PM8313
D3MX

D3MX Module of the PM4944 M13 Reference Design

Issue 2: October, 1996

CONTENTS

LIST OF FIGURES

LIST OF TABLES

REFERENCES

- American National Standards for Telecommunications, ANSI T1.103 (1987), "Digital Hierarchy - Synchronous DS3 Format Specifications"
- American National Standards for Telecommunications, ANSI T1.107 (1988), "Digital Hierarchy - Formats Specifications"
- American National Standards for Telecommunications, ANSI T1.107 (Draft 1995), "Digital Hierarchy - Formats Specifications"
- American National Standards for Telecommunications, ANSI T1.107a (1990), "Digital Hierarchy - Supplement to Formats Specifications"
- American National Standards for Telecommunications, ANSI T1.231 (1993), "Digital Hierarchy - Layer 1 In-Service Digital Transmission Performance Monitoring"
- AT&T Microelectronics, Data Sheet (November 1993), "T7295-6 DS3/SONET STS-1 Integrated Line Receiver"
- AT&T Microelectronics, Data Sheet (March 1992), "T7296 DS3/STS-1 Integrated Line Transmitter"
- EXAR Corporation, Data Book (1992)
- EXAR Corporation, Data Sheet (April 1994), "XR-T7296 DS3/STS-1, E3 Integrated Line Transmitter"
- Integrated Device Technology, Data Book (1995), "Specialized Memories, FIFOs & Modules"
- Microchip Technology Inc., Data Book (1994)
- Motorola Semiconductor, Technical Data Sheet (1992), "MC88915 Low Skew CMOS PLL Clock Driver"
- PMC-Sierra, Data Book (July 1994), "PM8313 D3MX M13 Multiplexer", PMC-920702S4
- PMC-Sierra, PMC910501 (February 1992) - "PM5101 Evaluation Motherboard"

OVERVIEW

The PM4944 M13RD (M13 Reference Design) embodies PMC-Sierra's guidelines and suggestions for designing with the PM8313 D3MX M13 Multiplexer device. The full M13 reference design consists of two or more types of modules: the D3MX Module (described in this document) and Tributary Modules.

In addition to the PM8313, the D3MX Module incorporates an on-board microcontroller (Microchip PIC16C74) for providing the local maintenance functions — including termination of the C-Bit Parity datalink in the DS-3 overhead. The DS-3 line interface unit (LIU) used is the AT&T Microelectronics T7295-6 and T7296 chip set. Note that EXAR Corporation produces the pin-compatible XR-T7295-6 and XR-T7296 devices.

The Tributary Modules contain circuitry to process the tributary streams of the multiplexing operation. Each Tributary Module processes up to four (quad density) of the duplex tributary channels. These modules may contain framers (such as the PM4344 TQUAD) or line interface units (such as the PM4314 QDSX). There are two reference designs available that are pin compatible with this reference design. They are theTQUAD Module of the M13 reference design (PMC-951003), and the S/UNI-MPH with QDSX reference design (PMC-960951). Seven such modules are connected to the D3MX Module for an M13 application.

The D3MX Module communicates with the host system a 96-pin connector. The pin-out of this connector is compatible with PMC-Sierra's PM1501 evaluation motherboard (described in document PMC-910501). The PM1501 motherboard contains a Motorola MC68HC11 with a serial communications interface.

The host does not have a direct connection to the microprocessor port of the D3MX. Rather, a dual-port RAM, shared by the host and the local PIC16C74, is used to pass control and status information about the D3MX to and from the host, where the actual register accesses to the D3MX are performed by the PIC16C74. The advantage to this architecture is that the host system is not burdened by the low-level monitoring and control functions. The host system and PIC16C74 communicate through the "mailbox" communication channels in the dual-port RAM.

The D3MX Module also communicates with up to seven Tributary Modules via seven 100-pin connectors. It is assumed that each Tributary Module has up to quad density (four duplex channels). It is also assumed that each Tributary Module has a local microcontroller for providing the local maintenance functions (including termination of the datalinks).

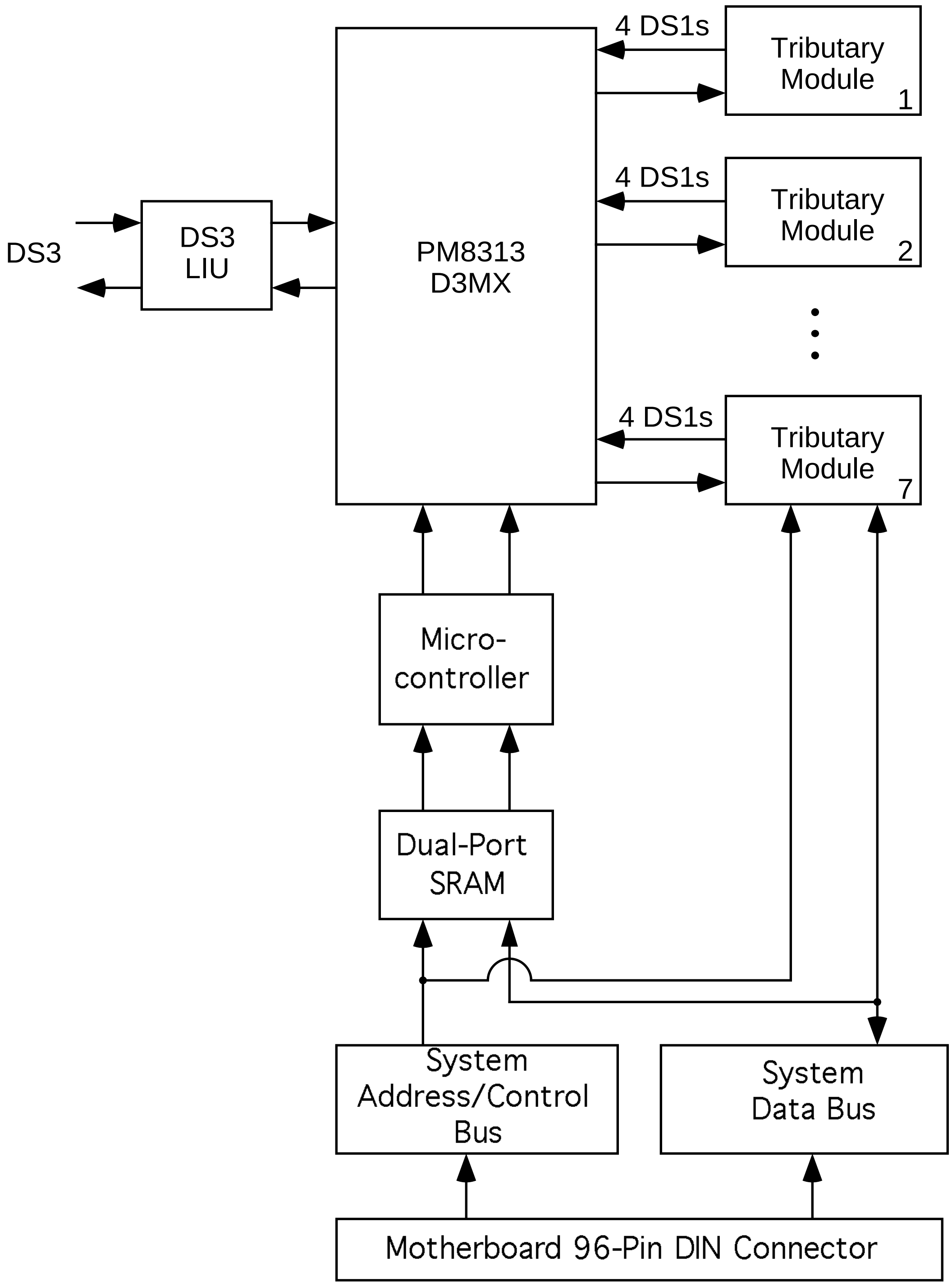
The host does have direct connection to the microprocessor bus of the Tributary Modules. However, it is expected that these modules would have a similar architecture (dual-port RAM shared by the host and a local microcontroller).

The D3MX Module supports all the multiplexing modes of the D3MX. In addition to M23 and C-Bit parity DS1 multiplexing schemes, E1 signals can be multiplexed to DS3 using the G.747 mode. Additionally, DS2 signals can be multiplexed to DS3. The actual mode would be determined by the Tributary Modules.

PMC-Sierra has a two companion reference designs available that are pin compatible with this reference design. They are theTQUAD Module of the M13 reference design (PMC-951003), and the S/UNI-MPH with QDSX reference design (PMC-960951). Future reference designs will be issued for Tributary Modules containing other PMC-Sierra devices.

FUNCTIONAL DESCRIPTION

Block Diagram



PM8313 D3MX

The D3MX is a monolithic integrated circuit that implements a M13 multiplexing function together with a full-featured DS-3 framer. It is the heart of the D3MX Module; all traffic goes through the D3MX. On the line side, the D3MX transmits and receives DS-3 frames (optionally with C-bit parity) through the AT&T line interface. On the tributary side, the de-multiplexed serial DS-1 streams are output and the serial DS-1 streams to be multiplexed are input. These tributary streams are terminated on Tributary Modules connected to the 100-pin connectors.

A line rate clock is required by the D3MX on the TICLK input. This clock is propagated to the TCLK output to which the line data is timed. The duty cycle of the TCLK output is regenerated by the Motorola MC88915 PLL to meet the tight duty cycle requirements (±2.5%) of the DS-3 LIU.

The D3MX is configured, controlled and monitored via a parallel microprocessor port, although it can also operate in a limited stand-alone mode. It is implemented in low power, +5 Volt CMOS technology. It has TTL-compatible inputs and outputs and is packaged in a 208-pin PQFP package.

For a complete description of the D3MX, please refer to PMC-Sierra's PM8313 databook (document number PMC-911105).

DS-3 Line Interface

The DS-3 line interface is implemented with the AT&T T7295-6 and T7296 devices. The use of these devices was modeled after information given in their respective datasheets.

An alternative second source for these components is EXAR Corporation.

The receive line interface directly feeds the received DS-3 signal into the T7295-6 RIN input pin via an AC-coupling capacitor. The coaxial line's 75Ω characteristic impedance is terminated with a 75Ω resistor to ground. The T7295-6's internal receive equalizer is used (REQB pin is held low).

The differential AMI outputs TTIP and TRING of the T7296 are passed through a 1:1 transformer before being put on the 75Ω coaxial transmit line via a BNC connector.

When not loop-timed, the transmit timing is taken from a 44.736MHz crystal oscillator (propagated through the D3MX and MC88915). For loop-timed operation, the D3MX's TICLK input is externally connected (under control of the local PIC16C74) to its RCLK input. Additionally, a manual switch is provided on the D3MX Module to control whether the board operates in loop-timed mode or not. Other jumpers are available for manual control over loopback modes in the DS3 LIU.

The TCLK input to the T7296 has its duty cycle regenerated with a Motorola MC88915 "low skew" CMOS PLL which ensures the tight tolerance on the output duty cycle necessary for transmitting standards-compliant DS3 pulses.

Dual-Port RAM

This a high-speed 2K by 8-bit dual-port static RAM with internal interrupt logic for inter-processor communications. Many manufactures (such as Integrated Device Technology, and Cypress Semiconductors) produce pin-compatible versions of this device.

There are two independent ports with separate control, address and data pins that permit independent, asynchronous access for both reads and writes to any location in memory.

Memory Map

Tables 1 to 5 define the memory map for the dual-port RAM. The memory map is organized to provide a generic flexible interface to the D3MX physical layer control and status functionality.

Table 1. Dual-port RAM Memory Map

Ram Address (hex)	Function (length)
000	RFDL receive buffer and status
080	XFDL transmit buffer and status
100	PMON Shadow registers, DS3
10C	PMON Shadow registers, DS2 #1
10F	PMON Shadow registers, DS2 #2
112	PMON Shadow registers, DS2 #3
115	PMON Shadow registers, DS2 #4
118	PMON Shadow registers, DS2 #5
11B	PMON Shadow registers, DS2 #6
11E	PMON Shadow registers, DS2 #7
121	Latest validated FEAC BOC
122	ID (number) of latest interrupting DS2 (1-7)
123	Last line loopback activated or deactivated (0=DS3,1-1C=DS1#,1D=ALL DS1s)
7F6	LOCK pin state mirror
7F7	RLOL pin state mirror
7F8	Value of last read D3MX register
7F9	Command argument #3

Ram Address (hex)	Function (length)
7FA	Command argument #2
7FB	Command argument #1
7FC	F/W revision #
7FD	F/W version #
7FE	Mailbox out (PIC->HOST)
7FF	Mailbox in (HOST->PIC)

Table 2. RFDL Memory block of 80H bytes

Relative Address (hex)	Function
00 - 7D	Packet Data (maximum 126 bytes)
7E	Channel Status
7F	Packet Length

Table 3. XFDL Memory block of 80H bytes

Relative Address (hex)	Function
00 - 7E	Packet Data (maximum 127 bytes)
7F	Packet Length

Table 4. PMON DS3 Memory block of CH bytes

Relative Address (hex)	Function
0	Line code violation count LSB
1	Line code violation count MSB
2	Framing error event count LSB
3	Framing error event count MSB
4	Excessive zeros count LSB
5	Excessive zeros count MSB
6	P-bit parity error event count LSB
7	P-bit parity error event count MSB
8	C-bit parity error event count LSB
9	C-bit parity error event count MSB
A	Far end block error event count LSB
B	Far end block error event count MSB

Table 5. PMON DS2 Memory block of 3H bytes

Relative Address (hex)	Function
0	Framing error event count
1	Parity error event count LSB
2	Parity error event count MSB

Mailbox Communication

Each port has one address location assigned as a "mailbox." When a port writes to its mailbox, the other port will be interrupted. This interrupt is cleared when the mailbox is read by the interrupted port. These mailboxes can be used as a control and status channel. By defining functions for certain values passed in the mailbox, each port can initiate actions of the other port. Additionally, a port can pass high-priority status information through the mailbox.

HOST-TO-PIC16C74 COMMUNICATION

The host sends control commands through the microcontroller's mailbox (location 7FF). Table 6 shows the mailbox codes for host-to-PIC16C74 messaging. Following the table is a description of each command and further processing (if necessary) required of the host.

Table 6. Host-to-PIC16C74 Mailbox Codes

Function	Code (hex)
Read D3MX register	01
Write D3MX register	02
Start HDLC packet transmission	03
Start BOC transmission	04
Stop BOC transmission	05
Line loopback activate request	06
Line loopback deactivate request	07
Enable loop timing	08
Disable loop timing	09

READ D3MX REGISTER (01)

This function is useful for diagnostic purposes to read the value of a D3MX register. To read a value from a specific D3MX register perform the following steps:

1. Write the LSB of the address of the register to RAM location 7FB.
2. Write the MSB of the address of the register to RAM location 7FA.
3. Write the mailbox command 01 to the PIC16C74 Mailbox (location 7FF).
4. When the Requested D3MX Register I/O Complete code (01) is received in the Host Mailbox (7FE), the read register value will be available in RAM location 7F8.

WRITE D3MX REGISTER (02)

This function is useful for diagnostic purposes to write a D3MX register with a value. To write a value from a specific D3MX register perform the following steps:

1. Write the LSB of the address of the register to RAM location 7FB.
2. Write the MSB of the address of the register to RAM location 7FA.
3. Write the data byte value to RAM location 7F9.
4. Write the mailbox command 02 to the PIC16C74 Mailbox (location 7FF).

START HDLC PACKET TRANSMISSION (03)

After HDLC packet data (up to 127 bytes) is placed in the XFDL transmit buffer (in dual-port RAM), this command can be used to initiate transmission. Once initiated, the packet transmission will be handled by the PIC16C74. To transmit an FDL packet perform the following steps:

1. Write the packet data to the XFDL transmit buffer.
2. Write the packet's length to the Packet Length byte of the quadrant's XFDL transmit buffer.
3. Write the mailbox command 03 to the PIC16C74 Mailbox (location 7FF).

START BIT-ORIENTED CODE TRANSMISSION (04)

To send a bit-oriented code on the facility data link perform the following steps:

1. Write the 6-bit BOC (least significant bit transmitted first) to RAM location 7FB.
2. Write the mailbox command 04 to the PIC16C74 Mailbox (location 7FF).

The bit-oriented code will be transmitted until stopped using the Stop Bit-Oriented Code Transmission command.

STOP BIT-ORIENTED CODE TRANSMISSION (05)

To stop the transmission of a bit-oriented code on the facility data link, write the mailbox command 05 to the PIC16C74 mailbox (location 7FF).

LINE LOOPBACK ACTIVATE (06)

To activate line loopback on a particular line, perform the following steps:

1. Write the line number (1-28) to RAM location 7FB.
2. Write the mailbox command 06 to the PIC16C74 mailbox (location 7FF).

LINE LOOPBACK DEACTIVATE (07)

To deactivate line loopback on a particular line, perform the following steps:

1. Write the line number (1-28) to RAM location 7FB.
2. Write the mailbox command 07 to the PIC16C74 mailbox (location 7FF).

ENABLE LOOP TIMING (08)

To enable loop timing, Write the mailbox command 08 to the PIC16C74 mailbox (location 7FF).

DISABLE LOOP TIMING (09)

To disable loop timing, Write the mailbox command 09 to the PIC16C74 mailbox (location 7FF).

PIC16C74 TO HOST COMMUNICATION

The microcontroller sends alarm and status information to the host through the host's mailbox (location 7FE). Table 7 shows the mailbox codes for microcontroller-to-host messaging. Following the table is a description of each message and further processing (if necessary) required of the host.

Because the PIC16C74 takes a finite time to process information, it will only indicate one interrupt at a time. The host must process the mailbox before the next mailbox code is written by the PIC16C74. With the current code, the shortest delay between consecutive mailbox interrupts occurs when multiple channels indicate a simple alarm (e.g. RED). Therefore, the host must process all mailbox interrupts within approximately 50 µs, otherwise some information will be lost (overwritten).

Table 7. PIC16C74-to-Host Mailbox Codes

Function	Code (hex)
D3MX register I/O complete	01
PMON registers updated	02
HDLC packet transmission complete	03
New HDLC packet received	04
New FEAC BOC validated	05

Function	Code (hex)
FEAC idle	06
DS3 AIS asserted	07
DS3 AIS cleared	08
DS3 RED alarm asserted	09
DS3 RED alarm cleared	0A
DS3 IDLE asserted	0B
DS3 IDLE cleared	0C
DS2 AIS asserted	0D
DS2 AIS cleared	0E
DS2 RED alarm asserted	0F
DS2 RED alarm cleared	10
Reserved	11
Reserved	12
Line loopback activated	13
Line loopback deactivated	14

D3MX REGISTER I/O COMPLETE (01)

This message is sent to indicate that the requested register read of the D3MX has been completed.

PMON REGISTERS UPDATED (02)

This message is sent to indicate that PMON register statistics have been updated to their locations in RAM.

HDLC PACKET TRANSMISSION COMPLETE (03)

This message is sent to indicate that the requested HDLC packet transmission has been completed.

NEW HDLC PACKET RECEIVED (04)

This message is sent when the D3MX is finished receiving a new HDLC packet. A separate code is assigned to each originating quadrant. The packet length, channel status and packet data can be read from the quadrant's RFDL receive buffer.

VALID BOC DETECTED (05)

This message is sent when the D3MX detects a valid BOC on the receive HDLC. The received BOC can be read from RAM location 121. A BOC is considered valid if it is repeated at least 8 times in a window of 10 consecutively received BOCs.

If the BOC matches a standardized loopback activate or deactivate command, the PIC16C74 will automatically respond by putting the line into the appropriate mode. Note that as specified in ANSI T1.403, a line will not be put into the Line Loopback mode until the Line-Loopback-Activate BOC has been removed (to ensure that the loopback command is not looped back itself).

FEAC IDLE (06)

This message is sent when the D3MX detects a transition from a valid BOC to idle code.

DS3 AIS ASSERTED (07)

This message is sent when the D3MX detects that an AIS is being received by the DS3 framer. The AIS detection will take 1.5 seconds to integrate in the presence of a continuously received AIS pattern.

DS3 AIS CLEARED (08)

This message is sent when the D3MX detects that the AIS is no longer being received by the DS3 framer. The AIS clear will take 16.8 seconds to integrate in the presence of a continuously received framed pattern.

DS3 RED ALARM ASSERTED (09)

This message is sent when the D3MX detects that the DS3 framer is in red alarm.

DS3 RED ALARM CLEARED (0A)

This message is sent when the D3MX DS3 framer is no longer in red alarm.

DS3 IDLE ASSERTED (0B)

This message is sent when the D3MX detects that the DS3 framer is idle.

DS3 IDLE ASSERTED (0C)

This message is sent when the D3MX DS3 framer is no longer idle.

DS2 AIS ASSERTED (0D)

This message is sent when the D3MX detects that an AIS is being received by one of the DS2 framers. The line number can be read from RAM location 122H. The AIS detection will take 1.5 seconds to integrate in the presence of a continuously received AIS pattern.

DS2 AIS CLEARED (0E)

This message is sent when the D3MX detects that an AIS is no longer being received by a particular DS2 framer. The line number can be read from ram location 122H. The AIS clear will take 16.8 seconds to integrate in the presence of a continuously received framed pattern.

DS2 RED ALARM ASSERTED (0F)

This message is sent when the D3MX detects that one of the DS2 framers is in red alarm. The DS2 number can be read from RAM location 122H.

DS2 RED ALARM ASSERTED (10)

This message is sent when the D3MX DS2 framer is no longer in red alarm.

LINE LOOPBACK ACTIVATED (13)

This message indicates that a line has been placed in loopback mode. The line can be obtained from RAM location 123H as described in Table 8 below.

Table 8. Line loopback codes

RAM location 123H	Line Number indicated
0	DS3
1 - 1C	DS1 #1 - 1C
1D	All DS1s

LINE LOOPBACK DEACTIVATED (14)

This message indicates that a line has been taken out of loopback mode. The line can be obtained from RAM location 123H as indicated in the previous section.

PIC16C74 Microcontroller

The Microchip PIC16C74 is a low-cost, high-performance, CMOS, fully-static EPROM-based 8-bit microcontroller. It employs a Harvard RISC-like architecture with 14-bit instruction words. Its two stage instruction pipeline allows most instructions to be executed in a single cycle. The PIC16C74 has enhanced core features, an eight-level deep stack, and multiple internal and external interrupt sources.

The PIC16C74 has 192 bytes of on-chip RAM and a 4k by 14-bit EPROM for program memory.

It has 33 I/O pins, each of which can source/sink 25mA.

The PIC16C74 also has a number of peripheral features (A/D converters, timer/counters, capture/compare/PWM modules, and two serial ports) which are not used in this reference design.

In the D3MX Module, the PIC16C74 is devoted to the local maintenance functions for the DS3 transmission (including termination of the C-Bit Parity datalink, if active). The maintenance functions for the tributaries will be provided by similar microcontrollers on the Tributary Modules.

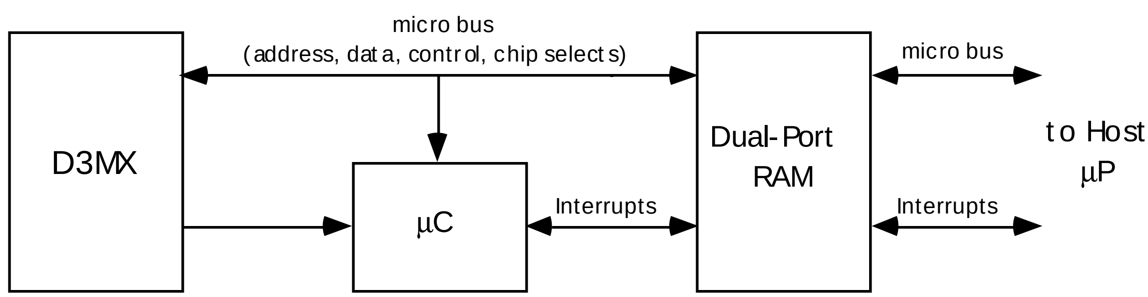
Firmware

Example firmware for the PIC16C74 is included in Appendix D. The firmware was developed in Assembly language using Microchip's PC-hosted symbolic assembler, MPASM (v1.30.01), available free from Microchip's bulletin board (MCHIPBBS on Compuserve) or Web site (http://www.microchip.com). The firmware was simulated using MPSIM (v5.20), also available free from Microchip. The PIC16C74 can be programmed on universal programmers from Sprint and Data I/O. Additionally, Microchip sells low cost programmers for this device.

External Memory Accesses

The PIC16C74 does not have a microprocessor bus suitable to access registers and memory within the D3MX and dual-port RAM. Therefore I/O pins on the PIC16C74 are defined to act as the address bus, data bus, chip selects, RDB, WRB and other signals necessary. Since the PIC16C74 is only accessing two external devices, it generates the chip selects itself instead of relying on external decode logic. Figure 1 shows a block diagram of the connections.

Figure 1. Block Diagram of PIC16C74 Connections



The firmware in Appendix D includes subroutines that operate dedicated I/O pins as the micro bus during external accesses. Since the interrupt service routines will often alternate between accessing the D3MX and the dual-port RAM, it saves instruction cycles to have separate subroutines dedicated to each, where the address would be taken from different pre defined registers. There are four general access routines: READ_D3MX, READ_RAM, WRITE_D3MX, and WRITE_RAM. All four routines share a common Data register so that data transfers could be optimized.

Initialization

The firmware initializes the D3MX to enable the appropriate interrupts, set the framing format, configure the interfaces, set the timing options, and initialize the HDLC serial controllers. The D3MX is initialized for C-Bit parity format.

The dual-port RAM is initialized with the version and revision number of the firmware. The microcontroller-side mailbox is read to clear any spurious interrupt.

Finally, the initialization routine enables the interrupts within the PIC16C74, and falls into a main loop.

Dual-Port Mailboxes

The dual-port RAM contains two mailbox registers, one in each direction, for passing data between the two ports. When the micro bus on one port writes a data byte to its mailbox address, the other port is interrupted. Using these mailboxes, a proprietary messaging scheme can be arranged. These interrupts from the dual-port RAM are processed by the PIC16C74.

Datalink Service Routines (RFDL/XFDL)

The C-Bit Parity facility datalink is a 28.2kbps channel provided for an HDLC datalink. The PM8313 D3MX provides detection and generation of the HDLC packet overhead (using its internal RFDL and XFDL functional blocks).

It is assumed that the internal RFDL and XFDL TSBs will be used to process the HDLC overhead, rather than using external HDLC controllers (the 28.2kbps datalink can be optionally extracted as a serial stream for external processing).

The RFDL and XFDL TSBs generate interrupt indications on two (each) dedicated outputs. Optionally, these interrupts can be configured as active low open drain outputs and wire-ORed with the common INTB output.

The procedure for handling the interrupts from the RFDL and XFDL blocks is explained in the "Operations" section of the D3MX databook (p. 119-26 in the July 1994 issue). At a minimum, the service routines will have to perform the three steps detailed in the D3MX databook. Added to these steps would be house-keeping operations in the dual-port RAM to inform the host of the datalink status.

Maintenance Functions

There are a number of maintenance functions specified in ANSI T1.107 and T1.231. These include FERF Alarm, Alarm Indication Signal (AIS), RED Alarm, Loopbacks, and Performance Monitoring.

FERF ALARM

A DS3 FERF Alarm (both X-Bits cleared to logic zero) should be indicated to the remote DS3 equipment in response to a out-of-frame condition or receipt of an AIS defect. This is performed in firmware by the D3MX.

ALARM INDICATION SIGNAL (AIS)

The D3MX has the option that when it receives DS3 AIS, it can propagate DS2 AIS to the receive DS2 tributaries. Similarly, the it can propagate DS1 AIS to the receive DS1 tributaries when it receives DS2 AIS.

Since this function is done automatically by the D3MX, the PIC16C74 does not need to perform any AIS activation.

LOOPBACKS

Loopbacks are used as a maintenance tool to aid problem resolution. There are two loopbacks defined by ANSI T1.107: Line Loopback and Sub-Rate Loopback, both of which are supported in the D3MX. In C-Bit parity format only Line Loopback is applicable.

In the C-Bit Parity format, line loopbacks can be applied to either the DS3 or to individual DS1 tributaries. Line loopback activation and deactivation is controlled by bit-oriented codes carried in the FEAC (Far-End Alarm and Control) channel. The protocol is to send the Loopback Activate or Deactivate code ten times, then send the code for the channel to be looped back ten times.

The RBOC (Receive Bit-Oriented Codes) functional block in the D3MX looks for bit-oriented codes and interrupts when one is detected. In response to these interrupts, the firmware enables or disables the loopbacks with the appropriate bits (LLBE in Register 04H, or LBA[4:0] in Registers 4BH, 5BH, 6BH, 7BH, 8BH, 9BH, and ABH) of the D3MX.

Performance Monitoring

All the performance monitoring parameters required by ANSI T1.231 are available within registers of the D3MX. Therefore, with a timer interrupt setting the accumulation interval, the PIC16C74 transfers all the performance monitoring indicators and parameters to known memory locations in the dual-port RAM.

IMPLEMENTATION DESCRIPTION

The schematic diagrams of the D3MX Module are contained in Appendix C. This section explains, sheet by sheet, the schematic diagram.

Hardware

The D3MX Module schematic shows four main functional blocks: a PIC16C74 microcontroller, dual-port RAM, line interface unit (LIU), and a PM8313 D3MX. Additionally, the schematic contains connectors, timing sources and miscellaneous "glue" circuitry.

PIC16C74 Microcontroller

Figure 2 shows how the I/O pins of the PIC16C74 have been defined for this reference design.

Figure 2. PIC16C74 Port Map

bit	7	6	5	4	3	2	1	0
PORTA	unused	unused	LOCK	RAM CE	D3MX CE	A10	A9	A8
PORTB	LED 4	LED 3	LED 2	RAM INT	LED 1	RLOL	BUSY	D3MX INT
PORTC	A7	A6	A5	A4	A3	A2	A1	A0
PORTD	D7	D6	D5	D4	D3	D2	D1	D0
PORTE	unused	unused	unused	unused	unused	LOOPT	WRB	RDB

Although the PIC16C74 is a simple microcontroller to program, there are some issues which can cause difficulties for the first-time programmer. Here is a list to be aware of:

- The internal register space (data memory) is partitioned into 2 banks. Accessing any particular register requires that the bank bit (bit 5 of the STATUS register) be set accordingly prior to the access. Care must be taken to select the register in the right bank, otherwise bugs which are hard to isolate will occur. Care must also be taken to preserve the state of the bank bit during interrupts.

- The internal program memory is partitioned into 2 pages. The page that is currently being executed is specified by the page bit (bit 3 of the PCLATH register). When a branch or call is made, the page bit is copied into the program counter. When making cross-page subroutine calls, the page bit must be set according to the location of the subroutine.

  The entire program counter is pushed onto the stack when a call is made, so when returning there is no need to program the page bit. This means that if a subroutine which manipulates the page bit is called, the state of the page bit may not be known upon return from that subroutine. Therefore, it is good programming practice to explicitly restore the page bit upon returning from a cross-page subroutine call to reflect the page from which the call was made.

- The PIC16C74 has four I/O pins (bits 4-7 of PORTB) which have an interrupt-on-change feature. They can be used as edge-triggered external interrupts. Each time the port is read (with any read or read-modify-write instruction) the state of these pins is latched into comparison logic. If at any time there is a pin transition such that a mismatch between the previous latched value and the current value occurs, an interrupt is asserted.

  The proper procedure for clearing the mismatch condition is explained in a Microchip application note (document number DS00566A) entitled "Using the Port B Interrupt on Change as an External Interrupt."

  There is a shortcoming in the interrupt logic of the PIC16C74 which makes it possible that transitions on the pins are occasionally missed. To work around this issue the interrupt-on-change pins should be occasionally polled for changes.

- Care must be taken when globally disabling interrupts because there exists the possibility that an interrupt will occur while the global interrupt enable bit is being cleared. To ensure that interrupts have been disabled, the following code fragment is recommended:

```
DISABLEINTS
         BCF                INTCON, GIE
         BTFSC    INTCON, GIE
         GOTO     DISABLEINTS
         ...
```

- Because of complications involving register file bank swapping and preservation of the zero (Z) bit in the STATUS register, the following code fragment is the recommended procedure for state preservation in an interrupt service routine:

```
MOVWF    TEMP_W
SWAPF    STATUS, W
BCF               STATUS, RP0
MOVWF    TEMP_STATUS
MOVF              PCLATH, W
MOVWF    TEMP_PCLATH
```

And the following code is recommended to restore the saved registers before returning from an interrupt service routine:

```
MOVF              TEMP_PCLATH, W
MOVWF    PCLATH
SWAPF    TEMP_STATUS, W
MOVWF    STATUS
SWAPF    TEMP_W, F
SWAPF    TEMP_W, W
```

Dual-Port RAM

The dual-port RAM is shared by the host system and the local PIC16C74 microcontroller. The PIC16C74 performs all the local maintenance functions (including termination of the facility datalink), while the dual-port RAM is used to pass information to and from the host system. This arrangement greatly reduces the real-time burden on the host system.

The dual-port RAM holds, as a minimum, the following information:

- packets received by the D3MX over the C-bit parity facility datalink
- packets to be transmitted by the D3MX over the C-bit parity facility datalink.
- status of the D3MX. This includes performance monitoring indications and parameters as specified in ANSI T1.231.

Each D3MX Module has been allocated 2 kB of space.

The two ports on the dual-port RAM are denoted the "Left" and "Right" ports. In this design, the Left Port is connected to the 100-pin connector which carries the host microprocessor bus (which is decoded, de-multiplexed and buffered on the D3MX Module). The Right Port is connected to the PIC16C74's microprocessor bus (shared with the microprocessor port of the D3MX).

The two ports are entirely independent from each other and allow asynchronous Read and Write accesses from either port.

Each port also has an interrupt indication line used for processing a "mailbox" communications channel within the dual-port RAM. Each port has a memory location, which, when it is written to, alerts the other port via the interrupt indication signal. Therefore, a protocol can be used to pass information through these mailboxes. Using the mailboxes allows:

- the host to "peek" and "poke" registers in the D3MX even though it is not directly connected to its microprocessor port,
- the host to initiate different configuration routines,
- the host to initiate different diagnostic routines, and
- the PIC16C74 to interrupt the host during alarm conditions and when HDLC packets or BOCs are received on the C-bit parity datalink or FEAC channel.

D3MX Functional Block

The D3MX functional block contains the PM8313 D3MX plus support circuitry. The D3MX is the heart of the M13 application — all traffic flows through it. It provides the standard multiplexes to DS-3 (ANSI T1.107 and ITU-T G.747) as well as standard DS-3 framing functions and performance monitoring (ANSI T1.231).

The full register set of the D3MX, including Test Mode registers are accessible to the PIC16C74 block. For a full description of these registers, refer to the D3MX Data Book.

In addition to the D3MX, this sheet contains a circuit for putting the DS3 line into loop-timing mode. This circuit is controlled by the LOOP_T signal from the PIC16C74, or alternatively by a manual switch. In loop-timed mode, the signal at the RCLK input of the D3MX is connected to its TICLK input. In locally-timed mode, the TICLK input is driven with the 44.736MHz oscillator on the D3MX Module.

This block also contains LEDs to visually indicate the state of the DS3 interface as monitored by the D3MX and DS3_LIU functional blocks. The RLOS, REXZ, RAIS, ROOF/RRED and RFERF status signals from the D3MX are connected to LEDs. Additionally, the RLOS, RLOL and LOCK status signals from the DS3_LIU block are connected to LEDs.

DECODE_LOGIC Functional Block

The DECODE_LOGIC functional block provides the chip select decoding of the address on the microprocessor bus. Additionally, this block de-multiplexes the AD[7:0] address/data lines from the 96-pin connector interface. The DECODE_LOGIC functional block buffers the remaining signals, and pulls up any that are tristateable.

The outputs of the DECODE_LOGIC functional block go to the Left Port of the dual-port RAM, as well as to each of the Tributary Modules.

The chip selects control access to the dual-port RAM as well as to the Tributary Modules (connected to the 100-pin connectors). The address space has been divided into eight 2kbyte ranges as follows:

Table 9. Tributary Address Space

Address Range (hexadecimal)	Device Selected
0000 to 07FF	Dual-Port RAM (left port)
0800 to 0FFF	Tributary Module #1
1000 to 17FF	Tributary Module #2
1800 to 1FFF	Tributary Module #3
2000 to 17FF	Tributary Module #4
2800 to 2FFF	Tributary Module #5
3000 to 37FF	Tributary Module #6
3800 to 3FFF	Tributary Module #7

The chip selects are generated by decoding the A[13:11] signals with a 1-of-8 decoder.

Additionally, the microprocessor bus is demultiplexed (a Motorola-type multiplexed bus is assumed), by using the ALE (Address Latch Enable) signal to latch the address.

The outputs from this functional block are connected to the dual-port RAM on the D3MX module, as well as the connectors to the seven Tributary Modules. The Tributary Modules may, or may not, have their own local microcontrollers.

Dual-Port RAM

The dual-port RAM is shared by the host system and the local PIC16C74 microcontroller. The PIC16C74 performs all the local maintenance functions (including termination of the facility datalink), while the dual-port RAM is used to pass information to and from the host system. This arrangement greatly reduces the real-time burden on the host system, allowing the host to interact with the D3MX Module at a higher level than direct access to the individual physical layer devices.

The dual-port RAM holds, as a minimum, the following information:

- packets received by the D3MX over the DS3 C-Bit parity facility datalink
- packets to be transmitted by the D3MX over the DS3 C-Bit parity facility datalink
- status of the D3MX. This includes performance monitoring indications and parameters as specified in ANSI T1.231.

The total amount of memory required to hold this information depends on the implementation. The 2k by 8-bit size was chosen to make it symmetrical with the space allocated per Tributary Module.

Each Tributary Module need 2 kbytes of space because they must accumulate information on *four* tributary channels. For example, permanent allocation of 128 bytes per tributary datalink (for each of transmit and receive) uses up 1kbyte alone. The remaining 1kbyte of memory should be sufficient for the status information (which would generally consist of bit flags).

The two ports on the dual-port RAM are denoted the "Left" and "Right" ports. In this design, the Left Port is connected to the output of the DECODE_LOGIC block which is a demultiplexed and buffered version of the host microprocessor bus. The Right Port is connected to the PIC16C74's microprocessor bus (shared with the microprocessor port of the D3MX).

The two ports are entirely independent from each other and allow asynchronous Read and Write accesses from either port.

Each port also has an interrupt indication line used for processing a "mailbox" communications channel within the dual-port RAM. Each port has a memory location, which, when it is written to, alerts the other port via the interrupt indication signal. Therefore, a protocol can be used to pass information through these mailboxes. Uses of this communications channels include:

- allowing the host to "peek" and "poke" registers in the D3MX even though it is not directly connected to the D3MX's microprocessor port,
- initiating different configuration routines, and
- initiating different diagnostic routines.

DS3_LIU

The DS-3 line interface is implemented with the AT&T T7295-6 and T7296 devices. The use of these devices was modeled after information given in their respective datasheets.

An alternative second source for these components is EXAR Corporation.

The receive line interface directly feeds the received DS-3 signal into the T7295-6 RIN input pin via an AC-coupling capacitor. The coaxial line's 75Ω characteristic impedance is terminated with a 75Ω resistor to ground. The T7295-6's internal receive equalizer is used (REQB pin is held low).

The differential AMI outputs TTIP and TRING of the T7296 are passed through a 1:1 transformer before being put on the 75Ω coaxial transmit line via a BNC connector.

When not loop-timed, the transmit timing is taken from a 44.736MHz crystal oscillator (propagated through the D3MX and MC88915). For loop-timed operation, the D3MX's TICLK input is externally connected (under control of the local PIC16C74) to its RCLK input. Additionally, a manual switch is provided on the D3MX Module to control whether the board operates in loop-timed mode or not. Other jumpers are available for manual control over loopback modes in the DS3 LIU.

The Motorola MC88915 "Low Skew CMOS PLL Clock Driver" is necessary to restore the duty cycle of the TCLK output of the D3MX to meet the tight duty cycle requirements of the TCLK input of the T7296. This input requires a duty cycle between 47.5% and 52.5%.

The TCLK output of the D3MX is a feed-through version of its TICLK input. In this design, the TICLK input comes from a crystal oscillator which has ±5% duty cycle. This clock is further distorted as it propagates through the D3MX. Therefore, it is obvious that the duty cycle needs to be restored before being input into the T7296.

The output duty cycle of the MC88915 is ±200ps which, at 44.736MHz, works out to less than ±1%. The timing of the feedback of the MC88915 was chosen so that the output could be used to sample the data without violating the set-up and hold time requirements of the T7296.

The data sheet for the MC88915 gives further information on that device.

Timing Distribution

The D3MX Module has on-board oscillators for driving the various functional blocks. The DS3_LIU and D3MX functional blocks require a DS-3 (44.736MHz) line clock. The PIC16C74 requires a high-speed clock, in this case a 20MHz oscillator is used. This clock is also distributed to the Tributary Modules to drive their local microcontrollers, if any. Finally, an oscillator is provided to distribute a high-speed clock to the Tributary Modules. The intended use is as a 37.056MHz clock for driving DS1 Tributary Modules based on PMC-Sierra's PM4344 TQUAD device, or similar.

There are also two external inputs (clock and frame pulse) which is distributed to all the Tributary Modules. These signals are used to synchronize the framer backplanes for synchronous switching applications.

All the clocks are buffered because they drive long traces at high frequencies.

D3MX Module Host 96-Pin Connector Interface

The D3MX Module host 96-pin connector interface carries all the signals required to connect the D3MX Module to a higher layer entity.

The connector is a 96-pin female connector, with the pin defined as described in the table below (only 64 of the pins are used). This connector was chosen because it is compatible with the PMC-Sierra PM1501 Evaluation Motherboard (described in PMC-910501).

Signals are provided for a Motorola-type multiplexed microprocessor bus (including interrupts, and a hardware RESET signal). Additionally, power and ground are provided through this connector. TTL signal levels are assumed on this interface.

In the following table the signal type is one of: I (input), O (output), I/O (bi-directional), N/C (no-connect), PWR (power), or GND (ground). The direction of the signal is with reference to this design.

Table 10 D3MX Module Host 96-Pin Connector

Signal Name	Type	Pin	Function
ALE	I	C1	Address latch enable. When high, identifies that address is valid on AD[7:0]. This signal is used for de-multiplexing the microprocessor bus.
E	I	C2	External data access indication. Active high.
RWB	I	C3	Active low write enable, active high read enable
RSTB	I	C4	Active low hardware RESET. This is connected to all devices providing a hardware RESET pin. After a hardware RESET, the D3MX Module should be re-initialized unless operating in stand-alone mode (no microprocessor).
A[15..8]	I	C5-C12	Address bus bits 15..8
AD[7..0]	I/O	C13-C20	Multiplexed address/data bus bits 7..0
PA3	I	C21	68HC11 Processor Port A bit 3
PA4	I	C22	68HC11 Processor Port A bit 4
PA5	I	C23	68HC11 Processor Port A bit 5
PA6	I	C24	68HC11 Processor Port A bit 6
PD2	O	C25	Master In Slave Out (MISO) of 68HC11 Port D bit 2 acting as SPI. Pulled up on motherboard.
PD3	I	C26	Master Out Slave In (MOSI) of 68HC11 Port D bit 3 acting as SPI. Pulled up on motherboard.
PD4	I	C27	Serial Clock (SCK) of 68HC11 Port D bit 4 acting as SPI. Pulled up on motherboard.
PD5	I	C28	Slave Select (SS) of 68HC11 Port D bit 5 acting as SPI active low. Pulled up on motherboard.
IRQB	O	C29	Maskable 68HC11 interrupt. Pulled up on motherboard.
BRB (XIRQB)	O	C30	Non Maskable 68HC11 interrupt. Pulled up on motherboard. This signal is used by the DLXC to indicate to the microprocessor that it requests the use of the microprocessor bus.
DISB	O	C31	EVMB memory disable. Pulling this signal low will disable MPU access to the EVMB's on-board RAM and EPROM. Pulled up on motherboard.

Signal Name	Type	Pin	Function
SP	I	C32	Spare
GND	GND	A1-A28	Ground
+5V	PWR	A29-A32	+5 Volts

Tributary Module 100-Pin Connectors

The Tributary Module Connector sheets contain the seven 100-pin connectors used to connect to the Tributary Modules. These connectors carry four duplex DS1 signals (clock and data), a microprocessor control, address, and data bus, as well as power to the Tributary Modules. The signals are defined in the following table.

In the following table, the signal type is one of: I (input), O (output), I/O (bi-directional), N/C (no-connect), PWR (power), or GND (ground). The direction of the signal is with reference to this design.

Table 11. Tributary Module 100-Pin Connector

Pin Name	Type	Pin	Function
GND	GND	A1, A2	Ground
N/C	N/C	A3	No connection
INTB	I	A4	Interrupt indication (active low).
BFPI	O	A5	Backplane frame pulse. This is an 8kHz signal used to synchronize the frame alignment of the framer backplanes for synchronous switching applications.
BCLK	O	A6	Backplane clock. This is a tributary line rate clock used to synchronize the timing of the framer backplanes for synchronous switching applications.
GND	GND	A7, A8	Ground
N/C	N/C	A9-A14	No connection
VCC	PWR	A15, A16	+5V
N/C	N/C	A17, A18	No connection
GND	GND	A19, A20	Ground
RCLKI[1]	O	A21	Receive Clock for tributary #1
RDD[1]	O	A22	Receive Digital Data for tributary #1
GND	GND	A23, A24	Ground
RCLKI[2]	O	A25	Receive Clock for tributary #2
RDD[2]	O	A26	Receive Digital Data for tributary #2
GND	GND	A27, A28	Ground
RCLKI[3]	O	A29	Receive Clock for tributary #3
RDD[3]	O	A30	Receive Digital Data for tributary #3
GND	GND	A31, A32	Ground
RCLKI[4]	O	A33	Receive Clock for tributary #4
RDD[4]	O	A34	Receive Digital Data for tributary #4
GND	GND	A35, A36	Ground
TCLKO[1]	I	A37	Transmit Clock for tributary #1

Pin Name	Type	Pin	Function
TDD[1]	I	A38	Transmit Digital Data for tributary #1
GND	GND	A39, A40	Ground
TCLKO[2]	I	A41	Transmit Clock for tributary #2
TDD[2]	I	A42	Transmit Digital Data for tributary #2
GND	GND	A43, A44	Ground
TCLKO[3]	I	A45	Transmit Clock for tributary #3
TDD[3]	I	A46	Transmit Digital Data for tributary #3
VDD	PWR	A47, A48	+5V
N/C	N/C	A49	No connection
RSTB	O	A50	Hardware Reset (active low)
TCLKO[4]	I	A51	Transmit Clock for tributary #4
TDD[4]	I	A52	Transmit Digital Data for tributary #4
GND	GND	A53, A54	Ground
A[3:0]	O	A55-A58	Address Bus [3:0]
GND	GND	A59, A60	Ground
A[5, 4]	O	A61, A62	Address Bus [5, 4]
VDD	PWR	A63, A64	+5V
A[7, 6]	O	A65, A66	Address Bus [7, 6]
GND	GND	A67, A68	Ground
A[10:8]	O	A69-A71	Address Bus [10:8]
N/C	N/C	A72	No connection
GND	GND	A73, A74	Ground
N/C	N/C	A75-A78	No connection
GND	GND	A79, A80	Ground
RWB	O	A81	Read/Write Bar signal of the microprocessor control bus.
N/C	N/C	A82	No connection
VDD	PWR	A83, A84	+5V
CSB	O	A85	Chip Select (active low). Selects the devices on the Tributary Module for microprocessor access.
N/C	N/C	A86	No connection
GND	GND	A87, A88	Ground
D[3:0]	I/O	A89-A92	Data Bus [3:0]
GND	GND	A93, A94	Ground
D[7:4]	I/O	A95-A98	Data Bus [7:4]
TQCLK	O	A99	High-Speed Clock for devices on Tributary Modules
MCLK	O	A100	High-Speed Clock for microcontrollers on Tributary Modules.

Firmware

Appendix D contains the source code listing of the assembly language firmware used to program the PIC16C74. This section gives a brief overview of the code in Appendix D, describing the functional routines and associated implementation issues.

Note: The source code in Appendix D is meant as a proof-of-concept that an inexpensive, simple microcontroller is capable of handling all the physical layer functions associated with the four DS1 channels of a PM8313 D3MX device. *It is the responsibility of the person(s) using or adapting the example code to ensure that the* *resulting system operation complies with both standardized and proprietary requirements.*

The manufacturer of the PIC16C74, Microchip, provides free firmware development software for the assembler (MPASM) and simulator (MPSIM). The firmware for this reference design was developed and tested using that free software.

An efficient way of processing events that may have a low frequency of occurrence is to use interrupt driven routines (instead of polling). The events (alarm conditions, timers, datalink servicing, and dual-port mailbox events) on the D3MX and the dual-port RAM will, on average, be low frequency. Therefore, the PIC16C74 firmware developed for this reference design is based on an interrupt driven scheme.

When the PIC16C74 detects an interrupt indication on its external interrupt input pins, it determines the source of the interrupt by polling its internal interrupt source register to determine if it was the D3MX or the dual-port RAM that interrupted. If it was the D3MX, then the D3MX's interrupt source register needs to be further polled to determine what event caused the interrupt. Once the PIC16C74 has determined the interrupt source, it can jump to a routine specific to that interrupt.

The *P16C74.INC* File

The *P16C74.INC* file is the standard include file for the PIC16C74 provided by Microchip which contains labels for all the special registers and bits of the microcontroller.

The *MACROS.INC* File

The *MACROS.INC* file defines some macros which are generally useful when using the PIC16C74.

<u>MACROS</u>

Table 12 describes the macros defined in the *MACROS.INC* file.

Table 12. Macros in *MACROS.INC*

Macro	Arguments	Description
BANK0	none	Selects the Bank 0 register space.
BANK1	none	Selects the Bank 1 register space
PAGE0	none	Selects Page 0 of program memory
PAGE1	none	Selects Page 1 of program memory
INTSOFF	none	Disables all interrupts by clearing the global interrupt enable, and testing to ensure that it is cleared.
INTSON	none	Enables all interrupts by setting the global interrupt enable.
SNE	Value, W	Skip the next instruction if Value is not equal to W
SE	Value, W	Skip the next instruction if Value equal to W

The *D3MX.ASM* File

The *D3MX.ASM* file is the main source file containing the macros, constants and routines specific to this reference design.

<u>CONSTANTS</u>

The first portion of the *D3MX.ASM* code contains a number of CBLOCK and CONSTANT statements which assign labels to the following:

- user registers within the Bank 0 register space.
- external LEDs driven by bits in the Port B register.
- D3MX direct access registers.
- D3MX indirect access registers.
- dual-port RAM memory locations as per memory map

Following this are CONSTANT statements defining dual-port RAM mailbox codes for host to microcontroller and microcontroller to host communication and FEAC bit-oriented codes. Also included here are a number of miscellaneous constant definitions whose purpose is explained in the code.

<u>MACROS</u>

Table 13 describes the macros used to perform I/O operations with the D3MX and the dual-port RAM.

Table 13. Description of Macros in *D3MX.ASM*

Macro	Arguments	Description
WR_RAM	*address*, *value*	Writes *value* (byte) to the *address* in the dual-port RAM.
WR_RAM_D	*address*	Writes the value in the data register (DATA_REG) to the *address* in the dual-port RAM.
WR_D3MX	*register*, *value*	Writes *value* (byte) to the *register* in the D3MX.
WR_D3MX_D	*register;*	Writes the value in the data register to the D3MX *register* i.
RD_RAM	*address*	Reads from the *address* in the dual-port RAM. The value is returned in the data register and the W register.
RD_D3MX	*register*	Reads from the *register* in the D3MX. The value is returned in the data register and the W register.

ROUTINES

The following subsections describe each of the routines in the *D3MX.asm* file.

Main Loop

The processor loops infinitely waiting for the 1s time flag to be set. The 1s timer flag is set by a routine which is invoked using the internal timer interrupt. When the 1s flag is set the processor executes the PMON subroutine, updating performance monitoring shadow registers in RAM, and returns to the main loop. Here also the watchdog timer is cleared.

Performance Monitoring

Shadowing of performance monitoring statistics in the dual-port RAM is done by the PMON subroutine.

The PMON subroutine writes to the Global PMON Update register of the D3MX to trigger the update of performance statistics. The routine then delays to allow for the update latency in the D3MX. After a delay, the routine reads the data from the D3MX registers to local PIC registers, and copies them to the pre defined locations in Dual-port RAM.

Read Access to D3MX

The READ_D3MX subroutine is used by the RD_D3MX macro. The address for the D3MX register is copied to the address latch (PORT C and bits 0-2 of PORT A) and the control lines WRB and CSB2 are toggled to perform a read cycle. The data bus (PORT D) is latched into the data register (DATA_REG).

Read Access to Dual-Port RAM

The READ_RAM subroutine is used by the RD_RAM macro. The RAM address is copied to the address latch (PORT C and bits 0-2 of PORT A) and the control lines WRB and CSB1 are toggled to perform a read cycle. The data bus (PORT D) is latched into the data register (DATA_REG).

Write Access to D3MX

The WRITE_D3MX subroutine is used by the WR_D3MX, WR_D3MXL and WR_D3MX_DL macros. The address for the D3MX register is copied to the address latch (PORT C and bits 0-2 of PORT A) and the data register (DATA_REG) is copied to the data bus (PORTD). The control lines WRB and CSB2 are toggled to perform a write cycle.

Write Access to Dual-Port RAM

The WRITE_RAM subroutine is used by the WR_RAM and WR_RAM_D macros. The RAM address is copied to the address latch (PORT C and bits 0-2 of PORT A) and the data register (DATA_REG) is copied to the data bus (PORTD). The control lines WRB and CSB1 are toggled to perform a write cycle.

Initialize Microcontroller

The INIT_MICRO routine makes all control pins inactive, turns off the LEDs and clears the user registers. Timer 0 is configured and the 1 second timer counter is initialized. Timer 2 is configured with a 1:16 prescaler. The input/output state of each pin is set. PORT A is configured as a digital input (can be a A/D port). PORT B is read to initialize the interrupt on change logic.

Initialize Dual-Port RAM

The INIT_RAM routine reads the micro's dual-port RAM mailbox to clear any spurious interrupt. The version and revision numbers are copied to their respective locations in RAM.

Initialize D3MX

The D3MX is first initialized by setting Bit 0 of register 0x00 to logic one and then clearing it to perform a software reset. Table 14 shows the D3MX registers initialized and the value to which they are set. The D3MX is initialized for C-bit parity format on all channels.

Table 14. D3MX Initialization Register Values

Register	Location	Value
Master HDLC configuration	03	00
Master interface configuration	05	09
Master alarm enable	06	8C
DS3 TRAN configuration	0C	01
MX23 configuration	28	02
DS3 FRMR configuration	34	83
XFDL configuration	20	03
RFDL interrupt status	25	02
RFDL configuration	24	01
DS3 FRMR interrupt enable	35	4D
RBOC configuration	32	05
DS2 FRMR interrupt enable	41,51,..A1	24

Interrupt Service

The INTHDLR routine deals with interrupts asserted by the D3MX, dual-port RAM and internally by timer 0 and timer 2 Each source's interrupt flag is polled in turn, and if asserted the corresponding interrupt service routine is called. Those registers saved on entry into the service routine are restored on exit.

The D3MX interrupt is serviced as long as it is asserted to minimize latency when there are multiple pending D3MX. This is important because the interrupts are edge-triggered.

The RAM interrupt pin is also polled as a work-around to solve the problem that the interrupt-on-change logic of the PIC16C74 can occasionally miss edges.

D3MX Interrupt Service

The D3MX_INT routine determines the source of a D3MX interrupt, (and DS2 FRMR # if appropriate) and then branches to the appropriate service routine.

XFDL Interrupt Service

An XFDL interrupt will occur when either the XFDL needs another data byte for transmission, or if an underrun has occurred in the XFDL FIFO. When an XFDL interrupt occurs, the firmware reads the XFDL interrupt status register to determine which condition has occurred.

If the XFDL is ready for a new byte of data, the firmware checks for end of message. If the message is over, the host is informed that the HDLC transmission is complete. Otherwise the next byte is copied to the transmit data register. If an underrun has occurred, the underrun bit in the interrupt status register is cleared.

RFDL Interrupt Service

The RFDL Receive Data register is read and stored locally. The RFDL Status register is read to determine if an overrun or abort has occurred. In the case of an overrun the overrun bit in the RAM copy of the RFDL status register is set. If an abort has occurred, the local link status is made inactive and the status and packet length count are reset. If the link has gone from inactive to active then the link status is set to active and the packet length counter is reset in preparation for a new packet. If this is the next byte of a packet currently being received, then the byte is copied into the RAM receive buffer, packet length counter incremented and the end of message bit in the RFDL status register is tested. If this byte is the end of the current message then the CRC error bit and NVB bits of the RAM RFDL status register are updated, local link status made inactive and packet length copied to RAM. If the packet length is greater than 3 (which all valid packets with CRC enabled will be) then the host is notified of the new packet through its mailbox.

DS3 Framer Interrupt Service

The DS3 framer interrupt status register is examined to determine if a loss of signal, AIS, or RED alarm has occurred, or if the DS3 channel is idle. A message is written to the host's mailbox accordingly.

RBOC Interrupt Service

The RBOC interrupt status register is examined to determine if a valid BOC has been detected or if the channel has gone idle. A message is written to the host's mailbox accordingly.

In the case that a new valid BOC has been detected, the BOC is copied to RAM. The firmware checks for the absence of a previous line loopback activate or deactivate command (as indicated by the FLAGS register) and sets the appropriate bits as indicated by the new BOC. If a previous line loopback activate or deactivate has occurred, the firmware performs the appropriate action on the line indicated by the new BOC.

DS2 Framer Interrupt Service

The DS2 framer interrupt status register is examined to determine if an AIS or RED alarm has been asserted or cleared. A message is written to the host's mailbox accordingly. In the case that an AIS has been asserted, all loopbacks are cleared.

Dual-Port RAM Interrupt Service

The micro's mailbox is read to determine the request issued by the host. If a valid code is read the corresponding routine is executed.

Timer 0 Interrupt Service

This routine decrements the 1s counter and reflects the status of the RLOL and LOCK pins in RAM. When the counter reaches 0, it toggles LED4, and reloads the counter.

Timer 2 Interrupt Service

This routine operates in four different modes. The first and third times through, it sets the timer flag. The second time, it changes the BOC to line codeword BOCl. The fourth time through it idles the FEAC channel. The fifth time it starts the cycle over again.

APPENDIX A: DESIGN CONSIDERATIONS

Power Supply Voltage Transients

High currents drawn during IC switching causes power supply voltage transients due to the inductance of the power lines. The magnitude of the noise voltage can be reduced by minimizing the inductance of the power lines and by decreasing the magnitude of the transient currents. The power line inductance can be minimized by using a power plane. The transient currents on the power rails can be minimized by supplying the power from a local source such as a de-coupling capacitor near the circuit drawing the current.

The de-coupling capacitance and the inductance of the connection between the capacitor and the power pin determine the noise voltage at the power pin. The effectiveness of the de-coupling capacitor depends on the frequencies of the transients. Large "bulk" de-coupling capacitors are used to supply the low-frequency current variations and the small "noise-bypassing" capacitors are used to supply the high-frequency transient current that is required when the circuit is switching.

Ground Noise

Return currents and power supply transients during high current consumption produce most of the ground noise. Since ground noise cannot be controlled by de-coupling capacitors, the only way to minimizing the effect of ground noise is to minimize ground impedance. The best way to minimize ground impedance is to use a ground plane. It is not advisable to use ferrite beads in the ground path as this will inhibit the return currents from leaving and raise the ground noise level.

Noise-Bypassing at Power Pins

The D3MX can generate a lot of simultaneous switching noise, especially if the tributary clocks are synchronous. It is important to provide a noise-bypassing capacitor at every power pin so that the switching currents can be supplied locally, thereby reducing the noise introduced into the power plane.

Values of Noise-Bypassing Capacitors

A rule of thumb is that the "bulk" noise-bypassing capacitor (placed where the power enters the circuit board) should have 10 times the value of all the noise-bypassing capacitors combined. Capacitors with low internal inductance should be used such as a tantalum electrolytic. Stay away from aluminum electrolytic as their inductances are an order of magnitude larger than tantalum capacitors.

The noise-bypassing capacitors (placed near the power pins) must be able to supply all the switching current. The minimum capacitance can be calculated by:

$$C = \Delta I * \Delta t / \Delta V$$

The transient voltage drop ΔV in the supply voltage is caused by the transient current ΔI occurring over time Δt. This equation shows that the voltage drop will be minimized as the capacitance is increased. However, using capacitors that are too large should be avoided due to their resonance characteristics.

Since all capacitors have some stray inductance in series with the capacitance, there will be a self-resonance at a certain frequency given by the equation:

$$f = \frac{1}{2\pi\sqrt{LC}}$$

Note that the larger the capacitance (for the same inductance) the lower the resonant frequency. If the capacitor is too large, the self-resonance will be too low to be an effective bypass but if the capacitor is not large enough, there will be insufficient current to supply the transient current during switching. The smallest value capacitor to satisfy the above equations should be used. It is rarely necessary that a capacitor larger than 0.01 µF be used.

Placement of Noise-Bypassing Capacitors

The de-coupling capacitor should be placed as close to the IC power pin as possible reduce the wiring inductance. There are five sources of inductance: the parasitic inductance of the capacitor, the inductance of the wiring between the capacitor and the IC power pin, the power pin lead inductance inside the IC, the ground pin lead inductance inside the IC, and the ground inductance between the IC pin and ground. The capacitor inductance is negligible if the correct capacitor is used. There is no control over the lead frame inductance. To keep the inductance low, both the power lead and the ground lead should be keep as short as possible (less than 1.5 inches). The inductance for a trace is given by:

$$L = 0.005 \log^{-1}\left(2\pi \frac{h}{w}\right) \quad \mu\text{H/inch}$$

where $\mathbf{h}$ is the height between the power or ground lead and the ground plane and $\mathbf{w}$ is the width of the power or ground lead. Note that doubling the width of the trace or reducing $\mathbf{h}$ will only decrease $\mathbf{L}$ approximately by 20%, but decreasing the length by 50% will decrease the inductance by 50%. A typical PCB trace has about 15nH of inductance per inch.

Ferrite Beads

Ferrite beads are mainly used on power rails to pass DC current but to attenuate the higher frequency noise that is riding on the DC rail. The impedance of ferrite beads increases with frequency; at DC the ferrite bead is like a short but at higher frequency the impedance of a ferrite bead can increase to over 100 ohms (depending on the bead and frequency). Ferrite beads attenuate high frequency noise from the power supply from getting into a circuit, but they also stop high frequency switching currents required by digital ICs. It is important, therefore, to use proper noise bypass capacitors when using ferrite beads to provide a local source of switching current.

Ferrite beads should be avoided on CMOS I/O power pins as the high current switching of the CMOS circuits causes a $\Delta I/\Delta t$ noise to be introduced into the power rail. This noise is induced because the ferrite beads "starve" the digital circuitry, causing the voltage to fluctuate locally. Ferrite beads should also be avoided on the ground bus as this inhibits the return currents.

As the noise frequencies and levels are different in every design, it is hard to decide if beads are necessary and at what frequency should they be effective. However, it is harder to insert a ferrite bead when no spot has been provided than it is to short out a bead if it is not needed.

According to AT&T's recommendations, ferrite beads are used on the DS-3 line interface components' (T7295-6 and T7296) analog power pins (V_{DDA}), presumably to block noise from the digital power pins.

Unused CMOS Inputs

"Floating" CMOS inputs (those that are left unconnected) may switch unpredictably, causing unwanted noise and power consumption. These phenomena can cause erratic system behavior. Therefore, all unused inputs should be connected to their inactive state: to ground or to the power rail (Vcc). Unused biDirectionals should be "pulled" through a series resistor (4.7k or greater) to avoid short-circuits occuring if the biDirectionals are erroneously configured as outputs.

How to Isolate Power Supplies

"Power supply isolation" in the context of this design means that precautions are made so that all power supply pins get the best possible power supply — low noise and required nominal voltage (within the specified tolerance of the device).

Since only one ground plane and one power plane is normally available, the transmit, receive, and digital power and grounds can be isolated by cutting away channels from the respective planes. The power and grounds should be brought from a quiet part of the board, usually where the power and grounds enter the board. Ferrite beads can also be used on the receive and transmit analog powers to prevent digital noise from entering analog circuits of the DS3 LIU.

Power Supply Isolation: Analog from Digital

Digital CMOS circuits have high immunity to external noise (approximately 30% of Vcc), but analog circuits can be "devastated" by small amount of external noise. Additionally, CMOS circuits can also generate a lot of switching noise, especially when a large number of circuits are running synchronously all timed to the same system clock (which can easily happen in an M13 application). Both of these facts suggest that effort should be made to ensure that noise from the digital circuitry does not get into any analog circuitry in a design. If a device is mixed-signal, one must trust that the device designers took every consideration necessary to avoid internal noise coupling.

In the D3MX Module, the analog circuitry is the circuitry in the AT&T LIU used to detect and generate DS-3 line pulses. Both the T7295-6 and T7296 provide separate analog power and ground pins.

Power Supply Isolation: Transmit Analog from Receive Analog

Any noise on the D3MX receive analog power and ground inputs the internal clock recovery PLL's ability to recover the clock from the incoming data. Added noise will degrade jitter tolerance and add jitter to the recovered clock.

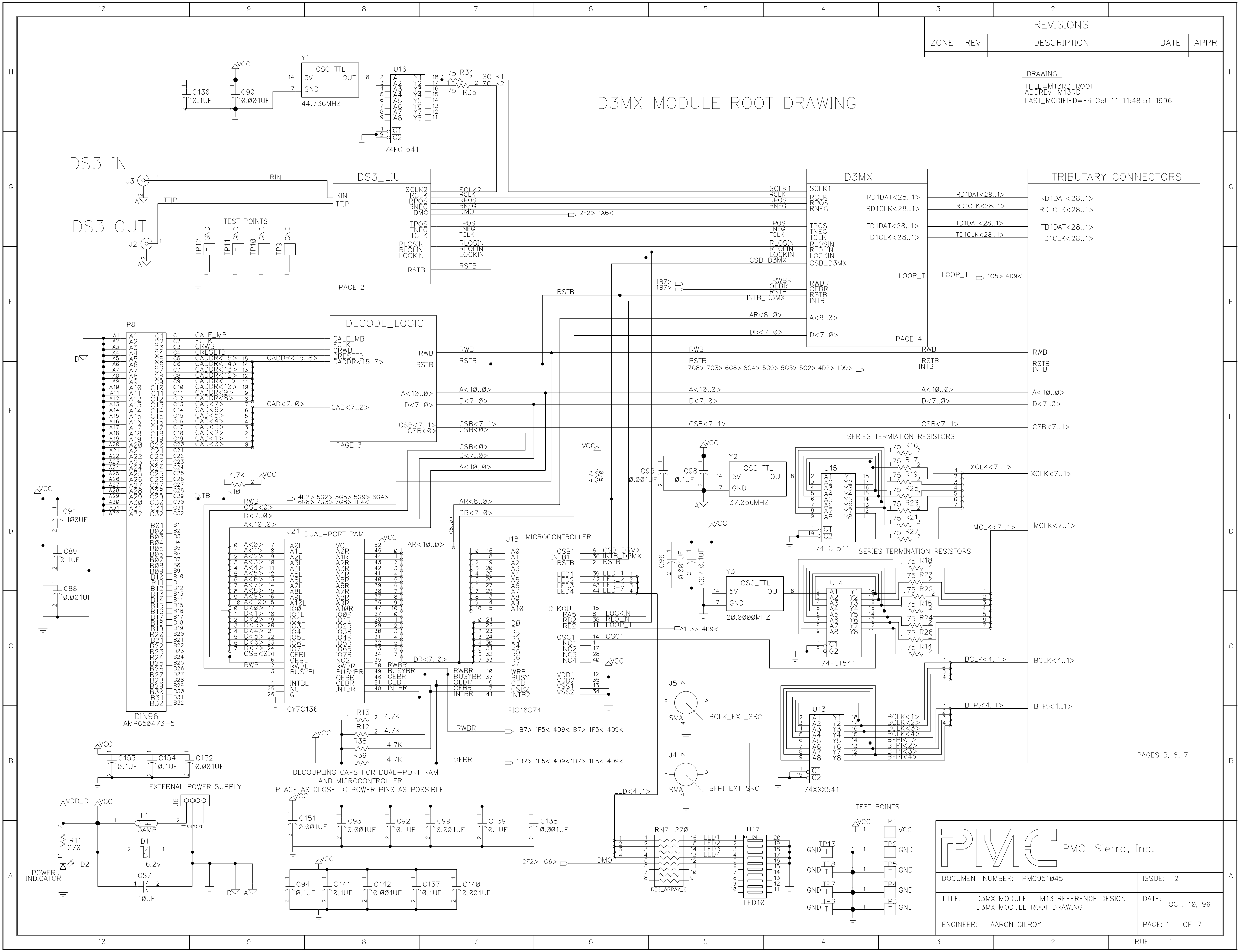
On the transmit side of the D3MX a PLL is used to restore the duty cycle of the 44.736MHz reference clock. Any added noise on the power or ground inputs impacts the resulting 44.736MHz clock. The added noise will increase the intrinsic jitter of the transmitter.

APPENDIX B: MATERIAL LIST

Item No.	Part Name - Value	JEDEC Type	Reference Designator	Qty
1	74FCT257_SOIC-BASE	SOIC16	U6	1
2	74FCT541_SOIC-BASE	SOIC20W	U14-U16	3
3	74XXX00_SOIC-HCMOS	SOIC14	U19	1
4	74XXX138_SOIC-HCMOS	SOIC16	U10, U20	2
5	74XXX245_SOIC-HCMOS	SOIC20W	U11	1
6	74XXX32_SOIC-HCMOS	SOIC14	U7	1
7	74XXX373_SOIC-HCMOS	SOIC20W	U9	1
8	74XXX541_SOIC-HCMOS	SOIC20W	U4, U12, U13	3
9	BNC_AMPHENOL-BASE	AMPHENOL_BNC	J2, J3	2
10	CAP-0.001UF	SMDCAP805	C88, C90, C93, C95, C96, C99-C105, C109, C111, C113, C118, C119, C138, C140, C142-C152	30
11	CAP-100000PF	SMDCAP1206	C1, C3-C61, C63, C66-C68, C71-C84, C89, C92, C94, C97, C98, C106-C108, C110, C112, C114-C117, C120-C137, C139, C141, C153, C154, C156, C157	116
12	CAP-10000PF	SMDCAP805	C2	1
13	CAP-10PF, NPO_805	SMDCAP805	C155	1
14	CAPACITOR POL-100UF, 16V, ELECTRO	CAP320	C91	1
15	CAPACITOR POL-10UF, 16V, TANT TEH	SMDTANCAP_C	C62, C64, C65, C69, C70, C85-C87	8
16	CONN100_MALE-AMP_1-104118-7	AMP_1-104118-7	P1-P7	7
17	CY7C136_-BASE	PLCC52	U21	1
18	D3MX-BASE	PQFP208	U8	1
19	DIN96_MALE-BASE	AMP_650473-5	P8	1
20	DIODEZENER_SMD-6.2V, 1W	DL_41	D1	1
21	FUSE3A1_SMD-3AMP, NANO	NANO_SMF	F1	1

Item No.	Part Name - Value	JEDEC Type	Reference Designator	Qty
22	HEADER2-BASE	JUMPER2	JP1	1
23	HEADER8-BASE	SIP8	J1	1
24	INDUCTOR-FB, 50, FAIR RITE	INDUCTOR_FB	L1-L4	4
25	LED-RED, PCB RIGHT ANGLE	LED	D2	1
26	LED10-RED, 25MA, 2.1V	DIP20_LED	U5, U17	2
27	MC88915-BASE	PLCC28	U3	1
28	OSC_TTL_DIP-20.0000M HZ, 100 PPMA	CRYS14	Y3	1
29	OSC_TTL_DIP-37.056MHZ, 25 PPM, CA	CRYS14	Y2	1
30	OSC_TTL_DIP-44.736MHZ, 20 PPM, CA	CRYS14	Y1	1
31	PE_65967-BASE	PE_65967	T1	1
32	PIC16C74-BASE	PIC16C74	U18	1
33	PWRBLOCK-BASE	CONN04END	J6	1
34	RESISTOR-1.0K, 5%	SMDRES805	R2	1
35	RESISTOR-1.0M, 5%	SMDRES805	R5	1
36	RESISTOR-270, 5%	SMDRES805	R6, R7, R11	3
37	RESISTOR-36, 5%	SMDRES805	R3, R4	2
38	RESISTOR-4.7K, 5%	SMDRES805	R8-R10, R12, R13, R36, R38-R40	9
39	RESISTOR-75, 1%	SMDRES805	R1, R14-R35	23
40	RESISTOR-75, 5%	SMDRES805	R37	1
41	RES_ARRAY_8_SMD-10K	SOIC16	RN2, RN5, RN6, RN13, RN14	5
42	RES_ARRAY_8_SMD-270	SOIC16	RN1, RN7	2
43	SMA-BASE	SMA	J4, J5	2
44	T7295_6-BASE	SOJ20	U2	1
45	T7296-BASE	SOJ28	U1	1
46	TST_PT-BASE	TST_PT_1	TP1-TP13	13

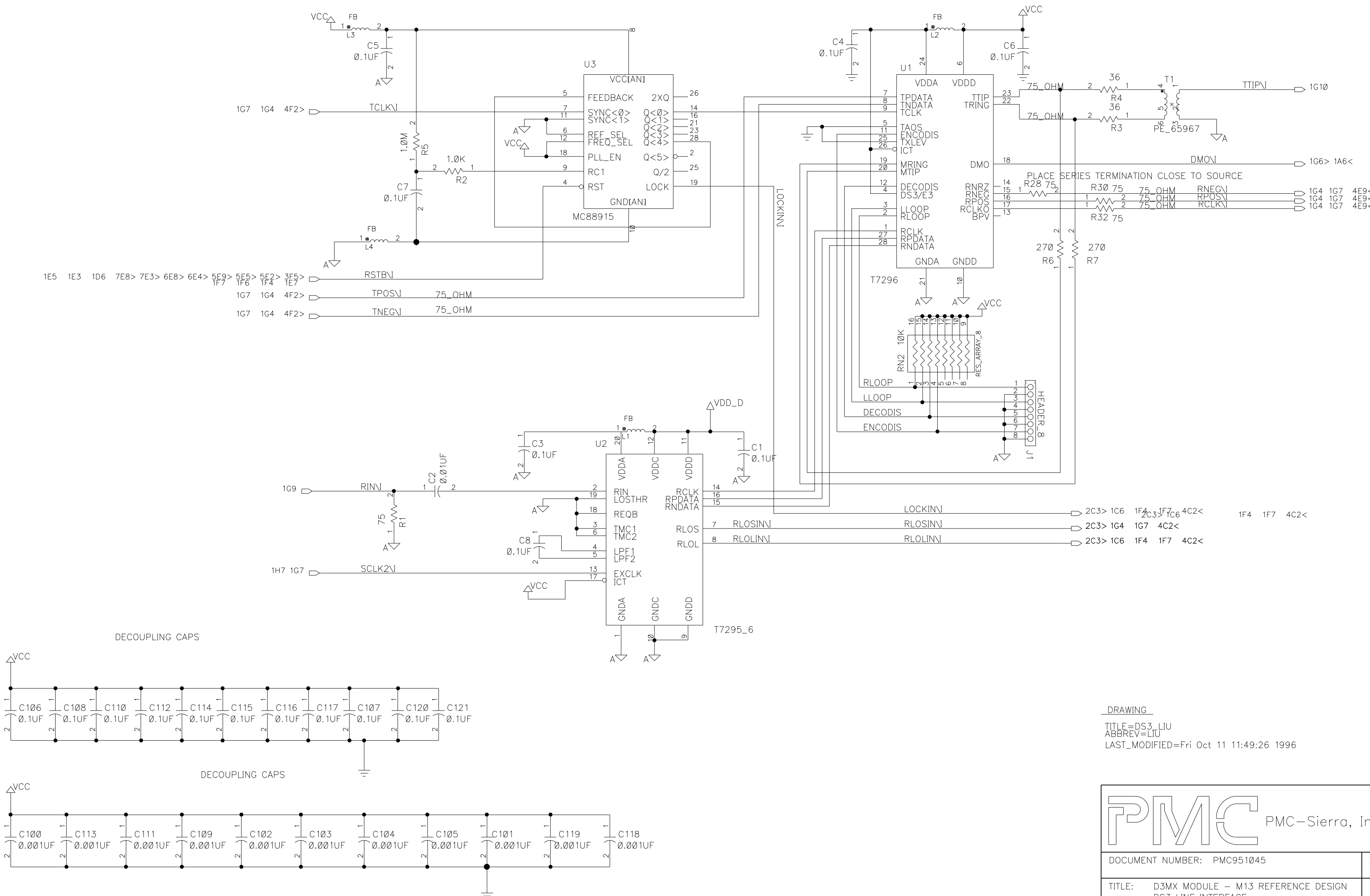
APPENDIX C: SCHEMATICS


D3MX MODULE ROOT DRAWING
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPR
DRAWING
TITLE=M13RD_ROOT
ABBREV=M13RD
LAST_MODIFIED=Fri Oct 11 11:48:51 1996
DS3 IN
DS3 OUT
TEST POINTS
DS3_LIU
PAGE 2
D3MX
PAGE 4
TRIBUTARY CONNECTORS
PAGES 5, 6, 7
DECODE_LOGIC
PAGE 3
DUAL-PORT RAM
CY7C136
MICROCONTROLLER
PIC16C74
SERIES TERMIATION RESISTORS
SERIES TERMINATION RESISTORS
74FCT541
74XXX541
OSC_TTL
44.736MHZ
37.056MHZ
20.0000MHZ
DIN96
AMP650473-5
EXTERNAL POWER SUPPLY
POWER INDICATOR
DECOUPLING CAPS FOR DUAL-PORT RAM
AND MICROCONTROLLER
PLACE AS CLOSE TO POWER PINS AS POSSIBLE
RES_ARRAY_8
LED10
PMC-Sierra, Inc.
DOCUMENT NUMBER: PMC951045
ISSUE: 2
TITLE: D3MX MODULE - M13 REFERENCE DESIGN
D3MX MODULE ROOT DRAWING
DATE: OCT. 10, 96
ENGINEER: AARON GILROY
PAGE: 1 OF 7
TRUE 1

ZONE	REV	DESCRIPTION	DATE	APPR

REVISIONS

DS3 LINE INTERFACE



DRAWING

TITLE=DS3_LIU
ABBREV=LIU
LAST_MODIFIED=Fri Oct 11 11:49:26 1996

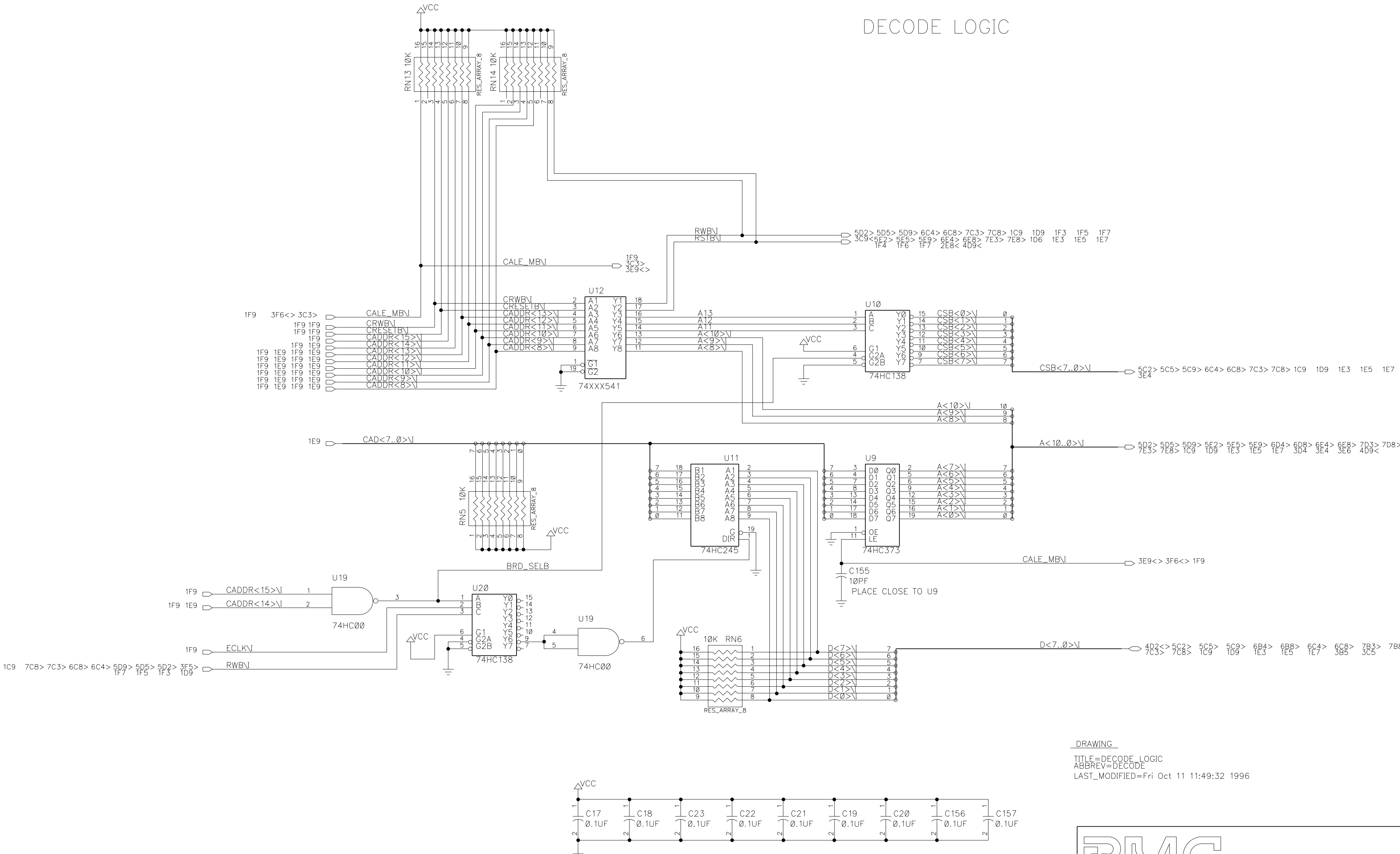
PMC-Sierra, Inc.

DOCUMENT NUMBER: PMC951045	ISSUE: 2
TITLE: D3MX MODULE – M13 REFERENCE DESIGN DS3 LINE INTERFACE	DATE: OCT. 10, 96
ENGINEER: AARON GILROY	PAGE: 2 OF 7

TRUE

ZONE	REV	DESCRIPTION	DATE	APPR

DECODE LOGIC



DECOUPLING CAPS FOR DECODING LOGIC

PLACE DECOUPLING CAPS CLOSE TO THE POWER PINS

DRAWING

TITLE=DECODE_LOGIC
ABBREV=DECODE
LAST_MODIFIED=Fri Oct 11 11:49:32 1996

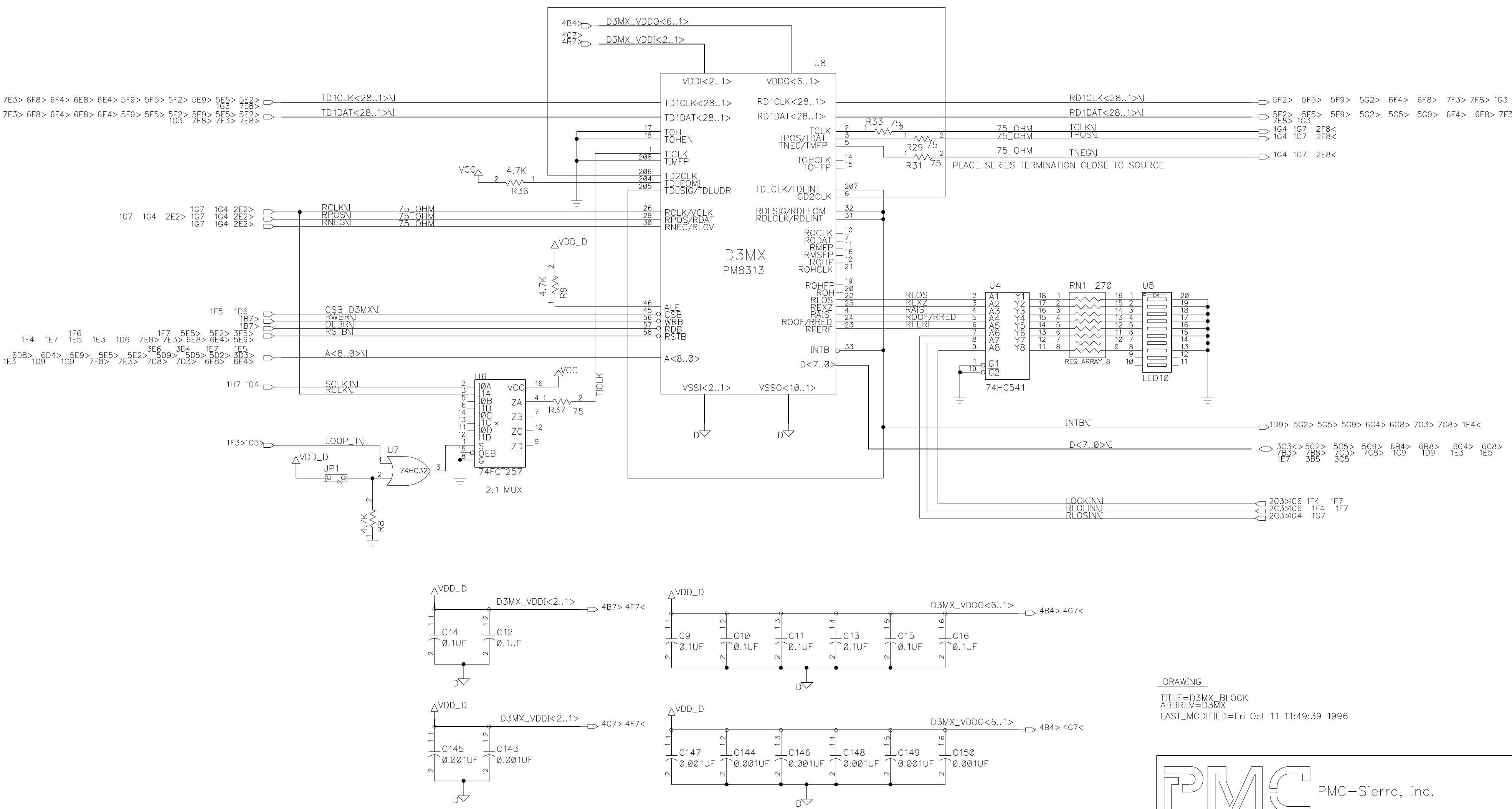
PMC-Sierra, Inc.

DOCUMENT NUMBER: PMC951045	ISSUE: 2
TITLE: D3MX MODULE – M13 REFERENCE DESIGN DECODE LOGIC	DATE: OCT. 10, 96
ENGINEER: AARON GILROY	PAGE: 3 OF 7

TRUE

M13 MULTIPLEXER/DEMULTIPLEXER

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPR



DRAWING

TITLE=D3MX_BLOCK
ABBREV=D3MX
LAST_MODIFIED=Fri Oct 11 11:49:39 1996

PMC-Sierra, Inc.

DOCUMENT NUMBER: PMC951045	ISSUE: 2
TITLE: D3MX MODULE - M13 REFERENCE DESIGN D3MX	DATE: OCT. 10, 96
ENGINEER: AARON GILROY	PAGE: 4 OF 7

TRUE

TRIBUTARY CONNECTORS 1 – 3

ZONE	REV	DESCRIPTION	DATE	APPR

REVISIONS

DRAWING

TITLE=TQUAD_BLOCK
ABBREV=TQUAD
LAST_MODIFIED=Fri Oct 11 11:49:03 1996

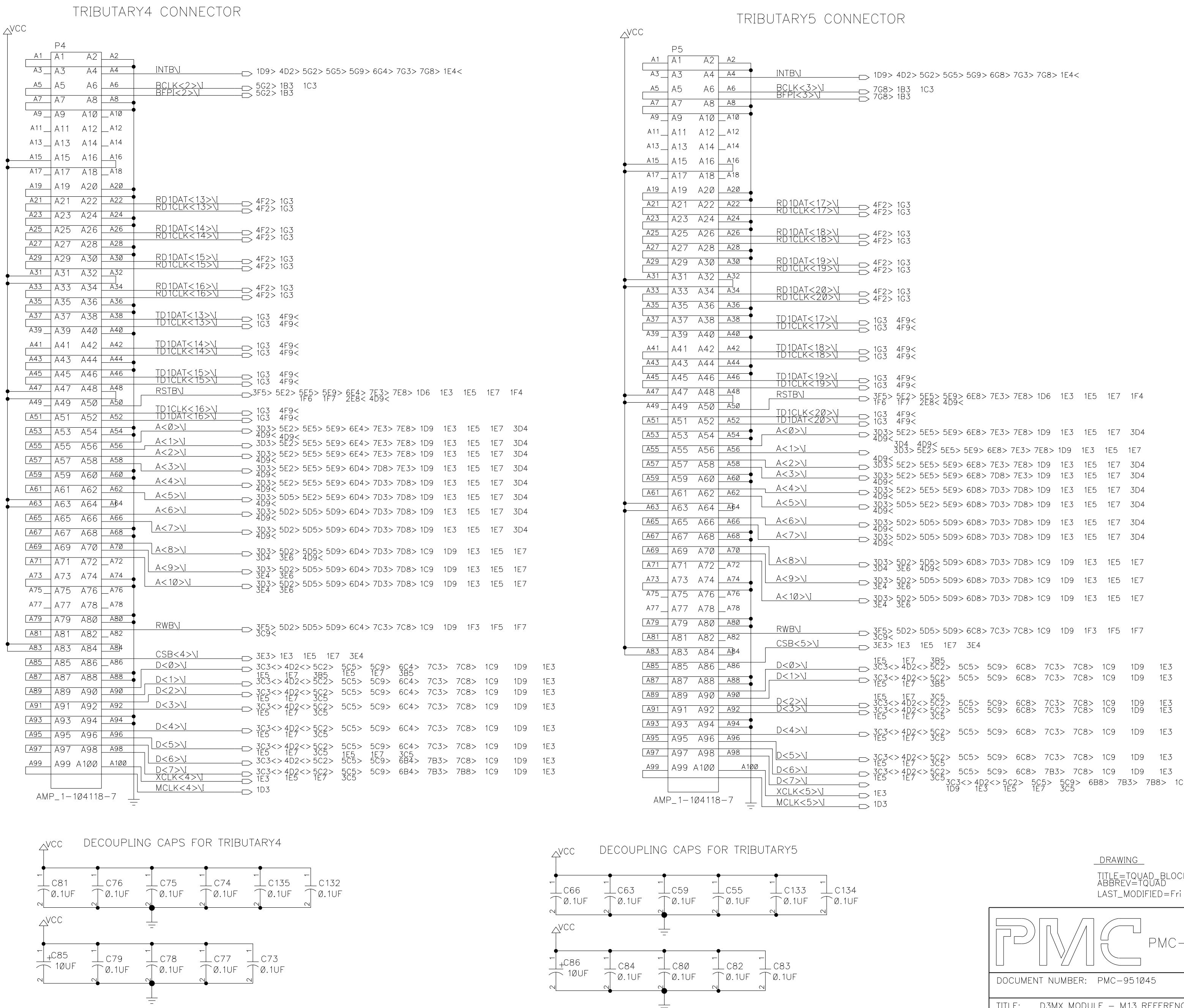
PMC-Sierra, Inc.

DOCUMENT NUMBER: PMC951045	ISSUE: 2
TITLE: D3MX MODULE – M13 REFERENCE DESIGN TQUAD	DATE: OCT. 10, 96
ENGINEER: AARON GILROY	PAGE: 5 OF 7

TRUE

TRIBUTARY CONNECTORS 4 & 5

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPR



DRAWING

TITLE=TQUAD_BLOCK
ABBREV=TQUAD
LAST_MODIFIED=Fri Oct 11 11:49:12 1996

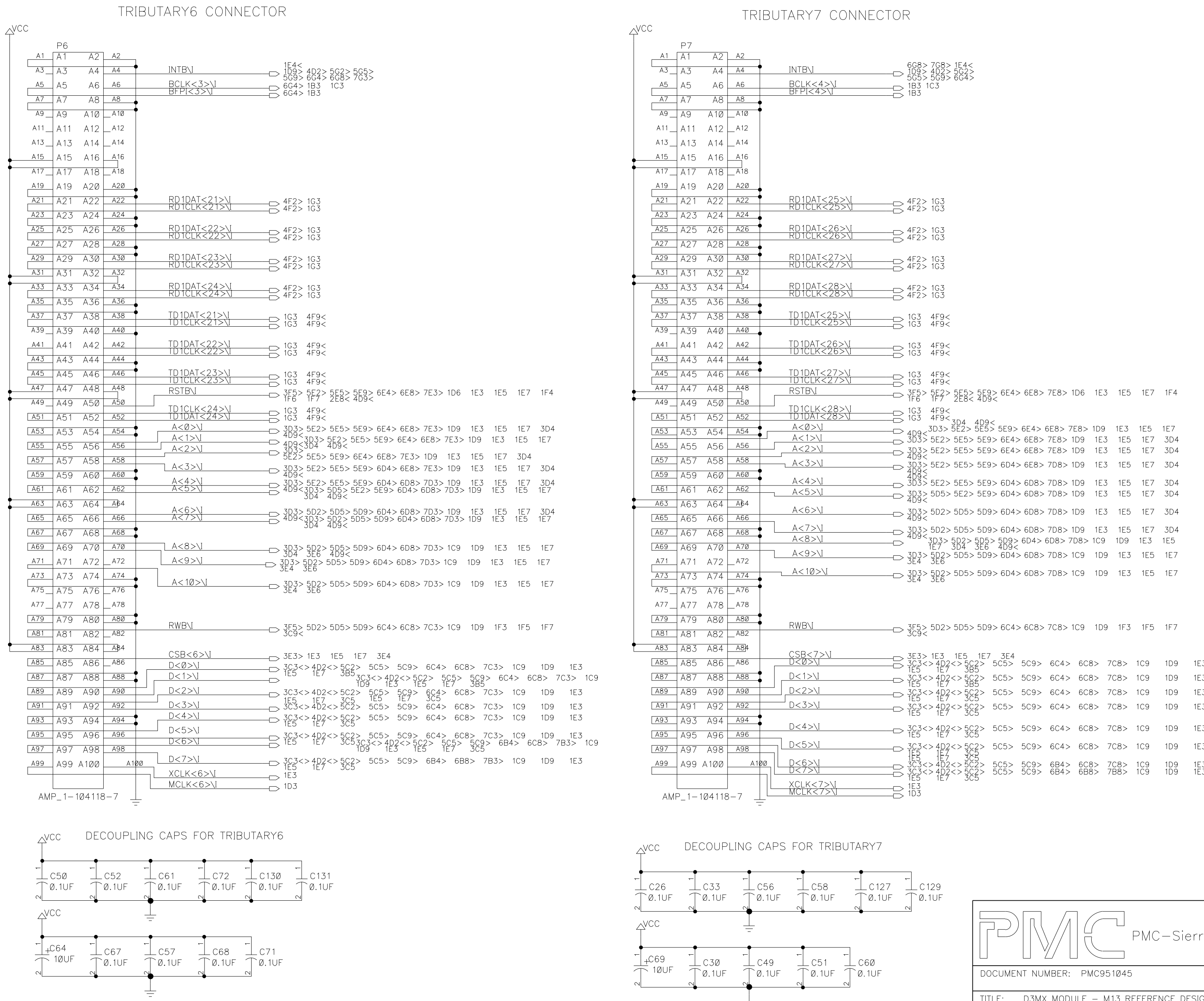
PMC-Sierra, Inc.

DOCUMENT NUMBER: PMC-951045	ISSUE: 2
TITLE: D3MX MODULE - M13 REFERENCE DESIGN TQUAD	DATE: OCT. 10, 96
ENGINEER: AARON GILROY	PAGE: 6 OF 7

TRUE 1

REVISIONS				
ZONE	REV	DESCRIPTION	DATE	APPR

TRIBUTARY CONNECTORS 6 & 7



PMC–Sierra, Inc.

DOCUMENT NUMBER: PMC951045	ISSUE: 2
TITLE: D3MX MODULE – M13 REFERENCE DESIGN TQUAD	DATE: OCT. 10, 96
ENGINEER: AARON GILROY	PAGE: 7 OF 7

TRUE

APPENDIX D: FIRMWARE

This appendix contains the source code for the firmware.

Note: It is the responsibility of the person(s) using or adapting this code to ensure that the resulting system operation complies with both standardized and proprietary requirements.

MACROS.INC file

```
;*******************************************
;* Commonly used general macro definitions *
;*******************************************

; select BANK 0 internal registers
BANK0 MACRO
      BCF   STATUS, RP0       ; select bank 0 (00h-7Fh)
      ENDM              ; END OF BANK0 MACRO

; select BANK 1 internal registers
BANK1 MACRO
      BSF   STATUS, RP0 ; select bank 1 (80h-FFh)
      ENDM              ; END OF BANK1 MACRO

; select program memory PAGE 0
PAGE0 MACRO
      BCF   PCLATH, 3   ; select page 0 (000h-7FFh)
      ENDM              ; END OF PAGE0 MACRO

; select program memory PAGE 1
PAGE1 MACRO
      BSF   PCLATH, 3   ; select page 1 (800h-FFFh)
      ENDM              ; END OF PAGE1 MACRO

; disable all interrupts
INTSOFF     MACRO
      LOCAL CLEAR
CLEAR BCF   INTCON, GIE ; clear global interrupt enable bit
      BTFSC INTCON, GIE ; make sure it was cleared
      GOTO  CLEAR       ; (could have been interrupted)
      ENDM              ; END OF INTSOFF MACRO

; enable all interrupts
INTSON      MACRO
      BSF   INTCON, GIE
      ENDM              ; END OF INTSON MACRO

; skip next instruction if W not equal to VALUE
SNE   MACRO VALUE
      XORLW VALUE       ; compare and skip if not equal
      BTFSC STATUS, Z
      ENDM              ; END OF SNE MACRO
```

```
; skip next instruction if W equal to VALUE
SE    MACRO VALUE
      XORLW VALUE       ; compare and skip if equal
      BTFSS STATUS, Z
      ENDM              ; END OF SE MACRO
```

PIC16C74.INC file

```
        LIST
; P16C74.INC  Standard Header File, Version 1.00    Microchip Technology, Inc.
        NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16C74 microcontroller.  These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included.  The processor may be selected the following ways:

;       1. Command line switch:
;               C:\ MPASM MYFILE.ASM /PIC16C74
;       2. LIST directive in the source file
;               LIST   P=PIC16C74
;       3. Processor Type entry in the MPASM full-screen interface

;==========================================================================
;
;       Revision History
;
;==========================================================================

;Rev:   Date:    Reason:

;1.00   10/31/95 Initial Release

;==========================================================================
;
;       Verify Processor
;
;==========================================================================

        IFNDEF __16C74
           MESSG "Processor-header file mismatch.  Verify selected processor."
        ENDIF

;==========================================================================
;
;       Register Definitions
;
;==========================================================================

W                            EQU     H'0000'
F                            EQU     H'0001'
```

```
;----- Register Files------------------------------------------------------

INDF                          EQU     H'0000'
TMR0                          EQU     H'0001'
PCL                           EQU     H'0002'
STATUS                        EQU     H'0003'
FSR                           EQU     H'0004'
PORTA                         EQU     H'0005'
PORTB                         EQU     H'0006'
PORTC                         EQU     H'0007'
PORTD                         EQU     H'0008'
PORTE                         EQU     H'0009'
PCLATH                        EQU     H'000A'
INTCON                        EQU     H'000B'
PIR1                          EQU     H'000C'
PIR2                          EQU     H'000D'
TMR1L                         EQU     H'000E'
TMR1H                         EQU     H'000F'
T1CON                         EQU     H'0010'
TMR2                          EQU     H'0011'
T2CON                         EQU     H'0012'
SSPBUF                        EQU     H'0013'
SSPCON                        EQU     H'0014'
CCPR1L                        EQU     H'0015'
CCPR1H                        EQU     H'0016'
CCP1CON                       EQU     H'0017'
RCSTA                         EQU     H'0018'
TXREG                         EQU     H'0019'
RCREG                         EQU     H'001A'
CCPR2L                        EQU     H'001B'
CCPR2H                        EQU     H'001C'
CCP2CON                       EQU     H'001D'
ADRES                         EQU     H'001E'
ADCON0                        EQU     H'001F'

OPTION_REG                    EQU     H'0081'
TRISA                         EQU     H'0085'
TRISB                         EQU     H'0086'
TRISC                         EQU     H'0087'
TRISD                         EQU     H'0088'
TRISE                         EQU     H'0089'
PIE1                          EQU     H'008C'
PIE2                          EQU     H'008D'
PCON                          EQU     H'008E'
PR2                           EQU     H'0092'
SSPADD                        EQU     H'0093'
SSPSTAT                       EQU     H'0094'
TXSTA                         EQU     H'0098'
SPBRG                         EQU     H'0099'
ADCON1                        EQU     H'009F'

;----- STATUS Bits --------------------------------------------------------

IRP                           EQU     H'0007'
RP1                           EQU     H'0006'
```

```
RP0                          EQU     H'0005'
NOT_TO                       EQU     H'0004'
NOT_PD                       EQU     H'0003'
Z                            EQU     H'0002'
DC                           EQU     H'0001'
C                            EQU     H'0000'

;----- INTCON Bits --------------------------------------------------------

GIE                          EQU     H'0007'
PEIE                         EQU     H'0006'
T0IE                         EQU     H'0005'
INTE                         EQU     H'0004'
RBIE                         EQU     H'0003'
T0IF                         EQU     H'0002'
INTF                         EQU     H'0001'
RBIF                         EQU     H'0000'

;----- PIR1 Bits ----------------------------------------------------------

PSPIF                        EQU     H'0007'
ADIF                         EQU     H'0006'
RCIF                         EQU     H'0005'
TXIF                         EQU     H'0004'
SSPIF                        EQU     H'0003'
CCP1IF                       EQU     H'0002'
TMR2IF                       EQU     H'0001'
TMR1IF                       EQU     H'0000'

;----- PIR2 Bits ----------------------------------------------------------

CCP2IF                       EQU     H'0000'

;----- T1CON Bits ---------------------------------------------------------

T1CKPS1                      EQU     H'0005'
T1CKPS0                      EQU     H'0004'
T1OSCEN                      EQU     H'0003'
NOT_T1SYNC                   EQU     H'0002'
T1INSYNC                     EQU     H'0002'    ; Backward compatibility only
TMR1CS                       EQU     H'0001'
TMR1ON                       EQU     H'0000'

;----- T2CON Bits ---------------------------------------------------------

TOUTPS3                      EQU     H'0006'
TOUTPS2                      EQU     H'0005'
TOUTPS1                      EQU     H'0004'
TOUTPS0                      EQU     H'0003'
TMR2ON                       EQU     H'0002'
T2CKPS1                      EQU     H'0001'
T2CKPS0                      EQU     H'0000'

;----- SSPCON Bits --------------------------------------------------------
```

```
WCOL                         EQU     H'0007'
SSPOV                        EQU     H'0006'
SSPEN                        EQU     H'0005'
CKP                          EQU     H'0004'
SSPM3                        EQU     H'0003'
SSPM2                        EQU     H'0002'
SSPM1                        EQU     H'0001'
SSPM0                        EQU     H'0000'

;----- CCP1CON Bits -------------------------------------------------------

CCP1X                        EQU     H'0005'
CCP1Y                        EQU     H'0004'
CCP1M3                       EQU     H'0003'
CCP1M2                       EQU     H'0002'
CCP1M1                       EQU     H'0001'
CCP1M0                       EQU     H'0000'

;----- RCSTA Bits ---------------------------------------------------------

SPEN                         EQU     H'0007'
RX9                          EQU     H'0006'
RC9                          EQU     H'0006'    ; Backward compatibility only
NOT_RC8                      EQU     H'0006'    ; Backward compatibility only
RC8_9                        EQU     H'0006'    ; Backward compatibility only
SREN                         EQU     H'0005'
CREN                         EQU     H'0004'
FERR                         EQU     H'0002'
OERR                         EQU     H'0001'
RX9D                         EQU     H'0000'
RCD8                         EQU     H'0000'    ; Backward compatibility only

;----- CCP2CON Bits -------------------------------------------------------

CCP2X                        EQU     H'0005'
CCP2Y                        EQU     H'0004'
CCP2M3                       EQU     H'0003'
CCP2M2                       EQU     H'0002'
CCP2M1                       EQU     H'0001'
CCP2M0                       EQU     H'0000'

;----- ADCON0 Bits --------------------------------------------------------

ADCS1                     EQU     H'0007'
ADCS0                     EQU     H'0006'
CHS2                      EQU     H'0005'
CHS1                      EQU     H'0004'
CHS0                      EQU     H'0003'
GO                        EQU     H'0002'
NOT_DONE                  EQU     H'0002'
GO_DONE                   EQU     H'0002'
ADON                      EQU     H'0000'

;----- OPTION Bits --------------------------------------------------------
```

```
NOT_RBPU                     EQU     H'0007'
INTEDG                       EQU     H'0006'
T0CS                         EQU     H'0005'
T0SE                         EQU     H'0004'
PSA                          EQU     H'0003'
PS2                          EQU     H'0002'
PS1                          EQU     H'0001'
PS0                          EQU     H'0000'

;----- TRISE Bits ---------------------------------------------------------

IBF                          EQU     H'0007'
OBF                          EQU     H'0006'
IBOV                         EQU     H'0005'
PSPMODE                      EQU     H'0004'
TRISE2                       EQU     H'0002'
TRISE1                       EQU     H'0001'
TRISE0                       EQU     H'0000'

;----- PIE1 Bits ----------------------------------------------------------

PSPIE                        EQU     H'0007'
ADIE                         EQU     H'0006'
RCIE                         EQU     H'0005'
TXIE                         EQU     H'0004'
SSPIE                        EQU     H'0003'
CCP1IE                       EQU     H'0002'
TMR2IE                       EQU     H'0001'
TMR1IE                       EQU     H'0000'

;----- PIE2 Bits ----------------------------------------------------------

CCP2IE                       EQU     H'0000'

;----- PCON Bits ----------------------------------------------------------

NOT_POR                      EQU     H'0001'

;----- SSPSTAT Bits -------------------------------------------------------

D                            EQU     H'0005'
I2C_DATA                     EQU     H'0005'
NOT_A                        EQU     H'0005'
NOT_ADDRESS                  EQU     H'0005'
D_A                          EQU     H'0005'
DATA_ADDRESS                 EQU     H'0005'
P                            EQU     H'0004'
I2C_STOP                     EQU     H'0004'
S                            EQU     H'0003'
I2C_START                    EQU     H'0003'
R                            EQU     H'0002'
I2C_READ                     EQU     H'0002'
NOT_W                        EQU     H'0002'
NOT_WRITE                    EQU     H'0002'
R_W                          EQU     H'0002'
```

```
READ_WRITE                   EQU     H'0002'
UA                           EQU     H'0001'
BF                           EQU     H'0000'

;----- TXSTA Bits ----------------------------------------------------------

CSRC                         EQU     H'0007'
TX9                          EQU     H'0006'
NOT_TX8                      EQU     H'0006'    ; Backward compatibility only
TX8_9                        EQU     H'0006'    ; Backward compatibility only
TXEN                         EQU     H'0005'
SYNC                         EQU     H'0004'
BRGH                         EQU     H'0002'
TRMT                         EQU     H'0001'
TX9D                         EQU     H'0000'
TXD8                         EQU     H'0000'    ; Backward compatibility only

;----- ADCON1 Bits ---------------------------------------------------------

PCFG2                     EQU     H'0002'
PCFG1                     EQU     H'0001'
PCFG0                     EQU     H'0000'

;==========================================================================
;
;       RAM Definition
;
;==========================================================================

        __MAXRAM H'FF'
        __BADRAM H'8F'-H'91', H'95'-H'97', H'9A'-H'9E'

;==========================================================================
;
;       Configuration Bits
;
;==========================================================================

_CP_ALL                      EQU     H'3F8F'
_CP_75                       EQU     H'3F9F'
_CP_50                       EQU     H'3FAF'
_CP_OFF                      EQU     H'3FBF'
_PWRTE_ON                    EQU     H'3FBF'
_PWRTE_OFF                   EQU     H'3FB7'
_WDT_ON                      EQU     H'3FBF'
_WDT_OFF                     EQU     H'3FBB'
_LP_OSC                      EQU     H'3FBC'
_XT_OSC                      EQU     H'3FBD'
_HS_OSC                      EQU     H'3FBE'
_RC_OSC                      EQU     H'3FBF'

        LIST
```

D3MX.ASM file

```
;***************************************************************
;* D3MX Module PIC Firmware                                     *
;* Version 1.00 : August 29, 1996                               *
;* Author: Sean Puttergill                                      *
;***************************************************************

      TITLE "ASSEMBLY SOURCE CODE FOR D3MX MODULE"

      PROCESSOR PIC16C74

      INCLUDE "p16c74.inc"
      INCLUDE "macros.inc"

      ERRORLEVEL 0, -306, -302      ; suppress page-crossing and
                              ; argument out of range messages

      __CONFIG    _CP_OFF&_PWRTE_ON&_WDT_OFF&_XT_OSC


;***************************************************************
;*                    CONSTANT DEFINITIONS                      *
;***************************************************************


;**********************
;* Revision Constants *
;**********************

      CONSTANT    VER=0x01    ; version #
      CONSTANT    REV=0x00    ; revision #
      __IDLOCS    0x0000            ; hardcode ver/rev into ID word


;***************************************
;* PIC Microcontroller Register Labels *
;***************************************

; define BANK 0 of microcontroller user registers
; 20 registers used, 76 remaining
      CBLOCK  20
            DATA_REG    ; data register
            ADDR_D3MX_HI      ; D3MX address register MSB
            ADDR_D3MX_LO      ; D3MX address register LSB
            ADDR_RAM_HI ; RAM address register MSB
            ADDR_RAM_LO ; RAM address register LSB
            TEMP_W            ; temporary W reg for context preservation
            TEMP_STATUS ; temporary STATUS reg for context preservation
            TEMP_PCLATH ; temporary PCLATH reg for context preservation
            TIME_COUNT  ; 1 second timer counter
            WORK        ; general working register
                        ; NOT TO BE USED IN INTERRUPT HANDLING ROUTINES
            WORK2       ; general working register
                        ; NOT TO BE USED IN INTERRUPT HANDLING ROUTINES
            WORK_I            ; general working register
```

```
                        ; TO BE USED IN INTERRUPT HANDLING ONLY
            WORK2_I           ; general working register
                        ; TO BE USED IN INTERRUPT HANDLING ONLY
            TIMER_FLAGS ; timer flags
            FLAGS       ; flag register
            XFDL_PKT_LEN      ; XFDL packet length
            XFDL_DATA_PTR     ; XFDL transmit buffer data pointer
            RFDL_DATA_PTR     ; RFDL receive buffer data pointer
            RFDL_LCL_STATUS   ; RFDL local link status byte
            LL_REQ_LINE_BOC   ; line loopback request line BOC
      ENDC

; BANK 1 user registers are unused


;************************
;* D3MX Register Labels *
;************************

      CBLOCK  00
            MSTR_RESET, PMON_UPDATE, MSTR_BYPASS
            MSTR_HDLC_CONFIG, MSTR_LB_CONFIG
            MSTR_IF_CONFIG, MSTR_ALARM_ENABLE
            MSTR_TEST, MSTR_INT_SOURCE_1, MSTR_INT_SOURCE_2
            MSTR_INT_SOURCE_3
      ENDC
      CBLOCK  0C
            TRAN_CONFIG, TRAN_DIAG
      ENDC
      CONSTANT        PMON_IS=11
      CBLOCK  14
            PMON_LCV_LO, PMON_LCV_HI, PMON_FERR_LO
            PMON_FERR_HI, PMON_EXZS_LO, PMON_EXZS_HI
            PMON_PERR_LO, PMON_PERR_HI, PMON_CPERR_LO
            PMON_CPERR_HI, PMON_FEBE_LO, PMON_FEBE_HI
            XFDL_CONFIG, XFDL_IS, XFDL_DATA
      ENDC
      CBLOCK  24
            RFDL_CONFIG, RFDL_IS, RFDL_STATUS, RFDL_DATA
            MX23_CONFIG, MX23_DEMUX_AIS, MX23_MUX_AIS
            MX23_LB_ACT, MX23_LB_REQ_INSERT
            MX23_LB_REQ_DETECT, M23_LB_REQ_INT
      ENDC
      CBLOCK  31
            XBOC_CODE, RBOC_CONFIG, RBOC_IS
            DS3_FRMR_CONFIG, DS3_FRMR_IE, DS3_FRMR_IS
            DS3_FRMR_STATUS
      ENDC
      CBLOCK  40
            DS2_FRMR_CONFIG, DS2_FRMR_IE
            DS2_FRMR_IS, DS2_FRMR_STATUS
            DS2_FRMR_MON_STATUS
            DS2_FRMR_FERR, DS2_FRMR_PERR_LO, DS2_FRMR_PERR_HI
            MX12_CONFIG, MX12_LB_CODE
            MX12_AIS_INSERT, MX12_LB_ACT, MX12_LB_INT
```

```
      ENDC

      ; NOTE: The remaining DS2 channels are not labeled since they
      ;       will be accessed using offsets from the DS2 channel #1


;**********************************
;* Dual Port RAM Register Labels *
;**********************************

      CONSTANT    MAILIN=0x07FF            ; IN mailbox HOST->PIC
      CONSTANT    MAILOUT=0x07FE           ; OUT mailbox PIC->HOST
      CONSTANT    VER_ADDR=0x07FD          ; location to store version #
      CONSTANT    REV_ADDR=0x07FC          ; location to store revision #
      CONSTANT    RAM_ARG_1=0x07FB  ; argument 1 for mailbox command
      CONSTANT    RAM_ARG_2=0x07FA  ; argument 2 for mailbox command
      CONSTANT    RAM_ARG_3=0x07F9  ; argument 3 for mailboc command
      CONSTANT    RAM_DATA_RETURN=0x07F8  ; storage for returning
                                    ; data read from D3MX
      CONSTANT    RAM_RLOL=0x07F7          ; RLOL pin mirror
      CONSTANT    RAM_LOCK=0x07F6          ; LOCK pin mirror
      CONSTANT    RAM_RFDL_BUFFER=0x0000  ; RFDL receive buffer
      CONSTANT    RAM_RFDL_STATUS=0x007E  ; RFDL link status
      CONSTANT    RAM_RFDL_PKT_LEN=0x007F ; RFDL packet length
      CONSTANT    RAM_XFDL_BUFFER=0x0080  ; XFDL transmit buffer
      CONSTANT    RAM_XFDL_PKT_LEN=0x00FF ; XFDL packet length
      CONSTANT    RAM_PMON_SHADOW=0x0100  ; PMON Shadow Registers begin
      CONSTANT    RAM_PMON_LCV_LO=0x0100  ; PMON LCV count LSB
      CONSTANT    RAM_PMON_LCV_HI=0x0101  ; PMON LCV count MSB
      CONSTANT    RAM_PMON_FERR_LO=0x0102 ; PMON FERR count LSB
      CONSTANT    RAM_PMON_FERR_HI=0x0103 ; PMON FERR count MSB
      CONSTANT    RAM_PMON_EXZS_LO=0x0104 ; PMON EXZS count LSB
      CONSTANT    RAM_PMON_EXZS_HI=0x0105 ; PMON EXZS count MSB
      CONSTANT    RAM_PMON_PERR_LO=0x0106 ; PMON PERR count LSB
      CONSTANT    RAM_PMON_PERR_HI=0x0107 ; PMON PERR count MSB
      CONSTANT    RAM_PMON_CPERR_LO=0x0108 ; PMON CPERR count LSB
      CONSTANT    RAM_PMON_CPERR_HI=0x0109 ; PMON CPERR count MSB
      CONSTANT    RAM_PMON_FEBE_LO=0x010A ; PMON FEBE count LSB
      CONSTANT    RAM_PMON_FEBE_HI=0x010B ; PMON FEBE count MSB
      CONSTANT    RAM_PMON_DS2_FERR=0x010C ; PMON DS2 FERR count
      CONSTANT    RAM_RBOC=0x0121          ; RBOC received BOC
      CONSTANT    RAM_DS2_ID=0x0122 ; interrupting DS2
      CONSTANT    RAM_LL_ID=0x0123  ; line which has had loopback
                                    ; activated or deactivated


;**************
;* Bit Labels *
;**************

      ; PORTA
      CONSTANT    CSB1=3            ; D3MX CSB pin
      CONSTANT    CSB2=4            ; RAM CSB pin
      CONSTANT    LOCK=5            ; LOCK pin
```

```
; PORTB
CONSTANT    INTB1=0            ; D3MX int pin
CONSTANT    RLOL=2             ; RLOL pin
CONSTANT    LED1=3             ; LED #1 bit
CONSTANT    INTB2=4            ; RAM int pin
CONSTANT    LED2=5             ; LED #2 bit
CONSTANT    LED3=6             ; LED #3 bit
CONSTANT    LED4=7             ; LED #4 bit

; PORTE
CONSTANT    RDB=0       ; RDB pin
CONSTANT    WRB=1       ; WRB pin
CONSTANT    LOOP_T=2    ; LOOP_T pin

; TIMER_FLAGS
CONSTANT    ONE_SEC=0   ; one second flag
CONSTANT    LL_REQ_TX_TMR1=1  ; timer bit 1 for line loopback
                              ; request transmission
CONSTANT    LL_REQ_TX_TMR2=2  ; timer bit 2 for line loopback
                              ; request transmission

; FLAGS
CONSTANT    RFDL_ACTIVE=0     ; RFDL active bit
CONSTANT    LL_ACTIVATE=1     ; line loopback activate code
                        ; received flag
CONSTANT    LL_DEACTIVATE=2   ; line loopback deactivate code
                        ; received flag

; Master Interrupt Source #1
CONSTANT    MIS1_RBOC=0 ; RBOC interrupt flag
CONSTANT    MIS1_RFDLINT=2    ; RFDL interrupt flag
CONSTANT    MIS1_DS3FRMR=3    ; DS3FRMR interrupt flag
CONSTANT    MIS1_MX23=4 ; MX23 interrupt flag
CONSTANT    MIS1_XFDLINT=5    ; XFDL interrupt flag
CONSTANT    MIS1_REG3=6 ; reg3 bit
CONSTANT    MIS1_REG2=7 ; reg2 bit

; XFDL Interrupt Status
CONSTANT    XFDL_UDR=0  ; XFDL underrun flag
CONSTANT    XFDL_INT=1  ; XFDL new byte flag

; XFDL Configuration
CONSTANT    XFDL_INTE=3 ; XFDL interrupt enable
CONSTANT    XFDL_EOM=4  ; XFDL end of message flag

; RFDL Interrupt Status
CONSTANT    RFDL_INT=0  ; RFDL interrupt flag

; RFDL Status and RFDL_LCL_STATUS
CONSTANT    RFDL_NVB0=0 ; RFDL NVB0 bit
CONSTANT    RFDL_NVB1=1 ; RFDL NVB1 bit
CONSTANT    RFDL_NVB2=2 ; RFDL NVB2 bit
CONSTANT    RFDL_CRC=3  ; RFDL CRC bit
CONSTANT    RFDL_EOM=4  ; RFDL end of message flag
CONSTANT    RFDL_FLG=5  ; RFDL FLG bit
```

```
      CONSTANT    RFDL_OVR=6  ; RFDL overrun flag
      CONSTANT    RFDL_FE=7   ; RFDL FIFO empty flag

      ; RBOC Interrupt Status
      CONSTANT    RBOC_IDLEI=7      ; RBOC idle interrupt flag
      CONSTANT    RBOC_BOCI=6 ; RBOC valid BOC interrupt flag

      ; DS3 FRMR Interrupt Status
      CONSTANT    DS3FIS_LOSI=0     ; DS3 FRMR LOS interrupt flag
      CONSTANT    DS3FIS_AISI=2     ; DS3 FRMR AIS interrupt flag
      CONSTANT    DS3FIS_IDLI=3     ; DS3 FRMR IDLE interrupt flag
      CONSTANT    DS3FIS_REDI=6     ; DS3 FRMR RED alarm interrupt flag

      ; DS3 FRMR Status
      CONSTANT    DS3FS_LOSV=0      ; DS3 FRMR LOS flag
      CONSTANT    DS3FS_AISV=2      ; DS3 FRMR AIS flag
      CONSTANT    DS3FS_IDLV=3      ; DS3 FRMR IDLE flag
      CONSTANT    DS3FS_REDV=6      ; DS3 FRMR RED alarm flag

      ; DS2 FRMR Interrupt Status
      CONSTANT    DS2FIS_AISI=2     ; DS2 FRMR AIS interrupt flag
      CONSTANT    DS2FIS_REDI=5     ; DS2 FRMR RED alarm interrupt flag

      ; DS2 FRMR Status
      CONSTANT    DS2FS_AISV=2      ; DS2 FRMR AIS flag
      CONSTANT    DS2FS_REDV=5      ; DS2 FRMR RED alarm flag

      ; Master Loopback Configuration
      CONSTANT    MLC_LLBE=1  ; master loopback config ll enable flag


;*****************
;* Mailbox Codes *
;*****************

      ; Host->PIC
      CONSTANT    RD_REG_CMD=0x01         ; read D3MX register command
      CONSTANT    WR_REG_CMD=0x02         ; write D3MX register command
      CONSTANT    XFDL_START_CMD=0x03     ; XFDL packet start command
      CONSTANT    START_BOC_CMD=0x04      ; BOC start command
      CONSTANT    STOP_BOC_CMD=0x05 ; BOC stop command
      CONSTANT    LLA_REQ_TX_CMD=0x06     ; ll activate request tx command
      CONSTANT    LLD_REQ_TX_CMD=0x07     ; ll deactivate request tx
                                    ; command
      CONSTANT    LOOPT_HIGH_CMD=0x08     ; set LOOP_T high command
      CONSTANT    LOOPT_LOW_CMD=0x09      ; set LOOP_T low command

      ; PIC->Host
      CONSTANT    IO_DONE=0x01      ; D3MX I/O operation complete
      CONSTANT    PMON_UPDATED=0x02 ; PMON shadow registers updated
      CONSTANT    XFDL_DONE=0x03    ; finished transmitting FDL packet
      CONSTANT    RFDL_NEW=0x04     ; new FDL packet received
      CONSTANT    RBOC_NEW=0x05     ; new BOC received on FEAC
      CONSTANT    RBOC_IDLE=0x06    ; FEAC has gone idle
      CONSTANT    DS3_AIS_A=0x07    ; DS3 AIS asserted
```

```
      CONSTANT    DS3_AIS_C=0x08    ; DS3 AIS cleared
      CONSTANT    DS3_RED_A=0x09    ; DS3 RED alarm asserted
      CONSTANT    DS3_RED_C=0x0A    ; DS3 RED alarm cleared
      CONSTANT    DS3_IDL_A=0x0B    ; DS3 idle asserted
      CONSTANT    DS3_IDL_C=0x0C    ; DS3 idle cleared
      CONSTANT    DS2_AIS_A=0x0D    ; DS2 AIS asserted
      CONSTANT    DS2_AIS_C=0x0E    ; DS2 AIS cleared
      CONSTANT    DS2_RED_A=0x0F    ; DS2 RED alarm asserted
      CONSTANT    DS2_RED_C=0x10    ; DS2 RED alarm cleared
      CONSTANT    LL_ACTIVATED=0x13 ; line loopback has been
                              ; activated upon reception of
                              ; a loopback request
      CONSTANT    LL_DEACTIVATED=0x14 ; line loopback has been
                                ; deactivated upon reception of
                                ; a loopback request


;**********************
;* Bit Oriented Codes *
;**********************

      CONSTANT    BOC_YELLOW=0x00         ; Yellow alarm BOC
      CONSTANT    BOC_LL_ACTIVATE=0x07    ; line loopback activate BOC
      CONSTANT    BOC_LL_DEACTIVATE=0x1C  ; line loopback deactivate BOC
      CONSTANT    BOC_LL_DS1ALL=0x13      ; ll all DS1s BOC
      CONSTANT    BOC_LL_DS3=0x1B         ; ll DS3 BOC
      CONSTANT    BOC_IDLE=0x3F           ; idle BOC


;*******************
;* Other Constants *
;*******************

      CONSTANT        ONE_SECOND=0x4C     ; 1 second counter reload value
                            ; for timer 0
      ; 1SEC=50NS(OSC Period)*4(Div/4)*256(Prescaler)*256(8bit cnt)*76
      CONSTANT    T2_PERIOD=0xB7    ; timer 2 period register reload value,
                            ; 1/2 the time taken to transmit a
                            ; FEAC 16 bit codeword 11 times
                            ; (to ensure code is transmitted
                            ;  10 times, must wait 11 periods)
      CONSTANT    MAX_XFDL_PKT_LEN=0x7D   ; 1 more than maximum length
                                  ; of an HDLC packet


;*************************************************************
;*                        MACRO DEFINITIONS                  *
;*************************************************************

WR_RAM      MACRO ADDRESS, VALUE

; FUNCTION: Write value to RAM
; TAKES:    ADDRESS, VALUE
; RETURNS:  nothing
; ASSUMES:  Page bit is set to 0
```

```
      MOVLW VALUE       ; move value to data register
      MOVWF DATA_REG
      MOVLW LOW ADDRESS ; move LSB of address to LSB of RAM address reg
      MOVWF ADDR_RAM_LO
      MOVLW HIGH ADDRESS      ; move MSB of address to MSB of RAM address reg
      MOVWF ADDR_RAM_HI
      CALL  WRITE_RAM   ; write to RAM
      ENDM              ; END OF WR_RAM   MACRO


WR_RAM_D    MACRO ADDRESS

; FUNCTION: Write DATA_REG to RAM
; TAKES:    ADDRESS
; RETURNS:  nothing
; ASSUMES:  Page bit is set to 0

      MOVLW LOW ADDRESS ; move LSB of address to LSB of RAM address reg
      MOVWF ADDR_RAM_LO
      MOVLW HIGH ADDRESS      ; move MSB of address to MSB of RAM address reg
      MOVWF ADDR_RAM_HI
      CALL  WRITE_RAM   ; write to RAM
      ENDM              ; END OF WR_RAM_D MACRO


WR_D3MX     MACRO REGISTER, VALUE

; FUNCTION: Write value to D3MX normal mode register
; TAKES:    REGISTER, VALUE
; RETURNS:  nothing
; ASSUMES:  Page bit is set to 0

      MOVLW VALUE       ; write to D3MX register
      MOVWF DATA_REG    ; copy data byte to DATA_REG
      MOVLW LOW REGISTER      ; copy register addr to LSB of D3MX addr reg
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI      ; clear MSB of D3MX address reg
      CALL  WRITE_D3MX
      ENDM              ; END OF WR_D3MX MACRO


WR_D3MX_D   MACRO REGISTER

; FUNCTION: Write DATA_REG to D3MX normal mode register
; TAKES:    REGISTER
; RETURNS:  nothing
; ASSUMES:  Page bit is set to 0

      MOVLW LOW REGISTER
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI      ; clear MSB of D3MX address reg
      CALL  WRITE_D3MX
      ENDM              ; END OF WR_D3MX_D MACRO
```

```
RD_RAM      MACRO ADDRESS

; FUNCTION: Read from RAM
; TAKES:    ADDRESS
; RETURNS:  value at RAM location in both W and DATA_REG registers
; ASSUMES:  Page bit is set to 0

      MOVLW LOW ADDRESS     ; read RAM
      MOVWF ADDR_RAM_LO
      MOVLW HIGH ADDRESS
      MOVWF ADDR_RAM_HI
      CALL  READ_RAM
      ENDM              ; END OF RD_RAM MACRO


RD_D3MX     MACRO REGISTER

; FUNCTION: Read from D3MX
; TAKES:    REGISTER
; RETURNS:  value in D3MX register in both W and DATA_REG registers
; ASSUMES:  Page bit is set to 0

      MOVLW LOW REGISTER      ; read D3MX register
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI      ; clear MSB of D3MX address reg
      CALL  READ_D3MX
      ENDM              ; END OF RD_D3MX MACRO


;*************************************************************
;*                         SOURCE                            *
;*************************************************************

;*****************
;* SET UP VECTORS *
;*****************

; RESET vector
      ORG   0x0000
      GOTO  INIT        ; initialize on reset

; interrupt vector
      ORG   0x0004
      GOTO  INTHDLR           ; point to INTHDLR


;*****************
;* INITIALIZATION *
;*****************

INIT
      CALL  INIT_MICRO  ; initialize microcontroller
      CALL  INIT_RAM    ; initialize dual port RAM
      CALL  INIT_D3MX   ; initialize D3MX
```

```
      CALL  ENABLE_INTS ; enable interrupts
      GOTO  MAIN_LOOP       ; enter main processing loop


;*************
;* MAIN LOOP *
;*************

MAIN_LOOP
      BTFSC TIMER_FLAGS, ONE_SEC    ; poll 1 second timer flag
      CALL  PMON               ; update PMON shadow registers
                               ; if flag is set
      CLRWDT                         ; clear watchdog timer
      GOTO  MAIN_LOOP


;*****************************
;* INIT SUBROUTINE FOR MICRO *
;*****************************

INIT_MICRO
      BANK0             ; switch to BANK0

      ; initialize ports
      MOVLW 0x18
      MOVWF PORTA       ; make RAM and D3MX CSB pins inactive,
                        ; clear A8-A10
      MOVLW 0x00        ; all LEDs off
      MOVWF PORTB
      CLRF  PORTC       ; clear A0-A7
      CLRF  PORTD       ; clear D0-D7
      MOVLW 0x07
      MOVWF PORTE       ; make RDB and WRB inactive,
                        ; LOOP_T high

      ; initialize user registers
      CLRF  DATA_REG
      CLRF  ADDR_D3MX_HI
      CLRF  ADDR_D3MX_LO
      CLRF  ADDR_RAM_HI
      CLRF  ADDR_RAM_LO
      CLRF  TEMP_W
      CLRF  TEMP_STATUS
      CLRF  TEMP_PCLATH
      CLRF  WORK
      CLRF  WORK2
      CLRF  WORK_I
      CLRF  WORK2_I
      CLRF  TIMER_FLAGS
      CLRF  FLAGS
      CLRF  XFDL_PKT_LEN
      CLRF  XFDL_DATA_PTR
      CLRF  RFDL_DATA_PTR
      CLRF  RFDL_LCL_STATUS
      CLRF  LL_REQ_LINE_BOC
```

```
      MOVLW ONE_SECOND  ; load 1s counter
      MOVWF TIME_COUNT

      ; configure ports
      BANK1             ; switch to BANK1
      MOVLW 0x07        ; configure PORTA as digital
      MOVWF ADCON1
      MOVLW 0x07        ; set OPTION and TMR0 prescalar to 1:256
      MOVWF OPTION_REG
      MOVLW 0x20        ; configure LOCKIN as input
      MOVWF TRISA
      MOVLW 0x13        ; configure INTB1, INTB2 and BUSY as inputs
      MOVWF TRISB
      MOVLW 0x00        ; configure address bus as output
      MOVWF TRISC
      MOVLW 0xFF        ; configure data bus as input
      MOVWF TRISD
      MOVLW 0x00        ; configure RDB, WRB and LOOP_T as outputs
      MOVWF TRISE
      BANK0             ; *NB* should switch back to this as default
                        ; everywhere in the code.  As long as this
                        ; convention is maintained then it is not
                        ; neccesary to switch to BANK0 at the
                        ; beginning of macros and subroutines.
      MOVLW 0x7B        ; setup timer 2 - prescaler=1:16,
      MOVWF T2CON       ; postscaler=1:16, timer off

      MOVF  PORTB, F    ; read PORTB to initialize latch value for RBIF

      RETURN                  ; END OF INIT_MICRO SUBROUTINE


;***************************
;* INIT SUBROUTINE FOR RAM *
;***************************

INIT_RAM
      RD_RAM      MAILIN            ; read IN mailbox to clear any interrupt
      WR_RAM      VER_ADDR, VER     ; copy version # to RAM
      WR_RAM      REV_ADDR, REV     ; copy revision # to RAM

      RETURN                  ; END OF INIT_RAM SUBROUTINE


;****************************
;* INIT SUBROUTINE FOR D3MX *
;****************************

INIT_D3MX
      WR_D3MX     MSTR_RESET, 0x01  ; software reset D3MX
      WR_D3MX     MSTR_RESET, 0x00
      WR_D3MX     MSTR_HDLC_CONFIG, 0x00  ; use internal HDLC controllers
      WR_D3MX     MSTR_IF_CONFIG, 0x09    ; select clock edges
      WR_D3MX     MSTR_ALARM_ENABLE, 0x8C ; propagates AIS to demux
      WR_D3MX     TRAN_CONFIG, 0x01 ; transmit C-BIT parity
```

```
      WR_D3MX     MX23_CONFIG, 0x02 ; mux C-BIT parity
      WR_D3MX     DS3_FRMR_CONFIG, 0x83   ; frame to C-BIT parity
      WR_D3MX     XFDL_CONFIG, 0x03 ; enable XFDL block with frame check
                              ; sequence generation
      WR_D3MX     RFDL_IS, 0x02           ; enable interrupt generation on
                              ; reception of data
      WR_D3MX     RFDL_CONFIG, 0x01 ; enable RFDL block
      WR_D3MX     DS3_FRMR_IE, 0x4D ; enable LOS, AIS, IDLE and RED alarm
                              ; interrupts
      WR_D3MX     RBOC_CONFIG, 0x05 ; enable RBOC interrupts when new BOC
                              ; validated and upon transition to
                              ; idle code

      MOVLW DS2_FRMR_IE       ; initialize DS2 FRMR IE register
      MOVWF ADDR_D3MX_LO            ; enable AIS, reserved bit and
      CLRF  ADDR_D3MX_HI            ; RED alarm interrupts
      MOVLW 0x24
      MOVWF DATA_REG
      MOVLW 0x07
      MOVWF WORK
INIT_DS2_LOOP
      CALL  WRITE_D3MX
      MOVLW 0x10
      ADDWF ADDR_D3MX_LO, F
      DECFSZ      WORK, F
      GOTO  INIT_DS2_LOOP

      RETURN                        ; END OF INIT_RAM SUBROUTINE


;**************************************
;* SUBROUTINE FOR ENABLING INTERRUPTS *
;**************************************

ENABLE_INTS
      BANK1
      CLRF  PIE1        ; disable all peripheral ints
      CLRF  PIE2
      BANK0
      MOVLW 0xF8        ; enable timer (TMR0), RAM (RBI)
                        ; peripheral (PEIE) and D3MX (INT) interrupts,
                        ; and set GIE
      MOVWF INTCON

      RETURN                  ; END OF ENABLE_INTS SUBROUTINE


;***************************
;* READ SUBROUTINE FOR D3MX *
;***************************

; FUNCTION: Performs read cycle with D3MX
; TAKES:    nothing
; RETURNS:  Data in D3MX register in DATA_REG and W
; ASSUMES:  nothing
```

```
READ_D3MX
      ; setup address bus
      MOVF  ADDR_D3MX_LO, W         ; transfer D3MX address to latch
      MOVWF PORTC
      BSF   PORTA, 0
      BTFSS ADDR_D3MX_HI, 0
      BCF   PORTA, 0

      ; perform read by toggling CSB1 and RDB
      BCF   PORTA, CSB1 ; activate CSB1
      BCF   PORTE, RDB  ; activate RDB
      MOVF  PORTD, W    ; latch data
      MOVWF DATA_REG    ; copy to DATA_REG
      BSF   PORTE, RDB  ; deactivate RDB
      BSF   PORTA, CSB1 ; deactivate CSB1

      RETURN                  ; END OF READ_D3MX SUBROUTINE


;***************************
;* READ SUBROUTINE FOR RAM *
;***************************

; FUNCTION: Performs read cycle with RAM
; TAKES:    nothing
; RETURNS:  Data in RAM location in DATA_REG and W
; ASSUMES:  nothing

READ_RAM
      ; setup address bus
      MOVF  ADDR_RAM_LO, W          ; transfer RAM address to latch
      MOVWF PORTC
      MOVLW 0xF8
      ANDWF PORTA, F                ; clear A8-A10
      MOVF  ADDR_RAM_HI, W          ; load W with A8-A10
      ANDLW 0x07                    ; mask off unused bits
      IORWF PORTA, F                ; insert A8-A10 into PORTA

      ; perform read by toggling CSB2 and RDB
      BCF   PORTA, CSB2 ; activate CSB2
      BCF   PORTE, RDB  ; activate RDB
      MOVF  PORTD, W    ; latch data
      MOVWF DATA_REG    ; copy to DATA_REG
      BSF   PORTE, RDB  ; deactivate RDB
      BSF   PORTA, CSB2 ; deactivate CSB2

      RETURN                  ; END OF READ_RAM SUBROUTINE


;*****************************
;* WRITE SUBROUTINE FOR D3MX *
;*****************************

; FUNCTION: Performs write cycle with D3MX
```

```
; TAKES:    Data byte in DATA_REG
; RETURNS:  nothing
; ASSUMES:  nothing

WRITE_D3MX
      ; activate data bus as output
      BANK1
      MOVLW 0x00
      MOVWF TRISD
      BANK0       ; select as default

      ; setup address bus
      MOVF  ADDR_D3MX_LO, W         ; transfer D3MX address to latch
      MOVWF PORTC
      BSF   PORTA, 0
      BTFSS ADDR_D3MX_HI, 0
      BCF   PORTA, 0

      ; setup data bus
      MOVF  DATA_REG, W       ; transfer DATA_REG to latch
      MOVWF PORTD

      ; perform write by toggling CSB1 and WRB
      BCF   PORTA, CSB1 ; activate CSB1
      BCF   PORTE, WRB  ; activate WRB
      BSF   PORTE, WRB  ; deactivate WRB
      BSF   PORTA, CSB1 ; deactivate CSB1

      ; tristate data bus
      BANK1
      MOVLW 0xFF
      MOVWF TRISD
      BANK0           ; select as default

      RETURN          ; END OF WRITE_D3MX SUBROUTINE


;****************************
;* WRITE SUBROUTINE FOR RAM *
;****************************

; FUNCTION: Performs write cycle with RAM
; TAKES:    Data byte in DATA_REG
; RETURNS:  nothing
; ASSUMES:  nothing

WRITE_RAM
      ; activate data bus as output
      BANK1
      MOVLW 0x00
      MOVWF TRISD
      BANK0       ; select as default

      ; setup address bus
      MOVF  ADDR_RAM_LO, W    ; transfer RAM address to latch
```

```
      MOVWF PORTC
      MOVLW 0xF8
      ANDWF PORTA, F    ; clear A8-A10
      MOVF  ADDR_RAM_HI, W    ; load W with A8-A10
      ANDLW 0x07        ; mask off unused bits
      IORWF PORTA, F    ; insert A8-A10 into PORTA

      ; setup data bus
      MOVF  DATA_REG, W ; transfer DATA_REG to latch
      MOVWF PORTD

      ; perform write by toggling CSB2 and WRB
      BCF   PORTA, CSB2 ; activate CSB2
      BCF   PORTE, WRB  ; activate WRB
      BSF   PORTE, WRB  ; deactivate WRB
      BSF   PORTA, CSB2 ; deactivate CSB2

      ; tristate data bus
      BANK1
      MOVLW 0xFF
      MOVWF TRISD
      BANK0       ; select as default

      RETURN          ; END OF WRITE_RAM SUBROUTINE


;*************************************
;* PERFORMANCE MONITORING SUBROUTINE *
;*************************************

; FUNCTION: Copies performance monitoring statistics from D3MX registers
;           to RAM shadow registers
; TAKES:    nothing
; RETURNS:  nothing
; ASSUMES:  nothing

PMON
      BCF   TIMER_FLAGS, ONE_SEC    ; clear 1 second flag
      INTSOFF
      WR_D3MX     PMON_UPDATE, 0x00 ; write to D3MX to trigger
                              ; update of PMON counters
      INTSON

      ; delay for 6us to allow for PMON update latency
      MOVLW 0A
      MOVWF WORK
PMON_DELAY
      DECFSZ      WORK, F
      GOTO  PMON_DELAY

      ; copy DS3 PMON registers from D3MX to RAM
      INTSOFF
      RD_D3MX     PMON_LCV_LO
      WR_RAM_D    RAM_PMON_LCV_LO
      INTSON
```

```
INTSOFF
RD_D3MX      PMON_LCV_HI
WR_RAM_D     RAM_PMON_LCV_HI
INTSON
INTSOFF
RD_D3MX      PMON_FERR_LO
WR_RAM_D     RAM_PMON_FERR_LO
INTSON
INTSOFF
RD_D3MX      PMON_FERR_HI
MOVLW 0x03                    ; mask out don't care bits
ANDWF DATA_REG, F
WR_RAM_D     RAM_PMON_FERR_HI
INTSON
INTSOFF
RD_D3MX      PMON_EXZS_LO
WR_RAM_D     RAM_PMON_EXZS_LO
INTSON
INTSOFF
RD_D3MX      PMON_EXZS_HI
WR_RAM_D     RAM_PMON_EXZS_HI
INTSON
INTSOFF
RD_D3MX      PMON_PERR_LO
WR_RAM_D     RAM_PMON_PERR_LO
INTSON
INTSOFF
RD_D3MX      PMON_PERR_HI
MOVLW 0x3F                    ; mask out don't care bits
ANDWF DATA_REG, F
WR_RAM_D     RAM_PMON_PERR_HI
INTSON
INTSOFF
RD_D3MX      PMON_CPERR_LO
WR_RAM_D     RAM_PMON_CPERR_LO
INTSON
INTSOFF
RD_D3MX      PMON_CPERR_HI
MOVLW 0x3F                    ; mask out don't care bits
ANDWF DATA_REG, F
WR_RAM_D     RAM_PMON_CPERR_HI
INTSON
INTSOFF
RD_D3MX      PMON_FEBE_LO
WR_RAM_D     RAM_PMON_FEBE_LO
INTSON
INTSOFF
RD_D3MX      PMON_FEBE_HI
MOVLW 0x3F                    ; mask out don't care bits
ANDWF DATA_REG, F
WR_RAM_D     RAM_PMON_FEBE_HI
INTSON

; setup to loop through DS2 PMON registers
MOVLW LOW RAM_PMON_DS2_FERR
```

```
      MOVWF WORK
      MOVLW LOW DS2_FRMR_FERR
      MOVWF WORK2

      ; copy DS2 PMON registers from D3MX to RAM
PMON_DS2_LOOP
      INTSOFF
      MOVF  WORK, W
      MOVWF ADDR_RAM_LO
      MOVLW HIGH RAM_PMON_SHADOW
      MOVWF ADDR_RAM_HI
      MOVF  WORK2, W
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI
      CALL  READ_D3MX
      CALL  WRITE_RAM
      INTSON
      INCF  WORK, F
      INCF  WORK2, F
      INTSOFF
      MOVF  WORK, W
      MOVWF ADDR_RAM_LO
      MOVLW HIGH RAM_PMON_SHADOW
      MOVWF ADDR_RAM_HI
      MOVF  WORK2, W
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI
      CALL  READ_D3MX
      CALL  WRITE_RAM
      INTSON
      INCF  WORK, F
      INCF  WORK2, F
      INTSOFF
      MOVF  WORK, W
      MOVWF ADDR_RAM_LO
      MOVLW HIGH RAM_PMON_SHADOW
      MOVWF ADDR_RAM_HI
      MOVF  WORK2, W
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI
      CALL  READ_D3MX
      MOVLW 0x1F                  ; mask out don't care bits
      ANDWF DATA_REG, F
      CALL  WRITE_RAM
      INTSON
      INCF  WORK, F
      MOVLW 0x0E
      ADDWF WORK2, F
      MOVF  WORK, W
      SE    0x21
      GOTO  PMON_DS2_LOOP         ; loop until all DS2 PMON regs copied

      INTSOFF
      WR_RAM      MAILOUT, PMON_UPDATED   ; signal update to HOST
      INTSON
```

```
      RETURN          ; END OF PMON SUBROUTINE


;******************************
;* INTERRUPT HANDLING ROUTINE *
;******************************

; FUNCTION: Determines source of interrupt and calls corresponding service
;           routine.  D3MX interrupts are continually processed until none
;           are pending.
; TAKES:    nothing
; RETURNS:  nothing
; ASSUMES:  Only branched to from interrupt vector.

INTHDLR
      ; save context
      MOVWF TEMP_W
      SWAPF STATUS, W
      BANK0             ; switch to BANK0, necessary here because we
                        ; could be in either bank currently
      MOVWF TEMP_STATUS
      MOVF  PCLATH, W
      MOVWF TEMP_PCLATH

DET_SOURCE
      BTFSC INTCON, INTF    ; check for D3MX interrupt
      CALL  D3MX_INT
      BTFSC INTCON, RBIF    ; check for RAM interrupt
      GOTO  CALL_RAM_INT
      BTFSS PORTB, INTB2      ; check for missed RAM interrupt
      GOTO  CALL_RAM_INT
      BTFSC INTCON, T0IF    ; check for Timer 0 interrupt
      CALL  TIMER0_INT
      BTFSC PIR1, TMR2IF      ; check for Timer 2 interrupt
      CALL  TIMER2_INT

D3MX_INT_REPEAT
      BTFSC PORTB, INTB1      ; keep processing D3MX interrupts
      GOTO  CLEAN_UP    ; while they are still pending
      CALL  D3MX_INT
      GOTO  D3MX_INT_REPEAT

CLEAN_UP
      ; restore context
      MOVF  TEMP_PCLATH, W
      MOVWF PCLATH
      SWAPF TEMP_STATUS, W
      MOVWF STATUS
      SWAPF TEMP_W, F
      SWAPF TEMP_W, W
      RETFIE                  ; RETURN FROM INTERRUPT

CALL_RAM_INT
```

```
      CALL  RAM_INT
      MOVF  PORTB, F    ; read PORTB to clear mismatch condition
      BCF   INTCON, RBIF      ; clear RBIF interrupt flag
      GOTO  CLEAN_UP


;*********************************
;* RAM INTERRUPT SERVICE ROUTINE *
;*********************************

; FUNCTION: Handles dual port RAM mailbox interrupts.  According to the
;           command sent by the host the neccesary service routine is
;           branched to from here.
; TAKES:    nothing
; RETURNS:  nothing
; ASSUMES:  Only called from INTHDLR.

RAM_INT
      RD_RAM      MAILIN            ; read IN mailbox
      SNE   RD_REG_CMD  ; check for read command
      GOTO    RD_REG
      MOVF    DATA_REG, W
      SNE     WR_REG_CMD      ; check for write command
      GOTO    WR_REG
      MOVF  DATA_REG, W
      SNE   XFDL_START_CMD    ; check for FDL packet send command
      GOTO  XFDL_START
      MOVF  DATA_REG, W
      SNE   START_BOC_CMD     ; check for BOC transmit command
      GOTO  START_BOC
      MOVF  DATA_REG, W
      SNE   STOP_BOC_CMD      ; check for BOC idle command
      GOTO  STOP_BOC
      MOVF  DATA_REG, W
      SNE   LLA_REQ_TX_CMD    ; check for transmit line loopback activate
                        ; request command
      GOTO  LL_REQ_TX
      MOVF  DATA_REG, W
      SNE   LLD_REQ_TX_CMD    ; check for transmit line loopback deactivate
                        ; request command
      GOTO  LL_REQ_TX
      MOVF  DATA_REG, W
      SNE   LOOPT_HIGH_CMD    ; check for disable loop timing command
      GOTO  LOOPT_HIGH
      MOVF  DATA_REG, W
      SNE   LOOPT_LOW_CMD     ; check for enable loop timing command
      GOTO  LOOPT_LOW

      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

RD_REG
      RD_RAM      RAM_ARG_1         ; transfer LSB of address
      MOVWF ADDR_D3MX_LO
      RD_RAM      RAM_ARG_2         ; transfer MSB of address
      MOVWF ADDR_D3MX_HI
```

```
      CALL  READ_D3MX
      WR_RAM_D    RAM_DATA_RETURN
      WR_RAM      MAILOUT, IO_DONE  ; signal end of I/O access
                              ; by mailing the host

      RETURN                        ; RETURN FROM RAM_INT SUBROUTINE

WR_REG
      RD_RAM  RAM_ARG_1 ; transfer LSB of address
      MOVWF   ADDR_D3MX_LO
      RD_RAM  RAM_ARG_2 ; transfer MSB of address
      MOVWF   ADDR_D3MX_HI
      RD_RAM  RAM_ARG_3 ; transfer data byte
      MOVWF   DATA_REG
      CALL    WRITE_D3MX
      WR_RAM  MAILOUT, IO_DONE      ; signal end of I/O access

      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

XFDL_START
      RD_RAM      RAM_XFDL_PKT_LEN  ; read in packet length
      MOVLW MAX_XFDL_PKT_LEN  ; abort if length>max length
      SUBWF DATA_REG, W
      BTFSC STATUS, C
      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

      CLRF  XFDL_DATA_PTR           ; clear XFDL buffer data pointer
      MOVF  DATA_REG, W
      MOVWF XFDL_PKT_LEN            ; make local copy of packet length

      ; enable XFDL interrupts
      RD_D3MX     XFDL_CONFIG
      BSF   DATA_REG, XFDL_INTE
      WR_D3MX_D   XFDL_CONFIG

      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

START_BOC
      RD_RAM      RAM_ARG_1         ; read BOC to transmit
      WR_D3MX_D   XBOC_CODE   ; copy to XBOC code register,
                              ; initiating BOC transmission

      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

STOP_BOC
      WR_D3MX     XBOC_CODE, BOC_IDLE     ; send idle code

      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

LL_REQ_TX
      MOVF  DATA_REG, W
      MOVWF WORK_I            ; store mailbox command
      RD_RAM      RAM_ARG_1   ; read line to request a loopback on
      BTFSC STATUS, Z
      GOTO  LL_REQ_DS3
```

```
      SNE   0x1D
      GOTO  LL_REQ_DS1ALL
      MOVLW 0x1D        ; check that line is valid (1-28)
      SUBWF DATA_REG, W ; return if not
      BTFSC STATUS, C
      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE
      BSF   DATA_REG, 5 ; compose line ID BOC
      MOVF  DATA_REG, W ; and save it for later
      MOVWF LL_REQ_LINE_BOC
      GOTO  LL_REQ_TX_CONT
LL_REQ_DS3
      MOVLW BOC_LL_DS3
      MOVWF LL_REQ_LINE_BOC
      GOTO  LL_REQ_TX_CONT
LL_REQ_DS1ALL
      MOVLW BOC_LL_DS1ALL
      MOVWF LL_REQ_LINE_BOC
LL_REQ_TX_CONT
      MOVF  WORK_I, W
      SNE   LLA_REQ_TX_CMD
      GOTO  LLA_REQ_TX
      MOVLW BOC_LL_DEACTIVATE
LL_REQ_TX_CONT2
      MOVWF DATA_REG
      BCF   TIMER_FLAGS, LL_REQ_TX_TMR1
      BCF   TIMER_FLAGS, LL_REQ_TX_TMR2
      MOVLW T2_PERIOD         ; load t2 period
      BANK1
      MOVWF PR2
      BANK0
      CLRF  TMR2              ; clear timer 2
      WR_D3MX_D   XBOC_CODE
      BSF   T2CON, TMR2ON           ; start timer 2
      BANK1
      BSF   PIE1, TMR2IE            ; enable timer 2 ints
      BANK0

      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE
LLA_REQ_TX
      MOVLW BOC_LL_ACTIVATE
      GOTO  LL_REQ_TX_CONT2

LOOPT_HIGH
      BSF   PORTE, LOOP_T
      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE

LOOPT_LOW
      BCF   PORTE, LOOP_T
      RETURN                  ; RETURN FROM RAM_INT SUBROUTINE


;*************************************
;* TIMER 0 INTERRUPT SERVICE ROUTINE *
;*************************************
```

```
; FUNCTION: Handles Timer 0 interrupts.  Decrements 1s counter and updates
;           RLOL and LOCK reflection in RAM.  Sets 1s flag, toggles LED 4,
;           and reloads counter when counter reaches zero.
; TAKES:    nothing
; RETURNS:  nothing
; ASSUMES:  Only called from INTHDLR.

TIMER0_INT
      BCF   INTCON, T0IF      ; clear timer 0 interrupt flag
      DECFSZ      TIME_COUNT, F     ; decrement 1s timer counter
                        ; and update RLOL and LOCK status
                        ; and return if not 0
      GOTO  COPY_PINS

      ; reset 1s timer counter
      MOVLW ONE_SECOND
      MOVWF TIME_COUNT
      BSF   TIMER_FLAGS, ONE_SEC    ; set 1 second flag
      MOVLW 0x80              ; toggle LED4
      XORWF PORTB, F

      RETURN                  ; RETURN FROM TIMER_INT SUBROUTINE

COPY_PINS
      CLRW                    ; reflect RLOL pin status in RAM
      BTFSC PORTB, RLOL
      MOVLW 0x01
      MOVWF DATA_REG
      WR_RAM_D    RAM_RLOL
      CLRW                    ; reflect LOCK pin status in RAM
      BTFSC PORTA, LOCK
      MOVLW 0x01
      MOVWF DATA_REG
      WR_RAM_D    RAM_LOCK

      RETURN                  ; RETURN FROM TIMER_INT SUBROUTINE


;*************************************
;* TIMER 2 INTERRUPT SERVICE ROUTINE *
;*************************************

; FUNCTION: Handles Timer 2 interrupts.
;           Operates in the following sequence:
;           1st time - sets timer flag, returns
;           2nd time - changes BOC to line codeword BOC, returns
;           3rd time - sets timer flag, returns
;           4th time - idles FEAC channel and returns
; TAKES:    nothing
; RETURNS:  nothing
; ASSUMES:  Only called from INTHDLR.

TIMER2_INT
      BCF   PIR1, TMR2IF      ; clear timer 2 interrupt flag
      BTFSC TIMER_FLAGS, LL_REQ_TX_TMR1
```

```
      GOTO  T2_CONT
      BSF   TIMER_FLAGS, LL_REQ_TX_TMR1
      RETURN                  ; RETURN FROM TIMER2_INT SUBROUTINE
T2_CONT
      BTFSC TIMER_FLAGS, LL_REQ_TX_TMR2
      GOTO  LL_REQ_TX_COMPLETE
      MOVF  LL_REQ_LINE_BOC, W      ; transmit line to loopback BOC
      MOVWF DATA_REG
      WR_D3MX_D   XBOC_CODE
      BCF   TIMER_FLAGS, LL_REQ_TX_TMR1
      BSF   TIMER_FLAGS, LL_REQ_TX_TMR2
      RETURN                  ; RETURN FROM TIMER2_INT SUBROUTINE

LL_REQ_TX_COMPLETE
      BCF   T2CON, TMR2ON     ; turn timer 2 off
      BANK1
      BCF   PIE1, TMR2IE      ; disable timer 2 interrupts
      BANK0
      MOVLW BOC_IDLE    ; idle the FEAC channel
      MOVWF DATA_REG
      WR_D3MX_D   XBOC_CODE
      RETURN                  ; RETURN FROM TIMER2_INT SUBROUTINE


;**********************************
;* D3MX INTERRUPT SERVICE ROUTINE *
;**********************************

; FUNCTION: Handles D3MX interrupts.  Reads Master Interrupt Source
;           registers to determine interrupt source and branches to
;           appropriate service routine.
; TAKES:    nothing
; RETURNS:  nothing
; ASSUMES:  Only called from INTHDLR.

D3MX_INT
      BCF   INTCON, INTF      ; clear INTF interrupt flag

      ; determine interrupt source
      RD_D3MX     MSTR_INT_SOURCE_1
      BTFSC DATA_REG, MIS1_REG2
      GOTO  CHECK_REG2
      BTFSC DATA_REG, MIS1_XFDLINT
      GOTO  XFDL
      BTFSC DATA_REG, MIS1_RFDLINT
      GOTO  RFDL
      BTFSC DATA_REG, MIS1_DS3FRMR
      GOTO  DS3_FRMR
      BTFSC DATA_REG, MIS1_RBOC
      GOTO  RBOC

      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

CHECK_REG2
      RD_D3MX     MSTR_INT_SOURCE_2
```

```
      ANDLW 0x7F        ; mask out XFDLUDR, and then verify interrupt
      BTFSC STATUS, Z   ; return if no interrupt
      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

      ; determine which DS2 interrupted (first one if multiple)
      MOVWF WORK_I
      CLRF  DATA_REG
      INCF  DATA_REG, F
      CLRF  WORK2_I
      MOVLW 0x10
      BCF   STATUS, C
SHIFT_LOOP
      BTFSC WORK_I, 0
      GOTO  END_SHIFT_LOOP
      RRF   WORK_I, F
      INCF  DATA_REG, F
      ADDWF WORK2_I, F
      GOTO  SHIFT_LOOP
END_SHIFT_LOOP
      WR_RAM_D    RAM_DS2_ID  ; write DS2 framer # to RAM

DS2_FRMR
      MOVLW DS2_FRMR_STATUS
      ADDWF WORK2_I, W
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI
      CALL  READ_D3MX
      MOVWF WORK_I                  ; save status in WORK
      MOVLW DS2_FRMR_IS
      ADDWF WORK2_I, W
      MOVWF ADDR_D3MX_LO
      CALL  READ_D3MX
      BTFSC DATA_REG, DS2FIS_AISI
      CALL  DS2_AIS
      BTFSC DATA_REG, DS2FIS_REDI
      CALL  DS2_RED_ALARM

      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

DS2_AIS
      MOVLW DS2_AIS_A
      BTFSS WORK_I, DS2FS_AISV
      MOVLW DS2_AIS_C
      GOTO  DS2_FRMR_FINISH
DS2_RED_ALARM
      MOVLW DS2_RED_A
      BTFSS WORK_I, DS2FS_REDV
      MOVLW DS2_RED_C

DS2_FRMR_FINISH
      MOVWF DATA_REG
      WR_RAM_D    MAILOUT           ; signal host

      RETURN
```

```
DS3_FRMR
      RD_D3MX     DS3_FRMR_STATUS
      MOVWF WORK_I                  ; save framer status in WORK
      RD_D3MX     DS3_FRMR_IS
      BTFSC DATA_REG, DS3FIS_LOSI
      CALL  DS3_LOS
      BTFSC DATA_REG, DS3FIS_IDLI
      CALL  DS3_IDLE
      BTFSC DATA_REG, DS3FIS_AISI
      CALL  DS3_AIS
      BTFSC DATA_REG, DS3FIS_REDI
      CALL  DS3_RED_ALARM

      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

DS3_LOS
      BTFSC WORK_I, DS3FS_LOSV
      BSF   PORTE, LOOP_T
      RETURN
DS3_IDLE
      MOVLW DS3_IDL_A
      BTFSS WORK_I, DS3FS_IDLV
      MOVLW DS3_IDL_C
      GOTO  DS3_FRMR_FINISH
DS3_AIS
      MOVLW DS3_AIS_A
      BTFSS WORK_I, DS3FS_AISV
      MOVLW DS3_AIS_C
      GOTO  DS3_FRMR_FINISH
DS3_RED_ALARM
      MOVLW DS3_RED_A
      BTFSS WORK_I, DS3FS_REDV
      MOVLW DS3_RED_C

DS3_FRMR_FINISH
      MOVWF DATA_REG
      WR_RAM_D    MAILOUT           ; signal host

      RETURN

XFDL
      ; check for underrun and verify interrupt
      RD_D3MX     XFDL_IS
      BTFSC DATA_REG, XFDL_UDR
      GOTO  XFDL_UNDERRUN           ; deal with underrun
      BTFSS DATA_REG, XFDL_INT      ; return if no interrupt
      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

      ; check for end of message
      MOVF  XFDL_DATA_PTR, W
      XORWF XFDL_PKT_LEN, W
      BTFSC STATUS, Z
      GOTO  XFDL_END_MSG            ; finish message

      ; copy next data byte to XFDL transmit data reg
```

```
      MOVLW LOW RAM_XFDL_BUFFER     ; calculate RAM address of next byte
      ADDWF XFDL_DATA_PTR, W  ; NOTE that it is assumed that buffer
      MOVWF ADDR_RAM_LO       ; will not span 2 values of ADDR_RAM_HI
      MOVLW HIGH RAM_XFDL_BUFFER    ; ie. no carry implemented
      MOVWF ADDR_RAM_HI
      CALL  READ_RAM
      WR_D3MX_D   XFDL_DATA
      INCF  XFDL_DATA_PTR, F

      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

XFDL_UNDERRUN
      ; clear underrun flag
      BCF   DATA_REG, XFDL_UDR
      WR_D3MX_D   XFDL_IS

      RETURN                  ; RETURN FROM D3MX_INT SUBROUTINE

XFDL_END_MSG
      RD_D3MX     XFDL_CONFIG
      BSF   DATA_REG, XFDL_EOM      ; set EOM bit
      BCF   DATA_REG, XFDL_INTE     ; disable XFDL interrupts
      WR_D3MX_D   XFDL_CONFIG

      WR_RAM      MAILOUT, XFDL_DONE      ; signal XFDL done to host

      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE


RFDL
      RD_D3MX     RFDL_DATA         ; get data byte
      MOVWF WORK_I
      RD_D3MX     RFDL_STATUS
      MOVWF WORK2_I
      BTFSC WORK2_I, RFDL_OVR ; check for overrun
      GOTO  RFDL_OVERRUN
      BTFSS WORK2_I, RFDL_FLG ; have we been receiving flags?
      GOTO  FLG_0

FLG_1
      BTFSC FLAGS, RFDL_ACTIVE      ; was link active?
      GOTO  NEW_BYTE          ; if yes then get new byte

      BSF   FLAGS, RFDL_ACTIVE      ; if no then mark active
      CLRF  RFDL_DATA_PTR           ; zero data pointer
      CLRF  RFDL_LCL_STATUS         ; clear local link status flags

      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

FLG_0
      BTFSS FLAGS, RFDL_ACTIVE      ; was link active?
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE
      BCF   FLAGS, RFDL_ACTIVE      ; this must be an abort, so clear
                              ; link active flag
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE
```

```
NEW_BYTE
      MOVF  WORK_I, W          ; setup to copy new data byte
      MOVWF DATA_REG           ; to RFDL receive data buffer
      MOVLW LOW RAM_RFDL_BUFFER     ; calculate RAM address of next byte
      ADDWF RFDL_DATA_PTR, W  ; NOTE that it is assumed that buffer
      MOVWF ADDR_RAM_LO        ; will not span 2 values of ADDR_RAM_HI
      MOVLW HIGH RAM_RFDL_BUFFER    ; ie. no carry implemented
      MOVWF ADDR_RAM_HI
      CALL  WRITE_RAM

      INCF  RFDL_DATA_PTR, F  ; increment data pointer

      BTFSC WORK2_I, RFDL_EOM ; check if this is the last byte
      GOTO  RFDL_END_MSG            ; if it is then wrap up

      BTFSS WORK2_I, RFDL_FE  ; is there another byte waiting?
      GOTO  RFDL               ; if yes then fetch it

      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

RFDL_END_MSG
      ; copy CRC and NVB bits to local link status
      BTFSC WORK2_I, RFDL_CRC
      BSF   RFDL_LCL_STATUS, RFDL_CRC
      BTFSC WORK2_I, RFDL_NVB2
      BSF   RFDL_LCL_STATUS, RFDL_NVB2
      BTFSC WORK2_I, RFDL_NVB1
      BSF   RFDL_LCL_STATUS, RFDL_NVB1
      BTFSC WORK2_I, RFDL_NVB0
      BSF   RFDL_LCL_STATUS, RFDL_NVB0

      ; copy local link status to RAM receive buffer
      MOVF  RFDL_LCL_STATUS, W
      MOVWF DATA_REG
      WR_RAM_D    RAM_RFDL_STATUS

      ; copy packet length to RAM receive buffer
      MOVF  RFDL_DATA_PTR, W
      MOVWF DATA_REG
      WR_RAM_D    RAM_RFDL_PKT_LEN

      CLRF  RFDL_DATA_PTR           ; zero data pointer
      CLRF  RFDL_LCL_STATUS         ; clear local link status flags

      MOVF  DATA_REG, W        ; check if packet length >= 3
      SUBLW 0x02               ; return if not
      BTFSC STATUS, C
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

      WR_RAM      MAILOUT, RFDL_NEW ; signal new packet reception to host

      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

RFDL_OVERRUN
```

```
      BSF   RFDL_LCL_STATUS, RFDL_OVR     ; set overrun flag in
                                    ; local link status byte
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

RBOC
      RD_D3MX     RBOC_IS
      BTFSC DATA_REG, RBOC_IDLEI    ; test for idle BOC interrupt
      GOTO  IDLE_BOC
      BTFSS DATA_REG, RBOC_BOCI     ; test for new boc interrupt
                              ; return if neither
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

NEW_BOC
      ANDLW 0x3F              ; mask out 6 bit BOC
      MOVWF DATA_REG
      MOVWF WORK_I                  ; and save it to WORK_I
      BTFSC FLAGS, LL_ACTIVATE      ; check for previous ll activate code
      GOTO  LL_LINE_CODE
      BTFSC FLAGS, LL_DEACTIVATE    ; check for previous ll deactivate code
      GOTO  LL_LINE_CODE
      MOVF  WORK_I, W
      SNE   BOC_LL_ACTIVATE
      GOTO  SET_LLA
      MOVF  WORK_I, W
      SNE   BOC_LL_DEACTIVATE
      GOTO  SET_LLD
      WR_RAM_D    RAM_RBOC    ; copy new BOC to RAM
      WR_RAM      MAILOUT, RBOC_NEW ; indicate new BOC to host
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

SET_LLA
      BSF   FLAGS, LL_ACTIVATE      ; set ll activate flag
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

SET_LLD
      BSF   FLAGS, LL_DEACTIVATE    ; set ll deactivate flag
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

LL_LINE_CODE
      BTFSS WORK_I, 5
      GOTO  DS3_OR_DS1ALL
      MOVLW 0x1F              ; check to see bits 0-4 of BOC are
      ANDWF WORK_I, F         ; in the range 1-28 otherwise abort
      MOVF  WORK_I, W         ; this line loopback request
      BTFSC STATUS, Z
      GOTO  LL_ABORT
      SUBLW 0x1D
      BTFSC STATUS, Z
      GOTO  LL_ABORT
      ; calculate MX12 register to access
      MOVF  WORK_I, W
      MOVWF DATA_REG          ; indicate which DS1 to host
      WR_RAM_D    RAM_LL_ID
      DECF  WORK_I, F         ; subtract 1
      MOVLW 0x03
```

```
      ANDWF WORK_I, W
      MOVWF WORK2_I
      MOVLW 0x1C              ; (x mod 4) * 4
      ANDWF WORK_I, F
      BCF   STATUS, C
      RLF   WORK_I, F         ; multiply by 4
      RLF   WORK_I, F
      MOVLW MX12_LB_ACT
      ADDWF WORK_I, W
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI
      CALL  READ_D3MX
      BTFSC FLAGS, LL_ACTIVATE
      GOTO  DS1_LLA
      MOVF  WORK2_I, W
      BTFSC STATUS, Z
      BCF   DATA_REG, 0
      MOVF  WORK2_I, W
      SNE   0x01
      BCF   DATA_REG, 1
      MOVF  WORK2_I, W
      SNE   0x02
      BCF   DATA_REG, 2
      MOVF  WORK2_I, W
      SNE   0x03
      BCF   DATA_REG, 3
      CALL  WRITE_D3MX
      WR_RAM      MAILOUT, LL_DEACTIVATED ; signal host
      BCF   FLAGS, LL_DEACTIVATE
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE
DS1_LLA
      MOVF  WORK2_I, W
      BTFSC STATUS, Z
      BSF   DATA_REG, 0
      MOVF  WORK2_I, W
      SNE   0x01
      BSF   DATA_REG, 1
      MOVF  WORK2_I, W
      SNE   0x02
      BSF   DATA_REG, 2
      MOVF  WORK2_I, W
      SNE   0x03
      BSF   DATA_REG, 3
      CALL  WRITE_D3MX
      WR_RAM      MAILOUT, LL_ACTIVATED   ; signal host
      BCF   FLAGS, LL_ACTIVATE
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

DS3_OR_DS1ALL
      MOVF  WORK_I, W
      SNE   BOC_LL_DS3
      GOTO  DS3_LINE
      MOVF  WORK_I, W
      SNE   BOC_LL_DS1ALL
      GOTO  DS1ALL
```

```
LL_ABORT
      BCF   FLAGS, LL_ACTIVATE      ; BOC is not one of the ones expected
      BCF   FLAGS, LL_DEACTIVATE    ; abort loopback request
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

DS3_LINE
      WR_RAM      RAM_LL_ID, 0x00         ; indicate DS3 loopback
                              ; activate/deactivate to host
      RD_D3MX     MSTR_LB_CONFIG
      BTFSC FLAGS, LL_ACTIVATE      ; ll activate ?
      GOTO  DS3_LLA
      BCF   DATA_REG, MLC_LLBE      ; no then ll deactivate
      BCF   FLAGS, LL_DEACTIVATE
      WR_D3MX_D   MSTR_LB_CONFIG
      WR_RAM      MAILOUT, LL_DEACTIVATED ; signal host
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE
DS3_LLA
      BSF   DATA_REG, MLC_LLBE      ; set LLBE in master LB config
      BCF   FLAGS, LL_ACTIVATE      ; clear ll activate flag
      WR_D3MX_D   MSTR_LB_CONFIG
      WR_RAM      MAILOUT, LL_ACTIVATED   ; signal host
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE
DS1ALL
      WR_RAM      RAM_LL_ID, 0x1D         ; indicate that all DS1 loopbacks
                              ; are being activated/deactivated
      MOVLW 0x07
      MOVWF WORK_I
      MOVLW MX12_LB_ACT
      MOVWF ADDR_D3MX_LO
      CLRF  ADDR_D3MX_HI
      BTFSC FLAGS, LL_ACTIVATE      ; ll activate ?
      GOTO  DS1ALL_LLA_LOOP

DS1ALL_LLD_LOOP
      CALL  READ_D3MX
      MOVLW 0xF0
      ANDWF DATA_REG, F
      CALL  WRITE_D3MX
      MOVLW 0x10
      ADDWF ADDR_D3MX_LO, F
      DECFSZ      WORK_I, F
      GOTO  DS1ALL_LLD_LOOP

      BCF   FLAGS, LL_DEACTIVATE    ; clear ll deactivate bit

      WR_RAM      MAILOUT, LL_DEACTIVATED ; signal host

      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

DS1ALL_LLA_LOOP
      CALL  READ_D3MX
      MOVLW 0x0F
      IORWF DATA_REG, F
      CALL  WRITE_D3MX
      MOVLW 0x10
```

```
      ADDWF ADDR_D3MX_LO, F
      DECFSZ      WORK_I, F
      GOTO  DS1ALL_LLA_LOOP

      BCF   FLAGS, LL_ACTIVATE      ; clear ll activate bit

      WR_RAM      MAILOUT, LL_ACTIVATED   ; signal host

      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE


IDLE_BOC
      WR_RAM      MAILOUT, RBOC_IDLE      ; indicate idle BOC to host
      RETURN                        ; RETURN FROM D3MX_INT SUBROUTINE

      END               ; *** END OF FILE ***
```

CONTACTING PMC-SIERRA

PMC-Sierra, Inc.
105 - 8555 Baxter Place
Burnaby, B.C.
Canada V5A 4V7

Telephone: 604-415-6000
Facsimile: 604-415-6200

Product Information: info@pmc-sierra.bc.ca
Applications information: apps@pmc-sierra.bc.ca

World Wide Web Site: http://www.pmc-sierra.com

NOTES

PMC-951045 Issue date: October, 1996.

Printed in Canada