

+-----+ HEWLETT
| hp |
+-----+ PACKARD

HP-DIO FOUR CHANNEL TERMINAL MULTIPLEXER

PROJECT FORDYCE - FIRMWARE

EXTERNAL REFERENCE SPECIFICATION (ERS)

HEWLETT-PACKARD COMPANY
Roseville Division
8000 Foothills Blvd.-
Roseville, California 95678

August 6, 1984

Table of Contents

1	INTRODUCTION	1
1.1	FEATURES	1
1.2	OTHER FEATURES (NOT OFFICIALLY SUPPORTED)	2
2	MEMORY ADDRESS SPACE	3
2.1	RAM MAP	4
3	OVERVIEW OF SHARED MEMORY SCHEME	6
3.1	PASSING DATA BETWEEN THE CARD & HOST	6
3.1.1	RECEIVE DATA FROM CHANNEL	7
3.1.2	TRANSMIT DATA TO THE CHANNEL	7
4	HARDWARE CONSIDERATIONS and DEFAULT SETTINGS	8
4.1	CTC TIMERS	8
4.2	SIO OUTPUT LINES	9
4.3	FIRMWARE PRIORITY SCHEME	9
4.4	TEST HOODS	10
4.5	DEFAULT LINE CHARACTERISTICS AND FORMAT	10
4.6	DEFAULT BIT MAP	11
4.7	DEFAULT TIMER SETTING	11
5	INTERFACE REGISTERS	12
5.1	HARDWARE REGISTERS	12
5.1.1	RESET/I.D. REGISTER	12
5.1.2	INTERRUPT REGISTER	13
5.1.3	SEMAPHORE REGISTER	14
5.2	REGISTERS WITH INTERRUPT CAPABILITIES	15
5.2.1	COMMAND Register	15
5.2.2	INT-COND Register	16
5.3	SPECIAL CHARACTER BIT MAP TABLE	18
5.4	OTHER SHARED MEMORY REGISTERS	18
5.4.1	RECEIVE FIFO HEAD POINTERS	20
5.4.2	RECEIVE FIFO TAIL POINTERS	20
5.4.3	TRANSMIT FIFO HEAD POINTERS	20
5.4.4	TRANSMIT FIFO TAIL POINTERS	20
5.4.5	CONFIGURATION DATA REGISTERS	20
5.4.6	MODEM INPUT LINES	22
5.4.7	MODEM OUTPUT LINES	22
5.4.8	MODEM MASK	23
5.4.9	INT-COND REGISTER INTERRUPT DATA (ICR-TAB)	23
5.4.10	COMMAND REGISTER INTERRUPT DATA (CMND-TAB)	23
5.4.11	SELF TEST RESULT REGISTER	23
6	SHARED MEMORY ACCESS AND DATA FORMATS	25
6.1	BUS ARBITRATION	25
6.2	POINTER MANAGEMENT - RECEIVE AND TRANSMIT BUFFERS	25
6.3	RECEIVE DATA FORMAT	26

6.4	RECEIVE FIFO BUFFER MANAGEMENT	27
6.4.1	TIME-OUT TIMER FOR RECEIVE CHARACTERS	27
6.4.2	CARD PROCESS FOR RECEIVE BUFFER MANAGEMENT	27
6.4.3	HOST PROCESS FOR RECEIVE BUFFER MANAGEMENT	28
6.5	TRANSMIT DATA FORMAT	28
6.6	TRANSMIT FIFO BUFFER MANAGEMENT	28
6.6.1	CARD PROCESSING FOR TRANSMIT BUFFER MANAGEMENT	29
6.6.2	HOST PROCESSING FOR TRANSMIT BUFFER MANAGEMENT	29
7	INTERRUPTS	30
7.1	INTERRUPT SENDING AND RECEIVING - SEMAPHORE REGISTER	30
7.2	HOST-TO-CARD INTERRUPTS	31
7.3	CARD-TO-HOST INTERRUPTS	32
8	MODEM SUPPORT	35
9	SELF TEST	36
9.1	VALUE OF ST-COND REGISTER UPON SELF TEST FAILURE	37
10	PSEUDOCODE OUTLINES OF BUFFER ACCESS ROUTINES	38
10.1	RECEIVE BUFFER ACCESSING	38
10.2	TRANSMIT BUFFER ACCESSING	39

The HP-DIO Four Channel Multiplexer is a microprocessor based (Z-80), four channel, asynchronous interface for the HP-DIO backplanes. Three of the channels are direct connect ports. The fourth is a modem port which may be used as a direct connect port. This product will be referred to as FORDYCE throughout this document.

Although the card has a processor on board, the card is basically dumb from an external viewpoint. The card does very little on-board character processing. The main function of the firmware is to manage the FIFO buffers in which the data will be passed between the card and the host.

The purpose of this document is to outline the preliminary design of the firmware for the FORDYCE card. The following areas will be covered:

1. Overview
2. Hardware Considerations & Default Settings
3. Interface Registers
4. Shared Memory Access and Data Formats
5. Interrupts
6. Self Test
7. Pseudocode Outlines of Buffer Access Routines

This document is for HP internal use only.

1.1 FEATURES

*EIA RS-232-C and CCITT V.28 compatibility

*Special character recognition

*Break detection

*Break generation via host command

*The card will generate an interrupt to the host every 16 millsecs.

*128 character receive buffers and 16 character transmit buffers for each of the four ports

*Supported baud rates: 110, 134.5, 150, 300, 600, 1200, 2400, 4800, 9600, or 19.2k

*Baud rate defaults to 9600 and is software programmable to any of 16 rates

*Parity checking: odd, even, or none

*Number of Stop bits: 1 or 2

*Number of Data bits per character: 7 or 8

*Transmission mode: Full-duplex

1.2 OTHER FEATURES (NOT OFFICIALLY SUPPORTED)

The following features exist on the card but are deemed to be important enough to be given official HP support and all that entails.

If invoked, the card will attempt to enable the desired feature, but no warranty is expressed or implied.

*Additional BAUD rates: 50, 75, 900, 1800, 3600, 7200, and 38400

*Additional character lengths of 5 and 6 bits

*1.5 stop bits

The card contains a total of 2K of shared RAM and 8K of ROM. However, the Z-80 has an address space of 64K bytes. The following diagram illustrates the practical division of this address space on the FORDYCE card.

Chipmunk Address (HEX)		Z80 Address (HEX)	
FFFF		FFFF	
	UNUSED RAM SPACE		
9001		C800	
	SHARED RAM	C7FF	
8FFF			
8005		C002	
	Command Reg.	C001	
8003		C000	
8001	Int-Cond Reg.		
	UNUSED REGISTER SPACE	BFFF	
7FFF			
0007		8003	
	Semaphore Reg.	8002	
0005		8001	
0003	Interrupt Reg.	8000	
0001	Reset/I.D. Reg		
	UNUSED PROGRAM SPACE	2000	
N/A			
	8K EPROM PROGRAM SPACE	1FFF	
N/A		0	

---2K RAM

--- SHARED
MEMORY
SPACE

MAP OF ENTIRE
FORDYCE MEMORY SPACE

2.1 RAM MAP

The following map displays the organization of the 2K of shared RAM on the card.

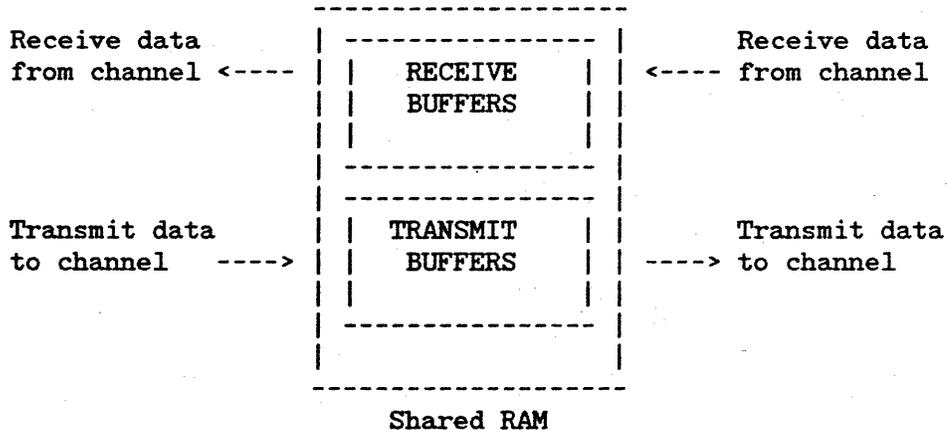
HOST ADDRESS (hex)		Z-80 ADDRESS (hex)
8FFF		C7FF
8F61	STACK - 80 BYTES	C7B0
8F5F	TRANSMIT 16 BYTES	C7AF
8F41	FIFO - PORT 0	C7A0
8F3F	TRANSMIT 16 BYTES	C79F
8F21	FIFO - PORT 1	C790
8F1F	TRANSMIT 16 BYTES	C78F
8F01	FIFO - PORT 2	C780
8EFF	TRANSMIT 16 BYTES	C77F
8EE1	FIFO - PORT 3	C770
8EDF	SHARED RAM REGISTERS	C76F
8E01	& CONFIG. DATA	C700
8DFF	BIT MAP - 256 BYTES	C6FF
8C01		C600
8BFF	RECEIVE 256 BYTES	C5FF
8A01	FIFO - PORT 0	C500
89FF	RECEIVE 256 BYTES	C4FF
8801	FIFO - PORT 1	C400
87FF	RECEIVE 256 BYTES	C3FF
8601	FIFO - PORT 2	C300
85FF	RECEIVE 256 BYTES	C2FF
8401	FIFO - PORT 3	C200
83FF	SCRATCH 510 BYTES	C1FF
8005	variables - card only	C002
8003	COMMAND REG.	C001
8001	INT-COND REG.	C000

RAM map

Data will be passed between the card and the host in circular FIFO data buffers. There are a total of eight of these buffers; four Receive buffers (one for each port) and four Transmit buffers (one for each port). These buffers will be accessed by both the card and the host. The shared memory scheme gives rise to four basic types of memory accesses (illustrated below). The handshaking between the driver and the firmware on the card must be coordinated to allow all four to function in an acceptable manner for the speed and priority constraints.

HOST BUFFER ACCESS

CARD BUFFER ACCESS



3.1 PASSING DATA BETWEEN THE CARD & HOST

The following is a general overview of how transmit and receive data is passed between the card and the host.

3.1.1 RECEIVE DATA FROM CHANNEL

Receive data processing is divided into two basic parts: Putting data into the buffers (card processing) and removing data from the buffers (host processing).

When a character arrives, the card will retrieve it from the UART, and check the Bit Map to see if it is to be processed as a special character. If so, the card will send a Special Character interrupt to the host. The card will then write the character to the appropriate Receive buffer.

Each character written to the Receive buffers will have an accompanying status byte. The status byte will indicate whether a framing error, parity error, overrun error, FIFO overflow error or break occurred for the character. If no error or break occurred, the status byte will contain a zero.

The host will check the Receive buffers for data each time it receives a Timer interrupt. The card will send the host a Timer interrupt every 16 milliseconds regardless of the state of the receive buffers. The host will check each buffer and empty all data it finds.

3.1.2 TRANSMIT DATA TO THE CHANNEL

When the host has transmit data to send it first checks the Transmit buffer. If full, the host must back off and wait for a TX Buffer Empty interrupt from the card. If the buffer is not full, the host will place characters in the buffer until either the buffer is full or the host is done. If the host put characters into a buffer that was empty, the host sends a TX Buffer Not Empty interrupt to the card indicating that there is now data in the buffer.

When the card receives the TX Buffer Not Empty interrupt from the host, it will begin to empty the Transmit buffer. When the card has finished, it will send a TX Buffer Empty to the host. If the host does not have data to send, it will simply ignore the interrupt.

The FORDYCE card consists of a Z-80A microprocessor, 2K of RAM (shared), 8K of ROM, two Z-80 CTC's (Counter Timer Clocks), and two Z-80 SIO/2 chips (UARTs). There are 8 DIP switches on the card.

DIP SWITCH	USED FOR	DEFAULT SETTING
0-4	SELECT CODE	13 (DECIMAL)
5,6	CARD INTERRUPT PRIORITY	3 (HIGHEST)
7	CONSOLE CONNECTION - YES OR NO	1 (YES)

4.1 CTC TIMERS

As mentioned previously, there are two CTC chips on the FORDYCE card. Each chip has four counter/timer channels for a total of 8 available on the card. Four of these are used as baud rate generators and one is used as a receive buffer time-out timer. The remaining three channels are unused.

CTC CHANNEL	USED FOR
CTC 0 CHANNEL 0	BAUD RATE GENERATOR FOR PORT 0
CHANNEL 1	BAUD RATE GENERATOR FOR PORT 1
CHANNEL 2	HOST INTERRUPT LINE
CHANNEL 3	UNUSED
CTC 1 CHANNEL 0	BAUD RATE GENERATOR FOR PORT 2
CHANNEL 1	BAUD RATE GENERATOR FOR PORT 3
CHANNEL 2	TIME-OUT TIMER FOR RX BUFFERS
CHANNEL 3	UNUSED

4.2 SIO OUTPUT LINES

There are two SIO chips, each of which has two channels and two sets of modem lines. The following is a summary of the uses of the modem lines. The SYNC lines are included because one will be used as a modem line.

	SIO LINE	MODEM SYMBOL	USED AS
	-----	-----	-----
SIO 0 CHANNEL A (PORT 0)	RTS	RS	REQUEST TO SEND - OUTPUT
	DTR	TR	TERMINAL READY - OUTPUT
	CTS	CS	CLEAR TO SEND - INPUT
	DCD	RR	RECEIVER READY - INPUT
	SYNC	DM	DATA MODE - INPUT
SIO 0 CHANNEL B (PORT 1)	RTS	SR	SIGNAL RATE SELECTOR - OUTPUT
	DTR		UNUSED
	CTS		HOOD DETECT - PORT 1
	DCD	IC	INCOMING CALL - INPUT
SIO 1 CHANNEL A (PORT 2)	RTS		UNUSED
	DTR		UNUSED
	CTS		HOOD DETECT - PORT 2
	DCD		UNUSED
	SYNC		UNUSED
SIO 1 CHANNEL B (PORT 3)	RTS		ENABLE FRONTPLANE DRIVERS
	DTR		UNUSED
	CTS		HOOD DETECT - PORT 3
	DCD		UNUSED

NOTE: To detect a hood for Port 0, the firmware will first clear the TR line, then perform a loopback check on the IC and SR lines (These two are looped together in the modem port test hood).

4.3 FIRMWARE PRIORITY SCHEME

All firmware events will be interrupt driven. When the Z-80 is executing an Interrupt Service Routine, interrupts will be disabled to prevent another interrupt from preempting the current routine.

Therefore, the priority of the interrupts is dependent upon the priority of the SIO and CTC channels and their placement on the interrupt daisy chain. The following is a list of the major firmware events in order of their priority. (high to low)

1. RECEIVE DATA - PORT 0
2. TRANSMIT DATA - PORT 0
3. MODEM LINE CHANGES - CS, DM, and RR
4. RECEIVE DATA - PORT 1
5. TRANSMIT DATA - PORT 1
6. MODEM LINE CHANGES - IC
7. RECEIVE DATA - PORT 2
8. TRANSMIT DATA - PORT 2
9. RECEIVE DATA - PORT 3
10. TRANSMIT DATA - PORT 3
11. TIMER INTERRUPTS
12. HOST INTERRUPTS

4.4 TEST HOODS

The FORDYCE card physically has three direct connect RJ-11 ports and one 25 pin standard modem port. As mentioned previously, the modem port may also be used as a direct connect port.

There will be two types of test hoods for the FORDYCE card, one for the direct connect port and one for the modem port. The test hoods for the direct connect ports consist of an RJ-11 plug with the wires looped back. The test hood for the modem port will be a standard 25 pin modem connector with the output modem lines connected to the input modem lines.

Any number of test hoods may be connected to the board for Self Test. In other words, during Self Test, the firmware will check each port for a test hood. If no test hood is present on a particular port, the Self Test will simply skip the external loopback test for that port.

4.5 DEFAULT LINE CHARACTERISTICS AND FORMAT

When the card powers up, it will set up the UARTs with the default line characteristics. The host will be able to change these after Self Test and Initialization. The following is a list of each line characteristic and its default value. The default line

characteristics will be the same for each port.

1. SPEED - 9600 BAUD
2. NUMBER OF STOP BITS - 1
3. PARITY - NONE
4. NUMBER OF BITS PER CHARACTER - 8

4.6 DEFAULT BIT MAP

After card initialization, the Bit Map will be cleared (i.e. all locations = 0). In other words, the card will not be set to recognize any character.

4.7 DEFAULT TIMER SETTING

The 16 millisecond timer will be off after power up and card initialization. The host is responsible for enabling the timer.

Interface Registers are the Hardware Registers and RAM locations (also called registers) that are accessed by both the card and the host. All of the communication between the card and the host will be performed by passing information between these registers. Although the FIFO buffers could also be included in this category, they will be discussed in another chapter since they involve special handshaking.

It should be noted that although technically speaking, all of the RAM on the card can be accessed by both the card and the host, there is a portion that is reserved for the card use only. Since there is no hardware protection mechanism for this portion of RAM, the host will have to be careful not to access those locations.

5.1 HARDWARE REGISTERS

The following is a brief description of the hardware registers on the FORDYCE card. These registers are:

1. Reset I.D. register
2. Interrupt register
3. Semaphore register

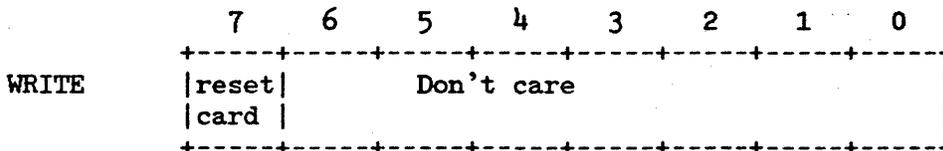
5.1.1 RESET/I.D. REGISTER

Z-80 ADDRESS: 8000H
MAINFRAME ADDRESS: 0001H

This register is used to reset the card and to contain the card identification information. On the FORDYCE card, using this register to reset the card causes a Non Maskable Interrupt (NMI) to the Z-80. The NMI in turn causes a fetch at location 66 in ROM which contains a jump instruction to the Initialize routine. At the end of the initialize code is the wait loop the card performs while waiting for interrupts. In other words, on the FORDYCE card, a card reset using

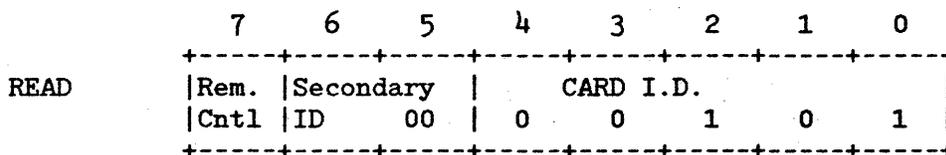
this register will reinitialize the card but will NOT return to the code that was being executed at the time the NMI was issued.

In the following diagrams, the first figure shows the definition of the bit locations when a write is issued to this register. The second shows the bit definitions when a read is issued. Both the card and the host will have occasion to write to this register. However, only the host will have need to read it.



Bit 7: When set (1) the card is RESET and a nonmaskable interrupt is generated to the Z-80. This causes a jump to location 066H in ROM which is the beginning of the Z-80 initialization code. This bit must be cleared before another RESET can be issued.

Bits 0-6: Not defined



Bit 7: This bit is set or reset by the console DIP switch discussed in the last chapter. When this bit is set it indicates that there is a system console hooked up to this card.

Bits 5,6: These bits constitute the cards' Secondary I.D. These bits are hardwired to 0.

Bits 0-4: These bits form the unique I.D. code of this card. The FORDYCE card I.D. is 5 and so these bits are hardwired as shown in the figure above.

5.1.2 INTERRUPT REGISTER

Z-80 ADDRESS: 8001H

MAINFRAME ADDRESS: 0003H

This register is used to enable interrupts to the host and to reflect the interrupt priority of the card. After card

initialization the card will not access the interrupt register again. The host will write to bit 7 when it wants to enable or disable interrupts.

	7	6	5	4	3	2	1	0	
WRITE	Int.	Don't care							
	Enbl.								

Bit 7: This bit enables and disables card interrupts to the host. When set (1), interrupts are enabled. When reset (0), interrupts to the host are disabled.

Bits 0-6: Not defined

	7	6	5	4	3	2	1	0
READ	Int	Int	Interrupt		Undefined for this			
	En'd	Rqst.	Level		card			

Bit 7: This bit indicates the current status of the host interrupt enable flip flop ('1'=enabled, '0'= disabled)

Bit 6: This bit is set when the card is requesting an interrupt and reset when its not.

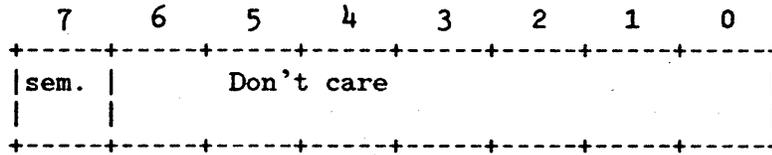
Bits 4-5: These bits indicate the interrupt level of this card. The interrupt level is set by the two interrupt DIP switches.

Bits 0-3: These bits are not defined for this card although they are defined for DMA on other DIO cards.

5.1.3 SEMAPHORE REGISTER

Z-80 ADDRESS: 8002H
HOST ADDRESS: 0005H

The semaphore register will be used by both the card and the host while sending and servicing interrupts generated by the interrupt registers (the INT-COND and COMMAND registers). The following is a description of the semaphore register and an explanation of its use.



Bit 7 - This bit gives the status of the semaphore: '0'=not busy, '1'=busy. The semaphore is automatically set after it is read.

Bits 0-6 - These bits are not defined.

This register is used by the card and the host to determine whether the shared RAM is currently available for access. The semaphore register performs an indivisible read and set operation. When either the host or the card reads this register, bit 7 is set to indicate that a memory access is in progress. When the access is completed, the semaphore register can be cleared by writing any value to it. Bits 0 to 6 are meaningless.

It should be noted that the Semaphore register does not perform any hardware lockout function. Its use is part of the backplane protocol. The semaphore register will only be used when either the card or the host wants to access one of the interrupt registers (the INT-COND and the COMMAND registers).

5.2 REGISTERS WITH INTERRUPT CAPABILITIES

There are two RAM registers which are capable of generating an interrupt when they are written to. These registers are used to send status and command information between the card and the host. Most of the software interfacing between the card and the host will be initiated through these registers. The following is a description of each. For further information on the interrupts each of these registers can generate, refer to Chapter 6, INTERRUPTS.

5.2.1 COMMAND Register

Z-80 ADDRESS: C001H
 HOST ADDRESS: 8003H

WRITE: HOST ONLY - GENERATES INTERRUPT TO THE Z-80

READ : CARD ONLY - TURNS OFF INTERRUPT

This register is used to send commands and status information from the host to the card. When the host writes to this register, an interrupt to the Z-80 is generated. The interrupt informs the card that there is a command to be read in the COMMAND register. When the card reads the register, the interrupt line is automatically cleared.

The bits in the COMMAND register are used to identify the type of interrupt request. There are two types of interrupts generated by the host; port specific interrupts and non-specific interrupts. If the interrupt is port specific, i.e. it pertains to a particular port, a bit will be set in the COMMAND register to indicate which port. The actual interrupt information will be contained in a 4 byte table called the CMND-TAB. This table will be discussed in detail in Chapter 7.

Since non-specific interrupts do not concern a particular port, there is a bit reserved for them in the COMMAND register. The CMND-TAB is not accessed.

7	6	5	4	3	2	1	0
NOT	SELF	TIMER	MODM	PORT	PORT	PORT	PORT
USED	TEST	ON/OFF		3	2	1	0

COMMAND REGISTER

BIT 0-3: A '1' in any of these bit positions indicates that there is a port-specific interrupt for that port. The card will check the correct byte in the CMND-TAB to identify the interrupt.

BIT 4 : A '1' in this bit position indicates that the host wants to change one of the modem lines. The card will access the MODM-OUT register to determine which line to change.

BIT 5 : A '1' in this bit position indicates that the host wants to turn off or on the 16 millisecond timer.

BIT 5 : A '1' in this bit position indicates that the host wants the card to perform its self test.

5.2.2 INT-COND Register

Z-80 ADDRESS: C000H
HOST ADDRESS: 8001H

WRITE: CARD ONLY - GENERATES INTERRUPT TO HOST

READ : HOST ONLY - CLEARS INTERRUPT

The INT-COND register is used to send status information and messages from the card to the host. When the card writes to this register, an interrupt to the host is generated. The interrupt informs the host that there is a interrupt to be read in the INT-COND register. When the host reads the register, the interrupt line is automatically cleared. The bits in the INT-COND register are used to identify the type of interrupt request.

As with the COMMAND register, there are two types of interrupts generated by the card; port specific interrupts and non-specific interrupts. If an interrupt is port-specific, a bit will be set in the INT-COND register to indicate which port the interrupt involves. The actual interrupt information will be contained in a 4 byte table called the ICR-TAB. This table will be discussed in detail in Chapter 6.

Since non-specific interrupts do not concern a particular port, there is a bit reserved for them in the INT-COND register. The ICR-TAB is not accessed.

7	6	5	4	3	2	1	0
NOT	TIME	MODM	ST	PORT	PORT	PORT	PORT
USED	OUT		DONE	3	2	1	0

INT-COND REGISTER

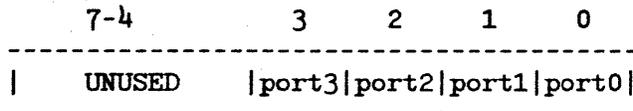
- BIT 0-3: A '1' in any of these bit positions indicates that there is a port-specific interrupt for that port. The card will check the correct byte in the ICR-TAB table to identify the interrupt.
- BIT 4 : This bit is set after the card has finished Self Test and card Initiation. This interrupt notifies the host that it may now communicate with the card.
- BIT 5 : A '1' in this bit position indicates that that a change occurred on one of the input modem lines. The host will access the MODM-IN register to determine which line changed.
- BIT 6 : A '1' in this bit position means that the 16 millisecond Receive buffer timer has gone off. The host will respond by retrieving any characters that are in the four Receive buffers.

5.3 SPECIAL CHARACTER BIT MAP TABLE

The Bit Map consists of 256 RAM locations, each byte representing one character. The first four bits in each byte correspond to the four ports on the card.

The purpose of the Bit Map is to enable the host to be notified immediately when a "special" character is received. The host defines a character as special by setting the bit representing the port, within the byte which represents the character.

When the card receives a character, it uses the character as an index into the Bit Map and checks the bit representing the port the character came from. If the bit is set, the card sends the host a Special Character interrupt. The following is an illustration of a Bit Map location.



5.4 OTHER SHARED MEMORY REGISTERS

There are a number of special purpose RAM locations which will be used to pass information between the host and the card. The following is a list of these shared RAM locations and a description of their usage.

RAM REGISTER	DESCRIPTION	Z-80 ADDRESS	HOST ADDRESS
RHEAD-0	RECEIVE FIFO HEAD POINTER - PORT 0	C700	8E01
RHEAD-1	" - PORT 1	C701	8E03
RHEAD-2	" - PORT 2	C702	8E05
RHEAD-3	" - PORT 3	C703	8E07
RTAIL-0	RECEIVE FIFO TAIL POINTER - PORT 0	C704	8E09
RTAIL-1	" - PORT 1	C705	8E0B
RTAIL-2	" - PORT 2	C706	8E0D
RTAIL-3	" - PORT 3	C707	8E0F

THEAD-0	TRANSMIT FIFO HEAD POINTER - PORT 0	C708	8E11
THEAD-1	" - PORT 1	C709	8E13
THEAD-2	" - PORT 2	C70A	8E15
THEAD-3	" - PORT 3	C70B	8E17
TTAIL-0	TRANSMIT FIFO TAIL POINTER - PORT 0	C70C	8E19
TTAIL-1	" - PORT 1	C70D	8E1B
TTAIL-2	" - PORT 2	C70E	8E1D
TTAIL-3	" - PORT 3	C70F	8E1F

CONFIGURATION DATA REGISTERS:

CONFIG-0	LINE SPECS REGISTER - PORT 0	C710	8E21
BD-0	BAUD RATE INDEX - "	C711	8E23
CONFIG-1	LINE SPECS REGISTER - PORT 1	C712	8E25
BD-1	BAUD RATE INDEX - "	C713	8E27
CONFIG-2	LINE SPECS REGISTER - PORT 2	C714	8E29
BD-2	BAUD RATE INDEX - "	C715	8E2B
CONFIG-3	LINE SPECS REGISTER - PORT 3	C716	8E2D
BD-3	BAUD RATE INDEX - "	C717	8E2F

MODM-IN	MODEM INPUT LINES	C718	8E31
MODM-OUT	MODEM OUTPUT LINES	C719	8E33
MODM-MASK	MODEM MASK FOR INPUT LINES	C71A	8E35
CMND-TAB	COMMAND REG. INTERRUPT DATA 4 BYTES - 1 PER PORT	C71B	8E37
ICR-TAB	INTCOND REG. INTERRUPT DATA 4 BYTES - 1 PER PORT	C71F	8E3F
ST-COND	SELF TEST RESULT REGISTER	C723	8E47

The following is a more detailed explanation of the uses of the Shared RAM registers.

5.4.1 RECEIVE FIFO HEAD POINTERS

These pointers contain the index of the current head of the Receive buffers. The receive buffer head pointers are updated by the host when it removes data from the Receive buffers.

5.4.2 RECEIVE FIFO TAIL POINTERS

These pointers contain the index of the current tail of the Receive buffers. The receive buffer tail pointers are updated by the card when it places new data into the Receive buffers.

5.4.3 TRANSMIT FIFO HEAD POINTERS

These pointers contain the index of the current head of the Transmit buffers. The transmit buffer head pointers are updated by the card when it removes data from the transmit buffers.

5.4.4 TRANSMIT FIFO TAIL POINTERS

These pointers contain the index of the current tail of the Transmit buffers. The transmit buffer tail pointers are updated by the host when it places new data into the Transmit buffers.

5.4.5 CONFIGURATION DATA REGISTERS

As shown above, there are two bytes of configuration data for each port. The first byte (CONFIG) is used to specify parity, bits per character, and number of stop bits per character for each port. The second byte, BD, contains a value which corresponds to the desired baud rate. Both registers are detailed in the following paragraphs.

CONFIG REGISTER

This register is used to specify three pieces of configuration

information; parity method, number of bits per character, and the number of stop bits per character. The options shown below are the only ones supported on the card. For example, the card can only support 5,6,7, or 8 bits per character.

7-6	5	4	3	2	1	0	
DONT CARE							
					0	0	- NO PARITY
					0	1	- ODD PARITY
					1	0	- EVEN PARITY
			0	0			- 1 STOP BIT/CHARACTER
			0	1			- 1-1/2 STOP BITS/CHARACTER
			1	0			- 2 STOP BITS/CHARACTER
0	0						- 5 BITS/CHARACTER
0	1						- 6 BITS/CHARACTER
1	0						- 7 BITS/CHARACTER
1	1						- 8 BITS/CHARACTER

BD REGISTER

This register is used to indicate the baud rate the host wants the port set to. The following is a list of the values which correspond to the available baud rates.

BD REGISTER VALUE (HEX)	BAUD RATE (BITS/SEC)
1	50
2	75
3	110
4	134.5
5	150
6	300
7	600
8	900
9	1,200
A	1,800
B	2,400
C	3,600
D	4,800
E	7,200
F	9,600
10	19,200
11	38,400

5.4.6 MODEM INPUT LINES

The first four bits in this register are used to represent the four input modem lines. The remaining four bits are unused. The host will read this register when it wants to know the status of the input modem lines - i.e. which ones are on and which are off. The card will update this register when it receives notification of an input modem line change from the UART. If one of these lines change, the card will access the MODM-MASK register to see if the host wants to be interrupted for a change on that particular line. MODM-IN will always contain a copy of the current status of the input modem lines.

MODM-IN REGISTER

7-4	3	2	1	0

DONT CARE	CS	DM	RR	IC

CS - Clear to Send
RR - Receiver Ready
IC - Incoming Call
DM - Data Mode

5.4.7 MODEM OUTPUT LINES

The first three bits in this register are used to represent the three output modem lines. The remaining five bits are unused. When the host wants to change a particular output modem line it will write to this register setting the appropriate bit position, and generate a Modem Output Change interrupt to the card. The MODM-OUT register will always contain the current status of the modem output lines.

MODM-OUT REGISTER

7-3	2	1	0

DONT CARE	SR	TR	RS

SR - Signal Rate Selector
TR - Terminal Ready
RS - Request to Send

5.4.8 MODEM MASK

The first four bits in this register correspond to the first four bits in the MODM-IN register. The other four bits will be set. If there is a change on one of the input modem lines, this register will be used by the card to determine whether the host wants to be interrupted. If the bit in MODM-MASK corresponding to the input line is set, the card will send the host a Modem Input Change interrupt. If the bit is reset, the card will not send an interrupt. The format of MODM-MASK exactly matches that of MODM-IN.

5.4.9 INT-COND REGISTER INTERRUPT DATA (ICR-TAB)

This is a 4 byte table which is used to identify port specific interrupts sent by the card to the host. This table is used in conjunction with the INT-COND register. Each of the bytes in the ICR-TAB is reserved for one of the 4 ports. When the card sends the host a port specific interrupt (one of the port bits (0-3) in the INT-COND register is set), the host will read the corresponding byte in ICR-TAB for the actual cause of the interrupt. See Chapter 7, INTERRUPTS for more detail on ICR-TAB.

5.4.10 COMMAND REGISTER INTERRUPT DATA (CMND-TAB)

This is also a 4 byte table. It is used the same as the ICR-TAB except that it identifies port specific interrupts from the host to the card. The CMND-TAB is used in conjunction with the COMMAND register. See Chapter 7 for more detail on CMND-TAB.

5.4.11 SELF TEST RESULT REGISTER

This register is used to indicate the result of Self Test. If Self Test passed, STCOND will contain the value OEOH. If Self Test failed, the ST-COND register contains the value of the IX register (internal to the Z-80) at the time of failure. This value indicates what routine the Self Test was executing when it failed. A list of of the Self Test routines and corresponding IX values can be found in Chapter 10, Self Test.

As previously mentioned, there is 2K bytes of shared RAM on the FORDYCE card. All of the communication between the card and the host will be via shared RAM (the INT-COND and COMMAND registers are part of the RAM address space). The purpose of this chapter is to describe shared memory access protocols. This includes a description of the FIFO buffers, receive and transmit data formats, FIFO pointer management, Transmit FIFO protocol, and Receive FIFO protocol.

6.1 BUS ARBITRATION

The host and the card will alternate RAM accesses when they both need the bus at the same time. This is accomplished in the hardware. As a result, if both the card and the host try to access RAM at the same time, the host will get the bus for one memory access, then the card. In a worst case situation the host and the card will have to wait one RAM cycle between each memory access.

There is one exception to the above. Both the card and the host will use the Semaphore register to lock each other out when sending or responding to card-to-host or host-to-card interrupts. These interrupt processes are critical regions for both the host and the card and, as such, will be protected by mutual use of the Semaphore register. The use of the Semaphore register in the interrupt processes will be described in more detail in Chapter 7.

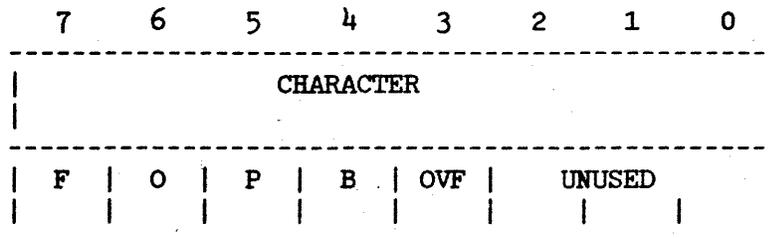
6.2 POINTER MANAGEMENT - RECEIVE AND TRANSMIT BUFFERS

As discussed previously, there are a total of eight buffers, each organized as a circular FIFO queue; one Receive buffer and one Transmit buffer for each of the four ports. There are two pointers associated with each of the buffers; a head pointer and a tail pointer. Both of these pointers will be indexes from a Base FIFO address. The base address will be hard-coded.

The management of the head and tail pointers is the responsibility of both the card and the host. The card will be adding data to the Receive buffers and removing data from the Transmit buffers. Therefore, it will be responsible for updating the Receive buffer Tail pointers and the Transmit buffer Head pointers. Conversely, the host will be removing data from the Receive buffers and adding data to the Transmit buffers. It will be responsible for updating the Receive buffer Head pointers and the Transmit buffer Tail pointers.

6.3 RECEIVE DATA FORMAT

The receive data format scheme requires two bytes per character. The first byte will be the character and the second byte will be the status byte which contains error information and break detection. The data format is illustrated below:



1. (F) Framing Error - This is to notify the host that a framing error occurred on this character.
2. (O) Overrun Error - This is to notify the host that a UART overrun condition occurred on this character.
2. (P) Parity Error - This is to notify the host that a parity error occurred on this character.
4. (B) Break Detection - This is to notify the host that a Break was received. The character will be null.
5. (OVF) Overflow Error - This is to notify the host that a Receive buffer overflow condition occurred before this character.

6.4 RECEIVE FIFO BUFFER MANAGEMENT

As discussed before, there are four receive buffers, one for each port. They are organized as circular FIFO data structures of 256 bytes each. As each character requires 2 bytes, this is enough buffer space for 128 characters per port.

6.4.1 TIME-OUT TIMER FOR RECEIVE CHARACTERS

As discussed previously, the card will interrupt the host every 16 milliseconds. The host will respond to this interrupt by emptying all of the characters in the four Receive buffers. The timer will cycle continuously whether there is Receive data in the buffers or not. The host does have the option to turn off the timer if it so desires by sending a Timer Off/On interrupt. This interrupt is discussed in more detail in Chapter 7.

6.4.2 CARD PROCESS FOR RECEIVE BUFFER MANAGEMENT

The card will only access the Receive FIFO buffers when a Receive character has arrived at a port. When a character arrives the following sequence of events is performed by the card.

1. Check if the buffer is full. If so, the card will simply exit this routine without retrieving the character from the UART.

NOTE: The UART has a three byte internal buffer which insures a little protection in the event the Receive buffer is full. However, if there is still no room in the Receive buffer when the fourth character arrives, the UART will overrun. It is the responsibility of the host to service the buffer enough to prevent this occurrence. There will be no overrun prevention done on the card.

2. Retrieve the character from the UART.
3. Strip any parity bits
4. Create the status byte
5. Check the Bit Map location for the character. If the correct bit is set, it identifies the character as a "special character" and

the card sends a Special Character interrupt to the host.

6. Write both the character and the status byte to the FIFO buffer and update the appropriate pointers.

6.4.3 HOST PROCESS FOR RECEIVE BUFFER MANAGEMENT

The host will only access the Receive FIFO buffers after it receives a Time-Out interrupt from the card. The Time-Out interrupt will occur every 16 milliseconds whether there is data in the Receive buffers or not. Upon receipt of the interrupt, the host will begin checking and emptying all four Receive buffers. The host will perform the following sequence of events for each Time-Out interrupt.

1. If head=tail then exit (buffer empty) else . . .
2. Retrieve data byte and status byte.
3. Update buffer pointers.
4. Begin sequence again.

6.5 TRANSMIT DATA FORMAT

There is really no transmit data format to speak of. As there is no status byte associated with transmit data, the transmit buffers will simply contain characters to transmit.

6.6 TRANSMIT FIFO BUFFER MANAGEMENT

As discussed previously, there are four transmit buffers, one for each port. They are organized as circular FIFO queues of 16 bytes each, one byte per character.

6.6.1 CARD PROCESSING FOR TRANSMIT BUFFER MANAGEMENT

The card begins to send transmit data out the port after it receives a TX Buffer Not Empty interrupt from the host informing it that the transmit buffer for the port is no longer empty. The card starts the UART and begins sending out characters. The card performs the following sequence of events.

1. If head=tail then exit (buffer empty) else. . .
2. Retrieve character and send to the UART.
3. Update necessary pointer(s).

6.6.2 HOST PROCESSING FOR TRANSMIT BUFFER MANAGEMENT

The host will add data to the Transmit buffers whenever it has the need unless the intended buffer is full. If the host encounters a full buffer, it will back off and wait for a TX Buffer Empty interrupt from the card. The TX Buffer Empty interrupt informs the card that there is now room in the Transmit buffer for more characters. The following is the sequence of events the host performs for each

If the buffer is empty when the host wants to put characters in the host will send the card a TX Buffer Not Empty interrupt. This interrupt tells the card that there are now more characters to send to the UART. The following is the sequence of events the host executes for each character it wants to place in a Transmit buffer.

1. Is the Transmit buffer full? If yes, exit routine.
2. Is the Transmit buffer empty? If yes, send a TX Buffer Not Empty interrupt to the card.
3. Do the following until either finished or buffer full
 - a. Put character into buffer
 - b. Update pointer.

This chapter will be divided into two general discussions. The first will be an overview of the Interrupt sending and receiving process between the card and the host. The second will be an explanation of each of the possible card-to-host and host-to-card interrupts.

NOTE: Both the host and the card will assume that there may be more than one bit set (more than one interrupt) in the interrupt register when the actual interrupt signal is received. This is the reason that a bit is reserved for each type of interrupt instead of using a value to represent a particular interrupt.

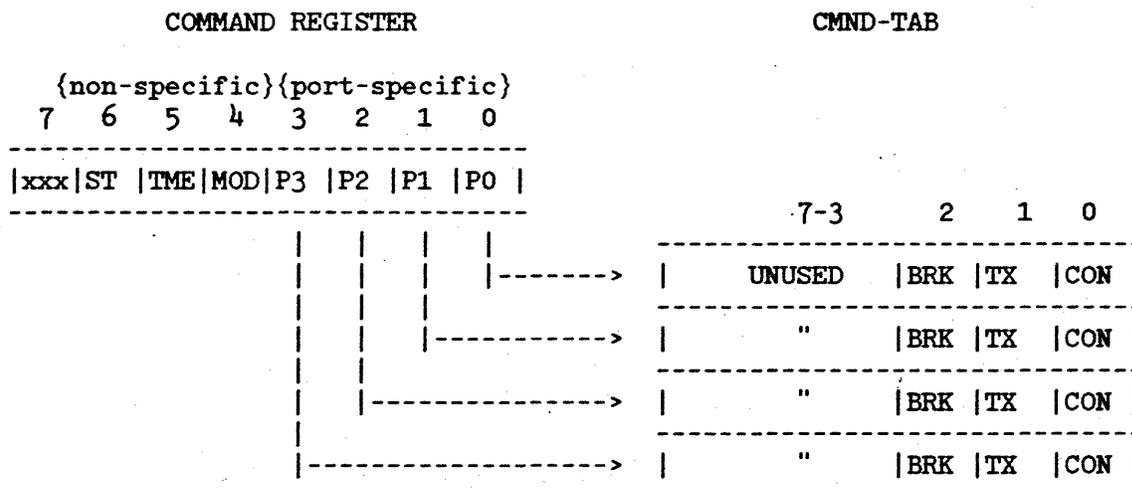
7.1 INTERRUPT SENDING AND RECEIVING - SEMAPHORE REGISTER

The interrupt process between the host and the card is critical to both and, as such, cannot tolerate a possible interleaving of memory accesses. As a result, the semaphore register will be used by both the card and the host as notification that a critical process is being performed.

Whenever either the host or the card is sending or receiving an interrupt, both will check the Semaphore register before accessing the interrupt register. If bit 7 in the Semaphore is "0", the interrupt registers are not being accessed and the side checking the register may proceed. If bit 7 is "1", the other side is in the critical region. The side wishing to begin must wait. For example, if the host wants to send an interrupt to the card, it will first check the Semaphore register. If bit 7 in the Semaphore register is set, the card is in the process of accessing one of the interrupt registers. The host will wait for the Semaphore register to be cleared before sending the interrupt.

7.2 HOST-TO-CARD INTERRUPTS

These interrupts are generated when the host writes to the COMMAND register. As mentioned previously, if the interrupt is port-specific, the bit in the COMMAND register indicating the port will be set and the bit in CMND-TAB indicating the interrupt will be set. In other words, if the interrupt is port-specific the card will check the corresponding byte in CMND-TAB for the interrupt. If the interrupt is not port-specific, CMND-TAB will not be accessed.



NON-SPECIFIC INTERRUPTS

1. MODEM OUTPUT CHANGE (MOD) - This interrupt is used in conjunction with the MODM-OUT register. The host will generate this interrupt when it wants the card to change one or more of the modem output lines. After receiving this interrupt, the card will read the MODM-OUT register and set the indicated modem lines.
2. TIMER OFF/ON (TME) - This interrupt is used to toggle the timer on and off. If the timer is on when this interrupt is received, the card will turn the timer off. If the timer is off, the card will turn it back on again.
3. SELF TEST ON (ST) - This interrupt tells the card to begin Self Test. The purpose of this interrupt is to give the host the capability of dynamically invoking Self Test without having to power the system down and back up.

NOTE: It is critical that the host does not interrupt the card after invoking Self Test until the card sends a Self Test Done interrupt.

PORT-SPECIFIC INTERRUPTS

1. CONFIGURATION DATA CHANGE (CON) - This interrupt informs the card that the host has changed the configuration data for the indicated port. Configuration data includes line characteristics and baud rate for the specified channels. The card will respond to this interrupt by changing the line configuration and baud rate as specified in the CONFIG and BD registers.

NOTE: The host waits for the TX buffer to be empty before sending the interrupt so that there is no timing collision.

2. TRANSMIT BUFFER NOT EMPTY (TX) - This interrupt tells the card that the host has put data into a previously empty Transmit buffer. Upon receipt of this interrupt, the card will start the UART and begin retrieving characters from the Transmit buffer.

NOTE: This interrupt is identified and processed after the configuration interrupt so that no transmit data is sent until the card has completely finished changing the line configuration for the port.

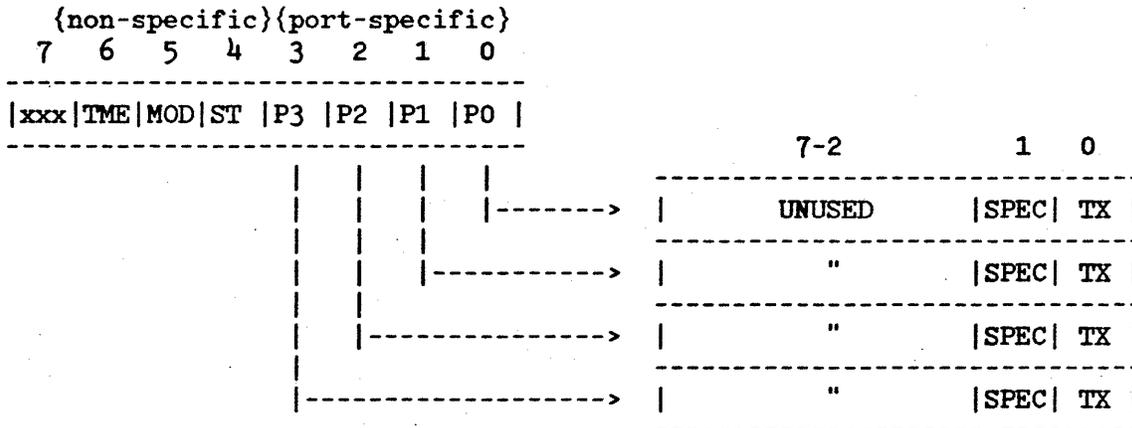
3. SEND BREAK (BRK) - This interrupt works as a toggle. The first time it is sent, it informs the card to send a break on the port specified. The break condition remains until the card receives this interrupt a second time which tells it to stop the break.

7.3 CARD-TO-HOST INTERRUPTS

These interrupts are generated when the card writes to the INT-COND register. As mentioned previously, if the interrupt is port-specific, the bit in the INT-COND register indicating the port will be set and the bit in ICR-TAB indicating the interrupt will be set. In other words, if the interrupt is port-specific the host will check the corresponding byte in ICR-TAB for the interrupt. If the interrupt is not port-specific, ICR-TAB will not be accessed.

INT-COND REGISTER

ICR-TAB



NON-SPECIFIC INTERRUPTS

1. **TIMER (TME)** - The card will send the host a Time-Out interrupt every 16 milliseconds. This signals the host to come retrieve any characters that might be in the Receive buffers. The host will respond to this interrupt by checking to see if the buffers are empty and retrieving all characters from the Receive buffers that are not empty.
2. **MODEM INPUT CHANGE (MOD)** - This interrupt is used in conjunction with the MODM-IN and the MODM-MASK registers. The card will send this interrupt to the host when there has been a change in one of the modem lines indicated by the MODM-MASK register. If there is a change in a modem line whose corresponding bit in the MODM-MASK is not set, the card will not issue this interrupt.
3. **SELF TEST COMPLETE (ST)** - This interrupt informs the host that the card has completed Self Test. The host will check the ST-COND register to determine whether Self Test passed or failed. If Self Test passed it also means that the card is initialized and ready for processing.

PORT-SPECIFIC INTERRUPTS

1. **SPECIAL CHARACTER RECEIVED (SPEC)** - This interrupt is sent when the card receives a character whose bit position in the Bit Map was set. Possible special characters might be XOFF, XON, etc. As mentioned previously, the host is responsible for designating which characters are special.

2. TRANSMIT BUFFER EMPTY (TX) - This interrupt informs the host that the Transmit Buffer for the port indicated is now empty. When the host wants to send the card a character but finds the Transmit buffer full, it will back off and wait for this interrupt before attempting to send any more characters.

FORDYCE supports full-duplex modem transmission. However, as this is a dumb card, the majority of the modem control will be the responsibility of the host. The firmware will only report changes in the input modem lines and set signals on the output modem lines per host request. The modem interface between the card and the host is limited to the MODM-IN and MODM-OUT registers which have been previously discussed in Chapters 5 and 7.

As the modem port can also be used as a direct connect port, it is the responsibility of the host to detect whether a modem is connected or not. The following modem lines will be supported:

SIGNAL CARD	DIRECTION DEVICE	MODEM LINE DESCRIPTION	EIA RS-449 SYMBOL
	---->	Send data	SD
	<----	Receive data	RD
	---->	Request to send	RS
	<----	Clear to Send	CS
	<----	Receiver ready	RR
	<----	Incoming call	IC
	---->	Terminal ready	TR
	<----	Data mode	DM
	---->	Signal rate selector	SR
	<----	Signal ground	SG

Self Test is the on-board diagnostic program which functionally tests all of the hardware on the card. It includes a ROM test, a RAM test, a CTC test, and a SIO test. There are two ways to invoke Self Test. It is automatically invoked during power up when the Auto Reset line is pulled and it may be invoked dynamically by a Self Test interrupt from the host.

WARNING: The host must NOT attempt to interrupt the card until it has received the Self Test Done interrupt from the card.

Upon successful completion of Self Test, the following will occur:

1. The ST-COND register contain the value OEOH to indicate that Self Test passed.
2. The card will send the host a Self Test Complete interrupt.
3. The card will execute the initialization routine.

Upon unsuccessful termination of Self Test, the following will occur:

1. The ST-COND register gets the value in the IX register. This value indicates where the test failed (see following section).
2. The card will send the host a Self Test Complete interrupt.
3. The host will display a message stating that Self Test on the card failed and give the number of the routine (in STCOND) in which the failure occurred (see the following section for the error numbers and their corresponding descriptions).
3. The card will execute the initialization routine.

NOTE: When the Self Test fails, the host will ignore the card that the failure occurred on. However, the card will still execute the initialization code. This is so the user may still access the card if it is deemed that the failure can be worked around. For example, if Self Test fails on the external loopback test for port 3, the user could still conceivably use the card, running the other 3 ports that passed. However, it should be noted that the Self Test checks each component in the order that they occur on the following page. If any test fails, Self Test quits executing and jumps to the initialization routine. Therefore, if the card fails the external loopback test for port 0, ports 1 through 3 have not executed the external loopback test. It cannot be assumed that they are safe to use.

9.1 VALUE OF ST-COND REGISTER UPON SELF TEST FAILURE

As stated above, when Self Test fails, a number representing the section of Self Test that failed is written to the ST-COND register. The following list defines each of these possible numbers. In the event of Self Test failure, the host can read the ST-COND register to determine where the failure occurred.

The ST-COND values are defined as follows:

ST-COND = 1: INT-COND/INTERRUPT register test
= 2: NMI/RESET I.D. register test
= 3: Semaphore register test
= 4: ROM Test
= 5: RAM Test
= 6: CTC 0 Test - ALG. 1
= 7: CTC 0 Test - ALG. 2
= 8: CTC 1 Test - ALG. 1
= 9: CTC 1 Test - ALG. 2
=10: SIO 0 CH A Test (Internal loopback)
=11: SIO 0 CH B Test (Internal loopback)
=12: SIO 1 CH A Test (Internal loopback)
=13: SIO 1 CH B Test (Internal loopback)
=14: SIO 0 CH A Test (with diag hood - external loopback)
=15: SIO 0 CH B Test (with diag hood - external loopback)
=16: SIO 1 CH A Test (with diag hood - external loopback)
=17: SIO 1 CH B Test (with diag hood - external loopback)

The purpose of this chapter is to define the FIFO buffer access routines for the card and the host in a pseudocode fashion. These pseudocode routines outline the general steps required to retrieve or place characters in the FIFO buffers. Since there is little difference in placing characters from different ports into their respective buffers, the following routines are not port specific. In other words, the same sequence of steps will be performed regardless of which port or which buffer is being serviced.

Most of the variable names used in the following pseudocode have been defined in Chapter 4 under Other Shared Memory Locations. The exception will be "FLAG" which is used in most of the routines. This refers to an unshared memory location used solely in the routine in which it appears. In other words, each "FLAG" is a local variable. The common use of the name "FLAG" is for descriptive simplicity.

In some of the following routines there will be descriptive sentences (in lowercase) instead of pseudocode. These describe a specific function that must be performed which does not lend itself readily to a pseudocode explanation.

10.1 RECEIVE BUFFER ACCESSING

HOST ROUTINE

This routine is generated by a Timer interrupt from the card. The purpose of this routine is to check the Receive buffer and empty any characters found. This routine loops until all characters and their corresponding status bytes have been retrieved from the designated buffer. This routine is performed for each Receive buffer.

```
WHILE (HEAD<->TAIL) DO
  BEGIN
    Retrieve character from buffer
    Retrieve status byte from buffer
    Update head pointer
    Host character processing
```

END

CARD ROUTINE

This routine is the Interrupt Service Routine performed by the card in response to a Receive character interrupt from the UART. This routine services one character.

```
IF (TTAIL+2)=THEAD THEN
  BEGIN
    Retrieve character from UART & discard
    Set Bit 3 in status byte for next character
  END
ELSE
  BEGIN
    Retrieve character & strip any parity bits
    Do Bit Map check
    IF special character THEN
      BEGIN
        Grab semaphore
        Send Special Character interrupt to host
        Release semaphore
      END
    Put character into FIFO
    Put status byte into FIFO
    Update pointer
  END
```

10.2 TRANSMIT BUFFER ACCESSING

HOST ROUTINE

The host generates this routine under two possible circumstances. The first circumstance is whenever the host has data it wants to send to the card. The second circumstance is in response to a Transmit Buffer Empty interrupt from the card (presuming the host has characters it wants to send).

```
IF (TAIL+2)<>HEAD THEN          If buffer not full then . . .
  BEGIN
    IF TAIL=HEAD THEN send TX Buffer Not Empty interrupt to card
    Put character into buffer
    Update tail pointer
  END
```

CARD ROUTINE

This is the Interrupt Service Routine performed by the card in response to a Transmitter Empty interrupt from the UART. This is also basically the routine performed when the host sends the card a TX Buffer Not Empty interrupt with one addition: the card must "start" the UART before sending the 1st character out. After that, the UART will interrupt the Z-80 when its ready for the next character.

```
IF HEAD=TAIL THEN                                If buffer empty . . .
  BEGIN
    Turn off UART
    Grab semaphore
    Send TX Buffer Empty interrupt
    Release semaphore
  END
ELSE
  BEGIN
    Retrieve character from FIFO
    Send character to UART
    Update head pointer
  END
```

REVISION HISTORY

The following is a detailed history of the bug fixes that have occurred since the firmware was first released on 5/01/85. The only other change made to the IMS is the ROM map reflects the load map of the new version of firmware.

BUG FIX - 6/13/85

BRIEF DESCRIPTION - BUG IN SELF TEST

SYMPTOM - The card was unable to receive interrupts after Self Test had failed if the Self Test failure was because a deadman timer had timed out (i.e., SIO test external loopback).

REASON - The deadman timer was set up as a countdown to 0 then interrupt in the CTC. The end of the deadman timer interrupt was a jump to ST_ERR which never did a RETI. Therefore, interrupts were never getting enabled again.

SOLUTION - The solution was to add the instructions:

```
LD    BC,INIT
PUSH BC
RETI
```

Self Test failures which do not involve ISR's (like the semaphore test) still jump to the ST_ERR routine and an extra RETI instruction only pops the stack.

BUG FIX - 4/09/86

BRIEF DESCRIPTION - THERE WAS ONLY ONE VARIABLE FOR RX CHAR BIT MASK

SYMPTOM - When the user set the bits-per-character configuration, it was set for all the ports. In other words, if the user set 1 port at 8 bits per character and another at 7 bits per character with parity, all the ports were stripping the top character and adding parity. The data on the port that the user thought was 8 bits-per character was erroneous.

REASON - There was only 1 variable to strip leading bits on RX characters for all four ports. This was BITS_MSK. There should have been 1 for each port.

SOLUTION - I eliminated the variable BITS_MSK and created four new