

# TABLE OF CONTENTS

SECTION		TITLE	PAGE
1.0	PROCES	SOR	
	1. 1	Theory of Operation	1- 1
	1. 2	Assembly - P. C. Board	1- 5
	1. 3	Block Diagram	1- 6
	1. 4	Schematic	1- 7
,	1. 5	Timing Diagram	1- 8
2.0	MEMORY	•	
	2. 1	Theory of Operation	2- 1
	2. 2	Assembly - P. C. Board, 4K RAM	2- 6
	2. 3	Block Diagram, 4K RAM	2- 7
	2. 4	Schematic, 4K RAM	2-8
	2. 5	Assembly, P. C. Board, 4K PROM	2- 9
	2.6	Block Diagram, 4K PROM	2-10
•	2. 7	Schematic, 44 PROM	2-11
3.0	I/O PO	RT	
	2. 1	Theory of Operation, Communications Board	3- 1
·	2. 2	Assembly - P. C. Board, Communications	3-18
	2. 3	Block Diagram, Communications	3-19
	2. 4	Schematic, Communications	3-20
	2. 5	Theory of Operation, Quad I/O	3-23
	2.6	Assembly, P. C. Board, Quad I/O	3-29
	2.7	Block Diagram, Quad I/O	3-30
	2.8	Schematic, Quad I/O	3-31
	2. 9	Assembly, P. C. Board, ROM Simulation	3-33
	2.10	Schematic, ROM Simulation	3-34

SECTION		<u>TITLE</u>	PAGE
4.0	CONSOL	Æ	
	4. 1	Theory of Operations, Console Display	4- 1
	4. 2	Assembly, P. C. Board	4- 8
	4. 3	Block Diagram	4- 9
	4. 4	Schematic	4-10
	4. 5	Theory of Operations, Console Control	4-11
	4. 6	Assembly, P. C. Board, Console Control	4-25
	4, 7	Block Diagram, Console Control	4-26
	4.8	Schematic, Console Control	4-27
	4. 9	Theory of Operation, Console Memory	4-29
	4.10	Assembly, Console Memory	4-32
	4.11	Block Diagram, Console Memory	4-33
•	4.12	Schematic, Console Memory	4-34
5.0	APPLI	CATIONS	
	5. 1	Assembly, Extender Board	5- 1
	5. 2	Schematic, Extender Board	5- 2
	5 3	Breadhoard	5- 3

Theory of Operation: PROCESSOR BOARD (Assembly No. 97380301)

### A.1 INTRODUCTION

The processor (CPU) board contains three elements of the F8 chip set:

- 1. 3850 CPU control element;
- 2. 3852 static memory interface (SMI) which is used only as an interrupt interface including internal (TIMER) and external interrupts; and
- 3. 3853 DYNAMIC MEMORY interface (DMI) which provides addressing and control lines for 64K bytes of dynamic or static memory. Also included are controls for the 3854 Direct Memory Access (DMA) chip.

### A.2 FUNCTIONAL DESCRIPTION

The function of the CPU board is to provide buffered control lines and address lines for the F8 FORMULATOR. Each group of these lines will now be discussed in detail.

A.2. 1 CPU.ROMCO-4\*: 5 lines ROM control to panel

PANEL.ROMCO-4\*: 5 lines ROM control from panel

These two groups are tied together when the panel option is not installed. Depending upon the instruction being executed, these lines define the type, source/destination, and direction of data transfers. Refer to the individual Theory of Operation sections for each chip in the set for the manner in which these lines will be interpreted.

### A.2. 2 ADROO-15\*: 16 LINES ADDRESS TO MEMORY

These lines are active from the CPU board during memory fetches, stores, or refresh operations. During DMA operations, the address source is the 3854 chip.

### A.2. 3 WRITE.DATAO-7\*: 8 LINES DATA TO RAM

These lines are active from the CPU board any time that the MEM.IDLE control line is false. The only time the data is meaningful is during a write or store operation.

# READ.DATAO-7\*: 8 LINES DATA FROM MEMORY

These lines are inputs to latches which are enabled during CPU.SLOT time. The latching operation is required to save the returned data for input to the CPU chip during the second half of the cycle. The data lines may be controlled from a DMA chip during this second half-cycle.

IO.DAT(0-7): 8 LINES OF I/O DATA TO OR FROM
AN I/O TYPE BOARD (Communication
board, Quad I/O board, ROM simulation board, DMA board, or Console
board)

#### A.2. 5 SYS.RESET:

A pulse issued during CYCLE.CLK time when /08 is decoded on the CPU.ROMC lines.

#### A.2. 6 INST.FETCH\*:

A signal provided to the panel option when installed. True during any instruction fetch. (/00 on CPU.ROMC lines.)

### A.2. 7 JUMP\*:

Signal to a test point. True during any jump or branch instruction. (/Ol on CPU.ROMC lines.)

#### A.2. 8 PH.CLK\*:

Phase clock issued by the CPU and buffered to the system. The clock rate is determined by 1 of 3 sources that are switch-selectable on the CPU board.

#### A. 2. 9 CYCLE, CLK\*:

A signal issued by the CPU and buffered to the system. The signal is true at the end of every CPU cycle. The trailing edge signifies the start of another cycle.

#### A. 2. 10 EXT. INT\*:

A global bus signal input to the 3853 SMI chip. When the 3853 interrupt control bit is "ON" and the bus signal is active and incoming priority line is proper, then an Interrupt Request is generated and buffered to the global interrupt line.

### A.2.11 INT.REQ\*:

Global interrupt line. When the panel option is installed, the signal is conditioned by panel hardware. If the option is not installed, then this signal is tied to PAN.INT.REQ\*.

### A.2.12 PAN.INT.REQ\*:

Input to the 3850 CPU chip. If the CPU's Interrupt Control Bit is set, then this signal will cause an Interrupt instruction fetch sequence to occur. The source of the interrupt vector is in the 3852 SMI chip, port /OC for the upper vector and /OD for the lower vector.

#### A.2.13 INT.CON.BIT:

A signal to the panel option if installed. If option not installed, then this signal is not used. It will be true when the Interrupt Control Bit is set in the 3850 CPU chip.

#### A. 2.14 EXT. RESET\*:

An input to the CPU from a global line. When activated it will clear the program counter to /0000 and reset the Interrupt Control Bit. (The same thing happens during CPU power-up.)

#### A. 2. 15 INT. PRI. IN\*:

An input to the 3853 SMI chip. This signal is from the next higher priority board with interrupt capability. When inactive and all other conditions are met, then the 3853 is allowed to interrupt. When active it indicates that some higher priority board is in the process of interrupting.

#### A.2.16 INT.PRI.OUT\*:

A signal to the next lower priority board capable of interrupting. This signal, when inactive, indicates that neither this nor any higher priority board is contemplating an interrupt.

### A. 2.17 CYCLE.REQ\*:

A signal buffered to the base for use in the 4K Dynamic RAM board.

#### A.2.18 CPU.READ\*:

A signal buffered to the base for user-designed memories if required.

#### A.2.19 CPU.SLOT:

This signal is output of the 3852 chip on the processor board. It is high during those portions of a cycle in which the F8 system is using the memory.

#### A. 2. 20 RAM. WRITE\*:

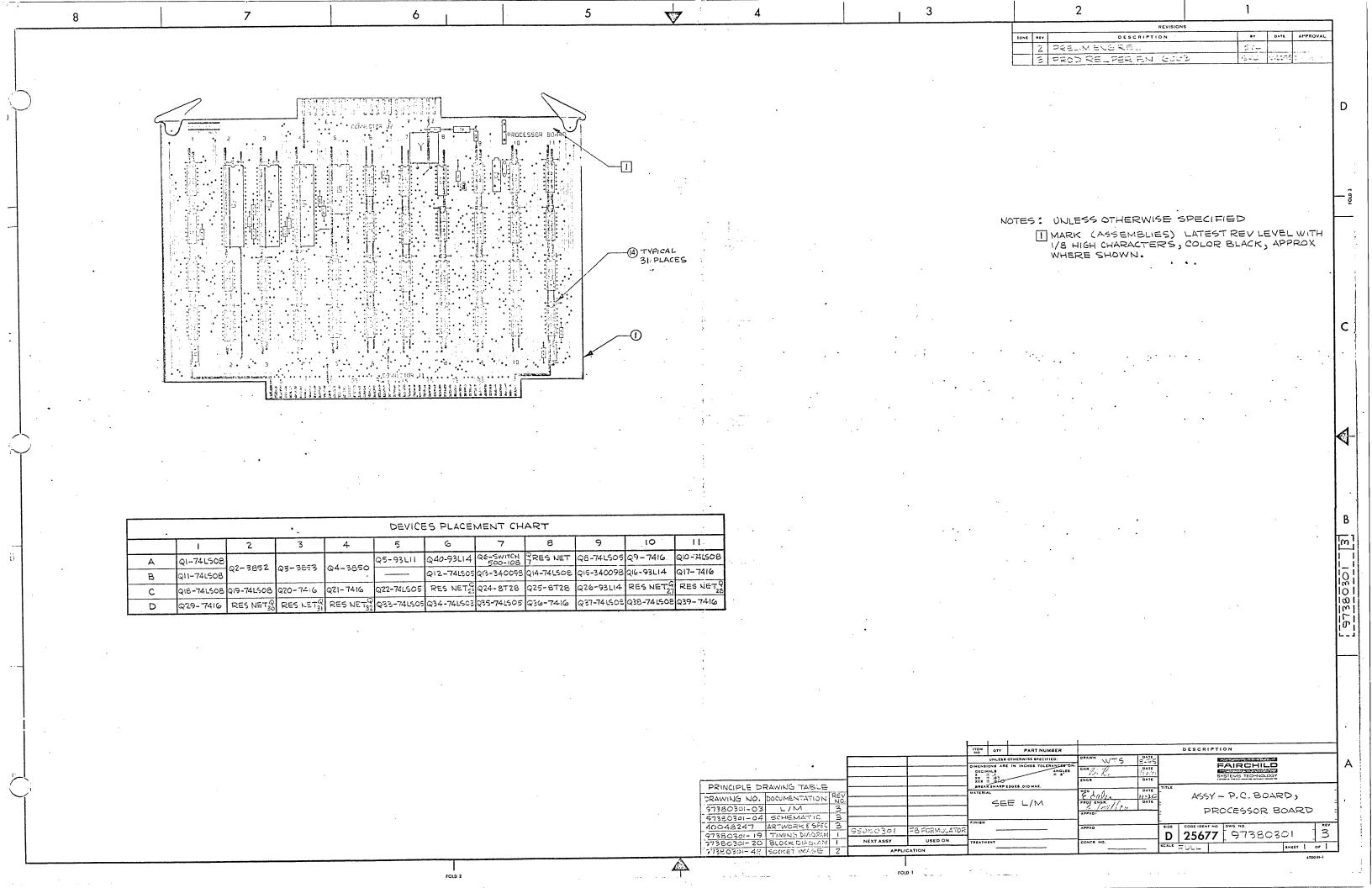
A strobe signal buffered to the RAM boards during a store cycle signifying "Data and addresses are stable, WRITE now."

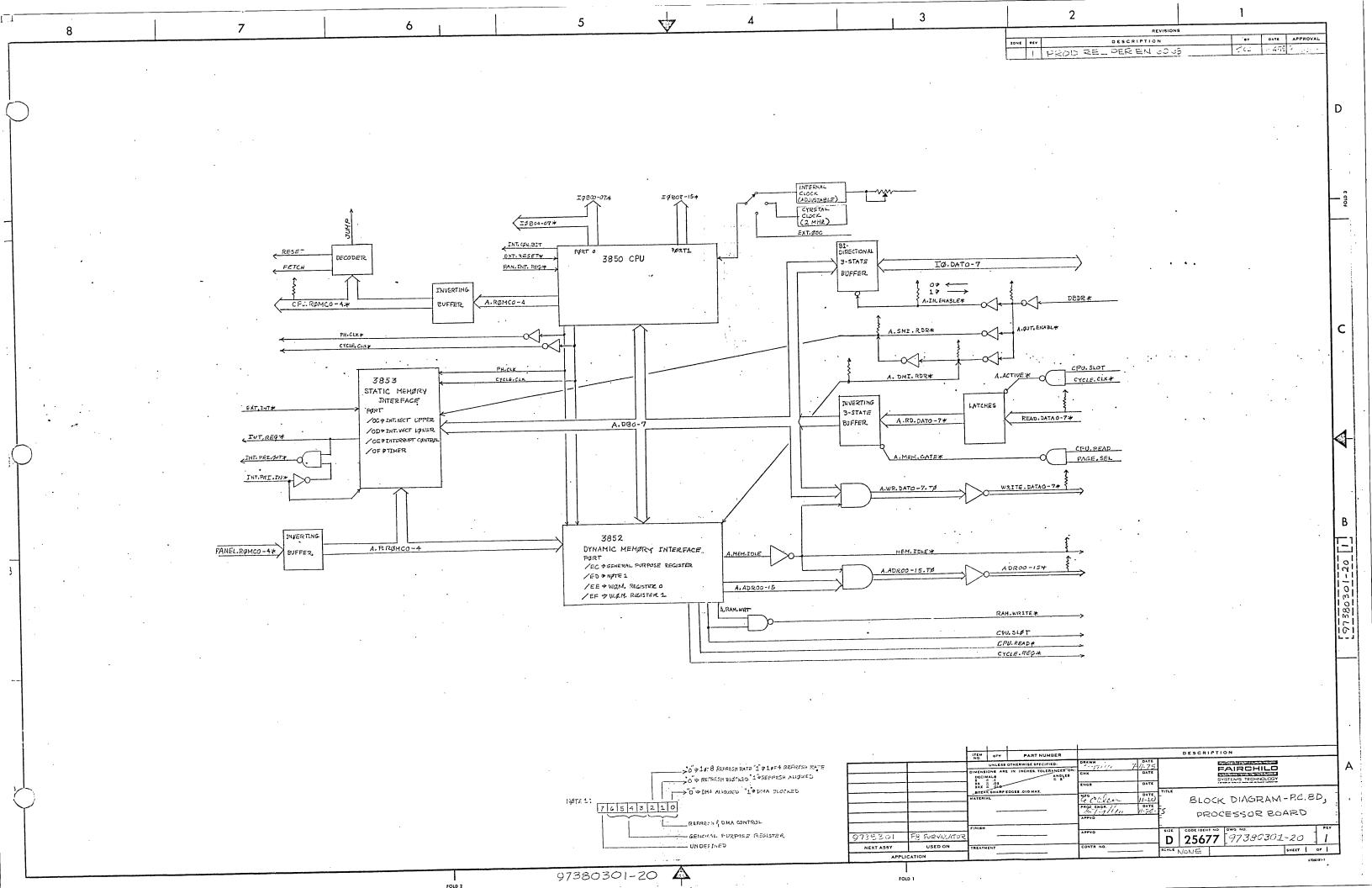
#### A. 2. 21 MEM. IDLE\*:

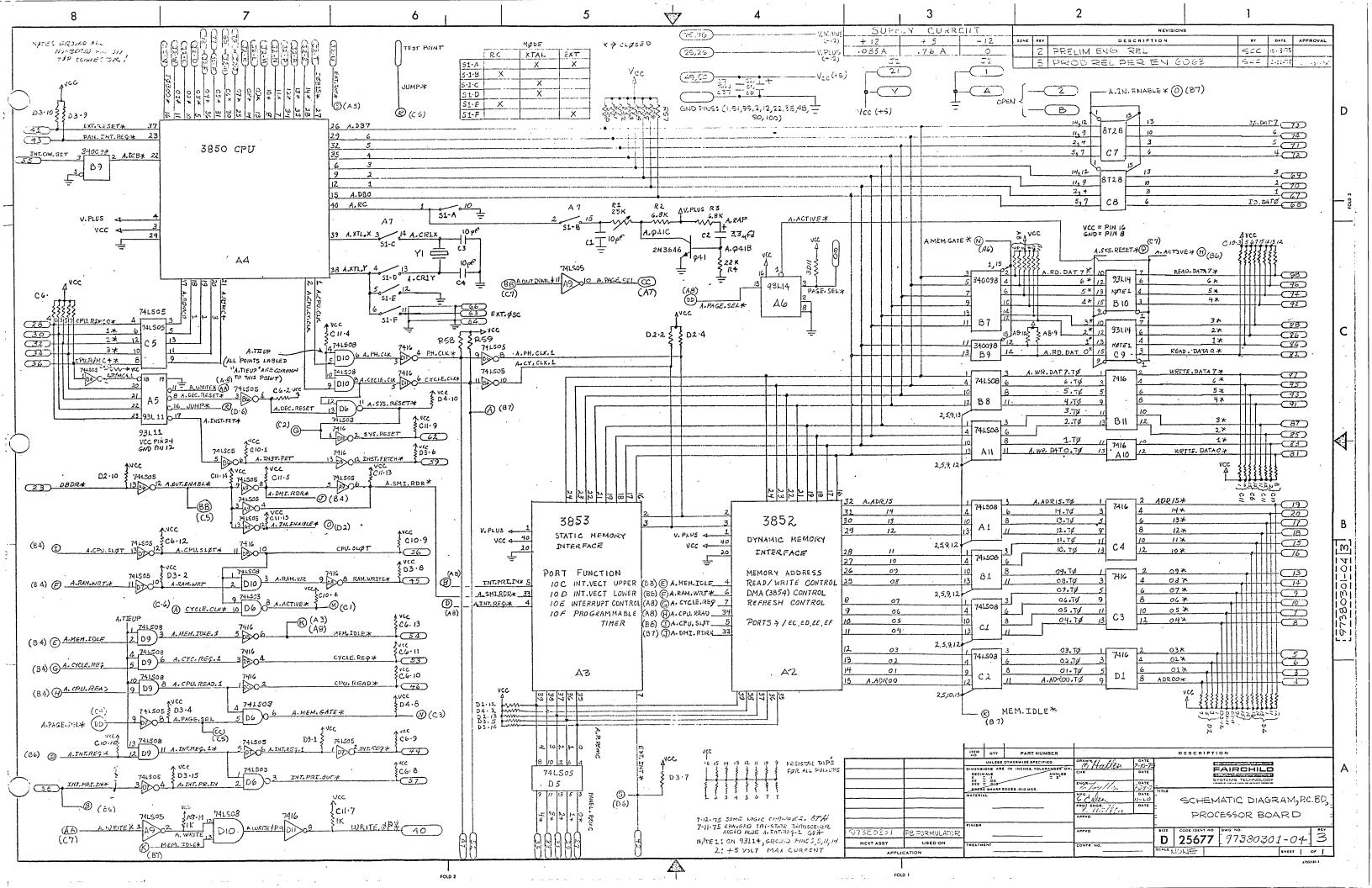
A buffered signal to the 3854 DMA chip. When active, indicates that a DMA cycle may occur if required.

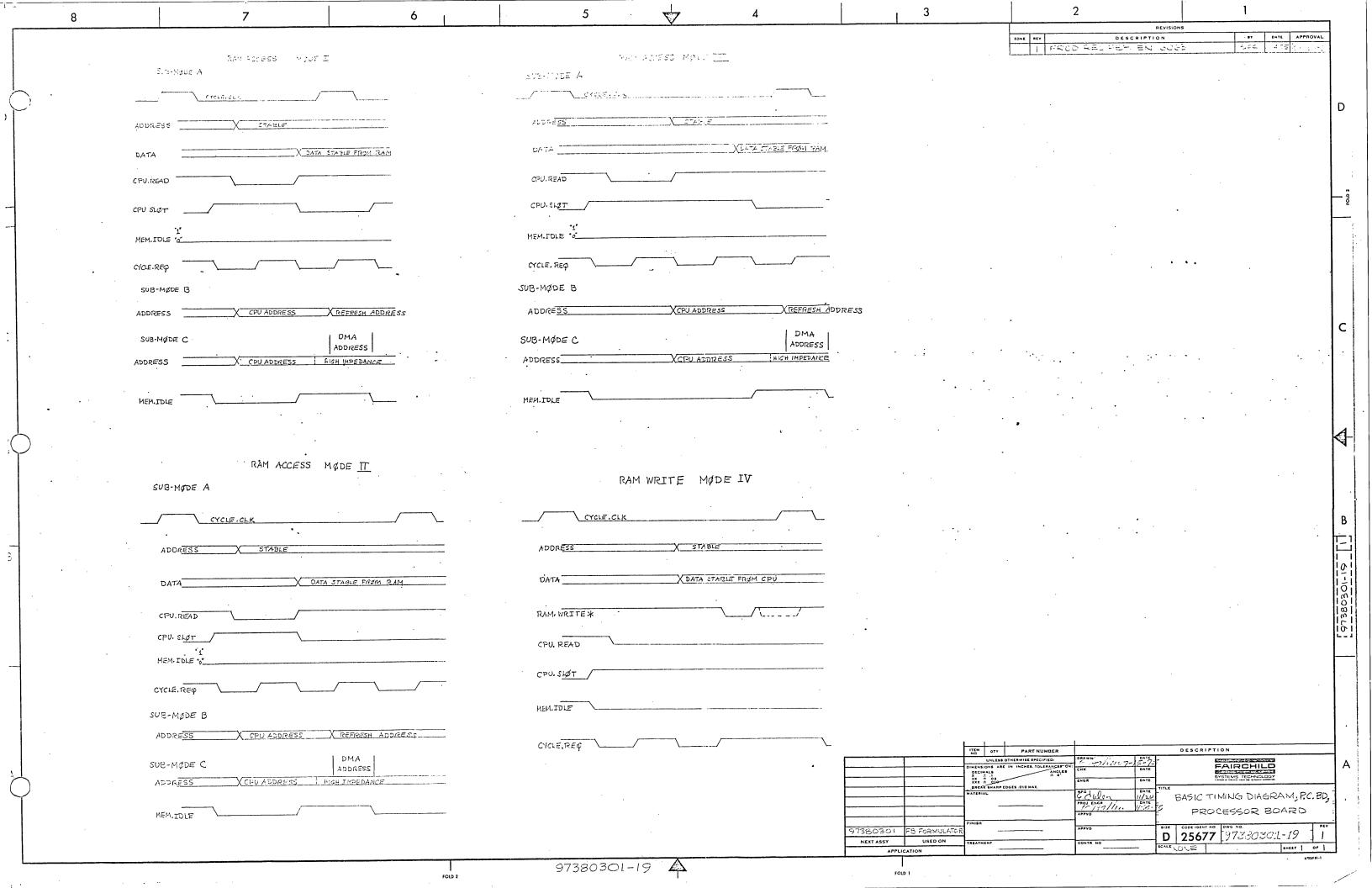
### A. 2. 22 WRITE. OP\*:

A buffered signal to the RAM boards indicating that a store operation is in progress. This will be used to determine direction of Data flow on the RAM boards after CHIP SELECT and before RAM.WRITE\*.









### F8 FORMULATOR MAIN MEMORY FACILITIES

### BC.1 INTRODUCTION

The circuits of the F8 chip set are designed to handle up to 64K of 8-bit words of memory. That is, the memory interface chips are equipped with 16-bit address registers. The F8 FORMULATOR is capable of using up to that maximum of 64K words of 8-bit memory. The memory boards, as presently designed, consist of 4K words each. This increment of 4K words per board will be referred to hereafter as a "page." Thus, the total space of 64K words is divided into 16 pages of 4K each.

Considering the fact that the user memory space will consist of both random access memory (RAM) and read only memory (ROM), the F8 FORMULATOR memory facility was designed so that the user can configure the various pages as he wishes; that is, he may want to have a number of pages of RAM at the bottom of the memory space and, at some point, he may wish to insert a number of pages of ROM and, at some other interval, he may want to continue with random access memory again.

In order to allow the user to achieve any configuration he wishes and also to have a linearized base board in the system where he can plug any memory board into any slot in the base board, the F8 FORMULATOR memory boards were designed with a switch that allows the user to define a particular board as occupying a particular page. Thus, a 4-pole switch is provided on each memory board, representing the four high-order address bits; that is, bits 12, 13, 14, and 15. When that switch is arranged in a particular binary configuration between /0 and /F, that board will respond to any address within that 4-bit, high-order address increment.

During actual operation, the contents of the 4-pole switch is compared with the four high-order address bits and, if a match is obtained, the board will respond.

Presently, only two types of memory boards have been designed --a 4K random access memory utilizing the static Fairchild 2102 memory chips each consisting of 1K x 1 bits, and a 4K programmable read only memory (PROM) utilizing the Fairchild 93446 field programmable memory chip. The 93446 chip is organized as 512 words by 4 bits each word.

# BC.2 4K RAM BOARD (Assy No. 97382004)

# BC.2.1 Functional Description

Looking at the block diagram of the RAM board we recognize the following five different areas of circuitry: the memory array, the address comparator and row decoder, the array address buffering, the RAM.WRITE buffering, and the DATA.IN, DATA.OUT buffering.

The memory array will be discussed in detail later.

At the bottom left of the block diagram is the 4-bit comparator which compares addresses Al2 through Al5 with the setting of the page select switch bank. The output of the comparator will be high whenever a match of addresses and switch setting has been obtained. That signal is then inverted and used to enable the row decoder which is shown in coordinates C6 and it is also driven through a hex inverter onto the bus as indicated by the signal PAGE.SEL\*.

The row decoder, when enabled, decodes address bits A10 and A11 in order to generate the chip select for the four rows of memory devices in the memory array. If the switch setting does not match A12 - A15, the decoder will not be enabled and, therefore, none of the rows of the memory array will be selected. Thus, the memory board will not be active.

Address bits A0 through A9 are buffered from the bus and are distributed uniformly to the address inputs of all the chips in the array.

Similarly, the RAM.WRITE\* signal is buffered from the bus and is also applied to every chip on the array. The function of the RAM.WRITE\* signal is to enable writing into the memory chips.

On the base of the F8 FORMULATOR there is a memory DATA IN bus and a memory DATA OUT bus. The reference here is with respect to the memory; that is, the DATA IN bus is inputted to the memory for writing, the RAM DATA OUT bus is outputted from the memory to the processor. The RAM DATA IN bus consists of eight lines. They are buffered from the F8 FORMULATOR but and each line is connected to one of the eight columns in the memory array. The RAM DATA OUT bus consists of eight lines, each line being the output of a column of memory chips. The RAM data out bus is buffered onto the F8 FORMULATOR DATA OUT bus; however, it is gated onto the bus with the PAGE.SEL\* signal. Therefore, no data will be allowed to exit a particular memory board unless the page select signal is activated.

## BC.2.2 Detailed Schematic Description

Looking at the lower left-hand side corner of the detail schematic of the RAM board we find ten hex inverters whose purpose is to invert and isolate the ten address bits, addresses 0 through 9, which, in turn, are applied to the memory array as discussed earlier. At the bottom middle of the schematic we find a set of 9LS03 gates which drive the DATA OUT bus. Also, in that same area, is a set of eight hex inverters which buffer the data in bus to the memory. It should be mentioned here that both the address and data busses are inverted on the F8 FORMULATOR mother board. For that reason, they are inverted again, both going into the board and coming out of the board.

The center of the schematic is occupied by the 2102 memory array. As indicated there, the 2102 requires only reference ground and +5 volts. It has ten address bits required to address the 1K bits of data in the memory. It has a DATA IN bit for the data coming into the memory and a DATA OUT bit going out of the memory device. It also has a chip select line which is used to activate the chip. The chip select lines are generated by the 9LS139 decoder which is located in the upper right-hand corner of the schematic. decoder is a 1-of-4 decoder and it decodes address bits 10 and 11 as they are presented from the base board. decoder is enabled by the signal called B.PAGE.SEL\*. The outputs of the decoder are indicated as B.CHIP.SELO\*, 1\*, 2\*, and 3\*. Each ship select is applied to one row in the array and they are also brought up to test points at the top connector of the board for testing purposes. Note that the chip select signals are labeled inversely with respect to the 9LS 139 outputs. This is because address bits A10 and A11 are inverted on the F8 FORMULATOR base board.

The page select signal is driven onto the bus through a 9LS05 hex inverter because the processor board of the F8 FORMULATOR needs to monitor that signal. When page select is active and a CPU READ cycle is in process, the processor board allows the READ DATA bus to be gated onto the 3850 CPU chip. It should be noted here that the 9LS05 gate driving the page select signal is an open collector device and, therefore, it requires a pull-up. The pull-up for that signal is located on the processor board. Similarly, the 9LS03 devices driving the READ DATA bus out of the memory board are open collector devices. The pull-ups for those signals are, again, located on the processor board. Therefore, when attempting to test the RAM board all by itself, one should take into consideration the missing pull-ups and provide for them. The comparator, which compares address bits 12 through 15 with the setting of the page selection switch, is implemented by using the 9LS266

quad exclusive NOR gate which is open collector. The output of that comparator is a signal called B.PAGE.SEL and that signal is also brought to a test point at the top connector of the board.

Each of the poles of the page selection switch is brought up to a test point at the top connector of the board for testing purposes.

The RAM.WRITE\* signal, which appears at the lower right-hand corner of the schematic, is the signal necessary to notify the memory board whether the CPU wishes to do a read or a write operation. In case of a write operation, the signal coming into pin 46 of the board is in the low state. That signal is then inverted twice for buffering purposes and, thereafter, applied to the memory array.

In order to provide the user with the capability of protecting a page of random access memory from being written, a toggle switch has been inserted in series with that signal and the toggle switch is physically located at the top of the board. Thus, when the switch is in the ON position, the signal is connected through and responds to commands from the processor board as to whether it should be high or low. However, if the switch is in the OFF position, the signal is always high because of the pull-up resistor on that line and, therefore, no data write operations will be permitted on this particular memory board until the switch is turned back to the ON position.

# BC.3 PROM BOARD (Assy No. 97384001)

# BC.3.1 Functional Description

The 4K PROM board utilizes the 93446 electrically programmable read only memory. The block diagram of the PROM board shows five major sections. Section 1 is the PROM array cell; Section 2, in the lower left-hand corner, is the page selection mechanism which consists of a comparator and a page select switch which operate in a manner similar to the RAM board; Section 3 is the row decoder which decodes address bits 9, 10, and 11, and turns those into chip selects 0 through 7 (note that the chip selects here are applied to the eight columns); Section 4 is the buffering of address bits A0 through A8; and, Section 5, at the top middle of the diagram, is that of the DATA OUT buffering which is gated with signal PAGE.SEL\*.

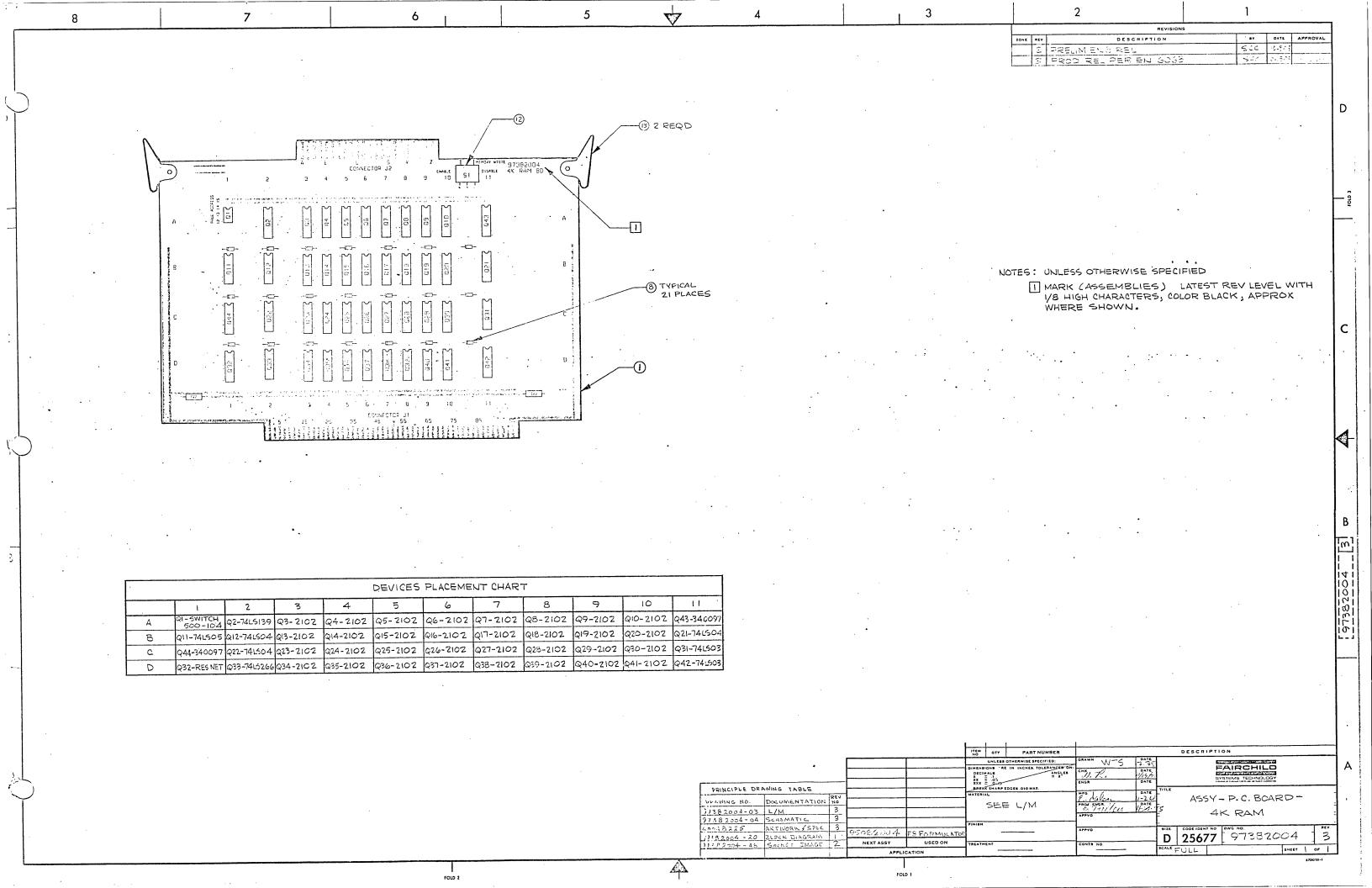
# BC.3.2 <u>Detailed Schematic Description</u>

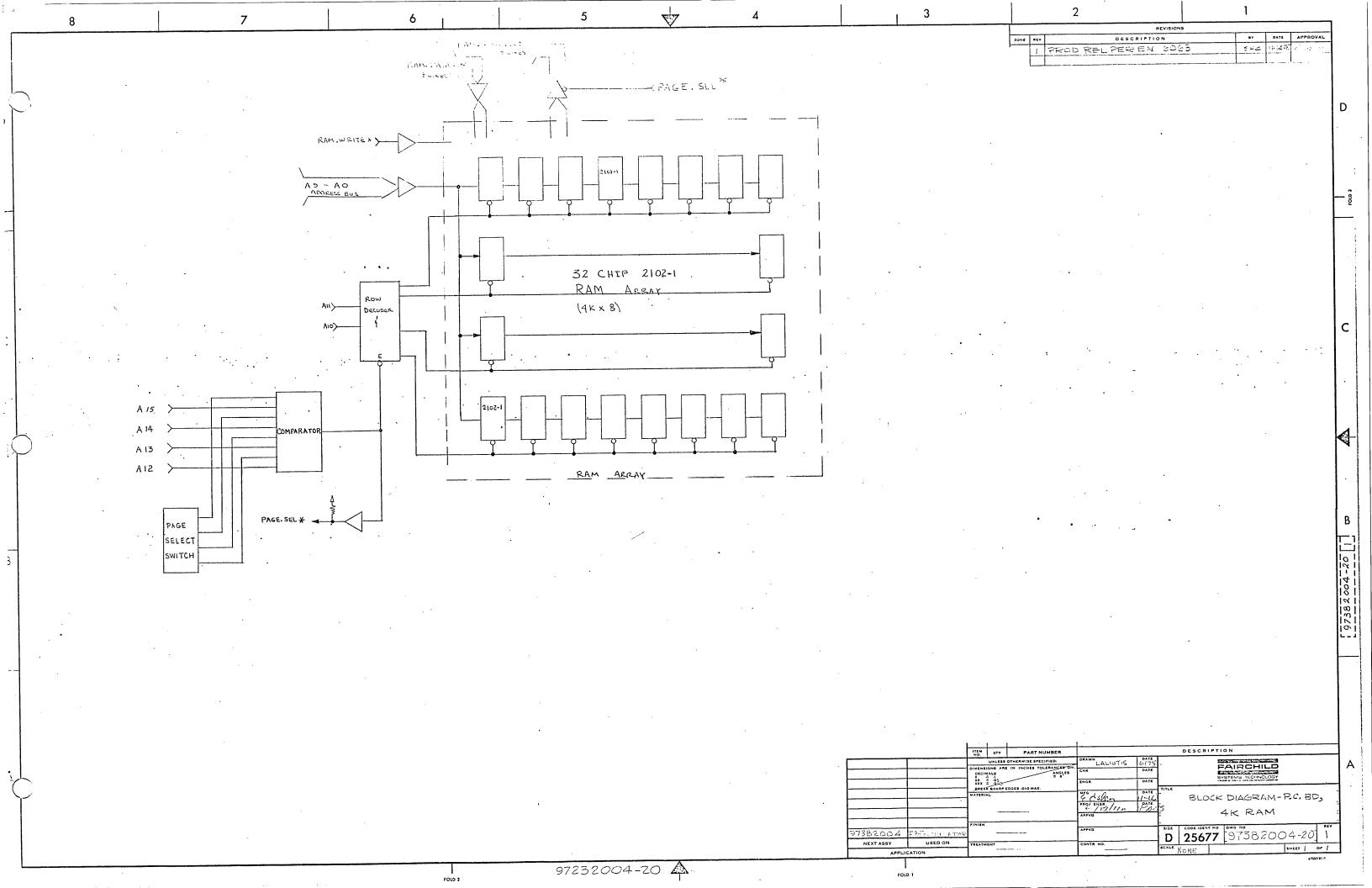
Looking at the detail schematic of the PROM board we see that the address lines, ADR(00-08)\*, and the DATA OUT lines.

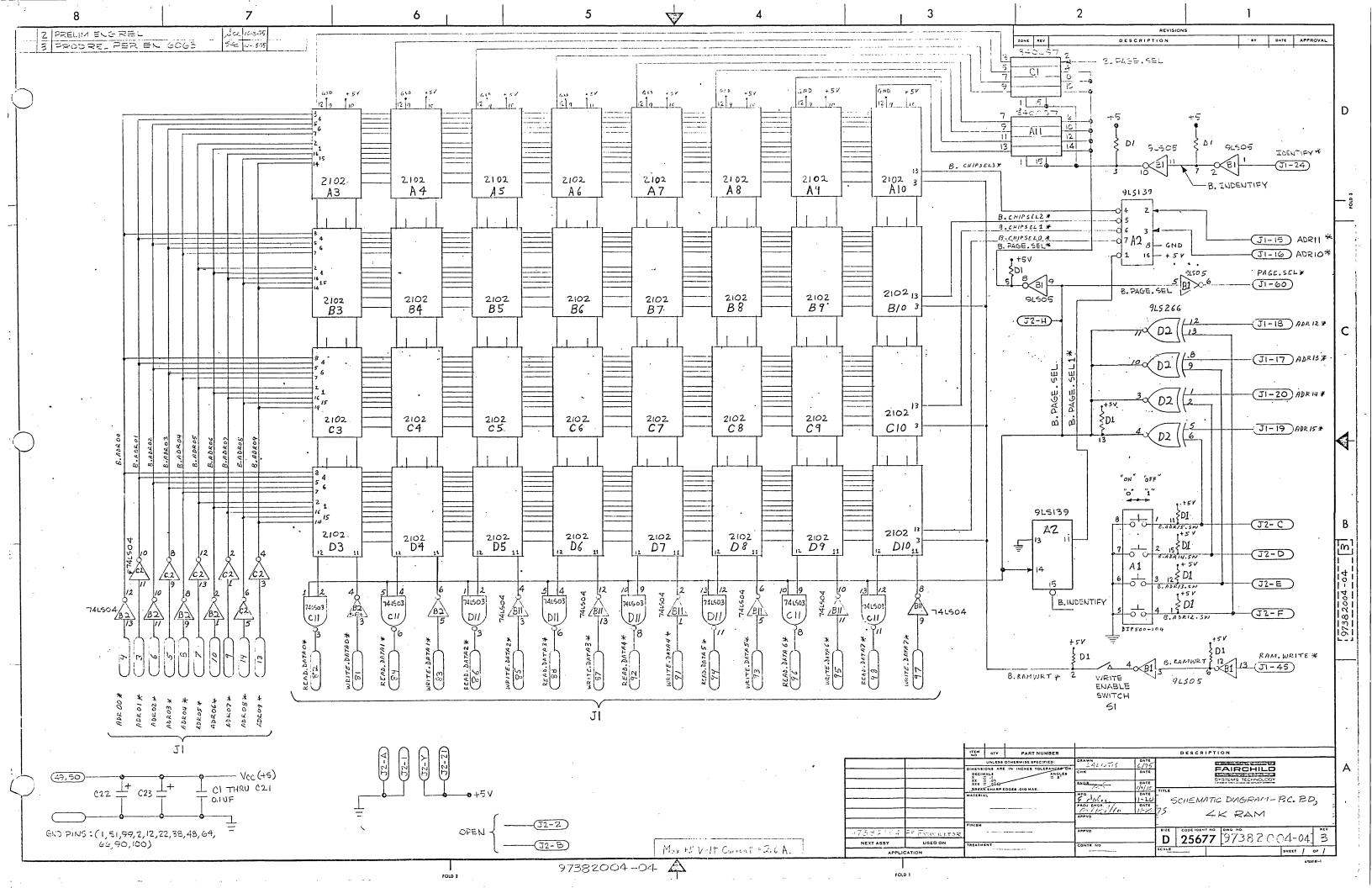
READ.DATA(0-7)\*, are buffered in a manner similar to the 4K RAM board. The page selection mechanism at the lower left-hand corner of the schematic is, again, identical to that of the RAM board.

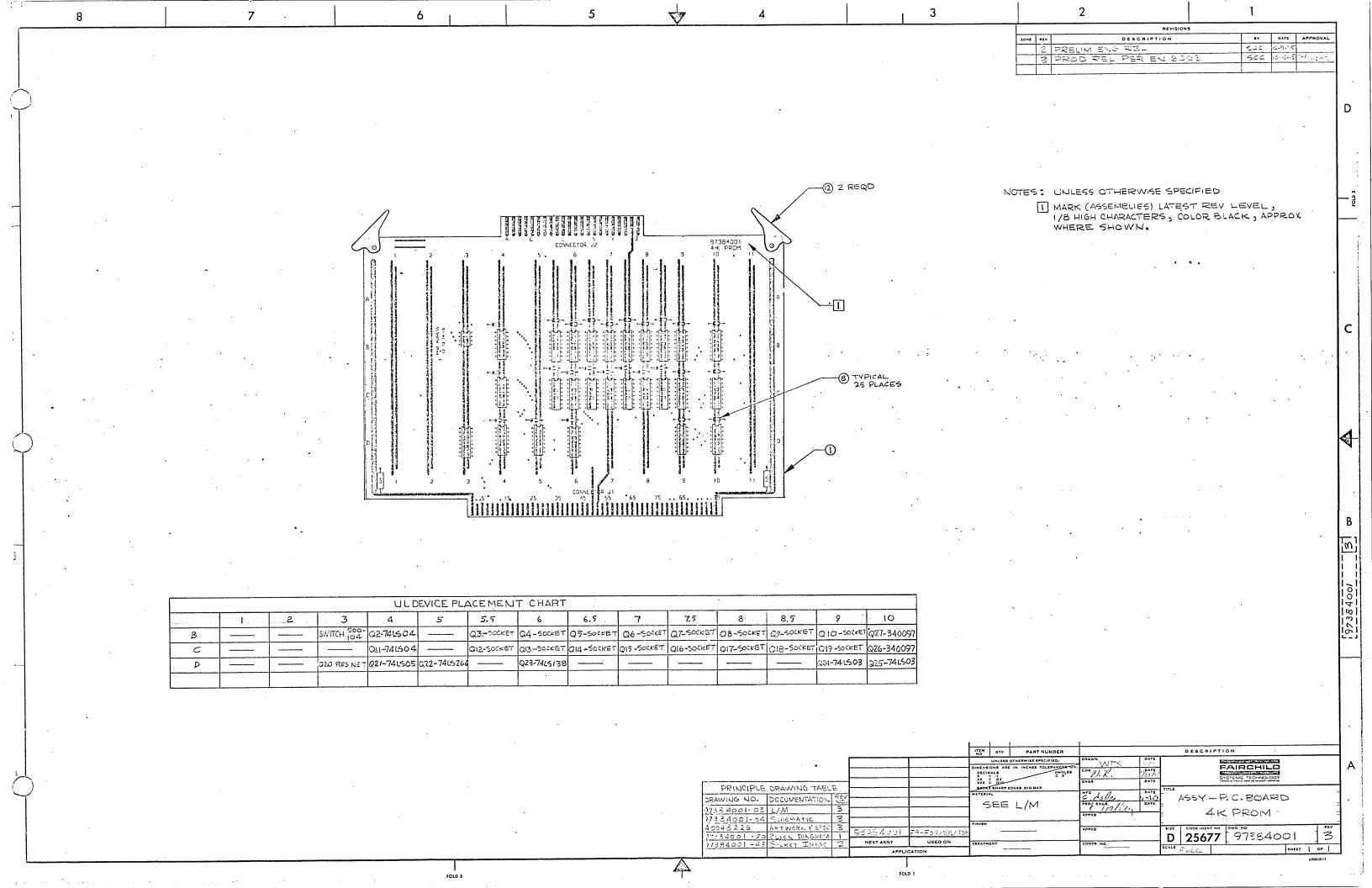
The memory array shown in the upper right corner of the schematic is organized in two rows by eight columns programmable read only memory device used is the 93446 which requires a ground reference and a +5V bias voltage. The 93446 is a 512-word by 4-bit memory device. It requires nine address bits and it provides four data bits out of each memory chip. A chip select signal is required on pin 13 of each memory device. The chip select signals are generated by the 9LS138 decoder which is located in the upper left corner of the schematic. The 9LS138 is a 1-of-10 decoder which gets enabled by the signal called C.PAGE.SEL and it decodes address bits 9, 10, and 11. Each chip select signal is applied simulataneously to a column of two memory devices. Again, note that the chip select signals are labeled inversely with respect to the 9LS138 outputs for the same reason given for the RAM board.

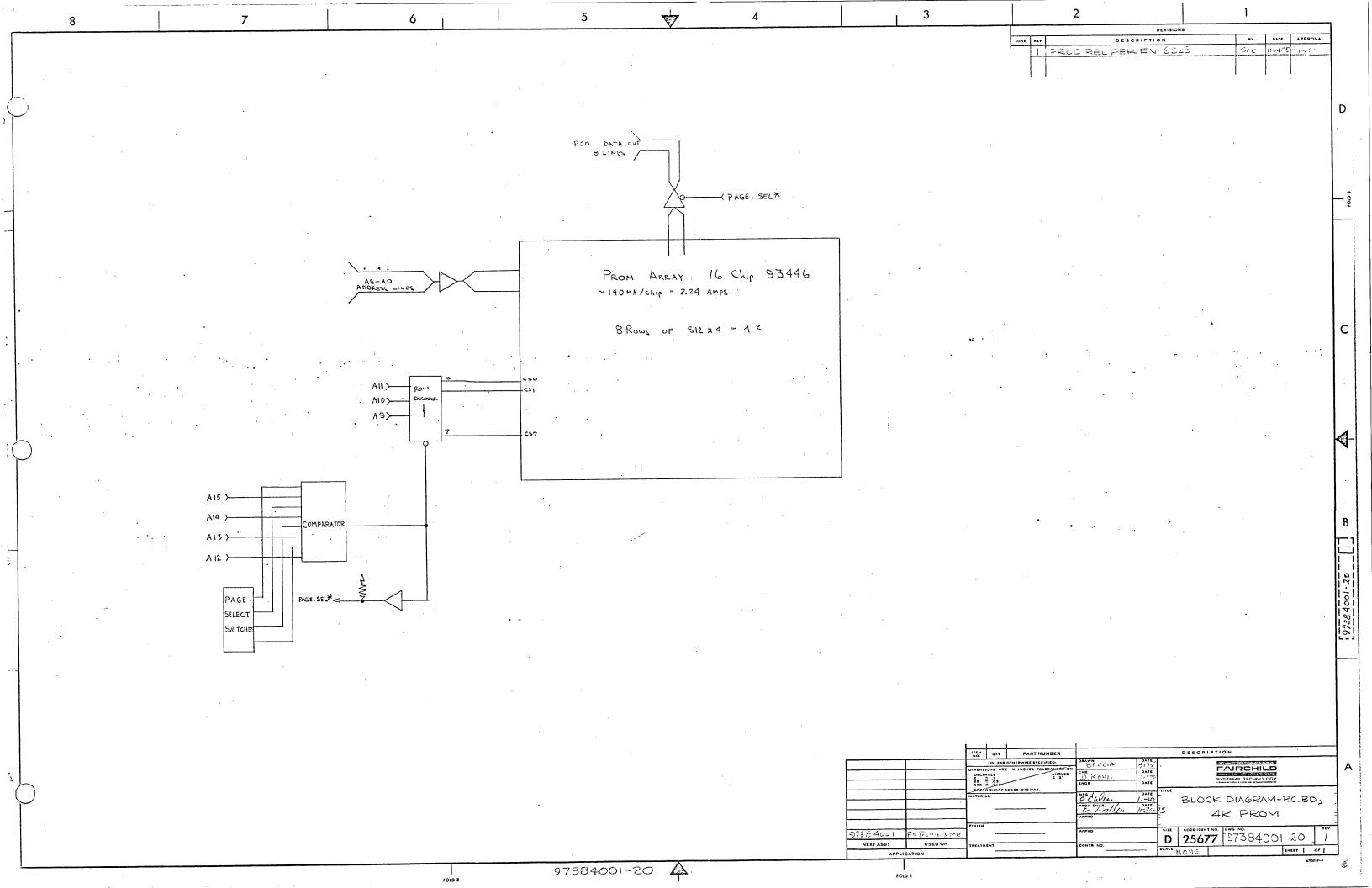
The 93446 memory device is electrically programmable and can be programmed by the user through the Fairchild PROM burner which can be purchased separately as an option with the F8 FORMULATOR.

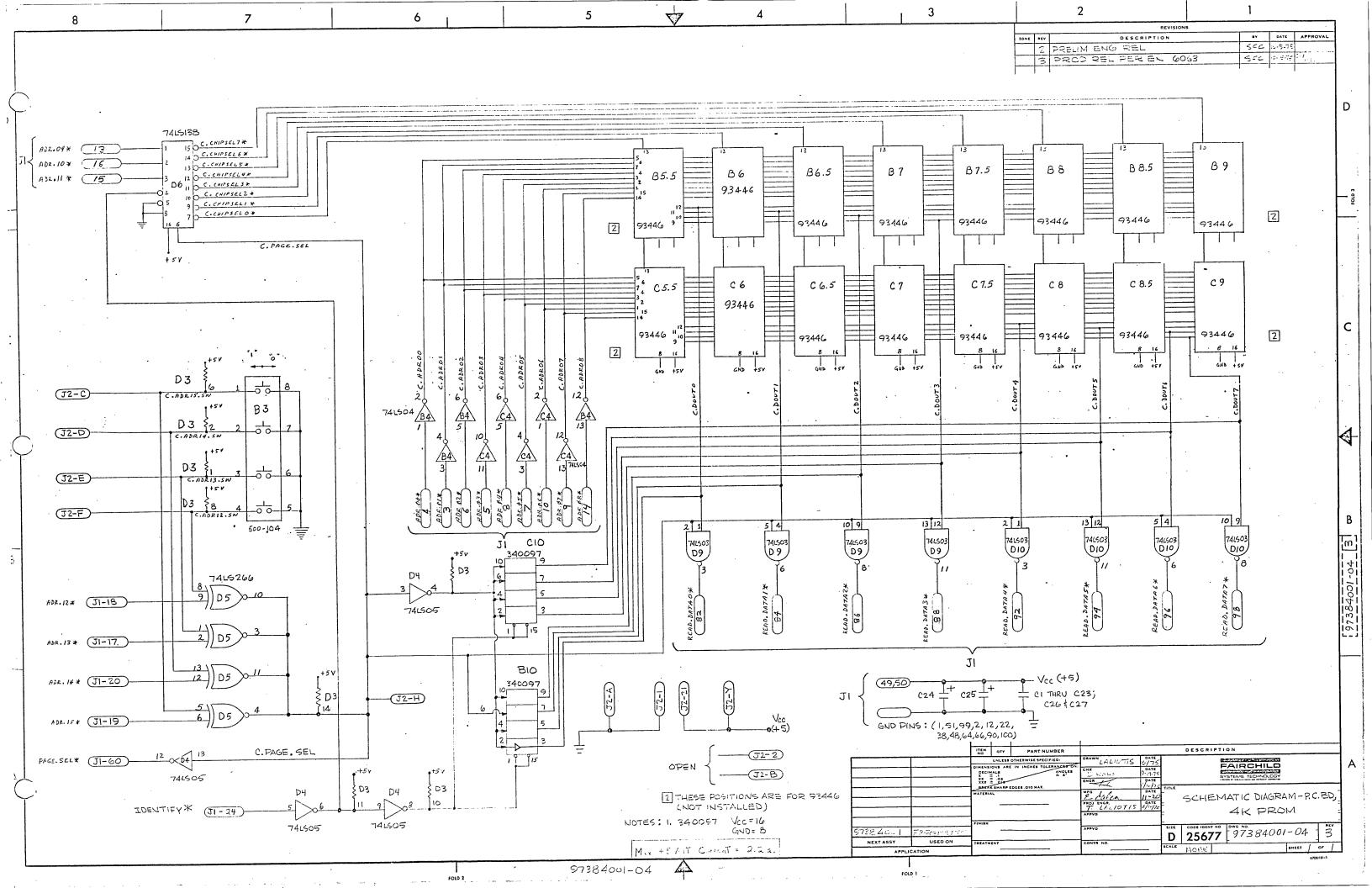












# Theory of Operation: COMMUNICATIONS BOARD (Assembly No. 97385001)

### E.1 INTRODUCTION

The communications board (COM BD) is designed to interface any one of three types of serial devices to the F8 FORMULATOR base board.

Type 1: Standard TTY current LOOP with READER CONTROL;

Type 2: Any TTL-compatible serial device;

Type 3: RS232 Modem.

Thirteen standard baud rates of 50 to 9600 are available through switch selections on the COM BD.

The board is designed using the TR1602 UART chip with additional hardware to make it appear as a USART operating in the asynchronous mode. This allows the Supervisor software to be written so that no changes will be required when the USART board becomes available.

### E.2 MODES OF OPERATION

The two I/O ports of a 3851 ROM chip are used as the direct interface to the UART. The 3851 is used only as an I/O device and a timer interrupt. Refer to 3851 Theory of Operations for details.

# E.2.1 Identify Mode

During system initialization, each I/O type board in the F8 FORMULATOR is querried as to its type via the identify logic. This is done by scanning through all I/O addresses with input instructions while holding the identify control line active.

Each I/O board has a switch-selectable block, or blocks, of I/O addresses which will allow that board to respond when its addresses are referenced. Normally the PORT data would be read into the CPU but, with the identify active, switch data is substituted to identify a unique device assigned to that PORT. The overall effect is automatic system configuration each time the system is initialized.

The identify word format for a COM BD is

7	,	6	5	4	3	2	1	0
1	-	1	1	1	1	N	N	N

where NNN specifies 1 of 7 types of serial devices.

(NNN = 111 is a forbidden code.)

### E.2.2 Interrupt Mode

Ordinarily, interrupt vectors are premasked onto the 3851 chip so that, if all other interrupt conditions are met, a predefined interrupt address is returned to the CPU during an interrupt fetch. On the COM BD, however, the high-order eight bits of the interrupt address is a switch-selectable substitute and the low-order eight bits are forced to zero except for Bit 7 which is used to specify "1", external interrupt, or "0", internal (TIMER) interrupt.

### E.2.3 I/O Mode

Normal Input/Output operation occurs on the COM BD when it senses its I/O address selection followed by an input or output code on the ROMC bus. Refer to the COM BD block diagram for a complete description of the operational sequences and the ROMC bus codes.

# E.3 COMMUNICATIONS BOARD OPERATIONS

As viewed by the CPU, each COM BD has a block of four I/O addresses assigned as follows:

7	6	5	4	3	2	1	0	
S	S	S	S	S	S	N	N	

where S---S specifies the switch settings for board response and NN specifies which port on the board.

 $^{\mathrm{N}}1^{\mathrm{N}}0$ 

00  $\Rightarrow$  Bidirectional data port to UART (/00)

01  $\Rightarrow$  Bidirectional control port to UART (/01)

10  $\Rightarrow$  Interrupt control port in the 3851 (/02)

11  $\Rightarrow$  Timer in the 3851 (/03)

Table 1 lists the control word formats and the response of the UART (USART) when the words are outputted to PORT /01.

Table 2 lists the word formats of the data port /00 during the control operations.

# E.4 SWITCH IDENTIFICATION TABLE

The tables on the following pages describe the locations and functions of all the switches on the COM BD.

# E.5 FUNCTIONAL OPERATION

The attached document is a functional description of the communications board from the software standpoint.

## Baud Rate Switches

Location 8A

Switch S3 X = Closed Section

		-		
4	3	2	1	Baud
				110
			Х	150
		X		300
		Х	Х	2400
	X			1200
	Х		Х	1800
	Х	Х		4800
	Х	Х	Х	9600
X				2400
X			Х	600
Х		X		200
Х		Х	Х	134.5
X	Х			75
X	Х		Х	50
Х	Х	Х		Zero
Х	Х	Х	Х	Zero

# I/O Port Selection Switches

Location 7C

	Location	70	
Switch Section		IO.DAT	bit
8	matches	7	MSB
7	matches	6	
/	matches	U	
6	matches	5	
Ū			
5	matches	4	
	. •	^	
4	matches	3	
· 3	matches	2	
J	IIIG CCIICO	-	

A closed switch matching a "1" on the IO.DAT bus will result in an enable for that bit. All six bits must match before the 3851 comparison register will be properly enabled.

# Interrupt Vector Switches

-			~~
Loc	201	 An.	6C
	.aı	 $o_{II}$	UU

Switch Sectio		IO.DAT bit	t
8	matches	7	
7	matches	6	
6	matches	5	
5	matches	4	
4	matches	3	
3	matches	2	
2	matches	1	
1	matches	0	

An open switch will result in a "l" on the corresponding IO.DAT bit at the time the high order interrupt vector is being fetched from the communications board.

# Identify Switches

Т	00	2	ن -	io	'n	٠.	Q	Δ	

Switch	S3	IO,DAT	bit
7	matches	2	
6	matches	1	
5	matches	0	

# COM.BD. Identify Work Format

IO.DAT bit	7	6	5	4	3	2	Τ	U
	1	1	1	1	1	С	В	Α

where A = Section 7

B = Section 6

C = Section 5

An open switch will result in a "1" in the corresponding IO.DAT bit.

# Miscellaneous

Switch S1-1 at Location 7C should be closed to establish interrupt priority when this I/O type board is electrically farthest (highest priority) from the processor board.

# COMMUNICATIONS BOARD CONTROL WORD FORMATS

 RESET	CONTROL	READ	WRITE	AUX, CON.	REQ.SEND	DTR	READER.ON
 7	6	5	4	3	2	1	0

OPERATION				V	WORD FORMAT				REMARKS
1	EXTERNAL MASTER	1	x	x	x	X	NOTE	1	ALL REGISTERS RESET AND CONTROLS INITIALIZED
1A	RESET	0	X	X	X	X			
2	LOAD MODE	0	1	0	1	0	NOTE		LOADS DATA PORT INTO MODE CONTROL REGISTER
2A	WORD	0	1	0	0	0			
3	LOAD COMMAND	0	1	0	1	1	NOTE	1	LOADS DATA PORT INTO COMMAND REGISTER IN USART
3A	WORD	0	. 1	0	0	1			THIS WORD "FLUSHED" BY UART
4	LOAD DATA	0	0	0	1	0	NOTE	1	LOADS DATA PORT INTO XMIT BUFFER REGISTER AND
4A	WORD	0	0	0	0	0			INITIATES SERIAL OUTPUT.
5	READ DATA WORD NOTE 2.	0	0	1	0	0	NOTE	1	PLACES ASSEMBLED CHARACTER ON DATA PORT
6	READ STATUS 1	0	1	1	0	0	NOTE	1	PLACES STATUS WORD 1 ON DATA PORT.
7	READ STATUS 2	0	1	1	0	1	NOTE	1	PLACES STATUS WORD 2 ON DATA PORT

- NOTE 1: These functions are unique to the UART board and will be exercised by user software. Internal software should sense these bits via an input from port /02 and ensure they remain unchanged except during system initialization.
- NOTE 2: After the character has been read from the data port, bit 5 should be reset in the control port. If not, the next characters to be received will be lost because DATA AVAIL will be forced continuously reset by the read bit.

TABLE 2

COMMUNICATIONS BOARD DATA WORD FORMATS

MODE WORD

BIT	7	S2	NA				
	6	S1	0 = 1 STOP BIT 1 = 2 STOP BITS				
	5	EP	<pre>0 = ODD PARITY CHECK/GEN. 1 = EVEN PARITY CHECK/GEN.</pre>				
	4	PEN	NOTE 1				
	3	L2	L1 L2 0 0 = 5 BITS/CHARACTER 0 1 = 6				
	2	L1	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$				
	1	B2 = "1"	NO SIGNIFICANCE ON UART BOARD. CAUSES USART BOARD TO SELECT				
BIT	.0	B1 = "0"	PROPER CLOCK RATE (16XBAUD RATE).				

NOTE 1: On the USART Board:
"1" = Parity enable.
"0" = Parity disable.

On the UART Board, parity is permanently disabled.

TABLE 2

COMMUNICATIONS BOARD DATA WORD FORMATS

COMMAND WORD

BIT	7		
	6		
	5	N.A. FOR UART BOARD	
	4		
	3		
	2		
	1	"1" = ENABLE RCVR "0" =	DISABLE RCVR
BIT	0	"1" = ENABLE XMTR "0" =	DISABLE XMTR

NOTE: This word is "flushed" by UART Board but Bits 0 and 1 should be set to "1" for board compatibility.

TABLE 2

COMMUNICATIONS BOARD DATA WORD FORMATS

DATA WORD

BIT	7	MSB,	LAST	OUT/IN
•	6			
	5			
	4			
	3			
	-2			
	. 1			
BIT	0	LSB,	FIRS'	r out/in

TABLE 2

COMMUNICATIONS BOARD DATA WORD FORMATS

STATUS WORD 1

BIT	7	NA
	6	NA
	5	FRAMING ERROR 1 = ERROR
	4	OVERRUN ERROR 1 = ERROR
	3	PARITY ERROR 1 = ERROR
	2	EOC 1 = SHIFT REGISTER EMPTY
	-1	DA 1 = DATA AVAILABLE (CHARACTER ASSEMBLED)
BIT	0	TBMT 1 = TRANSMITTER BUFFER REGISTER EMPTY

TABLE 2

COMMUNICATIONS BOARD DATA WORD FORMATS

STATUS WORD 2

BIT	7	CAR.DET	RS232 CARRIER DETECTED (UART ONLY)
	6	CTS	RS232 CLEAR TO SEND
	5	DSR	RS232 DATA SET READY (UART ONLY)
	4	RING	RS232 RING INDICATOR (UART ONLY)
	3		
	2	NA	
	.1		
BIT	0		

# COMMUNICATIONS BOARD Functional Operation

#### 1.0 GENERAL

The communications board provides serial communications hardware utilizing USART or UART control devices. Usage is typically control of a teletype, RS232 modem, or other terminal device types. Teletype control is via standard 20mA current loop with reader relay control.

#### 2.0 BAUD RATE

Transmission speed is switch-selectable for any one of thirteen standard baud rates. These include basic 110-baud teletype, 300-baud teletype replacement, and 2400-baud high-speed terminal.

### 3.0 I/O PORT SELECTION

The I/O addresses to which the communication board will respond are switch-selectable. The four ports provided on the board are contiguously assigned. These are:

MSB							LSB		
	S	S	S	S	S	S	0	0	- Data port
	S	S	S	S	S	S	0	1	- Command port
	S	S	S	S	S	S	1	0	- Interrupt control port
	S	S	S	S	S	S	1	1	- Timer port
	whe	re	'SS	SSS	s'	is	det	erm	ined by switch settings.

#### 3.1 Data Port

All transmissions of data to or from the USART/UART are passed in the data port. This data is forced to or from various registers within the USART/UART device via control lines provided at the command port (Section 3.2). The data port is an eight-bit bi-directional latched port. Output latches must be cleared (zero) through an output instruction prior to an input from this port.

#### 3.2 Command Port

Information placed in the data port (Section 3.1) is directed by the command port. Data from the USART/UART device may be read immediately once the command port lines

are set. Data issued to the USART/UART must be strobed by setting the command port output strobe line to a one, then to a zero. Refer to Table 1 for USART/UART internal registers and the control lines which are forced to access each register.

Three individual lines are provided for device control. These include:

Request send - RS232 control

Data terminal ready - RS232 control

Reader on - Teletype reader on (1), off (0)

# 3.3 Interrupt Control Port

This is the physical interrupt control port of an F8 3851 ROM chip. Refer to F8 documentation for definitions and operation.

### 3.4 Timer Port

This is the physical timer port of an F8 3851 ROM chip. Refer to F8 documentation for definitions and operation.

#### 4.0 INTERRUPTS

Interrupts may be produced by either the F8 3851 ROM timer or an external interrupt network. The external interrupting conditions are noted in Table 1. Interrupt addresses are switch-selectable as indicated.

MSB

S S S S S S S S X 0 0 0 0 0 0

where - 'SSSSSSS' is switch-selected

- X is 1 for external interrupt

O for timer interrupt

OUT

# 5.0 INITIALIZATION SEQUENCE

A typical assembly language sequence is shown for communications board initialization. Constants are shown expanded for clarity.

TDAT	EQU	X	Communications	data por	rt			
TCTL	EQU	Y	Communications	control	port			
*	Reset commun	ications b	oard					
	LI	B'1000000	0'					
	OUT	TCTL						
	LI	B'0000000	0'					
	OUT	TCTL		,				
*	Set mode con	trol	$\sim$					
•	LI	B'1110111	0'					
•	OUT	TDAT						
	LI	B'0101000	0'					
	OUT	TCTL			,			
	LI	B'0100000	0'					
	OUT	TCTL						
*	Set command control							
	LI	B'0000001	1'					
	OUT	TDAT						
	LI	B'0101100	0'					
	OUT	TCTL						
	LI	B'0100100	0'					

TCTL

# 6.0 COMMUNICATIONS BOARD OUTPUT

A typical assembly language sequence is shown for communications board output operating as a polled device. Constants are shown expanded for clarity.

\* Clear data port

LI

B'00000000'

OUT

TDAT

\* Read status for transmitter buffer empty

LI

B'01100000'

OUT

TCTL

LOOP

IN

TDAT

NI

B'0000001'

BZ

LOOP

\*

Reset status read command

LI

B'00000000'

OUT

TCTL

\* Set output data to transmitter buffer

LI

DATA

OUT

TDAT

LI

B'00010000'

OUT

TCTL

LI

B'00000000'

OUT

TCTL

\* Clear data port

LI

B'00000000'

OUT

TCTL

## 7.0 COMMUNICATIONS BOARD INPUT

A typical assembly language sequence is shown for communications board input operating as a polled device. Constants are shown expanded for clarity.

\* Clear data port

LI

B'00000000'

OUT

TDAT

\* Read status word for character ready

 $\star$  Z = 1

Tape reader on

 $\star$  Z = 0

Tape reader off

LI

B'0110000Z'

OUT

TCTL

LOOP -

IN

TDAT

NI

B'00000010'

BZ

LOOP

\* Read character and store in RO

LI

B'00100000'

OUT

TCTL

IN

TDAT

LR

RO,A

\*

Clear input command

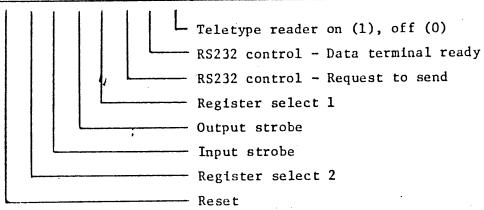
LI

B'00000000'

OUT

TCTL

Usage	м\$в		Comm	nand	l Po	ort		LSB	мѕв	•		Data	Port	·		LSB
Reset	1 0	0	0	0	0	0	0.	0		NA NA						
Set Mode	0	1 1	0	1 0	0	X X	Y Y	Z Z	Single Sync 1	Stop Bits 1	Parity State 1	Parity Enable <sup>0</sup>	Bits/cha	racter 1	Baud	0
Set Control	0	1	0	1 0	1	X X	Y Y	Z Z		NA Enable Enable RCVR 1 XMTR						· ·
Output Data	0	0	0	1 0	0	X X	Y Y	Z Z		Output Data						
Input Data	0	0	1	0	0	Х	Y	Z —	Input Data							
Input Status	0	1	1	0	0	Х	Y	Z 	NA	NA	Frame Error	Overrun Error	Parity V Error	Shift Register Empty	Char Ready	XMTR Empty
Input Status	0	1	1	0	1	Х	Y	Z	Carrier Detect	Clear to Send	Data Set Ready	Ring	NA	NA	NA	NA

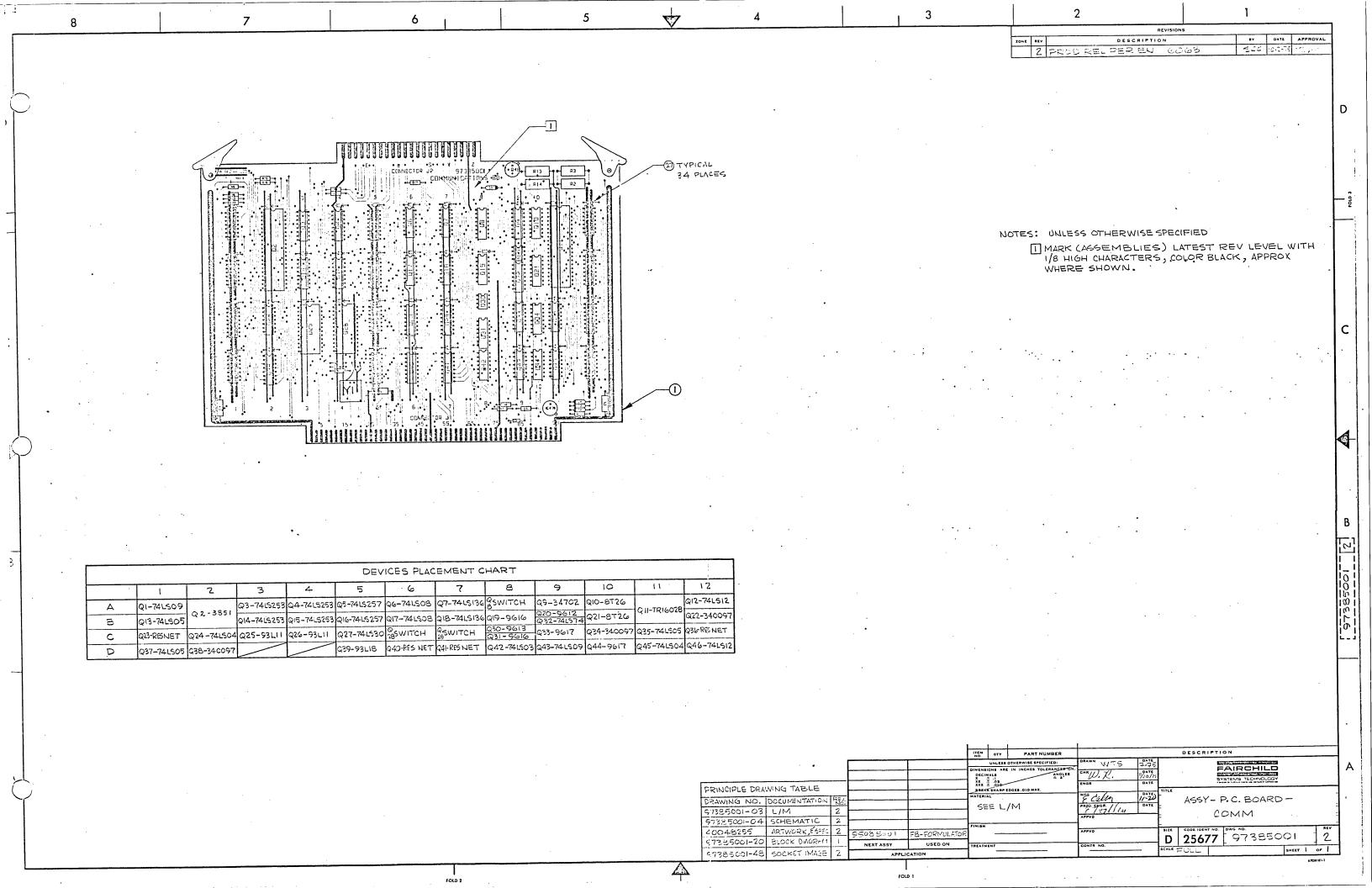


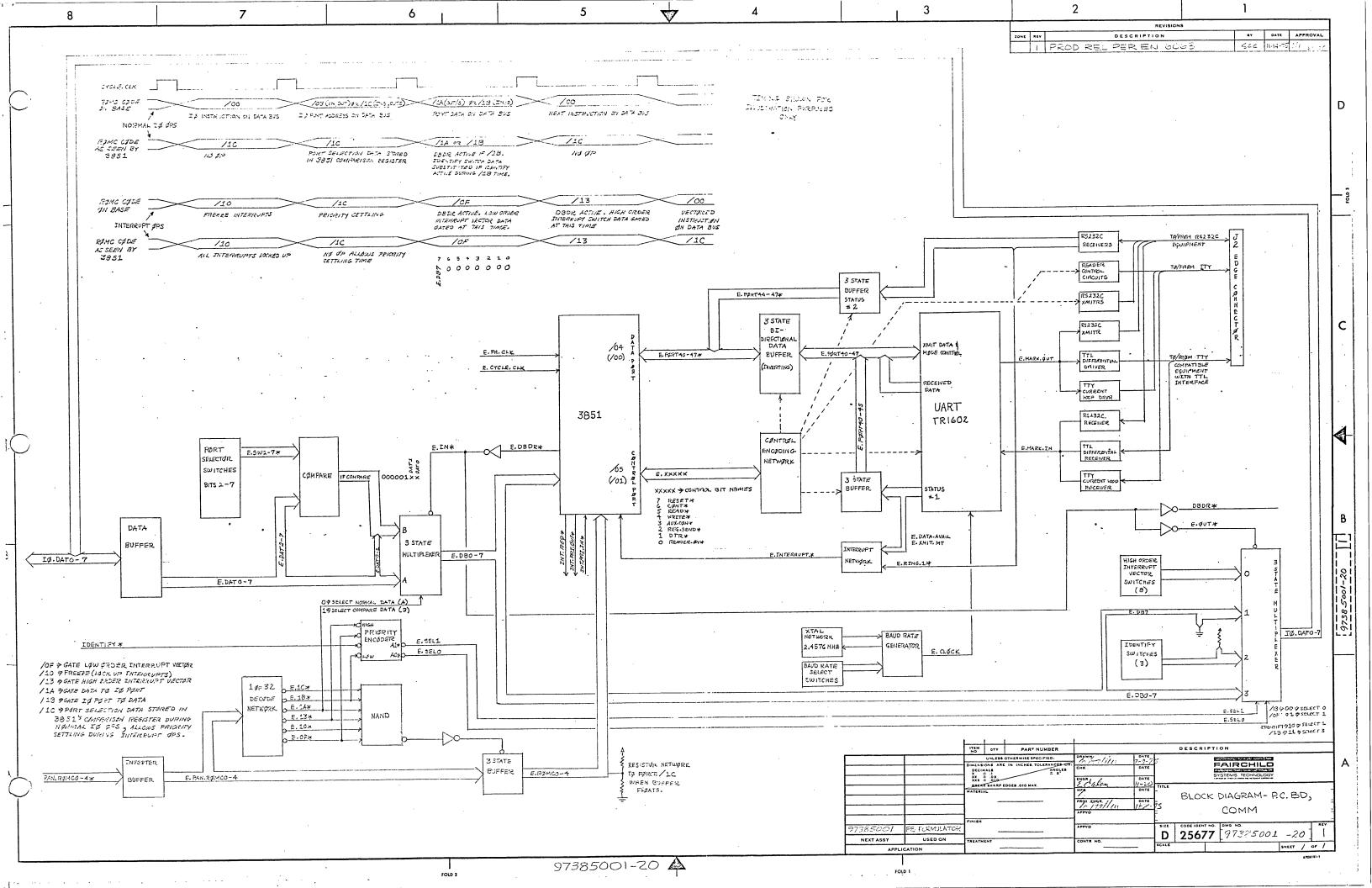
NOTES: v - Disabled on UART device board

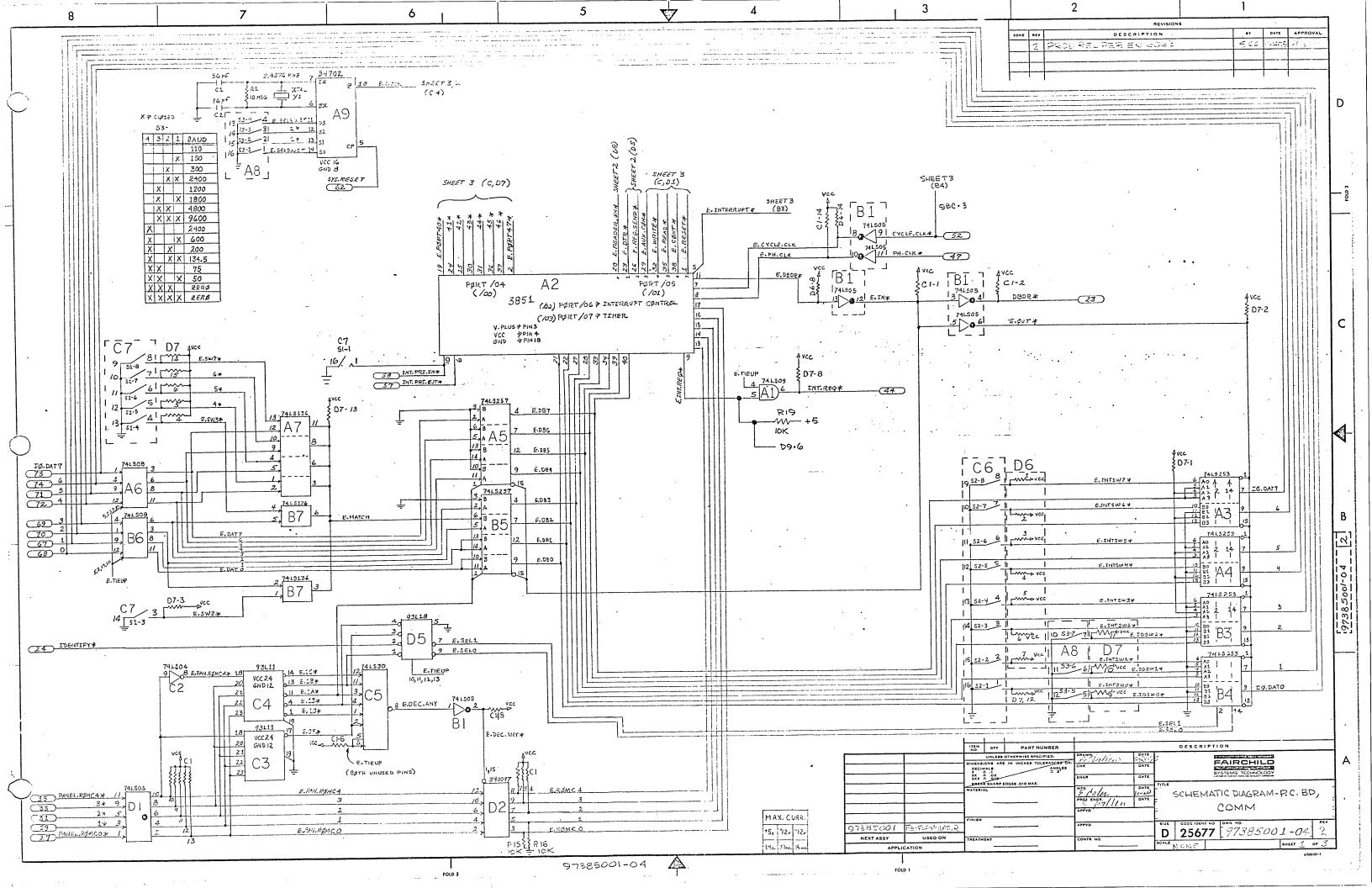
w - Causes interrupt condition

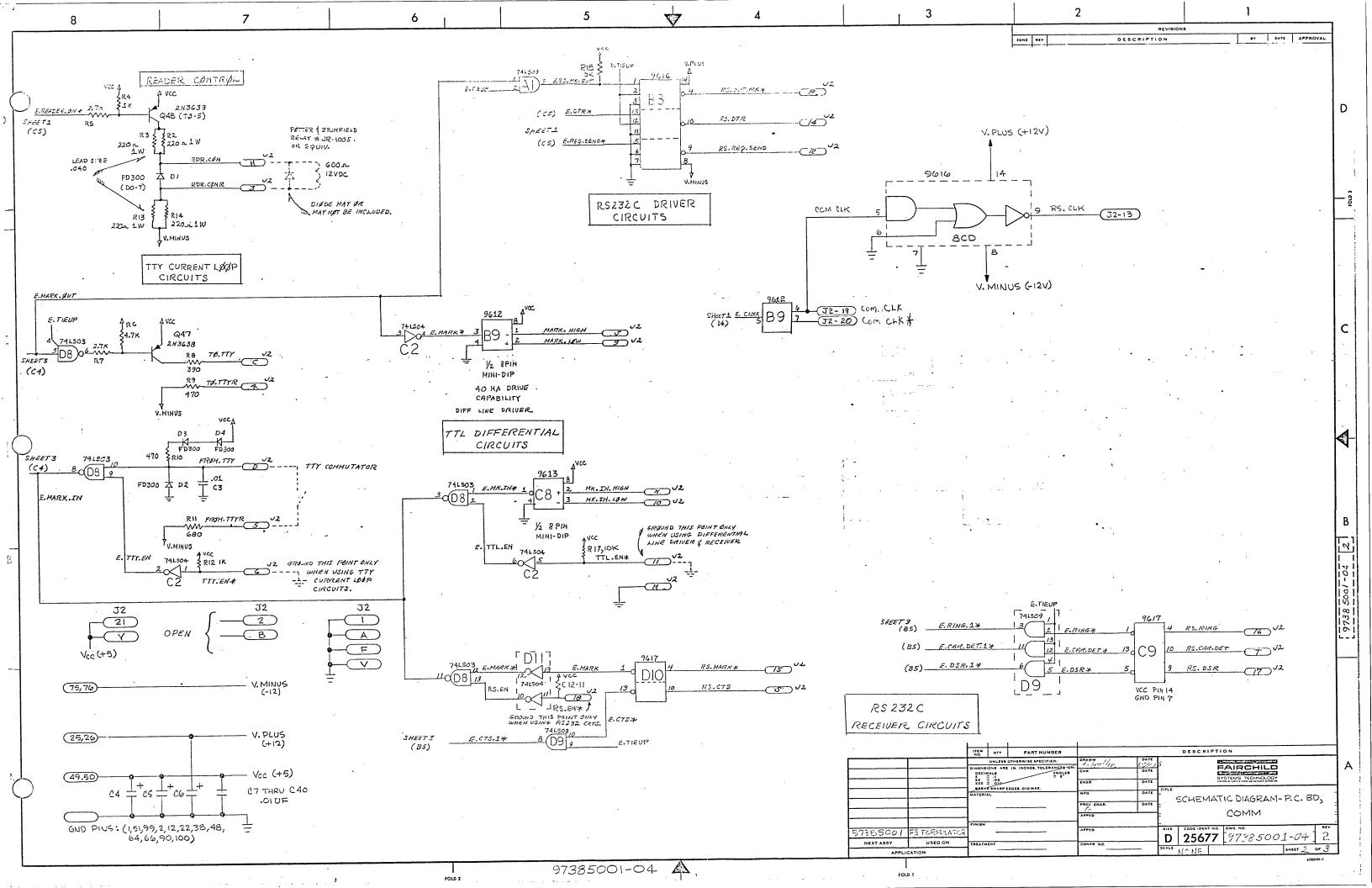
Refer to USART/UART descriptions for full control/error descriptions Numbers in lower right corner of frame indicate normal initialize value

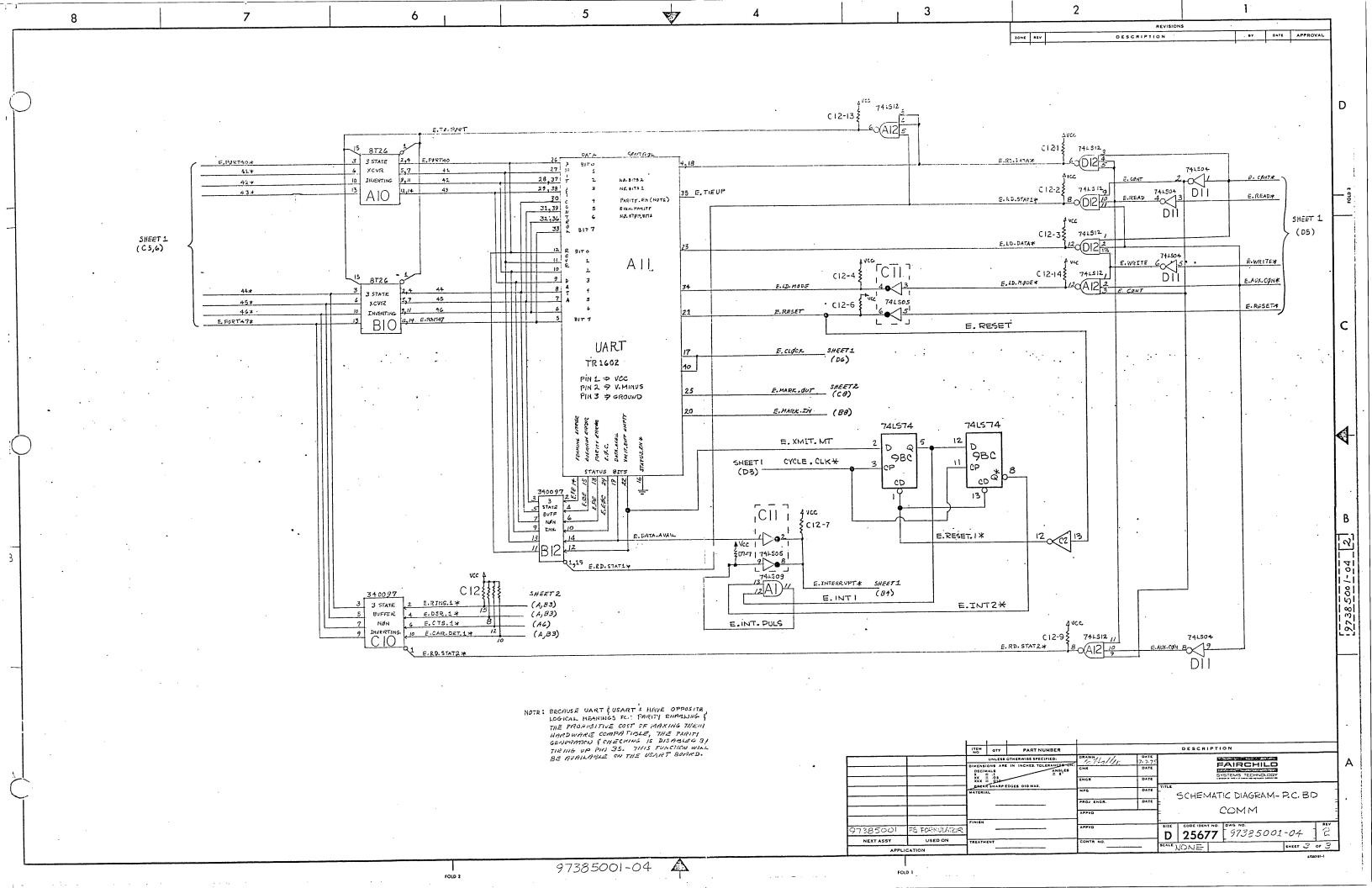
Table 1: Communications Board Control











Theory of Operation: QUAD I/O BOARD (Assembly No. 97385002)

## D.1 INTRODUCTION

The function of the quad I/O board is to provide the user with the capability of simulating the variable number of I/O ports he may want to build into his system. As such, the quad I/O board can be used as an interface between the processor of the F8C system and the outside world that the user may want to communicate data in and out of the processor. In order to be in a position to simulate a set of real F8 chips, the I/O functions must be simulated through the 3851 PSU. However, the PSU will be used here strictly as an I/O interface. Therefore, the only features of the PSU that will be used here will be the I/O port functions, the interrupt functions, and the interval timer functions. The other facilities, such as the program counter and the stack register will not be utilized at all.

The 3851 circuit in a real application is intended to have a masked I/O address specified by the user and also it is intended to hold interrupt vector addresses for external interrupts and interval timer interrupts. In order to achieve this, the quad I/O board is provided with banks of switches which the user can set up in order to provide the I/O addresses and also to provide the interrupt vector addresses.

## D.1.1 CPU - I/O Communication

The basic means of communication between the processor and the PSU is the ROMC lines and the data bus. The execution of each I/O command consists of a number of machine cycles where the processor issues a sequence of ROMC commands to the 3851 PSU and, at the same time, with each ROMC command, the I/O data bus contains certain data. During the first machine cycle of a typical I/O command, the processor will place a port selection address on the data bus and, at the same time, present a ROMC code related to the particular operation that it is presently performing. The second, and possibly the third, machine cycle will proceed similarly to the first one. The last machine cycle will consist of fetching the next instruction which is not related to the execution of the present instruction.

## D.1.2 Identify Mode

A feature built into the F8C system which is intended to give the operating system the capability of knowing what types of I/O devices are attached to the various I/O ports in the system is the so-called identify mode. The identify operation is nothing more than an input instruction being executed to a selected port with the IDENTIFY signal active. This signal, then, overrides the regular I/O instruction execution in the quad I/O board and it forces a device identification code to be placed on the data bus for the processor to pick up. In this manner the system software will make a complete scan and create a table for itself showing which I/O boards are active and what types of devices are on each port. The device identification code is to be set up by the user through a bank of switches in a manner similar to that used for setting up the I/O port addresses and the interrupt vector addresses.

# D.1.3 Summary

In summary, then, the functions of the quad I/O board are three:

- to allow the processor to pass data to and from the I/O devices attached to the ports;
- to provide the processor with the interrupt vector addresses that the user wishes to assign to the various ports; and
- 3. to provide the operating system with the device codes of the various types of I/O devices attached to the ports.

Thus, it is important to realize that the 1K of memory built into each 3851 PSU and the internal registers used for manipulating data and addresses are not used at all in the PSUs that are found on the quad I/O board.

# D.1.4 Ports

The quad I/O board contains two 3851 PSUs, thus providing two independently-addressed sets of dual 8-bit I/O ports that interface to the F8C base board. Each set is provided with independent interrupt capability and independent interrupt vectors that are switch-selectable by the user. Also provided, then, are two interval timers with interrupt capability for real time applications where the interrupt vectors of the timers are, again, switch-selectable by the user.

## D.2 FUNCTIONAL DESCRIPTION

## D.2.1 ROMC Selection Logic

Referring to the quad I/O board block diagram we see that the circuitry in the lower left-hand side corner is dedicated to the buffering, decoding, and encoding of the ROMC lines. After being inverted and buffered into the quad I/O board, the ROMC lines go to a 1-of-32 decode network and to a three-state The 1-of-32 decode network will decode all 32 combinations of the ROMC lines; however, only the ones that are meaningful to the quad I/O board will be utilized. Those meaningful codes are indicated at the left-hand side of the block diagram and also they are indicated at the output of the 1-of-32 decode network. If any one of those meaningful codes is presently on the ROMC bus, the NAND gate will force an enable signal to the three-state buffer shown in coordinates A6 to make the buffer active and, therefore, pass that meaningful ROMC code onto the 3851 PSUs. However, if the ROMC code presently on the bus is none of the quad I/O meaningful codes, then the NAND gate will force the three-state buffer to be in its high impedance state and the resistor network that is shown attached to the output of that three-state buffer will force the state of the ROMC lines presented to the 3851 PSUs to be always a /1C code which is the NO OP code as explained earlier. Therefore, unless a meaningful quad I/O operation is detected by the ROMC lines on the bus, the PSUs will see always the /1C code which will cause the PSUs to be in a NO OP state but stay prepared for a possibly meaningful ROMC code during the next machine cycle.

# D.2.2 Input and Interrupt Commands

During an input or interrupt type of command the quad I/O board needs to transmit the appropriate data or interrupt vector address back to the processor board. This is accomplished by taking three of the outputs of the 1-of-32 decode network which are related to an input or interrupt command, (codes OF, 13 and 1B), combining those three with the IDENTIFY signal into a priority encoder which is shown in coordinates B6. The output of that priority encoder consists of two signals called D.SEL1 and D.SEL0. The binary combination of those two signals is then used on the right-hand side of the block diagram in order to enable the appropriate multiplexers so that the appropriate data or interrupt vector address can be gated onto the bus by the 3851 as selected by the user in the switches.

# D.2.3 Output Commands

During an output operation, however, the quad I/O board must take the data from the processor data bus and pass it on to the appropriate I/O port. This is accomplished by the fact that the 1-of-32 decode network will see the code /lA. The output then of the 1-of-32 decode network that is indicated as D.lA\* is used as a multiplexer select input at the three-state multiplexers which are in the middle left of the block diagram in coordinates 6B and 6C. Thus, if the B.lA\* signal is active, it will be in the zero state and, as indicated there, the normal I/O data will be selected and passed through the three-state multiplexers on to the 3851 I/O data bus.

As long as D.1A\* is not active, that is, as long as it is a logic 1, the three-state multiplexers at coordinates 6B and 6C will be constantly transmitting onto the 3851 bus the outputs of the comparators that compare the data bus with the I/O port selector switches which are indicated in coordinates B7 and C7. Those comparators are the basic mechanism for the user to specify the desirable I/O address to be assigned to a particular quad I/O port.

The comparators perform not only a match operation between the I/O port address on the data bus and the I/O port address set up in the switches but they also perform a translation function of the I/O port address. The translation is necessary because of the fact that the 3851 PSUs which are used on the quad I/O board are pre-masked to respond only to I/O address values coded as /04, /05, /06, and /07. Thus, if a match occurs, the output of the comparator will force the five high-order bits to be zero, the sixth bit will be a one, and the seventh and eighth bits will be passed straight through from the I/O data bus representing the selection of one of the four ports on the PSU.

# D.2.5 ROMC Code Timing Sequences

As mentioned earlier, the execution of an I/O instruction or an interrupt instruction consists of a sequence of machine cycles where ROMC codes and data on the data bus are interpreted by the 3851. In order to get a better understanding of the ROMC decoding circuit's functions and the comparator and multiplexer circuit's functions, we will now refer to the timing diagrams which are shown at the top left-hand corner of the block diagram. Those timing diagrams will show the typical sequence of events for typical I/O and interrupt operations. The timing diagram shows four timing slots identified as TO, T1, T2, and T3. TO corresponds to

the instruction fetch cycle which is, in reality, the tail end of the execution of the previous instruction. During the instruction fetch of a normal I/O operation as shown at the top half of the timing diagram, the ROMC code on the bus will be /00; however, through the resistor forcing network at the output of the three-state buffer as discussed earlier, the 3851s are forced to see a /1C code which is a NO OP code for the 3851.

During T1 which is, in reality, the first cycle time of the I/O instruction time execution, the ROMC code will be a /03 or a /1C depending upon the type of I/O command under execution. In either case, the 3851 will, again, be forced to see a /1C code. It is during this T1 time that, if an I/O port address match had occurred, a legitimate address of /04, /05, /06, or /07 will be simultaneously presented to the 3851. The /1C code causes the 3851 to latch the port address and get prepared for a meaningful code that may follow at time T2.

At time T2, then, if the ROMC code is a /lA or a /lB, the 3851 will initiate the appropriate data transfer between the I/O port specified and the data bus of the CPU. During the following time interval, which is T3, the processor will force the ROMC lines back to zero and it will go on to fetch the next instruction. However, the 3851, through the forcing resistor network, will again be forced to observe a /lC ROMC code which, as mentioned earlier, causes the 3851 to be in a NO OP state waiting for the possibility of a meaningful ROMC code during the next time interval.

The lower half of the timing diagram indicates the sequence of ROMC codes during a typical interrupt operation. During TO, which would normally be the instruction fetch time, the processor is aware of the fact that it is in the process of servicing an interrupt request and, therefore, in this case, it specifically forces the ROMC lines to be in the /10 configuration as opposed to the normal I/O operations where the ROMC codes were /00. This /10 code on the ROMC lines is called a FREEZE INTERRUPT code and its function is to serve notice to all circuits capable of interrupting the CPU that the processor is intending to service the highest priority interrupt. During the next time slot, that is time Tl, the processor forces a /1C code on the ROM lines which causes the PSU to latch up the port address while allowing the priority networks to settle as to which one has the highest priority between them. During T2 the processor forces the ROMC lines to a /0F code which causes the PSU to dump the low-order

eight bits of the interrupt vector address onto the bus. During time interval T3, the processor forces a ROMC code of /13 which causes the PSU to dump the eight high-order bits of the interrupt vector address onto the bus. After time T3 the processor continues with the execution of the vectored instruction while the quad I/O board remains in its inactive but alert state (forced at /1C) waiting for the possibility of a new cycle.

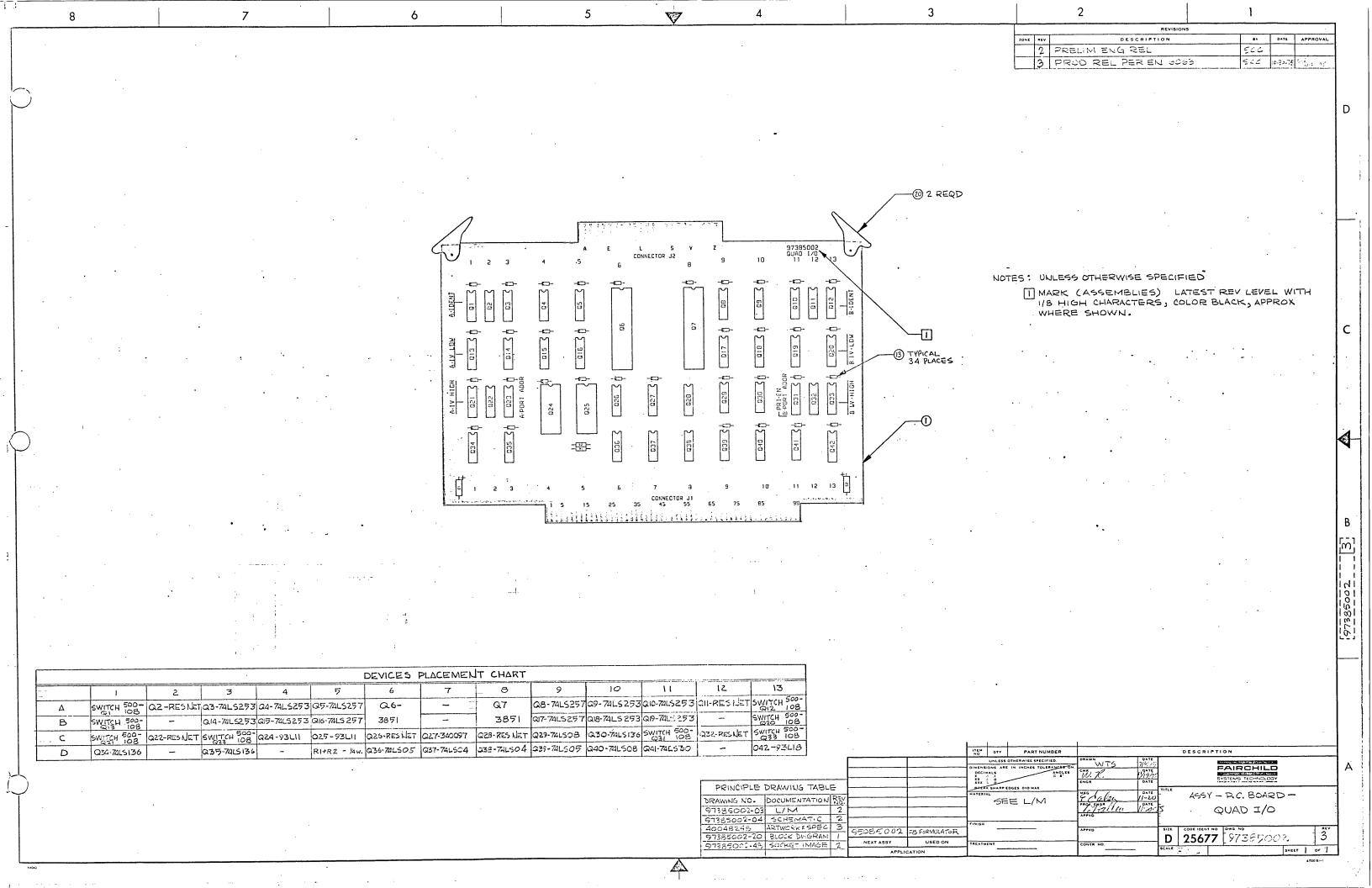
## D.2.6 Data Bus Priority

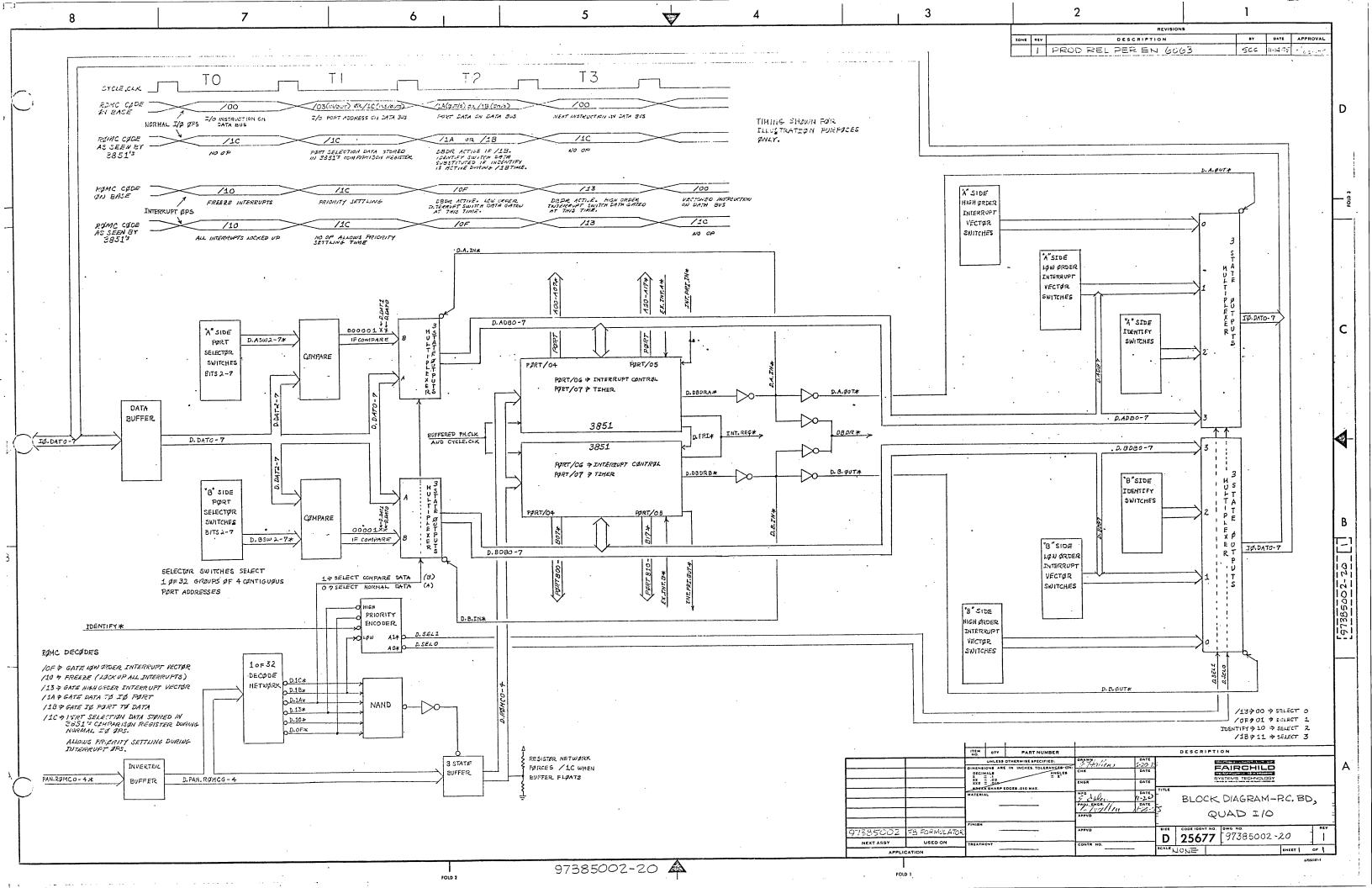
A signal of special interest that should be discussed here is the so-called data bus drive signal. That signal is the output of each 3851 PSU and is indicated at the output of each one of the two 3851s on the quad I/O board as D.DBDRA\* and D.DBDRB\*. The DBDR signal serves two purposes: It notifies all other units to stay off the bus and it signals the CPU that there is valid data on the bus.

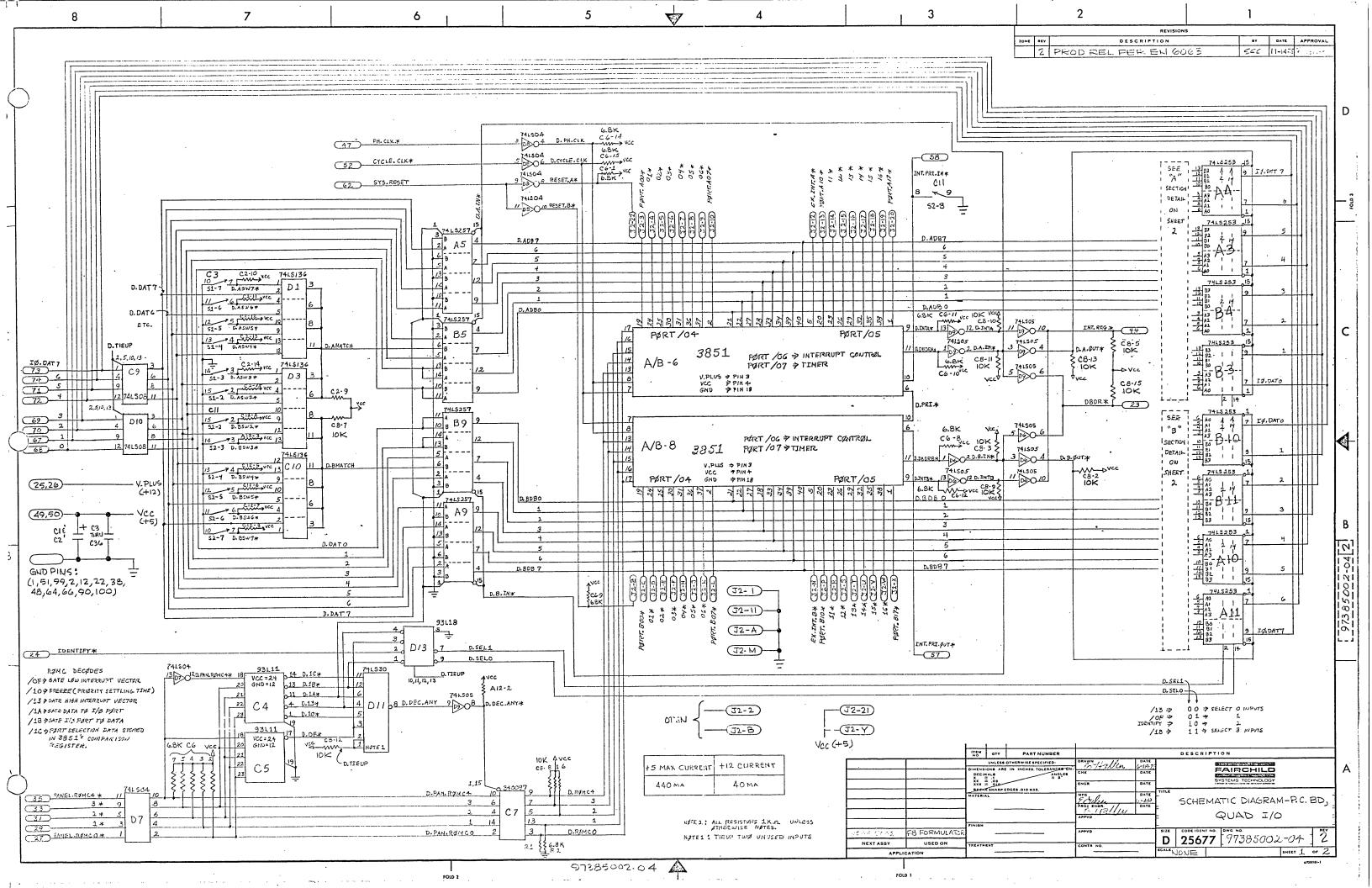
On the quad I/O board, the signal DBDR, through appropriate buffering, performs all the necessary gating of the three-state output multiplexers at both the data input area (co-ordinates 6B and 6C) and, also, at the three-state output multiplexers (coordinates 1B and 1C).

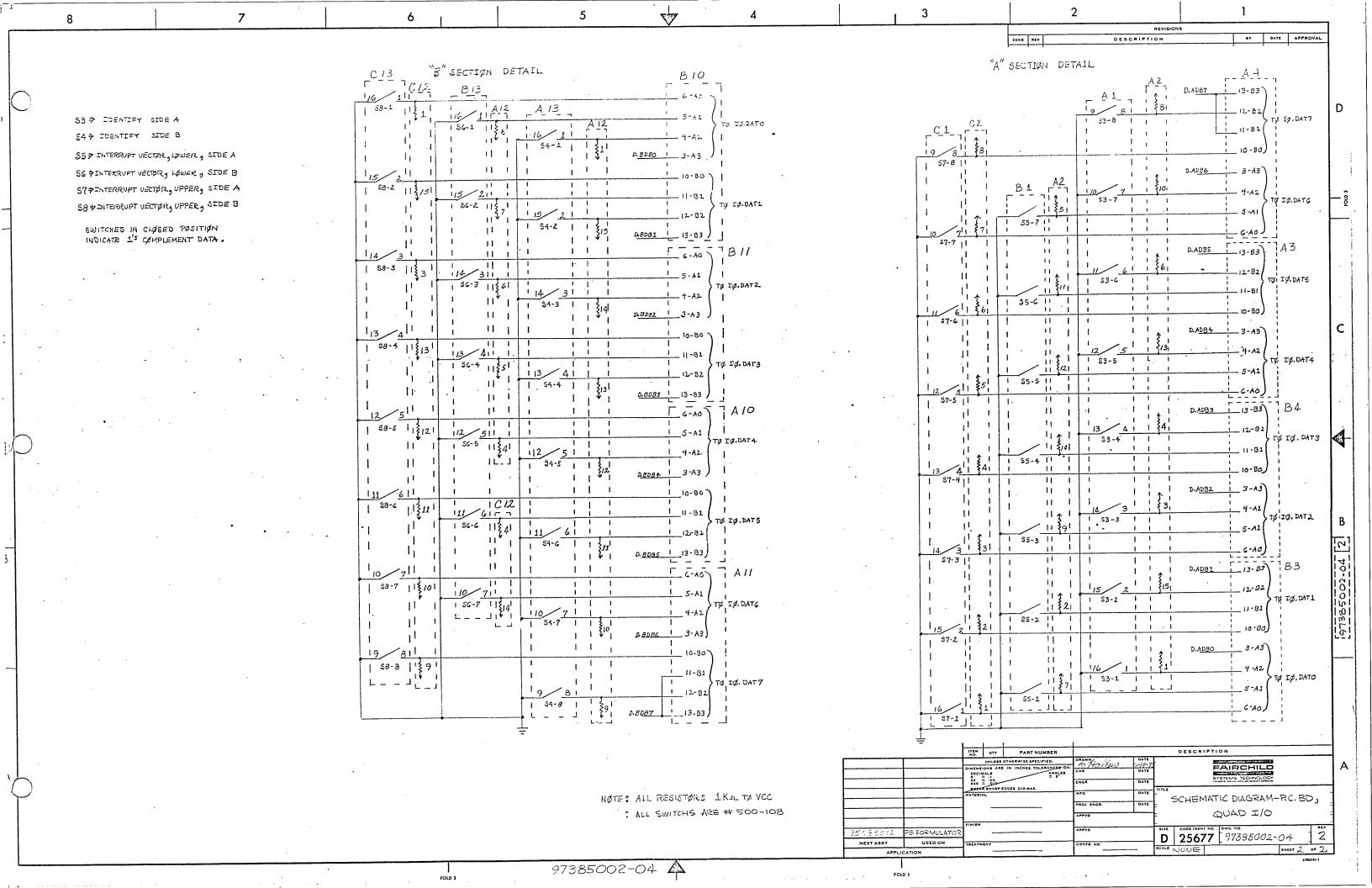
The DBDR signal will be active only during time T2 as shown on the timing diagram and, possibly, during time T3 for interrupt type of operations.

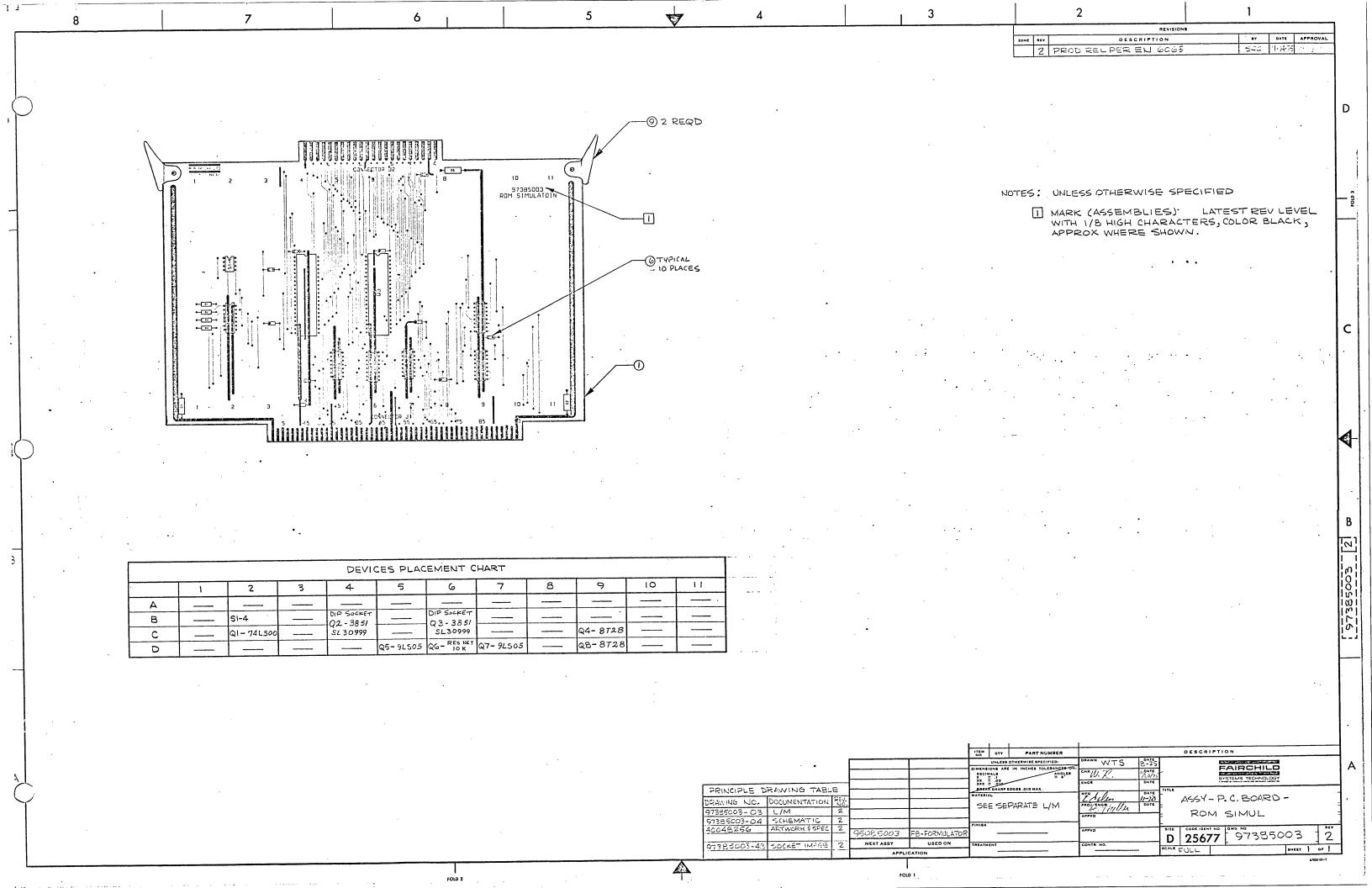
The schematic diagram of the quad I/O port board will describe all the detailed circuit connections found on the quad I/O board, the device identifications, and the I/O pins as utilized by the quad I/O board. Topologically speaking, the various circuits of the quad I/O board schematic are located in the same general coordinate areas as they were shown on the block diagram.

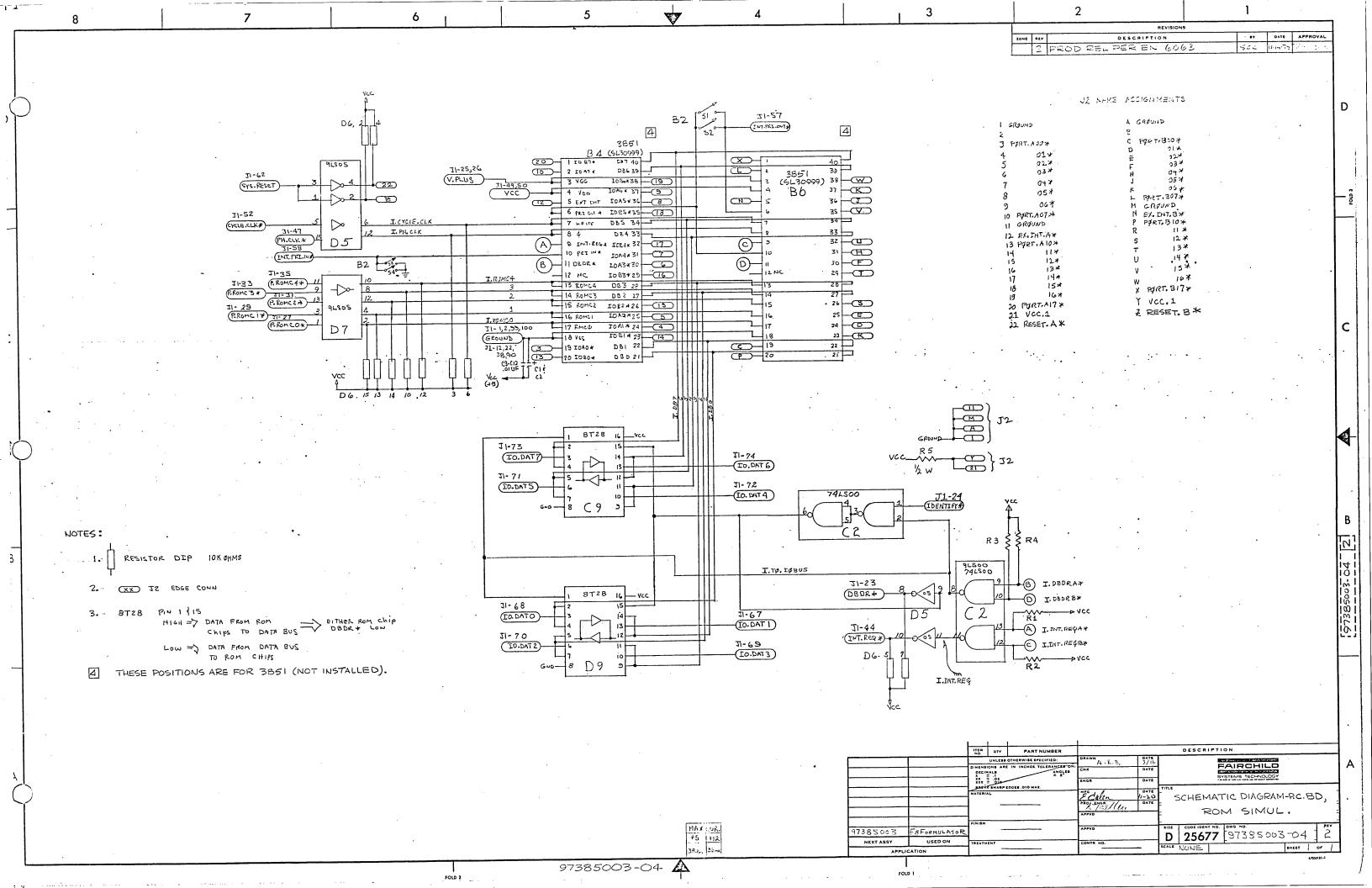












Theory of Operation: CONSOLE DISPLAY BOARD (Assembly No. 97381101)

# Z.1 INTRODUCTION

The console display board is one board of the set of three that performs the console fuction of the F8 FORMULATOR. The console display board holds the LEDs of the operator's display, the operator control switches, and a small number of interface and driving ICs. Its function is to be the input/output window that connects the operator to the console microprocessor.

# Z.1.1 Functional Description

#### SWITCHES:

The function of the 39 switches on the console control board are to be the tools used by the operator to control the RUN/HALT mode of the system and to communicate with the console microprocessor while halted. The switches fall into four groupings--system mode select, display mode select, data entry, and sense switches. Thirty-three of the switches are connected into a matrix and are scanned by the console microprocessor while the system is halted. The other six, plus two from the matrix, connect directly to hardware and are functional if the system is running or is halted.

#### LEDs:

The function of the 48 LED indicators on the console display board is to communicate information to the operator about system status and about the contents of system registers. The displays fall into four groupings: system status indication, display mode indication, data display, and address display. The address display is driven directly by the program counter register of the user's microprocessor; the other displays are driven by I/O ports of the console microprocessor.

# Compare and Count:

A 16-bit comparator and an 8-bit counter are included in the console control board. They function as part of the logic that will halt the system after a given number of passes through a given address. This comparison and count logic needs to function only during the RUN mode so that the same latches that hold the count and the address can be used to drive the displays.

The comparator is a compare of 16 bits to the system address; the 16 bits are set by the console microprocessor. Eight of the 16 bits are held directly in the I/O port that drives the DISPLAY DATA bus. The other eight bits are held in the DISPLAY BUS LATCH.

The function of the counter is to count up to all ones from a value loaded by the console microprocessor. While the system is running, the Console Control Board will generate a count pulse with every address compare. While halted, the 8-bit counter is used to drive the lower eight bits of the data display.

## Z.2 DETAILED THEORY OF OPERATION

There are three buses coming into the Console Control Board-the Address bus (ADRxx) [C8]\*, the Display Data bus (DISP.DATxx) [B8], and the Control Data bus (CNTRL.DATxx) [A8]. The Display Data bus is bidirectional; the other two inputs to logic on the board. Internally, the Display Data bus passes through the DISPLAY LATCH [B5] to become the Display Latch Bus (Z.DISPxx). All four buses are inverted-the active state is 0 volts.

## Z.2.1 Control Data Bus

The Control Data bus [A8] provides the latch enable signals, the display mode indicator enable, the switch enable, and a 1-of-8 switch row enable. The 1-of-8 decoder is buffered through 9N17 gates to provide the worst case 12 TTL UL load; the decode order of the 1-of-8 decoder is reversed because the Control Data bus is inverted. Table 1 gives the bit assignments.

## Z.2.2 Switch Matrix

Almost all of the switches are connected into a matrix [A6]. Eight switches are read at a time. The switches are diodeisolated so that more than one switch can be making at a time. Both sides of most switches—NO and NC—are available so that the console microprocessor program can debounce them easily. Table 2 relates switch number to row and function. The DISPLAY.DATA bus is driven out of the board while switches are being read. A contact closure is a low impedance to ground; pull-ups are provided on the Console Memory Board.

\*References in brackets are coordinates on the Console Display Board schematic drawing No. 97381101-04.

Bit No.	Assignment						
7	Load Display bus latch						
6	Load Data latch						
5	Load Counter latch						
4	Display enable						
•							
3	Switch enable						
2	Control.Data 2 Select						
1	Control.Data 1 $\begin{cases} 1-of-8 \\ switch \end{cases}$						
0	Control.Data 0 rows						

NOTE: All signals are active low

Table 1: Assignments of Control Data Bus

_			•						
Column	7	. 6	5	4	3	2	1	0 ,	
111	0	0	O   _ Data 5 _	O	O   _ Data 3 _	O   _ Data 2 _	,	O . Data 0	Off
110	Data 7 —	_ Data 6 _     O   S7	- Data 3 -	) 0 S5	) 0 84	)   ;   s3	52	o sı	Active
101	0	9	O Data 13	Data 12	O Data 11	O Data 10	O     Data 09 -	O     Data 08 —	Off
100	Data 15 _	Data 14 _	Data 13 _		S12		0 s10		Active
011	O Mode	O Mode	O Incre	O `. Decre — Addr —	Recall Adr (Active) S22	O Store Data -	O Exam Data -	Clr Display (Active) S18	Off.
010	Left -	Right -	Addr -	.  O s23	Load Adr (Active) S21	O S20	O S19	(Spare)	Active
001	0	0	O CMPR	(Spare)	Pass CNT	O RUN -	(Spare)	O SSTEP -	Off
000	Test O	IPL -	Run O S32	Break (Active)	O 53		HALT (Active) S29		Active
					-				

Table 2: Switch matrix map.

The enable to the row of eight system mode switches on the side which is normally open is forced low while the system is running to insure that the HALT (SW29) and BREAK (SW31) switches can function. The HALT and BREAK switches are broken out from the switch matrix individually because they must be able to drive the HALT logic on the Console Control Board while the system is running; the console microprocessor only scans the switch matrix while the system is halted. The BREAK switch is debounced.

The Sense, RESET, and EXT.INT switches are independent of the switch matrix--and of the RUN/HALT state of the system. The RESET and EXT.INT switches [B4] are debounced and then go to appropriate points on the Processor card via the Console Control card. The four sense switches [C8] are simply closures to ground; they go directly to one CPU port on the Processor card. The sense switches are the only non-momentary switches of the Console Display board.

# Z.2.3 Address Latches and Comparator

The Address bus is sampled into four latches [D7] which, in turn, drive the sixteen Address LEDs. The strobe for these latches, PCO.STROBE, is timed so that the latches sample the bus in the early part of each cycle--when the contents of the bus are known to always be the Program Counter (PCO); Figure 1 shows the timing. While the system is halted, the console microprocessor has control over the ADDRESS DISPLAY because it has complete control over the user's PCO register.

The Address bus also provides one 16-bit operand for the comparator. The comparator is sixteen EXCLUSIVE-NOR gates [C6] whose outputs are wire-ANDed together. The other operand comes in two halves--the lower half from the DISPLAY LATCH, the upper half directly from the DISPLAY.DATA bus. Both operands are inverted; if the operands differ in any bit position, the ADR.EQUAL signal [C4] will be pulled inactive (low). The comparison will not necessarily be with the Program Counter as the comparison is not strobed; the timing of the comparison process is discussed in the description of the Console Control Board.

# Z.2.4 DATA LATCHES AND COUNTER

The DATA Latches [C3] are loaded from the DISPLAY.LATCH bus. They drive the upper byte of the DATA DISPLAY LEDs.

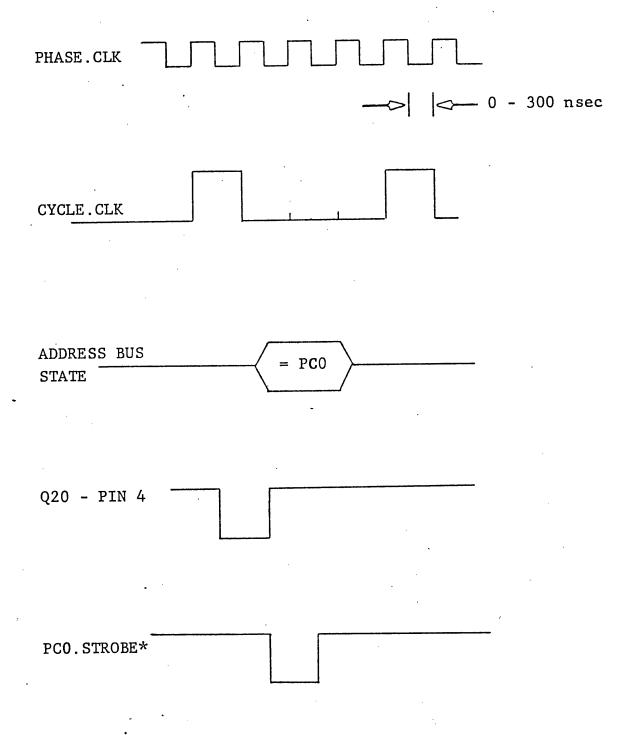


Figure 1: PCO Strobe Timing

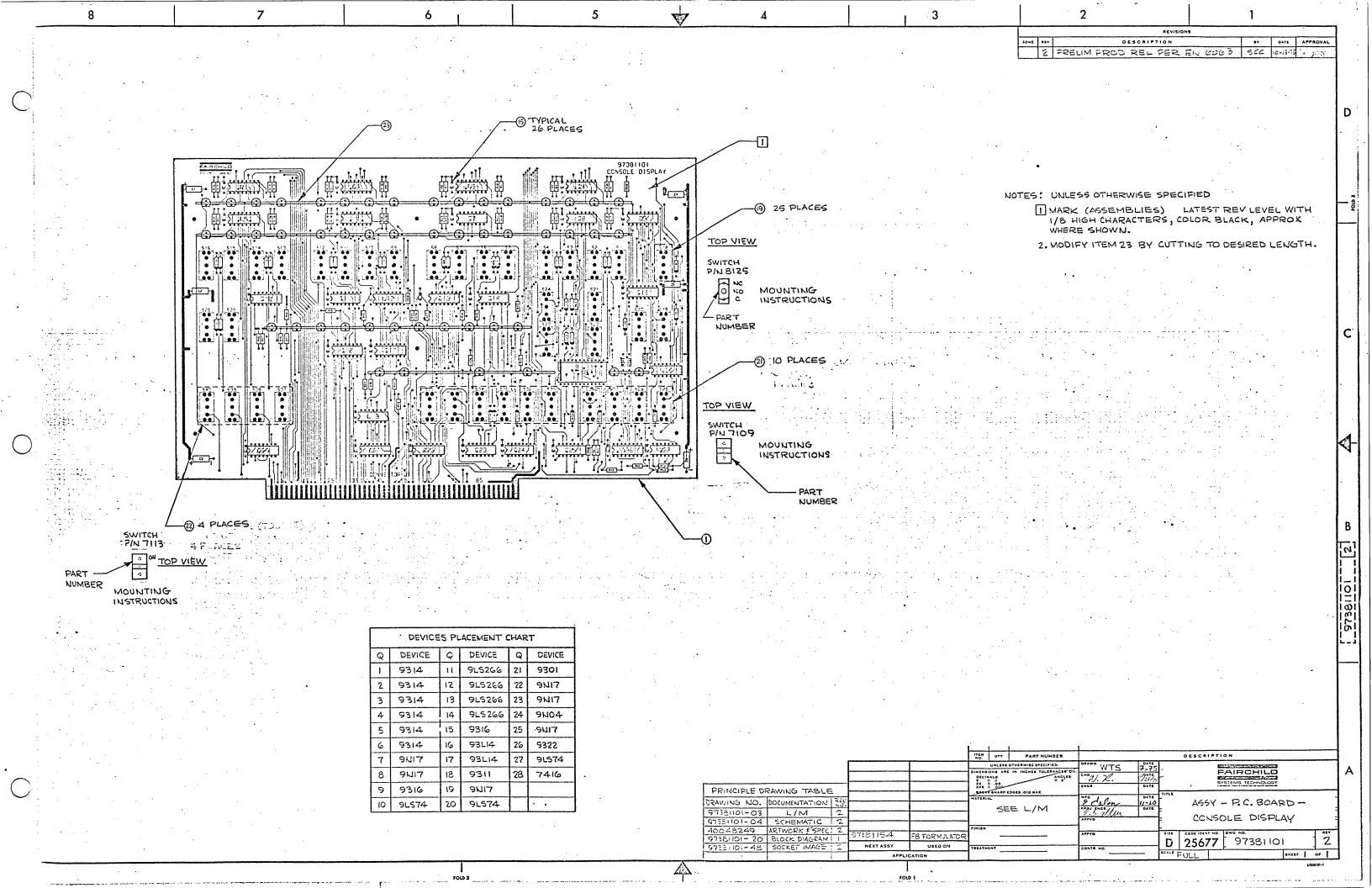
The 8-bit COUNTER Latches [C2] are also loaded from the DISPLAY.LATCH bus. They drive the lower byte of the DATA DISPLAY LEDs. The counters are not used to count while the system is halted; they merely drive the DATA DISPLAY at that time. While the system is running, the counters become part of the Compare Halt function. A count value can be loaded into the counter while halted; while running, the ADR.EQUAL signal [C4] drives hardware on the Console Control Board to generate the count enable pulse--CMPR.CNT. Because the buses are inverted, the counter appears to be counting down while it thinks it is counting up. The all ones state--which appears to be all zeros--is detected; it is a signal called CNT.DONE [C1]. The Console Control Board will block further counting when CNT.DONE is active or the system is halted. Consult the Console Control Board description for more information.

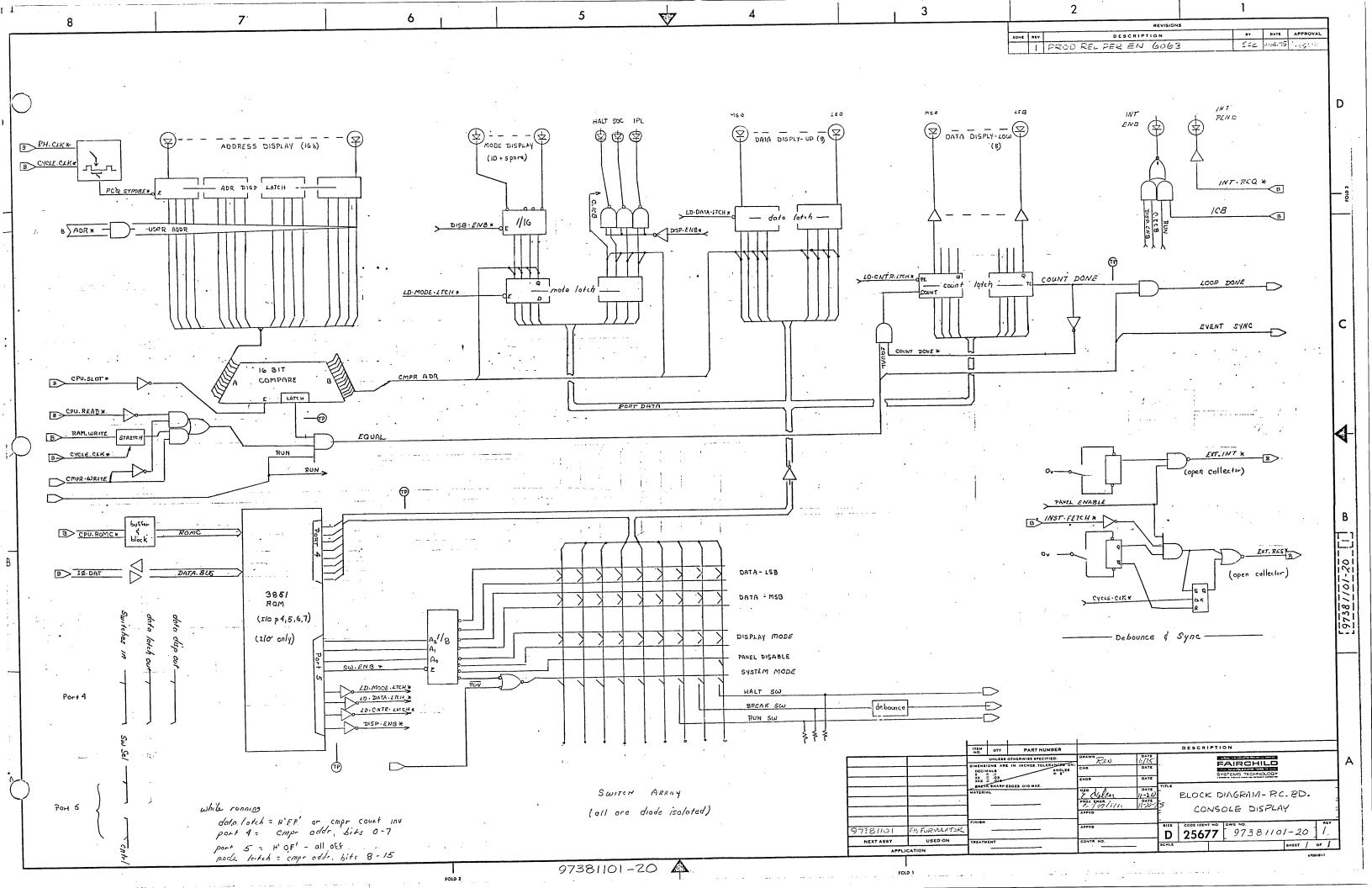
## Z.2.5 MODE DISPLAY

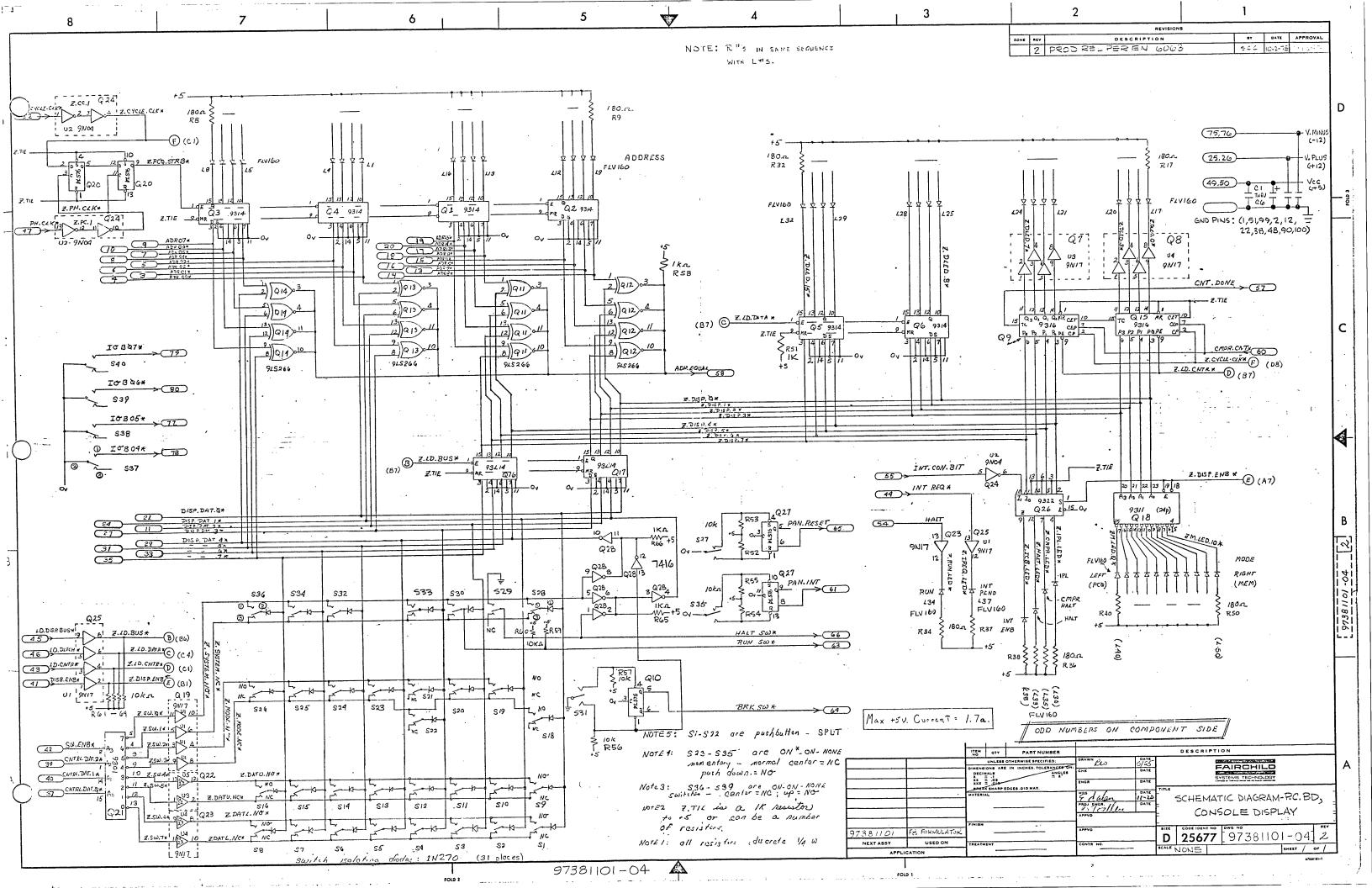
The mode display [B2] is driven directly from the DISPLAY. LATCH bus, rather than through latches. A control signal, Z.DISP.ENB, allows the console microprocessor to blank the mode display when the DISPLAY.LATCH bus is being used to load the DATA LATCH or the COUNTER.

The four low-order bits of the DISPLAY LATCH bus drive one of the ten display mode LEDs through a 1-of-16 decoder [B2]. The upper four bits are each assigned to drive a single LED; the four LEDs indicate IPL, COMPARE, HALT, and INT.ENB. These four LEDs are driven through a 9322 quad two input multiplexer. The MUX select input is DISP.ENB: three of the LEDs are dark when DISP.ENB is inactive. The fourth, INT.ENB, is connected to the ICB signal of the F8 CPU when DISP.ENB is inactive; the rationale behind this is to have the INT.ENB follow the ICB flag while the system is running but have it under control of the console microprocessor while the system is halted.

There are two remaining LEDs--HALT and INT.PEND [B3]. These are driven directly from base signals. The HALT signal comes from the Console Control Board.







# Theory of Operation: CONSOLE CONTROL BOARD (Assembly No. 97381102)

# X.1 INTRODUCTION

The console control board is the demarcation point between the user's microprocessor system and the console microprocessor system. On this board is the hardware that shares the single CPU between the two microprocessor systems and that determines which system is currently active. Also on this board is the hardware that lets the console microprocessor system look into the user microprocessor.

# X.1.1 User and Console Microprocessor Systems

Two separate microprocessor systems exist within the F8 FORMULATOR--the user's microprocessor and the console microprocessor. The user's microprocessor system consists of the processor card, RAM cards, I/O cards, and communication card; the user system is a general system that is defined by the user to fit his needs. The console microprocessor system is a small dedicated system whose job is to let the user freeze and then examine his system; the console microprocessor system consists of the console display card, console memory card, the console control card, and the processor card.

The goals of the console function are:

- to be able to examine all registers;
- to be able to alter all registers;
- to halt without destroying user parameters;
- to be able to continue with all user parameters restored
- so that it seems as if nothing had happened;
- to not restrict user configuration;
- to not deprive the user of memory or I/O;
- to be able to execute single instructions;
- to be able to display the program counter contents while running; and
- to be able to halt the system at some specified program point.

These goals are met by the console microprocessor system in conjunction with special hardware tools that let the console system delve into the user system. A microprocessor system is

used because it puts the complex portions of the console function into a software program that is stored in simple memory. The microprocessor system also allows flexibility and convenient inclusion of features that are logically simple but in hardware would be complex--such as switches that repeat if held down.

The processor card is a part of both systems; the CPU of the processor card is time-shared between the two systems. Each system has its own F8 memory interface logic, its own memory, and its own I/O structure; however, neither system has its own CPU. The situation, shown in Figure 1, is like having two tails but only one head. While at first this may seem to be a poor situation, it is, in fact, very useful for reaching the console's goals: when the system is running, the CPU is attached to the user's system and the memory structure of the console system is held idle; as the system is halted, the CPU is transferred to the console system and the user's memory structure is now held idle. With that transfer of the CPU to the console, the console microprocessor system has just gained access to lots of user registers and user status—such as all 64 registers of the scratchpad.

Other hardware lets the console system look into the user memory structure while halted. The user's I/O, program counters, data counters, and actual memory will all look like a part of console memory to the console microprocessor system. The hardware forces commands into the user memory while the console is executing some memory reference instructions. An overview of the timing relation of these events is given in Figure 2. Figure 3 is a block diagram.

The hardware that controls the RUN/HALT status is also considered part of the console function. The transition to HALT is triggered by hardware-monitored conditions. The transition to RUN state is in response to a command issued by the console microprocessor program. Single-step and compare-halt are conditioned forms of RUN--the condition flags are passed to the HALT hardware. The HALT conditions are:

HALT, RESET, or BREAK switch pushed
STOP line activated
SINGLE-STEP or CMPR-HALT conditions satisfied
BREAK instruction encountered in user's program.

While the user system is held idle in the HALT state, its clocks are not stopped; interval timers will continue to count down while the system is halted. Interrupts may become pending during the HALT state but they will not be serviced.

The idling of the console memory structure or the user memory structure is accomplished by switching the ROMC signals. These signals originate in the CPU and direct the action of the memory structures. The ROMC signals are fed to the user when the mode is RUN and are fed to the console while HALTed. The inactive memory structure is fed a NO-OP ROMC state to hold it idle.

## X.2 FUNCTIONAL DESCRIPTION

The console control card has six functional areas:

- RUN/HALT and ROMC steering;
- Console memory interface;
- Force and Run trigger timing;
- Force-ROMC and flag register;
- HALT conditions register; and
- BREAK and COMPARE-HALT detect.

## X.2.1 RUN/HALT and ROMC Steering

The function of the RUN/HALT and ROMC steering block is to maintain the RUN/HALT state and to steer the ROMC signals to the active memory structure. An inactive console memory structure gets a reset command; an inactive user system gets a NO-OP command.

## X.2.2 Console Memory Interface

The function of the console memory interface is to control console memory. Associated with it is page select circuitry which decodes the most significant four bits of address. The memory map is given in Figure 5. Console memory includes RAM, PROM, and special hardware registers. STORE instructions to some parts of memory trigger RUN or force-into-user hardware.

# X.2.3 Force and Run Trigger Timing

The function of the Force and RUN trigger logic is to accept special hardware commands from the page select circuitry and use a shift register to delay the transition of the hardware until the desired point in the instruction flow. The pattern being shifted distinguishes between a command to begin running and a command to force a command into the idle user memory structure. The RUN bit pattern, after causing the system to toggle into the RUN mode, is shifted once more to provide a gating signal for the single-step logic.

# X.2.4 Force-ROMC and Flag Register

The ROMC/FLAG register has a function in both RUN and HALT modes. In HALT mode the register holds a five-bit word that is substituted for the NO-OP on the user's ROMC bus for one cycle whenever the console system is accessing the user microprocessor system. Of the three remaining bits, two become the single-step and compare-halt flags during RUN mode. The third bit is used during HALT modes to designate 'IDENTIFY' operation for the user's memory and I/O boards. (IDENTIFY causes the user boards to substitute unique board identification codes in place of normal data; the console system uses IDENTIFY to build a table of the user system's configuration.)

# X.2.5 HALT-Conditions Register

The function of the HALT-conditions register is to hold the different HALT conditions. The console microprocessor program can read the HALT status byte held in the HALT-conditions register by treating the byte as a memory location at H'4000'. The contents of the register are ORed together to become the halt command to the RUN/HALT logic. The single-step condition halts the system but is not part of the HALT status byte available to the console program. The panel disable switch is part of the HALT status byte but doesn't cause a HALT command. Some of the bits of the HALT status byte are directly tied to switches; the other bits are latched and will remain latched once set until the system goes to the RUN mode.

# X.2.6 BREAK and COMPARE-HALT Detect

The BREAK logic and the COMPARE-HALT logic detect these two conditions during the RUN state. The outputs are fed into the HALT-conditions register and, from there, generate a HALT command. The BREAK logic responds to two different user instructions (OUTS, 2 and OUTS, 3) and to the BREAK switch. The BREAK switch is gated so that only one halt occurs per depression. The COMPARE-HALT logic works with the address comparator and counter that reside on the console display board. The address compare is gated with the user's CPU SLOT, CPU READ, and RAM WRITE to detect a valid memory reference. The COMPARE-HALT logic also has the function of driving the counter. Whenever there is a valid comparison, the COMPARE HALT flag is set. The logic will set the COMPARE bit of the HALT-conditions register when the loop count is zero and there is a valid comparison.

#### X.3 DETAILED CIRCUIT OPERATION

X.3.1 The RUN/HALT logic centers around two flip-flops [1A4].\* The two flip-flops are set in sequence during the course of one F8 cycle (defined by one period of the cycle clock). The first flop is strobed after one microsecond has elapsed in a cycle by X.RC.Strobe; the second flop is strobed at the end of the cycle by X.FCLK. The timings of the events of a RUN to HALT transition are given in Figure 6 and of HALT to RUN in Figure 7.

The HALT command is gated with Instruction Fetch to insure that HALT transitions occur at the end of an instruction. Keying the HALT transition off the fetch ROMC state insures that instruction execution is complete because the fetch of the next instruction always occurs during the last cycle of execution of the current instruction.

Note that during the cycle of the HALT transition, neither system is active and the hardware forces the instruction from location H'0000' of the console memory to be fetched into the CPU.

- X.3.2 The ROMC signals are steered to the console MI by a 2-input multiplexer [1C7]. The MUX is controlled by X.HALT; the two states are considered RUN and CONSOLE.HALT. A 4-input multiplexer [1B7] steers the ROMC signals for the user system; the outputs are called PANEL.ROMC and go along the base to all user boards. The MUX is controlled by X.FORCE and X.USER.HALT; three states are used: RUN, USER.HALT, USER.FORCE. The states and multiplexer outputs for both MUXs are given in Table 2. The difference between USER.HALT and X.HALT was shown on the timing chart of Figure 6; they differ only during the HALT transition.
- X.3.3 The console static memory interface (3853) [1D6] supplies the address and control for the console memory. Its interrupt structure is active while the F8 FORMULATOR is halted. The data bus lines are buffered by 8T28 drivers [1D8]. The console MI and the rest of the console memory structure is prevented from driving the IO.DAT bus while the F8 FORMULATOR is in RUN mode or while a command is being forced. At other times, either the drivers or receivers of the 8T28 are enabled. While the F8 FORMULATOR is running, neither receiver nor driver are on and resistors pull the X.MIDBx bus [1D7] to ground so that the MI's program counter is being continually forced to H'0000'.

<sup>\*</sup>Numbers in brackets refer to coordinates on the schematic of the console control board, drawing No. 97381102-04.

The most significant four bits of the MI's address go into a 1-of-8 decoder [1C3]. It produces the chip selects for PROM, RAM, the HALT status byte, the ROMC/FLAG register. The read enable signals are ANDed with MEM.TO.CPU [A2]. MEM.TO.CPU [A2] is active during those cycles that fetch instructions or get operands--since CPU.READ is active during those cycles. MEM.TO.CPU is also active during the cycle of the HALT transition; neither system is active during this particular cycle, so that it is up to the hardware to force a NO-OP instruction into the CPU from console memory. Because the console has been getting reset up to this point, the instruction will be fetched from location H'0000' of console memory.

X.3.4 The force and run command timing logic centers around the 9300 shift register [1B3]. The register is being clocked continuously; zeros are shifted in. A number is loaded in parallel whenever the console MI writes to its memory; the PAN.RAM.WRT signal is stretched by a flip-flop [1B3] so that it can be sampled by F.CLK--this timing is in Figure 8. The number will have one of three possible values--corresponding to a no-action condition, a RUN command, or a command to force a ROMC into the user's memory structure. The value of the PCO register, which is the address during the STORE instruction, decides which number will be loaded into the shift register. Address H'9xxx' is a command to enter the RUN mode; address H'5xxx' or H'7xxx' is a command to force a non-idle ROMC to the user's memory structure.

After the run or force command has loaded a pattern into the shift register, the register will begin shifting the pattern. The rationale for introducing this delay can be understood by looking at the program sequences that typically issue these commands. The force sequence has two cycles of delay between the RAM-WRITE cycle and the cycle during which the forcing ROMC state is inserted. A typical instruction sequence and its timing is shown in Figure 9. The second cycle of delay becomes useful when user I/O ports are accessed; the I/O port address is put on the IO.DATA bus during this cycle of delay as shown in Figure 10. Note that the typical sequence given had two successive "ST" instructions that wrote in locations within the range H'7xxx'; the X.FORCE\* signal applied to the shift register's parallel inputs prevents the second ST from starting an undesired force command.

A run command also loads a pattern into the shift register. Its pattern, B'0101', is distinct from the force pattern of B'0011'. The reason the run cummand is derived from the shift register is to provide a delayed signal necessary in detecting a single-step halt condition. The run command looks the same whether RUN, SINGLE.STEP, or CMPR.HALT switches were pressed;

flags are loaded into the ROMC/FLAG register prior to issuing the run command to distinguish the three cases. The signal X.1ST.FETCH alerts the single-step logic not to halt on this fetch of the first instruction, but, rather, to wait until after it has been executed, halting at the next fetch that follows.

The rest of the circuit description refers to Sheet 2 of the schematic of the console control board.

- X.3.5 The ROMC/FLAG register is comprised of two 93L14 latches [2B4]. It is loaded from the MI.DATA bus whenever the console MI executes an ST instruction to H'6xxx' or H'7xxx'. Its most significant five bits become the ROMC.FORCE bus [2B3] that supplies the forcing ROMC state to the user ROMC multiplexer [1B7]. The lower three bits indicate singlestep mode, compare-halt mode, or identify mode. The identify bit (bit 0) becomes the open collector base signal "IDENTIFY\*". IDENTIFY is used by the console program to alter the operation of the user's I/O and memory cards while building a table of the user's configuration. The signals X.CMPR.ENB and X.SSTEP.ENB [2B4] go up into the halt condition's logic.
- X.3.6 The HALT condition's logic involves a number of circuits. This explanation will start with the 93L14 latch [2C3] that holds the halt conditions and will then work both ways from there.
- X.3.6.1 The outputs of the 93L14 latch, of the X.RESET flip-flop, of the X.SSTEP.HLT NAND gate, and the three switch inputs--DISABLE\*, HALT.SW\*, and RUN.SW\*--are loosely grouped together under the heading HALT-conditions register. These nine signals and their meanings are given in the following table.

Signal	<u>Indicates</u>			
DISABLE*	Panel disable switch input			
RUN.SW*	RUN switch is being pushed			
HALT.SW*	HALT switch is being pushed			
X.RESET	A system reset has occurred			
X.STOP	The STOP line on the base has been activated			
X.CMPR	The conditions for a halt-on-compare have been satisfied			
X.BREAK	A user instruction of either H'B2' or H'B3' has been encountered or the BREAK switch has been pushed			
X.EXTEND	The H'B2' or H'B3' is, in fact, H'B3'			
X.SSTEP.HLT*	The conditions for halt after single instruction execution have been satisfied			

Eight of these signals are ORed together to become X.STOP.REQ [2D1]; if any of the eight are active, the STOP.REQ will set the RUN/HALT logic to the HALT state. The signal DISABLE\* is omitted from the OR structure--it can't cause the system to halt. Once the system is halted, the console microprocessor program would like to know why the system halted--this need is answered by logic that lets eight of the nine signals be read by the program as if the eight were a memory byte. The page select logic makes X.STAT.READ [2C1] active whenever an instruction reads from memory in the range H'4xxx'. The X.STAT.READ signal controls the three-state buffer that ties the halt conditions to the MI.DATA bus [2D2]. The table below gives the bit assignments; the byte is termed the halt status byte.

Bit No.	<u>Meaning</u>			
7	Disabled			
6	Reset			
5	Run switch			
4	Halt switch			
3	Break detected			
2	Software extension break			
1	Halt on compare			
0	Stop bus activated			

The single step condition is not part of the status byte; it is left out because the console program can anticipate a halt by single-step and because there wasn't a spare status bit or latch. The switch signals of the status byte aren't latched. The other five signals are latched; the latches are not cleared until the transition back to RUN mode occurs. It is possible for a bit, such as BREAK, to become set after HALT mode is entered. The different bits are not interlocked to prevent multiple bits being set; the console program will handle the situation and assign priorities.

X.3.6.2 The inputs to the halt conditions register work with some interesting combinations of signals. Starting from the top of the schematic, the three switches are encountered first; these come directly from the console display board or the base. The next five are latched; the latches are clocked at the end of every cycle by F.CLK and, once set, stay set until the next transition to RUN mode.

- X.3.6.2.1 The X.RESET signal is latched whenever X.SYS.CLR is active. X.SYS.CLR is derived from the ROMC state H'08' on the processor board; the CPU will activate the signal if it either does the automatic power-on reset sequence or if the RESET switch is pushed. Notice that the latch does not distinguish between resets that occur in the HALT or in the RUN mode; neither does the console program. X.RESET causes the system to halt if it had been running. The timing is shown in Figure 12. The halt will occur after the user's PCO register has been cleared.
- X.3.6.2.2 X.STOP is just the latched version of the base signal STOP\*. STOP\* has been set aside for user definition-perchance he may use STOP to halt the system after a certain event has occurred. STOP timing is shown in Figure 13.
- X.3.6.2.3 The X.CMPR latch will be set after the conditions of compare-halt have been met. The compare-halt feature is a debugging tool that will halt the system after a user program has gone past a specified address a given number of times. Compare-halt combines an address compare function and a counting function. The comparator and the counter live on the console display board; the count-pulse generating logic is on this board. The conditions necessary for setting the X.CMPR latch are compare mode set AND pass counter is at zero AND address matches the set address AND the current cycle is referencing memory. The conditions are explained in the following paragraphs. The timing of compare-halt is given in Figure 14.

The signal X.MEM.REF [2D6] is active if the current user cycle references memory--either to read an instruction, operand, or data byte or to write a data byte. Control signals from the user's memory interface handily designate these cycles. One of these, RAM.WRITE\*, is stretched in a flip-flop so that F.CLK will have a chance to strobe it at the end of a cycle; timing is the same as that of PAN.RAM.WRT which was shown in Figure 8.

The signal X.CMPR.EQU [2D7] is derived from the address compare. The user memory address is not necessarily valid throughout the entire cycle--for instance, it will change in the last half of a cycle if a memory refresh or DMA transfer occurs. Because the address is only valid some of the time, the ADR.EQUAL signal that comes from the comparator on the console display board [2D8] is latched in a cross-coupled pair of 9LS51 gates [2D7]; the latch is controlled by X.OPEN.LTCH--an AND of CPU.SLOT and F.CLK. CPU.SLOT is a signal that originates in the user's 3852 memory interface and that indicates valid address.

The need for ANDing CPU.SLOT with F.CLK is subtle. CPU.SLOT is derived from the PH.CLK; if there is enough skew between PH.CLK and CYCLE.CLK, driving the X.CMPR.EQU latch directly from CPU.SLOT would give the undesirable result of the X.CMPR.EQU latch [2D7] changing just as CYCLE.CLK was trying to strobe that signal into the X.CMPR latch [2C3]. ANDing CPU.SLOT with F.CLK eliminates the potential problem.

The signal CNT.DONE is the third condition for the compare The counter will be counted down to zero, gencondition. erating CNT.DONE, and then the halt request will be generated on the next address match. The count pulse, CMPR.CNT\*, is created by a 4-input NAND [2D6]; the inputs, besides X.CMPR.EQU and X.MEM.REF, include X.HALT\* and CNT.DONE\*. These second two inhibit the counting when the count is already zero and when the system is in the HALT mode. Note that the compare enable bit of the ROMC/FLAG register [2B4] is not used in the NAND gate. One other\_signal, EVENT.SYNC, is created within the compare logic [2D5]. EVENT.SYNC would typically be used to sync an oscilloscope; the signal goes to a connector on the back of the F8 FORMULATOR. will make a negative transition whenever a memory reference instruction coincides with an address match. The counter state and the compare enable bit are not a part of EVENT.SYNC.

X.3.6.2.4 X.BREAK and X.EXTEND are two related halt conditions. Two NO-OP instructions from the F8 FORMULATOR instruction set have been appropriated by the F8 FORMULATOR system; the console program is used to build instructions around these two op-codes, extending the F8 FORMULATOR instruction set. The responsibility of the console control board is to monitor the IO.DAT bus to detect when the user is fetching either of these two instructions. The two op-codes are H'B2' and H'B3'; if either is detected, the X.BREAK latch will be set and a halt command generated. If the op-code was H'B3', the X.EXTEND flip-flop will also be set. The detecting of the two op-codes is done by two cascaded 1-of-10 decoders [2C5]. The timing sequence of a BREAK halt is shown in Figure 15.

There is a front panel switch called BREAK. When it is pushed, the same events should happen as when the op-code H'B2' is detected. However, just one halt and break should happen each time the switch is pushed rather than a multitude of breaks. The one-halt-per-push is handled by a flip-flop and AND gate (in actuality, an AOI gate) [2C5]. The combination will let BREAK switch set the X.BREAK latch once but not a second time until the switch has been released. The debounce flip-flop on the console display board is important to this scheme.

- X.3.6.2.4 The remaining condition is X.SSTEP.HLT\*.
  X.SSTEP.HLT\* is created by a NAND gate. Inputs to the NAND gate are the single-step enable bit, the fact that an instruction is being fetched in the current cycle, and the fact that this is the fetch of the second instruction after the RUN mode transition rather than the fetch of that first instruction. The single-step condition is not latched; instead, it goes directly into the OR gate to toggle the RUN/HALT logic. Single-step timing is given in Figure 16.
- X.3.7 There are two switches on the console display board that bypass the console microprocessor program and that go to the CPU strictly in hardware. These switches are RESET and EXT.INT. Both pass from the display board to the processor board via the console control board. The outputs of the two switches are ANDed with PANEL.DISABLE\* on the control board [2B2]; the logic will disable the two switches if the panel is disabled. The two signals are driven from the control board by open collector gates.
- X.3.8 Interrupt requests from the user and from the console MI are multiplexed on the console control board [2B7]. The user requests an active in RUN mode. In HALT mode the user requests are blocked and console system interrupts become active.

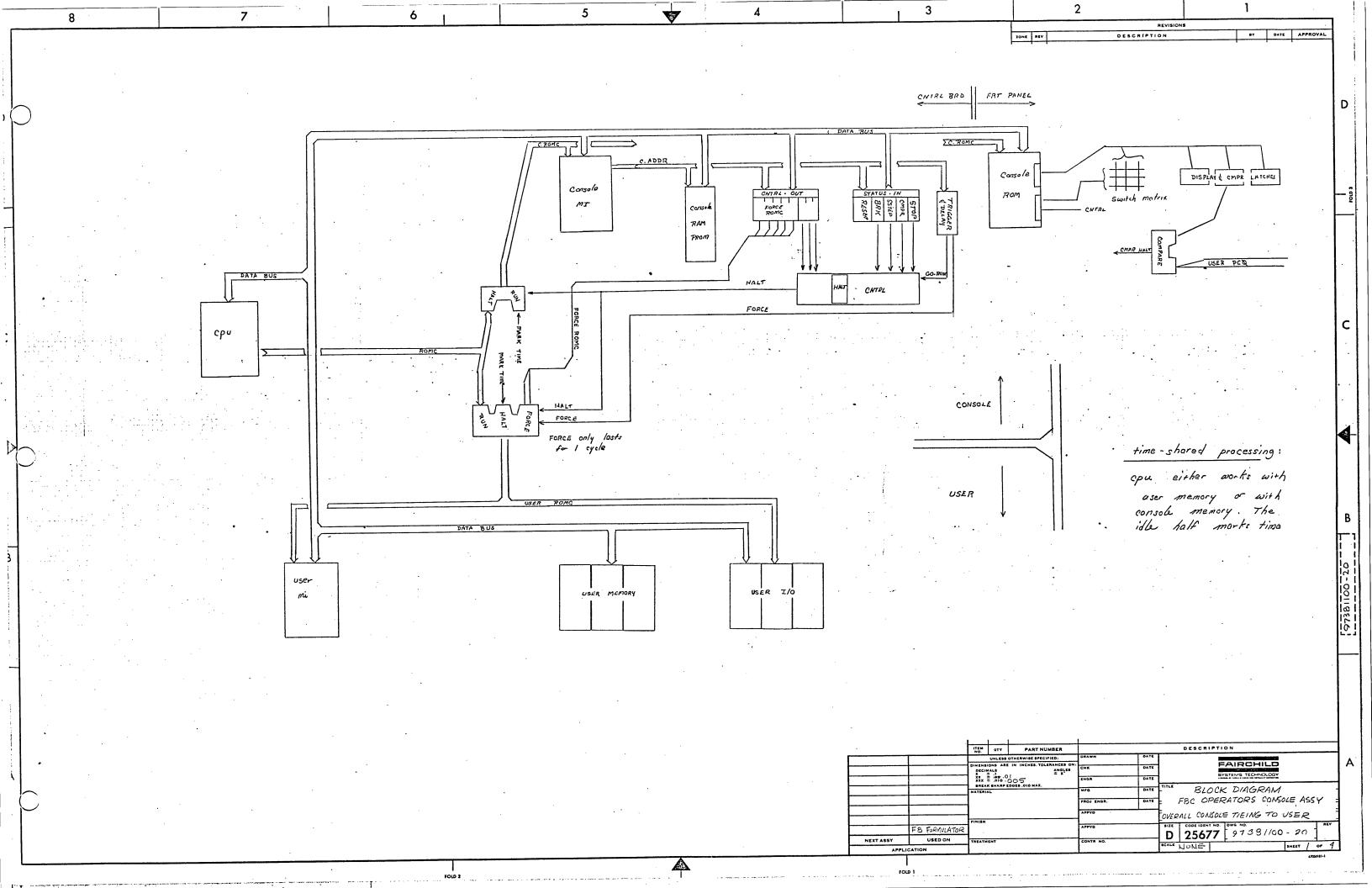


TABLE 2 - ROMC MULTIPLEXERS

	CONSOLE STATES		USER STATES
MUX INPUT OV	MUX OUTPUT H'08' = RESET	CONSOLERUN	USER
5V	CPU. ROMC×	CONSOLE	USER
		HALT	USER +5V OV (2) FORCE ROMCs FORCE (X.RCFx)

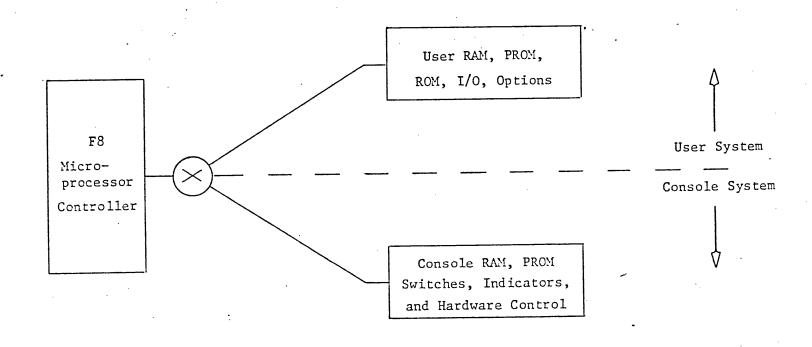


Figure 1=: FORMUL8 system gross breakdown.

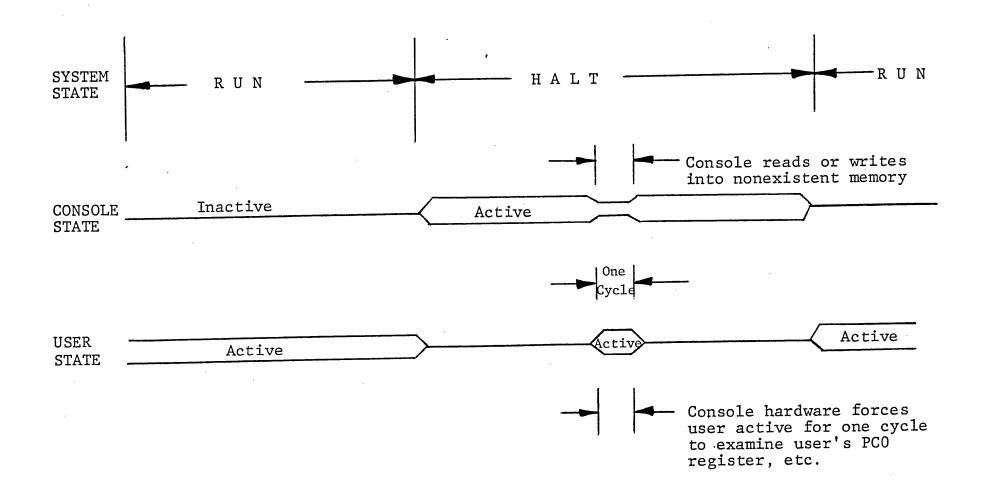


Figure 2.

_		)
FFFF		
F000		
E000		
Д000		Reserved
C000		
в000		
A000		
9000	HARDWARE TRIGGER	
8000	DEBUG	
7000	ROMC FORCE COMBINATION	
6000	ROMC FORCE DATA	
5000	ROMC FORCE TRIGGER	•
4000	STATUS FETCH	
3000	RESERVED	·
2000	RESERVED	
1000	CONSOLE RAM	
0000	CONSOLE PROM	

Figure 1-3: Console memory addressing

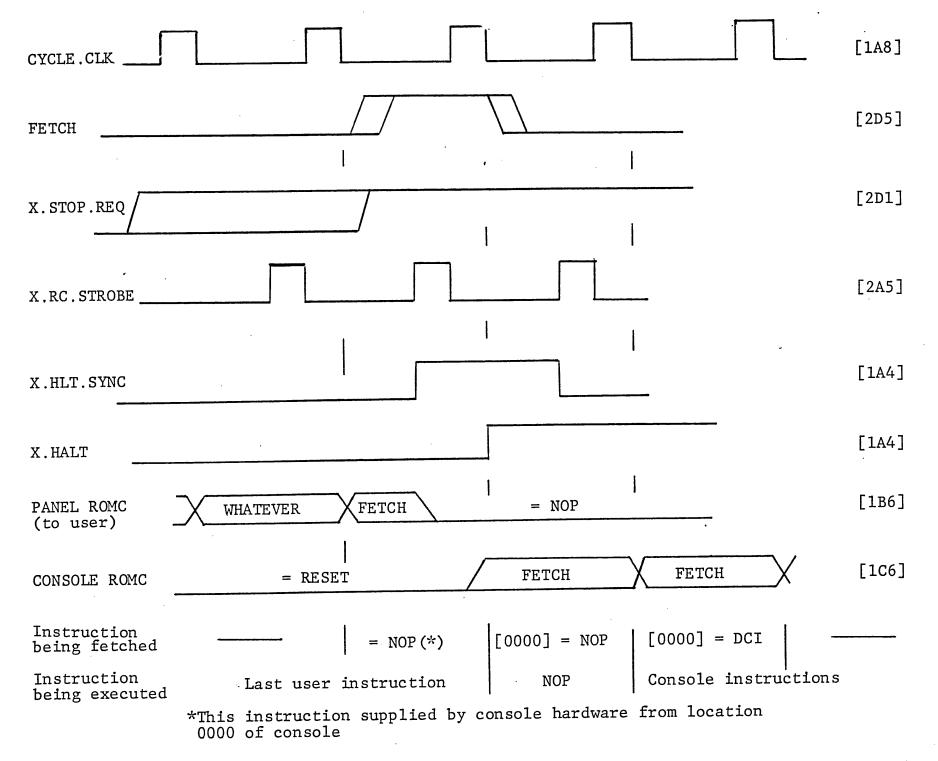
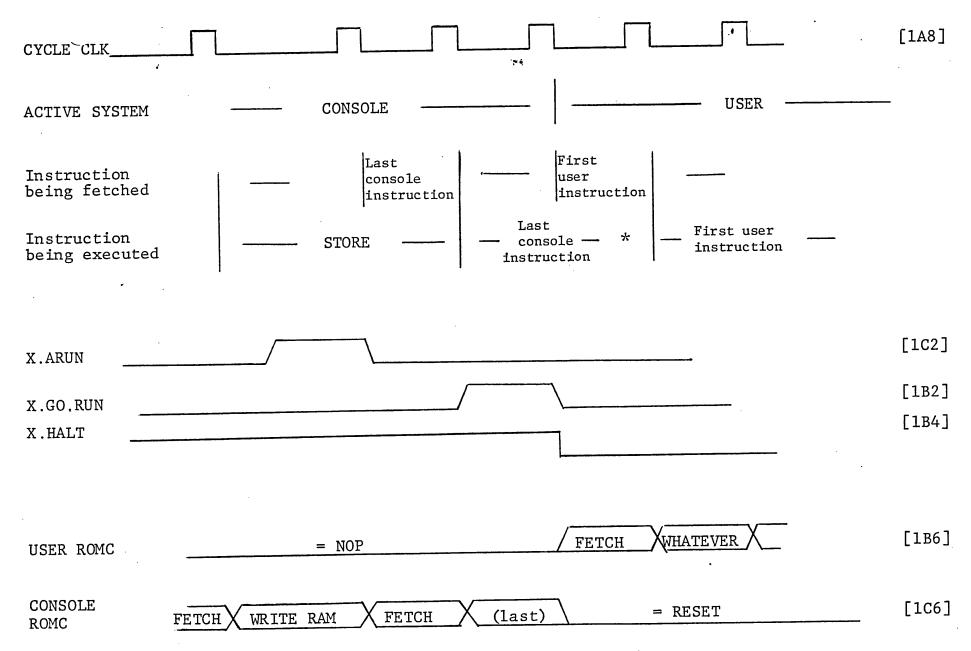


Figure 6. RUN to HALT transition



<sup>\*</sup>Last console instruction is privileged, guaranteeing at least one user instruction is executed before an interrupt occurs.

Figure 7. HALT to RUN transition

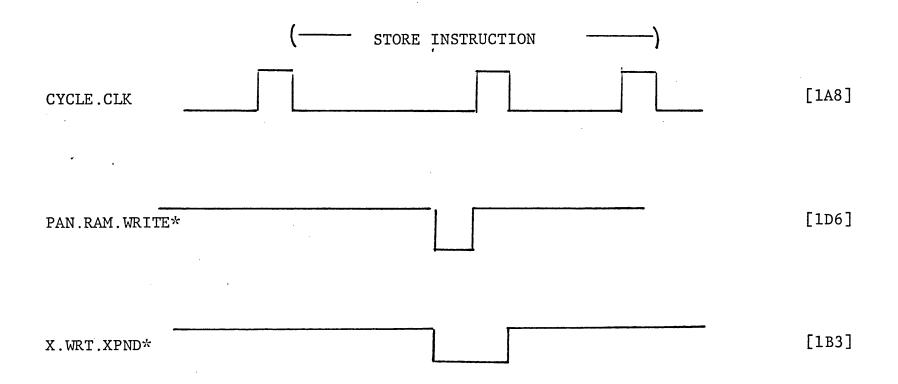


Figure 8. Stretching of RAM.WRITE

# A. The example and necessary program sequence

This example shows how to get the value of the lower byte of the user's PCO back into the accumulator.

The ROMC that moves the value of PCO-L onto the data bus is  $\mbox{H'1E'}$ .

### Program sequence:

* *	H'1E'	has been loa	aded into RC/FLAG latch
Begin DCI ST	ST	н'5000'	Set DCO to H'5000' Dummy store - trigger H/W
*	NOP LM		Value of PCO-L will be loaded
*	DONE LR	0,A	

Figure 9: Example of force hardware accessing user registers

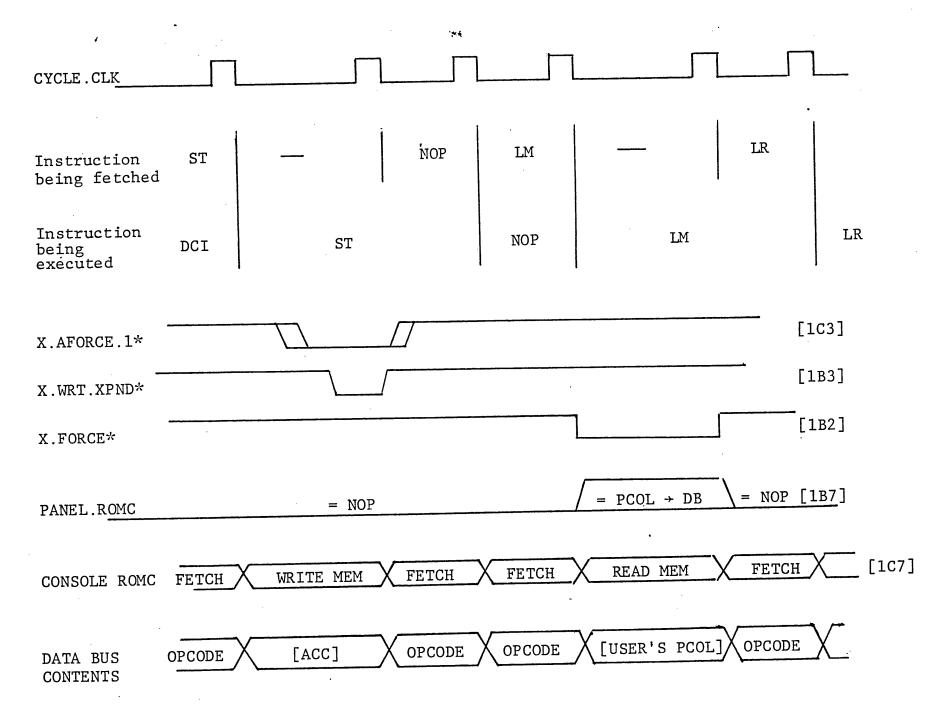


Figure 9b. Timing of user FORCE example

#### A. Necessary program sequence

This example shows how to input from a user port which has I/O address H'34'. The program will input it into the CPU scratchpad register KL. The port address will be loaded into KU. The ROMC that passes the I/O port address into the I/O port logic is H'1C', NOP. The ROMC that inputs from a port is H'1B'.

#### Program sequence:

```
H'34xx' has been loaded into console's PC1
ぉ
*
      H'1b' has been loaded into RC/FLAC latch
                          Set DCO to H'5000"
              H'5000'
Begin DCI
                          Dummy store - trigger h/w
      ST
ャ
                          User port 34 into KL register
      LR
              K,PC1
*
*DONE
              A,KL
      LR
```

Figure 10: Example of force hardware accessing user I/O port

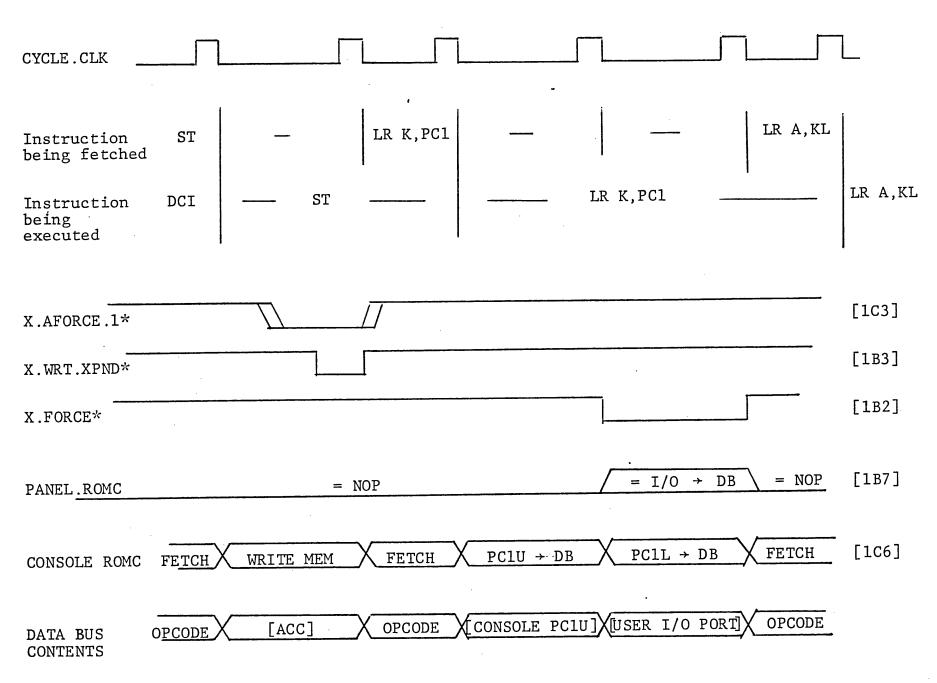


Figure 10b. Timing of user I/O access

. 🛪

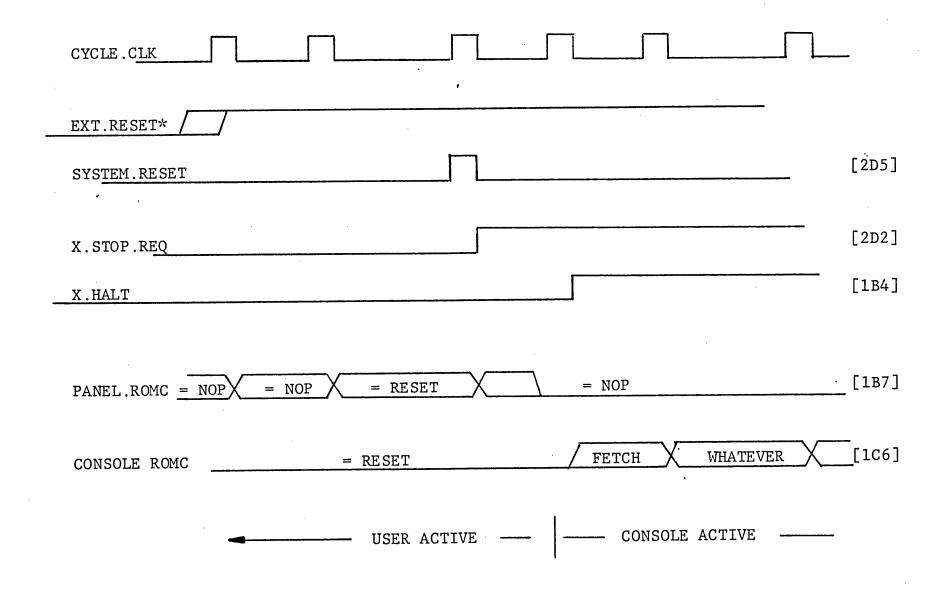


Figure 12. Timing of RESET while in RUN mode

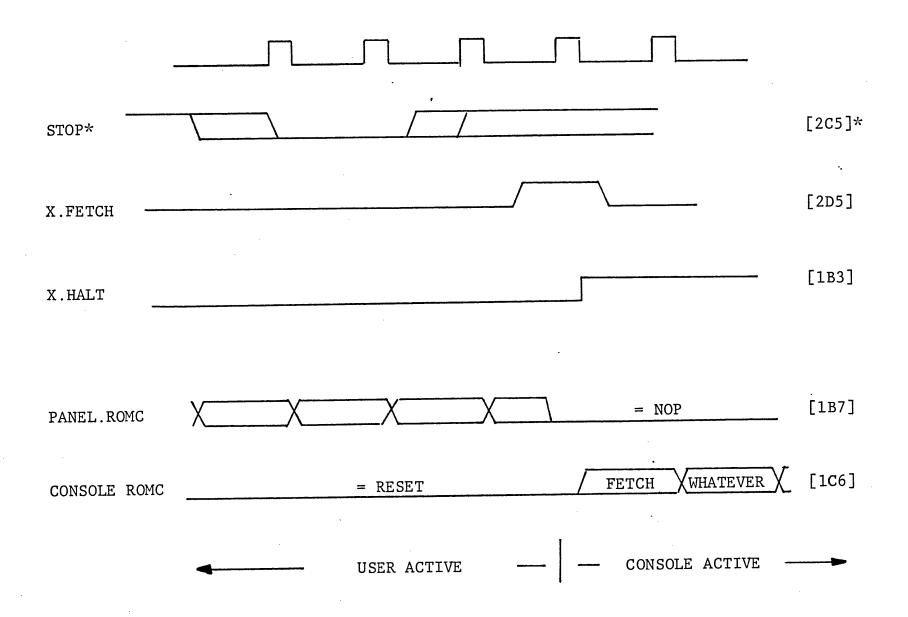


Figure 13, Timing of STOP

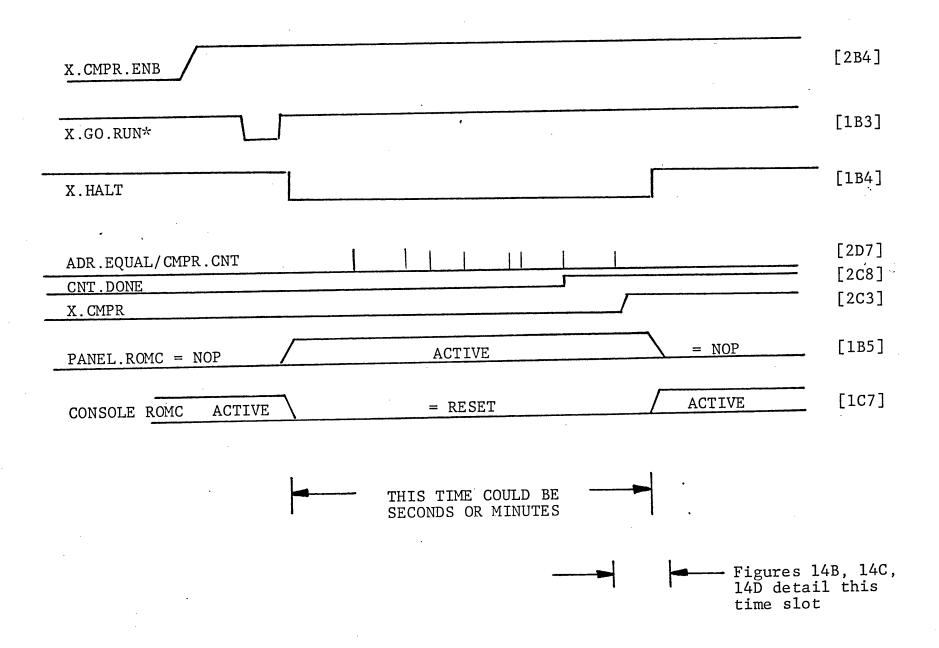


Figure 14A. Overview of COMPARE-RUN

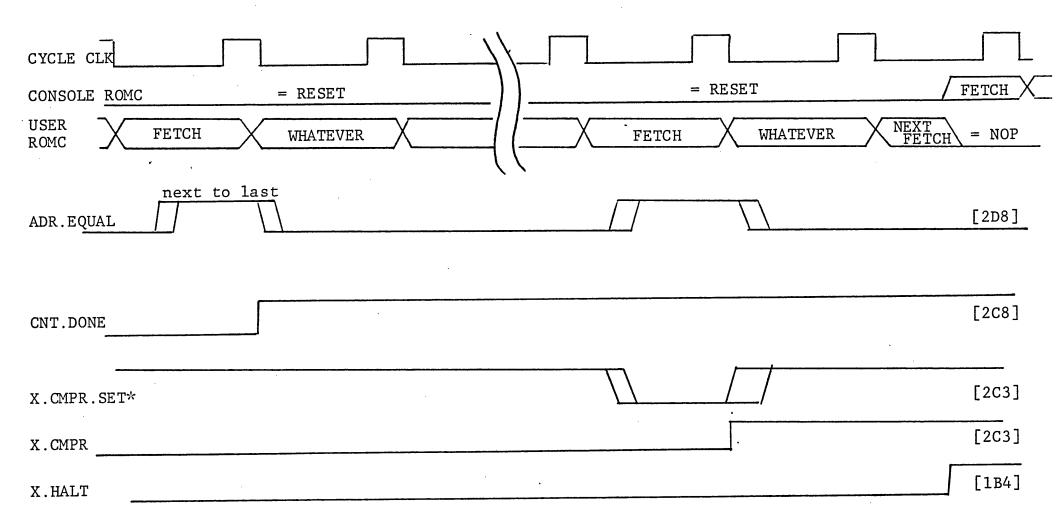


Figure 14b. COMPARE-RUN set to watch INSTRUCTION FETCH

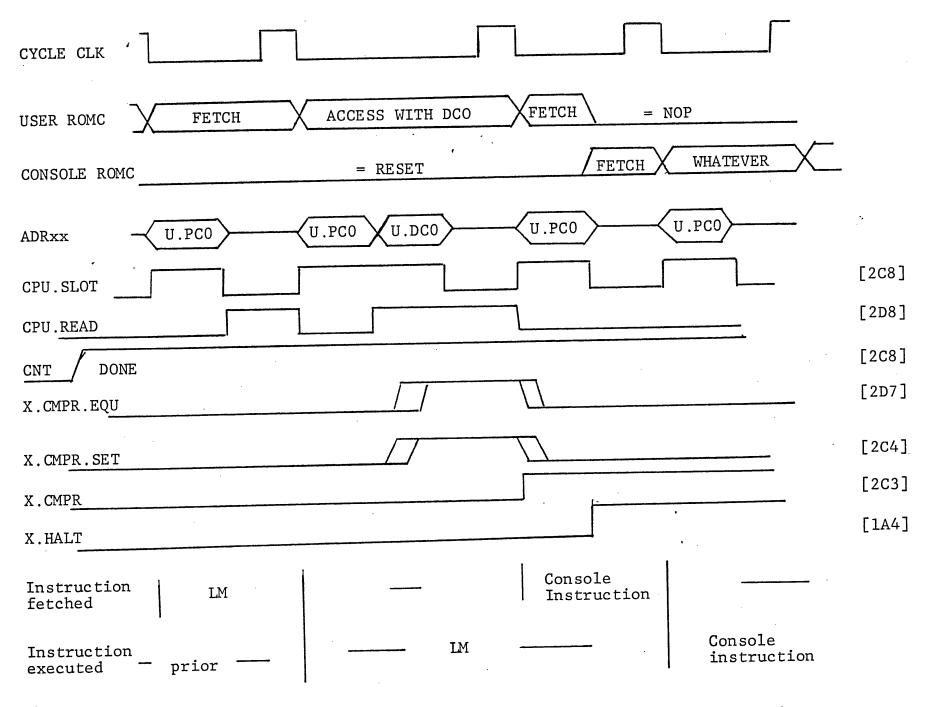


Figure 14c. COMPARE-RUN set to watch for DATA ACCESS with DCO

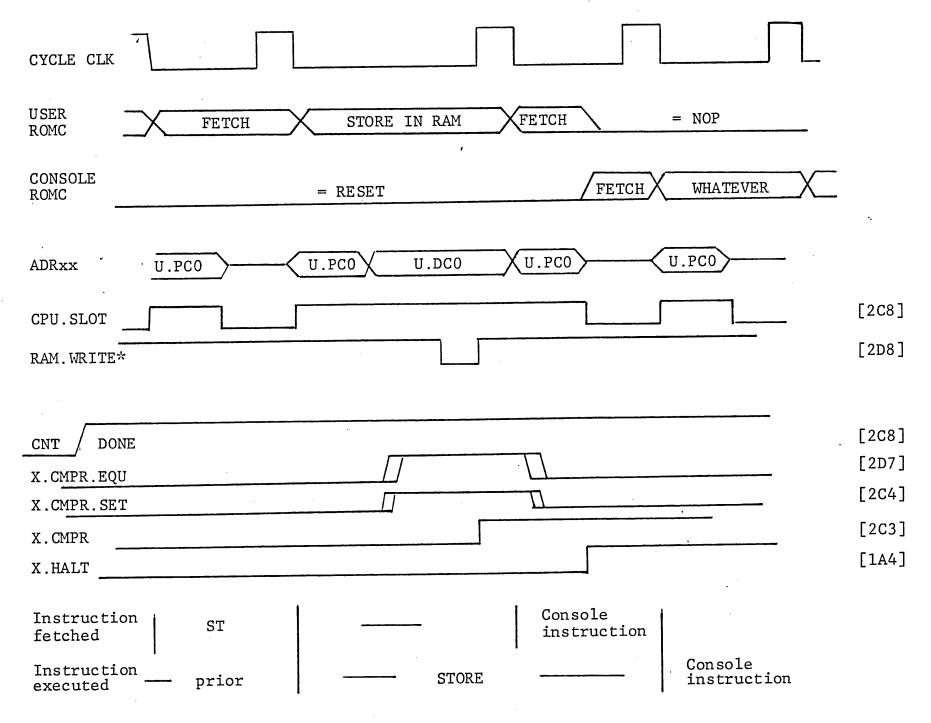


Figure 14d. COMPARE-RUN set to watch for MEM.WRITE

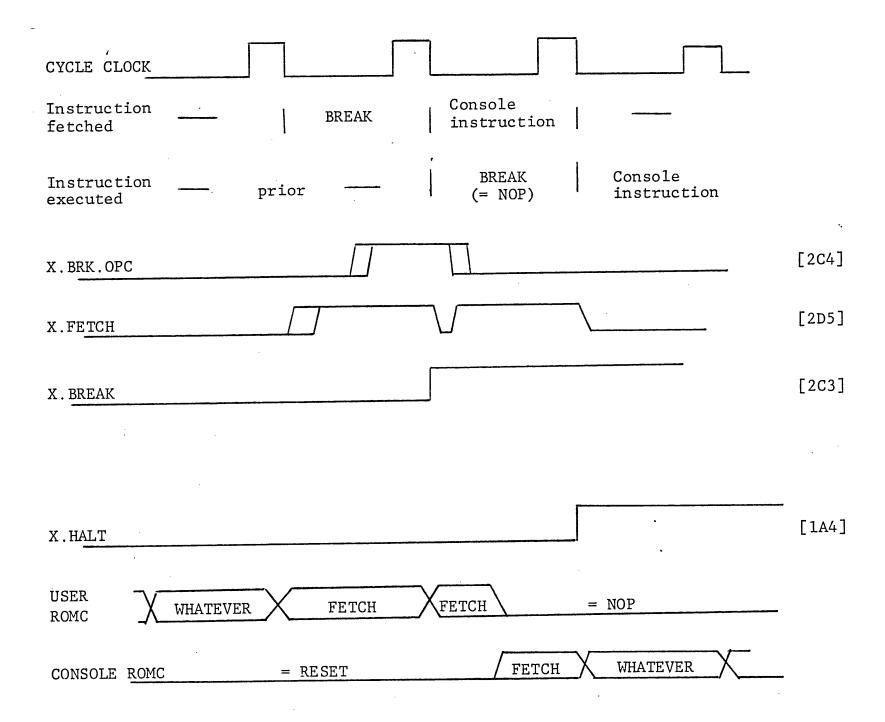
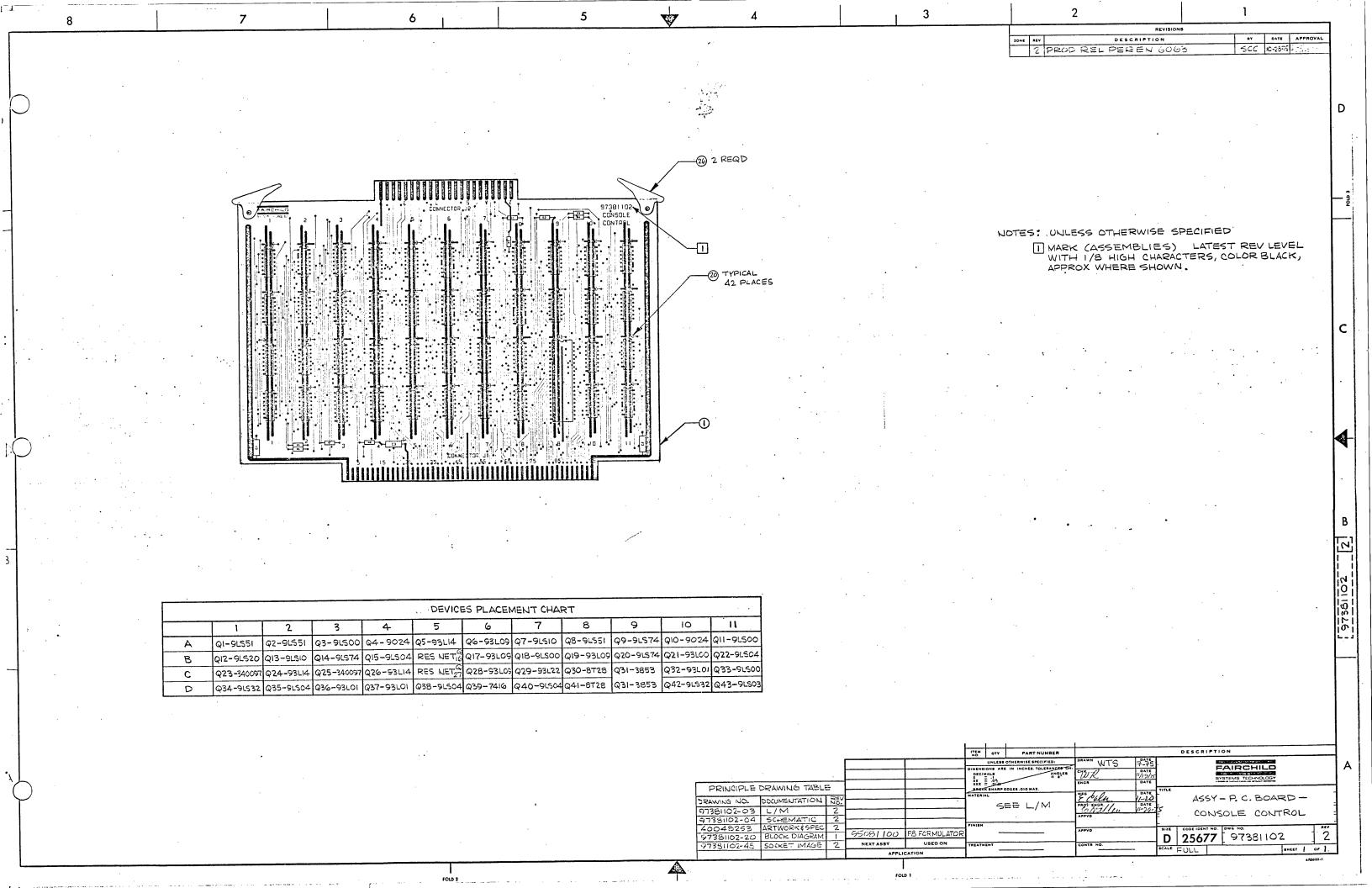
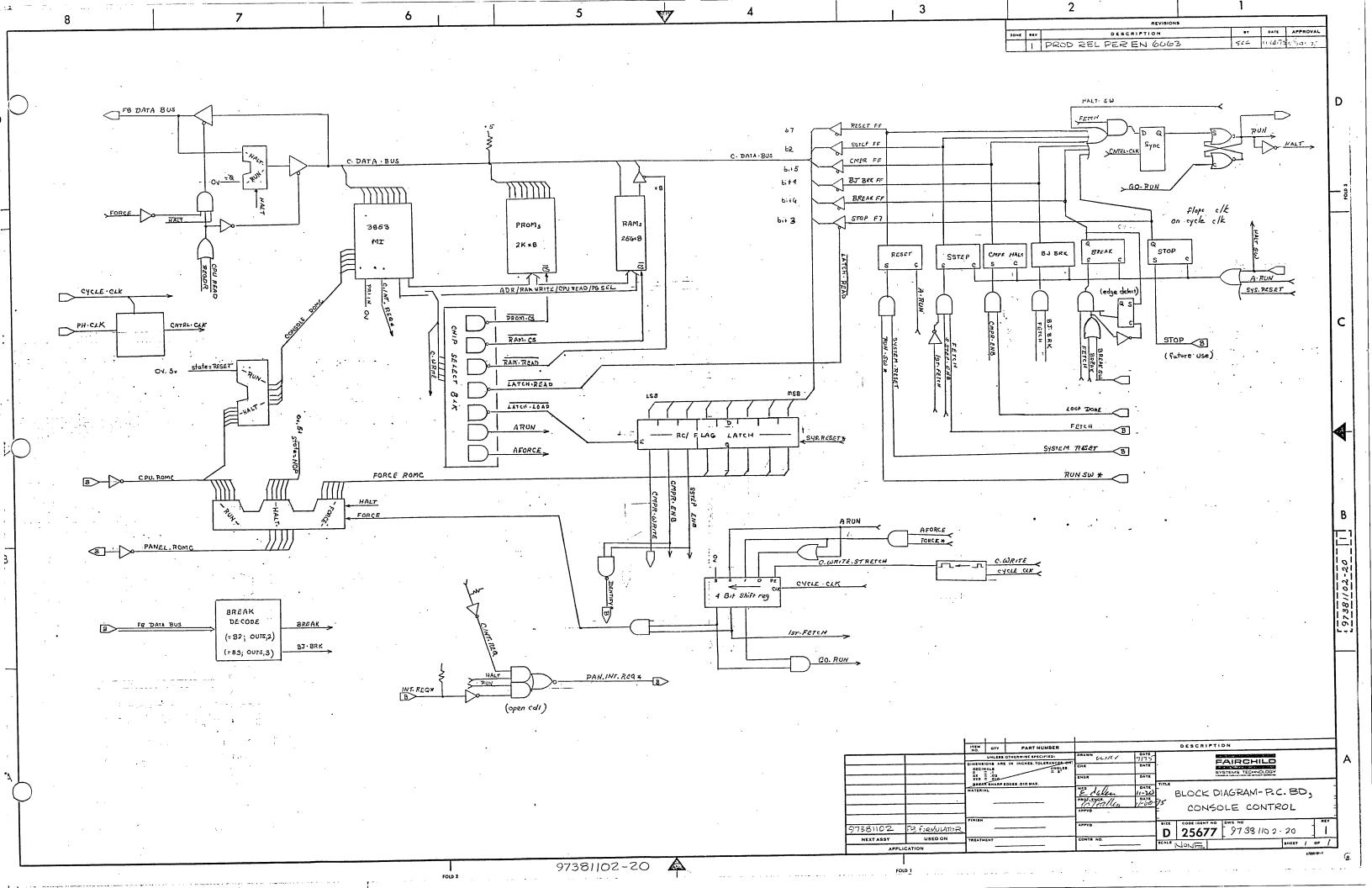
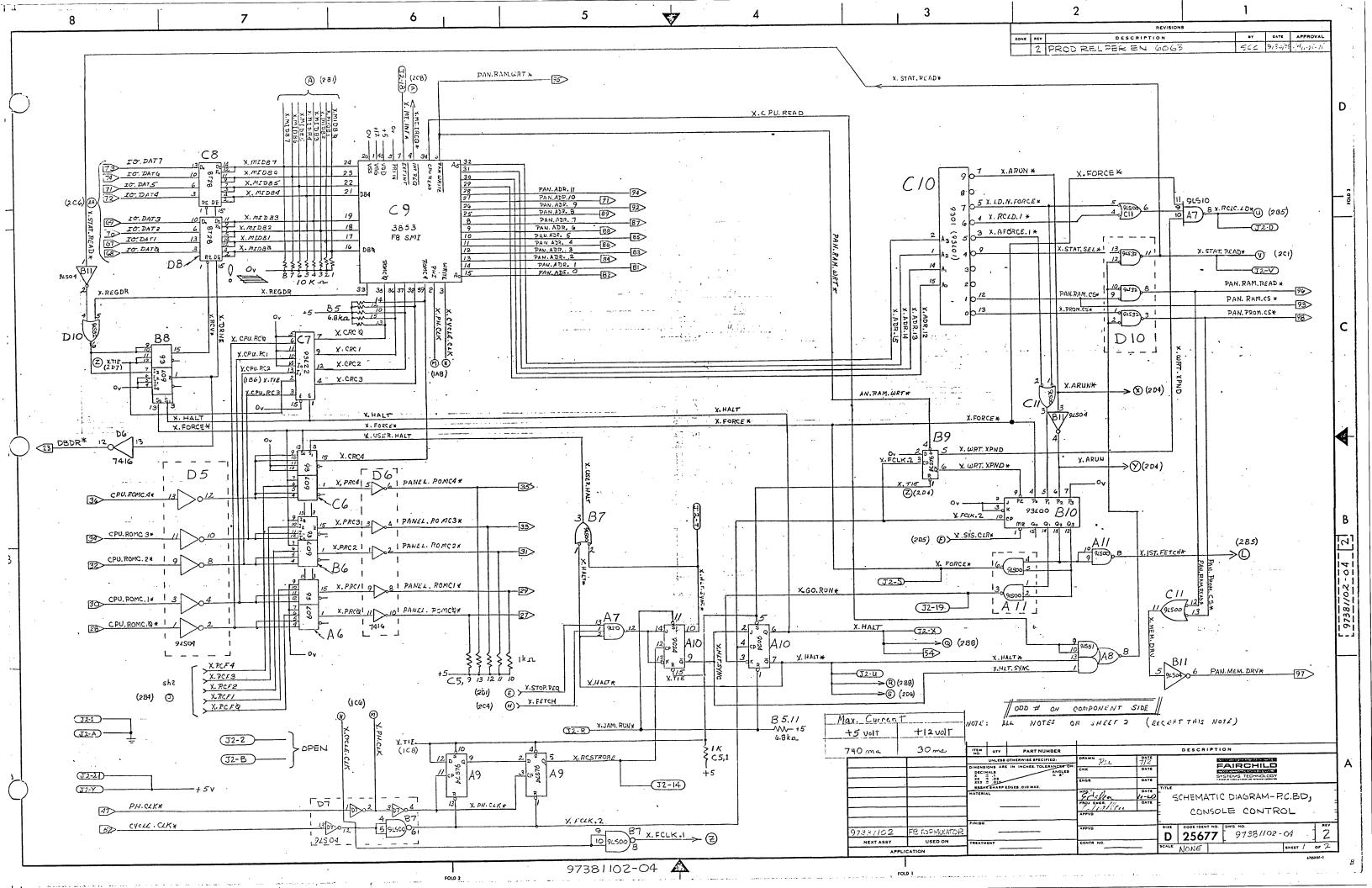
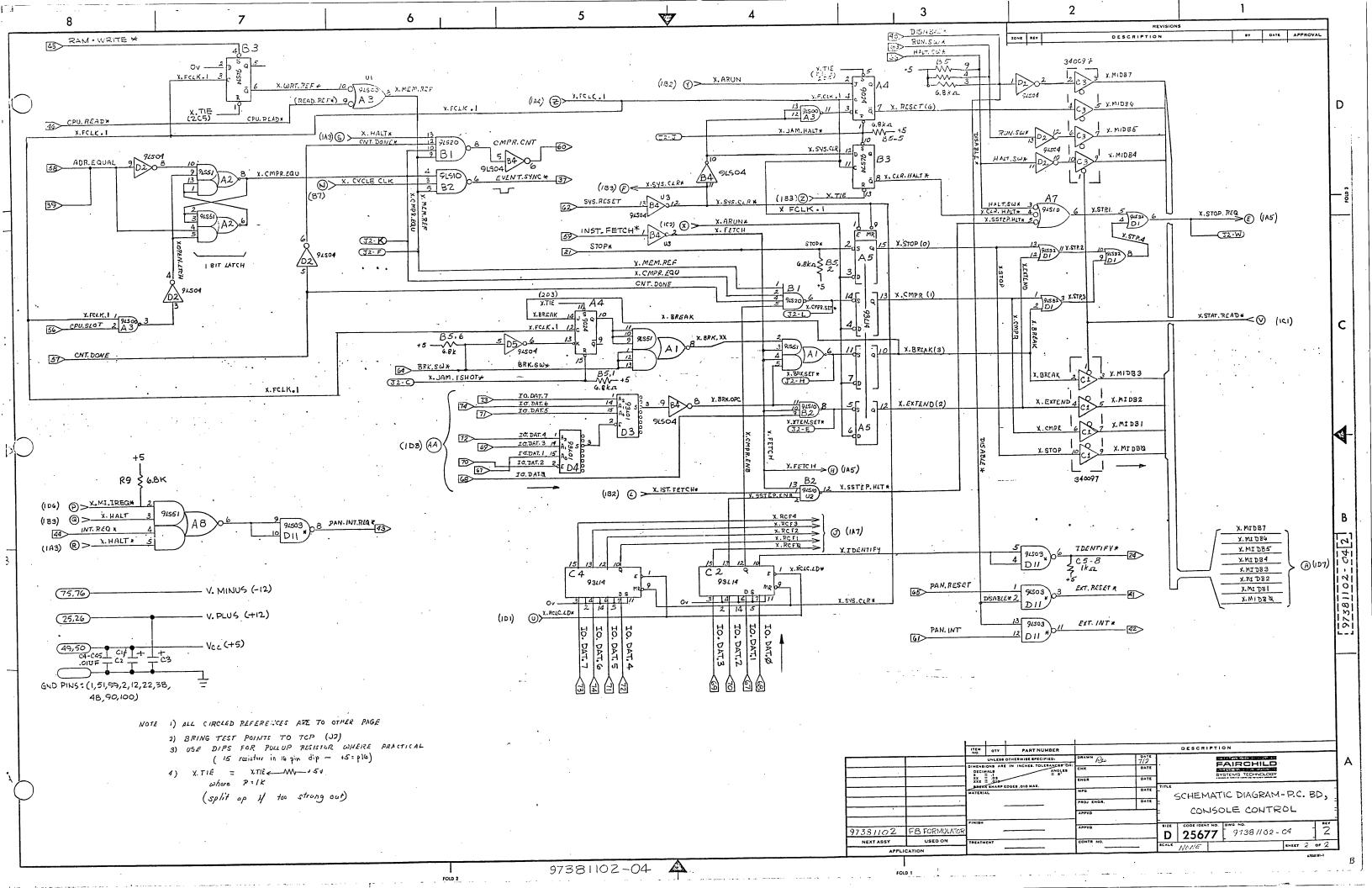


Figure 15. BREAK timing









Theory of Operation: CONSOLE MEMORY BOARD (Assembly No. 97381103)

#### Y.1 INTRODUCTION

The console memory board is one board of a set of three that perform the console function in the F8 FORMULATOR. The console memory board supplies the program storage, the data storage memory, and the I/O ports for the console microprocessor; on the board are 4K bytes of PROM, 1K byte of 2102 RAM, and two 8-bit F8 I/O ports.

### Y.1.1 Functional Description

PROM:

The 4K bytes of PROM hold the console microprocessor program. Instructions will be fetched from the PROM while the system is halted; the address and memory control come from the console's memory interface chip (3853) that lives on the console control board. The instructions fetched from the PROM move over the IO.DATA bus. There are sixteen PROMs in sets of two; the PROMs are 512x4 bipolar fusable link devices. The PROMs are held in sockets.

RAM:

The 1K bytes of RAM are used by the console microprocessor to save user's registers, such as his program counter, and to hold console flags while the system is running. The RAM is also controlled by the console MI and connects to the IO.DATA bus. The RAM is static 2102 devices.

I/0:

An F8 3851 PSU is used to provide 16 bits of I/O. The I/O bits control the console display board, connecting the operator's controls and indicators into the console microprocessor. The PSU connects to the IO.DATA bus. The ROMC lines that control the PSU are gated so that the PSU will be active only during I/O instructions executed while the system is halted.

### Y.2 DETAILED THEORY OF OPERATION

Four groups of lines are prominent that connect the console control board to other boards: The IO.DATA bus that connects

to the data bus of the CPU, the 16 bits of display data and control that connect to the console display, the CPU.ROMC bus that controls the 3851 PSU, and the 12 bits of address and five memory control lines that drive the memory. The bidirectional IO.DATA bus is buffered [D8]\* to become the bidirectional PANEL.DATA bus (Y.PDBx).

### Y.2.1 4K PROM Memory

The 16 PROM devices all have nine bits of address bused to them; the addresses are buffered to increase drive capability. Each PROM drives 4 bits of the PANEL DATA BUS (Y.PDBx) through a tri-state buffer [D4]; the buffer has some value during troubleshooting but is otherwise redundant. The necessary eight chip selects are created by a 1-of-8 decoder [B2] that accepts address bits 09 to 11. The signal PAN.PROM.CS [A1] controls the chip select decoder and the tri-state buffering; it comes from the console control board and is derived there from a decode of the four highest address lines ANDed with CPU.READ.

### Y.2.2 <u>1K RAM Memory</u>

ř.

The 8 RAM devices are 2102 MOS memories; the bit organization is 1024x1. Ten bits of address are connected to them; the address comes from the console memory interface (3853SMI) on the console control card. Read-write control and a single ship select line are brought in from the console control board. The write data for the RAMs comes directly from the PANEL.DATA bus. The data read from the RAMs feeds the PANEL.DATA bus through tri-state buffers [C5]; pull-up resistors are necessary to pull the RAM output high level up to CMOS requirements. The tri-state buffer is not redundant in this case; the PAN.RAM.CS signal can allow the slower MOS RAMs to begin their access early in the cycle because the tri-state buffer isolates the RAMs from the PANEL.DATA bus.

PAN.RAM.CS is just a decode on the console control board of A12 through A15; PAN.RAM.READ is the AND of PAN.RAM.CS with CPU.READ.

## Y.2.3 ROMC Gating

The ROMCs are gated before being fed to the 3851 PSU. The function of the gating is to block all ROMCs that would ask the PSU to drive the PANEL.DATA bus, except for the ROMC that says "input" (read from an I/O port into the CPU); the blocking is necessary because otherwise the PSU, which has a program in its ROM, will insist on spitting it out onto the

\*Numbers in brackets are coordinates on the schematic of the console memory board, Assembly No. 97381103-04.

data bus, interferring with the desired console microprocessor program. All the ROMCs are blocked while the system is running so that this PSU, which belongs to the console microprocessor, will not interfere with the user microprocessor. (The console and user system share the IO.DATA bus and the F8 CPU.) The blocking is done by a tri-state buffer [B7]; whenever its outputs are in the high impedance state, resistors pull the lines to the NO-OP ROMC state--H'IC'. There are 32 possible ROMC states. All those that must be blocked are in the first sixteen except for H'II'; the two input-output ROMCs, H'IA' and H'IB', are in the second sixteen. The three input HAND gate [A7] blocks the first sixteen ROMCs and blocks eight of the upper sixteen including H'II'; the IO ROMCs are passed. The other six ROMCs that are passed harmlessly alter the unused registers inside the PSU. The NAND gate blocks all ROMCs when the system is not halted.

#### Y.2.4 PSU IO Ports

The IO ports are used on the PSU. The ROM storage and the program registers--PCO, PCl, and DCO--are disabled by the blocking of the ROMCs. The interrupt structure is disabled by tieing PRIORITY.IN inactive (+5V). Each of the two IO ports drives a bus; the buses control the display and switch matrix that are on the console display board. One I/O port has the control signals (CNTRL.DATxx) and is used only for output. The other port is used bidirectionally to output and to receive data bytes over the DISP.DATxx bus [C8].

### Y.2.5 <u>IO.DATA Bus Drive/Receive</u>

The PANEL.DATA bus is buffered from the bidirectional IO.DATA bus by 8T28 transceivers [D8]; the buffers provide increased drive and lowered loading of the IO.DATA bus. The drive/receive state of the buffers is controlled by a 3-input NAND that is doing an OR function [D6]. The drivers will be turned on to drive onto the IO.DATA bus if the PSU has been selected by an input instruction or if the signal PAN.MEM.DRV, which comes from the console control board, is active [D8].

PAN.MEM.DRV will be made active by the console control board if either the RAM or PROM of this board is being accessed.

