse provides an interface bus PCM data strobe at 3II3 (signal RUNSTB).

The "beginning of frame signal" interface bus signal, BOF (36II0), asynchronously clocks flip-flop SFH (Appendix B, page 77), conditioned by "sub-frame mark" signal, SFM (36I8),

$$\text{sSFH} = \text{SFM} \cdot \underline{\text{BOF}} \ . \qquad \text{(A.32)}$$

This action occurs during the last frame of every sub-frame. At the next frame's beginning a one-CLK cycle pulse is produced (low-true) at 3II8,

$$\overline{\text{SFTB}} = \text{SFH} \cdot \text{RUNST} \ , \qquad \text{(A.33)}$$

and placed directly on the interface bus. The current "beginning of frame pulse" clocks SFH reset,

$$rSFH = \text{"1"} \cdot BOF \ . \tag{A.34}$$

A.5 FIRST-IN-FIRST-OUT MODULE

The FIFO module provides a means of smoothing data rates when editing and provides data buffering when the computer is block switching. Figure A.5 illustrates the module. Appendix B, pages 79 and 80, contains the module's logic diagram.

PCM data, $\overline{D01B}$ through $\overline{D12B}$, and flags, $\overline{MFLB}$, SFLB, and $\overline{BOFB}$, enter the FIFO through inverters 1I, 2I, and 3I (Appendix B, page 79) from the interface bus. An additional flag enters FIFO chip 12I at pin 6. Data and flags pass through FIFO chips 9I through 12I and 19I through 22I, inverters 26I through 28I, and through connectors 23I through 25I to the computer.

Signal $\overline{FCLR}$ (15I11 (Appendix B, page 80) and 12I9 (Appendix B, page 79)) clears data from the FIFO chips and resets flip-flops 14I5 and 14I9 (Appendix B, page 79),

$$rINREQ = RIRdc = FCLR + \ldots \ , \tag{A.35}$$

$$rOUTREQ = RORdc = FCLR + \ldots \ . \tag{A.36}$$

During quiescence, if the FIFO memory is not full, signals IR9 through IR12 (9I2 through 12I2, respectively) (Appendix B, page 79) are true and CIR is true,

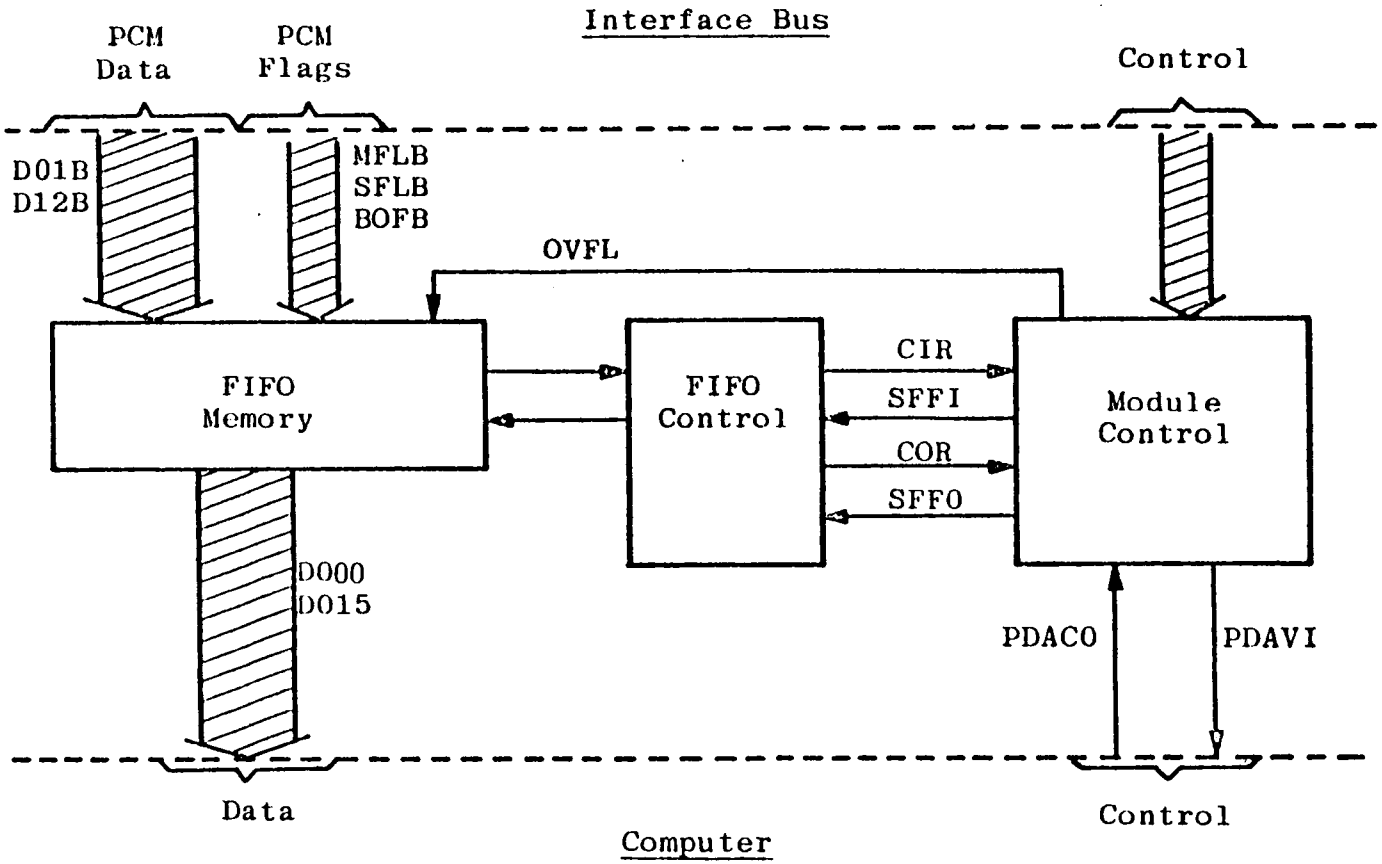


Figure A.5. First-in-first-out module.

$$\begin{aligned} CIR &= \overline{\overline{CIRG} \quad \overline{INREQ}} \\ &= CIRG + INREQ \ , \end{aligned} \tag{A.37}$$

where,

$$\begin{aligned} \overline{CIRG} &= \overline{\overline{IR9} + \overline{IR10} + \overline{IR11} + \overline{IR12}} \\ &= IR9 \cdot IR10 \cdot IR11 \cdot IR12 \ . \end{aligned} \tag{A.38}$$

When data are available to the FIFO, signal SFFI (17I8 (Appendix B, page 80) and 14I2 (Appendix B, page 79)) pulses. The pulse's rising edge sets INREQ,

$$sINREQ = CIRG \ \underline{SFFI} \ . \tag{A.39}$$

Strobing data into FIFO chips causes IR9 through IR12 to go false (temporarily, if the FIFO is not full), resetting INREQ,

$$\begin{aligned} rINREQ &= \overline{\overline{RIR}} \\ &= \overline{IR9} \cdot IR10 \cdot IR11 \cdot IR12 \ , \end{aligned} \tag{A.40}$$

which causes CIRG to go false (Eq. (A.38)) and CIR to go false (Eq. (A.37)). If the FIFO is not full, the circuitry is ready to accept subsequent data.

When data are available at the FIFO outputs, signals OR19 through OR22 are true. In the quiescent state, flip-flop OUTREQ is reset. With the previous two conditions, COR is true,

$$\begin{aligned} COR &= \overline{\overline{OUTREQ} \cdot \overline{CORG}} \\ &= OUTREQ + CORG \ , \end{aligned} \tag{A.41}$$

where,

$$CORG = \overline{\overline{OR19} + \overline{OR20} + \overline{OR21} + \overline{OR22}}$$

$$= OR19 \cdot OR20 \cdot OR21 \cdot OR22 \ . \qquad (A.42)$$

When FIFO output data have been read at their destination, signal SFFO (7I8 (Appendix B, page 80) and 14I11 (Appendix B, page 79)) pulses. The pulse's rising edge sets OUTREQ,

$$sOUTREQ = CORG \cdot \underline{SFFO} \ . \qquad (A.43)$$

Signal OUTREQ strobes new data to the FIFO chip outputs and causes OR19 through OR22 to go false (temporarily, until new data are available), resetting OUTREQ,

$$rOUTREQ = \overline{\overline{ROR}}$$

$$= \overline{IR19} \cdot \overline{IR20} \cdot \overline{IR21} \cdot \overline{IR22} \ , \qquad (A.44)$$

causing CORG to go false (Eq. (A.42)) and COR to go false (Eq. (A.41)). The circuitry has then returned to the quiescent state.

Flip-flop BOFH (35I6) (Appendix B, page 79) synchronizes signal BOF with the FIFO data input strobe. Signal BOFB sets flip-flop BOFH,

$$sBOFH = \overline{BOFH} \ dc \ , \qquad (A.45)$$

Flip-flop BOFH remains set until the end of FIFO input data transfer,

$$rBOFH = BOFH \cdot \underline{INREQ} \ . \qquad (A.46)$$

Circuits 5I, 6I, 8I, and the components 30I and 31I (Appendix B, page 80) are configured as unit selection logic. The switch positions selected on 30I determine what computer data and what interface signals the module responds to:

$$\begin{aligned} TURUN = {} & \overline{SW1} \cdot \overline{SW0} \cdot RUN0B \\ & + \overline{SW1} \cdot SW0 \cdot RUN1B \\ & + SW1 \cdot \overline{SW0} \cdot RUN2B \\ & + SW1 \cdot SW0 \cdot RUN3B \ , \end{aligned} \tag{A.47}$$

$$\begin{aligned} TUEQU = {} & \overline{SW1} \cdot \overline{SW0} \cdot EQU0B \\ & + \overline{SW1} \cdot SW0 \cdot EQU1B \\ & + SW1 \cdot \overline{SW0} \cdot EQU2B \\ & + SW1 \cdot SW0 \cdot EQU3B \ , \end{aligned} \tag{A.48}$$

and,

$$\begin{aligned} TUSEL = {} & (BD14D \oplus \overline{SW1}) \cdot (BD13D \oplus \overline{SW0}) \cdot ADRG \\ = {} & ((BD14D \cdot SW1 + \overline{BD14D} \cdot \overline{SW1}) \cdot \\ & (BD13D \cdot SW0 + \overline{BD13D} \cdot \overline{SW0})) \cdot ADRG \\ = {} & \overline{BD14D} \cdot \overline{BD13D} \cdot \overline{SW1} \cdot \overline{SW0} \cdot ADRG \\ & + \overline{BD14D} \cdot BD13D \cdot \overline{SW1} \cdot SW0 \cdot ADRG \\ & + BD14D \cdot \overline{BD13D} \cdot SW1 \cdot \overline{SW0} \cdot ADRG \\ & + BD14D \cdot BD13D \cdot SW1 \cdot SW0 \cdot ADRG \ . \end{aligned} \tag{A.49}$$

Signals RUN1B, RUN2B, RUN3B, EQU1B, EQU2B, and EQU3B are provided for future interface expansion.

Flip-flop 35I3 (SYNC) (Appendix B, page 80) synchronizes FIFO data with the interface system clock. SYNC is held reset when the

FIFO circuits are not ready,

$$rSYNC = \overline{CIR}\ dc\ . \tag{A.50}$$

When the FIFO circuits become ready and the interface is in the run mode, SYNC is set,

$$sSYNC = TURUN \cdot RUNSTB \cdot \underline{CLK}\ , \tag{A.51}$$

initiating the FIFO input process described above.

Gate 7I11 and inverters 37I2, 37I4, and 37I6 are configured as an edge detecting pulse generator. A pulse (approximately 30 ns) is produced at the leading edge of the signal at 7I3,

$$SETSYNC = TURUNSTB \cdot CLK\ , \tag{A.52}$$

$$OVSTB = SETSYNC \cdot \overline{SETSYNC}\ (\text{delayed})\ . \tag{A.53}$$

If the FIFO circuits are not ready (FIFO is full) when new data are available, flip-flop OVFL is set,

$$sOVFL = \overline{CIR} \cdot \underline{OVSTB}\ . \tag{A.54}$$

Flip-flop OVFL flags subsequent data entering the FIFO (12I6) (Appendix B, page 79). Signal FCLR clears FIFO data and resets OVFL,

$$FCLR = TUSEL \cdot FCLRB\ . \tag{A.55}$$

The remaining circuitry on page 80, Appendix B, provides handshaking for computer input and a strobe (SFFO, 7I8) (Appendix B, page 80) for FIFO output. Flip-flops 18I3 (HS0) and 18I5 (HS1)

(Appendix B, page 80) are held reset when the interface is not in the run mode,

$$rHSO = \overline{TURUN}\ dc\ , \tag{A.56}$$

and

$$rHS1 = \overline{TURUN}\ dc\ . \tag{A.57}$$

When the run mode exists and data become available, HS1 is set,

$$sHS1 = \overline{PDACO} \cdot TURUN \cdot COR \cdot \overline{HSO} \cdot \underline{CLK}\ , \tag{A.58}$$

and PDAVI (36I8) (Appendix B, page 80) goes true, signaling the computer that data are available. When the computer has accepted the data, PDACO (29I2) (Appendix B, page 80) goes true, PDAVI goes false, and HSO is set,

$$sHSO = HS1 \cdot PDACO \cdot \underline{CLK}\ , \tag{A.59}$$

and SFFO goes true,

$$SFFO = HSO \cdot HS1\ , \tag{A.60}$$

On the next clock HS1 is reset,

$$rHS1 = HSO \cdot CLK\ . \tag{A.61}$$

Subsequent to SFFO going true, COR (7I5) (Appendix B, page 80) goes false (response to SFFO), allowing HSO to be reset,

$$rHSO = \overline{TURUN \cdot COR} \cdot \underline{CLK}\ . \tag{A.62}$$

The circuitry is then ready for a new output cycle.

A.6 MEMORY AND COMPARATOR MODULE

Figure A.6 illustrates the memory and comparator module. Appendix B, pages 81 and 82, contains the module's logic diagram.

Computer data enter the module from the interface bus at bus driver inputs (1I, 2I, pins 2, 4, 6, 10, 12, and 14) (Appendix B, page 81). During memory load operation data are placed on the module from bus driver outputs (1I, 2I, pins 3, 5, 7, 9, 11, and 13) and enter the memory at memory input/output pins (12I, 13I, and 14I, pins 9, 10, 11, and 12).

PCM channel values enter the module from the interface bus at comparator inputs (4I, 5I, and 6I, pins 1, 9, 11, and 14). During the run mode, when edit is specified, memory data are placed on the module data bus and input to comparator inputs (4I, 5I, and 6I, pins 10, 12, 13, and 15).

Circuits 10I and 11I, configured as an eight-bit synchronous binary counter, provide addresses for the memory during both load and edit operation. The counter is synchronously cleared at 10I1 and 11I1 and by "loading" zeros (pins 3, 4, 5, and 6) at 10I9 and 11I9. The counter is incremented with a strobe at 10I7 and 10I10.

IC's 3I, 7I, 17I, and components in connectors 20I and 21I (Appendix B, page 82) are configured as unit selection circuitry similar to that described in Section A.5. Gate 19I11 (Appendix B, page 82) provides a signal to clear the address counter (10I1 and 11I1) (Appendix B, page 81),

$$\overline{MCLR} = \overline{TUSEL \cdot ADRGB \cdot MCLRG} \ . \qquad (A.63)$$

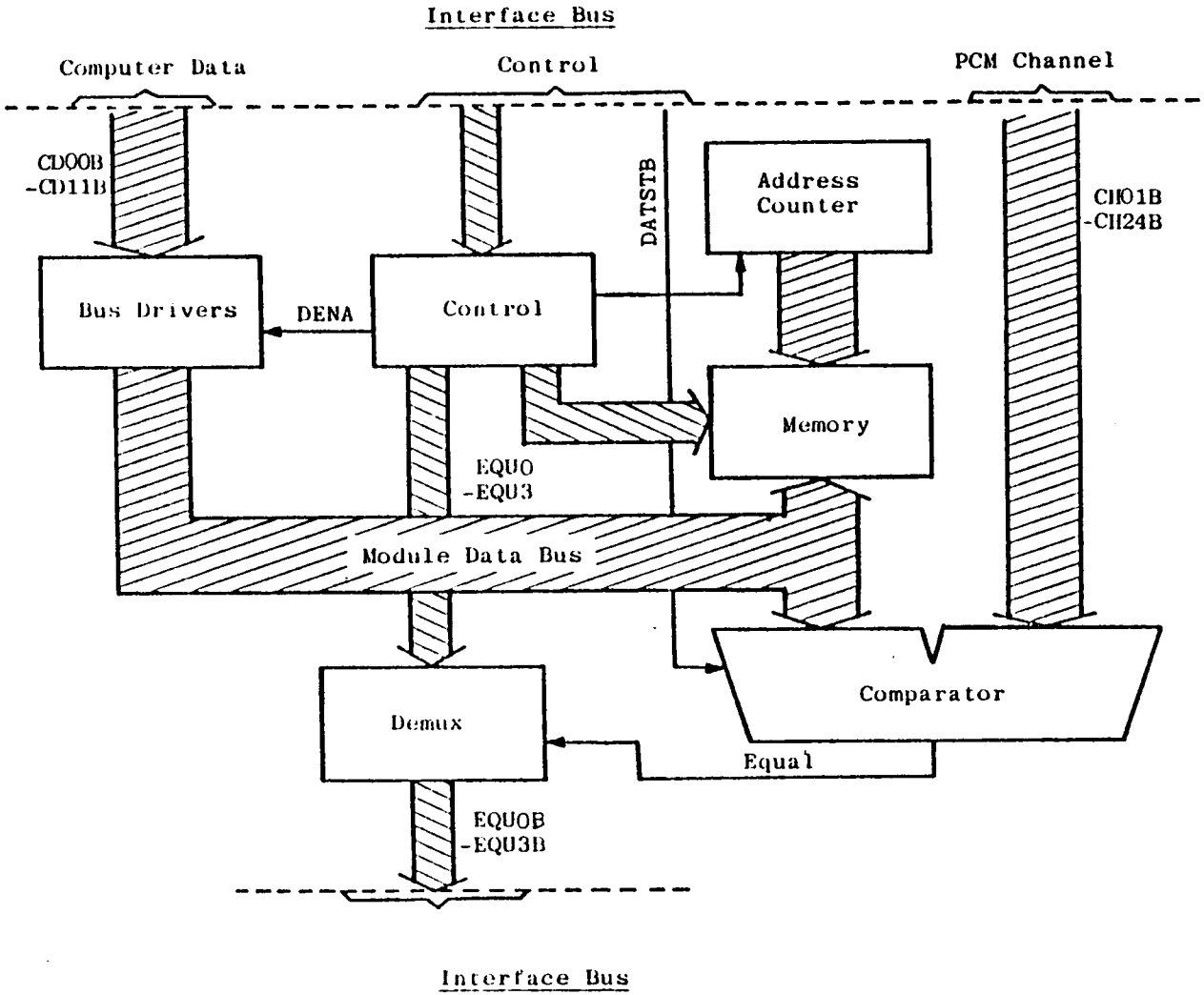


Figure A.6. Memory and comparator module.

During load operations, signals ADV, $\overline{\text{DENA}}$, $\overline{\text{CDST}}$, and $\overline{\text{CENA}}$ (18I, pins 6, 8, and 11, and 23I2, respectively) (Appendix B, page 82), control memory sequencing. Signal $\overline{\text{DENA}}$ places computer data on the module data bus (1I1, 1I15, 2I1, and 2I15) (Appendix B, page 81),

$$\overline{\text{DENA}} = \overline{\text{LOADB} \cdot \text{TUSEL} \cdot \text{DENAB}} \ . \qquad \text{(A.64)}$$

Signal $\overline{\text{CENA}}$ enables the memory IC's (12I13, 13I13, and 14I13) (Appendix B, page 81),

$$\overline{\text{CENA}} = \overline{\text{LOADB} \cdot \text{TUSEL} \cdot \text{CENAB}} \ . \qquad \text{(A.65)}$$

After each memory write, ADV increments the memory address counter (10I7 and 10) (Appendix B, page 81),

$$\text{ADV} = \text{LOADB} \cdot \text{TUSEL} \cdot \text{ADVB} + \ldots \ . \qquad \text{(A.66)}$$

During run operations, $\overline{\text{DENA}}$ (18I8) (Appendix B, page 82) is held high, disabling computer data to the module bus,

$$\overline{\text{DENA}} = \text{TUSEL} + \ldots \ , \qquad \text{(A.67)}$$

where,

$$\text{TUSEL} = \overline{\text{TURUN}} + \ldots \ . \qquad \text{(A.68)}$$

Similarly, $\overline{\text{CDST}}$ (18I11) is held high, placing data read from memory on the module data bus. Signal ADV (18I6) (Appendix B, page 82) increments the memory address counter each time EQUAL goes true when the edit mode is specified (EQUAL is described below),

$$\text{ADV} = \text{TURUN} \cdot \text{EQUAL} \cdot \text{RUNSTB} + \ldots \ . \qquad \text{(A.69)}$$

Signal OSTB (23I2) (Appendix B, page 82) allows the memory address counter to be cleared by RUNCLR (15I6) (Appendix B, page 81) when current memory data bit 11 (14I12) (Appendix B, page 81) is true,

$$\overline{\text{RUNCLR}} = \text{OSTB} \cdot \text{BIT11} \ . \qquad \text{(A.70)}$$

IC 16I provides an enable signal to 8I or 9I determined by switch settings at 20I (Appendix B, page 82),

$$\overline{\text{EQU0}} = \text{SW0} \cdot \text{SW1} \ , \qquad \text{(A.71)}$$

$$\overline{\text{EQU1}} = \overline{\text{SW0}} \cdot \text{SW1} \ , \qquad \text{(A.72)}$$

$$\overline{\text{EQU2}} = \text{SW0} \cdot \overline{\text{SW1}} \ , \qquad \text{(A.73)}$$

or,

$$\overline{\text{EQU3}} = \overline{\text{SW0}} \cdot \overline{\text{SW1}} \ . \qquad \text{(A.74)}$$

Signal EQU (15I11) (Appendix B, page 82) goes true when EQUAL goes true in the edit mode,

$$\begin{aligned} \text{EQU} &= \overline{\overline{\text{EQUAL}}} + \ldots \\ &= \text{EQUAL} + \ldots \ , \end{aligned} \qquad \text{(A.75)}$$

or is continually high when not in the edit mode,

$$\text{EQU} = \overline{\text{EDIT}} + \ldots \ . \qquad \text{(A.76)}$$

Signal EQU is passed through the selected bus driver circuit (8I and 9I) (Appendix B, page 82) onto the interface bus.

Signal EQUAL (6I6) goes high when the current memory data value and current PCM channel number are equal, and signal COMP (22I6) (Appendix B, page 81) is true,

$$\text{COMP} = \text{EDIT} \cdot \text{DATSTB} \ . \qquad \text{(A.77)}$$

A.7 PULSE GENERATOR AND LINE DRIVER MODULE

Appendix B, page 83, contains the pulse generator and line driver module circuitry. The module produces complementary signal pairs for driving balanced differential signal pairs. Signal pair RUN,$\overline{\text{RUN}}$ (6I6 and 8) are true (RUN high and $\overline{\text{RUN}}$ low) when the interface is in the run mode,

$$\text{RUN} = \text{RUNO} \ . \qquad \text{(A.78)}$$

As Eq. (A.72) implies, this module operates only for "unit 0" selection. Signal pair RUNINT,$\overline{\text{RUNINT}}$ (7I6 and 8) is pulsed when the interface enters the run mode,

$$\text{RUNINT} = \text{RUNOS} \ , \qquad \text{(A.79)}$$

$$\text{sRUNOS} = \text{RUNO} \ . \qquad \text{(A.80)}$$

Signal pair BOFINT,$\overline{\text{BOFINT}}$ (8I6 and 8) is pulsed when the interface is in the run mode and "beginning of frame" occurs,

$$\text{BOFINT} = \text{FRMOS} \ , \qquad \text{(A.81)}$$

$$\text{SFRMOS} = \text{RUNO} \cdot \text{MFTB} \ . \qquad \text{(A.82)}$$

A.8 MOTHERBOARD

A printed circuit board contains the 100-line interface bus. Each signal is terminated through a resistor to a regulated voltage. Appendix C provides a tabulation of interface bus pin numbers and corresponding signal names and signal functional descriptive phrases.

A.9 POWER SUPPLIES

Appendix B, page 87, contains the interface power supply schematic. The Powertec Model 2C5-6B provides primary five-volt power. IC 1 and resistor divider R1,R2 provide -12-volt power referenced to ground. IC 2 provides +15-volt power referenced to the -12-volt supply, or +3 volts referenced to ground.

Fuses and power switch are accessible on the enclosure front panel. Indicators I1, I2, and I3 (power-on) are visible on the enclosure front panel.

Connector P9, accessible at the enclosure rear panel, is permanently configured and attached in this enclosure. The connector provides a means of supplying external power to the motherboard when the enclosure is used in other configurations.

APPENDIX B

LOGIC DIAGRAMS

This appendix contains the system logic diagrams and auxiliary drawings of cable diagrams and power supply circuitry (Figs. B.1 through B.7).

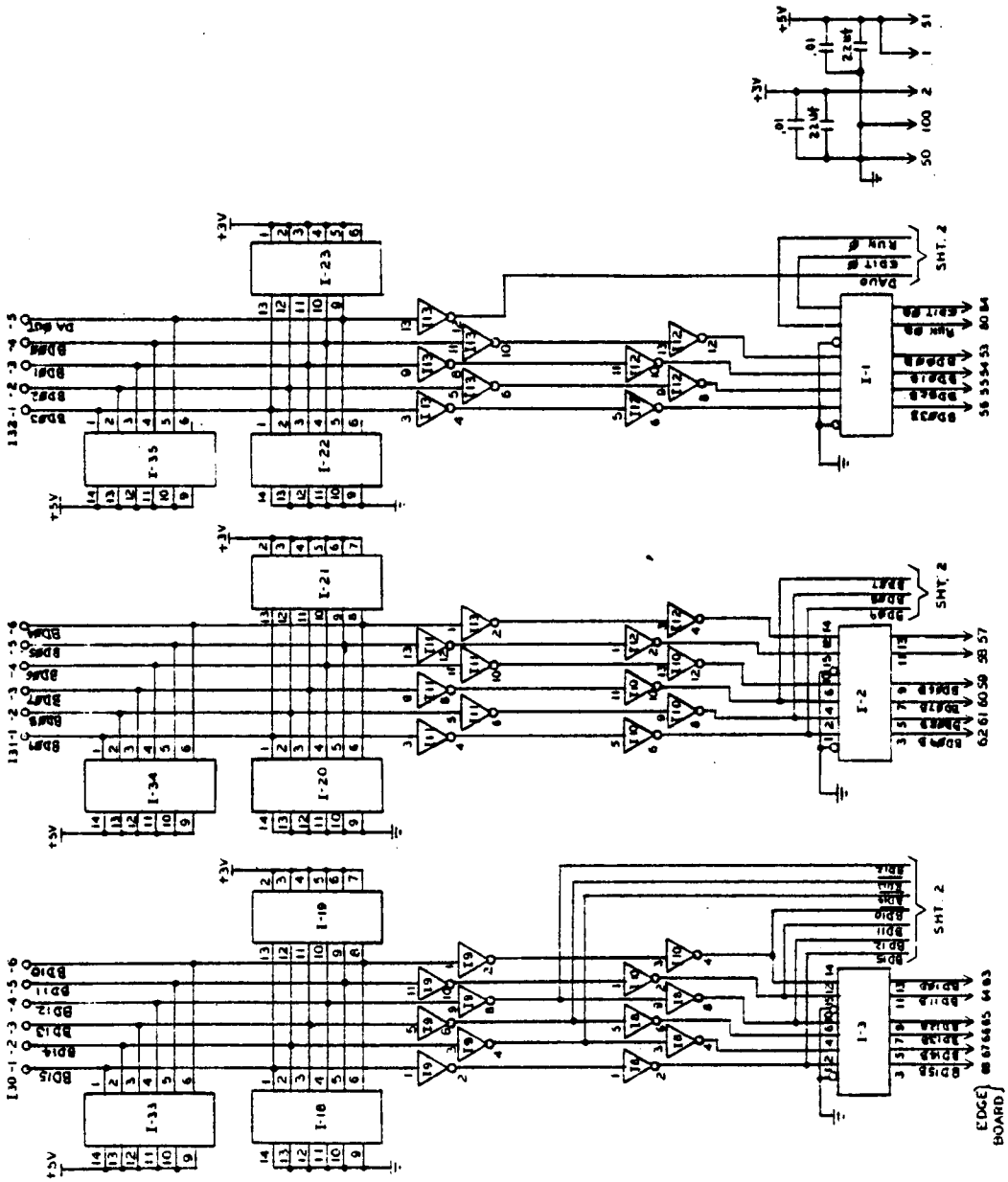


Figure B.1. Telemetry decommutator interface, control.

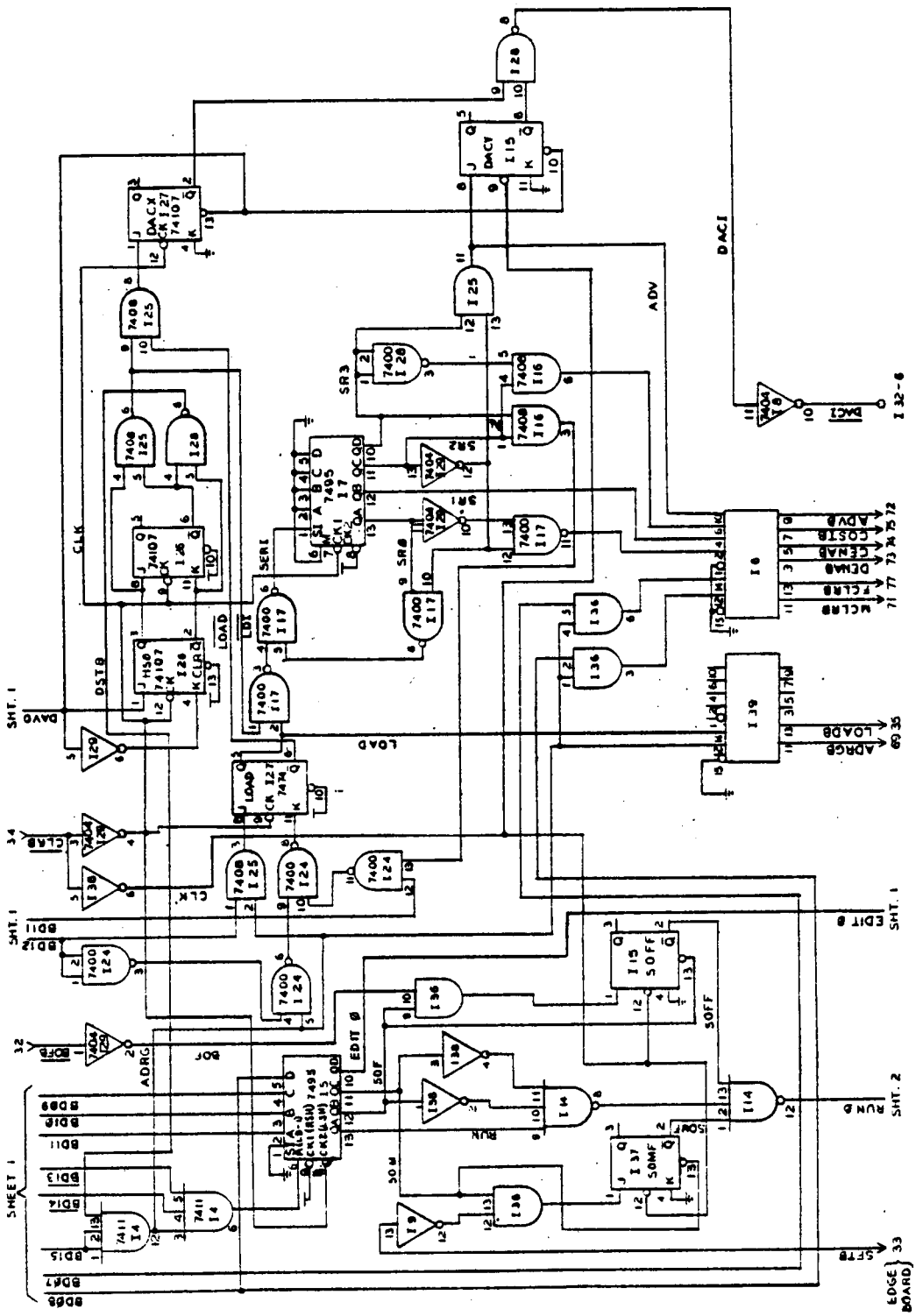


Figure B.1. (Continued)

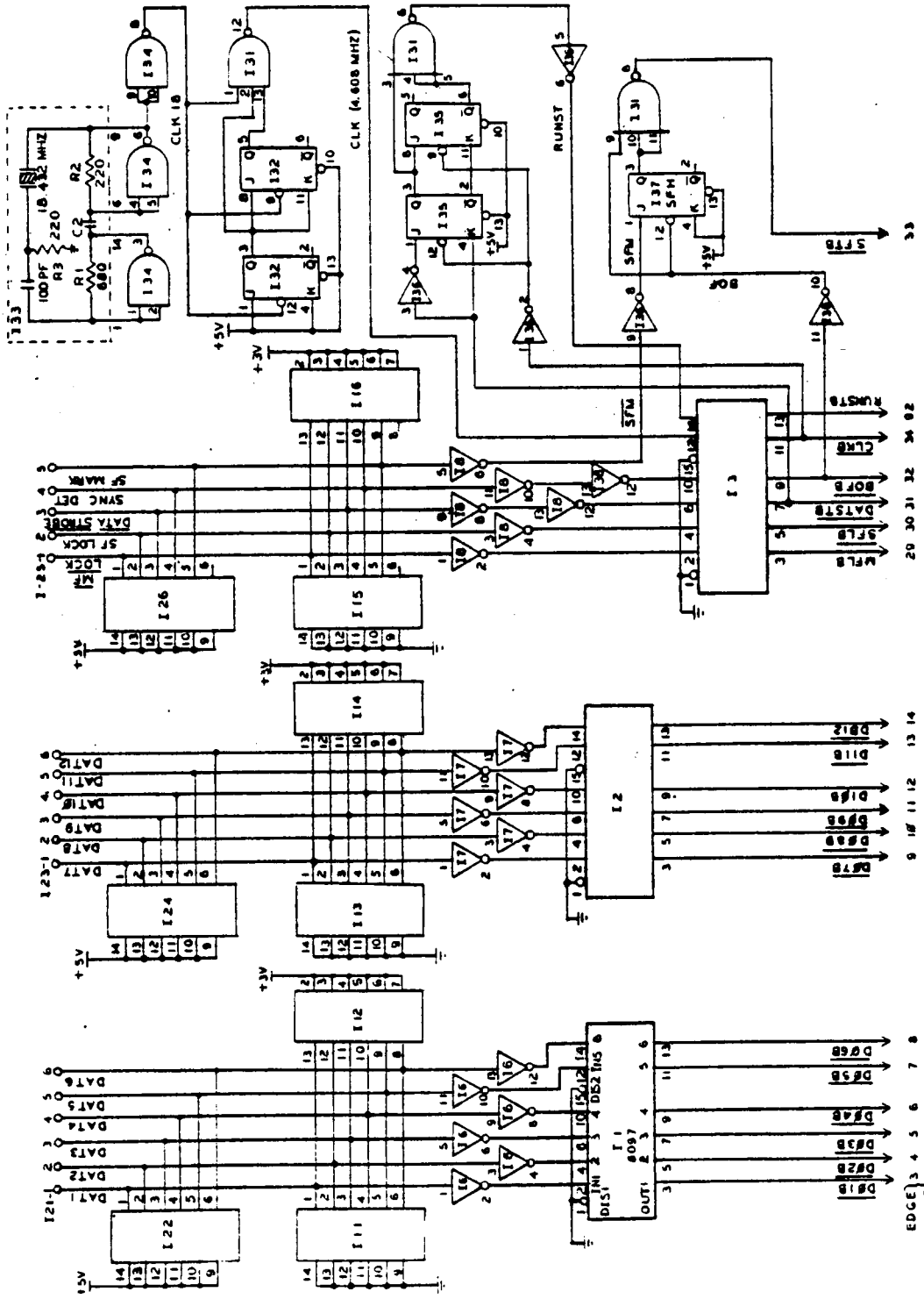


Figure B.2. Telemetry decommutator interface, decommutator line receiver.

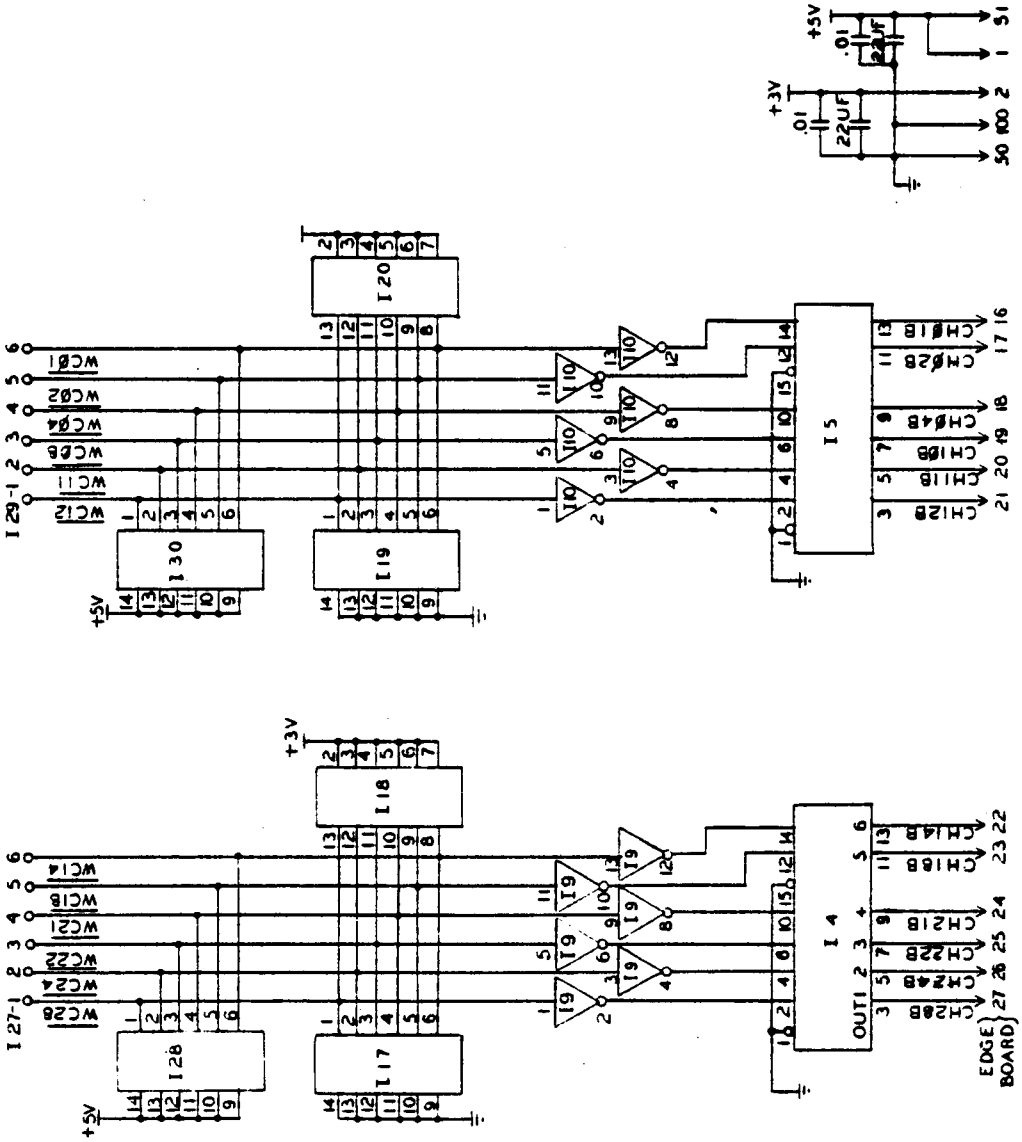


Figure B.2. (Continued)

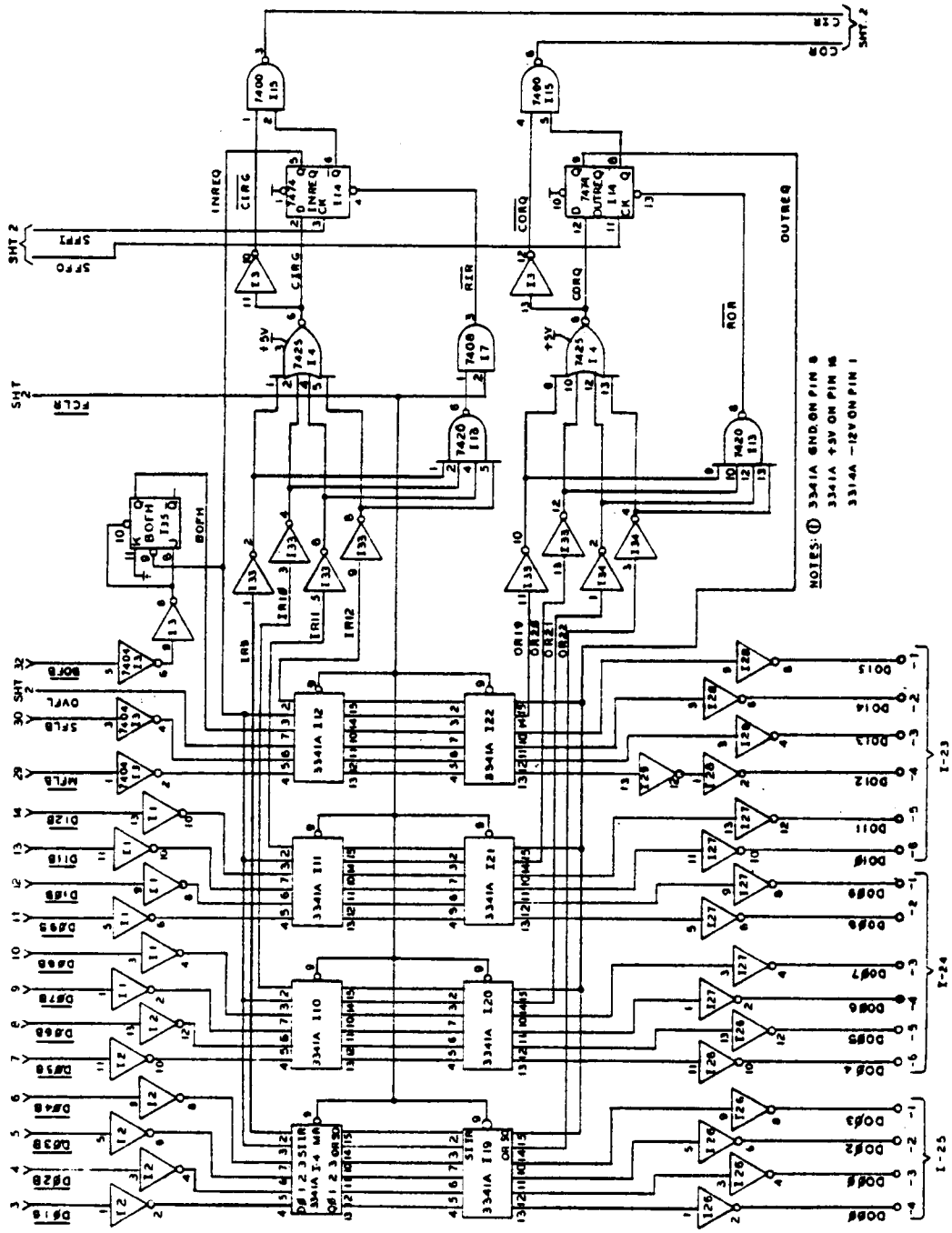


Figure B.3. Telemetry decommutator interface, first-in-first-out.

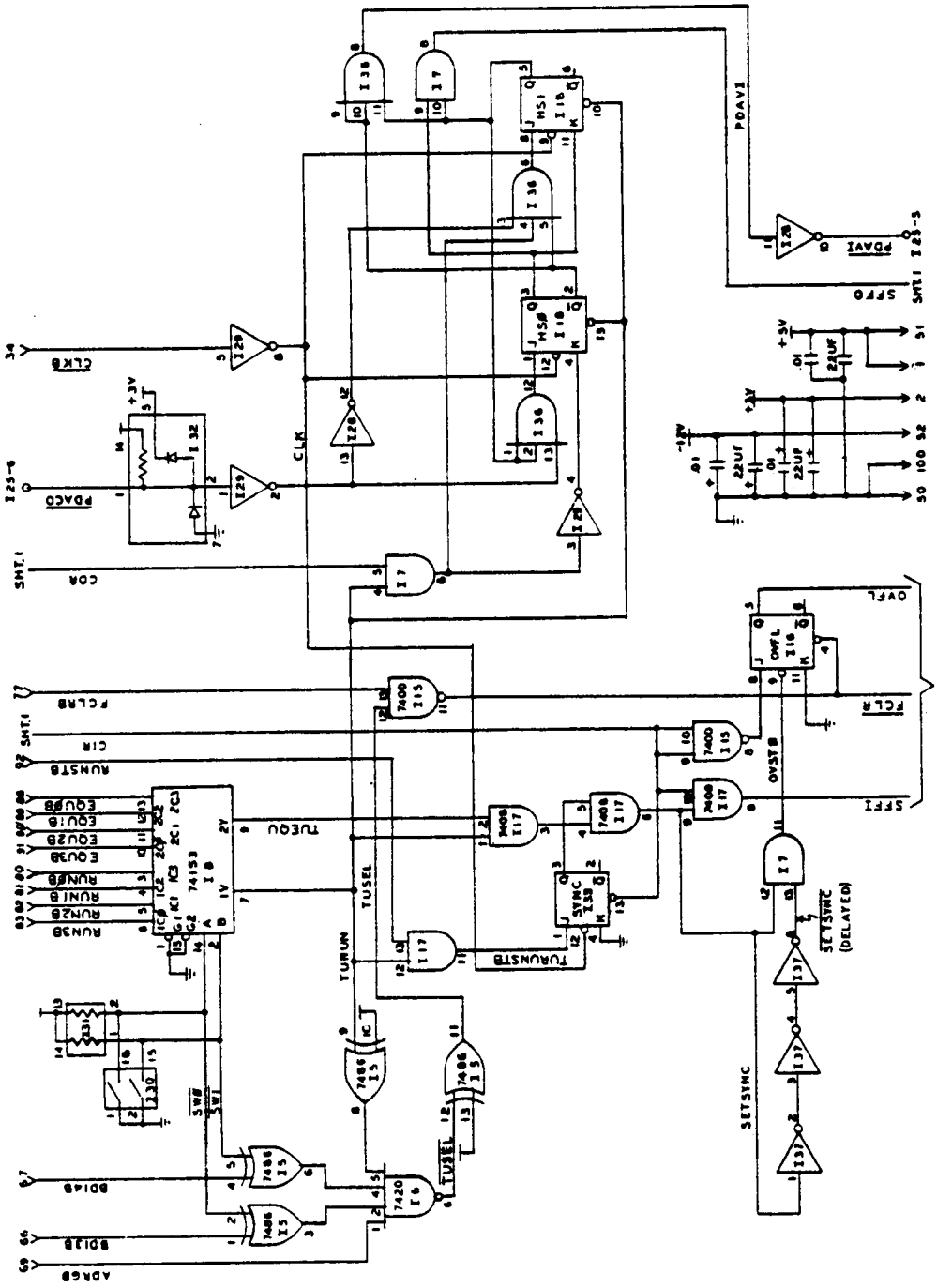


Figure B.3. (Continued)

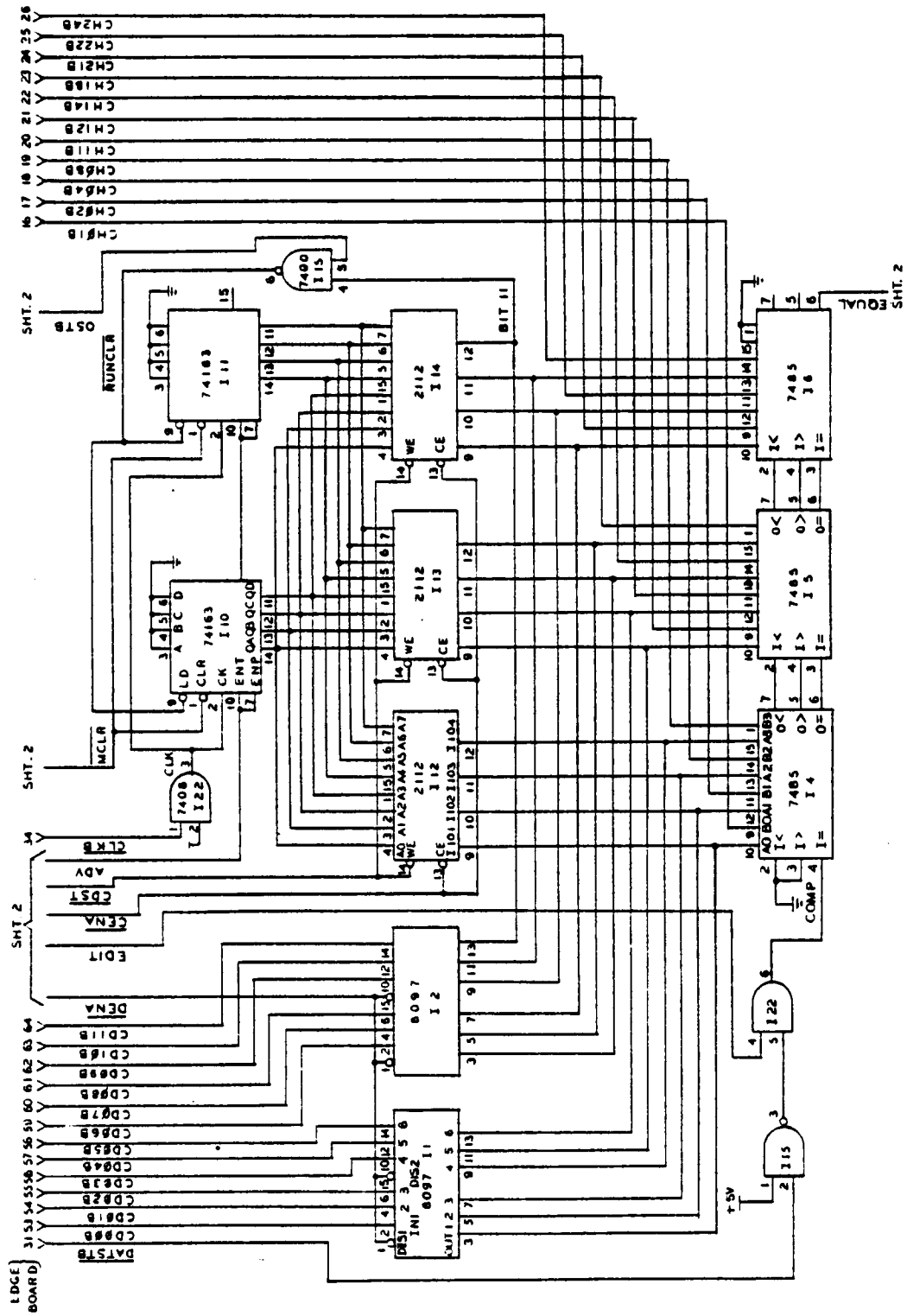


Figure B.4. Telemetry decommutator interface, memory and comparator.

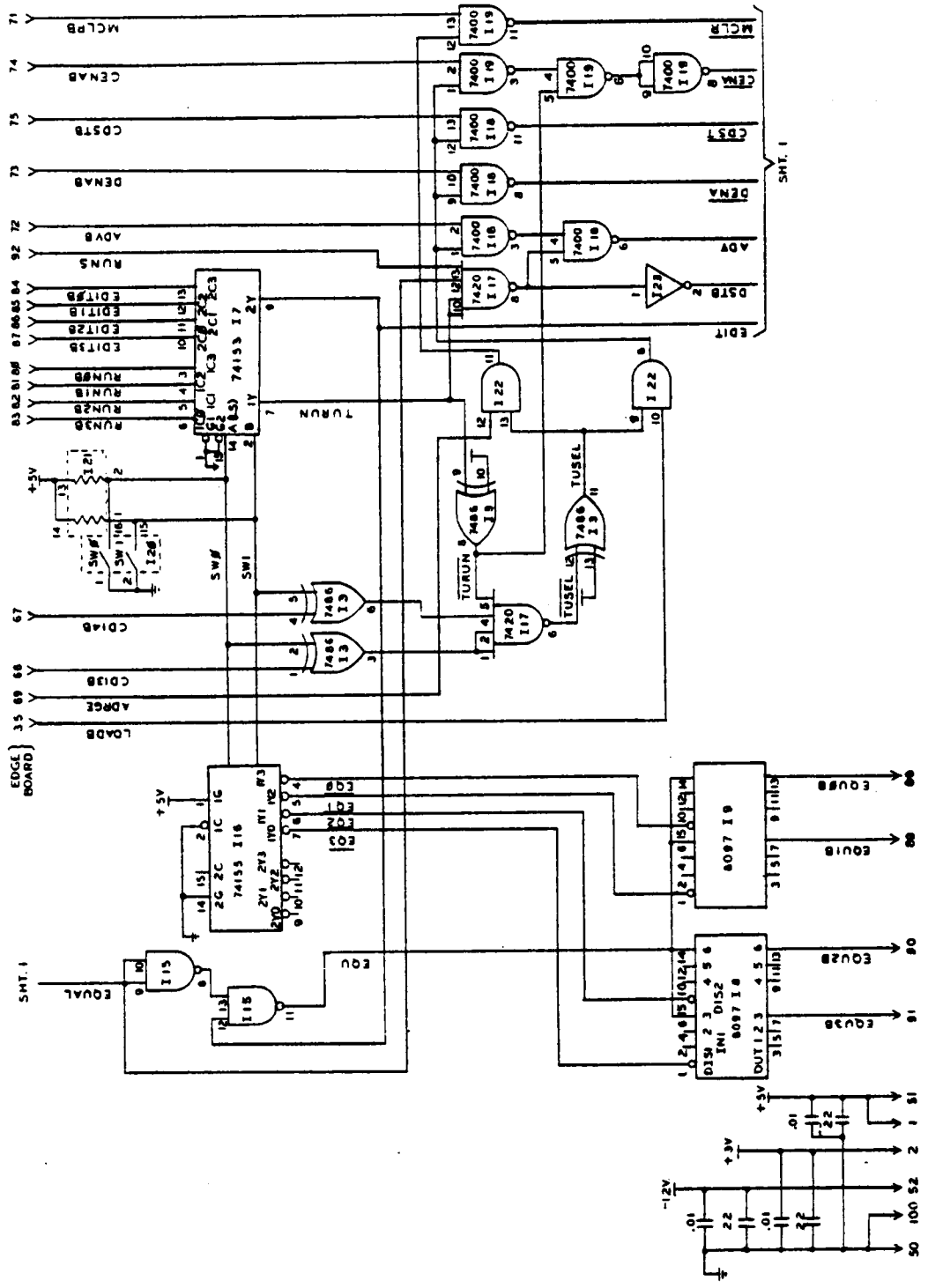


Figure B.4. (Continued)

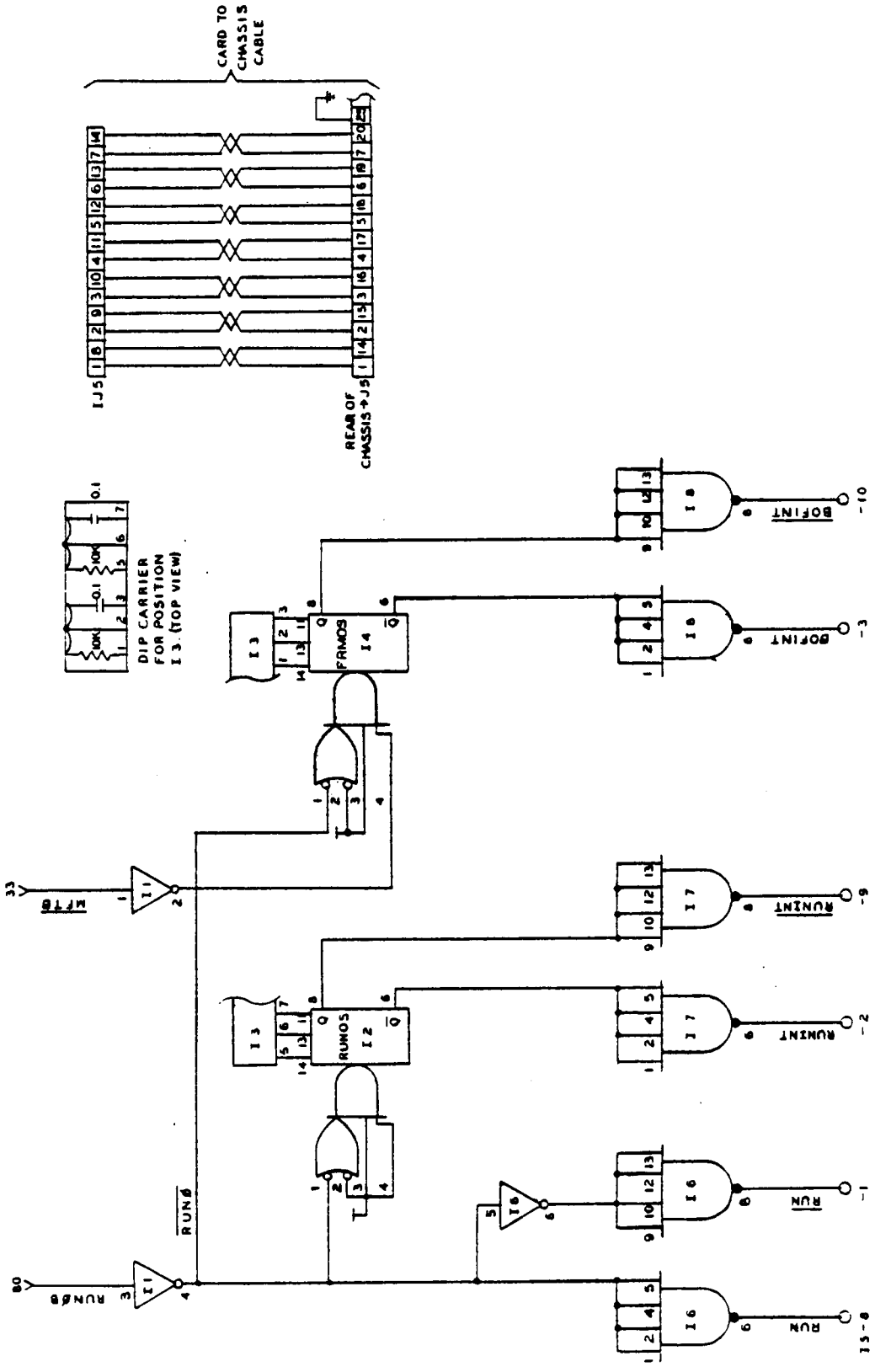


Figure B.5. Telemetry decommutator interface, pulse generator and line driver.

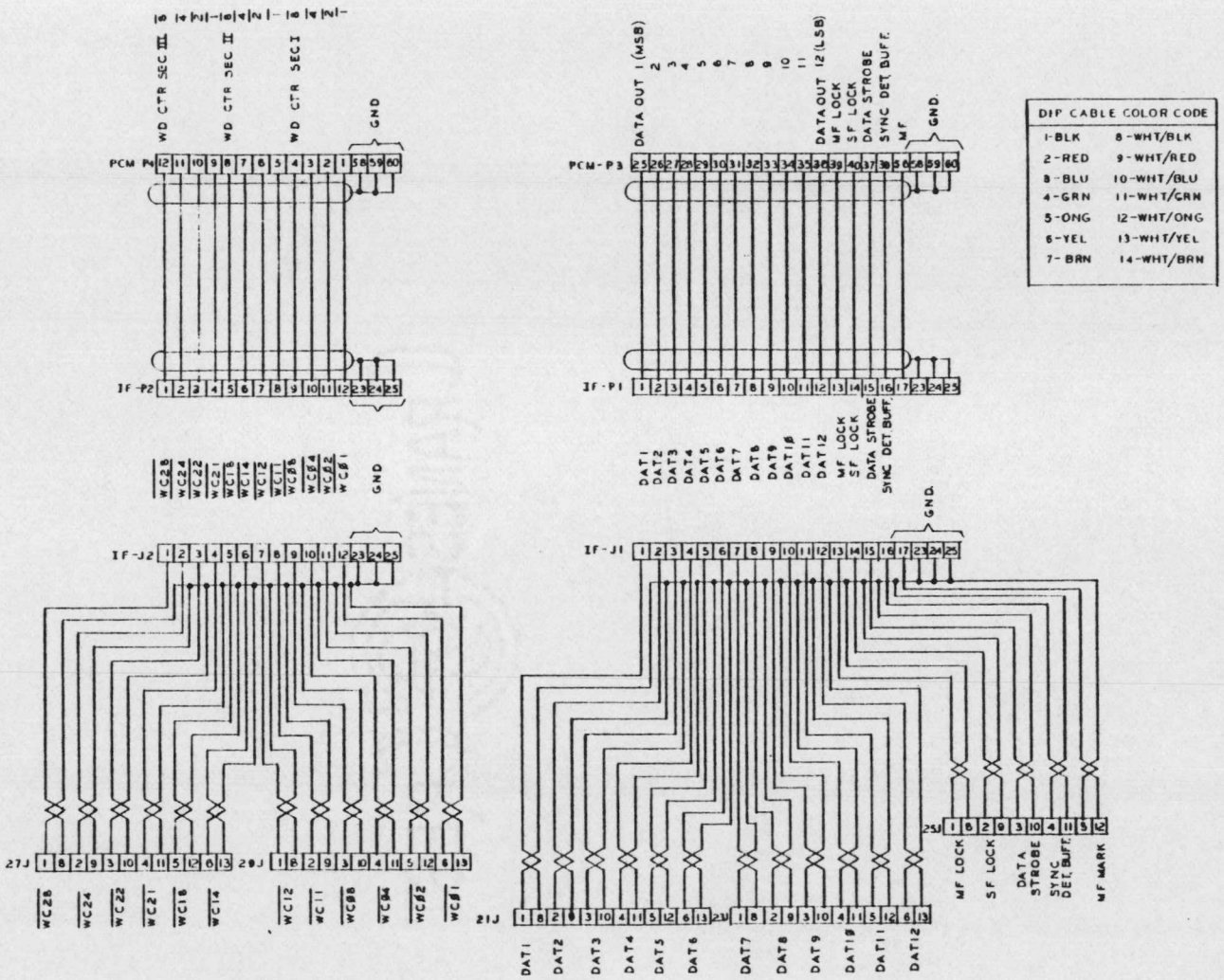


Figure B.6. Telemetry decommutator interface, cabling.

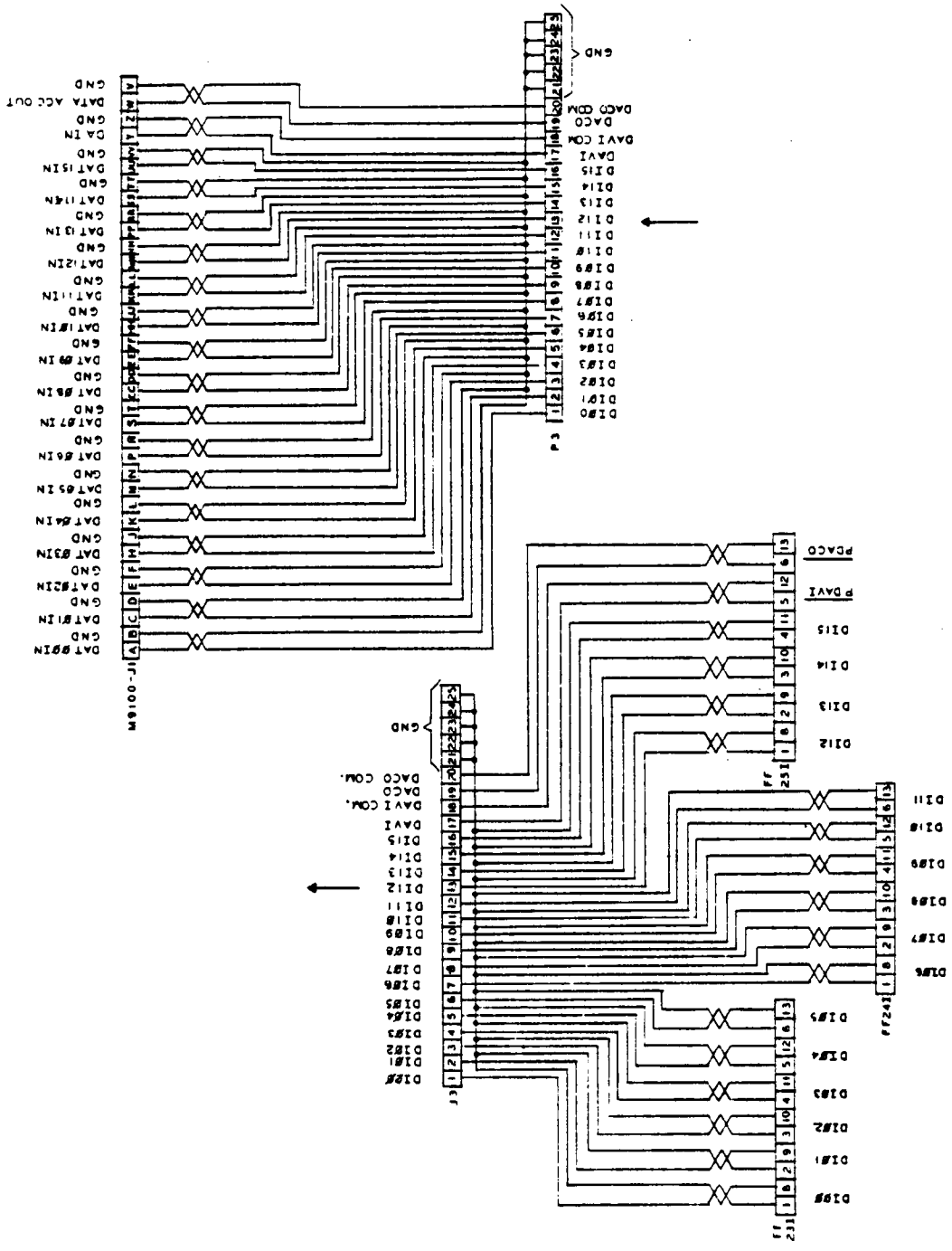


Figure B.6. (Continued)

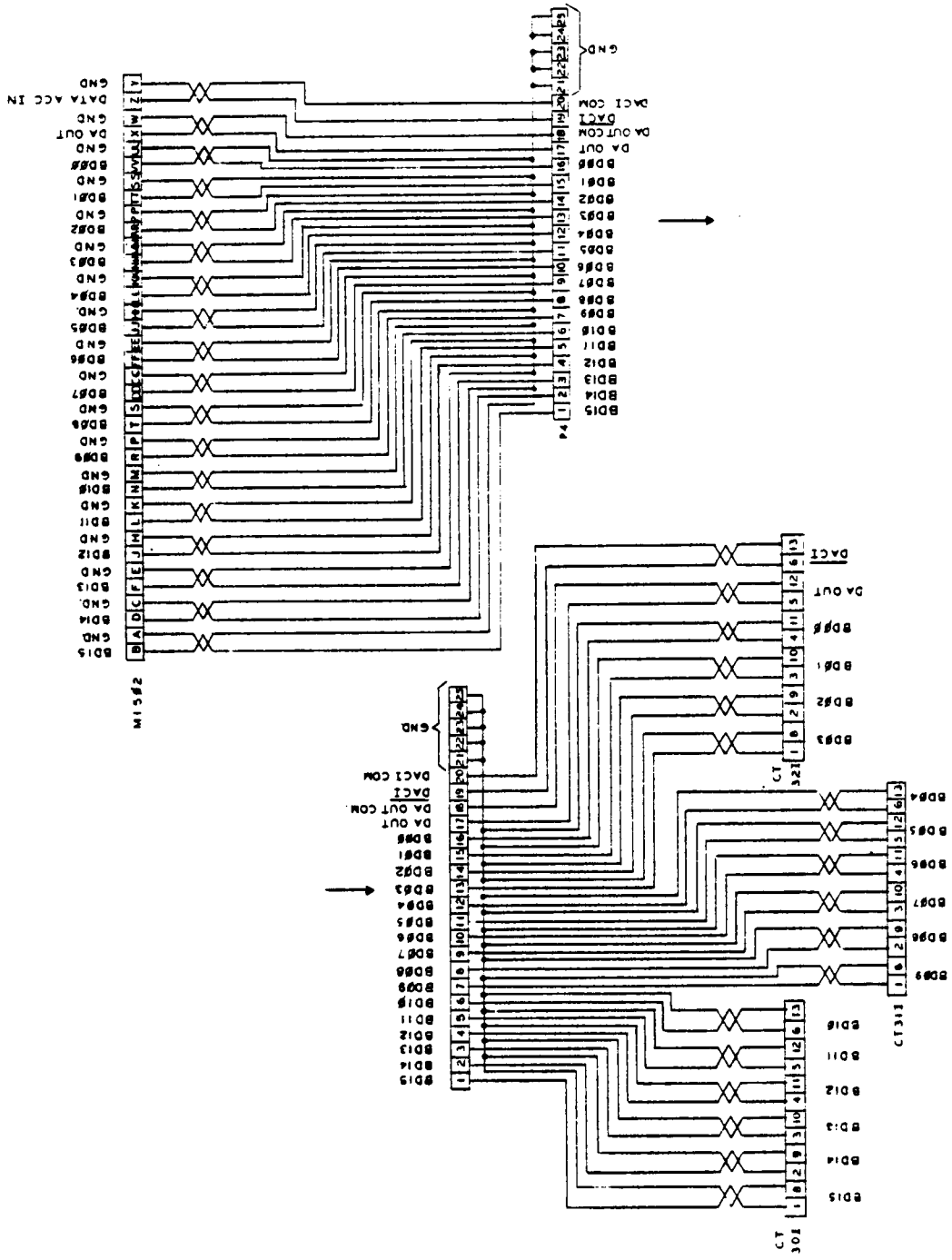


Figure B.6. (Continued)

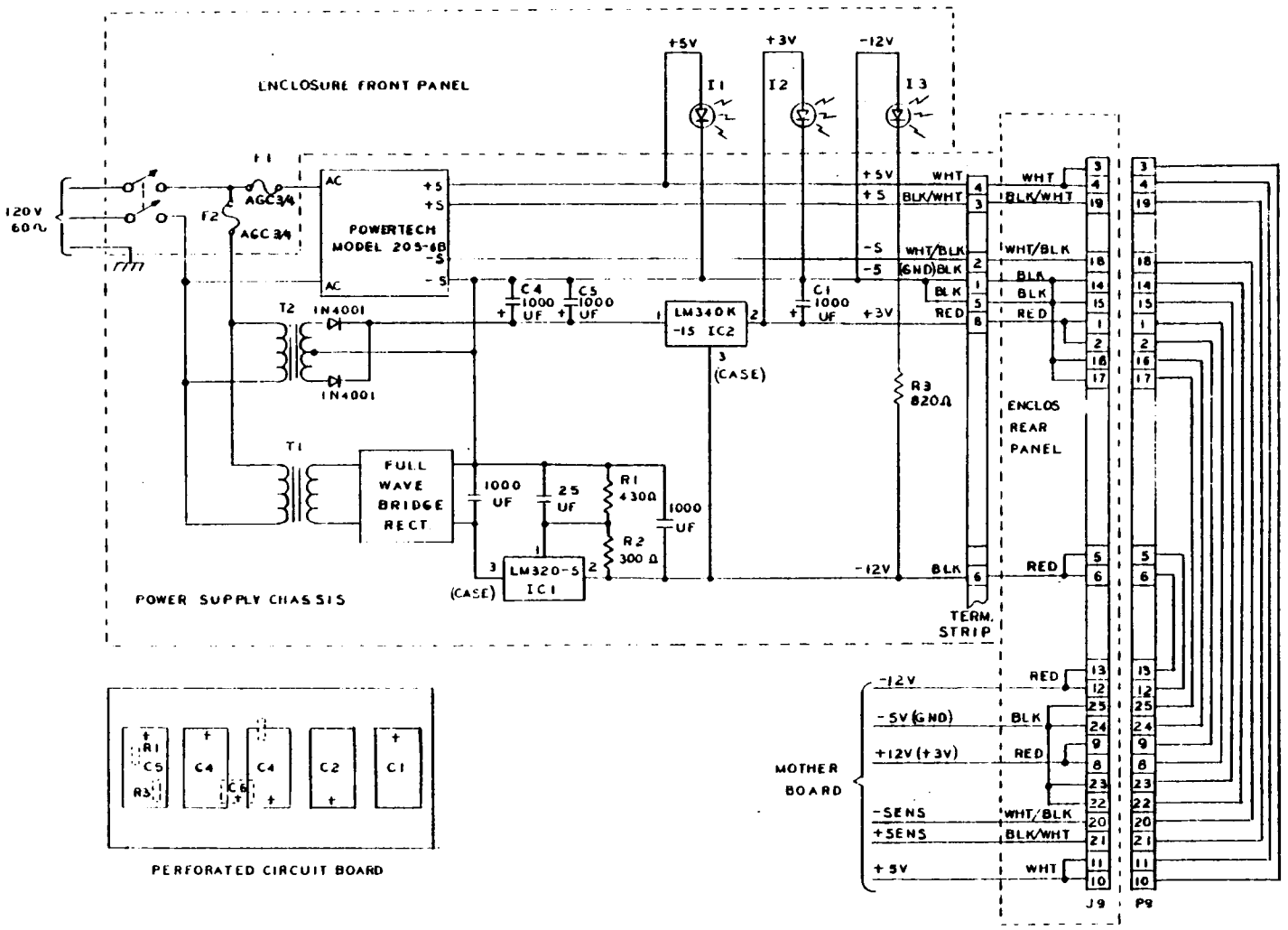


Figure B.7. Telemetry decommutator interface, power supply.

APPENDIX C

BUS PIN NUMBERS AND SIGNALS

This appendix contains the PCM Interface Bus Signal Directory (Table C.1).

Table C.1. Pulse-Code-Modulated Telemetry Decommutator Interface Bus Signal Directory

Pin No.	Mnemonic	Function				Connects to Module			
						PCM Line Rec.	FIFO	Mem.	CTRL
1	+5V	+5 Volt Power Supply				✓	✓	✓	✓
2	+3V	+3 Volt Power Supply				✓	✓		✓
3	BD01B	PCM	Data	Bit	1	✓	✓		
4	BD02B	|	|	|	2	✓	✓		
5	BD03B	|	|	|	3	✓	✓		
6	BD04B	|	|	|	4	✓	✓		
7	BD05B	|	|	|	5	✓	✓		
8	BD06B	|	|	|	6	✓	✓		
9	BD07B	|	|	|	7	✓	✓		
10	BD08B	|	|	|	8	✓	✓		
11	BD09B	|	|	|	9	✓	✓		
12	BD10B	|	|	|	10	✓	✓		
13	BD11B	|	|	|	11	✓	✓		
14	BD12B	PCM	Data	Bit	12	✓	✓		
15	N/U					✓			
16	CH01B	PCM	Channel	Bit	01	✓		✓	
17	CH02B	|	|	|	02	✓		✓	
18	CH04B	|	|	|	04	✓		✓	
19	CH08B	|	|	|	08	✓		✓	
20	CH11B	|	|	|	11	✓		✓	
21	CH12B	|	|	|	12	✓		✓	
22	CH14B	|	|	|	14	✓		✓	
23	CH18B	|	|	|	18	✓		✓	
24	CH21B	PCM	Channel	Bit	21	✓		✓	

Table C.1. (Continued)

Pin No.	Mnemonic	Function	Connects to Module			
			PCM Line Rec.	FIFO	Mem.	CTRL
25	CH22B	PCM Channel Bit 22	✓		✓	
26	CH24B	PCM Channel Bit 24	✓		✓	
27	CH28B	PCM Channel Bit 28	✓			
28	N/U					
29	MFLB	Main Frame Lock Flag	✓	✓		
30	$\overline{\text{SFLB}}$	Sub-frame Lock Flag	✓	✓		
31	$\overline{\text{DATSTB}}$	PCM Data Strobe	✓			
32	BOFB	Beginning of Frame Flag	✓			
33	$\overline{\text{MFTB}}$	Beginning of Frame "Tick"	✓			✓
34	$\overline{\text{CLKB}}$	Bus Clock (4.608 MHz)	✓	✓	✓	✓
35	LOADB	Edit Memory Load Enable			✓	✓
36	N/U					
37						
38						
39						
40						
41						
42						
43						
44						
45						
46						
47						
48						
49	N/U					
50	GND	System Ground	✓	✓	✓	✓

Table C.1. (Continued)

Pin No.	Mnemonic	Function	Connects to Module: PCM Line Rec.	FIFO	Mem.	CTRL
51	+5V	+5 Volt Power Supply				
52	-12V	-12 Volt Power Supply				
53	CD00B	Computer Output Data Bit 0				
54	CD01B	1				
55	CD02B	2				
56	CD03B	3				
57	CD04B	4				
58	CD05B	5				
59	CD06B	6				
60	CD07B	7				
61	CD08B	8				
62	CD09B	9				
63	CD10B	10				
64	CD11B	11				
65	CD12B	12				
66	CD13B	13				
67	CD14B	14				
68	CD15B	Computer Output Data Bit				
69	ADRGB	Address Gate				✓
70	N/U					
71	MCLRB	Memory (Addr. Ctr.) Clear				✓
72	ADVB	Advance (Addr. Ctr.)				✓
73	DENAB	Data Enable				✓
74	CENAB	Chip Enable				✓
75	CDSTB	Computer Data Strobe				✓

Table C.1. (Continued)

Pin No.	Mnemonic	Function	Connects to Module			
			PCM Line Rec.	FIFO	Mem.	CTRL
76	N/U					
77	FCLRB	FIFO Clear		✓		✓
78	N/U					
79	N/U					
80	RUN0B	Run Unit				✓
81	RUN1B	|				✓
82	RUN2B	|				✓
83	RUN3B	Run Unit				✓
84	EDIT0B	Edit				✓
85	EDIT1B	|				✓
86	EDIT2B	|				✓
87	EDIT3B	Edit				✓
88	EQU0B	Equal				
89	EQU1B	|				
90	EQU2B	|				
91	EQU3B	Equal				
92	RUNSTB	PCM Data Strobe "Tick"				
93	N/U					
94	N/U					
95	N/U					
96	N/U					
97	N/U					
98	N/U					
99	N/U					
100	GND	System Ground	✓	✓	✓	✓

VITA

Wilson Elijah McCreary was born in Tuscaloosa, Alabama, on March 17, 1941. He attended elementary school in Brooklyn, Alabama, and graduated from Conecuh County High School in Castleberry, Alabama, in 1959. In June 1959 he entered Auburn University, and in March 1964 he received a Bachelor of Science degree in Electrical Engineering. Subsequently, he accepted a position with Chrysler Corporation's Space Division in Huntsville, Alabama. He remained with Chrysler Corporation until September 1966 when he accepted a position with ARO, Inc., contract operator for the Air Force's Arnold Engineering Development Center near Tullahoma, Tennessee, where he remained until December 1980. He is currently employed at Lockheed Georgia Company, Marietta, Georgia.

He entered the Graduate School at The University of Tennessee Space Institute in January 1967. He received the Master of Science degree with a major in Electrical Engineering and a minor in Computer Science in March 1981. He is a member of the Institute of Electrical and Electronic Engineers, the National Society of Professional Engineers, and the Association for Computing Machinery.

Mr. McCreary is married to the former Mary Alice Hallyburton of Columbus, Georgia. They have two daughters, Cynthia Reneé and Elizabeth Anne.