

H E W L E T T
P A C K A R D

HP-DIO FOUR CHANNEL TERMINAL MULTIPLEXER
FIRMWARE INTERNAL MAINTENANCE SPECIFICATION (IMS)

HEWLETT PACKARD COMPANY
Roseville Networks Division
8000 Foothills Boulevard
Roseville, California 95678

MAY 9, 1986
Elizabeth Poteet

ones; BITS_0, BITS_1, BITS_2, BITS_3. Each of these is set with the bits to strip on incoming characters according to the data in the most recent configuration change interrupt for that port.

FILES CHANGED -

MX_VAR - Took out declaration for BITS_MSK; added BITS_0, BITS_1, BITS_2, and BITS_3.

MX4INIT - Took out initialization for BITS_MSK. Initialized BITS_0, BITS_1, BITS_2, and BITS_3 instead.

MX4RX - Added parameter to RX macro for the bits mask variable.

RXERR - Same as MX4RX; added parameter to RX macro for bits mask variable.

MX4TX - Took BITS_MSK out of the extrn area. I don't know where it was used or why it was there in the first place. Probably an oversight.

MXSBR - The routine, HSTCON, now passes the bits per character mask value back to the calling routine instead of setting the value in the BITS_MSK variable as before.

MXPT0, MXPT1, MXPT2, MXPT3 - These routines now receive the bits per character mask value in the A register after HSTCON has finished execution and returned. They then save the mask value in the appropriate variable location for that port.

MX4EQUS - The RX macro was changed to include the bit mask variable.

RELIABILITY IMPROVEMENT (1/89 - Randy Stout - Production Engineer)

BRIEF DESCRIPTION - INTERMITTENT BUG IN SELF TEST

SYMPTOM - The 98642 would occasionally (frequency ranging from 1 failure in 10 passes to 1 failure in 800 passes) fail its on-board Self Test.

REASON - Contrary to the published information, Zilog's CTC chip requires a Reset before interrupts are enabled. Issuing the Reset and the Enable Interrupt in the same control word can lead to a race condition on the chip. If there is an interrupt pending AND the Interrupt Enable occurs before the RESET, the CTC will issue a Z80 Interrupt. This combination occurred during the SIO Test portion of the Self Test, leading technicians to believe the SIO was at fault. The board would often pass Self Test with the new SIO - at least

during the first few cycles through the test.

SOLUTION - Since the fault occurred because there was an interrupt pending during the signal race on the CTC, it proved cleaner to remove the pending interrupt than to change the CTC initialization. The pending interrupt resulted from setting all four channels of CTC 1 as 19.2 Kbaud generators (at the beginning of the SIO Test). Only the first two channels were actually used as baud rate generators. The third channel was not required until later in the SIO test. There, it is used as a "dead man" timer. By changing

LD B,4 to
LD B,2 at line 0304 (hex) in MX4ST, the intermittent interrupt was eliminated.

OTHER CHANGES - In addition to this one bug the code was cleaned up in other places. Redundant code was removed and the SIOs were initialized according to Zilog's latest recommendations. The source listing is clearly commented at each change.

SECTION SCOPE

This document describes the internal structure of the firmware implemented for the HP-DIO RS-232 4 channel terminal multiplexer card (98642A). Specifically, this document describes:

1. All symbols used by the firmware except for symbols used as labels in the instruction sequence
2. The firmware data structure
3. The function of each firmware module
4. Any other information pertinent to the understanding of the firmware

The reader is referred to the following related documents:

1. HP-DIO FOUR CHANNEL TERMINAL MULTIPLEXER FIRMWARE EXTERNAL REFERENCE SPECIFICATION (ERS)
by Elizabeth Poteet
2. HP-DIO FOUR CHANNEL TERMINAL MULTIPLEXER HARDWARE EXTERNAL REFERENCE SPECIFICATION (ERS)
by Bob Bortolotto
3. Zilog Z80 CPU Technical Manual

4. Zilog Z80-SIO Technical Manual

5. Zilog Z80-CTC Technical Manual

This document assumes the reader has the full understanding of all the information given in the Hardware and Firmware ERS.

COMMONLY USED ABBREVIATIONS

The following is a list of the abbreviations used in this document.

RX - This is used for Receive, most commonly used to describe the characters which are entering the card from the frontplane.

TX - Transmit; most commonly used to describe the characters that the host is sending to the card to be sent out one of the ports.

ISR - Interrupt Service Routine

OVERVIEW OF FIRMWARE

The purpose of this chapter is to give an overview of the basic structure of the firmware on the FORDYCE card. Except for the Self Test and Initialization routines (see next section), all of the firmware on the card is completely interrupt driven. The interrupts

can come from either the host via the INT-COND register in the card, the UARTs (SIO's) or the CTC's. The first portion of this chapter gives a somewhat external explanation of all the possible interrupts the card may get. The second portion of this chapter will give an overview of the possible interrupts the card may send to the host (via the COMMAND register) with an explanation of the circumstances which cause the card to send them.

The firmware on the 98642A card can be accessed in three ways:

1. System power-up - This causes a card reset and a jump to location 0 in the ROM (the beginning of Self Test). Self Test is executed and, if successful, is followed by Init (card initialization routine). The end of Init is an idle loop that is in essence the main routine of the firmware. All the other firmware on the card is interrupt driven.

2. Soft Reset - A soft reset is triggered by the host writing a 80H to the RESET/I.D register (Z-80 address 8000H). This causes a NMI interrupt to the Z-80 causing a jump to location 66H in the ROM. This location contains a jump to the Init routine described above.
3. Z-80 Interrupt - The Z-80 may be interrupted by either the UARTs (SIO) or the Counter Timer Chips (CTC). The following illustrates the types of interrupts associated with each type of chip.
 - SIO - Receive interrupt: Incoming data to card
 - Transmit interrupt: Outgoing data from the host
 - External status interrupt: Modem line changes *(put note about ST RTS line)
- CTC - Host interrupt via the COMMAND register
- Timer interrupts (16 millsec.)

In the following sections, the CTC and SIO interrupt service routines will be identified and briefly described.

SIO - RECEIVE INTERRUPT ROUTINES

There are four ports on the card, there is a Receive Interrupt associated with each port. An incoming character causes a jump to the routine associated with the port which received the character. These routines are functionally identical.

The Receive Interrupt routine basically retrieves the character from the port the interrupt was received on, strips any parity bits, checks the Bit Map location for the character, sends the host a Special Character interrupt if the Bit Map location is set, and places the character in the Receive buffer along with a status byte.

There are also four Receive Error routines which are called instead of the four described above when there is an error (parity, overflow, or framing) associated with a Receive character. These routines are also functionally identical.

The Receive Error routines only differ from the Receive routines in that they first retrieve the type of error from the SIO, and set the appropriate bits in the status byte to indicate the type of error. After this, the rest of the Receive Error routine is the same as the

Receive routine.

SIO - TRANSMIT INTERRUPT ROUTINES

As with Receive Interrupts, there is a Transmit routine associated with each port. These interrupts occur when the SIO has already transmitted a character and is ready to transmit another. These routines are also functionally identical.

The Transmit routine first checks that the Transmit buffer is not empty. If not, a character is retrieved and sent to the SIO to be transmitted.

SIO - EXTERNAL STATUS INTERRUPT ROUTINES

These interrupt service routines are called when one of the SIO

channels has a transition on either one of the modem lines or a Break has occurred. There is an External Status routine for each port to determine which condition caused the interrupt. There is also a Break subroutine which all four External Status routines call if the reason for the interrupt was a incoming Break.

CTC - 16 MILLISECOND TIMER INTERRUPT

When the 16 millisecond timer in the CTC times out the Z-80 is interrupted and the Timer Interrupt Service routine is invoked. The purpose of this routine is to send an interrupt to the host to inform it to check the Receive buffers for characters.

CTC - HOST INTERRUPTS

Whenever the host writes a value to the COMMAND register, a Host Interrupt service routine is invoked via CTC 1. The interrupt service routine accesses the COMMAND register to determine the

type of host interrupt called. As described in the ERS, the bits in the COMMAND register represent the types of host interrupts available. They are:

- Port 0 Transmit Buffer Not Empty
- Port 1 Transmit Buffer Not Empty
- Port 2 Transmit Buffer Not Empty
- Port 3 Transmit Buffer Not Empty
- Port 0 Configuration Change
- Port 1 Configuration Change

Port 2 Configuration Change
Port 3 Configuration Change
Port 0 Send Break
Port 1 Send Break
Port 2 Send Break
Port 3 Send Break
Modem Output Change
Timer On/Off
Self Test On

EQUATE & VARIABLE SYMBOLS DICTIONARY

This chapter defines all the symbols which are not used as a label or subprogram name. All equates and variables used in the firmware are contained in two files: &MX-VAR and &MX4EQUS. The labels defined in &MX-VAR are all of the variables used in the firmware. They will be defined in two sections; those that are accessed by both the card and the host and those that are only accessed by the card. The labels defined in &MX4EQUS are equates used throughout the firmware. This file is copied to almost every other file. The labels defined in &MX4EQUS are cross referenced by the files which use each in the individual file descriptions further in this document. This chapter will merely give a description of the usage of each without specifying which firmware module uses them.

SHARED VARIABLES IN &MX-VAR

BD-0 : This contains the baud rate value for port 0.
BD-1 : This contains the baud rate value for port 1.
BD-2 : This contains the baud rate value for port 2.
BD-3 : This contains the baud rate value for port 3.

BIT-MAP : This defines the starting address for the Bit Map table

CMND-TAB : This defines the starting address of the COMMAND register port specific interrupt table.

CONFIG-0 : Contains the current configuration data code for port 0
CONFIG-1 : Contains the current configuration data code for port 1
CONFIG-2 : Contains the current configuration data code for port 2
CONFIG-3 : Contains the current configuration data code for port 3

ICR-TAB : This defines the starting address of the INT-COND register port specific interrupt table.

MODM-IN : Contains the current status of the input modem lines

MODM-MASK : Contains the information designating which modem input lines the host wants to be notified of in the event

of a change.

MODM-OUT : Contains the current status of the output modem lines

RHEAD-0 : The head pointer index for the Receive FIFO for port 0

RHEAD-1 : The head pointer index for the Receive FIFO for port 1

RHEAD-2 : The head pointer index for the Receive FIFO for port 2

RHEAD-3 : The head pointer index for the Receive FIFO for port 3

RTAIL-0 : The tail pointer index for the Transmit FIFO for port 0

RTAIL-1 : The tail pointer index for the Transmit FIFO for port 1

RTAIL-2 : The tail pointer index for the Transmit FIFO for port 2

RTAIL-3 : The tail pointer index for the Transmit FIFO for port 3

ST-COND : Contains the result of Self Test

THEAD-0 : The head pointer index for the Transmit FIFO for port 0

THEAD-1 : The head pointer index for the Transmit FIFO for port 1

THEAD-2 : The head pointer index for the Transmit FIFO for port 2

THEAD-3 : The head pointer index for the Transmit FIFO for port 3

TTAIL-0 : The tail pointer index for the Transmit FIFO for port 0

TTAIL-1 : The tail pointer index for the Transmit FIFO for port 1

TTAIL-2 : The tail pointer index for the Transmit FIFO for port 2

TTAIL-3 : The tail pointer index for the Transmit FIFO for port 3

UNSHARED VARIABLES (CARD ONLY) - &MX-VAR

BITS-MSK : Contains the mask to strip off parity bits on RX characters

RBRK-0 : This is the end-of-break-detected flag for port 0

RBRK-1 : This is the end-of-break-detected flag for port 1

RBRK-2 : This is the end-of-break-detected flag for port 2

RBRK-3 : This is the end-of-break-detected flag for port 3

STAT-0 : Contains the bit pattern for the status register - port 0

STAT-1 : Contains the bit pattern for the status register - port 1

STAT-2 : Contains the bit pattern for the status register - port 2

STAT-3 : Contains the bit pattern for the status register - port 3

TEST : This is a general purpose location used in Self Test

TMPTAB : The starting addr. of the temporary table for CMND-TAB data

TMRFLG : The flag which indicates whether the timer is off or on

TON0 : Transmitter on/off flag for port 0

TON1 : Transmitter on/off flag for port 1

TON2 : Transmitter on/off flag for port 2

TON3 : Transmitter on/off flag for port 3

WR3-0 : Contains the current value in SIO write register 3 for port 0

WR4-0 : Contains the current value in SIO write register 4 for port 0

WR5-0 : Contains the current value in SIO write register 5 for port 0

WR3-1 : Contains the current value in SIO write register 3 for port 1

WR4-1 : Contains the current value in SIO write register 4 for port 1

WR5-1 : Contains the current value in SIO write register 5 for port 1

WR3-2 : Contains the current value in SIO write register 3 for port 2

WR4-2 : Contains the current value in SIO write register 4 for port 2

WR5-2 : Contains the current value in SIO write register 5 for port 2

WR3-3 : Contains the current value in SIO write register 3 for port 3

WR4-3 : Contains the current value in SIO write register 4 for port 3

WR5-3 : Contains the current value in SIO write register 5 for port 3

EQUATES - &MX4EQUS

BEG-BD : Initial value for BD registers - 9600 baud

BEG-CONFG : Initial value for CONFG registers

BREAK : Contains bit position value for the status byte break bit

CTC-0-C0 : CTC #0 Channel 0 address (used as pt 0 baud rate generator)

CTC-0-C1 : CTC #0 Channel 1 address (used as pt 1 baud rate generator)

CTC-0-C2 : CTC #0 Channel 2 address (used for host interrupts)

CTC-0-C3 : CTC #0 Channel 3 address (unused)

CTC-1-C0 : CTC #1 Channel 0 address (used as pt 2 baud rate generator)

CTC-1-C1 : CTC #1 Channel 1 address (used as pt 3 baud rate generator)

CTC-1-C2 : CTC #1 Channel 2 address (used for timer interrupts)

CTC-1-C3 : CTC #1 Channel 3 address (unused)

COM-REG : Address of COMMAND register

CTC-V0 : Beginning CTC 0 vector in RAM for Self Test CTC tests

CTC-V1 : Beginning CTC 1 vector in RAM for Self Test CTC tests

CTCW RD : CTC Channel Control word value for 16 millsec. timer

ERR-MSK : Mask used to isolate status byte bits in RX Error ISR

ESMSK1 : Mask used to isolate modem line input bits in Ext. Stat. ISR
ESMSK2 : Mask used to isolate Bit 0 in MODM-IN register
EVAL : Test value in Self Test - NMI test
FRAME : Contains bit position value for Framing error in Status byte
IC-BIT : Bit position in MODM-MASK and MODM-IN reg. for IC bit
INT-CODE : INT-COND register value of Self Test Done interrupt
INT-COND : Address of INT-COND register

INT-REG : Address of Hardware status register - INT-REG
MOD-INT : INT-COND bit for Input Modem Line Change interrupt
OVERRUN : Contains bit position value for Overrun error in Status byte
PARITY : Contains bit position value for Parity error in Status byte
PASS : Value of ST-COND register when Self Test passes

PORT0 : Bit position for port specific int. in INT-COND reg. - port 0
PORT1 : Bit position for port specific int. in INT-COND reg. - port 1
PORT2 : Bit position for port specific int. in INT-COND reg. - port 2
PORT3 : Bit position for port specific int. in INT-COND reg. - port 3

RAM-BEG : Address of beginning of RAM
RAM-SEG : Number of 256 byte segments in RAM - used in Self Test
RAM-SIZ : Number of bytes in RAM

RESET : Address of RESET/ID register
ROM-BEG : Address of beginning of ROM
ROM-END : Address of last byte of ROM
ROM-SEG : Number of 4K segments of ROM - used in Self Test

RX-BASE0 : High byte of RX FIFO tail pointer indexes - port 0
RX-BASE1 : High byte of RX FIFO tail pointer indexes - port 1
RX-BASE2 : High byte of RX FIFO tail pointer indexes - port 2
RX-BASE3 : High byte of RX FIFO tail pointer indexes - port 3

OM-REG : Address of Semaphore register

SIO-0-AD : SIO #0 Channel A data address
SIO-0-AC : SIO #0 Channel A control address
SIO-0-BD : SIO #0 Channel B data address
SIO-0-BC : SIO #0 Channel B control address

SIO-1-AD : SIO #1 Channel A data address

SIO-1-AC : SIO #1 Channel A control address
SIO-1-BD : SIO #1 Channel B data address
SIO-1-BC : SIO #1 Channel B control address

SPEC-ICR : ICR-TAB bit position value for Special Character interrupt

ST-COND : Address of ST-COND register (this is also defined in &MX-VAR
Its a case of overkill but was done before this was written)

ST-INT : INT-COND bit position value for Self Test interrupt

SVAL : Test value used in Self Test - NMI test

TEST : Address of general purpose test location (this is also defined
in &MX-VAR as is ST-COND - another case of overkill)

TFIFO-0 : Low byte base for TX head pointer index for port 0
TFIFO-1 : Low byte base for TX head pointer index for port 1
TFIFO-2 : Low byte base for TX head pointer index for port 2
TFIFO-3 : Low byte base for TX head pointer index for port 3

TME-INT : INT-COND bit position value for Time-Out Timer interrupt

TMSK : Mask to isolate the low nibble of the TX head pointer

TMRPRE : CTC prescale value for the 16 millisecond timer

TX-BASE : Contains the high byte value for all of the TX head pointers

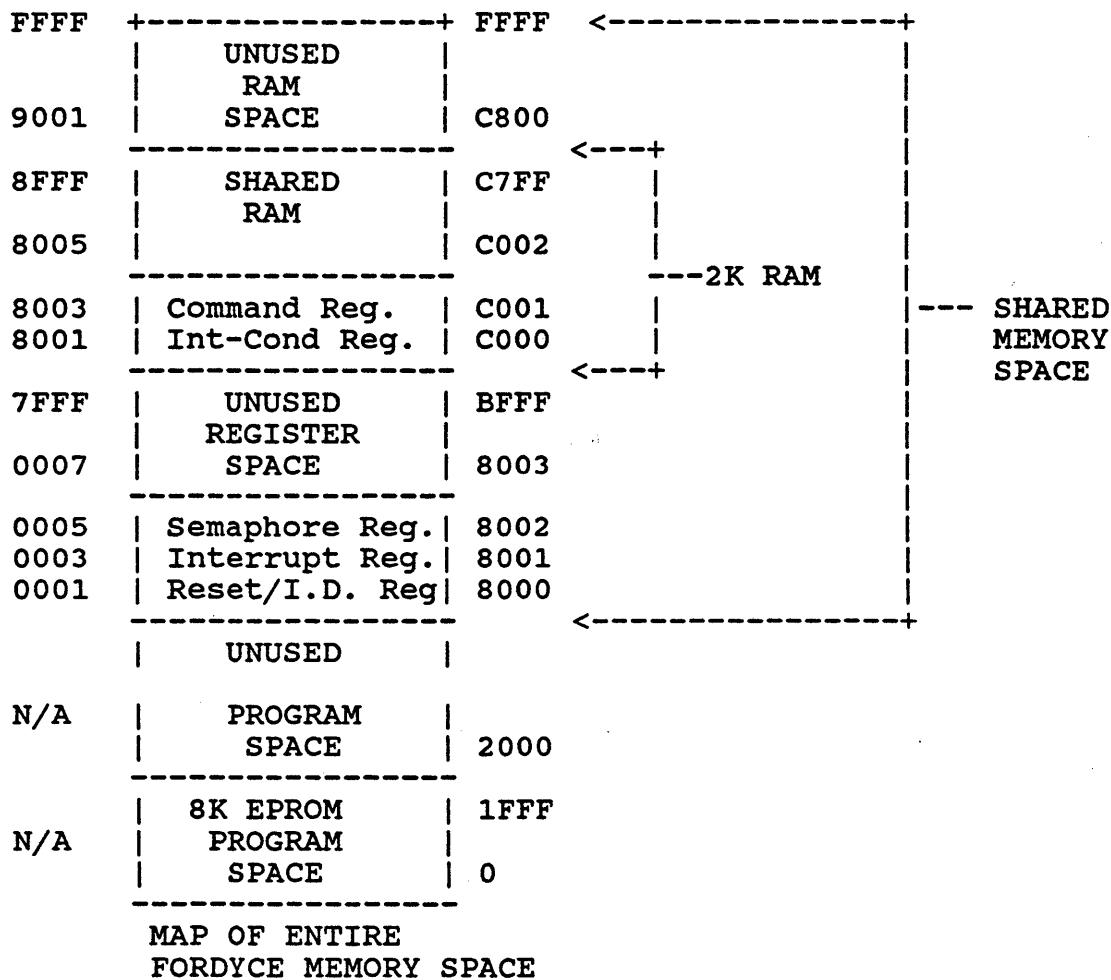
VEC : The beginning address of the interrupt vectors in ROM

MEMORY ADDRESS SPACE

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)



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	C7B0
8F5F	C7AF
STACK - 80 BYTES	
TRANSMIT 16 BYTES	

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 & CONFIG. DATA	C76F
8E01		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

The following is a illustration of ROM showing the file entry points and their placement. The value at the high byte of ROM is a CRC checksum value which is used to test ROM in the Self Test.

1FFC | CRC CHECKSUM |

1FC0	SIO & CTC ISR VECTORS
0F2D	unused
0F04	EXTMR
0EA7	MXMOD
0E5A	MXSBR
0DBA	MXPT3
0D1B	MXPT2
0C7D	MXPT1
0BE0	MXPT0
0AFD	MXEXT
0A97	MXHST
0A80	MXTMR
0964	MX4TX
0798	RXERR
0624	RX-ISR
0624	MX-VAR
04C0	INIT
0000	MX4ST

ROM MAP

HARDWARE CONSIDERATIONS and DEFAULT SETTINGS

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.

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

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

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	REQUEST TO SEND - OUTPUT
	DTR	TERMINAL READY - OUTPUT
	CTS	CLEAR TO SEND - INPUT
	DCD	RECEIVER READY - INPUT

	SYNC	DM	DATA MODE - INPUT
SIO 0 CHANNEL B (PORT 1)	RTS	SR IC	SIGNAL RATE SELECTOR - OUTPUT
	DTR		UNUSED
	CTS		HOOD DETECT - PORT 1
	DCD		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).

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

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.

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

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.

DEFAULT TIMER SETTING

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

FIRMWARE FILE LIST & OVERVIEW

The following is a list of the files which comprise the firmware and a brief description of which portion of code they contain. Included in the description is the following information: All labels which are either used or defined in the file which have impact on other files (simple in-file jump labels are not included), variables used in the file (all variables in the firmware are defined in the file %MX-VAR), and all macros called in the file (all macros used in the firmware are contained in the file %MX4EQUS).

The order in which the files are described below is the order that they were loaded in on the final loading of the FORDYCE firmware. The first file listed occupies address 0000 and the rest follow in order. Several of the files contain absolute code segments. However, this detail will be described in the individual detailed explanations of each code module. The "%" in front of each filename denotes that the file contains the object code.

The term "Public Labels" will be used to denote those labels which are defined in the file being described but used in other files. The term "External Labels" will be used to denote the opposite; those labels which are defined in other files and used in the currently described file. The term "Variables" will be used to describe those labels which are used to define a portion of RAM address space. As mentioned above, all of the variables in the firmware are defined in the file %MX-VAR and are therefore external to all of the other files.

%MX4ST - This file contains the entire Self Test.

Public Labels: MX4ST, CTC-ERR0
External Labels: INIT,
Variables: ST-COND,TEST

%MX4INIT - The cards' initialization code is contained in this file.

At the end of the initialization, the file also contains the "do nothing" loop that occupies the card while waiting for interrupts. %MX4EQUS is copied to this file.

Public Labels: INIT, BD-TAB
External Labels: TX-0, TX-1, TX-2, TX-3, REC-0, REC-1, REC-2, REC-3, RX-ERR0, RX-ERR1, RX-ERR2, RX-ERR3, TX-0, TX-1, TX-2, TX-3, TMR-ISR, HSTINT, CTC-ERR0, CONFIG-0, CONFIG-1, CONFIG-2, CONFIG-3, BD-0, BD-1, BD-2, BD-3, MDM1-SUB,

MDM3-SUB

Variables: BITS-MSK, THEAD-0, THEAD-1, THEAD-2, THEAD-3, TTAIL-0, TTAIL-1, TTAIL-2, TTAIL-3, WR3-0, WR3-1, WR3-2, WR3-3, WR4-0, WR4-1, WR4-2, WR4-3, WR5-0, WR5-1, WR5-2, WR5-3, TMRFLG

Macros: none

%MX-VAR - This file contains all of the variable labels which are used used in the firmware. The file is divided into two segments, a data segment and an absolute segment. The variables defined in the data segment (DSEG) are those used only by the firmware and the variables defined in the absolute segment (ASEG) are used by both the card and the host.

Public Labels: Every label defined in the file is public. All of the labels listed as "Variables" in the other file descriptions are Public Labels in this file

External Labels: None

Variables: Not applicable

Macros: Not applicable

%MX4RX - This file contains the Receive interrupt service routines for all four ports. These routines are expanded macros which are defined in the file &MX4EQUS. &MX4EQUS is copied to this file.

Public Labels: REC-0, REC-1, REC-2, REC-3

External Labels: None

Variables: STAT-0, STAT-1, STAT-2, STAT-3, RHEAD-0, RHEAD-1, RHEAD-2, RHEAD-3, RTAIL-0, RTAIL-1, RTAIL-2, RTAIL-3, BIT-MAP, ICR-TAB, BITS-MSK

Macros: RECISR

%RXERR - This file contains the Receive Error interrupt service routines for all four ports. These routines are expanded macros which are defined in the file &MX4EQUS. &MX4EQUS is copied to this file.

Public Labels: RX-ERR0, RX-ERR1, RX-ERR2, RX-ERR3

External Labels: None

Variables: STAT-0, STAT-1, STAT-2, STAT-3, RHEAD-0, RHEAD-1, RHEAD-2, RHEAD-3, RTAIL-0, RTAIL-1, RTAIL-2, RTAIL-3, BIT-MAP, ICR-TAB, BITS-MSK

Macros: SPEC-RX
RECISR

%MX4TX - This file contains the Transmit interrupt service routines for all four ports. These routines are expanded macros which are defined

in the file &MX4EQUS. &MX4EQUS is copied to this file.

Public Labels: TX-0,TX-1,TX-2,TX-3

External Labels: None

Variables: THEAD-0,THEAD-1,THEAD-2,THEAD-3,TTAIL-0,TTAIL-1,TTAIL-2,
TTAIL-3,TON0,TON1,TON2,TON3,ICR-TAB,BITS-MSK

Macros: TX-ISR

%MXTMR - This file contains the CTC interrupt service routine which
sends a Timer interrupt to the host. &MX4EQUS is copied to this
file.

Public Labels: TMR-ISR

External Labels: None

Variables: None

Macros: None

%MXHST - This file contains the beginning of the interrupt service
routine which is invoked by CTC 0,CH 2 when the host puts a value
in the COMMAND register. This file contains the portion of the
host ISR which decodes the COMMAND register to decipher the reason

for the interrupt. &MX4EQUS is copied to this file.

Public Labels: HSTINT,EEE2,EEE3,EEE4,EEE5,EEE6,EEE7

External Labels: ISRPT0,ISRPT1,ISRPT2,ISRPT3,MODOUT,TMROFF,MX4ST

Variables: TMPTAB,CMND-TAB

Macros: None

%MXEXT - This file contains the SIO External Status interrupt service
routines for all four ports. An external status interrupt occurs
when either a Break has been received or a change has occurred on
one of the modem lines. &MX4EQUS is copied to this file.

Public Labels: EX-0,EX-1,EX-2,EX-3,MDM3-SUB,MDM1-SUB

External Labels: None

Variables: RBRK-0,RBRK-1,RBRK-2,RBRK-3,STAT-0,STAT-1,STAT-2,STAT-3,
MODM-IN,MODM-MASK

Macros: None

%MXPT0 - This file contains part of the Interrupt Service routine for
a host interrupt. In particular it contains the routine for a port
specific interrupt for port 0. &MX4EQUS is copied to this file.

Public Labels: ISRPT0

External Labels: BD-TAB,EEE2,SNDBRK,HSTCON

Variables: TMPTAB,CONFIG-0,WR3-0,WR4-0,WR5-0,BD-0,TTAIL-0,THEAD-0,
TON0

Macros: HOSTTX

O %MXPT1 - This file contains part of the Interrupt Service routine for a host interrupt. In particular it contains the routine for a port specific interrupt for port 1. &MX4EQUS is copied to this file.

Public Labels: ISRPT1
External Labels: BD-TAB,EEE3,SNDBRK,HSTCON
Variables: TMPTAB,CONFIG-1,WR3-1,WR4-1,WR5-1,BD-1,TTAIL-1,THEAD-1,
TON1
Macros: HOSTTX

%MXPT2 - This file contains part of the Interrupt Service routine for a host interrupt. In particular it contains the routine for a port specific interrupt for port 2. &MX4EQUS is copied to this file.

Public Labels: ISRPT2
External Labels: BD-TAB,EEE4,SNDBRK,HSTCON
Variables: TMPTAB,CONFIG-2,WR3-2,WR4-2,WR5-2,BD-2,TTAIL-2,THEAD-2,
TON2
Macros: HOSTTX

%MXPT3 - This file contains part of the Interrupt Service routine for a host interrupt. In particular it contains the routine for a port specific interrupt for port 3. &MX4EQUS is copied to this file.

Public Labels: ISRPT3
External Labels: BD-TAB,EEE5,SNDBRK,HSTCON
Variables: TMPTAB,CONFIG-3,WR3-3,WR4-3,WR5-3,BD-3,TTAIL-3,THEAD-3,
TON3
Macros: HOSTTX

%MXSBR - This file contains two subroutines which are part of the host interrupt service routine. These subroutines are called by ISRPT0, ISRPT1, ISRPT2, and ISRPT3. The first subroutine is part of a Configuration Change interrupt from the host. The second subroutine is part of the Send Break interrupt from the host.

&MX4EQUS is copied to this file.

Public Labels: HSTCON,SNDBRK
External Labels: None
Variables: BITS-MSK
Macros: None

%MXMOD - This file contains the part of the host interrupt service routine which is responsible for handling a Modem Output Line Change interrupt. &MX4EQUS is copied to this file.

Public Labels: MODOUT

External Labels: EEE6
Variables: WR5-0,WR5-1,MODM-OUT

%EXTMR - This file contains the part of the host interrupt service routine which is responsible for handling a Timer On/Off interrupt. &MX4EQUS is copied to this file.

Public Labels: TMROFF
External Labels: EEE7

Variables: TMRFLG
Macros: None

&MX4EQUS - This file is not part of the object code. It is a sort of service file which contains all of the equates used in the firmware and defines all of the macros. This file is copied to almost every other file in the firmware with the exception of %MX4ST and %MX-VAR.

The names of the macros contained in this file are:

RECISR - Used in %MX4RX
SPEC-RX - Used in %RXERR
TX-ISR - Used in %MX4TX
HOSTTX - Used in %MXPT0,%MXPT1,%MXPT2,%MXPT3

DETAILED DESCRIPTION OF FIRMWARE MODULES

This chapter is devoted to a detailed description of each of the firmware modules. A firmware module is rather loosely defined as a piece of code with one entry point and one exit point which

performs one basic function. The first part of this chapter identifies each of these modules by entry point name and shows the source of interrupt which causes the execution of the routine. (Remember, all of the firmware on the card is interrupt driven). The second section is an explanation of the basic algorithms used in the firmware modules. This is in a separate section because most of the algorithms are used in more than one module. The sections that follow the algorithm explanation describe each of the firmware modules listed in the first section.

OUTLINE OF FIRMWARE MODULE RELATIONSHIPS

The following is an outline of the relationships of the firmware modules to each other and the source of the interrupt that starts off a particular chain of events. The labels that are indented are those routines that are called by the preceding label. For example, HSTINT calls ISRPT0 which in turn calls SNDBRK and HSTCON. In the sections that follow, each of these routines will be explained in

detail.

Interrupt Source	Label	Description
power-up*	MX4ST	Self Test
soft reset	INIT	Card initialization routine
SIO	REC-0	Receive interrupt routine - pt 0
SIO	REC-1	" - pt 1
SIO	REC-2	" - pt 2
SIO	REC-3	" - pt 3
SIO	RX-ERRO	Receive Error int. routine - pt 0
SIO	RX-ERR1	" - pt 1
SIO	RX-ERR2	" - pt 2
SIO	RX-ERR3	" - pt 3
SIO	TX-0	Transmit interrupt routine - pt 0
SIO	TX-1	" - pt 1
SIO	TX-2	" - pt 2
SIO	TX-3	" - pt 3
CTC	TMR-ISR	16 mill. timer interrupt routine
CTC	HSTINT	Host interrupt - from COMMAND reg.
	ISRP TO	Port specific interrupt - pt 0
	SNDBRK	Send Break int. routine
	HSTCON	Config. Change interrupt routine
	ISRP T1	Port specific interrupt - pt 1
	SNDBRK	See Above
	HSTCON	"
	ISRP T2	Port specific interrupt - pt 2
	SNDBRK	See Above
	HSTCON	"
	ISRP T3	Port specific interrupt - pt 3
	SNDBRK	See above
	HSTCON	"
	MODOUT	Modem Output Line Change int. routine
	TMROFF	Timer On/Off interrupt routine
SIO	EX-0	External Status Change int. - pt 0
	MDM3-SUB	Modem Input Line chg for port 0
	BRK-SUB	Break Detection routine
SIO	EX-1	External Status Change int. - pt 1
	MDM1-SUB	Modem Input Line chg for port 1
	BRK-SUB	Break Detection routine
SIO	EX-2	External Status Change int. - pt 2
	BRK-SUB	Break Detection routine
SIO	EX-3	External Status Change int. - pt 3

EXPLANATION OF COMMON ALGORITHMS

1. RECEIVE BUFFER, EMPTY/FULL DECISION - The Receive buffers are handled as circular FIFO data structures with an associated head and tail index for each. The algorithm used here never lets the buffer get completely full, so when then head and tail indexes are equal it means that the buffer is empty, not full. The method for making sure the buffer never gets completely full is to add 2 to the tail index and check for equality with the head index before a Receive character is placed into the buffer. If they are equal, the buffer is assumed full and the character received is discarded. In essence this means that there is really only room for 127 characters per Receive buffer instead of 128.
2. HEAD & TAIL POINTER HANDLING - The head and tail pointers for the Receive buffers consist of Head and Tail index pointers and a base pointer address. The base pointer is the upper byte of the Receive buffer address and the head and tail index pointers are the lower byte. The effective address then for any address in a Receive buffer is the concatenation of the base and the head or tail index. As the Receive buffers are only 256 bytes, buffer wraparound is automatically taken care of as the head and tail indexes are 8 bit quantities.
3. BIT MAP CHECK - As explained in the Firmware ERS, the Bit Map is a 256 byte table with each byte representing a character. In other words, the character whose value is 56 is associated with the byte in the Bit Map whose relative placement in the table is 56 from the beginning of the table. The first four bits in each Bit Map location represent the four ports. When a character is received, it is concatenated with the Bit Map Base value to form the effective address of the Bit Map location associated with the character. Once the byte is retrieved, the bit representing the port the character was received at is checked. If the bit is on, the character is a "special" character and a Special Character interrupt is sent to the host. If the bit is off, no interrupt is sent.
4. STRIPPING PARITY BITS - After each receive character is retrieved from the SIO, a logical AND is performed with it and the contents of a location called BITS-MSK. This location contains a mask designed to strip off any possible parity bits that might be attached to the character. The value of BITS-MSK is based upon the number of bits per character the card is configured to. If the

card is configured to 7 bits per character, BITS-MSK will contain a "1" in the first 7 bit locations and a "0" in the 8th bit. If the card is set to 6 bits per character, BITS-MSK is 00111111 or 3F hex. The same idea holds for other bits per character settings. BITS-MSK is updated every time the card is reconfigured.

5. SENDING AN INTERRUPT TO THE HOST - USE OF SEMAPHORE REGISTER - As described in the ERS, when the card wants to send an interrupt to the host it writes a value to the INT-COND register. However, before writing to either the INT-COND register or the ICR-TAB, The card will first "grab" the Semaphore register. In other words, the card will check the Semaphore register to see if it is free. If not, the card will sit and cycle, continually checking the Semaphore register until the host releases it. The basic protocol is the same for the card and the host. Both grab the Semaphore before accessing either the COMMAND register, the INT-COND register, the CMND-TAB registers, or the ICR-TAB registers.
6. STATUS BYTE - There is a location reserved for the status byte which is initially set to zero in the initialization routine. This byte is retrieved and written to the appropriate Receive buffer as each character is placed in the buffer. If there is no room in the buffer and the receive character is discarded, the buffer overflow bit is set in this byte. The next character that is placed in the Receive buffer will also have the status byte with the overflow bit set, notifying the host that there are missing characters between the last one picked up and the current character. The Receive error routine also can alter the status byte to display error conditions associated with an incoming character. However, once a character is placed in the buffer with the status byte, the status byte register is cleared for the next character.
7. TRANSMIT BUFFER, EMPTY/FULL DECISION - As with the Receive buffers, the Transmit buffers are also handled as circular FIFO buffers. Also, the Transmit algorithm which resolves empty or full buffer arbitration is the same for Transmit Buffers as it is for Receive buffers. In the case of the Transmit buffers, the host never lets the buffer get completely full, so when the Head and Tail pointer indexes are equal, the buffer is empty.
8. HEAD & TAIL POINTER HANDLING - As with the Receive buffer pointers, the head and tail pointers are actually a concatenation of head and tail pointer indexes and a Transmit buffer base address (which represents the upper byte of the actual Transmit buffer address). However, unlike the Receive buffer

pointers, the Transmit head index actually consists of two values, the base lower byte and the head pointer index. Buffer wraparound is handled by incrementing the head pointer index, and masking off the top nibble. When the actual pointer address is needed, the head index is added to the base lower byte and the result is concatenated with the base upper byte.

9. REASON FOR CYCLING IN HSTINT ROUTINE - As with all of the interrupt service routines, the HSTINT is non-interruptable. In other words, interrupts are disabled at the start of the routine and reenabled at the end of the routine. Consequently, during the course of this routine, if the host sends another interrupt it will be lost because the CTC can't buffer interrupts. Therefore, this routine will keep checking for and servicing interrupts until the COMMAND register is empty.
10. DECIPHERING THE TYPE OF INTERRUPT - The E register is used to hold the contents of the COMMAND register as it is being deciphered. Each bit position in the COMMAND register represents a particular interrupt (with the exception of bit 7). Therefore, the interrupts are deciphered by putting the value in the COMMAND register into the E register and rotating each bit to the right one by one testing the carry bit each time. If a bit is on, this routine jumps to the subroutine responsible for handling that particular interrupt. It is possible for there to be more than one interrupt set in the COMMAND register. When program control returns from a subroutine, this routine resumes checking the rest of the bits.
11. CHANGING THE SIO WRITE REGISTERS TO NEW CONFIGURATION - The subroutine, HSTCON, is responsible for changing the bit pattern in the CONFG register to match the format in the SIO Write registers (This is explained in more detail in the section on HSTCON.) Upon return from HSTCON the B register contains the changed bit pattern. Write Register #4 is updated by clearing out the old lower byte and ANDing it with the lower byte of the B register value which contains the bits representing the new parity and stop bits information. The new value in Write register 4 is then written to the SIO. SIO Write register 5 is updated next. Bits 5 and 6 in the B register value represent the new TX bits-per-character information. Bits 5 and 6 in the old Write register 5 are cleared and the replaced with those in the B register. This is then written to the SIO. Finally, bits 6 & 7 in the B register are substituted for bits 6 and 7 in Write register 3. These bits represent the RX bits-per-character information. The new copy of Write register is then written to the SIO.
12. CHANGING THE BAUD RATE - The BD register contains a number which

represents an index into the BD-TAB, the table which contains the CTC Channel Control Words and the CTC Time Constant values which determine a specific baud rate. The value in the BD register is multiplied by 2 (since each baud rate has the two associated CTC values) and added to the base BD-TAB address to form the effective address. The correct Channel Control Word and the Time Constant value are then sent to the CTC.

REC-0,REC-1,REC-2,REC-3 - RECEIVE INTERRUPT ROUTINES

EXTERNAL DESCRIPTION:

The four routines, REC-0, REC-1, REC-2, and REC-3 will be described together as they are virtually the same routine. The code for all four is defined in the macro, RECISR, which resides in the file &MX4EQUS. (Remember that this file is copied to the file &MX4RX which contains these entry points.)

The Receive routines are called when the SIO has received a character at one of the ports. The Z-80 accesses the correct vector location for the interrupt and causes a jump to the correct Receive routine.

The Basic purpose of the Receive routine is to retrieve the character from the UART (SIO) and place it in the correct Receive buffer in RAM along with an accompanying status byte. However, before placing the character in RAM, the Bit Map location for the character is checked to see if it is a special character, i.e., the host wants to know of its presence immediately. If the correct bit for the character and the port is set, a Special Character interrupt is sent to the host. It is the responsibility of the host to determine which character is special because the Special Character interrupt only notifies the host that such a character has been received. It doesn't specify which character it is and where in the buffer it has been placed.

INTERNAL DESCRIPTION:

```
Retrieve head pointer index for Receive buffer
Retrieve tail pointer index for Receive buffer
Tail pointer = Tail pointer + 2
If Head = Tail then           ;no more room in buffer
    Retrieve character and discard
    Set 'buffer overflow' bit in Status byte
    Go to exit

else                         ;available buffer space
    Retrieve character from SIO
    Mask off any parity bits
    Check correct Bit Map location
```

```
If Bit Map position for port set then ;special character
  Grab semaphore
  Set bit in ICR-TAB for port-specific interrupt
  Set bit in INT-COND register
  Clear semaphore
  Effective tail pointer = base + tail pointer index
  Put character into buffer
  Increment buffer address
  Put status byte into buffer
  Clear status byte register
  Tail pointer index = tail pointer index + 1
```

UPON ENTRY: No relevant values in any registers.

UPON EXIT: No relevant values in any registers.

CALLED ROUTINES: none

RXERR0, RXERR1, RXERR2, RXERR3 - RECEIVE ERROR INTERRUPTS

EXTERNAL DESCRIPTION:

As with the Receive routines, these four routines will be described together as they too are virtually identical except for the port references. In addition, except for the addition of code to decipher the type of error, these routines are the same as the Receive routines. As a matter of fact, the same macro is called. Therefore, only the first portion of these routines will be described. At the end of that each Receive error routine is identical to the regular Receive routine for that port.

The Receive Error routines are called when the SIO detects either a parity, framing, or SIO overflow error on the received character. The error type is denoted in the status byte and the Receive error then proceeds as the regular receive routine.

INTERNAL DESCRIPTION:

```
Retrieve contents of SIO Read Register 1
Shift 1 bit to left ;so aligns with status byte
Mask off all but bits 7,6, & 5
Retrieve status byte register
"OR" status byte with masked RR 1 value
Write new value to status byte
Reset SIO error latches
```

*The macro RECISR is now called - the routine proceeds exactly as a Receive routine.

UPON ENTRY: no relevant register values

UPON EXIT: no relevant register values

CALLED BY: SIO - RECEIVE ERROR INTERRUPT

CALLED ROUTINES: none

TX-0,TX-1,TX-2,TX-3 - TRANSMIT INTERRUPT ROUTINES

EXTERNAL DESCRIPTION:

As with the Receive and the Receive Error routines, these four are also functionally identical, i.e. all four call the same macro. A Transmit interrupt is generated by the SIO as the SIO transmit buffer goes empty. In other words, the SIO interrupts the Z-80 when it is ready for another character to transmit.

The Transmit interrupt routine is responsible for retrieving a character from the appropriate Transmit buffer and sending it to the SIO. The Head and Tail index pointers for the Transmit buffer are first checked to see if the buffer is empty and the card sends the host a TX Buffer Empty interrupt. If it is, a value is sent to the SIO to turn off TX interrupts. If there are characters in the buffer, the next character is retrieved and sent to the SIO and the Head index is updated.

INTERNAL DESCRIPTION:

```
Retrieve Head pointer index for Transmit buffer
Retrieve Tail pointer index for Transmit buffer
If Head = Tail then                                ;buffer is empty
    Turn off Transmitter interrupts from SIO
    Clear Transmitter on/off flag
    Grab Semaphore register
    Send host a TX Buffer Empty interrupt
    Release Semaphore register
else
    Effective Head pointer address = Head index + Base
    Retrieve character from TX buffer
    Send character to SIO
    Increment Head index
```

UPON ENTRY: no relevant register values

UPON EXIT: no relevant register values

CALLED BY: SIO - TRANSMIT BUFFER EMPTY INTERRUPT

CALLED ROUTINES: none

TMR-ISR - 16 MILLSEC. TIMER INTERRUPT

EXTERNAL DESCRIPTION:

This routine is called every time the CTC timer associated with the 16 millisecond time-out downcounts to zero. The basic purpose of the routine is to send a Timer interrupt to the host.

INTERNAL DESCRIPTION:

Grab Semaphore register
Send Timer interrupt to host
Release Semaphore

;set bit in INT-COND register

UPON ENTRY: no relevant register values

UPON EXIT: no relevant register values

CALLED BY: CTC - TIME OUT INTERRUPT

CALLED ROUTINES: none

HSTINT - HOST INTERRUPT SERVICE ROUTINE

EXTERNAL DESCRIPTION:

This routine is called when the host writes a value to the COMMAND register, i.e. sends an interrupt to the card. This routine empties the contents of the CMND-TAB and COMMAND registers and begins checking the bits in both to determine what type of host interrupt was requested. When the interrupt has been interpreted the correct service routine is called. Once the interrupt has been completely serviced, control will return to this routine and a jump will be made to the beginning of the routine again to see if the host has sent another interrupt during the course of servicing the current one. This cycle will continue until the COMMAND register is empty.

INTERNAL DESCRIPTION:

Grab Semaphore register
Retrieve value in COMMAND register
If COMMAND register = 0 then goto exit
else
 Retrieve value in CMND-TAB
 Clear COMMAND and CMND-TAB registers
 Release Semaphore register
 Check each bit in COMMAND reg. and jump to appropriate routine if set
 Go to beginning of routine

UPON ENTRY: no relevant register (Z-80) values

UPON EXIT: E register contains the remaining bits to be checked from the original value in the COMMAND register.

CALLED BY: CTC - HOST INTERRUPT

CALLED ROUTINES: ISRPT0, ISRPT1, ISRPT2, ISRPT3, MODOUT, TMROFF, MX4ST

EX-0, EX-1, EX-2, EX-3 - EXTERNAL STATUS SERVICE ROUTINES

EXTERNAL DESCRIPTION:

As with the Receive, Receive Error, and Transmit routines, these routines will be described together in this section. These interrupt service routines are called when one of the SIO channels has a transition on either the Break, DCD, CTS, or SYNC inputs. A TX underrun will also cause this interrupt although these routines will not take any action if that is what has triggered the ISR. Each of the ports expects different combinations of transitions. The following are the valid transitions for each port and an explanation of what these lines represent.

PORt 0 -	BREAK	Beginning or end of Break occurrence
	DCD	Receiver Ready modem line change
	CTS	Clear to Send modem line change
	SYNC	Data Mode modem line change
PORt 1 -	BREAK	Beginning or end of Break occurrence
	DCD	Incoming Call modem line change
PORt 2 -	BREAK	Beginning or end of Break occurrence
PORt 3 -	BREAK	Beginning or end of Break occurrence

Break (BRK-SUB) is a subroutine which is called by all four routines. It will be described in detail in its own section.

INTERNAL DESCRIPTION:

EX-0 - Load parameters for Break subroutine
Call BRK-SUB
Call MDM3-SUB

EX-1 - Load parameters for Break subroutine
Call BRK-SUB
Call MDM1-SUB

EX-2 - Load parameters for Break subroutine
Call BRK-SUB

EX-3 - Load parameters for Break subroutine
Call BRK-SUB

UPON ENTRY: no relevant registers

UPON EXIT: Before calling BRK-SUB -
C reg = SIO control address for port
HL reg = Address of Break on/off flag for port
DE reg = Address of Status byte for port

CALLED BY: SIO EXTERNAL STATUS INTERRUPT

CALLED ROUTINES: BRK-SUB, MDM3-SUB, MDM1-SUB

BRK-SUB - SUBROUTINE WHICH DETECTS INCOMING BREAKS

EXTERNAL DESCRIPTION:

BRK-SUB is a subroutine which is called by EX-0, EX-1, EX-2, and EX-3, the External Status interrupt service routines for ports 0 through 3. The purpose of this subroutine is to detect both the beginning of an

incoming Break and the end of an incoming Break in the SIO. (See the Zilog Z80-SIO Technical Manual for details on how a Break is detected by the SIO).

INTERNAL DESCRIPTION:

If Start-of-Break then	(BRK flag=0 and Break bit in SIO=1)
Break Flag:=1	
Turn off RX interrupt	(To prevent interrupt for null char)
else	
If End-of-Break then	(BRK flag=1 and Break bit in SIO=0)
Break Flag=0	

Error reset the port (In case SIO is programmed for odd parity - null causes parity error)
Set Break bit in status word (Will get RX interrupt for the
Reinable RX interrupt null char. when reinable)

UPON ENTRY: C reg - SIO control address for port
HL reg - Address of Break on/off flag for port
DE reg - Address of Status byte for port

UPON EXIT: B reg - Contains contents of SIO Read register #0

CALLED BY: EX-0, EX-1, EX-2, EX-3

CALLED ROUTINES: none

MDM3-SUB - MODEM LINE CHANGE SUBROUTINE FOR RR, CS, DM LINES

EXTERNAL DESCRIPTION:

The purpose of this subroutine is to check the status of the three modem input lines in SIO 0 channel A and see whether or not there has been a change in the lines. If there has been a change, this routine then reflects that change in the MODM-IN register and checks the MODM-MASK register to see if the host wants to be interrupted. If the bit in MODM-MASK representing the changed line is on, the card will then send an INPUT MODEM LINE CHANGE INTERRUPT to the host. The three lines that this routine deals with are the DCD, CTS, and SYNC lines in SIO 0 Channel A which are used as modem lines RR, CS, and DM.

INTERNAL DESCRIPTION:

```
Rotate B reg (SIO READ REG.0) right 2 bits    (align with MODM-IN)
isolate DCD, SYNC, and CTS bits
If bits = RR, CS, and DM bits in MODM-IN then Exit
else
  MODM-IN (bits 1-3):=READ REG. 0 bits
  If MODM-IN.AND.MODM-MSK > 0 then
    grab semaphore
    send Input Modem Line Change Interrupt to host
    release semaphore
```

UPON ENTRY: B Reg - value of Read Reg 0 in SIO CH. A

UPON EXIT: no relevant register values

CALLED BY: EX-0, INIT

CALLS ROUTINES: none

MDM1-SUB - MODEM LINE CHANGE SUBROUTINE FOR IC LINE

EXTERNAL DESCRIPTION:

The purpose of this subroutine is to determine whether or not the DCD line in SIO 0 channel B changed. This modem line represents the IC line. The SIO Read Register 0 is read to determine the current status of the DCD line. This bit is then compared with bit 0 of the MODM-IN register to see if there has been a change. If so, bit 0 in MODM-IN is set to reflect the change and the MODM-MASK register is checked to see if the host wants an interrupt for the change. As with MDM1-SUB, if the bit in MODM-MASK representing the IC line (bit 0) is on, the card will then send an INPUT MODEM LINE CHANGE INTERRUPT to the host.

INTERNAL DESCRIPTION:

Compare DCD bit with IC bit in MODM-IN register

If not equal then

 Change IC bit in MODM-IN register to match DCD bit

 Retrieve MODM-MASK

 If MODM-MASK (IC bit) set then send host an Input Modem
 Line Change interrupt

UPON ENTRY: B Reg - value of Read Reg 0 in SIO CH. A

UPON EXIT: no relevant register values

CALLED BY: EX-1, INIT

CALLED ROUTINES: none

ISRPT0, ISRPT1, ISRPT2, ISRPT3 - PORT SPECIFIC INTERRUPT ROUTINES

EXTERNAL DESCRIPTION:

These four routines will be documented together as they are virtually identical except for variable names. These four routines (one for each of the four ports) identify which port specific interrupt the host is sending from the bits in ISR-TAB. The interrupt can be either a Configuration Change interrupt, a TX Buffer Not Empty interrupt, or a Send Break interrupt (or any combination of the three). These routines are called by HSTINT.

The purpose of the Configuration Change interrupt is to reconfigure the line characteristics of the SIO and change the baud rate as desired by

the host. The CONFG register contains the parity type, the number of stop bits, and the number of bits per character. This register is set by the host and accessed in this routine by the card. The BD register is the index to the BD table which contain the CTC Channel Control Word and prescale value for the baud rate requested.

The TX Buffer Not Empty interrupt is a message from the host to restart the transmitter because there are now characters in the Transmit buffer to send out. This routine merely retrieves a character from the buffer and sends it to the SIO.

The Send Break interrupt is fully contained in a subroutine called SNDBRK which will be described in its own section later in this document.

INTERNAL DESCRIPTION:

Retrieve bit 0 from TMP-TAB (bit determined Config. interrupt)
If bit 0 = 1 then

Call HSTCON (routine does 1st part of Config.)
Load SIO Write Reg. 4 with new value
Load SIO Write Reg. 5 with new value
Load SIO Write Reg. 3 with new value
Get contents of BD register
Multiply by 2
Add to BD Table base
Retrieve CTC Channel Control Word from BD-TAB
End to CTC
Inc pointer
Retrieve CTC Time Constant value
Send to CTC
Retrieve remaining bits from CMND-TAB

If bit 1 = 1 then (bit for TX Buffer Not Empty ISR)

If Transmitter flag off then
Retrieve Head Pointer Index for TX Buffer
Retrieve Tail Pointer Index for TX Buffer
If Head <> Tail then
Obtain effective TX Buffer address
Retrieve character
Send character to UART (SIO)

Increment index
Turn on Transmitter flag
If bit 2 = 1 then (bit for Send Break interrupt)
Call SEND BREAK routine
Return to calling routine (calling routine is HSTINT)

UPON ENTRY: REGISTER D - contains the TMPTAB bits which were retrieved from CMND-TAB

REGISTER E - DO NOT USE! The HSTCON routine uses this register to hold the contents of the COMMAND register. Remember, there can be more than one interrupt at a time sent.

UPON EXIT: REGISTER E - Unchanged

CALLED BY: HSTINT

CALLS ROUTINES: HSTCON, SNDBRK

HSTCON - SUBROUTINES FOR PORT SPECIFIC INTERRUPT ROUTINES

EXTERNAL DESCRIPTION:

This subroutine is the first part of the processing of a port specific Configuration Data Change Interrupt from the host. This routine basically changes the order of the bits read from the CONFG register to the corresponding bit patterns needed to program the SIO write registers. The basic algorithm of this routine is to start with the value of CONFG and change first the parity bits, then the stop bits, then the bits per character, to match the corresponding patterns needed to program the SIO write registers correctly. At the end of this routine, the A register will contain the three pieces of information in from the CONFG register with the bits changed so they match the bit patterns needed by the SIO write registers to make the actual configuration changes. However, this routine does not include actually programming the SIO write registers. That is done in the calling routine (as explained in the section on ISRPT0, ISRPT1, ISRPT2, ISRPT3 - PORT SPECIFIC INTERRUPT ROUTINES). This routine does include programming the mask value in the BITS-MSK register which will be used to strip parity bits off of Receive characters. This mask is based on the number of Receive bits per character requested by the change. (This algorithm is described in the section entitled Explanation of Common Algorithms)

INTERNAL DESCRIPTION:

If bit 1 in CONFG=0 then bit 0=1 (even parity; set parity enable for WR4)	
Rotate 2 bits right	(stop bits pattern same if add 1)
Increment A register	(contains original value of CONFG)
Rotate back 2 bits left	
Swap bits 4 & 5	(now matches bits per char in WR3 & 5)
If 8 bits per character	
BITS-MSK=FF	
If 7 bits per character	
BITS-MSK=7F	

If 6 bits per character

BITS-MSK=3F

Or 5 bits per character

BITS-MSK=1F

UPON ENTRY: A Reg - contains the CONFG register value
D reg - used by calling routine - DO NOT USE

E reg - used by calling routine - DO NOT USE

UPON EXIT: A Reg - contains the altered value of CONFG reg.

D reg - unaltered

E reg - unaltered

CALLED BY: ISRPT0, ISRPT1, ISRPT2, ISRPT3

CALLS ROUTINES: none

SNDBRK - SEND BREAK SUBROUTINE

EXTERNAL DESCRIPTION:

This routine is used when the host sends the card a Send Break interrupt. A break interrupt can be notifying the card to either begin or end a break. The card determines which by checking the Break bit in WR5. If the Break bit (bit 4)=0 then this is a start of break. If bit 4=1 then this is a signal to end a break.

INTERNAL DESCRIPTION:

If Break bit = 0 then

Set WR5 bit 4 in WR5 variable
Send new WR5 value to SIO

Else

Reset WR5 bit 4 in WR5 variable
Send new WR5 value to SIO

UPON ENTRY: D reg - Used in the calling routine - DO NOT ALTER
E reg - Used in the calling routine - DO NOT ALTER

UPON EXIT: D reg unaltered
E reg unaltered

CALLED BY: ISRPT0, ISRPT1, ISRPT2, ISRPT3

CALLS ROUTINES: none

MODOUT - MODEM OUTPUT LINE CHANGE ROUTINE

EXTERNAL DESCRIPTION:

This routine is basically a subroutine called by the Host interrupt routine when a Modem Output Change is sent by the host. The purpose of this routine is to set the modem output lines to match the bit pattern in the MODM-OUT register. Bit 0 represents the RS line. Bit 1 represents the TR line and bit 2 represents the SR line. As there is no record of which line is different, this routine sets all the lines as indicated by the MODM-OUT register.

INTERNAL DESCRIPTION:

```
If bit 0=1 then set RTS bit in WR5-0
Else reset RTS bit in WR5-0          (sets or resets RS line)
Send new WR5 value to SIO
If bit 1=1 then set DTR bit in WR5-0
Else reset DTR bit in WR5-0
Send new WR5 value to SIO          (sets or resets TR line)
If bit 2=1 then set RTS bit in WR5-1
Else reset RTS bit in WR5-1
Send new WR5 value to SIO          (sets or resets SR line)
```

UPON ENTRY: E Reg - used in the calling routine - DO NOT ALTER

UPON EXIT: E Reg unaltered

CALLED BY: HSTINT

CALLS ROUTINES: none although returns to HSTINT by a jump

TMROFF - TIMER ON/OFF ROUTINE

EXTERNAL DESCRIPTION:

This routine is part of the Host interrupt Timer On/Off interrupt service routine. The purpose of this routine is to either turn the 16 millisecond Receive buffer timer on or off. A flag is used to determine whether it is already on or off. If the flag is off, this routine turns the timer on and if the flag is on this routine turns the timer off. The flag is changed accordingly at the end of the routine.

INTERNAL DESCRIPTION:

```
Retrieve the Timer flag
If Timer flag=1          (timer is already on)
  Turn off CTC timer
```

```
Timer flag=0          (update timer flag)
If Timer flag=0      (timer is off)
    Retrieve CTC Channel Control Word
    Send to CTC
    Retrieve Time Constant Register value
    Send to CTC          (restarts timer)
    Timer flag=1        (update timer flag)
Return to caller
```

UPON ENTRY: E Reg - used in the calling routine - DO NOT ALTER

UPON EXIT: E Reg Unaltered

CALLED BY: HSTINT

CALLS ROUTINES: none although returns to HSTINT by jump

SELF TEST

The purpose of this chapter is to give a detailed explanation of the Self Test firmware. Self Test is that portion of code which attempts to functionally test all accessible hardware on the board. It includes a ROM test, a RAM test, a CTC test, and a SIO test. Self Test is approximately 2K bytes long and resides in the EPROM beginning at address 0000. There are two ways that Self Test can be invoked: it is automatically invoked during power up when the Auto Reset line on the Z-80 is pulled and it may be invoked by a Self Test interrupt from the host.

The following sections will contain a detailed explanation of each of the component tests in the Self Test. I'm using the term "component" rather loosely here to refer piece of hardware that I can separately test. The "components" are actually tested in the same order in which they will be explained in this document.

SELF TEST INITIALIZATION

The following tasks are done before the any of the hardware component tests:

1. Interrupts are disabled, the stack pointer is initialized, and the reset bit in the Reset/I.D. register is cleared. The reason for the latter is that the state of this bit is indeterminate after power up and must be cleared for a Soft Reset to be issued (which is tested in Self Test).

2. The CTC and SIO channels are all reset. This is really precautionary. They should all have been reset automatically during the power up.

3. The IX register which is used to identify the type of failure (in case Self Test fails) is set to zero. As each test is performed, the IX register is incremented. The value it contains upon Self Test failure identifies which test it failed on.

4. The COMMAND register is first set to zero, then read back into the A register. This is to insure that it is both cleared and that the interrupt line is reset (remember that a write from the host to the COMMAND register sends an interrupt to the card and a read from the card to the COMMAND register clears it. Also, the state of the COMMAND register is indeterminate after power up so this puts it in a known state).

NOTE: The COMMAND register cannot really be tested in Self Test because, as explained above, the host must write to it to generate an interrupt.

RESERVED ADDRESSES (038H, 066H, INTERRUPT VECTORS)

There are a few reserved locations in the first part of the Self Test. These are addresses that have (or might have) fixed significance in certain circumstances. There are only two that are fixed; addresses 038H and 066H. The interrupt vectors for the CTC test are placed in between (addresses 048H to 057H) simply because there was room here and the Self Test jumps around this section of addresses (035H to 06DH).

Address 038H - This address is the one triggered if the Z-80 ever gets the value OFFH as an opcode. If the program ever jumps outside the

legal address space, ie physical memory, the value the Z-80 gets will most likely be OFFH (tri-state line assuming high) in which case this will be the address which is jumped to. The routine at this address adds 100H to whatever is in the IX register to identify the error and then jumps to the Self Test failure section of code.

Address 066H - This address is the one triggered when the NMI (non maskable interrupt) line on the Z-80 is pulled. Setting bit 7 in the Reset/I.D. register causes an NMI. The code at this location disables interrupts, calls the INIT routine, reenables interrupts and returns from the NMI. Recall that setting bit 7 in the Reset/I.D. register is called a Soft Reset. When a Soft Reset is issued by the host, a jump is made to the INIT routine and THERE IS NO RETURN FROM THE NMI. The Init routine jumps over the return in this circumstance. However, to test the NMI, the

Self Test also issues an Soft Reset. In this case, due to a value set in a test variable, the init routine returns control back to the calling routine and the RETN instruction at location 06BH is executed.

INT_COND AND INTERRUPT REGISTER TEST

The INT_COND and INTERRUPT registers are tested together because the function of INT_COND register impacts the INTERRUPT register. In other words, they are intertwined in some respects. The INTERRUPT test is split into two parts in the firmware separated by the NMI and RESET/I.D. register test. The reason for this is that the first part of the test writes to the INT_COND register which causes the IRQ (bit 6) bit to be set in the INTERRUPT register. Keeping in mind that we do not want to send an interrupt to the host, the only way to clear this is either a read from the host or a reset. Since the NMI test causes a Soft Reset, this bit is cleared. Then the second part of the INTERRUPT register test is performed.

TEST OUTLINE

Increment IX

Clear bit 7 (IEN) of the INTERRUPT register

Read INTERRUPT register

If bit 7 <> 0 then jump to Self Test error routine

Write to the INT_COND register (should set the IRQ bit in INTERRUPT reg)

Read the INTERRUPT register

Should be IEN=0 (bit 7) and IRQ=1 (bit 6). If not, jump to the Self Test error routine

NOTE: AT THIS POINT THE TEST IS SEPARATED BY THE NMI & RESET/I.D. TEST

Set bit 7 of the INTERRUPT register

Read INTERRUPT register

Should be IEN=1, IRQ=0. If not, jump to the Self Test error routine

Clear the INTERRUPT register

Read the INTERRUPT register

Should be IEN=0, IRQ=0

NMI AND RESET/I.D. TEST

The first portion of this test masks off the upper 3 bits of the Reset/I.D. register and tests the remaining 5 bits for correct card I.D. The correct card I.D. for the FORDYCE card is 5. The second portion of this code causes a Soft Reset to the Z-80 and tests whether

an NMI is actually caused. A value (SVAL) is written to a test register before the NMI is executed. If the NMI executes correctly, control will be passed to the init routine where the value in the test register will be changed to match EVAL. The SVAL and EVAL matching algorithm works in the following manner: if the init routine finds the value SVAL in the test register, it changes the test register value to EVAL and returns to the calling routine. When the NMI test has regained control, it verifies that the routine actually executed the NMI by identifying the value EVAL in the test register which was set by the init routine. If the init routine is called and the test register does not have the value, SVAL in the test register, the init routine executes the rest of its routine and does not return to the caller.

TEST OUTLINE

Increment IX

Retrieve value in RESET/I.D. register

Mask off bit 7 (bits 5 and 6 are hardwired to 0)

If lower bits <> 5 then jump to Self Test error routine

Load Test with Sval

Cause a Z-80 reset by setting bit 7 in the RESET/I.D. register

Wait for return from interrupt

Retrieve value in Test

Compare with EVAL. If different, jump to Self Test error routine

Clear Test register

SEMAPHORE REGISTER TEST

There are basically three parts to the Semaphore register test. The first part puts the register into a known state by writing to it and testing that the write set it to zero. The second part of the test checks the Semaphore register again to see if the read (which was performed to check the write results) set it. The third part of the test is another write to the Semaphore register and a check to see if bit 7 went from a 1 to 0 correctly. The Semaphore register is left set (bit 7=1) by the last read. It will be cleared in the Initialize routine. The reason for this is added protection against the host attempting to send or receive an interrupt before the card has completed its Self Test and Initialize routines.

NOTE: Remember that if the Semaphore register is initially reset

(bit 7=0), reading it will return the value with bit 7=0, but the act of reading the Semaphore register will have set bit 7 to 1. A second read would confirm this.

TEST OUTLINE

Increment IX

Write to the Semaphore register

Read the Semaphore register

If bit 7=1 then jump to Self Test error routine

Read the Semaphore register

If bit 7=0 then jump to Self Test error routine

Write to the Semaphore register

Read the Semaphore register

If bit 7=1 then jump to Self Test error routine

ROM TEST

This test performs a CRC using the polynomial $X^{16}+X^2+X+1$ and checks it against a previously calculated CRC already stored in the upper two bytes of ROM. This check character is generated by the program CRC4K which is run on the object code before it is burned into the ROM. The check character is stored with the low order byte first.

RAM TEST

This performs a test of the static RAM for stuck-at-0 and stuck-at-1 faults and address decoder failures. The test basically consists of four stages. In the first, a pattern is written to every location in RAM. The second pass consists of reading each location and checking the value read against the pattern written. An alternate pattern is then written to every location. The final is pass is a second read of each location, checking each value read against the second pattern written.

TEST OUTLINE

Increment IX

Write 01010101 to each RAM location

I=0

While (I=I+1) <= end of RAM Do

Begin

If RAM(I) <> 01010101 then jump to Self Test error routine

RAM(I)=10101010

End

I=Index of last RAM location

```
While (I=I-1) >= Beginning of RAM Do
  Begin
    If RAM(I) <> 10101010 then jump to Self Test error routine
    RAM(I) = 01010101
  End
```

Reset Stack Pointer with Stack Address

CTC TEST

There are basically two CTC tests, both of which are executed on all four channels of both CTC's. The first test checks the downcounting ability of the CTC channels by setting the CTC time constant to a known value, then checking whether it downcounts correctly within a known time period. The second test checks the timer ability and interrupt priority each of the CTC channels by setting them up in sequence and letting them downcount to zero, cause an interrupt, and jump to the test interrupt vector, altering the vector for the next interrupt expected. This is a very tricky test with strict timing constraints. For further explanation see below.

The two tests are performed on the two CTC's in the following order: Algorithm 1 (the first test) is performed on both CTC first, then Algorithm 2.

TEST OUTLINE FOR ALGORITHM #1 - This test is done on 1 CTC at a time

Increment IX

Reset all CTC channels

Set up interrupt vectors in RAM with the data in the ROM test interrupt vectors for the CTC - 8 locations with a jump instruction to a CTC

error routine (in the first test, if the downcounter counts to zero, its an error and the CTC error routine will call the Self Test error routine).

Set the Interrupt mode to 2

Enable interrupts

NOTE: For the first part of this test, all channels must be read before 256 T states have elapsed from the time each channel is started - i.e. before the time constant register downcounts.

For each channel . . .

 Load Channel Control Word (interrupt enabled, timer mode, prescale value = 256)

 Load Time Constant Register = 0

For each channel . . .

 Read Time Constant Register

 If <> 0 then jump to Self Test error

NOTE: For the second part of this test, all channels must be read after 256 T states but before 512 T states from the time the channel is started. Remember that the time constant register was originally set to zero, so when it downcounts (after 256T states) it will be 255.

For each channel . . .

 Read Time Constant Register
 If <> 255 then jump to Self Test error

Execute this test a second time (without resetting the timing)

TEST OUTLINE - ALGORITHM #2

NOTE: Remember that Algorithm #1 set the jump-to-error-routine instructions in RAM locations and passed the RAM address to the CTC as interrupt vector addresses. In the following test, each CTC channel is set to interrupt in a controlled sequence. The sequence should be channel 2, followed by channel 0, then channel 1, finally channel 3 (interrupt priority also determines sequence). The sequence is verified by giving a different interrupt vector address to the channel which is expected to interrupt. This new vector address points to a routine which changes the interrupt vector address of the second channel which is supposed to interrupt. In other words, if the channels interrupt in the right sequence the interrupt does not jump them to the error routine originally pointed to. Also, each alternate interrupt routine sets a bit in the B register which verifies that the routines were actually executed.

Increment IX

Load all CTC channels - interrupt enabled, prescale value = 16, timer mode

Load address of second interrupt routine into RAM CTC vector location
Load Time Constant for each channel (these are carefully calculated as is the order that the channels are triggered to insure the correct interrupt sequence.)

Enable interrupts

Wait (2 NOP instructions)

Disable interrupts

If B register <> OF0H then call Self Test error routines

SECOND INTERRUPT ROUTINE EXPLANATION

Reload vector address for this channel with old error routine address
Reset channel

Set bit in B register

Load RAM vector address for next expected channel with address of
the second interrupt routine for that channel

Enable interrupts

Return from interrupt

SIO TEST

This test basically performs loopbacks on the transmit, receive, and modem lines of all of the SIO ports. The test has three parts. The first part tests to see whether there are loopback hoods on the ports. For channels 1-3 the CTS line is tested. If set (equals '1'), a test hood is present. For channel 0, data is looped from the IC line and read back in on the SR line. If the same pattern is read that was sent, a test hood is present on this channel too. Whenever a test hood is detected present, a bit is set in the E' register. Bits 0-3 in E' represent channels 0-3. A '1' in any of these bit positions means that a hood is attached to the channel represented by that bit.

The second part of the test will perform an "internal" loopback which disables the frontplane drivers and sends data out the SIO and back again. Internal loopback is determined by the status of the RTS line on ch. B in SIO #1. If RTS=1, the frontplane drivers are disabled and the TX lines loop back into the RX lines.

The final part of the SIO test depends on whether or not there are any test hoods on any of the ports. The frontplane drivers are reenabled and the E' register is tested to determine which (if any) ports have a loopback hood attached to them. Data is sent out the channels with loopback hoods attached and read back in on the RX lines. It should be noted that the loopback tests use the "poll" mode of the SIO. In other words, the firmware sends data out the TX lines on each channel and loops on the RX Character Available bit in Read Register 0 of the SIO. A deadman timer of approx. 8 milliseconds is set initially to insure that the code does not loop forever in case there is an error.

External loopback on channel 0 is a little different from the rest of the channels because it is a modem port. The modem lines which are used are physically located on both channels of SIO #0 (see the firmware ERS for the designation of the SIO output lines) because of the number needed. Logically, the following loopbacks are performed if a hood is present on channel 0:

OUTPUT LINE	to	INPUT LINE
-----		-----

TX	RX
RS (CH. A - RTS)	CS (CH. A - CTS)
TR (CH. A - DTR)	DM (CH. A - SYNC)
SR (CH. B - RTS)	RR (CH. A - DCD)

- NOTE: 1. All of the modem lines for port 0 are in SIO # 0.
2. The IC & SR lines are also connected to each other in the test hood. However, since they were used to detect the presence of the test hood in the first place, they are not tested again.

TEST OUTLINE

Reset all CTC channels and program for 19.2K baud
 Set up interrupt vector for deadman timer on CTC #1, CH. 2
 Program all SIO channels - 8 bit, 1 stop bit, no parity
 Hood detect for channel 0-3. Set bit in E' register if hood is present.

Internal loopback

```

  Start deadman timer
  Enable interrupts
  For I=1 to 8 do
    Wait until TX buffer empty
    If RX Character Available bit set (Read Reg 0) then jump to
      Self Test error
    Send OAAH on TX line
    Wait until RX Character Available bit set
    Read in character
    If <> character sent go to Self Test error routine
    Complement test character (now 055H)
  
```

External Loopback

For each port - test corresponding bit in E' register to see if

hood is present.

If hood is present, loop data from input line to output line. If input data does not = output data, jump to Self Test failure routine.

NOTE: There are two subroutines used to loop back data, LOOP-TEST and SIO-TEST. LOOP-TEST is used for loopbacks in the hood detect section and the modem line loopback. It turns on and off the designated modem line in the designated SIO channel according to a fixed pattern and matches the input modem line signal with the output signal. SIO-TEST on the other hand sends a character out the SIO transmit line as described above in the Internal Loopback explanation.

SUCCESSFUL TERMINATION OF SELF TEST

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

1. The ST-COND register will contain the value E0 (hex) 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

UNSUCCESSFUL TERMINATION OF SELF TEST

Upon unsuccessful termination of self-test, the following will occur:

1. The ST-COND register receives the value in the IX register. This value indicates where the test failed.
2. The system console will display an error message identifying the card by ID number and select code, and specify a number which indicates the type of failure (the value in the ST-COND register).
3. The card sends the host a Self Test Complete interrupt
4. The card will execute (or attempt to execute) the initialization routine.

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 test hood - external loopback)
=17: SIO 1 CH B Test (with test hood - external loopback)

Notes on 98642 Firmware revisions

The Firmware source code is the latest revision. It is the code contained in the EPROM marked 98642-81004. It is very slightly different from the revision marked 98642-81003. The differences are noted and commented in the source code listing and the IMS.

PCS 2236

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

```
1      <870902.0106>
2      SOURCE: &STV1
3      PRGRM: EAP
4
5      HP-DIO 98642A 4-CHANNEL MUX SELF-TEST
6
7      ****
8      * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1982. ALL RIGHTS *
9      * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED *
10     * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT *
11     * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY. *
12     ****
13
14     LIST B,0,S
15     NLIST M,†
16
17     NAME MX4ST
18
19
20     PUBLIC MX4ST,CTC_ERRO
21
22     EXTRN INIT,ST_COND,TEST
23
24
25
26
27
28     HP-DIO 4-CHANNEL MUX CARD SELF-TEST
29
30     The following is the self-test code for the MUX Card. It performs
31     tests on the following major components and data paths on the card:
32     1. ROM
33     2. RAM
34     3. CTC
35     4. SIO
36
37     Self-test is invoked at EVERY card reset (at entry MX4ST).
38
39     Upon successful completion, the following state of the card
40     will result:
41     1. Z80 interrupts are enabled.
42     2. SIO is reset
43     3. CTC is reset
44     4. Interrupt mode = 2
45
46     Upon unsuccessful completion, Self-Test will put the value of the
47     IX register into the ST_COND register. The value in the IX register
48     indicates where the Self Test failed (see the IX values below for
49     their interpretation). After saving the IX register in the ST_COND
50     register, A Self Test Done interrupt is sent to the host, and the
51     self test executes (or attempts to) the initialization routine. In
52     other words, the card is left in basically the same state upon Self
53     Test failure that it is upon a successful completion of Self Test.
54
55     'IX' values are defined as follows:
56     IX = 1: INT_COND/INTERRUPT register Test
```

```

55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
      = 2: NMI/RESET I.D. register Test
      = 3: Semaphore 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)
=1xx: Jumped outside of address space

```

REGISTER USAGE DESCRIPTION

REG E': First four bits are used to indicate whether or not test hoods are present on each of the 4 ports. If the bit corresponding to the port is set, a test hood is present on that port.

REVISION HISTORY:

BUG FIX - 06/13/85 - THE CARD WAS UNABLE TO RECEIVE INTERRUPTS AFTER SELF TEST HAD FAILED IF THE ERROR WAS BECAUSE A DEADMAN TIMER TIMED OUT (I.E. SIO TEST EXTERNAL LOOPBACK). THE REASON WAS THAT THE END OF THE DEADMAN TIMER INTERRUPT WAS A JUMP TO ST_ERR WHICH NEVER DID A RETI. THE PROBLEM WAS HANDLED BY INCLUDING THE FOLLOWING INSTRUCTIONS IN THE ST_ERR ROUTINE:

```

LD BC,INIT
PUSH BC
RETI

```

THE PUSH INSTRUCTION IS BECAUSE 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 WOULD POP THE STACK.

; Bug Fix 27-July-1988 by Randy Stout - RHO Production Engineering
After isolating an intermittent interrupt, Zilog admitted having a problem with the CTC. Apparently there is a race condition between the bit that enables interrupts and the bit that resets the CTC. If the interrupt is enabled before the reset, a pending interrupt causes the CTC to invoke the ISR.

This fix alters the original version by changing beginning of the ASYNC SIO TEST to set only channels 0 and 1 of CTC1 as baud rate generators. Channel 2 state is left from the end of the CTC TEST

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST SYSTEM EQUATES

PAGE 4

```

113
114
115
116
117 *****
118 *          SYSTEM EQUATES *
119 *****
120 ;
121 0000 ROM_BEG EQU 0           Addr of beginning of ROM
122 1FFF ROM_END EQU 1FFFFH      Addr of last byte of ROM
123 0002 ROM_SEG EQU 2           Number of 4K segments of ROM
124 ;
125 ;      RAM   EQUATES
126 C002 RAM_BEG EQU 0C002H      Addr of beginning of RAM
127 07FE RAM_SIZ EQU 0C7FFH-RAM_BEG+1 # of bytes in RAM
128 0007 RAM_SEG EQU RAM_SIZ/256 # of 256 bytes segments in RAM
129 C800 STK_ADDR EQU 0C800H      Stack address
130
131 ;
132 ;      I/O   EQUATES
133 0070 SIO_0_A0 EQU 70H         SIO #0 CHANNEL A DATA
134 0071 SIO_0_AC EQU 71H         SIO #0 CHANNEL A CONTROL
135 0072 SIO_0_B0 EQU 72H         SIO #0 CHANNEL B DATA
136 0073 SIO_0_BC EQU 73H         SIO #0 CHANNEL B CONTROL
137 00B0 SIO_1_A0 EQU 0B0H        SIO #1 CHANNEL A DATA
138 00B1 SIO_1_AC EQU 0B1H        SIO #1 CHANNEL A CONTROL
139 00B2 SIO_1_B0 EQU 0B2H        SIO #1 CHANNEL B DATA
140 00B3 SIO_1_BC EQU 0B3H        SIO #1 CHANNEL B CONTROL
141
142 ;
143 00D0 CTC_0_C0 EQU 0D0H        CTC #0 CH 0 PORT 0 BAUD RATE GENERATOR
144 00D1 CTC_0_C1 EQU 0D1H        CTC #0 CH 1 PORT 1 BAUD RATE GENERATOR
145 00D2 CTC_0_C2 EQU 0D2H        CTC #0 CH 2 UNUSED
146 00D3 CTC_0_C3 EQU 0D3H        CTC #0 CH 3 UNUSED
147 00E0 CTC_1_C0 EQU 0E0H        CTC #1 CH 0 PORT 2 BAUD RATE GENERATOR
148 00E1 CTC_1_C1 EQU 0E1H        CTC #1 CH 1 PORT 3 BAUD RATE GENERATOR
149 00E2 CTC_1_C2 EQU 0E2H        CTC #1 CH 2 INTERNAL TIMER
150 00E3 CTC_1_C3 EQU 0E3H        CTC #1 CH 3 UNUSED
151
152 ;
153 ;MISCELLANEOUS EQUATES

```

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST SYSTEM EQUATES

PAGE 5

```

154 8000 RESET EQU 8000H        ADDRESS OF HW RESET REGISTER
155 8001 INT_REG EQU 8001H      ADDRESS OF HW INTERRUPT REGISTER
156 8002 SEM_REG EQU 8002H      ADDRESS OF SEMAPHORE REGISTER
157 C001 COM_REG EQU 0C001H      ADDRESS OF COMMAND REGISTER
158 C000 INT_COND EQU 0C000H      ADDRESS OF INT_COND REGISTER
159 00E0 PASS EQU 0EH          VALUE INDICATING ST PASSED - ST_COND REG.
160 0010 INT_CODE EQU 10H        VALUE TO INT. HOST THAT ST. DONE
161 0034 SVRL EQU 34H          USED TO DETERMINE NMI ORIGIN-BEG. VALUE
162 0043 EVAL EQU 43H          SAME AS ABOVE EXCEPT THIS IS END VALUE
163 C010 CTC_V0 EQU 0C010H      BEG. CTC VECTOR LOCATION IN RAM - CTC 0
164 C030 CTC_V1 EQU 0C030H      BEG. CTC VECTOR LOCATION IN RAM - CTC 1
165
166 ;

```

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST

PAGE 6

```

168      CSEG
169 0000
170 0000 F3
171 0001 AF
172 0002 32 00 80
173 0005 31 00 C8
174
175 : . . . RESET ALL CTC & SIO
176
177 0008 06 04
178 000A 0E D0
179 000C 3E 03
180 000E
181 000E ED 79
182 0010 OC
183 0011 10 FB
185 0013 06 04
186 0015 0E E0
187 0017
188 0017 ED 79
189 0019 OC
190 001A 10 FB
192 001C 06 02
193 001E 0E 71
194 0020 3E 18
195 0022
196 0022 ED 79
197 0024 OC
198 0025 OC
199 0026 10 FA
201 0028 06 02
202 0029 0E B1
203 002C
204 002C ED 79
205 002E OC
206 002F OC
207 0030 10 FA
210 0032 C3 6E 00   C     JP DOIT      Jump around the following stuff.
211
212
213 : . . . WARNING WARNING WARNING

```

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST

PAGE 7

```

214
215
216
217
218
219
220
221
222
223
224 0035 FF FF
225 0037 FF
226 0038 F3
227 0039 01 00 01
228 003C DD 09
229 003E C3 B0 04   C     DEFU OFFFFH    SPACE FILLERS SO THAT THE FOLLOWING
230 0041 FF          DI      DEFB OFFH    ROUTINE WILL BEGIN AT 38H
231 0042 FF FF
232 0044 FF FF
233 0046 FF FF
234
235 : . . . CTC SELF-TEST INTERRUPT VECTOR TABLE
236
237 0048
238 0048 F2 02   C     CTCIT1:  DEFU CTC_ERRO
239 004A F2 02
240 004C F2 02
241 004E F2 02
242 0050 F2 02
243 0052 F2 02
244 0054 F2 02
245 0056 F2 02
246
247 0058 FF FF
248 005A FF FF
249 005C FF FF
250 005E FF FF
251 0060 FF FF
252
253
254 : . . . NMI
255
256 0062 FF FF
257 0064 FF FF
258 0066 F3
259 0067 CD 00 00   E     DEFU OFFFFH    FILLERS - NMI ROUTINE MUST BE AT 066H
260 006A FB
261 006B ED 45
262 006D FF

```

264 006E DOIT:

CALL INIT CALL THE INIT SUBROUTINE

EI

RETN

DEFB OFFH

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST

PAGE 8

266 006E DD 21 00 00 LD IX,0 INITIALIZE TEST COUNTER
267 0072 D9 EXX
268 0073 01 00 00 LD BC,0 INITIALIZE ALTERNATE REGISTER SET
269 0076 11 00 00 LD DE,0
270 0079 21 00 00 LD HL,0
271 007C D9 EXX

273
274 . . . CLEAR THE COMMAND REGISTER
275 THEN READ TO SET IT
276
277 007D AF XOR A CLEAR A REGISTER
278 007E 32 01 C0 LD (COM_REG),A SET COM_REG TO ZERO - MAY NOT BE ALREADY
279 0081 3A 01 C0 LD A,(COM_REG) THIS IS TO INSURE THAT INT. LINE IS HIGH

281
282
283
284
285
286
287
288
289
290
291
292
293 ; INT_COND AND INTERRUPT REGISTER TEST

THIS TEST IS DIVIDED IN TWO PARTS. THE FIRST PART TESTS THE IEN BIT (BIT 6) IN THE INTERRUPT REGISTER BY MAKING SURE THAT A WRITE TO THE INT_COND REGISTER SETS IT. HOWEVER, SINCE WE DO NOT WANT TO ACTUALLY SEND AN INTERRUPT TO THE HOST AND SINCE THE IRQ BIT (BIT 7) ALSO NEEDS TO BE TESTED, THE PORTION OF THE TEST THAT TESTS THE SETTING AND RESETTING OF THE IRQ BIT IS PERFORMED AFTER THE NMI TEST. THIS IS BECAUSE THE NMI TEST GENERATES A SOFT RESET TO THE CARD AND CAUSES THE IEN BIT TO BE CLEARED WHICH IS SOMETHING I CAN'T DO FROM THE CARD SIDE. REMEMBER THAT HAVING THE IEN AND IRQ BITS SET AT THE SAME TIME TRIGGERS AN INTERRUPT TO THE HOST.

295 0084 DD 23 INC IX
296 0086 AF XOR A CLEAR BIT 7 OF THE STATUS REG.
297 0087 32 01 80 LD (INT_REG),A
298 008A 3A 01 80 LD A,(INT_REG)
299 008D E6 80 AND 80H MAKE SURE THAT BIT 7 (IEN) IS CLEARED
300 008F FE 00 CP 0
301 0091 C2 B0 04 C JP NZ,ST_ERR

303 0094 3E 80 LD A,80H
304 0096 32 00 C0 LD (INT_COND),A GENERATE INTERRUPT TO HOST
305 0099 3A 01 80 LD A,(INT_REG) BY WRITING TO THE INT_COND REG.
306 009C E6 C0 AND 0C0H MASK OFF BITS 6 & 7 IN STAT REG.
307 009E FE 40 CP 40H SHOULD BE IEN=0, IRQ=1
308 00A0 C2 B0 04 C JP NZ,ST_ERR

310 ;

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST

PAGE 9

311
312
313
314
315
316
317
318
319
320 ; NMI AND RESET/I.D. REGISTER TEST

THE FIRST PORTION OF THIS TEST MASKS OFF THE UPPER BIT OF THE RESET/I.D. REGISTER AND TEST THE REMAINING SEVEN BITS FOR CORRECT CARD I.D. THE CORRECT CARD ID FOR THE FORDYCE CARD IS 5. THE SECOND PORTION OF THIS CODE CAUSES A SOFT RESET TO THE Z-80 AND TESTS WHETHER AN NMI IS CAUSED. A VALUE (SVAL) IS PUT INTO THE TEST REGISTER BEFORE NMI IS INVOKED. IF NMI EXECUTES CORRECTLY CONTROL WILL BE PASSED TO THE INIT ROUTINE WHERE THE VALUE IN THE TEST REGISTER WILL BE CHANGED TO MATCH THE PATTERN IN EVAL.

322 00A3 DD 23 INC IX
323 00A5 3A 00 80 LD A,(RESET)
324 00A8 E6 7F AND 7FH MASK OFF REMOTE BIT- SEE IF LOWER 5 BITS
325 00A9 FE 05 CP 5 ARE CORRECT CARD ID
326 00AC C2 B0 04 C JP NZ,ST_ERR IF LOWER BITS <> 5 THEN ERROR

327
328 00AF 3E 34 LD A,SVAL STARTING VALUE TO TEST REG.
329 00B1 32 00 00 E LD (TEST),A PUT STARTING VALUE INTO TEST REGISTER
330 00B4 3E 80 LD A,80H
331 00B6 32 00 80 LD (RESET),A CAUSE A Z-80 RESET BY SETTING BIT 7
332 ; IN THE RESET REGISTER
333 00B9 00 NOP
334 00B0 00 NOP WAITING FOR NMI TO COME THROUGH
335 00B1 3A 00 00 E LD A,(TEST) CHECK IF TEST REG. WAS CHANGED IN INIT
336 00B6 FE 43 CP EVAL
337 00C0 28 03 JR Z,NMI1 JUMP IF TEST WAS SUCCESSFUL
338 00C2 C3 B0 04 C JP ST_ERR IF NOT, GOTO ST_ERR

340 00C5 AF NMII: XOR A CLEARING THE TEST REG.
341 00C6 32 00 00 E LD (TEST),A

343
344
345
346
347 ; THE FOLLOWING PORTION OF CODE IS TO TEST BIT 7 IN THE STATUS OR INTERRUPT REGISTER (THE IEN LINE). THIS TEST SIMPLY SETS BIT 7, THEN READS IT BACK AGAIN.

348 00C9 21 01 80 LD HL,INT_REG
349 00CC CB FE SET 7,(HL) SET IEN BIT (7) IN STATUS REG.
350 00CE 3A 01 80 LD A,(INT_REG) READ STAT REG TO SEE IF BIT SET
351 00D1 E6 C0 AND 0C0H MASK OFF BITS 6 & 7
352 00D3 FE 80 CP 80H SHOULD BE IEN=1, IRQ=0
353 00D5 C2 B0 04 C JP NZ,ST_ERR

355 00D8 AF XOR A CLEAR A SO CAN CLEAR IEN BIT
356 00D9 32 01 80 LD (INT_REG),A
357 ;*****
358 ; Update 8/10/88 by Randy Stout - RMO Production Engineering

ERR LINE ADDR B1 B2 B3 B4

MUX SELF-TEST

PAGE 10

```

359      ; Added the following line to assure INT_REG was really checked
360  00DC  3A 01 80    LD A,(INT_REG)  Get content of INT_REG
361      ;*****SHOULD BE IEN=0, IRQ=0*****
362  00DF  E6 C0    AND 0COH
363  00E1  FE 00    CP 0
364  00E3  C2 B0 04    JP NZ,ST_ERR
C

```

```

366
367
368
369
370
371
372
373
374
375
376
377
378
379

```

SEMAPHORE REGISTER TEST:

The following is a test for the Semaphore register. There are basically four parts to this test. The first part puts the register into a known state by writing to it and testing that the write set it to zero. The second part of the test then checks the Semaphore register again to see if the read (to check the write results) set it. The final part of the test is another write to the Semaphore register and a check to see if it went from a 1 to 0 correctly. The Semaphore register is left set (bit 7=1) by the last read. It will be cleared in the Initialize routine. The reason for this is added protection against the host attempting to send or receive an interrupt before the card has completed its Self Test and Initialize routines.

```

381  00E6  DD 23    INC IX
382  00E8  3E 80    LD A,80H   SETTING INITIAL VALUE = 0 B WRITING TO IT
383  00EA  32 02 80    LD A,(SEM_REG),A
384  00ED  3A 02 80    LD A,(SEM_REG)
385  00F0  CB 7F
386  00F2  C2 B0 04    C     BIT 7,A SEE IF SEMAPHORE BIT = 0
387      JP NZ,ST_ERR  IF NO, ERROR
388  00F5  3A 02 80    LD A,(SEM_REG)
389  00F8  CB 7F
390  00FA  CA B0 04    C     BIT 7,A GET CONTENTS OF SEM_REG AGAIN
391      JP Z,ST_ERR  A REG ALREADY HAS ADDRESS OF SEM_REG
392  00FD  3E 80    LD A,80H
393  00FF  32 02 80    LD A,(SEM_REG),A
394  0102  3A 02 80    LD A,(SEM_REG)
395  0105  CB 7F
396  0107  C2 B0 04    C     BIT 7,A DID THE WRITE CLEAR THE SEM. REG?
397      JP NZ,ST_ERR  IF NO, ERROR

```

```

398
399
400
401

```

NOTE: THE SEMAPHORE REGISTER IS LEFT SET FROM THIS TEST. IT IS CLEARED AT THE END OF INIT. THIS WAY IT IS AN ADDED PREVENTION IN CASE THE HOST TRIES TO SEND AN INTERRUPT BEFORE THE CARD IS READY.

ERR LINE ADDR B1 B2 B3 B4

MUX ROM TEST

PAGE 11

```

405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441

```

ROM TEST

PERFORMS A CRC USING THE POLYNOMIAL $X^{16}+X^{12}+X^5+1$. IT EXPECTS A 16 BIT CHECK CHARACTER TO BE STORED IN THE UPPER BYTES OF ROM. THE CHECK CHARACTERS ARE STORED WITH THE LOW ORDER BYTE FIRST.

THE MOST SIGNIFICANT BIT OF THE CRC POLYNOMIAL IS THE LEAST SIGNIFICANT BIT OF THE Z80 NUMBER.
E IS THE HIGH ORDER CRC REMAINDER.
D IS THE LOW ORDER CRC REMAINDER.
THIS ALGORITHM, WITH S <- (E XOR CHAR)* X^{16} AND
T <- D*X 16 , CALCULATES T*X 16 +S*X 12 +S*X 5 +S.

Note: The polynomial $X^{16}+X^{12}+X^5+1$ has the same "goodness" (i.e., error detection wise) as the CRC-CCITT polynomial $X^{16}+X^{12}+X^5+1$ which is slightly better than the CRC-16 polynomial $X^{16}+X^{12}+X^5+1$ because of the algorithm execution speed.

RELEVANT REGISTER VALUES UPON UNSUCCESSFUL COMPLETION:

1. (ROM_SEG - B') = 4K SEGMENT WITH CRC ERROR.
2. SEGMENTS NUMBERED FROM 0
3. DE = COMPUTED CRC FOR SEGMENT

GLOBAL EQUATES:

ROM_BEG: THE STARTING ADDRESS OF ROM
ROM_END: ADDRESS OF THE LAST BYTE IN ROM
ROM_SEG: THE NUMBER OF 4K SEGMENTS IN ROM

```

443  010A  DD 23    C  ROM: EQU $           ROM: Inc test counter
444  010A          :           The algorithm assumes that RAM has not yet been
445          :           tested and is therefore not usable. Both the
446          :           normal and alternate register sets are used.
447          :
448
449  010C  21 00 00    LD HL,ROM_BEG  HL->Start of ROM
450  010F  D9          EXX
451  0110  06 02      LD B,ROM_SEG  B' = # 4K segments for DJNZ loop

```

ERR LINE ADDR B1 B2 B3 B4

MUX ROM TEST

PAGE 12

```

452 0112 21 FF 1F
454 0115 D9
455 0116 01 10 FE
456 0119 11 FF FF
457
458 ; Calculate CRC on the next 4094 characters.
459
460 011C 7E
461 011D AB
462 011E 5A
463 011F 57
464 0120 17
465 0121 AA
466 0122 0F
467 0123 0F
468 0124 E6 C0
469 0126 AB
470 0127 SF
471 0128 7A
472 0129 1F
473 012A AA
474 012B 1F
475 012C AA
476 012D 57
477 012E 23
478 012F 10 EB
479 0131 0D
480 0132 20 E8

482 ; DE contains the CRC. Switch to alternate register
483 ; set to check CRC against check character.

485 0134 7A
486 0135 D9
487 0136 4F
488 0137 D9
489 0138 7B
490 0139 D9
491 013A BE
492 013B 20 05
493 013D 2B
494 013E 79
495 013F BE
496 0140 28 04
497 0142 D9
498 0142 C3 B0 04
499 0143 C3 B0 04 C

ROM_10 LD HL,ROM_END HL'->1st CRC check char.
ROM_15 LD BC,0FE10H Set up loop for 4K-2 length CRC
LD DE,0FFFFH Initialize partial remainder

ROM_20 LD A,(HL) Get character
XOR E, Calculate CRC
LD E,D
LD D,A
RLA
XOR D
RRCA
RRCA
AND OC0H
XOR E
LD E,A
LD A,D
RRA
XOR D
RRA
XOR D
LD D,A
INC HL
DJNZ ROM_20 L 0
DEC C 0
JR NZ,ROM_20 P

DE contains the CRC. Switch to alternate register
set to check CRC against check character.

LD A,D
EXX
LD C,A D'=CRC low byte
EXX
LD A,E A=CRC high byte
EXX
CP (HL) high bytes compare?
JR NZ,ROM_40 If not!
DEC HL
LD A,C A=CRC low byte
CP (HL) low bytes compare?
JR Z,ROM_50 If OK

ROM_40: EXX
JP ST_ERR Go to Self Test Error routine

```

ERR LINE ADDR B1 B2 B3 B4

MUX ROM TEST

PAGE 13

```

501
502 ; . . . AT FAILURE THE REGISTER CONTENTS ARE:
503
504 REG BC: LOOP COUNTER WHICH SHOULD BE 0
505 REG DE: COMPUTED CRC IN ERROR
506 REG HL: ROM ADDR FOR NEXT ROM SEGMENT
507 REG B': ROM SEGMENT DOWN COUNTER
508 ; ROM_SEG - (B') = SEGMENT WHICH CONTAINS THE
509 ; ERROR (COUNT STARTS FROM 0)
510 REG HL': ADDR TO ROM CRC VALUE WHICH DID NOT MATCH
511 ; COMPUTED VALUE

ROM_50: DEC HL
DJNZ ROM_10 Loop ends after ROM_SEG times
EXX Back to regular register set

```

```

520 ****
521 THIS IS THE SHORT RAM TEST TAKEN FROM THE ASI SELF-TEST
522 Performs a test of static RAM for stuck-at-0 and stuck-at-1
523 faults and address decoder failures. The algorithm will
524 test the following:
525 1. Stuck-at-0 and stuck-at-1 faults in every location
526 in RAM.
527 2. Decoder failures where writing to one location in
528 RAM actually causes a write to another location (the
529 wrong location gets written to).
530 3. Failures which cause multiple locations to be updated
531 when a particular RAM location is written to.
532
533 Algorithm:
534
535 WRITE 01010101 to all RAM
536 I=0
537 WHILE (I=I+1) <= END OF RAM
538 DO BEGIN
539   IF RAM(I) <> 01010101
540     THEN GOTO ERROR
541   RAM(I) = 10101010
542 END
543
544 I=INDEX OF LAST RAM LOCATION
545 WHILE (I=I-1) >= BEGINNING OF RAM
546 DO BEGIN
547   IF RAM(I) <> 10101010
548     THEN GOTO ERROR
549   RAM(I) = 01010101
550 END
551
552 ERROR: PROCESS RAM ERROR
553
554 No relevant register values upon unsuccessful
555 completion.
556
557 ****

```

```

559 014A RAM:
560 014A DD 23 ; INC IX
561           ; Store 01010101 in all RAM locations.
563 014C 21 02 C0 LD HL, RAM_BEG    HL<- starting address
564 014F 3E 55 LD A, 01010101B
565 0151 77 LD (HL), A
566 0152 11 03 C0 LD DE, RAM_BEG+

```

```

567 0155 01 FD 07 LD BC, RAM_SIZ-1
568 0158 ED B0 LDIR
569
570 The following loop is used for both the scan of RAM
571 from bottom to top and from top to bottom. Each
572 RAM location is retrieved, verified to contain the
573 appropriate value, and then set with the complement
574 of that value. If RAM (n) does not match then the
575 following fault has been detected:
576 1. There is a stuck-at fault in RAM (n).
577 2. When RAM (n) was written to, a decoder
578    fault caused the value to be written
579    somewhere else.
580 3. When RAM (n-x) was written to with the
581    complement its value, RAM (n) was changed
582    also.
583 By performing the write and test operation in both
584 directions, all possible stuck-at and decoder
585 fault combinations can be tested.

```

```

587 015A 21 02 C0 LD HL, RAM_BEG    Start 1st from bottom to top
588 015D 01 07 00 LD BC, RAM_SEG    # of 256 byte segments
589 0160 16 55 LD D, 01010101B   D= Test pattern
590
591 0162 7E RAM_10: LD A, (HL)    A=RAM (n)
592 0163 BA CP D          match on pattern?
593 0164 C2 B0 04 JP NZ, ST_ERR  If not...ERROR
594 0167 2F CPL
595 0168 77 LD (HL), A    RAM (n) <-
596 0169 FE 55 CP 01010101B complement of RAM (n)
597 016B 20 03 JR NZ, RAM_15  1st time thru loop?
598 016D 28 DEC HL        If yes
599 016E 18 01 JR RAM_16  Top to bottom scan (n=n-1)
600 0170 23 RAM_15: INC HL
601 0171 10 EF RAM_16: DJNZ RAM_10 Bottom to top scan (n=n+1)
602 0173 0D DEC C          Loop to access RAM_SIZ bytes
603 0174 20 EC JR NZ, RAM_10 of RAM.(RAM_SIZ * RAM_SEG=256)
604
605 0176 01 07 00 RAM_20: LD BC, RAM_SEG Reset for 2nd pass thru loop
606 0179 28 DEC HL
607 017A 57 LD A
608 017B FE AA CP 10101010B Is 2nd pass already done?
609 017D 28 E3 JR Z, RAM_10 If not...scan top to bottom
610
611 OK TO USE RAM FOR STACK
612
613 017F RAM_XIT: ; RAM is now usable, establish stack
614

```

ERR LINE ADDR B1 B2 B3 B4

MUX RAM Test

616 017F 31 00 C8

LD SP,STK_ADDR

PAGE 16

ERR LINE ADDR B1 B2 B3 B4

MUX CTC TEST

PAGE 17

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

MUX CTC #0 TEST

The CTC test implements two algorithms to test the CTC:
Algorithm 1:
tests for stuck-at-1 & 0 in the time constant register,
the down counter, and the data lines. A 256 prescale is
also checked. All channels are set up with all zeros in
the time constant register (256). The down counter is
read before the first increment and then again after the
first increment. This is repeated with all one's initially
in the time constant register (255).

Algorithm 2:
This algorithm is only performed on the CTC which is
connected to the interrupt priority daisy chain (CTC #1).
It tests external trigger, more combinations of the channel
control register, ZC/T02, and some channel independence.
Also, the ability to hold off interrupts (an interrupt
should disable further interrupts) and correct daisy
chain interrupt servicing are checked. The interrupt
vector register is loaded such that channel 2, then
channel 3, then 2, then 0 interrupt. The order of service
should be 2,0,1, and 3. As each channel gains control, it
sets its interrupt vector to point to an error, resets
itself, adds to the checksum, sets next-channel-to-get-
control's interrupt vector to point to that routine,
enables interrupts, and returns.

649 0182 DD 23
650 0182 3E 03
651 0184 3E 03
652 0186 D3 D0
653 0188 D3 D1
654 018A D3 D2
655 018C D3 D3

C CTC: EQU \$
INC IX
LD A,03H RESET ALL CTC #0 CHANNELS
OUT (CTC_0_C0),A
OUT (CTC_0_C1),A
OUT (CTC_0_C2),A
OUT (CTC_0_C3),A

657

658

659

BEGIN ALGORITHM 1 - CTC #0

661 018E 3E C0
662 0190 ED 47
663 0192 3E 10
664 0194 D3 D0
665 0196 21 48 00
666 0199 11 10 C0
667 019C 01 08 00

C LD A,HIGH.CTC_V0 SET UP INT VECTORS FOR CTC #0
LD I,A
LD A,10H
OUT (CTC_0_C0),A SET UP VECTOR IN CTC WITH 10H
LD HL,CTCIT1 THIS IS ADDRESS OF INTERRUPT VECTORS
LD DE,CTC_V0
LD BC,8

ERR LINE ADDR B1 B2 B3 B4 MUX CTC TEST PAGE 18

```

668 019F ED B0      LDIR
669 01A1 ED SE      IM 2
670 01A3 AF          XOR A
671 01A4 FB          EI
672
673 01A5 11 00 B7      LD DE,0B700H D -> CHANNEL CONTROL REG
674                               INTS,TIMER_256PS, IN TRIG
675                               B -> TIME CONSTANT
676                               B=00=256, B=FF=255
677 01A8 01 D0 04      XCTCL0: LD BC,256*4+CTC_0_CO
678
679 01AB ED 51      XCTCL1: OUT (C),D LOAD CHANNELS
680 01AD ED 59      OUT (C),E TIMER LOADED W 0/PRESCAL.=256
681 01AF OC          INC C
682 01B0 10 F9      DJNZ XCTCL1
683
684 01B2 01 D0 04      LD BC,256*4+CTC_0_CO
685
686                               ALL CHANNELS MUST BE READ
687                               BEFORE 256T STATES ELAPSE
688                               FROM THE TIME THE CHANNEL
689 01B5 ED 58      XCTCL2: IN E,(C) IS STARTED. (ALL ARE READ
690 01B7 BB          CP E,BETWEEN 153 & 196 T)
691 01B8 C2 B0 04      JP NZ,ST_ERR A=EXPECTED(E=ACTUAL,C=CTC*)
692 01B8 OC          INC C
693 01BC 10 F7      DJNZ XCTCL2
694
695 01BE 3D          DEC A
696 01BF 01 D0 04      LD BC,256*4+CTC_0_CO
697
698                               ALL CHANNELS MUST BE READ
699                               AFTER 256T STATES, BUT BEFORE
700                               512T STATES HAVE ELAPSED FROM
701                               THE TIME THE CHANNEL IS
702 01C2 ED 58      XCTCL3: IN E,(C) STARTED. (ALL ARE READ
703 01C4 BB          CP E,BETWEEN 334 & 421 T)
704 01C5 C2 B0 04      JP NZ,ST_ERR A=EXPECTED(E=ACTUAL,C=CTC*)
705 01C8 OC          INC C
706 01C9 10 F7      DJNZ XCTCL3
707
708 01CB FE FE      CP OFEH
709 01CD 20 D9      JR NZ,XCTCL0 TEST RUN TWICE-T=D;T=255
710
711 01CF F3          DI
712
713
714
715
716

```

713

714

715

716

Begin Algorithm 2 - CTC #0

ERR LINE ADDR B1 B2 B3 B4 MUX CTC TEST PAGE 19

```

717
718
719
720
721
722 01D0 DD 23      INC IX ASSUME:
723 01D2 06 00      LD B,0 I=D0 CTC0 INT VEC REG=00
724 01D4 3E 97      LD A,97H INT TRANSFER VEC SET TO ERROR
725 01D6 D3 D0      OUT (CTC_0_CO),A IM 2, DI
726 01D8 D3 D1      OUT (CTC_0_C1),A
727 01DA D3 D2      OUT (CTC_0_C2),A
728 01DC D3 D3      OUT (CTC_0_C3),A
729 01DE 21 01 02      LD HL,IX12
730 01E1 22 14 CO      LD (CTC_V0+4),HL CLEAR CHECKSUM
731 01E4 3E 09      LD A,9 INT, TIMER,16PS,EXT TRIG
732 01E6 D3 D1      OUT (CTC_0_C1),A CH3 SHOULD TRY TO INT AFTER
733 01E8 D3 D0      OUT (CTC_0_C0),A CH2 & BEFORE CH1 OR CHO
734 01E9 3E 01      LD A,1 CH2 & BEFORE CH1 OR CHO
735 01EC D3 D3      OUT (CTC_0_C3),A PUT ADDR. OF INT. ROUTINE IN RAM
736 01EE D3 D2      OUT (CTC_0_C2),R VECTOR LOCATION
737 01F0 3E 03      LD A,3 TIME CONST SUCH THAT CH1
738 01F2 21 F2 02      LD HL,CTC_ERR0 FOLLOWED BY CHO TRY TO INT
739 01F5 FB          EI WHILE IN CH2 INT ROUTINE
740 01F6 00          NOP
741 01F7 00          NOP
742 01F8 F3          DI
743 01F9 3E F0      LD A,0FOH
744 01FB B8          CP B
745 01FC C2 B0 04      JP NZ,ST_ERR
746 01FF 18 3C      JR XOK

```

748

749

750

751

752

753 0201 22 14 CO XI2: LD (CTC_V0+4),HL POINT INT VEC TO ERROR
754 0204 D3 D2 OUT (CTC_0_C2),A RESET CHANNEL
755 0206 CB CO SET 0,B ADD TO CHECKSUM
756 0208 11 11 02 LD DE,XI0 SET INT TRANSFER VEC
757 0208 ED 53 10 CO LD (CTC_V0),DE OF NEXT CHANNEL
758 020F 18 27 JR XIC
759
760 0211 22 10 CO XI0: LD (CTC_V0),HL EACH OF THESE INTERRUPT ROUTINES
761 0214 D3 D0 OUT (CTC_0_C0),A SETS THE ADDRESS OF THE NEXT
762 0216 CB D0 SET 2,B ISR EXPECTED INTO ITS PROPER
763 0218 11 21 02 LD DE,XI1 RAM LOCATION, REPLACING THE
764 0218 ED 53 12 CO LD (CTC_V0+2),DE EXISTING JP CTC_ERR0 INSTRUCTION.
765 021F 18 17 JR XIC

ERR LINE ADDR B1 B2 B3 B4

MUX CTC TEST

PAGE 20

```

767 0221 22 12 C0      XI1: LD (CTC_V0+2),HL
768 0224 D3 D1          OUT (CTC_0_C1),A
769 0226 CB E0          SET 4,B
770 0228 11 31 02      C   LD DE,XI3
771 022B ED 53 16 C0      LD (CTC_V0+6),DE
772 022F 18 07          JR XIC
773
774 0231 22 16 C0      XI3: LD (CTC_V0+6),HL
775 0234 D3 D3          OUT (CTC_0_C3),A
776 0236 CB F0          SET 6,B
777 0238 CB 20          XIC: SLA B
778 023A FB             EI
779 023B ED 4D          RETI
780
781 023D               C   XOK: EQU $

783
784
785      . . . REPEAT CTC ALGORITHM 1 FOR CTC #1
786

788 023D DD 23          INC IX
789 023F 3E 03          LD A,03H      RESET ALL CTC #1 CHANNELS
790 0241 D3 E0          OUT (CTC_1_C0),A
791 0243 D3 E1          OUT (CTC_1_C1),A
792 0245 D3 E2          OUT (CTC_1_C2),A
793 0247 D3 E3          OUT (CTC_1_C3),A

795 0249 3E C0          LD A,HIGH.CTC_V1    SET UP INTERRUPT VECTORS IN RAM FOR
796 024B ED 47          LD I,A          CTC #1
797 024D 3E 30          LD A,30H      PUT CTC VECTOR ADDRESS IN
798 024F D3 E0          OUT (CTC_1_C0),A      CTC.
799 0251 21 48 00      C   LD HL,CTCTT1
800 0254 11 30 C0      LD DE,CTC_V1
801 0257 01 08 00      LD BC,8
802 025A ED 80          LDIR
803 025C AF             XOR A      CLEAR A - USED IN DOWN CNTR CHECK
804 025D FB             EI

805
806 025E 11 00 B7      LD DE,0B700H    D -> CHANNEL CONTROL REG
807
808
809
810 0261 01 E0 04      XCTCL0R: LD BC,256*4+CTC_1_C0
811
812 0264 ED 51          XCTCL1A: OUT (C),D      LOAD CHANNELS
813 0266 ED 59          OUT (C),E
814 0268 OC             INC C
815 0269 10 F9          DJNZ XCTCL1A

```

ERR LINE ADDR B1 B2 B3 B4

MUX CTC TEST

PAGE 21

```

816
817 026B 01 E0 04      LD BC,256*4+CTC_1_C0
818
819
820
821
822 026E ED 58          XCTCL2A: IN E,(C)      ALL CHANNELS MUST BE READ
823 0270 BB              CP E          BEFORE 256T STATES ELAPSE
824 0271 C2 B0 04      C   JP NZ,ST_ERR  FROM THE TIME THE CHANNEL
825 0274 OC              INC C          IS STARTED. (ALL ARE READ
826 0275 10 F7          DJNZ XCTCL2A  BETWEEN 153 & 196 T)
827
828 0277 3D
829 0278 01 E0 04      LD BC,256*4+CTC_1_C0
830
831
832
833
834
835 027B ED 58          XCTCL3A: IN E,(C)      ALL CHANNELS MUST BE READ
836 027D BB              CP E          AFTER 256T STATES, BUT BEFORE
837 027E C2 B0 04      C   JP NZ,ST_ERR  512T STATES HAVE ELAPSED FROM
838 0281 OC              INC C          THE TIME THE CHANNEL IS
839 0282 10 F7          DJNZ XCTCL3A  STARTED. (ALL ARE READ
840
841 0284 F3              DI
842
843
844
845
846
847
848
849
850
851
852 0285 DD 23          INC IX
853 0287 06 00          LD B,0
854 0289 3E 97          LD A,97H      CLEAR CHECKSUM
855 028B D3 E0          OUT (CTC_1_C0),A  INT,TIMER,16PS,EXT TRIG
856 028D D3 E1          OUT (CTC_1_C1),A  CH3 SHOULD TRY TO INT AFTER
857 028F D3 E2          OUT (CTC_1_C2),A  CH2 & BEFORE CH1 OR CHO
858 0291 D3 E3          OUT (CTC_1_C3),A  CH2 & BEFORE CH1 OR CHO
859 0293 21 B6 02      C   LD HL,XCTCI2  PUT ADDR. OF INT. ROUTINE IN RAM
860 0296 22 34 C0          LD (CTC_V1+4),HL  VECTOR LOCATION
861 0299 3E 09          LD A,9
862 029B D3 E1          OUT (CTC_1_C1),A  TIME CONST SUCH THAT CH1
863 029D D3 E0          OUT (CTC_1_C0),A  FOLLOWED BY CHO TRY TO INT
864 029F 3E 01          LD A,1
865 02A1 D3 E3          OUT (CTC_1_C3),A  WHILE IN CH2 INT ROUTINE

```

ERR	LINE	ADDR	B1	B2	B3	B4	MUX	CTC	TEST	PAGE	22
866		02A3	D3	E2				OUT	(CTC_1_C2),A		
867		02A5	3E	03				LD	A,3	A <- CHAN CONTROL RESET	
868		02A7	21	F2	02		C	LD	HL,CTC_ERR0		
869		02AA	FB					EI		(MUST BE >23T AFTER CH2 LOADED	
870		02AB	00					NOP		(CH2 SHOULD INT AFTER THIS)	
871		02AC	00					NOP		(FOR NMI PROTECTION)	
872		02AD	F3					DI		ALL CHS SHOULD HAVE INT BY NOW	
873		02AE	3E	F0				LD	A,0FOH	SEE IF ALL 4 CHS INT IN ORDER	
874		02B0	B8					CP	B		
875		02B1	C2	B0	04		C	JP	NZ ST_ERR		
876		02B4	18	40				JR	XCTCOR		

```

879
880
881          : . . . INTERRUPT ROUTINES FOR XCTC ALG 2
882 02B6 22 34 C0      XCTCI2: LD   (CTC_V1+4),HL  POINT INT VEC TO ERROR
883 02B9 D3 E2          OUT  (CTC_1_C2),A   RESET CHANNEL
884 02BB CB C0          SET   0,B           ADD TO CHECKSUM
885 02BD 11 C6 02      LD   DE,XCTC10    SET INT TRANSFER VEC
886 02C0 ED S3 30 C0      LD   (CTC_V1),DE  OF NEXT CHANNEL
887 02C4 18 27          JR   XCTCIC
888
889 02C6 22 30 C0      XCTCIO: LD   (CTC_V1),HL  EACH OF THESE INTERRUPT ROUTINES
890 02C9 D3 E0          OUT  (CTC_I_C0),A   SETS THE ADDRESS OF THE NEXT
891 02CB CB D0          SET   2,B           ISR EXPECTED INTO ITS PROPER
892 02CD 11 D6 02      LD   DE,XCTC11    RAM LOCATION, REPLACING THE
893 02D0 ED S3 32 C0      LD   (CTC_V1+2),DE EXISTING JP CTC_ERR0 INSTRUCTION.
894 02D4 18 17          JR   XCTCIC
895
896 02D6 22 32 C0      XCTCI1: LD   (CTC_V1+2),HL
897 02D9 D3 E1          OUT  (CTC_1_C1),A
898 02DB CB E0          SET   4,B
899 02DD 11 E6 02      LD   DE,XCTC13
900 02E0 ED S3 36 C0      LD   (CTC_V1+6),DE
901 02E4 18 07          JR   XCTCIC
902
903 02E6 22 36 C0      XCTCI3: LD   (CTC_V1+6),HL
904 02E9 D3 E3          OUT  (CTC_1_C3),A
905 02EB CB F0          SET   6,B
906 02ED CB 20          XCTCIC: SLA B
907 02EF FB             EI
908 02F0 ED 4D          RETI
909 ; . . .
911
912          : . . . CTC #0 ERROR ENTRY

```

```

922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
*****  

      Async SIO Test  

      Performs a basic functional test of channels A & B  

      of the SIO to asynchronously transmit 8 bytes  

      of data at 19.2 KBAUD.  

      The test uses the 'poll mode' capability of  

      the SIO, not the 'interrupt mode'.  

      If there is a stuck-at-0 condition in the 'RD'  

      lines or if there is a generally malfunctioning  

      SIO, the test could loop forever waiting for 'Tx  

      Buffer Available' or 'RX character available'. A  

      deadman timer is used to both detect this condition  

      and to detect the case where the SIO is transmitting,  

      but at a rate much slower than the configured 9600 BAUD.  

      This test also verifies that the status signals RD  

      will toggle between 0 and 1.  

*****

```

```

944 02F6 0E 00      LD  C,CTC_0_C0    RESET ALL CTC & PROGRAM FOR 19.2K BAUD
945 02F8 06 04      LD  B,4
946 02FA
MSIO_20:          LD  A,57H
947 02FA 3E 57      OUT (C),A
948 02FC ED 79      LD  A,6
949 02FE 3E 06      OUT (C),A
950 0300 ED 79      INC C
951 0302 OC         DJNZ MSIO_20
952 0303 10 F5

954 0305 0E E0      LD  C,CTC_1_C0
955
956 ; Bug fix 7/27/88 - Change "LD B,4" to "LD B,2"
957
958 0307 06 02      LD  B,2
959 0309
MSIO_30:          LD  A,57H
960 0309 3E 57      OUT (C),A
961 030B ED 79      LD  A,6
962 030D 3E 06      OUT (C),A
963 030F ED 79      INC C
964 0311 OC         INC C
965 0312 10 F5      DJNZ MSIO_30

967 0314 21 48 00    C      LD  HL,CTCIT1    SET UP INTERRUPT STUFF

```

```

968 0317 7C          LD  A,H        FOR DEADMAN TIMER ON CTC #1 CHANNEL #2
969 0318 ED 47          LD  I,A
970 031A 7D          LD  R,L
971 031B D3 E0          OUT (CTC_1_C0),A  THE INTERRUPT VECTOR GOES TO CH #0

973 031D 0E 71          LD  C,SIO_0_AC  PROGRAM ALL SIO CHANNELS
974 031F 1E 02          LD  E,2
975 0321 21 67 04    C   LD  HL,ROM_SIO
976 0324
MSIO_120:          LD  B,SIO_SIZ
977 0324 06 07          OTIR
978 0326 ED B3          INC C        INCREMENT TO NEXT PORT
979 0328 OC
980 0329 OC
981 032A 1D
982 032B 20 F7          INC C
                               DEC E        FINISH YET?
                               JR NZ,MSIO_120  NO

984 032D 0E B1          LD  C,SIO_1_AC  PROGRAM SIO-1 CHANNELS A AND B
985 032F 1E 02          LD  E,2
986 0331 21 95 04    C   LD  HL,ROM_SIO+14
987 0334
MSIO_170:          LD  B,SIO_SIZ
988 0334 06 07          OTIR
989 0336 ED B3          INC C
990 0338 OC
991 0339 OC
992 033A 1D
993 033B 20 F7          INC C
                               DEC E
                               JR NZ,MSIO_170  NO

994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
      . . . HOOD DETECTION
      THERE CAN BE A HOOD ON ANY OF THE FOUR CHANNELS. FOR CHANNELS
      1-3, A HOOD CAN BE DETECTED BY HAVING CERTAIN INPUT LINES SET.
      AS CHANNEL 0 IS A MODEM PORT, THE HOOD DETECTION TEST IS
      DEFFERENT. THE FOLLOWING IS AN EXPLANATION OF HOW HOODS ARE
      DETECTED ON EACH PORT.
      CHANNEL 0 (MODEM PORT) - THE DTR LINE WILL BE RESET, THEN A
      LOOPBACK TEST WILL BE PERFORMED ON MODEM LINES IC & SR.
      HOOD DETECTION WILL BE ACHIEVED BY WIGGLING SIO #0 CHANNEL
      B DCD LINE USING A PREDEFINED PATTERN. CHECK SIO #0
      CHANNEL B RTS LINE TO SEE IF IT WIGGLES IN THE SAME WAY.
      IF YES, A HOOD IS THERE.
      CHANNEL 1 - IF THE CTS LINE ON SIO-0-B IS SET, A HOOD IS
      CONNECTED.
      CHANNEL 2 - IF THE CTS LINE ON SIO-1-A IS SET, A HOOD IS
      CONNECTED.
      CHANNEL 3 - IF THE CTS LINE ON SIO-1-B IS SET, A HOOD IS
      CONNECTED.
      E' WILL CONTAIN A CODE REPRESENTING WHICH CHANNELS HAVE
      HOODS ATTACHED AND WHICH DON'T. BITS 0-4 IN E' REPRESENT
      CHANNELS 0-4. A '1' IN ONE OF THESE BIT POSITIONS MEANS THAT A

```

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 26

1019 ; : HOOD IS ATTACHED TO THE CHANNELS REPRESENTED BY THAT BIT.
 1020
 1022 033D 3E 10 LD A,10H RESET SIO CHANNELS - EXT. STAT. INT.
 1023 033F D3 71 OUT (\$SIO_0_AC),A
 1024 0341 D3 73 OUT (\$SIO_0_BC),A
 1025 0343 D3 B1 OUT (\$SIO_1_AC),A
 1026 0345 D3 B3 OUT (\$SIO_1_BC),A
 1028 ; : BEGIN HOOD DETECTION CODE FOR PORT 0 - MODEM PORT
 1029 ;
 1030 ; LD B,B WIGGLING SIGNALS 8 TIMES
 1031 0347 06 08 LD D,E,04D35H SEND PATTERN 0100 1101
 1032 0349 11 35 4D ; RECEIVE PATTERN 00110101
 1033 ; LD HL,0208H SEND MASK 0000 0010 - MASK RTS BIT
 1034 034C 21 08 02 ; RECEIVE MASK 0000 1000 - MASK DCD BIT
 1035 ; : H00D0:
 1036 034F LD A,15H SET REG. POINTER TO 5
 1037 034F 3E 15 OUT (\$SIO_0_BC),A
 1038 0351 D3 73 LD A,D OUTPUT THE RTS PATTERN
 1039 0353 7A AND H
 1040 0354 A4 OUT (\$SIO_0_BC),A
 1041 0355 D3 73 LD A,E DETERMINE THE INPUT PATTERN
 1042 0357 7B AND L
 1043 0358 A5 LD C,A SAVE THE PATTERN IN REG. C
 1044 0359 4F LD A,10H RESET THE EXTERNAL STATUS
 1045 035A 3E 10 OUT (\$SIO_0_BC),A
 1046 035C D3 73 IN A,(\$SIO_0_BC) READ THE INPUT PATTERN
 1047 035E DB 73 AND L
 1048 0360 A5 CP C SIGNAL MATCHES?
 1049 0361 B9 JR NZ,H00D1 NO
 1050 0362 20 0B RLC D MOVE TO NEXT PATTERN
 1051 0364 CB 02 RLC E
 1052 0366 CB 03
 1053 0368
 1054 0368 10 E5 DJNZ H00D0
 1055 036A 06 01 LD B,1 SET BIT 0 IN B REG.- HOOD ON PORT #0
 1056 036C C3 71 03 C JP H00D2
 1058 036F 06 00 H00D1: LD B,0 CLEAR B REG. IF NO HOOD ON PORT #0
 1060 ; :Reconfigure WR5 on SIO-0 channel B
 1062 0371 3E 15 H00D2: LD A,15H
 1063 0373 D3 73 OUT (\$SIO_0_BC),A
 1064 0375 3E 68 LD A,68H WR3,8BIT TX,TX ENABLE
 1065 0377 D3 73 OUT (\$SIO_0_BC),A

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 27

1067 ; : BEGIN HOOD TESTS ON PORTS 1-3
 1068 ;
 1069 ;
 1070 0379 DB 73 IN A,(\$SIO_0_BC)
 1071 037B CB 6F BIT S,A
 1072 037D CA 82 03 C JP Z,H00D3
 1073 0380 CB C8 SET 1,B SET BIT 1 IN B REG IF HOOD ON PORT #1
 1075 0382 DB B1 H00D3: IN A,(\$SIO_1_AC)
 1076 0384 CB 6F BIT S,A
 1077 0386 CA 88 03 C JP Z,H00D4
 1078 0389 CB D0 SET 2,B SET BIT 2 IN B REG IF HOOD ON PORT #2
 1080 038B DB B3 H00D4: IN A,(\$SIO_1_BC)
 1081 038D CB 6F BIT S,A
 1082 038F CA 94 03 C JP Z,H00D5
 1083 0392 CB D8 SET 3,B SET BIT 3 IN B REG IF HOOD ON PORT #3
 1085 0394 78 H00D5: LD A,B
 1086 0395 D9 EXX
 1087 0396 5F LD E,A STORE THE HOODS-DETECTED PATTERN IN E'
 1088 0397 D9 EXX
 1090 ; : THE FIRST RUN THROUGH THE FOLLOWING LOOP IS DONE WITH INTERNAL
 1091 LOOPBACK ENABLED, THE SECOND WITH INTERNAL LOOPBACK DISABLED AND
 1092 DRIVERS ENABLED.
 1093 ;
 1094 ;
 1095 0398 DD 23 INC IX
 1096 039A OE 71 LD C,\$IO_0_AC
 1097 039C CD 58 04 C CALL \$IO_TEST
 1098 039F DD 23 INC IX
 1099 03A1 OE 73 LD C,\$IO_0_BC
 1100 03A3 CD 58 04 C CALL \$IO_TEST
 1101 03A6 DD 23 INC IX
 1102 03A8 OE B1 LD C,\$IO_1_AC
 1103 03A9 CD 58 04 C CALL \$IO_TEST
 1104 03AD DD 23 INC IX
 1105 03AF OE B3 LD C,\$IO_1_BC
 1106 03B1 CD 58 04 C CALL \$IO_TEST
 1108 03B4 D9 EXX
 1109 03B5 7B LD A,E RETRIEVE HOOD DETECT BITS IN E REG
 1110 03B6 D9 EXX
 1111 03B7 5F LD E,A USING E AS TEMP STORAGE OF ORIG. PATTERN
 1112 03B8 B7 OR A
 1113 03B9 20 03 JR NZ,MS10_200
 1114 03BB C3 A3 04 C JP \$IO_DONE
 1115 ;

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 28

```

1116      ; EXTERNAL LOOPBACK THRU TEST HOODS
1117      ;
1119 03BE    MSIO_200:
1120 03BE 3E 15   LD A,15H
1121 03C0 D3 B3   OUT (SIO_1_BC),A
1122 03C2 3E 68   LD A,01TO000B
1123 03C4 D3 B3   OUT (SIO_1_BC),A   TURN OFF BIT FOR INTERNAL LOOPBACK (RTSB)

1125      ; EXTERNAL LOOPBACK FOR PORT #0 (MODEM PORT)
1126      ;
1127      Will be looping data through the following line combinations:
1128      1. TX & RX
1129      2. RS & CS
1130      3. TR & DM
1131      4. SR & RR
1132      note: IC & SR are also connected to each other in the test hood.
1133      However, they will not be tested here as they were used to detect
1134      the presence of the hood in the first place. They were effectively
1135      tested in the hood detection routine.

1138 02C6 DD 23   INC IX
1139 03C8 7B       LD A,E
1140 03C9 E6 01   AND I
1141 03CB CA F5 03  JP Z,PORT2   RETRIEVE ORIGINAL HOOD DETECT. PATTERN
                           MASK OFF BIT 0
                           IS THERE A HOOD ON PORT #0? JUMP IF NO

1143      ;test TX & RX lines
1144 03CE 0E 71   LD C,SIO_0_AC
1145 03D0 CD 58 04  C CALL SIO_TEST   OUTPUT PORT

1147      ;test RS & CS lines
1148 03D3 16 71   LD D,SIO_0_AC
1149 03D5 5A       LD E,D
1150 03D6 2E 02   LD L,2
1151 03D8 26 20   LD H,20H
1152 03D9 CD 2D 04  C CALL LOOP_TEST   INPUT PORT
                           TO INDICATE RTS BIT IN WRS
                           TO INDICATE CTS BIT IN RRO

1154      ;test TR & DM lines
1155 03DD 2E 80   LD L,80H
1156 03DF 26 10   LD H,10H
1157 03E1 CD 2D 04  C CALL LOOP_TEST   TO INDICATE DTR BIT IN WRS
                           TO INDICATE SYNC BIT IN RRO

1159      ;test SR & RR lines
1160 03E4 16 73   LD D,SIO_0_BC
1161 03E6 2E 02   LD L,2
1162 03E8 26 08   LD H,6   TO INDICATE RTS BIT IN WRS
                           TO INDICATE DCD BIT IN WRO

```

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 29

```

1163 03EA CD 2D 04  C CALL LOOP_TEST
1165      ;Reconfigure WRS in SIO-0 channel a
1167 03ED 3E 15   LD A,15H
1168 03EF D3 71   OUT (SIO_0_AC),A
1169 03F1 3E 68   LD A,68H
1170 03F3 D3 71   OUT (SIO_0_AC),A   WRS:8 BIT TX, TX ENABLE

1172      ; EXTERNAL LOOPBACK FOR PORTS 1-3
1173      ;
1174 03F5    PORT2:
1175 03F5 DD 23   INC IX
1176 03F7 D9     EXX
1177 03F8 7B     LD A,E
1178 03F9 D9     EXX
1179 03FA E6 02   RETRIEVE ORIGINAL PATTERN
1180 03FC CR 0C 04  C AND 2H
1181 03FF 0E 73   JP Z,PORT3
1182 0401 3E 15   LD C,SIO_0_BC
1183 0403 ED 79   LD A,15H
1184 0405 3E 68   RESET WRITE REG 5 TO CORRECT BIT PATTERN
1185 0407 ED 79   OUT (C),A
1186 0409 CD 58 04  C LD A,68H
1187 040C          OUT (C),A
1188 040C DD 23   CALL SIO_TEST
1189 040E D9     PORT3:
1190 040F 7B     INC IX
1191 0410 D9     EXX
1192 0411 E6 04   LD A,E
1193 0413 CA 1B 04  C AND 4H
1194 0416 0E B1   JP Z,PORT4
1195 0418 CD 58 04  C LD C,SIO_1_AC
1196 041B          CALL SIO_TEST
1197 041B DD 23   PORT4:
1198 041D D9     INC IX
1199 041E 7B     EXX
1200 041F D9     LD A,E
1201 0420 E6 08   RETRIEVE ORIGINAL PATTERN
1202 0422 CR A3 04  C AND 8H
1203 0425 0E B3   JP Z,SIO_DONE
1204 0427 CD 58 04  C LD C,SIO_1_BC
1205 042A C3 A3 04  C CALL SIO_TEST
                           IS THERE A HOOD ON PORT #3? JUMP IF NO
                           JP SIO_DONE

1207      ****
1208      SUBROUTINE NAME: LOOP_TEST
1209      ;
1210      This subroutine is used in the Hood detect section and the modem
1211      external loopback portion of Self Test.

```

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 30

1212
 1213 Upon Entry: D register contains the SIO address of the output line
 1214 E register contains the SIO address of the input line
 1215 L register contains the bit position to set in Write
 1216 H register contains the mask to isolate the input line
 1217 in RRO in the SIO.
 1218
 1219
 1220 Update 8/10/88 Delete reference to deadman timer.
 1221 PROCEDURE: A deadman timer is first set (approx. 8 millisecs.)
 1222 in the event that the loopback fails. The appropriate output
 1223 line in the SIO is then set (either RTS or DTR). The
 1224 corresponding input bit is then checked to see if it is also set.
 1225 The same procedure is then followed again with the output bit
 1226 reset and a check that the looped back input bit is also reset.
 1227 The whole sequence is repeated 4 times.
 1228
 1229 This routine assumes that the SIO channel for the input line and
 1230 the SIO channel for the output line are the same.
 1231 ****
 1233 042D LOOP_TEST:
 1235
 1236 ****
 1237 Update 8/10/88 by Randy Stout - RM0 Production Engineering
 1238 Since a deadman timer is not necessary here, these lines have
 1239 been deleted.
 1240 LD A,0B7H SET DEADMAN TIMER-PRESC.=256
 1241 OUT (CTC_1_C2),A
 1242 XOR A
 1243 OUT (CTC_1_C2),A
 1244 EI ENABLE INTERRUPTS
 1245 ****
 1247 042D 06 04 LD B,4 LOOP COUNTER=4
 1249 042F 3E 15 MLOOP1: LD A,15H RESET SIO, POINT TO WWS
 1250 0431 4A OUT (C),A OUTPUT CHANNEL TO C REG.
 1251 0432 ED 79 LD A,L
 1252 0434 7D OUT (C),A OUTPUT LINE DESIGNATED BY L REG.
 1253 0435 ED 79
 1255 0437 4B LD C,E INPUT CHANNEL TO C REG.
 1256 0438 3E 10 LD A,10H RESET EXT. STATUS
 1257 043A ED 79 OUT (C),A
 1258 043C ED 78 IN A,(C) READ RRO IN THE SIO
 1259 043E A4 AND H ISOLATE THE INPUT LINE (CTS OR DCD)
 1260 043F CA B0 04 JP Z,ST_ERR LOOP UNTIL THE INPUT LINE IS ALSO SET

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 31

1262 0442 3E 15 LD A,15H CLEARING THE OUTPUT LINE THIS TIME
 1263 0444 4A LD C,D OUTPUT CHANNEL TO C REG.
 1264 0445 ED 79 OUT (C),A
 1265 0447 AF XOR A
 1266 0448 ED 79 OUT (C),A
 1267 0449 3E 10 LD A,10H
 1268 044C 4B LD C,E
 1269 044D ED 79 OUT (C),A
 1271 044F ED 78 IN A,(C)
 1272 0451 A4 AND H
 1273 0452 C2 B0 04 C JP NZ,ST_ERR
 1274 0455 10 D8 DJNZ MLOOP1 REPEAT ROUTINE UNTIL B REG. =0
 1275 ***** Deleted with deadman timer set up - 8/10/88
 1276 DI
 1277 *****
 1278 0457 C9 RET
 1280
 1281 *****
 1282 SUBROUTINE NAME: SIO_TEST
 1283 PURPOSE: TO SEND OUT 8 PATTERNS ON THE TX LINE FOR THE PORT, POLL THE
 1284 RX LINE UNTIL EACH PATTERN IS RECEIVED, THEN MATCH THE TWO TO
 1285 CORRECT OPERATION OF THE PORT. IF A TRANSMITTED PATTERN DOES NOT
 1286 MATCH THE RECEIVED PATTERN, A JUMP WILL BE MADE OUT OF THE
 1287 SUBROUTINE TO ST_ERR. THIS WILL BE ABNORMAL TERMINATION, NOT A
 1288 RET.
 1289 A DEADMAN TIMER WILL BE SET BEFORE THE TEST BEGINS SO THAT IF
 1290 NOTHING IS RECEIVED, THE TIMER WILL INDICATE ERROR.
 1291 RELEVANT REGISTER USAGE UPON ENTRY:
 1292 C - CONTAINS THE ADDRESS OF THE PORT TO BE TESTED
 1293 *****

1295 0458 16 AA
 1296 0458 16 AA
 1298 045A 06 08
 1300 045C 3E B7
 1301 045E D3 E2
 1302 0460 AF
 1303 0461 D3 E2
 1304 0463 FB

SIO_TEST:
 LD D,0RAH SET START TEST PATTERN
 LD B,8 NEED TO LOOP 8 TIMES PER PORT
 LD A,0B7H SET DEADMAN TIMER
 OUT (CTC_1_C2),A
 XOR A
 OUT (CTC_1_C2),A
 EI

1306 ;

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 32

```

1307          . . . POLL TX BUFFER EMPTY STATUS IN SIO READ REGISTER 0
1308          UNTIL THE TX BUFFER BECOMES AVAILABLE. THEN OUTPUT
1309          A CHARACTER. ALSO VERIFY THAT SIO RX CHAR AVAILABLE
1310          STATUS IS NOT SET.
1311          NEED TO MAKE SURE THAT DRIVER ENABLES ARE OFF AND
1312          INTERNAL LOOPBACK MUX IS SET TO LOOPBACK.
1313
1314 0464      MSIO_300:
1315 0464 ED 78  IN  A,(C)      READ SIO READ REGISTER 0
1316 0466 CB 57  BIT 2,A      TX BUFFER EMPTY?
1317 0468 28 FA  JR  Z,MSIO_300 NO
1318 046A CB 47  BIT 0,A      RX CHARA AVAIL SET?
1319 046C C2 B0 04  C   JP  NZ,ST_ERR YES ... ERROR
1320 046F OD     DEC C       DECREMENT TO DATA CHANNEL
1321 0470 ED 51  OUT (C),D   SEND A CHARACTER
1322 0472 OC     INC C       GO BACK TO CONTROL CHANNEL
1323
1324          . . . POLL SIO READ REGISTER 0 UNTIL RX CHAR AVAILABLE
1325          STATUS. INPUT THE CHARACTER AND COMPARE TO
1326          TRANSMITTED CHARACTER.
1327
1328 0473      MSIO_400:
1329 0473 ED 78  IN  A,(C)      READ SIO READ REGISTER 0
1330 0475 CB 47  BIT 0,A      RX CHAR AVAILABLE?
1331 0477 28 FA  JR  Z,MSIO_400 NO, GO POLL AGAIN
1332 0479 00     DEC C       DECREMENT TO THE DATA CHANNEL
1333 047A ED 78  IN  A,(C)      READ THE DATA BYTE
1334 047C OC     INC C       GO BACK TO CONTROL CHANNEL
1335 047D BA     CP  D       IS IT THE SAME AS THE TRANSMITTED CHAR
1336 047E C2 B0 04  C   JP  NZ,ST_ERR NO
1337 0481 2F     CPL          CHANGE THE TEST PATTERN
1338 0482 57     LD  D,A
1339 0483 10 DF   DJNZ MSIO_300 GO TRANSMIT ANOTHER CHARACTER
1340 0485 F3     DI
1341 0486 C9     RET
1343
1344          . . . SIO CONTROL WORDS
1345
1346 0487      ROM_SIO:
1347
1348          . . . PORT #0 -- SIO #0 CH A
1349
1350 0487 18    DEFB 18H
1351 0488 14    DEFB 14H
1352 0489 44    DEFB 44H      WR4, X16, 1 STOP, NO PARITY
1353 048A 13    DEFB 13H
1354 048B C1    DEFB 0C1H     WR3, 8BIT RX, RX ENABLE
1355 048C 15    DEFB 15H
1356 048D 68    DEFB 01101000B  WR5, 8BIT TX, TX ENABLE
1357 0007      SIO_SIZ EQU $-ROM_SIO

```

ERR LINE ADDR B1 B2 B3 B4

MUX ASYNCHRONOUS SIO TEST

PAGE 33

```

1358          . . . PORT #1 -- SIO #0 CH B
1359
1360
1361 048E 18    DEFB 18H
1362 048F 14    DEFB 14H
1363 0490 44    DEFB 44H
1364 0491 13    DEFB 13H
1365 0492 C1    DEFB 0C1H
1366 0493 15    DEFB 15H
1367 0494 68    DEFB 01101000B
1368
1369          . . . PORT #2 -- SIO #1 CH A
1370
1371 0495 18    DEFB 18H
1372 0496 14    DEFB 14H
1373 0497 44    DEFB 44H
1374 0498 13    DEFB 13H
1375 0499 C1    DEFB 0C1H
1376 049A 15    DEFB 15H
1377 049B 68    DEFB 08H
1378
1379          . . . PORT #3 -- SIO #1 CH B
1380
1381 049C 18    DEFB 18H
1382 049D 14    DEFB 14H
1383 049E 44    DEFB 44H
1384 049F 13    DEFB 13H
1385 04A0 C1    DEFB 0C1H
1386 04A1 15    DEFB 15H
1387 04A2 68    DEFB 08H      SET RTSB LINE - INTERNAL LOOPBACK MODE
1388 04A3
1390          SIO_DONE:           ;FINISHED SIO TEST
1392 04A3
1393 04A3

```

1396 04A3 SLF_PASS:
1397 ; UPON ENTRY HERE, SELF-TEST WAS EXECUTED AND PASSED. A
1398 ; VALUE WILL BE PLACED IN THE ST_COND REGISTER TO INDICATE SUCCESS
1399 ; AND AN INTERRUPT WILL BE SENT TO THE HOST. SELF TEST WILL THEN
1400 ; CALL INIT (CARD INITIALIZATION ROUTINE).
1401
1402 04A3 21 00 00 E LD HL,ST_COND ADDRESS OF ST_COND REGISTER TO HL
1403 04A6 36 E0 (HL),PASS GET VARIABLE INDICATING PASSED TEST
1404 04A8 21 00 C0 LD HL,INT_COND GET ADDRESS OF INT_COND REGISTER
1405 04AB 36 10 LD (HL),INT_CODE INTERRUPT HOST THAT SELF TEST DONE
1407 04AD C3 BF 04 C JP ST_DONE

1409 04B0 ST_ERR:
1410 ; Self-test failed.
1412 04B0 DD 22 00 00 E LD (ST_COND),IX SAVE ERROR NUMBER
1413 04B4 21 00 C0 LD HL,INT_COND
1414 04B7 36 10 LD (HL),INT_CODE SEND HOST ST DONE INTERRUPT
1416
1417 ; If the error was the deadman timer timing out, the daisy
1418 ; chain must be released by a return from interrupt. However,
1419 ; we don't want to return to the point the interrupt was called.
1420 ; Therefore the address of Init is being pushed before the RETI
; so that it is the address the PC gets after the RETI executes.
1422 04B9 01 00 00 E LD BC,INIT
1423 04BC C5 PUSH BC
1424 04BD ED 4D RETI

1426 04BF 00 ST_DONE: NOP
1427 04C0 END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

```

1      SOURCE: &MX4INIT
2      PROGRAMMER: LIZ POTEET
3
4      4 CHANNEL DIO MUX (FORDYCE) - INITIALIZATION CODE & MAIN IDLE LOOP
5
6      ****
7      * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
8      * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
9      * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
10     * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
11
12
13      LIST B
14      NAME INIT
15      COPY &MX4EQUS
16      LIST S
17
18      PUBLIC INIT,BD_TAB
19
20      EXTRN TX_1,EX_1,REC_1,RX_ERR1,TX_0,EX_0,REC_0,RX_ERR0
21      EXTRN TX_3,EX_3,REC_3,RX_ERR3,TX_2,EX_2,REC_2,RX_ERR2
22      EXTRN TMR_ISR,HSTINT,BITS_0,BITS_1,BITS_2,BITS_3
23      EXTRN CTC_ERR0,THEAD_0,THEAD_1,THEAD_2,THEAD_3,TTAIL_0
24      EXTRN TTAIL_1,TTAIL_2,TTAIL_3,CONFIG_0,CONFIG_1,CONFIG_2
25      EXTRN CONFIG_3,BD_0,BD_1,BD_2,BD_3
26      EXTRN WR3_0,WR4_0,WRS_0,WR3_1,WR4_1,WRS_1,WR3_2,WR4_2,WRS_2
27      EXTRN WR3_3,WR4_3,WRS_3,TMRFLG,M0M1_SUB,M0M3_SUB
28
29
30      INIT IS CALLED EITHER BY SELF TEST (TWICE - ONCE DURING THE
31      NM1 TEST AND ONCE WHEN SELF TEST HAS COMPLETED) OR DURING A
32      SOFT RESET OF THE CARD
33
34      THE INTERRUPT VECTORS ARE PLACED HERE BECAUSE THEY WILL BE PUT
35      IN HIGH ROM AND NEED TO BE ASEG'ED. THE ACTUAL INITIALIZATION CODE
36      BEGINS AFTER THESE VECTOR PLACEMENTS.
37      ASEG
38      ORG VEC      THIS IS 48 LOCATIONS BEFORE THE END OF ROM
39      (LAST LOCATION - FFF - IS THE CRC CODE)
40      ;VECTOR LOCATIONS FOR SIO #0 CHANNEL A & B
41
42      481 1FC0 00 00      E      DEFW TX_1
43      482 1FC2 00 00      E      DEFW EX_1
44      483 1FC4 00 00      E      DEFW REC_1
45      484 1FC6 00 00      E      DEFW RX_ERR1
46      485 1FC8 00 00      E      DEFW TX_0
47      486 1FC9 00 00      E      DEFW EX_0
48      487 1FCC 00 00      E      DEFW REC_0
49      488 1FCE 00 00      E      DEFW RX_ERR0

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

```

490      ;VECTOR LOCATIONS FOR SIO #1 CHANNEL A & B
491
492 1FD0 00 00      M      DEFW TX_3
493 1FD2 00 00      M      DEFW EX_3
494 1FD4 00 00      M      DEFW REC_3
495 1FD6 00 00      M      DEFW RX_ERR3
496 1FD8 00 00      M      DEFW TX_2
497 1FDA 00 00      M      DEFW EX_2
498 1FDC 00 00      M      DEFW REC_2
499 1FDE 00 00      M      DEFW RX_ERR2
500
501 1FE0 00 00      E      CTC0_VEC:
502 1FE0 00 00      E      DEFW CTC_ERR0
503 1FE2 00 00      E      DEFW CTC_ERR0
504 1FE4 00 00      E      DEFW HSTINT
505 1FE6 00 00      E      DEFW CTC_ERR0
506 1FE8 00 00      E      DEFS 8
507
508 1FF0 00 00      E      CTC1_VEC:
509 1FF0 00 00      E      DEFW CTC_ERR0
510 1FF2 00 00      E      DEFW CTC_ERR0
511 1FF4 00 00      E      DEFW TMR_ISR
512 1FF6 00 00      E      DEFW CTC_ERR0
513
514      ;BEGINNING FORMAL INITIALIZATION CODE
515
516      CSEG
517
518 0000          INIT:    DI
519 0000  F3          XOR A
520 0001  AF          LD (RESET),A      CLEAR THE RESET REGISTER
521 0002  32 00 80      LD A,(TEST)      RETRIEVE TEST VALUE
522 0005  3A 02 C0      LD A,(TEST)      WAS INIT CALLED IN NM1 ROUTINE OF ST?
523 0008  FE 34          CP SVRL
524 000A  C2 13 00      JP NZ,INIT1      IF NO, JUMP TO INIT1
525 000D  3E 43          LD A,EVAL      ELSE, PLACE RETURN VALUE IN TEST
526 000F  32 02 C0      LD (TEST),A
527 0012  C9          RET      ;RETURN FROM SUBROUTINE CALL
528
529
530      SET INTERRUPT MODE AND INITIALIZE STACK POINTER
531
532 0013          INIT1:   IM 2      RESET CLEARED INTERRUPT MODE
533 0013  ED SE

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 3

534 0015 31 00 C8 LD SP,STK_ADDR SET INITIAL STACK ADDRESS IN SP
 535
 536
 537 | RESET ALL SIO CHANNELS

539 0018 06 02 LD B,2
 540 001A 0E 71 LD C,SIO_0_AC
 541 001C 3E 18 LD A,18H
 542 001E INIT2: OUT (C),A
 543 001E ED 79 INC C
 544 0020 0C INC C
 545 0021 0C INC C
 546 0022 10 FA DJNZ INIT2

548 0024 06 02 LD B,2
 549 0026 0E B1 LD C,SIO_1_AC
 550 0028 INIT3: OUT (C),A
 551 0028 ED 79 INC C
 552 002A 0C INC C
 553 002B 0C INC C
 554 002C 10 FA DJNZ INIT3

556 | SET UP CTC'S - CTC 0 CHANNEL 0 = BAUD RATE GENERATOR FOR PORT 0
 557 | CHANNEL 1 = BAUD RATE GENERATOR FOR PORT 1
 558 | CHANNELS 2 & 3 UNUSED
 559 | CTC 1 CHANNEL 0 = BAUD RATE GENERATOR FOR PORT 2
 560 | CHANNEL 1 = BAUD RATE GENERATOR FOR PORT 3
 561 | CHANNEL 2 = TIME OUT TIMER (SET LATER)
 562 | CHANNEL 3 UNUSED
 563 | ALL BAUD RATE GENERATORS SET TO DEFAULT BAUD OF
 564 | 9600
 565 |
 566 |
 567 002E 3E 47 LD R,01000111B LOAD CHANNEL CONTROL WORD TO
 568 0030 D3 D0 OUT (CTC_0_C0),A BAUD RATE GENERATOR CHANNELS
 569 0032 D3 D1 OUT (CTC_0_C1),A
 570 0034 D3 E0 OUT (CTC_1_C0),A
 571 0036 D3 E1 OUT (CTC_1_C1),A

573 0038 3E 0C LD A,12 TIME CONSTANT VALUE FOR 9600 BD
 574 003A D3 D0 OUT (CTC_0_C0),A
 575 003C D3 D1 OUT (CTC_0_C1),A
 576 003E D3 E0 OUT (CTC_1_C0),A
 577 0040 D3 E1 OUT (CTC_1_C1),A

578 | CONFIGURE INTERRUPT VECTOR ADDRESSES IN THE SIO'S
 579 | (CTC #1 CHANNEL 2 WILL BE SET UP AT THE END OF THE CODE
 580 | WHEN THE TIME-OUT TIMER IS INITIALIZED).
 581 | THE I REGISTER WILL ALSO BE SET IN THE FOLLOWING CODE
 582 |
 583 |

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 4

584 0042 01 C0 1F LD BC,VEC RETRIEVE THE BEG. VECTOR ADDRESS
 585 0045 78 LD A,B
 586 0046 ED 47 LD I,A HIGH VECTOR ADDRESS TO I REGISTER

588 0048 3E 12 LD A,12H PROGRAM WR #2 IN SIO #0
 589 004A D3 73 OUT (SIO_0_BC),A
 590 004C 79 LD A,C
 591 004D D3 73 OUT (SIO_0_BC),A

593 004F 3E 12 LD A,12H PROGRAM WR #2 IN SIO #1
 594 0051 D3 B3 OUT (SIO_1_BC),A
 595 0053 79 LD A,C VECTOR ADDRESS TO A REG
 596 0054 C8 10 ADD R,10H GET TO NEXT SET OF VECTOR ADDRESSES
 597 0056 D3 B3 OUT (SIO_1_BC),A

599 |
 600 | PROGRAM ALL SIO CHANNELS (WR 2 IN CHANNEL B OF BOTH SIO'S IS
 601 | PREVIOUSLY SET)
 602 |
 603 0058 0E 71 LD C,SIO_0_AC PROGRAM ALL SIO CHANNELS
 604 005A 1E 02 LD E,2
 605 005C 21 78 00 C LD HL,SIO_TAB
 606 005F INIT4: LD B,TAB_SIZ
 607 005F 06 08 OTIR
 608 0061 ED B3 INC C
 609 0063 0C INC C
 610 0064 0C INC C
 611 0065 1D DEC E
 612 0066 20 F7 JR NZ,INIT4 FINISH YET?
 NO

614 0068 0E B1 LD C,SIO_1_AC PROGRAM SIO 1 CHANNELS A AND B
 615 006A 1E 02 LD E,2
 616 006C 21 88 00 C LD HL,SIO_TAB+16 INCR. PAST CHAN.0&1 DATA IN SIO_TAB

617 006F INIT5: LD B,TAB_SIZ
 618 006F 06 08 OTIR
 619 0071 ED B3 INC C
 620 0073 0C INC C
 621 0074 0C INC C
 622 0075 1D DEC E
 623 0076 20 F7 JR NZ,INIT5
 624 0078 C3 BD 00 C JP JMP

625 |
 626 | PUT IN THE ACTUAL SIO BYTES AND THE INTERRUPT VECTOR ADDRESSES
 627 | GOT THE PRECEDING STUFF FROM SELF TEST P.22. DONT FORGET TO
 628 | SET THE CTC CHANNEL 2 TIMER AT THE END OF THIS AND TO CLEAR OUT
 629 | RAM.
 630 |
 631 | SIO CONTROL WORDS
 632 |

```

633          ; . . . PORT #0 - SIO #0 CHANNEL A
634
635 007B    SIO_TAB:   DEFB 14H      POINTER TO WR#4
636 0078 14  DEFB 44H      X16 CLOCK, 1 STOP BIT
637 007C 44  DEFB 13H      POINTER TO WR#3
638 007D 13  DEFB OC1H     RX ENABLE, 8 BITS PER CHARACTER
639 007E C1  DEFB 15H      POINTER TO WR#5
640 007F 15  DEFB 68H      TX ENABLE, 8 BITS PER CHARACTER
641 0080 68  DEFB 11H      POINTER TO WR#1
642 0081 11  DEFB 13H      EXT INT EN, TX INT EN, INT ON RX CHAR
643 0082 13  TAB_SIZ EQU $-SIO_TAB
644 0008
645
646
647          ; . . . PORT #1 - SIO #0 CHANNEL B
648 0083 14  DEFB 14H      POINTER TO WR#4
649 0084 44  DEFB 44H      X16 CLOCK, 1 STOP BIT
650 0085 13  DEFB 13H      POINTER TO WR#3
651 0086 C1  DEFB OC1H     RX ENABLE, 8 BITS PER CHARACTER
652 0087 15  DEFB 15H      POINTER TO WR#5
653 0088 68  DEFB 68H      TX ENABLE, 8 BITS PER CHARACTER
654 0089 11  DEFB 11H      POINTER TO WR#1
655 008A 17  DEFB 17H      TX & EXT INT EN, VECTOR, INT ON RX
656
657
658          ; . . . PORT #2 - SIO #1 CHANNEL A
659 008B 14  DEFB 14H      POINTER TO WR#4
660 008C 44  DEFB 44H      X16 CLOCK, 1 STOP BIT
661 008D 13  DEFB 13H      POINTER TO WR#3
662 008E C1  DEFB OC1H     RX ENABLE, 8 BITS PER CHARACTER
663 008F 15  DEFB 15H      POINTER TO WR#5
664 0090 68  DEFB 68H      TX ENABLE, 8 BITS PER CHARACTER
665 0091 11  DEFB 11H      POINTER TO WR#1
666 0092 13  DEFB 13H      EXT & TX INT EN, INT ON RX CHAR
667
668
669          ; . . . PORT #3 - SIO #1 CHANNEL B
670 0093 14  DEFB 14H      POINTER TO WR#4
671 0094 44  DEFB 44H      X16 CLOCK, 1 STOP BIT
672 0095 13  DEFB 13H      POINTER TO WR#3
673 0096 C1  DEFB OC1H     RX ENABLE, 8 BITS PER CHARACTER
674 0097 15  DEFB 15H      POINTER TO WR#5
675 0098 68  DEFB 68H      TX ENABLE, 8 BITS PER CHARACTER
676 0099 11  DEFB 11H      POINTER TO WR#1
677 009A 17  DEFB 17H      EXT & TX INT EN, VECTOR, INT ON RX
678
679          ;BAUD RATE TABLE - THIS TABLE CONTAINS THE PRESCALE VALUE AND THE
680          ;CHANNEL CONTROL WORD FOR EACH OF THE 17 BAUD RATES SUPPORTED BY
681          ;THIS CARD. THE FIRST ENTRY FOR EACH BAUD RATE IS THE CONTROL WORD
682          ;AND THE SECOND THE TIME CONSTANT VALUE.

```

```

684 009B    BD_TAB:   DEFB 27H      BAUD RATE = 50
685 009B 27  DEFB 18      = 75
686 009C 12  DEFB 07H     = 110
687 009D 07  DEFB 192
688 009E C0  DEFB 07H
689 009F 07  DEFB 131
690 00A0 83  DEFB 07H
691 00A1 07  DEFB 107
692 00A2 68  DEFB 07H
693 00A3 07  DEFB 96
694 00A4 60  DEFB 07H
695 00A5 07  DEFB 48
696 00A6 30  DEFB 47H
697 00A7 47  DEFB 47H
698 00A8 C0  DEFB 192
699 00A9 47  DEFB 47H
700 00AA 80  DEFB 128
701 00AB 47  DEFB 47H
702 00AC 60  DEFB 96
703 00AD 47  DEFB 47H
704 00AE 40  DEFB 64
705 00AF 47  DEFB 47H
706 00B0 30  DEFB 48
707 00B1 47  DEFB 47H
708 00B2 20  DEFB 32
709 00B3 47  DEFB 47H
710 00B4 18  DEFB 24
711 00B5 47  DEFB 47H
712 00B6 10  DEFB 16
713 00B7 47  DEFB 47H
714 00B8 C0  DEFB 12
715 00B9 47  DEFB 47H
716 00B8 06  DEFB 6
717 00B8 47  DEFB 47H
718 00BC 03  DEFB 3

720          ;INITIALIZE RAM TO 0'S - REASON: BECAUSE SO MANY OF THE VARIABLES
721          ;PLUS THE BIT MAP NEED TO BE = TO ZERO. EXCEPTION: THE ST_COND
722          ;REGISTER IS NOT CLEARED BECAUSE IT CONTAINS THE VALUE OF
723          ;THE RESULT OF SELF TEST
724
725 00B0 3E 00  JMP: LD A,0
726 00BF 21 02 C0  LD HL, RAM_BEG
727 00C2 77  LD (HL), A      Clear beginning location in RAM
728 00C3 01 20 07  LD BC, ST_COND-RAM_BEG-1 Stop at ST_COND register
729 00C6 11 03 C0  LD DE, RAM_BEG+1
730 00C9 ED B0  LDIR
731 00CB 21 24 C7  LD HL, ST_COND+1
732 00CE 77  LD (HL), A      Clear location 1 past ST_COND
733 00CF 01 DB 00  LD BC, OC7FFH-ST_COND-1 Stop at end of RAM
734 00D2 11 25 C7  LD DE, ST_COND+2
735 00D5 ED B0  LDIR

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 7

738 ;INITIALIZE BITS MASKS (PARITY MASK) TO FF FOR EIGHT BITS PER CHARACTER

740 00D7 3E FF E LD A,0FFH
 741 00D9 32 00 00 E LD (BITS_0),A
 742 00DC 32 00 00 E LD (BITS_1),A
 743 00DF 32 00 00 E LD (BITS_2),A
 744 00E2 32 00 00 E LD (BITS_3),A

746 ;INITIALIZE THE SIO WRITE REGISTER VARIABLES - MATCH ACTUAL WR VALUES

748 00E5 3E C1 E LD A,0C1H
 749 00E7 32 00 00 E LD (UR3_0),A
 750 00E9 32 00 00 E LD (UR3_1),A
 751 00ED 32 00 00 E LD (UR3_2),A
 752 00F0 32 00 00 E LD (UR3_3),A
 753 00F3 3E 44 E LD A,44H
 754 00F5 32 00 00 E LD (UR4_0),A
 755 00F8 32 00 00 E LD (UR4_1),A
 756 00FB 32 00 00 E LD (UR4_2),A
 757 00FE 32 00 00 E LD (UR4_3),A
 758 0101 3E 68 E LD A,68H
 759 0103 32 00 00 E LD (UR5_0),A
 760 0106 32 00 00 E LD (UR5_1),A
 761 0109 32 00 00 E LD (UR5_2),A
 762 010C 32 00 00 E LD (UR5_3),A

764 ;INITIALIZE CONFIGURATION AND BAUD RATE REGISTERS (NO PARITY,1 STOP BIT
 765 ; 8 BITS PER CHAR., 9600 BAUD

767 010F 3E 30 E LD A,BEG_CONF INIT DATA FOR CONFIG. REG.
 768 0111 32 00 00 E LD (CONFG_0),A
 769 0114 32 00 00 E LD (CONFG_1),A
 770 0117 32 00 00 E LD (CONFG_2),A
 771 011A 32 00 00 E LD (CONFG_3),A

773 011D 3E 0F E LD A,BEG_BD INIT DATA FOR BAUD REGISTER
 774 011F 32 00 00 E LD (BD_0),A
 775 0122 32 00 00 E LD (BD_1),A
 776 0125 32 00 00 E LD (BD_2),A
 777 0128 32 00 00 E LD (BD_3),A

779 ;CHECK THE MODEM LINES AND SET MODEM TO REFLECT CURRENT STATUS
 780 ;
 781 ;

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 8

783 0128 0E 71
 784 012D ED 40
 785 012F CD 00 00 E IN B,(C) RETRIEVE CONTENTS OF READ REG. 0
 786 0132 0E 73 CALL MDH3_SUB CHECK STATUS OF CTS,RR,DM LINES
 787 0134 ED 40 IN B,(C)
 788 0136 CD 00 00 E CALL MDH1_SUB CHECK STATUS OF IC LINE

790 ;RELEASE SEMAPHORE REGISTER - LEAVE BIT 7=0
 791 ;
 792 ;
 793 0139 3E 80 LD A,80H WRITING A PATTERN TO SEM-REG
 794 013B 32 02 80 LD (SEM_REG),A

796 ;INITIALIZE CTC INTERRUPT VECTOR AND SET TIMER FLAG TO ZERO
 797 ;
 798 ;
 799 013E 3E F0 LD A,_LOW.CTC1_VEC RETRIEVE VECTOR ADDRESS FOR CTC
 800 0140 D3 E0 OUT (CTC_1_C0),A INTERRUPT VECTOR TO CH. #0
 801 0142 3E 23 LD A,00100011B DISABLE TIMER INTERRUPTS
 802 0144 D3 E2 OUT (CTC_1_C2),A
 803 0146 AF XOR A
 804 0147 32 00 00 E LD (TMRFLG),A INDICATES THAT THE TIMER IS OFF

806 ;INITIALIZE CTC-0 CH 2 FOR HOST INTERRUPTS - THE CHANNEL CONTROL WORD
 807 ; IS PROGRAMMED FOR: INTERRUPT, COUNTER OPER., NEG. EDGE TRIG.,
 808 ;
 810 0148 3E E0 LD A,_LOW.CTC0_VEC RETRIEVE VECTOR ADDRESS FOR CTC
 811 014C D3 D0 OUT (CTC_0_C0),A INTERRUPT VECTOR TO CH. #0
 812 014E 3E C7 LD A,11000111B PROGRAM CHANNEL CONTROL WORD
 813 0150 D3 D2 OUT (CTC_0_C2),A
 814 0152 3E 01 LD A,1 TIME CONSTANT VALUE
 815 0154 D3 D2 OUT (CTC_0_C2),A

817 0156 FB EI REINABLE INTERRUPTS

819 ;THIS IS THE MAINLINE IDLE LOOP THAT THE CARD OPERATES IN WHILE
 820 ;WAITING FOR AN INTERRUPT. THIS LOOP ONLY CYCLES FROM ADDRESSES
 821 ;C002 - C7FF AS THAT IS ALL THE RAM AVAILABLE ON THIS CARD

823 0157 21 02 C0 MAIN: LD HL,0C002H
 824 015A 7E MAIN2: LD A,(HL) This routine facilitates 1611 debugging
 825 015B 23 INC HL
 826 015C 3E C8 LD A,0C8H

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 9

827 015E BC
828 015F 20 F9
829 0161 C3 57 01 C
831 0164

CP H Has it reached C800H yet?
JR NZ MAIN2 No. Continue cycling
JP MAIN Yes. Reload the starting RAM address
END

ASSEMBLER ERRORS = 0

```

1      ;SOURCE: &MX_VAR
2      ;PROGRAMMER: LIZ POTEET
3      ;          4 CHANNEL DIO MUX (FORDYCE) - VARIABLES (DSEG)
4      ****
5      ;***** (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
6      ;***** RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
7      ;***** REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
8      ;***** THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
9      ;*****
10     ;
11     LIST 8
12     NAME MX_VAR
13     ;
14     PUBLIC TEST,TON0,TON1,TON2,TON3
15     PUBLIC RBRK_0,RBRK_1,RBRK_2,RBRK_3,TMPTAB
16     PUBLIC WR3_0,WR3_1,WR3_2,WR3_3,WR4_0,WR4_1,WR4_2,WR4_3
17     PUBLIC WRS_0,WRS_1,WRS_2,WRS_3,TMRFLG
18     PUBLIC BIT_MAP,RHEAD_0,RHEAD_1,RHEAD_2,RHEAD_3,RTAIL_0
19     PUBLIC RTAIL_1,RTAIL_2,RTAIL_3,THEAD_0,THEAD_1,THEAD_2
20     PUBLIC THEAD_3,TTAIL_0,TTAIL_1,TTAIL_2,TTAIL_3,STAT_0
21     PUBLIC STAT_1,STAT_2,STAT_3,CONF0,CONF1,CONF2
22     PUBLIC CONF3,BD_0,BD_1,BD_2,BD_3,MODM_IN,MODM_OUT
23     PUBLIC MODM_MASK,CMD_TAB,ICR_TAB,ST_COND
24     PUBLIC BITS_0,BITS_1,BITS_2,BITS_3
25     ;
26     ;
27     DSEG
28     ;
29     ;CARD VARIABLES - OCCUPY RAM ADDRESSES C002 - C1FF
30
31     TEST      DEFS 1           General purpose - Used in Self Test
32     0000      TEST      DEFS 1           TX ISR: 1=Transmitter on; 0=transm. off
33     0001      TON0      DEFS 1           :
34     0002      TON1      DEFS 1           Port 1
35     0003      TON2      DEFS 1           :
36     0004      TON3      DEFS 1           Port 2
37     0005      TON3      DEFS 1           :
38     0006      STAT_0    DEFS 1           Port 3
39     0007      STAT_1    DEFS 1           RX ISR's - bit pattern for status register
40     0008      STAT_2    DEFS 1           Port 1
41     0009      STAT_3    DEFS 1           Port 2
42     000A      RBRK_0   DEFS 1           Port 3
43     000B      RBRK_1   DEFS 1           EXT/STAT ISR's - End of break flag - Pt 0
44     000C      RBRK_2   DEFS 1           (1=START OF BREAK DETECTED) - Pt 1
45     000D      RBRK_3   DEFS 1           - Pt 2
46     000E      RBRK_4   DEFS 1           - Pt 3
47     000F      WR3_0    DEFS 1           SIO WRITE REGISTER VALUES
48     0000      WR3_0    DEFS 1           :
49     0001      WR4_0    DEFS 1           :
50     0002      WRS_0    DEFS 1           :

```

```

52     0010      WR3_1    DEFS 1           :
53     0011      WR4_1    DEFS 1           :
54     0012      WRS_1    DEFS 1           :
55
56     0013      WR3_2    DEFS 1           :
57     0014      WR4_2    DEFS 1           :
58     0015      WRS_2    DEFS 1           :
59
60     0016      WR3_3    DEFS 1           :
61     0017      WR4_3    DEFS 1           :
62     0018      WRS_3    DEFS 1           :
63
64     0019      TMRFLG   DEFS 1           Flag which indicates if timer is off or on
65
66     001A      TMPTAB   DEFS 4           Temporary Table for CMND_TAB data
67
68     001E      BITS_0   DEFS 1           Masks off parity bits on RX chars - port 0
69     001F      BITS_1   DEFS 1           Masks off parity bits on RX chars - port 1
70     0020      BITS_2   DEFS 1           Masks off parity bits on RX chars - port 2
71     0021      BITS_3   DEFS 1           Masks off parity bits on RX chars - port 3
72
73     ;SHARED VARIABLES - OCCUPY RAM ADDRESSES C700 - C76FH
74
75     ASEG
76     ORG 0C600H
77
78     C600      BIT_MAP  DEFS 256
79
80     C700      RHEAD_0  DEFS 1
81     C701      RHEAD_1  DEFS 1
82     C702      RHEAD_2  DEFS 1
83     C703      RHEAD_3  DEFS 1
84
85     C704      RTAIL_0  DEFS 1
86     C705      RTAIL_1  DEFS 1
87     C706      RTAIL_2  DEFS 1
88     C707      RTAIL_3  DEFS 1
89
90     C708      THEAD_0  DEFS 1
91     C709      THEAD_1  DEFS 1
92     C70A      THEAD_2  DEFS 1

```

```
93 C70B      THEAD_3  DEFS 1
95 C70C      TTAIL_0  DEFS 1
96 C70D      TTAIL_1  DEFS 1
97 C70E      TTAIL_2  DEFS 1
98 C70F      TTAIL_3  DEFS 1
100 C710     CONFG_0  DEFS 1
101 C711     BD_0    DEFS 1
103 C712     CONFG_1  DEFS 1
104 C713     BD_1    DEFS 1
106 C714     CONFG_2  DEFS 1
107 C715     BD_2    DEFS 1
109 C716     CONFG_3  DEFS 1
110 C717     BD_3    DEFS 1
112 C718     MODM_IN  DEFS 1
113 C719     MODM_OUT DEFS 1
114 C71A     MODM_MASK DEFS 1
115 C71B     CMND_TAB DEFS 4
116 C71F     ICR_TAB  DEFS 4
117 C723     ST_COND  DEFS 1
119 C724     END
```

ASSEMBLER ERRORS = 0

```

1 SOURCE: &MX4RX
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - RECEIVE ISR'S FOR PORTS 0 THRU 3
5 ****
6 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
7 * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
8 * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
9 * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
10 ****
11
12 LIST B
13 NAME RX_ISR
14 COPY &MX4EQU
15 LIST S
16
17 PUBLIC REC_0,REC_1,REC_2,REC_3
18
19 EXTRN STAT_0,STAT_1,STAT_2,STAT_3,RHEAD_0,RHEAD_1,RHEAD_2
20 EXTRN RHEAD_3,RTAIL_0,RTAIL_1,RTAIL_2,RTAIL_3,BIT_MAP
21 EXTRN ICR_TAB,BITS_0,BITS_1,BITS_2,BITS_3
22
23
24 CSEG
25
26 **** DESCRIPTION: This file contains the Receive Interrupt Service
27 Routines which are invoked when the UART has received a character
28 at one of the four ports. The entry points for the four routines
29 are: REC_0, REC_1, REC_2, and REC_3.
30 All four routines are basically the same and call the macro,
31 RECISR. A description of this macro may be found in the file
32 &MX4EQU.
33 ****
34
35 ; RECEIVE ISR FOR PORT #0
36
37 477 0000 D9 REC_0 EXX CAUTION! IF THIS ISR CAN BE INTERRUPTED
38 0001 08 EX AF,AF USE PUSH & POP OR MAY LOSE REG. CONTENTS
39 0002 00 RECISR RHEAD_0,RTAIL_0,SIO_0_AD,PORT0,RX_BASE0,STAT_0,BITS_0
40
41 +; CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
42 +; IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
43 +
44 484 0002 3A 00 00 E + LD A,(RHEAD_0) Retrieve value of head pointer
45 0005 47 + LD B,A
46 0006 3A 00 00 E + LD A,(RTAIL_0) Retrieve value of tail pointer

```

```

487 0009 3C + INC A
488 000A 3C + INC A Add 2 to tail pointer
489 000B B8 + CP B Compare the two pointers
490 000C 20 0A + JR NZ,??0001 If pointers are not equal, jump
491 000E DB 70 + IN A,(SIO_0_AD) Retrieve character and discard
492 0010 21 00 00 E + LD HL,STAT_0 Get address of Status byte
493 0013 CB DE + SET 3,(HL) Set bit 3 in the Status byte
494 0015 C3 58 00 C + JP ??0004 Jump to exit routine
495
496 +; RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
497 FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
498
499 0018 DB 70 ??0001: IN A,(SIO_0_AD) RETRIEVE CHARACTER FROM UAR
500 001A 21 00 00 E + LD HL,BITS_0 Mask off parity bits
501 001D A6 + AND (HL)
502 001E 4F + LD C,A
503 001F 06 00 + LD B,0 Obtaining a 2-reg. quantity
504 0021 21 00 00 E + LD HL,BIT_MAP Get address of Bit Map base
505 0024 09 + ADD HL,BC Bit Map Base+char.=effective address
506 0025 CB 46 + BIT PORT0,(HL) Is the bit for the port = 1?
507 0027 CA 43 00 C + JP Z,??0002 Jump if not a special character
508
509 +; BIT MAP POSITION SET - CHECK SEMAPHORE REG. BEFORE SENDING INTERRUPT
510
511 002A 3A 02 80 ??0003: LD A,(SEM_REG) Retrieve address of semaphore register
512 002D CB 7F + BIT 7,A If bit 7=0, continue, else check again
513 002F C2 2A 00 C + JP NZ,??0003
514
515 +; SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEM_REG
516
517 0032 21 00 C0 + LD HL,INT_COND
518 0035 CB C6 + SET PORT0,(HL) Retrieve address of Int_Cond register
519 0037 21 00 00 E + LD HL,ICR_TAB Set bit for port specific interrupt
520 003A 11 00 00 + LD DE,PORT0 Retrieve address of ICR TAB base
521 003D 19 + ADD HL,DE Get index into ICR tab - port specific
522 003E CB CE + SET SPEC_ICR,(HL) index + ICR TAB base = effective address
523 0040 32 02 80 + LD (SEM_REG),A Set correct bit in ICR TAB
524
525 +; PUT CHARACTER IN FIFO AND UPDATE THE TAIL POINTER
526
527 0043 26 C5 ??0002: LD H,RX_BASE0 Get upper byte of FIFO base
528 0045 3A 00 00 E + LD A,(RTAIL_0) Get the value of the tail index
529 0048 6F + LD L,A Lower byte of tail pointer to 1 register
530 0049 71 + LD (HL),C Put character into buffer
531 004A 23 + INC HL Increment address for status byte
532 004B 3A 00 00 E + LD A,(STAT_0) Retrieve status byte
533 004E 77 + LD (HL),A Put status byte into buffer
534 004F FF + XOR A
535 0050 32 00 00 E + LD (STAT_0),A Clear status register
536 0053 7D + LD A,L Put tail pointer into A reg.
537 0054 3C + INC A Update pointer for next char.
538 0055 32 00 00 E + LD (RTAIL_0),A Update tail pointer index
539
540 0058 D9 ??0004 EXX

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

541 0059 08 + EX AF,AF'
 542 005A FB + EI
 543 005B ED 4D + RETI

545 ; RECEIVE ISR FOR PORT #1

547 005D D9 REC_1 EXX
 548 005E 08 EX AF,AF'
 549 005F RECISR RHEAD_1,RTAIL_1,SIO_0_BD,PORT1,RX_BASE1,STAT_1,BITS_1

550 ;
 551 ;CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
 552 ;IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
 553 ;
 554 005F 3A 00 00 E + LD A,(RHEAD_1) Retrieve value of head pointer
 555 0062 47 + LD B,A
 556 0063 3A 00 00 E + LD A,(RTAIL_1) Retrieve value of tail pointer
 557 0066 3C + INC A
 558 0067 3C + INC A Add 2 to tail pointer
 559 0068 B8 + CP B Compare the two pointers
 560 0069 20 0A + JR NZ,??0005 If pointers are not equal, jump
 561 006B DB 72 + IN A,(SIO_0_BD) Retrieve character and discard
 562 006D 21 00 00 E + LD HL,STAT_1 Get address of Status byte
 563 0070 CB DE + SET 3,(HL) Set bit 3 in the Status byte
 564 0072 C3 B5 00 C + JP ??0008 Jump to exit routine

565 ;
 566 ;RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
 567 ;FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
 568 ;
 569 0075 DB 72 REC_0005: IN A,(SIO_0_BD) RETRIEVE CHARACTER FROM UART
 570 0077 21 00 00 E + LD HL,BITS_1
 571 007A A6 AND (HL) Mask off parity bits
 572 007B 4F + LD C,A
 573 007C 06 00 + LD B'0 Obtaining a 2-reg. quantity
 574 007E 21 00 00 E + LD HL,BIT_MAP Get address of Bit Map base
 575 0081 09 + ADD HL,BC Bit Map Base+char.effective address
 576 0082 CB 4E + BIT PORT1,(HL) Is the bit for the port = 1?
 577 0084 CA A0 00 C + JP Z,??0006 Jump if not a special character

578 ;
 579 ;BIT MAP POSITION SET - CHECK SEMAPHORE REG. BEFORE SENDING INTERRUPT
 580 ;
 581 0087 3A 02 80 REC_0007: LD A,(SEM_REG) Retrieve address of semaphore register
 582 008A CB 7F + BIT 7,A If bit 7=0, continue, else check again
 583 008C C2 87 00 C + JP NZ,??0007

584 ;
 585 ;SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEM_REG
 586 ;
 587 008F 21 00 C0 + LD HL,INT_COND Retrieve address of Int_Cond register
 588 0092 CB CE + SET PORT1,(HL) Set bit for port specific interrupt
 589 0094 21 00 00 E + LD HL,ICR_TAB Retrieve address of ICR TAB base
 590 0097 11 01 00 C + LD DE,PORT1 Get index into ICR tab - port specific

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

591 009A 19 + ADD HL,DE index + ICR TAB base = effective address
 592 009B CB CE + SET SPEC_ICR,(HL) Set correct bit in ICR TAB
 593 009D 32 02 80 + LD (SEM_REG),A Clear bit 7 of Sem. reg. by writing to it

594 ;
 595 ;PUT CHARACTER IN FIFO AND UPDATE THE TAIL POINTER
 596 ;
 597 00A0 26 C4 REC_0006: LD H,RX_BASE1 Get upper byte of FIFO base
 598 00A2 3A 00 00 E + LD A,(RTAIL_1) Get the value of the tail index
 599 00A5 6F + LD L,A Lower byte of tail pointer to 1 register
 600 00A6 71 + LD (HL),C Put character into buffer
 601 00A7 23 + INC HL Increment address for status byte
 602 00A8 3A 00 00 E + LD A,(STAT_1) Retrieve status byte
 603 00A9 77 + LD (HL),A Put status byte into buffer
 604 00AC AF + XOR A
 605 00AD 32 00 00 E + LD (STAT_1),A Clear status register
 606 00B0 7D + LD A,L Put tail pointer into A reg.
 607 00B1 3C + INC A Update pointer for next char.
 608 00B2 32 00 00 E + LD (RTAIL_1),A Update tail pointer index

609 ;
 610 00B5 D9 REC_0008 EXX
 611 00B6 08 + EX AF,AF'
 612 00B7 FB + EI
 613 00B8 ED 4D + RETI

615 ; RECEIVE ISR FOR PORT #2

617 00B8 D9 REC_2 EXX
 618 00B9 08 EX AF,AF'
 619 00BC RECISR RHEAD_2,RTAIL_2,SIO_1_AD,PORT2,RX_BASE2,STAT_2,BITS_2

620 ;
 621 ;CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
 622 ;IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
 623 ;
 624 00BC 3A 00 00 E + LD A,(RHEAD_2) Retrieve value of head pointer
 625 00BF 47 + LD B,A
 626 00C0 3A 00 00 E + LD A,(RTAIL_2) Retrieve value of tail pointer
 627 00C3 3C + INC A
 628 00C4 3C + INC A Add 2 to tail pointer
 629 00C5 B8 + CP B Compare the two pointers
 630 00C6 20 0A + JR NZ,??0009 If pointers are not equal, jump
 631 00C8 DB B0 + IN A,(SIO_1_AD) Retrieve character and discard
 632 00CA 21 00 00 E + LD HL,STAT_Z Get address of Status byte
 633 00CD CB DE + SET 3,(HL) Set bit 3 in the Status byte
 634 00CF C3 12 01 C + JP ??0012 Jump to exit routine

635 ;
 636 ;RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
 637 ;FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
 638 ;
 639 00D2 DB B0 REC_0009: IN A,(SIO_1_AD) RETRIEVE CHARACTER FROM UART
 640 00D4 21 00 00 E + LD HL,BITS_2

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 5

```

641 00D7 A6      + AND (HL)      Mask off parity bits
642 00D8 4F      + LD C,A
643 00D9 06 00    E + LD B,O      Obtaining a 2-reg. quantity
644 00DB 21 00 00 E + LD HL,BIT_MAP Get address of Bit Map base
645 00DE 09      E + ADD HL,BC   Bit Map Base+char.=effective address
646 00DF CB 56    C + BIT PORT2,(HL) Is the bit for the port = 1?
647 00E1 CA FD 00 C + JP Z,?20010 Jump if not a special character
648
649
650
651 00E4 3A 02 80 E + ?20011: LD A,(SEM_REG) Retrieve address of semaphore register
652 00E7 CB 7F    C + BIT 7,A     If bit 7=0, continue, else check again
653 00E9 C2 E4 00
654
655
656
657 00EC 21 00 CO E + LD HL,INT_COND Retrieve address of Int Cond register
658 00EF CB D6    E + SET PORT2,(HL) Set bit for port specific interrupt
659 00F1 21 00 00 E + LD HL,ICR_TAB Retrieve address of ICR TAB base
660 00F4 11 02 00 E + LD DE,PORT2 Get index into ICR tab - port specific
661 00F7 19      E + ADD HL,DE   index + ICR TAB base = effective address
662 00F8 CB CE    C + SET SPEC_ICR,(HL) Set correct bit in ICR TAB
663 00FA 32 02 80 C + LD (SEM_REG),A Clear bit 7 of Sem. reg. by writing to it
664
665
666
667 00FD 26 C3 00 E + ?20010: LD H,RX_BASE2 Get upper byte of FIFO base
668 00FF 3A 00 00 E + LD A,(RTAIL_2) Get the value of the tail index
669 0102 6F      E + LD L,A     Lower byte of tail pointer to 1 register
670 0103 71      E + LD (HL),C Put character into buffer
671 0104 23      E + INC HL     Increment address for status byte
672 0105 3A 00 00 E + LD A,(STAT_2) Retrieve status byte
673 0108 77      E + LD (HL),A Put status byte into buffer
674 0109 AF      E + XOR A
675 010A 32 00 00 E + LD (STAT_2),A Clear status register
676 010D 7D      E + LD A,L     Put tail pointer into A reg.
677 010E 3C      E + INC A     Update pointer for next char.
678 010F 32 00 00 E + LD (RTAIL_2),A Update tail pointer index
679
680 0112 D9      E + ?20012 EXX
681 0113 08      E + EX AF,AF'
682 0114 FB      E + EI
683 0115 ED 4D    C + RETI
684
685 ; RECEIVE ISR FOR PORT #3
686
687 0117 D9      REC_3 EXX
688 0118 08      REC_3 EX AF,AF'
689 0119 08      RECISR RHEAD_3,RTAIL_3,SIO_1_BD,PORT3,RX_BASE3,STAT_3,BITS_3
690

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 6

```

691
692
693
694 0119 3A 00 00 E + ?CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
695 011C 47      E + ?IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
696
697 0120 3C      E + LD A,(RHEAD_3) Retrieve value of head pointer
698 0121 3C      E + LD B,A     Retrieve value of tail pointer
699 0122 B8      E + INC A     Add 2 to tail pointer
700 0123 20 0A    E + CP B     Compare the two pointers
701 0125 DB B2    E + JR NZ,?20013 If pointers are not equal, jump
702 0127 21 00 00 E + IN A,(SIO_1_BD) Retrieve character and discard
703 0129 CB DE    C + LD HL,STAT_3 Get address of Status byte
704 012C C3 6F 01 C + SET 3,(HL) Set bit 3 in the Status byte
705 012F DB B2    C + JP ?20016 Jump to exit routine
706
707
708
709 012F DB B2    E + ?RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
710 0131 21 00 00 E + FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
711 0134 A6      E + ?20013: IN A,(SIO_1_BD) RETRIEVE CHARACTER FROM UART
712 0135 4F      E + LD HL,BITS_3 Mask off parity bits
713 0136 06 00    E + LD C,A
714 0138 21 00 00 E + LD B,O      Obtaining a 2-reg. quantity
715 0138 09      E + LD HL,BIT_MAP Get address of Bit Map base
716 013C CB 5E    C + ADD HL,BC   Bit Map Base+char.=effective address
717 013E CA 5A 01 C + BIT PORT3,(HL) Is the bit for the port = 1?
718 013F 21 00 00 C + JP Z,?20014 Jump if not a special character
719
720
721 0141 3A 02 80 E + ?BIT MAP POSITION SET - CHECK SEMAPHORE REG. BEFORE SENDING INTERRUPT
722 0144 CB 7F    C + ?20015: LD A,(SEM_REG) Retrieve address of semaphore register
723 0146 C2 41 01 C + BIT 7,A     If bit 7=0, continue, else check again
724
725
726
727 0149 21 00 CO E + ?SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEM_REG
728 014C CB DE    E + LD HL,INT_COND Retrieve address of Int Cond register
729 014E 21 00 00 E + SET PORT3,(HL) Set bit for port specific interrupt
730 0151 11 03 00 E + LD HL,ICR_TAB Retrieve address of ICR TAB base
731 0154 19      E + LD DE,PORT3 Get index into ICR tab - port specific
732 0155 CB CE    C + ADD HL,DE   index + ICR TAB base = effective address
733 0157 32 02 80 C + SET SPEC_ICR,(HL) Set correct bit in ICR TAB
734 0158 21 00 00 C + LD (SEM_REG),A Clear bit 7 of Sem. reg. by writing to it
735
736
737 015A 26 C2 00 E + ?PUT CHARACTER IN FIFO AND UPDATE THE TAIL POINTER
738 015C 3A 00 00 E + ?20014: LD H,RX_BASE3 Get upper byte of FIFO base
739 015F 6F      E + LD A,(RTAIL_3) Get the value of the tail index
740 0160 71      E + LD L,A     Lower byte of tail pointer to 1 register
741 0161 23      E + LD (HL),C Put character into buffer
742 0162 3A 00 00 E + INC HL     Increment address for status byte
743 0165 77      E + LD A,(STAT_3) Retrieve status byte
744 0166 AF      E + XOR A Put status byte into buffer

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 7

745 0167 32 00 00 E + LD (STAT_3),A Clear status register
746 016A 7D + LD A,L Put tail pointer into A reg.
747 016B 3C + INC A Update pointer for next char.
748 016C 32 00 00 E + LD (RTRAIL_3),A Update tail pointer index
749 +:
750 016F D9 +?0016 EXX
751 0170 08 + EX AF,AF'
752 0171 FB + EI
753 0172 ED 4D + RETI

755 0174 END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 1

```

1      SOURCE: &RXERR
2      PROGRAMMER: LIZ POTEET
3
4      4 CHANNEL DIO MUX (FORDYCE) - SPECIAL RECEIVE CONDITION ISR'S
5      FOR PORTS 0 THRU 3 - PARITY, OVERRUN, FRAMING ERRORS
6
7      ****
8      * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9      * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED.
10     * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11     * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12     ****
13
14     LIST B
15     NAME RXERR
16     COPY &MX4EQU
17     LIST S
18
19     PUBLIC RX_ERR0,RX_ERR1,RX_ERR2,RX_ERR3
20
21     EXTRN STAT_0,STAT_1,STAT_2,STAT_3,RHEAD_0,RHEAD_1,RHEAD_2
22     EXTRN RTAIL_0,RTAIL_1,RTAIL_2,RTAIL_3,BIT_MAP
23     EXTRN ICR_TAB,BITS_0,BITS_1,BITS_2,BITS_3
24
25     ****
26     DESCRIPTION: This file contains the Special Receive Condition
27     Interrupt Service Routines which are invoked when the UART has
28     received a character which has one of three error conditions
29     associated with it: a parity error, a UART overflow error, or a
30     framing error. There is one Special Receive Condition ISR for
31     each port. All four are basically the same. The entry points are:
32     RX_ERR0, RX_ERR1, RX_ERR2, and RX_ERR3.
33     The Special Receive Condition ISR's consist of two macros. The
34     first is SPEC_RX and the second RECISR. A description of both of
35     these may be found in the file &MX4EQU.
36     ****
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

476

CSEG

478

; RECEIVE ISR FOR PORT #0

480 0000	RX_ERR0	SPEC_RX SI0_0_AC,STAT_0
481 0000 D9	+ +	EXX CRUTION! IF THIS ROUTINE IS INTERRUPTED
482 0001 08	+ +	EX AF,AF' USE PUSH & POP ELSE MAY LOSE REG. VALUES
483 0002 3E 01	+ +	LD A, (STAT_0)
484 0004 D3 71	+ +	OUT (SI0_0_AC),A Point to Read Register 1
485 0006 DB 71	+ +	IN A,(SI0_0_AC) Read contents of Read Reg. 1
486 0008 CB 27	+ +	SLA A Shift 1 bit left for status byte
487 000A E6 E0	+ +	AND ERR_MSK Mask off all but error bits

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 2

```

488 000C 47      LD B,A      B reg is status byte parameter in RECISR
489 0000 3A 00 00  E + LD A,(STAT_0)  Retrieve value in Status byte
490 0010 B0        + + OR B      Combine with any other possible bits
491 0011 32 00 00  E + LD (STAT_0),A Put new value in Status Register
492 0014 3E 30    + + LD A,30H Code for Error Reset
493 0016 D3 71    + + OUT (SI0_0_AC),A Error latches now reset
494 0018          + + RECISR RHEAD_0,RTAIL_0,SI0_0_AD,PORT0,RX_BASE0,STAT_0,BITS_0
495
496
497
498
499 0018 3A 00 00  E + LD A,(RHEAD_0) Retrieve value of head pointer
500 0018 47        + + LD B,H
501 001C 3A 00 00  E + LD A,(RTAIL_0) Retrieve value of tail pointer
502 001F 3C        + + INC A Add 2 to tail pointer
503 0020 3C        + + CP B Compare the two pointers
504 0021 B8        + + JR NZ,??0001 If pointers are not equal, jump
505 0022 20 0A      + + IN A,(SI0_0_AD) Retrieve character and discard
506 0024 DB 70      + + LD HL,STAT_0 Get address of Status byte
507 0026 21 00 00  E + SET 3,(HL) Set bit 3 in the Status byte
508 0029 CB DE      + + JP ??0004 Jump to exit routine
509 002B C3 6E 00  C +           B
510
511
512
513
514 002E DB 70      ?0001: IN A,(SI0_0_AD) RETRIEVE CHARACTER FROM UART
515 0030 21 00 00  E + LD HL,BITS_0
516 0033 A6        + + AND (HL) Mask off parity bits
517 0034 4F        + + LD C,A
518 0035 06 00      + + LD B,0 Obtaining a 2-reg. quantity
519 0037 21 00 00  E + LD HL,BIT_MAP Get address of Bit Map base
520 003A 09        + + ADD HL,BC Bit Map Base+char.=effective address
521 003B CB 46      + + BIT PORT0,(HL) Is the bit for the port = 1?
522 003D CA 59 00  C + JP Z,??0002 Jump if not a special character
523
524
525
526 0040 3A 02 80  ?0003: LD A,(SEM_REG) Retrieve address of semaphore register
527 0043 CB 7F      + + BIT 7,A If bit 7=0, continue, else check again
528 0045 C2 40 00  C + JP NZ,??0003
529
530
531
532 0048 21 00 C0  ?:
533 0048 CB C6      + + LD HL,INT_COND Retrieve address of Int_Cond register
534 004D 21 00 00  E + SET PORT0,(HL) Set bit for port specific interrupt
535 0050 11 00 00  + + LD HL,ICR_TAB Retrieve address of ICR TAB base
536 0053 19        + + LD DE,PORT0 Get index into ICR tab - port specific
537 0054 CB CE      + + ADD HL,DE index + ICR TAB base = effective address
538 0056 32 02 80  + + SET SPEC_ICR,(HL) Set correct bit in ICR TAB
539
540
541

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

```

542 0059 26 C5      +??0002: LD H_RX_BASE0    Get upper byte of FIFO base
543 005B 3A 00 00   E + LD A,(RTAIL_0)  Get the value of the tail index
544 005E 6F          + LD L,A           Lower byte of tail pointer to l register
545 005F 71          + LD (HL),C        Put character into buffer
546 0060 23          + INC HL           Increment address for status byte
547 0061 3A 00 00   E + LD A,(STAT_0)  Retrieve status byte
548 0064 77          + LD (HL),A        Put status byte into buffer
549 0065 AF          + XOR A            Clear status register
550 0066 32 00 00   E + LD (STAT_0),A  Put tail pointer into A reg.
551 0069 7D          + INC A,L         Update pointer for next char.
552 006A 3C          + LD (HL),A        Update tail pointer index
553 006B 32 00 00   E + LD (RTAIL_0),A
554
555 006E D9          +??0004: EXX
556 006F 08          + EX AF,AF'
557 0070 FB          + EI
558 0071 ED 4D          + RETI

560 ; RECEIVE ISR FOR PORT #1

562 0073 D9          RX_ERR1 SPEC_RX SIO_0_BC,STAT_1
563 0073 D9          + EXX
564 0074 08          + EX AF,AF'
565 0075 3E 01          + LD A,1
566 0077 D3 73          + OUT (SIO_0_BC),A  Point to Read Register 1
567 0079 DB 73          + IN A,(SIO_0_BC)  Read contents of Read Reg. 1
568 007B CB 27          + SLA A
569 007D E6 E0          + AND ERR_MSK  Shift 1 bit left for status byte
570 007F 47          + LD B,A-1
571 0080 3A 00 00   E + LD A,(STAT_1)  Mask off all but error bits
572 0083 B0          + OR B
573 0084 32 00 00   E + LD (STAT_1),A  B reg is status byte parameter in RECISR
574 0087 3E 30          + LD A,30H           Retrieve value in Status byte
575 0089 D3 73          + OUT (SIO_0_BC),A  Combine with any other possible bits
576 008B 08          + LD A,(STAT_1),A  Put new value in Status Register
577
578 ;CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
579 ;IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
580
581 008B 3A 00 00   E + LD A,(RHEAD_1)  Code For Error Reset
582 008E 47          + LD B,A
583 008F 3A 00 00   E + LD A,(RTAIL_1)  Error latches now reset
584 0092 3C          + INC A
585 0093 3C          + INC A
586 0094 B8          + CP B
587 0095 20 0A          + JR NZ,??0005  Add 2 to tail pointer
588 0097 DB 72          + IN A,(SIO_0_BD)  Compare the two pointers
589 0099 21 00 00   E + LD HL,STAT_I  If pointers are not equal, jump
590 009C CB DE          + SET 3,(HL)  Retrieve character and discard
591 009E C3 E1 00   C + JP ??0008  Get address of Status byte
                                         Set bit 3 in the Status byte
                                         Jump to exit routine

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

```

592
593
594
595
596 00A1 DB 72          + RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
597 00A3 21 00 00   E + FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
598 00A6 A6          +??0005: IN A,(SIO_0_BD)  RETRIEVE CHARACTER FROM UART
599 00A7 4F          + LD HL,BITS_1
600 00A8 06 00          + AND (HL)  Mask off parity bits
601 00AA 21 00 00   E + LD C,A
602 00AD 09          + LD B,0
603 00AE CB 4E          + LD HL,BIT_MAP  Obtaining a 2-reg. quantity
604 00B0 CA CC 00   C + ADD HL,BC  Get address of Bit Map base
605
606
607
608 00B3 3A 02 80          +??0007: LD A,(SEM_REG)  Bit Map Base+char.=effective address
609 00B6 CB 7F          + BIT 7,A
610 00B8 C2 B3 00   C + JP NZ,??0007  Is the bit for the port = 1?
                                         If bit 7=0, continue, else check again
611
612
613
614 00BB 21 00 C0          + SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEM_REG
615 00BE CB CE          + LD HL,INT_COND  Retrieve address of Int_Cond register
616 00C0 21 00 00   E + SET PORT1,(HL)  Set bit for port specific interrupt
617 00C3 11 01 00          + LD HL,ICR_TAB  Retrieve address of ICR TAB base
618 00C6 19          + LD DE,PORT1  Get index into ICR tab - port specific
619 00C7 CB CE          + ADD HL,DE  index + ICR TAB base = effective address
620 00C9 32 02 80          + SET SPEC_ICR,(HL)  Set correct bit in ICR TAB
621
622
623
624 00CC 26 C4          + PUT CHARACTER IN FIFO AND UPDATE THE TAIL POINTER
625 00CE 3A 00 00   E +??0006: LD H_RX_BASE1  Get upper byte of FIFO base
626 00D1 6F          + LD A,(RTAIL_1)  Get the value of the tail index
627 00D2 71          + LD L,A
628 00D3 23          + LD (HL),C
629 00D4 3A 00 00   E + INC HL           Put character into buffer
630 00D7 77          + LD A,(STAT_1)  Increment address for status byte
631 00D8 AF          + XOR A
632 00D9 32 00 00   E + LD (STAT_1),A  Retrieve status byte
633 00DC 7D          + LD A,L
634 00DD 3C          + INC A
635 00DE 32 00 00   E + LD (RTAIL_1),A  Put status byte into buffer
636
637 00E1 D9          +??0008: EXX
638 00E2 08          + EX AF,AF'
639 00E3 FB          + EI
640 00E4 ED 4D          + RETI

642 ; RECEIVE ISR FOR PORT #2

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 5

```

644 00E6 RX_ERR2 SPEC_RX SIO_1_AC,STAT_2
645 00E6 D9 EXX CRUTION! IF THIS ROUTINE IS INTERRUPTED
646 00E7 08 EX AF,AF' USE PUSH & POP ELSE MAY LOSE REG. VALUES
647 00E8 3E 01 LD A,I Point to Read Register 1
648 00EA D3 B1 OUT (SIO_1_AC),A Read contents of Read Reg. 1
649 00EC DB B1 IN A,(SIO_1_AC) Shift 1 bit left for status byte
650 00EE CB 27 SLA A Mask off all but error bits
651 00F0 E6 E0 AND ERR_MSK B reg is status byte parameter in RECISR
652 00F2 47 LD B,A Retrieve value in Status byte
653 00F3 3A 00 00 E OR B Combine with any other possible bits
654 00F6 B0 LD (STAT_2),A Put new value in Status Register
655 00F7 32 00 00 E LD (STAT_2),A Code for Error Reset
656 00FA 3E 30 LD A,30H Error latches now reset
657 00FC D3 B1 OUT (SIO_1_AC),A RECISR RHEAD_2,RTAIL_2,SIO_1_AD,PORT2,RX_BASE2,STAT_2,BITS_2
658 00FE

660 :CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
661 :IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
662

663 00FE 3A 00 00 E LD A,(RHEAD_2) Retrieve value of head pointer
664 0101 47 LD B,A
665 0102 3A 00 00 E LD A,(RTAIL_2) Retrieve value of tail pointer
666 0105 3C INC A Add 2 to tail pointer
667 0106 3C CP B Compare the two pointers
668 0107 B8 JR NZ,??0009 If pointers are not equal, jump
669 0108 20 0A IN A,(SIO_1_AD) Retrieve character and discard
670 010A DB 80 LD HL,STAT_2 Get address of Status byte
671 010C 21 00 00 E SET 3,(HL) Set bit 3 in the Status byte
672 010F CB DE JP ??0012 Jump to exit routine
673 0111 C3 54 01 C

674 :RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
675 :FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
676
677

678 0114 DB 80 IN A,(SIO_1_AD) RETRIEVE CHARACTER FROM UART
679 0116 21 00 00 E LD HL,BITS_2
680 0119 A6 AND (HL) Mask off parity bits
681 011A 4F LD C,A
682 011B 06 00 LD B,0 Obtaining a 2-reg. quantity
683 011D 21 00 00 E LD HL,BIT_MAP Get address of Bit Map base
684 0120 09 ADD HL,BC Bit Map Base+char.=effective address
685 0121 CB 56 BIT PORT2,(HL) Is the bit for the port = 1?
686 0123 CA 3F 01 C JP Z,??0010 Jump if not a special character
687

688 :BIT MAP POSITION SET - CHECK SEMAPHORE REG. BEFORE SENDING INTERRUPT
689
690 0126 3A 02 80 ??0011: LD A,(SEM_REG) Retrieve address of semaphore register
691 0129 CB 7F BIT 7,A If bit 7=0, continue, else check again
692 012B C2 26 01 C JP NZ,??0011
693
694 :SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEM_REG
695
696 012E 21 00 CO LD HL,INT_COND Retrieve address of Int_Cond register

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 6

```

697 0131 CB D6 SET PORT2,(HL) Set bit for port specific interrupt
698 0133 21 00 00 E LD HL,ICR_TAB Retrieve address of ICR TAB base
699 0136 11 02 00 LD DE,PORT2 Get index into ICR tab - port specific
700 0139 19 ADD HL,DE index + ICR TAB base = effective address
701 013A CB CE SET SPEC_ICR,(HL) Set correct bit in ICR TAB
702 013C 32 02 80 LD (SEM_REG),H Clear bit 7 of Sem. reg. by writing to it
703
704 :PUT CHARACTER IN FIFO AND UPDATE THE TAIL POINTER
705
706 013F 26 C3 ??0010: LD H,RX_BASE2 Get upper byte of FIFO base
707 0141 3A 00 00 E LD A,(RTAIL_2) Get the value of the tail index
708 0144 6F LD L,A Lower byte of tail pointer to L register
709 0145 71 LD (HL),C Put character into buffer
710 0146 23 INC HL Increment address for status byte
711 0147 3A 00 00 E LD A,(STAT_2) Retrieve status byte
712 014A 77 LD (HL),A Put status byte into buffer
713 014B AF XOR A
714 014C 32 00 00 E LD (STAT_2),A Clear status register
715 014F 7D LD A,L Put tail pointer into A reg.
716 0150 3C INC A Update pointer for next char.
717 0151 32 00 00 E LD (RTAIL_2),A Update tail pointer index
718
719 0154 D9 ??0012 EXX
720 0155 08 EX AF,AF'
721 0156 FB EI
722 0157 ED 4D RETI

724 ; RECEIVE ISR FOR PORT #3

726 0159 D9 RX_ERR3 SPEC_RX SIO_1_BC,STAT_3
727 0159 D9 EXX CRUTION! IF THIS ROUTINE IS INTERRUPTED
728 015A 08 EX AF,AF' USE PUSH & POP ELSE MAY LOSE REG. VALUES
729 015B 3E 01 LD A,I Point to Read Register 1
730 015D D3 B3 OUT (SIO_1_BC),A Read contents of Read Reg. 1
731 015F DB B3 IN A,(SIO_1_BC) Shift 1 bit left for status byte
732 0161 CB 27 SLA A Mask off all but error bits
733 0163 E6 E0 AND ERR_MSK B reg is status byte parameter in RECISR
734 0165 47 LD B,A Retrieve value in Status byte
735 0166 3A 00 00 E OR B Combine with any other possible bits
736 0169 B0 LD (STAT_3),A Put new value in Status Register
737 016A 32 00 00 E LD (STAT_3),A Code for Error Reset
738 016D 3E 30 LD A,30H Error latches now reset
739 016F D3 B3 OUT (SIO_1_BC),A RECISR RHEAD_3,RTAIL_3,SIO_1_BD,PORT3,RX_BASE3,STAT_3,BITS_3
740 0171

742 :CHECK IF BUFFER IS FULL - COMPARE HEAD AND TAIL POINTERS
743 :IF EQUAL THEN RETRIEVE CHAR. AND DISCARD. SET BIT IN STATUS WORD
744
745 0171 3A 00 00 E LD A,(RHEAD_3) Retrieve value of head pointer
746 0174 47 LD B,A

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 7

```

747 0175 3A 00 00   E + LD A,(RTAIL_3)      Retrieve value of tail pointer
748 0178 3C          + INC A
749 0179 3C          + INC A
750 017A B8          + CP B
751 017B 20 0A       + JR NZ,?0013    Compare the two pointers
752 017D DB B2       + IN A,(SIO_1_BD)  If pointers are not equal, jump
753 017F 21 00 00   E + LD HL,STAT_3  Retrieve character and discard
754 0182 CB DE       + SET 3,(HL)    Get address of Status byte
755 0184 C3 C7 01   C + JP ??0016    Set bit 3 in the Status byte
756
757          + RETRIEVE CHARACTER AND DO BIT MAP CHECK. IF SPECIAL CHARACTER (BIT
758          + FOR THE PORT IS SET), SEND A SPECIAL CHARACTER INTERRUPT TO HOST.
759
760 0187 DB B2       E + ??0013: IN A,(SIO_1_BD)  RETRIEVE CHARACTER FROM UART
761 0189 21 00 00   E + LD HL,BITS_3
762 018C A6          + AND (HL)
763 018D 4F          + LD C,A
764 018E 06 00       + LD B,0
765 0190 21 00 00   E + LD HL,BIT_MAP
766 0193 09          + ADD HL,BC
767 0194 CB SE       + BIT PORT3,(HL)
768 0196 CA B2 01   C + JP Z,??0014  Is the bit for the port = 1?
769
770          + BIT MAP POSITION SET - CHECK SEMAPHORE REG. BEFORE SENDING INTERRUPT
771
772 0199 3A 02 80   E + ??0015: LD A,(SEM_REG)  Retrieve address of semaphore register
773 019C CB 7F       + BIT 7,A
774 019E C2 99 01   C + JP NZ,??0015  If bit 7=0, continue, else check again
775
776          + SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEM_REG
777
778 01A1 21 00 C0   E + LD HL,INT_COND
779 01A4 CB DE       + SET PORT3,(HL)  Retrieve address of Int_Cond register
780 01A6 21 00 00   E + LD HL,ICR_TAB
781 01A9 11 03 00   E + LD DE,PORT3
782 01AC 19          + ADD HL,DE
783 01AD CB CE       + SET SPEC_ICR,(HL)
784 01AF 32 02 80   E + LD (SEM_REG),A
785
786          + PUT CHARACTER IN FIFO AND UPDATE THE TAIL POINTER
787
788 01B2 26 C2       E + ??0014: LD H,RX_BASE3
789 01B4 3A 00 00   E + LD A,(RTAIL_3)  Get upper byte of FIFO base
790 01B7 6F          + LD L,A
791 01B8 71          + LD (HL),C
792 01B9 23          + INC HL
793 01B9 3A 00 00   E + LD A,(STAT_3)
794 01BD 77          + LD (HL),A
795 01BE AF          + XOR A
796 01BF 32 00 00   E + LD (STAT_3),A
797 01C2 7D          + LD A,L
798 01C3 3C          + INC A
799 01C4 32 00 00   E + LD (RTAIL_3),A
800          +;

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 8

```

801 01C7 D9          +??0016  EXX
802 01C8 08          + EX AF,AF'
803 01C9 FB          + EI
804 01CA ED 4D       + RETI

```

806 01CC

END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

```
1      SOURCE: &MX4TX
2      PROGRAMMER: LIZ POTEET
3
4      4 CHANNEL DIO MUX (FORDYCE) - TRANSMIT ISR - UART TRIGGERED
5
6      ****
7      * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
8      * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
9      * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
10     * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
11
12
13     LIST X
14     NAME MX4TX
15     COPY &MX4EQU$S
16     LIST S
17
18     PUBLIC TX_0,TX_1,TX_2,TX_3
19
20     ; EXTRN THEAD_0,THEAD_1,THEAD_2
21     ; EXTRN TTAIL_0,TTAIL_1,TTAIL_2,TTAIL_3,TON0
22     ; EXTRN TON1,TON2,TON3,ICR_TAB
23
24
25     ****
26     DESCRIPTION: This file contains the Transmit Interrupt Service
27     Routines which are invoked when one of the SIO channels has finished
28     sending out a character and is ready for the next. The transmit
29     ISR is fundamentally the same for each of the four ports. The macro
30     TX_ISR is called in each and appropriate parameters are passed. A
31     description of the macro is contained in the file &MX4EQU$. The
32     entry points for the ISR's are: TX_0, TX_1, TX_2, and TX_3.
33
34
35     CSEG
36
37     ; TRANSMIT ISR FOR PORT #0
38
39
40     TX_0    TX_ISR SIO_0_AC,TON0,THEAD_0,SIO_0_AD,TFIFO_0,TTAIL_0,PORT0
41     +;
42     +;
43     +; EXX          !CAUTION If this routine is interrupted,
44     +; EX  AF,AF'      use push and pop or may lose reg. data
45     +;
46     +; TEST IF TX BUFFER IS EMPTY
47     +;
48     E   LD  A,(THEAD_0)      Retrieve value in Head pointer index
49     +; LD  B,A
50     +; LD  B,(TTAIL_0)      Retrieve value in Tail pointer index
51     E   CP  B                  Compare; if Head=Tail then buffer is empty
52
53     0000  D9
54     0001  08
55
56     0002  3A 00 00
57     0005  47
58     0006  3A 00 00
59     0009  B8
```

Z80 ASSEMBLER VER 3.0MR

PAGE 2

ERR LINE	ADDR	B1	B2	B3	B4	
488	000A	20	24			JR NZ,??20001 If not empty, jump over next code sectio
489						BUFFER EMPTY - TURN OFF UART, SET TRAN-ON FLAG TO 0, EXIT ROUTINE
490						LD A,28H Code for Reset TX Pending in WR1
491						OUT (\$IO_0_AC),A Send to UART. TX interrupts now stopped
492	000C	3E	28			XOR A Register
493	000E	D3	71			LD (TON0),A Flag now indicates transmitter off
494	0010	RF				SEND HOST A TX EMPTY INTERRUPT - CHECK SEMAPHORE REG. FIRST
495	0011	32	00	00	E	?20002 LD A,(SEM_REG) Retrieve contents of Semaphore register
496						BIT 7,A Is bit 7 set? If so check again else cont.
497						JP NZ,??20002
498	0014	3A	02	80	C	SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEMAPHORE
500	0017	CB	7F			LD HL,INT_COND Retrieve address of Int_Cond register
501	0019	C2	14	00		SET PORT0,(THL) Set bit for Port Specific interrupt
502						LD HL,ICR_TAB Retrieve address of ICR_TAB
503						LD DE,PORT0 Get index to ICR_TAB
504						ADD HL,DE Obtain effective address
505	001C	21	00	C0		SET TX_ICR,(HL) Set correct bit in ICR_TAB
506	001F	CB	C6			LD (SEM_REG),A Clear bit 7 of sem. reg by writing to it
507	0021	21	00	00	E	JP ??20003 Go to end of routine
508	0024	11	00	00		RETRIEVE CHARACTER FROM FIFO AND INCREMENT POINTERS
509	0027	19				LD D,TX_BASE Retrieve the base upper byte of pointer
510	0028	CB	C6		E	LD A,(THEAD_0) Get lower byte of head pointer index
511	002A	32	02	80		ADD A,TFIFO_0 Add base lower byte to index
512	002D	C3	42	00	C	LD E,A Save head pointer so can use A register
513						LD A,(DE) RETRIEVE CHARACTER
514						OUT (\$IO_0_AD),A Send character to UART
515						LD A,E Get lower byte again
516	0030	16	C7		E	INC A,E Increment lower byte of head pointer
517	0032	3A	00	00		AND TMSK TMSK isolates lower nibble (index)
518	0035	C6	A0			LD (THEAD_0),A Save updated pointer (lower byte)
519	0037	SF				EXX
520	0038	1A				EX AF,AF'
521	0039	D3	70			EI
522	003B	7B				RET
523	003C	3C				
524	003D	E6	0F			
525	003F	32	00	00	E	
526						
527	0042	D9				
528	0043	08				
529	0044	FB				
530	0045	ED	4D			
532						; TRANSMIT ISR FOR PORT #1
534	0047					TX_1 TX_ISR SIO_0_BC,TON1,THEAD_1,SIO_0_BD,TFIFO_1,TTAIL_1,PORT1
535						
536	0047	D9				!CAUTION If this routine is interrupted,
537	0048	08				use push and pop or may lose reg. data

Z80 ASSEMBLER VER 3.0MR

PAGE 3

ERR LINE	ADDR	B1	B2	B3	B4	
538						TEST IF TX BUFFER IS EMPTY
539						LD A,(THEAD_1) Retrieve value in Head pointer index
540						LD B,A
541	0049	3A	00	00	E	LD A,(TTAIL_1) Retrieve value in Tail pointer index
542	004C	47				CP B,B Compare; if Head=Tail then buffer is empty
543	004D	3A	00	00	E	JR NZ,??20004 If not empty, jump over next code sectio
544	0050	B8				BUFFER EMPTY - TURN OFF UART, SET TRAN-ON FLAG TO 0, EXIT ROUTINE
545	0051	20	24			LD A,28H Code for Reset TX Pending in WR1
546						OUT (\$IO_0_BC),A Send to UART. TX interrupts now stopped
547						XOR A Register
548						LD (TON1),A Flag now indicates transmitter off
549	0053	3E	28			SEND HOST A TX EMPTY INTERRUPT - CHECK SEMAPHORE REG. FIRST
550	0055	D3	73			?20005 LD A,(SEM_REG) Retrieve contents of Semaphore register
551	0057	RF				BIT 7,A Is bit 7 set? If so check again else cont.
552	0058	32	00	00	E	JP NZ,??20005
553						SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEMAPHORE
554						LD HL,INT_COND Retrieve address of Int_Cond register
555						SET PORT1,(THL) Set bit for Port Specific interrupt
556	005B	3A	02	80	C	LD HL,ICR_TAB Retrieve address of ICR_TAB
557	005E	CB	7F			LD DE,PORT1 Get index to ICR_TAB
558	0060	C2	5B	00		ADD HL,DE Obtain effective address
559						SET TX_ICR,(HL) Set correct bit in ICR_TAB
560						LD (SEM_REG),A Clear bit 7 of sem. reg by writing to it
561						JP ??20006 Go to end of routine
571						RETRIEVE CHARACTER FROM FIFO AND INCREMENT POINTERS
572	0077	16	C7		E	?20004 LD D,TX_BASE Retrieve the base upper byte of pointer
574	0079	3A	00	00		LD A,(THEAD_1) Get lower byte of head pointer index
575	007C	C6	90			ADD A,TFIFO_0 Add base lower byte to index
576	007E	SF				LD E,A Save head pointer so can use A register
577	007F	1A				LD A,(DE) RETRIEVE CHARACTER
578	0080	D3	72			OUT (\$IO_0_BD),A Send character to UART
579	0082	7B				LD A,E Get lower byte again
580	0083	3C				INC A,E Increment lower byte of head pointer
581	0084	E6	0F			AND TMSK TMSK isolates lower nibble (index)
582	0086	32	00	00	E	LD (THEAD_1),A Save updated pointer (lower byte)
583						EXX
584	0089	D9				EX AF,AF'
585	008A	08				EI
587	008C	ED	4D			RET
589						; TRANSMIT ISR FOR PORT #2

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 4

591 008E TX_2 TX_ISR SIO_1_AC,TON2,THEAD_2,SIO_1_AD,TFIFO_2,TTAIL_2,PORT2
 592 +;
 593 008E D9 EXX !CAUTION If this routine is interrupted,
 594 008F 08 EX AF,AF' use push and pop or may lose reg. data
 595
 596 ;TEST IF TX BUFFER IS EMPTY
 597 598 0090 3A 00 00 E LD A,(THEAD_2) Retrieve value in Head pointer index
 599 0093 47 LD B,A
 600 0094 3A 00 00 E LD A,(TTAIL_2) Retrieve value in Tail pointer index
 601 0097 B8 CP B Compare; if Head=Tail then buffer is empty
 602 0098 20 24 JR NZ,??0007 If not empty, jump over next code section
 603
 604 ;BUFFER EMPTY - TURN OFF UART, SET TRAN-ON FLAG TO 0, EXIT ROUTINE
 605
 606 009A 3E 28 LD A,28H Code for Reset TX Pending in WR1
 607 009C D3 B1 OUT (SIO_1_AC),A Send to UART. TX interrupts now stopped
 608 009E AF XOR A Clear A register
 609 009F 32 00 00 E LD (TON2),A Flag now indicates transmitter off
 610
 611 ;SEND HOST A TX EMPTY INTERRUPT - CHECK SEMAPHORE REG. FIRST
 612
 613 00A2 3A 02 80 ??0008 LD A,(SEM_REG) Retrieve contents of Semaphore register
 614 00A5 CB 7F BIT 7,A Is bit 7 set? If so check again else cont.
 615 00A7 C2 A2 00 C JP NZ,??0008
 616
 617 ;SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEMAPHORE
 618
 619 00AA 21 00 C0 LD HL,INT_COND Retrieve address of Int_Cond register
 620 00AD CB D6 SET PORT2,(HL) Set bit for Port Specific interrupt
 621 00AF 21 00 00 E LD HL,ICR_TAB Retrieve address of ICR_TAB
 622 00B2 11 02 00 LD DE,PORT2 Get index to ICR_TAB
 623 00B5 19 ADD HL,DE Obtain effective address
 624 00B6 CB C6 SET TX_ICR,(HL) Set correct bit in ICR_TAB
 625 00B8 32 02 80 LD (SEM_REG),A Clear bit 7 of sem. reg by writing to it
 626 00B8 C3 D0 00 C JP ??0009 Go to end of routine
 627
 628 ;RETRIEVE CHARACTER FROM FIFO AND INCREMENT POINTERS
 629
 630 00B8 16 C7 ??0007 LD D,TX_BASE Retrieve the base upper byte of pointer
 631 00C0 3A 00 00 E LD A,(THEAD_2) Get lower byte of head pointer index
 632 00C3 C6 80 ADD A,TFIFO_2 Add base lower byte to index
 633 00C5 5F LD E,A Save head pointer so can use A register
 634 00C6 1A LD A,(DE) RETRIEVE CHARACTER
 635 00C7 D3 B0 OUT (SIO_1_AD),A Send character to UART
 636 00C9 7B LD A,E Get lower byte again
 637 00CA 3C INC A Increment lower byte of head pointer
 638 00CB E6 0F AND TMSK TMSK isolates lower nibble (index)
 639 00CD 32 00 00 E LD (THEAD_2),A Save updated pointer (lower byte)
 640
 641 00D0 D9 ??0009 EXX
 642 00D1 08 EX AF,AF'
 643 00D2 FB EI

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 5

644 00D3 ED 4D + RETI
 646 ; TRANSMIT ISR FOR PORT #3
 648 00D5 TX_3 TX_ISR SIO_1_BC,TON3,THEAD_3,SIO_1_BD,TFIFO_3,TTAIL_3,PORT3
 649 +;
 650 00D5 D9 EXX !CAUTION If this routine is interrupted,
 651 00D6 08 EX AF,AF' use push and pop or may lose reg. data
 652
 653 ;TEST IF TX BUFFER IS EMPTY
 654 655 00D7 3A 00 00 E LD A,(THEAD_3) Retrieve value in Head pointer index
 656 00DA 47 LD B,A
 657 00DB 3A 00 00 E LD A,(TTAIL_3) Retrieve value in Tail pointer index
 658 00DE B8 CP B Compare; if Head=Tail then buffer is empty
 659 00DF 20 24 JR NZ,??0010 If not empty, jump over next code section
 660
 661 ;BUFFER EMPTY - TURN OFF UART, SET TRAN-ON FLAG TO 0, EXIT ROUTINE
 662
 663 00E1 3E 28 LD A,28H Code for Reset TX Pending in WR1
 664 00E3 D3 B3 OUT (SIO_1_BC),A Send to UART. TX interrupts now stopped
 665 00E5 AF XOR A Clear A register
 666 00E6 32 00 00 E LD (TON3),A Flag now indicates transmitter off
 667
 668 ;SEND HOST A TX EMPTY INTERRUPT - CHECK SEMAPHORE REG. FIRST
 669
 670 00E9 3A 02 80 ??0011 LD A,(SEM_REG) Retrieve contents of Semaphore register
 671 00EC CB 7F BIT 7,A Is bit 7 set? If so check again else cont.
 672 00EE C2 E9 00 C JP NZ,??0011
 673
 674 ;SEMAPHORE REGISTER SET - SEND INTERRUPT TO HOST & RELEASE SEMAPHORE
 675
 676 00F1 21 00 C0 LD HL,INT_COND Retrieve address of Int_Cond register
 677 00F4 CB DE SET PORT3,(HL) Set bit for Port Specific interrupt
 678 00F6 21 00 00 E LD HL,ICR_TAB Retrieve address of ICR_TAB
 679 00F9 11 03 00 LD DE,PORT3 Get index to ICR_TAB
 680 00FC 19 ADD HL,DE Obtain effective address
 681 00FD CB C6 SET TX_ICR,(HL) Set correct bit in ICR_TAB
 682 00FF 32 02 80 LD (SEM_REG),A Clear bit 7 of sem. reg by writing to it
 683 0102 C3 17 01 C JP ??0012 Go to end of routine
 684
 685 ;RETRIEVE CHARACTER FROM FIFO AND INCREMENT POINTERS
 686
 687 0105 16 C7 ??0010 LD D,TX_BASE Retrieve the base upper byte of pointer
 688 0107 3A 00 00 E LD A,(THEAD_3) Get lower byte of head pointer index
 689 0109 C6 70 ADD A,TFIFO_3 Add base lower byte to index
 690 010C 5F LD E,A Save head pointer so can use A register
 691 010D 1A LD A,(DE) RETRIEVE CHARACTER
 692 010E D3 B2 OUT (SIO_1_BD),A Send character to UART
 693 0110 78 LD A,E Get lower byte again

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 6

694 0111 3C + INC A Increment lower byte of head pointer
695 0112 E6 0F + AND TMSK TMSK isolates lower nibble (index)
696 0114 32 00 00 E + LD (THEAD_3),A Save updated pointer (lower byte)
697 +:
698 0117 D9 +?0012 EXX
699 0118 08 + EX AF,AF'
700 0119 FB + EI
701 011A ED 4D + RETI

703 011C END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 1

```
1 SOURCE: &MXTMR
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - TIMER INTERRUPT SERVICE ROUTINE
5 FOR 16 MILLSEC. TIME OUT
6
7 ****
8 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9 RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
10 REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11 THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12 ****
13
14 LIST B
15 NAME MXTMR
16 COPY &MX4EQU$S
17 LIST S
18
19 ; PUBLIC TMR_ISR
20
21 ****
22 NAME: TMR_ISR
23
24 DESCRIPTION:
25 This ISR is called every 16 milliseconds when the CTC downcounts
26 to zero. The purpose of this routine is to send a Timer interrupt
27 to the host.
28 ****
29
30 CSEG
31
32 TMR_ISR:
33 EXX CAUTION! IF ANOTHER ISR CAN INTERRUPT, USE
34 EX AF,AF' PUSH & POP SO THAT DONT LOSE REG. CONTENTS
35
36 ;CHECK SEMAPHORE BEFORE SETTING INTERRUPT
37
38 TM1: LD A,(SEM_REG) Get contents of Sem_Reg
39 BIT 7,A If bit 7=0 continue else recycle
40 JP NZ,TM1
41
42 ;SEMAPHORE REG. SET - SEND INTERRUPT TO HOST AND RELEASE SEM. REGISTER
43
44 LD HL,INT_COND Get address of Int_Cond register
45 SET 6,(HL) Set bit for timer Interrupt
46
47
48
49
```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 2

```
466 000F 32 02 80 LD (SEM_REG),A Clear bit 7 of Sem. reg by writing to it
467
468 0012 D9 EXX
469 0013 08 EX AF,AF'
470 0014 FB EI
471 0015 ED 4D RETI
472 0017 END
```

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

280 ASSEMBLER VER 3.0MR

PAGE 1

```
1 SOURCE: &MXHST
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - HOST INTERRUPT SERVICE ROUTINE
5 (GENERATED BY CTC INTERRUPT)
6 ****
7 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
8 * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
9 * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
10 * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
11 ****
12
13 NAME MXHST
14 COPY &MX4EQU8
15 LIST S
16
17 PUBLIC HSTINT
18 PUBLIC EEE2,EEE3,EEE4,EEE5,EEE6,EEE7
19
20 EXTRN ISRPT0,ISRPT1,ISRPT2,ISRPT3
21 EXTRN TMPTAB,CHND_TAB
22 EXTRN MODOUT,THROFF,MX4ST
23
24 ****
25 ;HOST INTERRUPT SERVICE ROUTINE
26
27 ; This file contains the service routine which is invoked by CTC-0
28 ; when the host puts an interrupt into the COMMAND register. This
29 ; routine empties the contents of the CHND_TAB and COMMAND registers
30 ; and begins checking the bits in both to determine what type of host
31 ; interrupt was requested. When the interrupt has been interpreted and
32 ; serviced, the program jumps back to the beginning of this file to
33 ; see if the host sent another interrupt during the course of the
34 ; program. If the COM_REG is empty, the program will jump to the exit.
35 ; If there is data in the COM_REG, the program will begin servicing
36 ; that data. In other words, it is possible for this routine to service
37 ; more than one host interrupt. The reason for this is that if the
38 ; host sends an interrupt during the course of this routine (when
39 ; interrupts are disabled) the card will not get it due to the inability
40 ; of the CTC to buffer interrupts.
41
42 ;FILES CALLED: MXPT0, MXPT1, MXPT2, MXPT3, MXMOD, EXTMR, MX4ST
43 ;*****
44
45 CSEG
46
47 HSTINT:
48     EXX
49     EX AF,AF
50
51 CAUTION! IF ANOTHER ISR CAN INTERRUPT, USE
52 PUSH & POP SO THAT DONT LOSE REG. CONTENTS
```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

490 ;GRAB SEMAPHORE

492 0002 3A 02 80 EEE1: LD A,(SEM_REG) Contents of SEM_REG to A reg.
493 0005 CB 7F C BIT 7,A Test bit 7. If=1 then recycle
494 0007 C2 02 00 C JP NZ,EEE1

496 ;RETRIEVE VALUE IN COMMAND REG. & CMND_TAB.

498 0008 3A 01 C0 LD A,(COM_REG)
499 000D FE 00 CP 0 IS THE COMMAND REG. EMPTY?
500 000F 20 03 JR NZ,TST NO
501 0011 C3 5E 00 C JP EEE9 YES - EXIT ROUTINE

503 0014 01 04 00 TST: LD BC,4 Writing contents of CMND_TAB to TMPTAB
504 0017 11 00 00 E LD DE,TMPTAB so that I can release the semaphore
505 001A 21 00 00 LD HL,CMND_TAB quickly.
506 001D ED 80 LDIR

508 ;CLEAR COMMAND REG. AND CMND_TAB

510 001F SF LD E,A Save COMMAND reg value in E reg.
511 0020 AF XOR A
512 0021 32 01 C0 LD (COM_REG),A Clear COMMAND register
513 0024 21 00 00 E LD HL,CMND_TAB Get address if command table
514 0027 77 LD (HL),A Clear 1st byte in CMND-TAB
515 0028 23 INC HL
516 0029 77 LD (HL),A Clear 2nd byte
517 002B 23 INC HL
518 002B 77 LD (HL),A Clear 3rd byte
519 002C 23 INC HL
520 002D 77 LD (HL),A Clear 4th byte
521 002E 32 02 80 LD (SEM_REG),A Release SEM_REG by writing to it

523 BEGIN DETERMINING WHICH BITS IN CMND_TAB (NOW IN TMPTAB) ARE SET

524

525 NOTE: The following jumps should be treated as subroutine jumps.
526 The jump instructions are used instead because they are faster and
527 there is a time constraint on this code.

528

529 REG. USAGE - The E register is used to hold the contents of the A
530 register during the jump to subroutine. In other words, the E
531 register must not be used in any of the following routines.

533 0031 7B LD A,E
534 0032 1F RRA A,E Rotate Bit 0 of COMMAND reg data to Carry
535 0033 5F LD E,A Store temporarily in E

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

536 0034 DA 00 00 E JP C,ISRPT0 If bit 0=1, jmp to Pt 0 interrupt routine

538 0037 7B EEE2: LD A,E Retrieve rotated pattern of A register
539 0038 1F RRA A,E Rotate bit 1 into Carry flag
540 0039 5F LD E,A Store value temporarily in E reg.
541 003A DR 00 00 E JP C,ISRPT1 If bit 1=1,jmp to port 1 interrupt routine

543 003D 7B EEE3: LD A,E Retrieve rotated bit pattern again
544 003E 1F RRA A,E Rotate bit 2 into Carry flag
545 003F 5F LD E,A Save temporarily in E register
546 0040 DA 00 00 E JP C,ISRPT2 If bit 2=1,jmp to port 2 interrupt routine

548 0043 7B EEE4: LD A,E Retrieve rotated bit pattern again
549 0044 1F RRA A,E Rotate bit 3 into Carry flag
550 0045 5F LD E,A Save bit pattern temporarily in E reg.
551 0046 DA 00 00 E JP C,ISRPT3 If bit 3=1,jmp to port 3 interrupt routine

553 0049 7B EEE5: LD A,E Retrieve rotated bit pattern again
554 0049 1F RRA A,E Rotate bit 4 into Carry flag
555 0048 5F LD E,A Save bit pattern temporarily in E reg.
556 004C DA 00 00 E JP C,MODOUT If bit 4=1, jmp to Modem Out. line routine

558 004F 7B EEE6: LD A,E Rotate bit 5 into Carry flag
559 0050 1F RRA A,E
560 0051 5F LD E,A
561 0052 DA 00 00 E JP C,TMROFF If bit 5=1,jmp to Timer On/Off routine

563 0055 7B EEE7: LD A,E Rotate bit 6 into Carry flag
564 0056 1F RRA A,E
565 0057 5F LD E,A
566 0058 DA 00 00 E JP C,MX4ST If bit 6=1,jmp to beginning of Self Test

568 005B C3 02 00 C EEE8: JP EEE1 Return to beginning of routine
569 005E D9 EEX (SEM_REG),A Release SEM_REG by writing to it
570 005F 32 02 80 LD AF,AF'
571 0062 08 EX AF,AF'
572 0063 FB EI
573 0064 ED 40 RETI
574 0066 END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

```

1 SOURCE: &MXEXT
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - EXTERNAL STATUS INTERRUPT ISR'S
5 ****
6 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
7 * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
8 * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
9 * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
10 ****
11
12
13 LIST X
14 NAME MXEXT
15 COPY &MX4EQU$U
16 LIST S
17
18 PUBLIC EX_0,EX_1,EX_2,EX_3,MDM3_SUB,MDM1_SUB
19
20 EXTRN RBRK_0,RBRK_1,RBRK_2,RBRK_3,STAT_0,STAT_1
21 EXTRN STAT_2,STAT_3,MDM_IN,MDM_MASK
22
23 ****
24 EXTERNAL STATUS INTERRUPT ROUTINES
25
26 These ISR's are called when one of the SIO channels has a trans-
27 ition on either the Break,DCD,CTS, or SYNC inputs. A TX underrun
28 will also cause this interrupt although these routines will not take
29 any action if that is what has triggered the ISR. This file contains
30 the External Status ISR's for all four ports. However, each of the
31 ports expects different combinations of transitions. Therefore the
32 ISR's are different for each. The only thing in common is that each
33 of the ports may get a Break condition. The following are the valid
34 transitions for each port and an explanation of what these lines
35 represent.
36
37 PORT 0 - BREAK Beginning or end of Break occurrence
38 DCD Receiver Ready modem line change
39 CTS Clear to Send modem line change
40 SYNC Data Mode modem line change
41
42 PORT 1 - BREAK Beginning or end of Break occurrence
43 DCD Incoming Call modem line change
44
45 PORT 2 - BREAK Beginning or end of Break occurrence
46
47 PORT 3 - BREAK Beginning or end of Break occurrence
48
49 The Break processing code of each ISR is contained in a macro
50 called BREAK. This macro is part of the file &MX4EQU.
51
52 UPON EXIT OF BREAK MACRO:
53 B Register contains contents of SIO Read Reg. #0

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

493 ; External Status interrupts have already been set
494 ;*****

496 **CSEG**

498 ; EXTERNAL STATUS ISR FOR PORT #0

500 0000 D9 EX_0 EXX !CAUTION - IF THIS ROUTINE IS INTERRUPTED
501 0001 08 EX AF,AF' USE PUSH & POP ELSE MAY LOSE REG. VALUES

503 ;WHY IS THE INTERRUPT CAUSED BY BEGINNING OR END OF BREAK?

S05 0002 0E 71 LD C,SIO_0.AC PARAMETERS FOR BREAK SUBROUTINE
 S06 0004 21 00 00 LD HLRBRK_0
 S07 0007 11 00 00 LD DE_STAT_0
 S08 0009 CD 4E 00 CALL BRK_SUB
 S09 000D CD 80 00 CALL MDM3_SUB

511	0010	D9	EXX
512	0011	08	EX AF,AF'
513	0012	FB	EI
514	0013	ED 4D	RETI

516 ; EXTERNAL STATUS ISR FOR PORT #1

518 0015 D9 EX_1 EXX !CAUTION - IF THIS ROUTINE IS INTERRUPTED
519 0016 08 EX AF,AF' USE PUSH & POP ELSE MAY LOSE REG. VALUES

521 ;WAS THE INTERRUPT CAUSED BY BEGINNING OR END OF BREAK?

523 0017 0E 73 LD C,SIO_0_BC PARAMETERS FOR BREAK SUBROUTINE
 524 0019 21 00 00 LD HL,RBRK_1
 525 001C 11 00 00 LD DE,STAT_1
 526 001F CD 4E 00 CALL BRK_SUB

528 0022 CD AF 00 C ;CALL SUBROUTINE WHICH DETERMINES WHETHER THE DCD INPUT LINE CHANGED
529 0025 D9 CALL MM1_SUB
530 0025 D9 EXX

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

531 0026 08
532 0027 FB
533 0028 ED 4D

535 : EXTERNAL STATUS ISR FOR PORT #2

537 002A D9 **EX_2:** **EXX** **!CAUTION - IF THIS ROUTINE IS INTERRUPTED**
538 002B 08 **EX AF,AF'** **USE PUSH & POP ELSE MAY LOSE REG. VALUES**

540 ; WAS THE INTERRUPT CAUSED BY BEGINNING OR END OF BREAK?

542 002C 0E B1 LD C,SIO_1_AC PARAMETERS FOR BREAK SUBROUTINE
543 002E 21 00 00 LD HL,RBRK_2
544 0031 11 00 00 LD DE,STAT_2
545 0034 CD 4E 00 CALL BRK SUB

547 0037 D9 EXX
548 0038 08 EX AF,AF'
549 0039 FB EI
550 0020 ED PETT

552 : EXTERNAL STATUS ISR FOR PORT #3

554 003C D9 **EX_3:** **EXX** **!CAUTION - IF THIS ROUTINE IS INTERRUPTED**
555 003D 08 **EX AF,AF'** **USE PUSH & POP, ELSE MAY LOSE REG. VALUES**

557 ; WAS THE INTERRUPT CAUSED BY BEGINNING OR END OF BREAK?

559 003E 0E B3 LD C,SIO_1_BC PARAMETERS FOR BREAK SUBROUTINE
 560 0040 21 00 00 LD HL,RBRK_3
 561 0043 11 00 00 LD DE,STAT_3
 562 0046 CD 4E 00 CALL BRK SUB

564	0049	D9		EXX
565	004A	08		EX AF,AF'
566	004B	FB		EI
567	004C	ED AD		PETI

568

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

```

569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602 004E ED 40      BRK_SUB: IN  B,(C)           Get contents of SIO Read Reg. 0
603 0050 3E 10      LD   A,10H
604 0052 ED 79      OUT  (C),A
605 0054 CB 46      BIT   0,(HL)
606 0056 20 11      JR   NZ,B1
607 0058 CB 78      BIT   7,B
608 005A 28 23      JR   Z,B2
609
610
611
612 005C CB C6      SET   0,(HL)
613 005E 3E 01      LD   A,1
614 0060 ED 79      OUT  (C),A
615 0062 3E 07      LD   A,7
616 0064 ED 79      OUT  (C),A
617 0066 C3 7F 00    C    JP   B2
618
619
620
621
622 0069 CB 78      B1:   BIT   7,B           Is this the end of bread detection?
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667 0080 78
668 0080 78
669 0081 1F
670 0082 1F
671 0083 E6 0E

```

SUBROUTINE NAME: BRK_SUB

DESCRIPTION: This purpose of this subroutine is to detect both the beginning of Break and the end of Break in the SIO. The general process of this routine is as follows:

IF START-OF-BREAK THEN (*BRK flag=0 and Break bit in SIO=1
BEGIN
BREAK FLAG:=1
TURN OFF RX INTERRUPT (*To prevent interrupt for null char
END
ELSE
IF END-OF-BREAK THEN (*BRK flag=1 and Break bit in SIO=0
BEGIN
BREAK FLAG:=0
ERROR RESET THE CARD (*In case SIO is programmed for odd parity - null causes parity error
SET BREAK BIT IN STATUS WORD ***assuming will get RX interrupt REINABLE RX INTERRUPT for the null char. when reinable
END

PARAMETERS:
#SIO - SIO_0_AC, SIO_0_BC, SIO_1_AC, SIO_0_BC passed in C reg
#BRK - RBRK_0, RBRK_1, RBRK_2, RBRK_3 passed in HL reg
#STAT - STAT_0, STAT_1, STAT_2, STAT_3 passed in DE reg

UPON EXIT - B-REGISTER CONTAINS CONTENTS OF SIO READ REGISTER #0

602 004E ED 40 BRK_SUB: IN B,(C) Get contents of SIO Read Reg. 0
603 0050 3E 10 LD A,10H
604 0052 ED 79 OUT (C),A
605 0054 CB 46 BIT 0,(HL)
606 0056 20 11 JR NZ,B1
607 0058 CB 78 BIT 7,B
608 005A 28 23 JR Z,B2
609
610
611
612 005C CB C6 SET 0,(HL)
613 005E 3E 01 LD A,1
614 0060 ED 79 OUT (C),A
615 0062 3E 07 LD A,7
616 0064 ED 79 OUT (C),A
617 0066 C3 7F 00 C JP B2
618
619
620
621
622 0069 CB 78 B1: BIT 7,B Is this the end of bread detection?

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 5

```

623 006B 20 12      JR   NZ,B2
624 006D CB 86      RES  0,(HL)
625 006F 3E 30      LD   A,30H
626 0071 ED 79      OUT  (C),A
627 0073 62
628 0074 6B
629 0075 CB E6      LD   H,D
630 0077 3E 01      LD   L,E
631 0079 ED 79      SET   4,(HL)
632 007B 3E 17      LD   A,1
633 007D ED 79      OUT  (C),A
634
635 007F C9          B2:   RET             ;Return to calling address
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667 0080 78
668 0080 78
669 0081 1F
670 0082 1F
671 0083 E6 0E

```

SUBROUTINE NAME: MDM3_SUB

DESCRIPTION: The purpose of this subroutine is to detect the status of the three modem input lines in SIO 0 channel A and see whether or not there has been a change in the lines (i.e. whether the lines are the same as recorded in the MODM_IN register). The three lines are the DCD, CTS, and SYNC lines which are used as modem lines RR, CS, and DN. If there has been a change in the lines, the MODM_IN register will be change accordingly and an interrupt will be sent to the host. The following is a pseudocode version of the following routine.

ROTATE CONTENTS OF READ REG. 0 RIGHT 2 BITS (*To align with MODM_IN ISOLATE DCD, SYNC, AND CTS BITS
IF BITS FROM READ REGISTER 0 ARE SAME AS CORRESPONDING BITS FROM MODM_IN, EXIT
ELSE
MODM_IN(BITS 1-3):=READ REG. 0 BITS
IF MODM_IN.AND.MODM MASK>0 THEN
SEND HOST MODEM INPUT LINE CHANGE INTERRUPT

UPON ENTRY: B REGISTER CONTAINS VALUE OF RRO IN SIO 0 CHA A
CALLED BY: EX_0 and INIT

```

667 0080 78
668 0080 78
669 0081 1F
670 0082 1F
671 0083 E6 0E

```

MDM3_SUB:
LD A,B
RRA
RRA
AND ESMASK1
Contents of SIO Read Reg. 0 to A-reg
Rotate right two bits positions for a
position match with MODM-IN register
Isolate DCD,SYNC, and CTS bits

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 6

```

672 0085 47 LD B,A Temporary hold in B-reg
673 0086 3A 00 00 E LD A,(MODM_IN) Get contents of MODM_IN register
674 0089 4F LD C,A Temporary hold in C-reg
675 008A E6 OE AND ESMSK1 Isolate RR,DM, &CS bits from MODM_IN
676 008C A8 XOR B
677 008D 28 1F JR Z,E2 No - go to exit

679 ;SET NEW BITS IN MODM_IN REGISTER
681 008F 57 LD D,A Save XOR'd value in D; holds changed lines
682 0090 79 LD A,C Get original contents of MODM_IN reg.
683 0091 E6 01 AND ESMSK2 Isolate bit 0 - IC line
684 0093 B0 OR B New modem line bits with old IC bit
685 0094 32 00 00 E LD (MODM_IN),A Save new value in MODM_IN
686 0097 7A LD A,D Get changed lines byte
687 0098 21 00 00 E LD HL,MODM_MASK Get address of Modem Mask
688 009B A6 AND (HL) Does the host want an interrupt on changes
689 009C 28 10 JR Z,E2 No - go to exit

691 ;SEND HOST A MODEM INPUT CHANGE INTERRUPT - FIRST GRAB SEMAPHORE
693 009E 3A 02 80 E1: LD A,(SEM_REG) Retrieve address of semaphore register
694 00A1 CB 7F BIT 7,A If bit 7=0, continue, else check again
695 00A3 C2 9E 00 C JP NZ,E1

697 ;SEM_REG SET - SEND INTERRUPT TO HOST AND RELEASE SEMAPHORE REGISTER
699 00A6 21 00 C0 LD HL,INT_COND Get address of INT_COND register
700 00A9 CB EE SET MOD_INT,(HL) Set bit for Modem Input line change
701 00B8 32 02 80 LD (SEM_REG),A Clear bit 7 of Sem. reg by writing to it

703 00AE C9 E2: RET

```

705
706
707
708
709
710
711
712
713
714
715
716

```

*****  
SUBROUTINE NAME: MODM1_SUB  
DESCRIPTION: The purpose of this subroutine is to determine whether  
or not the DCD line in SIO 0 channel B changed. This modem line  
represents the IC line. The SIO Read Register 0 is read to determine  
the current status of the DCD line. This bit is then compared with  
bit 0 of the MODM_IN register to see if there has been a change. If  
so, bit 0 in MODM_IN is set to reflect the change and the MODM_MASK  
register is checked to see if the host wants an interrupt. The  
following is a pseudocode version of the routine.

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 7

```

717
718 ;COMPARE DCD BIT WITH IC BIT IN MODM_IN REGISTER
719 IF NOT EQUAL THEN
720 BEGIN
721     CHANGE IC BIT IN MODM_IN REGISTER TO MATCH DCD BIT
722     RETRIEVE MODM MASK
723     IF MODM MASK (IC bit) SET THEN SEND HOST AN INPUT MODEM
724     LINE CHANGE INTERRUPT
725 END.
726
727 UPON ENTRY: B register contains value of SIO 0 cha B Read Reg. 0
728 CALLED BY: EX_1 and INIT
729
730 *****

731 ;*****  
MODM1_SUB:  
733 00AF LD A,(MODM_IN) Get contents of MODM_IN register
734 00B0 3A 00 00 E BIT 3,B B-reg has RRO contents - is DCD bit set?
735 00B2 CB 58 JR Z,EE1 NO
736 00B4 28 0C
737
738 00B6 CB 47 BIT IC_BIT,A Is IC bit in MODM_IN reg. set?
739 00B8 20 28 JR NZ,EE4 Yes - bits match - go to exit
740 00B9 CB C7 SET IC_BIT,A
741 00BC 32 00 00 E LD (MODM_IN),A Set IC bit in MODM_IN register
742 00BF C3 CB 00 JP EE2
743
744 00C2 CB 47 EE1: BIT IC_BIT,A Is IC bit in MODM_IN reg. set?
745 00C4 28 1C JR Z,EE4 No - bits match - go to exit
746 00C6 CB 87 RES IC_BIT,A
747 00C8 32 00 00 E LD (MODM_IN),A Reset IC bit in MODM_IN register
748
749 ;IF IC BIT IN MODM_MASK SET, SEND THE HOST AN INTERRUPT
750
751 00CB 21 00 00 E EE2: LD HL,MODM_MASK Get address of Modem Mask
752 00CE CB 46 BIT IC_BIT,(HL) Test the IC bit in Modem Mask
753 00D0 28 10 JR Z,EE4 If set, send interrupt, else jump to exit
754
755 ;SEND HOST A MODEM INPUT CHANGE INTERRUPT - FIRST GRAB SEMAPHORE
756
757 00D2 3A 02 80 EE3: LD A,(SEM_REG) Retrieve address of semaphore register
758 00D5 CB 7F BIT 7,A If bit 7=0, continue, else check again
759 00D7 C2 D2 00 C JP NZ,EE3
760
761 ;SEM_REG SET - SEND INTERRUPT TO HOST AND RELEASE SEMAPHORE REGISTER

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 8

763 00DA 21 00 C0

LD HL, INT_COND Get address of INT_COND register

764 00DD CB EE

SET MOD_INT(HL) Set bit for Modem Input line change

765 00DF 32 02 80

LD (SEM_REG),A Clear bit 7 of Sem. reg by writing to it

767 00E2 C9

EE4: RET

769 00E3

END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

```

1 SOURCE: &MXPTO
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL D10 MUX (FORDYCE) - HOST ISR - PORT SPECIFIC INTERRUPTS
5 FOR PORT 0
6
7 ****
8 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9 RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
10 REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11 THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12 ****
13
14 NAME MXPTO
15 COPY &MX4EQU$  

16 LIST S
17 ;
18 PUBLIC ISRPTO
19 ;
20 EXTRN TMPTAB,CONFIG_0,WR3_0,WR4_0,WRS_0,BD_0,TTAIL_0
21 EXTRN THEAD_0,TONO_BD_TAB,EEE2_SNDBRK,HSTCON,BITS_0
22
23 ****
24
25 DESCRIPTION: This file contains part of the Interrupt Service for a
26 host interrupt. In particular, it contains the routines for a port
27 specific interrupt for port 0. The first part of this file contains
28 the code which accesses TMPTAB, the four byte table which identifies
29 port specific interrupts. Each of the relevant bits in the TMPTAB
30 location is then checked (there can be more than one interrupt at a
31 time).
32
33 REGISTER USAGE: Reg. D - contains the TMPTAB bits
34 Reg. E - DO NOT USE. Reserved by Caller
35
36 CALLED BY (FILE): &MXHST
37
38 CALLS(FILE): &MXSBR
39
40 ****
41
42 CSEG
43
44 ISRPTO:
45 LD HL,TMPTAB
46 LD A,(HL)
47 RRA
48 LD D,A
49 JP NC,P1
50
51 No index for 0. Retrieve TMPTAB data
52 Rotate bit 0 into Carry flag
53 Save remaining bits in D
54 If bit 0=0,jump
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

```

489 ;CONFIGURATION DATA CHANGE INTERRUPT
490
491 ; The purpose of this routine is to reconfigure the line
492 ; characteristics of the UART and change the baud rate as desired
493 ; by the host. This is done by decoding the CONFIG register and
494 ; the BD registers. The CONFIG register contains the parity type,
495 ; number of stop bits, and number of bits per character. The BD
496 ; register is the index to the BD table which contains the CTC Channel
497 ; Control Word and prescale value for the baud rate requested.

499 ;DECIPHER CONFIGURATION REGISTER
501 0009 3A 00 00      LD A,(CONFIG_0)   Retrieve contents of configuration reg.
502 000C CD 00 00      CALL HSTCON
503 000F 32 00 00      LD (BITS_0),A   Save bits mask for RX data

505 ;LOAD SIO REGISTER 4 WITH APPROPRIATE BITS
507 0012 78            LD A,B          Obtain original pattern again
508 0013 E6 0F          AND OFH         Clear out upper byte
509 0015 47            LD B,A          Save pattern temporarily
510 0016 3A 00 00      LD A,(WR4_0)   Get copy of current WR4 value
511 0019 E6 F0          AND OFOH        Clear out lower byte
512 0018 B0            OR B           Merge new lower byte with orig. upper byte
513 001C 0E 71          LD C,SIO_0_AC
514 001E 2E 04          LD L4
515 0020 ED 69          OUT (C),L     Set SIO register pointer to WR4
516 0022 ED 79          OUT (C),H     Send out new WR4 value
517 0024 32 00 00      LD (WR4_0),A   Save copy of WR4

519 ;LOAD SIO REGISTER 5 WITH APPROPRIATE BITS
521 0027 7C            LD A,H          Retrieve original value of bits 4 & 5
522 0028 17            RLA             Move one position to left to match WRS
523 0029 47            LD B,A          Hold temporarily
524 002A 3A 00 00      LD A,(WR5_0)
525 002D E6 9F          AND 100111TIB   Clear out bit positions 5 & 6 in WRS
526 002F B0            OR B           Put new bits in WRS location
527 0030 2E 05          LD L5
528 0032 ED 69          OUT (C),L     Pointer to WRS
529 0034 ED 79          OUT (C),A     Pattern to WRS
530 0036 32 00 00      LD (WR5_0),A   Save new value of WRS

532 ;LOAD SIO REGISTER 3 WITH APPROPRIATE BITS
534 0039 78            LD A,B          Retrieve bits for # of bits per character

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

535 003A 17 RLA Rotate one position for WR3
 536 003B 47 LD B,A Save temporarily
 537 003C 3A 00 00 E LD A,(WR3_0) Get contents of WR3
 538 003F E6 3F AND 001111T1B Clear bits 6 & 7
 539 0041 B0 OR B Put new bits in WR3 pattern
 540 0042 32 00 00 E LD (WR3_0),A Store pattern
 541 0045 2E 03 LD L,3
 542 0047 ED 69 OUT (C),L Register pointer to WR3
 543 0049 ED 79 OUT (C),A Pattern to WR3

 545 ;CHANGE BAUD RATE - BD REG. IS INDEX TO BD_TAB
 547 004B 3A 00 00 E LD A,(BD_0) Get contents of BD reg - index to BD_TAB
 548 004E 87 ADD A,A Multiply by 2 (2 dimensional table)
 549 004F DE 02 SBC A,2 Subtract 2 for effective address
 550 0051 21 00 00 E LD HL,BD_TAB Get address of BD_TAB
 551 0054 4F LD C,A Trying to put A reg value into a
 552 0055 AF XOR A register pair
 553 0056 47 LD B,A Clear upper byte of BC reg. pair
 554 0057 09 ADD HL,BC Add index to base
 555 0058 7E LD A,(HL) Get CTC Channel Control Word from BD_TAB
 556 0059 D3 D0 OUT (CTC_0_CO),A Send to CTC channel*
 557 005B 23 INC HL Get second byte in table
 558 005C 7E LD A,(HL) Get CTC Time Constant value
 559 005D D3 D0 OUT (CTC_0_CO),A Send to CTC channel*

 561 005F 7A P1: LD A,D Retrieve remaining bit pattern from TMPTAB
 562 0060 1F RRA A,D Rotate bit 2 into Carry flag
 563 0061 57 LD D,A Save remaining bit in D
 564 0062 D2 8C 00 C JP NC,P2 If bit 2=0, jump

 566 ;TRANSMIT BUFFER NOT EMPTY INTERRUPT
 568 0065 HOSTTX TONO,THEAD_0,TTAIL_0,SIO_0_AD,TFIFO_0
 569 0065 3A 00 00 E +: LD A,(TONO) Get contents of Transmitter Flag
 570 0068 1F RRA A,TFIFO_0 Rotate bit 0 into Carry flag
 571 0069 38 21 JR C,?20001 If flag=1 jump
 573 ;TRANSMITTER IS OFF. CHECK IF HEAD = TAIL
 574 006B 3A 00 00 E +: LD A,(THEAD_0) Retrieve value in Head pointer index
 575 006E 47 LD B,A
 576 006F 3A 00 00 E +: LD A,(TTAIL_0) Retrieve value in Tail pointer index
 577 0072 B8 CP B Compare; if Head=Tail then buffer is empty
 578 0073 28 17 JR Z,?20001 If empty, jump to end
 581 +;

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

582 +;HEAD<>TAIL - THERE ARE CHARACTER TO RETRIEVE
 583 +;
 584 0075 26 C7 LD H,TX_BASE Retrieve the base upper byte of pointer
 585 0077 3A 00 00 E +: LD A,(THEAD_0) Get head pointer index
 586 007A C6 A0 ADD A,TFIFO_0 Add base of lower byte
 587 007C 6F LD L,A Save head pointer so can use A reg.
 588 007D 7E LD A,(HL) Retrieve character
 589 007E D3 70 OUT (SIO_0_AD),A Send character to UART
 590 0080 7D LD A,L Get lower byte again
 591 0081 3C INC A Isolate lower nibble(index)of pointer
 592 0082 E6 0F AND TMSK Save updated pointer (lower byte)
 593 0084 32 00 00 E +: LD (THEAD_0),A
 594 +;TURN ON TRANSMITTER FLAG
 595 +;
 597 0087 3E 01 LD A,1
 598 0089 32 00 00 E +: LD (TONO),A
 599 008F D2 9A 00 C ?20001: ENDM

 602 008C 7A P2: LD A,D Retrieve remaining bit pattern from TMPTAB
 603 008D 1F RRA A,D Rotate bit 1 into Carry flag
 604 008E 57 LD D,A Save remaining bit in D
 605 008F D2 9A 00 C JP NC,P3 If bit 1=0, jump

 607 ;SEND BREAK INTERRUPT
 609 0092 21 00 00 E LD HL,WR5_0 Parameters for SNDTRK subroutine
 610 0095 0E 71 LD C,SIO_0_AC
 611 0097 CD 00 00 E CALL SNDTRK

 613 009A C3 00 00 E P3: JP EEE2 Return to caller
 615 009D END

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 1

```
1 SOURCE: &MXPT1
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - HOST ISR - PORT SPECIFIC INTERRUPTS
5 FOR PORT 1
6 ****
7 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
8 * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED.
9 * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
10 * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
11 ****
12
13 NAME MXPT1
14 COPY &MX4EQU$  
LIST S
15
16 PUBLIC ISRPT1
17
18 EXTRN TMPTAB,CONFIG_1,WR3_1,WR4_1,WR5_1,BD_1,TTAIL_1
19 EXTRN THREAD_1,TON1,BD_TAB,EEE3,BITS_1,SNOBRK,HSTCON
20
21 ****
22 DESCRIPTION: This file contains part of the Interrupt Service for a
23 host interrupt. In particular, it contains the routines for a port
24 specific interrupt for port 1. The first part of this file contains
25 the code which accesses TMPTAB, the four byte table which identifies
26 port specific interrupts. Each of the relevant bits in the TMPTAB
27 location is then checked (there can be more than one interrupt at a
28 time).
29
30 REGISTER USAGE: Reg. D - contains the TMPTAB bits
31 Reg. E - DO NOT USE. Reserved by Caller
32
33 CALLED BY (FILE): &MXHST
34
35 CALLS(FILE): &MXSBR
36 ****
37
38 CSEG
39
40 ISRPT1:
41     LD   HL,TMPTAB
42     INC  HL
43     LD   A,(HL)
44     RRA
45     LD   D,A
46     JP   NC,P1
47
48     Get to correct byte in table for PT 1
49     No index for 0. Retrieve TMPTAB data
50     Rotate bit 0 into Carry flag
51     Save remaining bits in D
52     If bit 0=0,jump
```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

490 ;CONFIGURATION DATA CHANGE INTERRUPT
 491
 492 The purpose of this routine is to reconfigure the line
 493 characteristics of the UART and change the baud rate as desired
 494 by the host. This is done by decoding the CONFG register and
 495 the BD registers. The CONFG register contains the parity type,
 496 number of stop bits, and number of bits per character. The BD
 497 register is the index to the BD table which contains the CTC Channel
 498 Control Word and prescale value for the baud rate requested.

500 ;DECIPHER CONFIGURATION REGISTER

502 000A 3A 00 00 E LD A,(CONFG_1) Retrieve contents of configuration reg.
 503 000D CD 00 00 E CALL H\$TCON
 504 0010 32 00 00 E LD (BITS_1),A Save bits mask for RX data

506 ;LOAD SIO REGISTER 4 WITH APPROPRIATE BITS

508 0013 78 LD A,B Obtain original pattern again
 509 0014 E6 0F AND OFH Clear out upper byte
 510 0016 47 LD B,A Save pattern temporarily
 511 0017 3A 00 00 E LD A,(WR4_1) Get copy of current WR4 value
 512 001A E6 F0 AND OFOH Clear out lower byte
 513 001C B0 OR B Merge new lower byte with orig. upper byte
 514 001D 0E 73 LD C,SIO_0_BC
 515 001F 2E 04 LD L Set SIO register pointer to WR4
 516 0021 ED 69 OUT (C),L Send out new WR4 value
 517 0023 ED 79 OUT (C),A
 518 0025 32 00 00 E LD (WR4_1),A Save copy of WR4

520 ;LOAD SIO REGISTER 5 WITH APPROPRIATE BITS

522 0028 7C LD A,H Retrieve original value of bits 4 & 5
 523 0029 17 RLA Move one position to left to match WRS
 524 002A 47 LD B,A Hold temporarily
 525 002B 3A 00 00 E LD A,(WRS_1)
 526 002E E6 9F AND 10011111B Clear out bit positions 5 & 6 in WRS
 527 0030 B0 OR B Put new bits in WRS location
 528 0031 2E 05 LD L,5 Pointer to WRS
 529 0033 ED 69 OUT (C),L Pattern to WRS
 530 0035 ED 79 OUT (C),A
 531 0037 32 00 00 E LD (WRS_1),A Save new value of WRS

533 ;LOAD SIO REGISTER 3 WITH APPROPRIATE BITS

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

535 003A 78 LD A,B Retrieve bits for # of bits per character
 536 003B 17 RLA Rotate one position for WR3
 537 003C 47 LD B,A Save temporarily
 538 003D 3A 00 00 E LD A,(WR3_1)
 539 0040 E6 3F AND 00111111B Get contents of WR3
 540 0042 B0 OR B Clear bits 6 & 7
 541 0043 32 00 00 E LD (WR3_1),A Put new bits in WR3 pattern
 542 0046 2E 03 LD L,3 Store pattern
 543 0048 ED 69 OUT (C),L Register pointer to WR3
 544 004A ED 79 OUT (C),A Pattern to WR3

546 ;CHANGE BAUD RATE - BD REG. IS INDEX TO BD_TAB

548 004C 3A 00 00 E LD A,(BD_1) Get contents of BD reg - index to BD_TAB
 549 004F 87 ADD A,A Multiply by 2 [2 dimensional table]
 550 0050 DE 02 SBC A,2 Subtract 2 for effective address
 551 0052 21 00 00 E LD HL, BD_TAB Get address of BD_TAB
 552 0055 4F LD C,A Trying to put A reg value into a
 553 0056 AF XOR A,0 register pair
 554 0057 47 LD B,A Clear upper byte of BC reg. pair
 555 0058 09 ADD HL,BC Add index to base
 556 0059 7E LD A,(HL) Get CTC Channel Control Word from BD_TAB
 557 005A D3 D1 OUT (CTC_0_C1),A Send to CTC channel^x
 558 005C 23 INC HL Get second byte in table
 559 005D 7E LD A,(HL) Get CTC Time Constant value
 560 005E D3 D1 OUT (CTC_0_C1),A Send to CTC channel^x
 561 0060 7A P1: LD A,D Retrieve remaining bit pattern from TMPTAB
 562 0061 1F RRA Rotate bit 2 into Carry flag
 563 0062 57 LD D,A Save remaining bit in D
 564 0063 D2 8D 00 C JP NC,P2 If bit 2=0,jump

566 ;TRANSMIT BUFFER NOT EMPTY

568 0066 HOSTTX TON1,THEAD_1,TTAIL_1,SIO_0_BD,TFIFO_1
 569 +: E LD A,(TON1) Get contents of Transmitter Flag
 570 0066 3A 00 00 E JR C,?20001 Rotate bit 0 into Carry flag
 571 0069 1F +: JR C,?20001 If flag=1 jump
 572 006A 38 21 +:
 573 +:
 574 +:
 575 +:
 576 006C 3A 00 00 E LD A,(THEAD_1) Retrieve value in Head pointer index
 577 006F 47 +: LD B,A Retrieve value in Tail pointer index
 578 0070 3A 00 00 E CP B Compare; if Head=Tail then buffer is empty
 579 0073 B8 +: JR Z,?20001 If empty, jump to end
 580 0074 28 17 +:
 581 +:
 582 +:
 583 +:
 584 0076 26 C7 LD H, TX_BASE Retrieve the base upper byte of pointer

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

585 0078 3A 00 00 E + LD A,(THEAD_1) Get head pointer index
586 007B C6 90 + ADD A,T FIFO_I Add base of lower byte
587 007D 6F + LD L,A Save head pointer so can use A reg.
588 007E 7E + LD A,(HL) Retrieve character
589 007F D3 72 + OUT (S10_0_BD),A Send character to UART
590 0081 7D + LD A,L Get lower byte again
591 0082 3C INC A Increment lower byte of head pointer
592 0083 E6 0F AND TMSK Isolate lower nibble(index)of pointer
593 0085 32 00 00 E + LD (THEAD_1),A Save updated pointer (lower byte)
594 +;
595 +;TURN ON TRANSMITTER FLAG
596 +;
597 0088 3E 01 + LD A,1
598 008A 32 00 00 E + LD (TON1),A
599 +;
600 ??0001: ENDM
602 008D 7A P2: LD A,D Retrieve remaining bit pattern from TMPTAB
603 008E 1F RRA LD A,D
604 008F 57 LD D,A Rotate bit 1 into Carry flag
605 0090 D2 98 00 C JP NC,P3 If bit=0,jump
607 ;SEND BREAK INTERRUPT
609 0093 21 00 00 E LD HL,WRS_1 Parameters for SND BRK subroutine
610 0096 0E 73 LD C,S10_0_BC
611 0098 CD 00 00 E CALL SND BRK
613 0098 C3 00 00 E P3: JP EEE3 Return to caller
615 009E END

ASSEMBLER ERRORS = 0

```

1 SOURCE: &MXPT2
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - HOST ISR - PORT SPECIFIC INTERRUPTS
5 FOR PORT 2
6
7 ****
8 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9 RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
10 REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11 THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12 ****
13
14 NAME MXPT2
15 COPY &MX4EQU$S
16 LIST S
17
18 ; PUBLIC ISRPT2
19
20 ; EXTRN TMPTAB,CONFIG_2,WR3_2,WR4_2,WR5_2,BD_2,TTAIL_2
21 ; EXTRN THED_2,TON2,BD_TAB,EEE4,BITS_2,SNDBRK,HSTCON
22
23 ****
24 ;DESCRIPTION: This file contains part of the Interrupt Service for a
25 host interrupt. In particular, it contains the routines for a port
26 specific interrupt for port 2. The first part of this file contains
27 the code which accesses TMPTAB, the four byte table which identifies
28 port specific interrupts. Each of the relevant bits in the TMPTAB
29 location is then checked (there can be more than one interrupt at a
30 time).
31
32 ;REGISTER USAGE: Reg. D - contains the TMPTAB bits
33 ;Reg. E - DO NOT USE. Reserved by Caller
34
35 ;CALLED BY (FILE): &MXHST
36
37 ;CALLS(FILE): &MXSBR
38 ****
39
40 CSEG
41
42 0000 21 00 00 E ISRPT2:
43 0003 23 LD HL,TMPTAB
44 0004 23 INC HL
45 0005 7E INC HL Get to correct byte in table for PT 1
46 0006 1F LD A,(HL) No index for 0. Retrieve TMPTAB data
47 0007 57 RRA Rotate bit 0 into Carry flag
48 0008 D2 61 00 C LD D,A Save remaining bits in D
49 0009 D2 61 00 C JP NC,P1 If bit 0=0,jump

```

```

491
492
493
494
495
496
497
498
499 ;CONFIGURATION DATA CHANGE INTERRUPT
500
501
502 ;DECIPHER CONFIGURATION REGISTER
503 0008 39 00 00 E LD A,(CONFIG_2) Retrieve contents of configuration reg.
504 000E CD 00 00 E CALL HSTCON
505 0011 32 00 00 E LD (BITS_2),A Save bits mask data for RX characters
506
507 ;LOAD SIO REGISTER 4 WITH APPROPRIATE BITS
508 0014 78 LD A,B Obtain original pattern again
509 0015 E6 0F AND 0FH Clear out upper byte
510 0017 47 LD B,A Save pattern temporarily
511 0018 3A 00 00 E LD A,(WR4_2) Get copy of current WR4 value
512 0018 E6 F0 AND 0FOH Clear out lower byte
513 001D BD OR B Merge new lower byte with orig. upper byte
514 001E 0E B1 LD C,SIO_1_AC
515 0020 2E 04 LD L,4
516 0022 ED 69 OUT (C),L Set SIO register pointer to WR4
517 0024 ED 79 OUT (C),A Send out new WR4 value
518 0026 32 00 00 E LD (WR4_2),A Save copy of WR4
519
520 ;LOAD SIO REGISTER 5 WITH APPROPRIATE BITS
521
522 0029 7C LD A,H Retrieve original value of bits 4 & 5
523 002A 17 RLA Move one position to left to match WRS
524 002B 47 LD B,A Hold temporarily
525 002C 39 00 00 E LD A,(WRS_2)
526 002C E6 9F AND 1001111B Clear out bit positions 5 & 6 in WRS
527 0031 BD OR B Put new bits in WRS location
528 0032 2E 05 LD L,5
529 0034 ED 69 OUT (C),L Pointer to WRS
530 0036 ED 79 OUT (C),A Pattern to WRS
531 0038 32 00 00 E LD (WRS_2),A Save new value of WRS
532
533 ;LOAD SIO REGISTER 3 WITH APPROPRIATE BITS
534

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

```

536 003B 78
537 003C 17
538 003D 47
539 003E 3A 00 00 E LD A,B Retrieve bits for # of bits per character
540 0041 E6 3F RLA Rotate one position for WR3
541 0043 B0 LD B,A Save temporarily
542 0044 32 00 00 E LD A,(WR3_2) Get contents of WR3
543 0047 2E 03 AND 0011111B Clear bits 6 & 7
544 0049 ED 69 OR B Put new bits in WR3 pattern
545 004B ED 79 LD (WR3_2),A Store pattern
547 ;CHANGE BAUD RATE - BD REG. IS INDEX TO BD_TAB
549 004D 3A 00 00 E LD A,(BD_2) Get contents of BD reg - index to BD_TAB
550 0050 87 ADD A,A Multiply by 2 (2 dimensional table)
551 0051 DE 02 SBC A,'2 Subtract 2 for effective address
552 0053 21 00 00 E LD H, BD_TAB Get address of BD_TAB
553 0056 4F LD C,A Trying to put A reg value into a
554 0057 AF XOR A register pair
555 0058 47 LD B,A Clear upper byte of BC reg. pair
556 0059 09 ADD HL,BC Add index to base
557 005A 7E LD A,(HL) Get CTC Channel Control Word from BD_TAB
558 005B D3 E0 OUT (CTC_1_CO),A Send to CTC channel*
559 005D 23 INC HL Get second byte in table
560 005E 7E LD A,(HL) Get CTC Time Constant value
561 005F D3 E0 OUT (CTC_1_CO),A Send to CTC channel*
563 0061 7A P1: LD A,D Retrieve remaining bit pattern from TMPTAB
564 0062 1F RRA LD D,A Rotate bit 2 into Carry flag
565 0063 57 LD D,A Save remaining bit in D
566 0064 D2 8E 00 C JP NC,P2 If bit 2=0, jump
568 ;TRANSMIT BUFFER NOT EMPTY INTERRUPT
570 0067 HOSTTX TON2,THEAD_2,TTAIL_2,SIO_1_AD,TFIFO_2
571 0067 3A 00 00 E +; LD A,(TON2) Get contents of Transmitter Flag
573 006A 1F RRA LD A,TFIFO_2 Rotate bit 0 into Carry flag
574 006B 38 21 JR C,?20001 If flag=1 jump
575 +; ;TRANSMITTER IS OFF. CHECK IF HEAD = TAIL
577 +; 578 006D 3A 00 00 E +; LD A,(THEAD_2) Retrieve value in Head pointer index
579 0070 47 +; LD B,A
580 0071 3A 00 00 E +; LD A,(TTAIL_2) Retrieve value in Tail pointer index
581 0074 B8 +; CP B Compare; if Head=Tail then buffer is empty
582 0075 28 17 +; JR Z,?20001 If empty, jump to end
583 +;

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

```

584 +;HEAD<->TAIL - THERE ARE CHARACTER TO RETRIEVE
585 +;
586 0077 26 C7 E LD H,TX_BASE Retrieve the base upper byte of pointer
587 0079 3A 00 00 E LD A,(THEAD_2) Get head pointer index
588 007C C6 80 +; ADD A,TFIFO_2 Add base of lower byte
589 007E 6F +; LD L,A Save head pointer so can use A reg.
590 007F 7E +; LD A,(HL) Retrieve character
591 0080 D3 B0 +; OUT (SIO_1_AD),A Send character to UART
592 0082 7D +; LD A,L Get lower byte again
593 0083 3C +; INC A Iscrement lower byte of head pointer
594 0084 E6 0F +; AND TMSK Isolate lower nibble(index)of pointer
595 0086 32 00 00 E LD (THEAD_2),A Save updated pointer (lower byte)
596 +;TURN ON TRANSMITTER FLAG
597 +;
598 599 0089 3E 01 E LD A,1
600 008B 32 00 00 E LD (TON2),A
601 602 ?20001: ENDM
604 008E 7A P2: LD A,D Retrieve remaining bit pattern from TMPTAB
605 008F 1F RRA LD D,A Rotate bit 1 into Carry flag
606 0090 57 LD D,A Save remaining bit in D
607 0091 D2 9C 00 C JP NC,P3 If bit 1=0, jump
609 ;SEND BREAK INTERRUPT
611 0094 21 00 00 E LD HL,WR5_2 Parameters for SNDBRK subroutine
612 0097 0E B1 LD C,SIO_1_AC
613 0099 CD 00 00 E CALL SNDBRK
615 009C C3 00 00 E P3: JP EEE4 Return to caller
617 009F END

```

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.00MR

PAGE 1

```
1      SOURCE: &MXPT3
2      PROGRAMMER: LIZ POTEET
3
4      4 CHANNEL DIO MUX (FORDYCE) -- HOST ISR - PORT SPECIFIC INTERRUPTS
5          IFOR PORT 3
6
7      ****
8      * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9      * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
10     * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11     * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12     ****
13
14     NAME MXPT3
15     COPY &MX4EQU
16     LIST S
17
18     PUBLIC ISRPT3
19
20     EXTRN TMPTAB,CONFIG_3,WR3_3,WR4_3,WR5_3,BD_3,TTAIL_3
21     EXTRN THREAD_3,TONG3,BD_TAB,EEES,BITS_3,SNOBRK,NSTCON
22
23     ****
24     DESCRIPTION: This file contains part of the Interrupt Service for a
25     host interrupt. In particular, it contains the routines for a port
26     specific interrupt for port 3. The first part of this file contains
27     the code which accesses TMPTAB, the four byte table which identifies
28     port specific interrupts. Each of the relevant bits in the TMPTAB
29     location is then checked (there can be more than one interrupt at a
30     time).
31
32     REGISTER USAGE: Reg. D - contains the TMPTAB bits
33             Reg. E - DO NOT USE. Reserved by Caller
34
35     CALLED BY (FILE): &MXHST
36
37     CALLS(FILE): &MDXSBR
38
39     ****
40
41     CSEG
42
43     0000      ISRPT3: LD   HL, TMPTAB
44     0000 21 00 00    E   INC  HL
45     0003 23          INC  HL
46     0004 23          INC  HL
47     0005 23          INC  HL      Get to correct byte in table for PT 1
48     0006 7E          LD   A,(HL)  No index for 0. Retrieve TMPTAB data
49     0007 1F          RRA   A
50     0008 57          LD   D,A      Rotate bit 0 into Carry flag
51                           Save remaining bits in D
```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 2

490 0009 D2 62 00 C JP NC,P1 If bit 0=0,jump

492 ;CONFIGURATION DATA CHANGE INTERRUPT
 493 ; The purpose of this routine is to reconfigure the line
 494 ; characteristics of the UART and change the baud rate as desired
 495 ; by the host. This is done by decoding the CONFG register and
 496 ; the BD registers. The CONFG register contains the parity type,
 497 ; number of stop bits, and number of bits per character. The BD
 498 ; register is the index to the BD table which contains the CTC Channel
 499 ; Control Word and prescale value for the baud rate requested.
 500

502 ;DECIPHER CONFIGURATION REGISTER

504 000C 3A 00 00 E LD A,(CONFG_3) Retrieve contents of configuration reg.
 505 000F CD 00 00 E CALL HSTCON
 506 0012 32 00 00 E LD (BITS_3),A Save the bits mask data for RX characters

508 ;LOAD SIO REGISTER 4 WITH APPROPRIATE BITS

510 0015 78 LD A,B Obtain original pattern again
 511 0016 E6 0F AND OFH Clear out upper byte
 512 0018 47 LD B,A Save pattern temporarily
 513 0019 3A 00 00 E LD A,(WR4_3)
 514 001C E6 F0 AND OFOH Get copy of current WR4 value
 515 001E B0 OR B Clear out lower byte
 516 001F 0E B3 LD C,SIO_1_BC Merge new lower byte with orig. upper byte
 517 0021 2E 04 LD L4
 518 0023 ED 69 OUT (C),L Set SIO register pointer to WR4
 519 0025 ED 79 OUT (C),A Send out new WR4 value
 520 0027 32 00 00 E LD (WR4_3),A Save copy of WR4

522 ;LOAD SIO REGISTER 5 WITH APPROPRIATE BITS

524 002A 7C LD A,H Retrieve original value of bits 4 & 5
 525 002B 17 RLA Move one position to left to match WRS
 526 002C 47 LD B,A Hold temporarily
 527 002D 3A 00 00 E LD A,(WR5_3)
 528 0030 E6 9F AND 1001111B Clear out bit positions 5 & 6 in WRS
 529 0032 B0 OR B Put new bits in WRS location
 530 0033 2E 05 LD L5
 531 0035 ED 69 OUT (C),L Pointer to WRS
 532 0037 ED 79 OUT (C),A Pattern to WRS
 533 0039 32 00 00 E LD (WR5_3),A Save new value of WRS

535 ;LOAD SIO REGISTER 3 WITH APPROPRIATE BITS

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0.MR

PAGE 3

537 003C 78 LD A,B Retrieve bits for # of bits per character
 538 003D 17 RLA Rotate one position for WR3
 539 003E 47 LD B,A Save temporarily
 540 003F 3A 00 00 E LD A,(WR3_3)
 541 0042 E6 3F AND 0011111B Get contents of WR3
 542 0044 B0 OR B Clear bits 6 & 7
 543 0045 32 00 00 E LD (WR3_3),A Put new bits in WR3 pattern
 544 0048 2E 03 LD L3 Store pattern
 545 004A ED 69 OUT (C),L Register pointer to WR3
 546 004C ED 79 OUT (C),A Pattern to WR3

548 ;CHANGE BAUD RATE - BD REG. IS INDEX TO BD_TAB

550 004E 3A 00 00 E LD A,(BD_3) Get contents of BD reg - index to BD_TAB
 551 0051 87 ADD A,A Multiply by 2 (2 dimensional table)
 552 0052 DE 02 SBC A,2 Subtract 2 for effective address
 553 0054 21 00 00 E LD HL, BD_TAB Get address of BD_TAB
 554 0057 4F LD C,A Trying to put A reg value into a
 555 0058 AF XOR A register pair
 556 0059 47 LD B,A Clear upper byte of BC reg. pair
 557 005A 09 ADD HL,BC Add index to base
 558 005B 7E LD A,(HL) Get CTC Channel Control Word from BD_TAB
 559 005C D3 E1 OUT (CTC_1_C1),A Send to CTC channel
 560 005E 23 INC HL Get second byte in table
 561 005F 7E LD A,(HL) Get CTC Time Constant value
 562 0060 D3 E1 OUT (CTC_1_C1),A Send to CTC channel

564 0062 7A P1: LD A,D Retrieve remaining bit pattern from TMPTAB
 565 0063 1F RRA Rotate bit 2 into Carry flag
 566 0064 57 LD D,A Save remaining bit in D
 567 0065 D2 8F 00 C JP NC,P2 If bit 2=0,jump

569 ;TRANSMIT BUFFER NOT EMPTY INTERRUPT

571 0068 HOSTTX TON3,THEAD_3,TTAIL_3,SIO_1_BD,TFIFO_3

572 0068 3A 00 00 E +: LD A,(TON3) Get contents of Transmitter Flag
 573 0068 1F +: RRA Rotate bit 0 into Carry flag
 574 0068 38 21 +: JR C,?0001 If flag=1 jump

576 006C 38 21 +: ;TRANSMITTER IS OFF. CHECK IF HEAD = TAIL

577 006E 3A 00 00 E +: LD A,(THEAD_3) Retrieve value in Head pointer index
 578 0071 47 +: LD B,A
 580 0072 3A 00 00 E +: LD A,(TTAIL_3) Retrieve value in Tail pointer index
 581 0075 B8 +: CP B Compare; if Head=Tail then buffer is empty
 583 0076 28 17 +: JR Z,?0001 If empty, jump to end

584 0076 28 17 +:

585 ;HEAD<>TAIL - THERE ARE CHARACTER TO RETRIEVE
586
587 0078 26 C7 E + LD H,TX_BASE Retrieve the base upper byte of pointer
588 007A 3A 00 00 + LD A,(THEAD_3) Get head pointer index
589 007D C6 70 ADD A, FIFO_3 Add base of lower byte
590 007F 6F + LD L,A Save head pointer so can use A reg.
591 0080 7E + LD A,(HL) Retrieve character
592 0081 D3 B2 OUT (\$IO_1_BD),A Send character to UART
593 0083 7D + LD A,L Get lower byte again
594 0084 3C INC A Increment lower byte of head pointer
595 0085 E6 0F AND TMSK Isolate lower nibble(index)of pointer
596 0087 32 00 00 E + LD (THEAD_3),A Save updated pointer (lower byte)
597
598 ;TURN ON TRANSMITTER FLAG
599
600 008A 3E 01 E + LD A,1
601 008C 32 00 00 + LD (f0N3),A
602 +?
603 ?0001: ENDM
605 008F 7A P2: LD A,D Retrieve remaining bit pattern from TMPTAB
606 0090 1F RRA Rotate bit 1 into Carry flag
607 0091 57 LD D,A Save remaining bit in D
608 0092 D2 9D 00 C JP NC,P3 If bit i=0,jump
610 ;SEND BREAK INTERRUPT
612 0095 21 00 00 E LD HL,WR5_3 Parameters for SNDLK subroutine
613 0098 0E B3 LD C,\$IO_T_BC
614 009A CD 00 00 E CALL SNDLK
616 009D C3 00 00 E P3: JP EEE5 Return to caller
618 00A0 END

ASSEMBLER ERRORS = 0

```

1 SOURCE: &MXSBR
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - SUBROUTINES FOR FILES &MXPT0,&MXPT1,
5 &MXPT2,&MXPT3
6
7 ****
8 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9 * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
10 * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11 * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12 ****
13
14 NAME MXSBR
15 COPY &MX4EQU8
16 LIST S
17
18 PUBLIC HSTCON,SNDBRK
19
20 ****
21
22 DESCRIPTION: This file contains two subroutines, HSTCON and SNDBRK
23 which are used by the port specific interrupt routines. HSTCON is
24 part of the Configuration Change interrupt service routine and
25 SNDBRK is part of the Send Break interrupt service routine.
26
27 HISTORY:
28
29 4/3/86 - BUG FIX: Did have 1 variable, BITS_MSK which was set in
30 this routine and used to mask of bits in the RX data depending on
31 what the bits per character was set to in the CONFIG register.
32 Problem was that there was only 1 variable but four ports, so
33 whatever the most recent BITS_MSK value was, masked RX characters
34 on all 4 ports. Changed HSTCON to pass back the bits/char value
35 to the caller in the C register.
36
37 ****
38
39 CSEG
40
41 ****
42 ;SUBROUTINE NAME: HSTCON
43
44 ;DESCRIPTION: This subroutine is the first part of the processing
45 ;of a port specific configuration data change interrupt from the
46 ;host. This routine basically changes the order of the bits read
47 ;from the CONFG register to the corresponding bit patterns needed
48 ;to program the SIO write registers. The receive mask is also
49 ;identified and programmed based on the number of receive bits
50 ;per character requested by the change.
51

```

```

490 ;UPON ENTRY: A REG. CONTAINS CONFG REG. VALUE
491 D REG - USED BY CALLING ROUTINE - DO NOT ACCESS
492 E REG - USED BY CALLING ROUTINE - DO NOT ACCESS
493
494 ;UPON EXIT: A REG - CONTAINS RX BIT MASK VALUE
495 B REG - CONTAINS CONFIG. DATA - LOWER NIBBLE
496 C REG - CONTAINS A REG DATA + BITS/CHAR IN FORMAT
497 FOR SIO REGISTERS
498
499 ;CALLED BY(FILES): &MXPT0, &MXPT1, &MXPT2, &MXPT3
500
501
503 ;DECIPHER CONFIGURATION REGISTER
504
505 0000 CB 4F      HSTCON: BIT 1,A      Testing for parity check type
506 0002 CA 07 00      JP Z,HC1      If bit=0, parity<>even - no change needed
507 0005 CB C7      SET 0,A      Parity even. Pattern must match SIO WR4
508
509 ;CHANGE STOP BITS PATTERN - ADD 1 TO VALUE IN BITS 2&3
510
511 0007 1F      HC1: RRA      Rotate two bits right so can add the 1
512 0008 1F      RRA      Rotate two bits right so can add the 1
513 0009 3C      INC A      Rotate bits back in original positions
514 000A 17      RLA      Rotate bits back in original positions
515 000B 17      RLA
516
517 ;CHANGE BITS-PER-CHAR. PATTERN TO MATCH SIO WR3 & WR5 - SWAP BITS 4 & 5
518
519 000C 47      LD B,A      Save current value temporarily in B reg
520 000D E6 10      AND 10H      Isolate bit 4
521 000F 17      RLA      Move to bit 5 position
522 0010 67      LD H,A      Save temporarily
523 0011 78      LD A,B      Retrieve other pattern
524 0012 E6 20      AND 20H      Isolate bit 5
525 0014 1F      RRA      Move to the Bit 4 position
526 0015 B4      OR H      'Or' both bits together - now swapped
527 0016 67      LD H,A      Save swapped bits temp.
528
529 ;DETERMINE NUMBER OF BITS PER CHARACTER AND PASS VALUE BACK TO
530 ;CALLING ROUTINE IN THE A REGISTER.
531
532 0017 FE 30      CP 30H      Eight bits per character?
533 0019 20 03      JR NZ,CON1      No
534 0018 3E FF      LD A,OFFH      Mask value for eight bit character

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

```

535 001D C9           RET

537 001E FE 10        CON1: CP 10H    Seven bits per character?
538 0020 20 03        JR NZ,CON2   No
539 0022 3E 7F        LD A,7FH   Mask value for seven bit character
540 0024 C9           RET

542 0025 FE 20        CON2: CP 20H    Six bits per character?
543 0027 C2 20 00      JR NZ,CON3   No
544 002A 3E 3F        LD A,3FH   Mask value for six bit character
545 002C C9           RET

547 002D 3E 1F        CON3: LD A,1FH   Assume here five bits per character
548 002F C9           RET

```

```

550 ****
551
552
553
554 SUBROUTINE NAME: SNDBRK
555
556 DESCRIPTION: This routine is used when the host sends the card a Send
557 Break interrupt. A break interrupt can be either notifying the card
558 to begin or to end a break. The card determines whether to begin
559 or end break by checking the BREAK bit in the WRS variable. If this
560 bit (4) = 0 then this is the Start of Break. If bit 4=1 then this
561 is the end of break and this routine will turn off the Break bit
562 in SIO WRS. The following is a description of the process.
563
564 IF BREAK bit=0 THEN          (*Is this the beg. of break?
565 BEGIN
566     SET WRS BIT 4 IN WRS VARIABLE (*Yes
567     SEND NEW WRS VALUE TO REAL SIO WRS
568 END
569 ELSE                         (*This is end of Break
570 BEGIN
571     RESET WRS BIT 4 IN WRS VARIABLE
572     SEND NEW WRS VALUE TO REAL SIO WRS
573
574 PARAMETERS & REGISTER RESTRICTION:
575 Reg. HL : WRS_0,WRS_1,WRS_2,WRS_3
576 Reg. C : SIO_0_AC,SIO_0_BC,SIO_1_AC,SIO_1_BC
577 Reg. D : Used In Calling routine - DO NOT ALTER
578 Reg. E : Used In Calling routine - DO NOT ALTER
579
580 SUBROUTINE CALLED BY (FILES):
581     ISRP0,ISRP1,ISRP2,ISRP3

```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 4

```

581 ****

583 0030 7E           SNDBRK: LD A,(HL)   Get contents of WRS value
584 0031 CB 67         BIT 4,A      Test Break bit
585 0033 20 00         JR NZ,SB1   If bit=1, end of break - jump
586
587 START BREAK - TURN ON BREAK BIT IN WRS AND SEND TO SIO
588
589 0035 3E 05         LD A,5       Register pointer to WRS
590 0037 ED 79         OUT (C),A   Get contents of WRS location
591 0039 7E             LD A,(HL)   Set the Break bit
592 003A CB E7         SET 4,A     Send amended WRS copy to SIO WRS
593 003C ED 79         OUT (C),A   Update stored WRS copy
594 003E 77             LD (HL),A
595 003F C3 4C 00         JP SB2
596
597 END BREAK -TURN OFF BREAK BIT IN WRS AND SEND TO SIO
598
599 0042 3E 05         SB1: LD A,5      Set register pointer to WRS
600 0044 ED 79         OUT (C),A   Get contents of stored WRS copy
601 0046 7E             LD A,(HL)   WRS bit 4 set to 0
602 0047 CB A7         RES 4,A    WRS value to SIO
603 0049 ED 79         OUT (C),A   Update stored WRS copy
604 004B 77             LD (HL),A
605
606 004C C9             SB2: RET      END
607 004D

```

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

```
1      SOURCE: &MXMOD
2      PROGRAMMER: LIZ POTEET
3
4      4 CHANNEL DIO MUX (FORDYCE) - HOST ISR - MODEM OUTPUT CHANGE
5          INTERRUPT.
6
7      ****
8      * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
9      * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
10     * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
11     * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
12     ****
13
14     LIST X
15     NAME MXMOD
16     COPY &MX4EQU
17     LIST S
18
19     PUBLIC MODOUT
20
21     EXTRN WRS_0,WRS_1,MODM_OUT,EEE6
22
23     ****
24     DESCRIPTION:
25         This routine is basically a subroutine called by &MXHST when
26         a Modem Output Change is sent by the host. The purpose of this
27         routine is to set the modem output lines to match the bits pattern
28         in the MODM_OUT register. The procedure is then as follows:
29         The MODM_OUT register is fetched and rotated right 1 bits so that
30         bit 0 is in the Carry flag. If bit 0=1, the RS line will be set
31         in the SIO (SIO #0 CH A RTS line). If bit 0=0, the RS line will be
32         cleared. The same sequence is performed on bits 1 & 2. Bit 1 will
33         determine whether the TR line (SIO #0 CH A DTR line) will be cleared
34         or set. Bit 2 will determine whether the SR line (SIO #1 CH A RTS
35         line). When all three bits are finished, the ISR will jump back to
36         a label in &MXEXT.
37
38     NOTE: This routine does not determine which bits in the MOD_OUT
39     register have changed or not. Consequently, even though only one
40     bit may have changed, this routine will affect all of the modem
41     output lines, i.e. a line may be cleared which is already clear.
42
43     REGISTER RESTRICTIONS:
44         E - Used in the Calling routine - Do Not Use Here
45
46     CALLED BY (FILE): &MXHST
47
48     ****
```

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

491	0000	3A	00	00	E	MODOUT:	LD RRA A,(MODM_OUT)	Get the contents of the Modm Out register	
492	0003	1F					JR NC,M1	Bit 0 (RS) to Carry	
493	0004	30	09					If RS bit<>1, jump	
495 ;SET RS OUTPUT LINE									
497	0006	47			E	LD B,A	Save value in B reg. temporarily		
498	0007	3A	00	00		LD A,(WRS_0)	Retrieve contents of WRS copy		
499	000A	CB	CF		C	SET 1,A	Set RTS bit in WRS		
500	000C	C3	15	00		JP M2			
502	000F	47			E	M1:	LD B,A	Save value temporarily in B	
503	0010	3A	00	00		LD A,(WRS_0)	Retrieve contents of WRS copy		
504	0013	CB	8F			RES 1,A	Reset RTS bit in WRS		
506	0015	0E	71		E	M2:	LD C,SIO_0_AC	Get address of SIO channel	
507	0017	16	05			LD D,S			
508	0019	ED	51			OUT (C),D	Register pointer to WRS		
509	001B	ED	79			OUT (C),A	Send new WRS value to SIO		
510	001D	32	00	00	E	LD (WRS_0),A	Save copy of WRS		
511	0020	78				LD A,B	Get rotated copy of MODM reg for next chk		
513 ;CHECK TR BIT									
515	0021	1F				RRA JR NC,M3	Rotate Bit 1 (TR) into Carry		
516	0022	30	09				If Tr bit <> 1, jump		
518 ;SET TR LINE									
520	0024	47			E	LD B,A	Save value in B reg temporarily		
521	0025	3A	00	00		LD A,(WRS_0)	Get saved copy of WRS		
522	0028	CB	FF		C	SET 7,A	Set DTR bit in WRS copy		
523	002A	C3	33	00		JP M4			
525	002D	47			E	M3:	LD B,A	Save value in B reg temporarily	
526	002E	3A	00	00		LD A,(WRS_0)	Get saved copy of WRS		
527	0031	CB	BF			RES 7,A	Reset DTR bit in WRS copy		
529	0033	0E	71		E	M4:	LD C,SIO_0_AC	Get address of SIO register	
530	0035	16	05			LD D,S			
531	0037	ED	51			OUT (C),D	Register pointer to WRS		
532	0039	ED	79			OUT (C),A	Send new WRS value to SIO		
533	003B	32	00	00	E	LD (WRS_0),A	Store updated copy of WRS		

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 3

534	003E	78				LD A,B			
536 ;CHECK SR BIT									
538	003F	1F				RRA JR NC,MS	Rotate Bit 3 (SR) into Carry		
539	0040	30	08				If not set, go to exit		
541 ;SET SR LINE									
543	0042	3A	00	00	E	LD A,(WRS_1)	Get saved copy of WRS		
544	0045	CB	CF			SET 1,A	Set RTS bit in WRS		
545	0047	C3	4F	00	C	JP M6			
547	004A	3A	00	00	E	M5:	LD A,(WRS_1)	Get saved copy of WRS	
548	004D	CB	8F			RES 1,A	Reset RTS bit in WRS		
550	004F	0E	73		E	M6:	LD C,SIO_0_BC		
551	0051	16	05			LD D,S			
552	0053	ED	51			OUT (C),D	Register pointer to WRS		
553	0055	ED	79			OUT (C),A	Send WRS value to SIO		
554	0057	32	00	00	E	LD (WRS_1),A	Save updated value in WRS		
556	005A	C3	00	00	E	JP EEE8	Return to calling routine		
558	005D					END			

ASSEMBLER ERRORS = 0

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 1

1 SOURCE: &EXTMR
2 PROGRAMMER: LIZ POTEET
3
4 4 CHANNEL DIO MUX (FORDYCE) - HOST ISR - CTC TIMER ON/OFF
5 INTERRUPT
6 *****
7 * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1983. ALL RIGHTS
8 * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,
9 * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT
10 * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.
11 *****
12
13 LIST X
14 NAME EXTMR
15 COPY &MX4EQUS
16 LIST S
17 ;
18 PUBLIC TMROFF
19 ;
20 EXTRN TMRFLG,EEE7
21 *****
22
23 DESCRIPTION:
24 This routine is part of the Host Interrupt, Timer On/Off, and
25 is 'called' by a routine in &MXEXT. The purpose of the following
26 code is to either turn on or off the 16 Millisec Timer. A flag is
27 checked to determine whether the timer is to be turned off or on.
28 If TMRFLG=1, the timer is currently on and the interrupt is to
29 turn it off. Conversely, if TMRFLG=0, the timer is currently off
30 and the interrupt is signalling it to be turned back on.
31
32 REGISTER RESTRICTION:
33 E - Used in the calling routine - Do Not Use
34
35 CALLED BY (FILE): &MXHST
36 *****
37
38 CSEG
39
40 491 0000 0E E2 E TMROFF: LD C,CTC_1_C2 Get address of CTC channel
41 492 0002 3A 00 00 LD A,(TMRFLG) Get contents of Timer flag
42 493 0005 1F RRA A,0 Rotate bit 0 into Carry flag
43 494 0006 38 10 JR C,T01 If timer now on, jump
44
45 ;TIMER OFF - TURN BACK ON

ERR LINE ADDR B1 B2 B3 B4

Z80 ASSEMBLER VER 3.0MR

PAGE 2

488	0008	3E A7		LD A,CTCWRD	Get CTC channel control word
489	000A	ED 79		OUT (C),A	Send to CTC
490	000C	3E E7		LD A,TMRPRE	Get time constant register value
491	000E	ED 79		OUT (C),A	Send to CTC
492	0010	3E 01		LD A,1	Setting TMRFLG to 1
493	0012	32 00 00	E	LD (TMRFLG),A	
494	0015	C3 20 00	E	JP TO2	Go to exit
496				;TIMER ON NOW - TURN OFF	
498	0018	3E 03		T01: LD A,3	Reset channel
499	001A	ED 79		OUT (C),A	Send to CTC
500	001C	AF		XOR A	
501	001D	32 00 00	E	LD (TMRFLG),A	Set timer flag=0 to reflect timer off
503	0020	C3 00 00	E	TO2: JP EEE7	Return to Caller
505	0023			END	

ASSEMBLER ERRORS = 0

ok]
I.62>