PCD-E12

REFERENCE MANUAL

for Revision 0 boards

MANUAL Revision 1.0

EMAC, inc.
EQUIPMENT MONITOR AND CONTROL
2390 EMAC Way, Carbondale, Illinois 62901
World Wide Web: http://www.emacinc.com
Phone: (618) 529-4525 Fax: (618) 457-0110

DISCLAIMER

EMAC Inc. does not assume any liability arising out of the application or use of any of its products or designs. Products designed or distributed by EMAC Inc. are not intended for, or authorized to be used in, applications such as life support systems or for any other use in which the failure of the product could potentially result in personal injury, death or property damage.

If EMAC Inc. products are used in any of the aforementioned unintended or unauthorized applications, Purchaser shall indemnify and hold EMAC Inc. and its employees and officers harmless against all claims, costs, damages, expenses, and attorney fees that may directly or indirectly arise out of any claim of personal injury, death or property damage associated with such unintended or unauthorized use, even if it is alleged that EMAC Inc. was negligent in the design, documentation or manufacture of the product.

EMAC Inc. reserves the right to make changes to any products with the intent to improve overall quality, without further notification.

FCC COMPLIANCE

EMAC's PCD series of PC daughtercards are classified as sub-assemblies per FCC CST bulletin, No. 62, May 1984. The bulletin states that sub-assemblies are not themselves subject to the FCC rules. Only the end product is directly subject to the rules.

TECHNICAL SUPPORT

Please refer technical support issues to http://www.emacinc.com/support/

REVISION NOTES

HARDWARE REVISIONS

REV. 0

The CAN interface header in this revision is not designed for standard 10pin to DB9 female connectors. Only the EMAC supplied CAN cable will provide a standard DB9 connection. This will be corrected in a later revision, so to maintain wiring compatibility with future revisions you should not make custom cable connections directly to HDR8, but only through the supplied DB9 female connector.

INTRODUCTION

The PCD-E12 Analog and Digital I/O Daughter Board adds expanded I/O capability to the PCM-53E52 SBC. It adds the following features:

- Motorola HS12 processor.
- An additional RS-232 port.
- CAN 2.0B port
- 24 Digital I/O lines in a standard 50 pin I/O Rack compatible header.
  - 8 are high current sink outputs which also have 8 bit resolution PWM capability.
  - 8 have 16 bit resolution PWM capability.
- 16 channels of A/D with 10-bit resolution (0 to 5V).
- Keypad interface for decoding 3x4, 4x4, 5x4 or 6x4 matrix keypads.

Optional features
- Additional 8 channels of 12-bit resolution A/D (0 to 5V)
- 4 channels of D/A with 12-bit resolution (0 to 5V)
- RS-422/485 serial port

The PCD can also be custom programmed by EMAC (contact a salesperson for details). Some features that are available in custom programmed applications:
- SPI port that can be used in master or slave mode
- I^2C port
- LCD interface which can drive character or graphic type displays
- RS-232 and RS-485 ports with configurable protocol and standard baud rates up to 230kbaud and higher non-standard baud rates.
- Additional PWM channels.
- 21 additional I/O lines available .
  - 11 available if LCD interface is not needed.
  - 10 available if keypad interface is not needed.
- 16 additional input lines available if the 10 bit A/Ds are not needed.
- DAC waveform generator/sequencer.
- Up to 14 edge triggered interrupt inputs.
  - 6 available if keypad interface is not needed.
  - 8 available if the 8 channels of 16 bit resolution PWM capability is not needed.
- Larger matrix keypad decoder.
- Stepper motor driver.
- Quadrature decoder(s).
- Mouse/trackball/touchpad driver.
- Touchscreen controller.
- Stand-alone operation (operation without PCM-53E52).
- Direct desktop PC connection via RS-232 port
- Stand-alone multi-node network.
  - RS-458 network.
  - CAN 2.0B network.

DIGITAL I/O

PX0-PX7 (Block 0, Port 0)

These are abbreviated B0P0 in the diagram. Each line of PX0-PX7 is individually capable of sinking 500mA, and the driver package power dissipation limit is typically 0.93W. Each line is capable of withstanding an open circuit voltage of up to 26V. B0P0 is not configurable and is set up as output only. These port lines can only be controlled using the PWM control functions. To activate or deactivate a line, set the PWM percentage to 100 or 0, respectively.

PX8-PX15 (Block 0, Port 1)

These pins are referenced in the C drivers as block 0, port 1 (abbreviated B0P1 in the diagram). Each port line can be individually configured as an output, input, or 16 bit counter input. When a particular port is configured as a counter, in the C drivers the port counter number corresponds to the port bit number (i.e. counter numbers 0 through 7 correspond to pins B0P1.0 through B0P1.7, respectively).

PX16-PX23 (Block 0, Port 2)

These pins are referenced in the C drivers as block 0, port 1 (abbreviated B0P1 in the diagram). Each port line can be individually configured as an input or output.

PWM

Each line of B0P0 is an 8 bit PWM output and each line of B0P1 can be individually configured as a 16 bit PWM output. The 16 PWM channel numbers referred to in the C drivers correspond to the respective PXx number shown in the diagram.

HDR3

	50	49	
GND	o	o	VCC
GND	o	o	PX0 B0P0.0
GND	o	o	PX1 B0P0.1
GND	o	o	PX2 B0P0.2
GND	o	o	PX3 B0P0.3
GND	o	o	PX4 B0P0.4
GND	o	o	PX5 B0P0.5
GND	o	o	PX6 B0P0.6
GND	o	o	PX7 B0P0.7
GND	o	o	PX8 **B0P1.0**
GND	o	o	PX9 **B0P1.1**
GND	o	o	PX10 **B0P1.2**
GND	o	o	PX11 **B0P1.3**
GND	o	o	PX12 **B0P1.4**
GND	o	o	PX13 **B0P1.5**
GND	o	o	PX14 **B0P1.6**
GND	o	o	PX15 **B0P1.7**
GND	o	o	PX16 B0P2.0
GND	o	o	PX17 B0P2.1
GND	o	o	PX18 B0P2.2
GND	o	o	PX19 B0P2.3
GND	o	o	PX20 B0P2.4
GND	o	o	PX21 B0P2.5
GND	o	o	PX22 B0P2.6
GND	o	o	PX23 B0P2.7
	2	1	

ANALOG I/O

DAC0-DAC3 *(optional)*
These correspond to the 4 channels of the optional 12 bit DAC (digital to analog converter). A value of 0x000 to 0xFFF output to a DAC will produce a 0 to 5V output, respectively.

B1A0-B1A7 *(optional)*
These correspond to the 8 channels of the optional 12 bit ADC (referred to as block 1 in the C drivers). A 0V to 5V signal applied to any of these channels will result in a digital conversion of 0x000 to 0xFFF, respectively.

B0A0-B0A15
These correspond to the 16 channels of the 10 bit ADC (referred to as block 0 in the C drivers). A 0V to 5V signal applied to any of these channels will result in a digital conversion of 0x000 to 0x3FF, respectively.

HDR5

	40	39	
DAC3	o	o	DAC2
DAC1	o	o	DAC0
GND	o	o	GND
B1A7	o	o	B1A6
B1A5	o	o	B1A4
GND	o	o	GND
B1A3	o	o	B1A2
B1A1	o	o	B1A0
GND	o	o	GND
B0A15	o	o	B0A14
B0A13	o	o	B0A12
GND	o	o	GND
B0A11	o	o	B0A10
B0A9	o	o	B0A8
GND	o	o	GND
B0A7	o	o	B0A6
B0A5	o	o	B0A4
GND	o	o	GND
B0A3	o	o	B0A2
B0A1	o	o	B0A0
	2	1	

LCD INTERFACE

The LCD interface allows the PCD to control character displays and graphic displays depending on the JP1 jumper settings shown below:

Character mode with backlight always on	Character mode with backlight controlled	Graphic mode
1 2 o o A | | B | | C	1 2 | o A | | B o | C	1 2 | | A | | B o o C

The interface can support character LCD's that have HD44780 or compatible controllers. It can also support graphic LCDs which have HD61830 or compatible controllers. Currently, however, the PCD firmware only supports "Character mode with backlight always on".

```
        HDR2
        1   2
    VCC  o o  GND
     RS  o o  CNTR
      E  o o  R/W*
     D1  o o  DO
     D3  o o  D2
     D5  o o  D4
     D7  o o  D6
      K  o o  A
       15    16

Character Mode
(backlight voltage)
K = cathode
A = anode

Graphic Mode.
K = RES*
A = CS*
```

KEYPAD INTERFACE

The PCD decodes a keypad according to the "RETURN VALUES" table. A connection between an X and Y pin will produce the ASCII value as shown in the table. For example a connection between X2 and Y3 will product the ASCII character 'N'. The "4x4" and "3x4" columns show the proper method of connecting EMAC's 4x4 and 3x4 keypads.

HDR 1	4x4	3x4
1		
o X6		
o X5		
o X4	1	
o X3	2	1
o X2	3	2
o X1	4	3
o Y4	5	4
o Y3	6	5
o Y2	7	6
o Y1	8	7
o GND	9	8
11		

RETURN VALUES TABLE

	X1	X2	X3	X4	X5	X6
Y1-	A	B	C	D	E	F
Y2-	G	H	I	J	K	L
Y3-	M	N	O	P	Q	R
Y4-	S	T	U	V	W	X

CAN INTERFACE

The current version of the firmware has a fixed rate of 125kbaud, with no masking.

Inserting a shunt in JP2 will put a 120 ohm terminating resistor across the CANH and CANL lines.

Only connect the DB9 female cable adapter provided by EMAC to HDR8. Standard DB9 female cables will not work on HDR8, and likewise this cable cannot be used in place of other standard DB9 female cables.

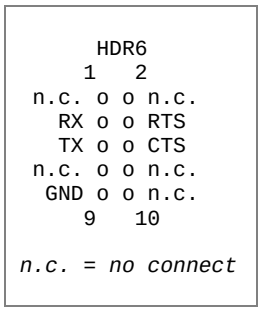
```
   DB9 FEMALE
 CAN CONNECTOR

n.c. 1-o
         o-6 n.c.
CANL 2-o
         o-7 CANH
GND  3-o
         o-8 n.c.
n.c. 4-o
         o-9 n.c.
n.c. 5-o


n.c. = no connect
```

RS-232 PORT

The current version of the firmware has a fixed baud rate of 9600, with 8 data bits, no parity, 1 stop bit and no hardware flow control.



USING THE PCD-E12 WITH THE PCM-53E52 IN LINUX

The standard EMAC Linux distribution for the PCM-53E52 contains an application program called PCDConsol which demonstrates the various functions of the PCD-E12. This program has 2 layers, The application layer and the wrapper functions.

The application layer contains all of the menus and the user interface. It is meant to be an example of a high level interface to the PCD-E12. The next level down, the wrapper layer, contains the functions called by the application layer. These functions make calls to the Linux PCD-E12 driver by opening certain character devices that are created at boot time. The wrapper functions have been designed to provide a portable interface between drivers in different operating systems, and are meant to provide easy way to develop OEM code for the PCD-E12.

Running PCDConsol from the SIB Configuration Menu

The SIB configuration menu starts on any successful root login, and contains an option to run the PCDConsol program. Typing "d" or "D" will start the application and exiting the application will return the user to the configuration menu.

Running PCDConsol from the Bash Prompt

For those not familiar with Bash, programs within the path can be executed simply by typing their name into the prompt. Since the application is located in the /usr/bin, which is within the path, the root user can run it at any time by typing "PCDConsol" and hitting enter.

By default the application is only executable by the root user, but this could be changed with chmod (ie. chmod 777 /usr/bin/PCDConsol).

C FUNCTION DEFINITIONS

```
BOOL PCDInitialize(void);                       //passed parameters (0)
/*********************************************************************



Pre: The PCD Driver Driver must be Closed

Post:
        The PCD Driver will be initialized upon Returning (TRUE)(1)
        (SUCCESS)

        The PCD Driver will not be initialized upon Returning (FALSE)(0)
        (FAILURE)

Description:
        A Call to this function will initialize the PCD Device Driver


*********************************************************************/


BOOL PCDOpen(void);  //passed parameters (0)
/*********************************************************************



Pre:    The Device must be Closed.
        The Device must be Initialized

Post:
        The PCD Driver will be open upon Returning (TRUE)(1)
        (SUCCESS)

        The PCD Driver will not be open upon Returning (FALSE)(0)
        (FAILURE)


Description:

        A successful return from this function will result in the
        open state of the PCD and make it available for Read/Write
        operations.
*********************************************************************/


BOOL PCDClose(void);                            //passed parameters (0)
/*********************************************************************




Pre:    PCD must be Open.
        PCD must be Initialized

Post:
        The PCD Driver will be closed upon Returning (TRUE)(1)
        (SUCCESS)

        The PCD Driver will not be closed upon Returning (FALSE)(0)
        (FAILURE)

Description:
                A successful return from this function will result in the
                closed state of the PCD and free driver allocated memory.


*********************************************************************/
```

```
BOOL PCDGetKeypad(PUCHAR pTargetBuffer,ULONG BufferLength);
/*********************************************************************


Pre:    PCD must be Open.
        PCD must be Initialized

Post:
        Returns a unsigned long number of bytes that was put into
        pTargetBuffer

Description
        This function gets a user-defined number of data bytes from
        the keypad and puts the data into the pTargetBuffer.
*********************************************************************/



WORD PCDGetA2DPort(int Block,               //Block 0 selects 10 bit A/D.
                                            //Block 1 selects optional 12 bit A/D
int Channel);                               //Block 0 channel range is 0 to 15
                                            //Block 1 channel range is 0 to 7

/*********************************************************************
Pre:    PCD must be Open.
        PCD must be Initialized
Post:
        PCD Returns a 16 bit unsigned short integer representing
        the user defined Block and Channel.


Description:
                The PCD Driver returns a 16 bit unsigned short integer
                representing the user selected Block and Channel.

*********************************************************************/



BOOL PCDSetD2APort(int Block,                   //Block 0
int Channel,                                    //Channels 0-3
int DigitalValue);                              //0-0x3FF
/*********************************************************************


Pre:    PCD must be Open.
        PCD must be Initialized
Post:
        The PCD Driver successfuly set Block, Channel upon Returning
        (TRUE)(1)(SUCCESS)

         The PCD Driver failed to set Block, Channelupon Returning
        (FALSE)(0)(FAILURE)

Description:
        This function sets the analog output of the selected DAC channel.
*********************************************************************/



BOOL PCDSetDigitalConfig(int PortMask);         //PortMask 0-65,535
/*********************************************************************

Pre:    PCD must be Open.
        PCD must be Initialized
Post:
        Port Mask was set upon Returning (TRUE)(1)
        (SUCCESS)
```

```
        Port Mask was not set upon Returning (FALSE)(0)
        (FAILURE)


Description:
Bits 0 to 15 of PortMask correspond directly to digital lines PX8 to PX23,
respectively.A bit set to 1 configures the respective digital line as an
output and a bit set to 0 makes it an input.

**********************************************************************/


BYTE PCDGetDigitalPort(int Block,               //Block 0
int Port);                                      //Ports 1, 2
/**********************************************************************

Pre:    PCD must be Open.
        PCD must be Initialized.
        PCD Digital Port Direction must be Set
Post:
        The PCD Driver Returns Byte for Block and Port

Description:
        This function currently supports one Block and two Ports.
        A modification will be made to output pins only.
        Port 0 Currently Not Supported

**********************************************************************/


BOOL PCDSetDigitalPort(int Block,               //Block 0
int Port,                                       //Ports 1, 2
BYTE Value);                                    //Value 8-bit
/**********************************************************************

Pre:    PCD must be Open.
        PCD must be Initialized.
Post:
        The PCD Driver successfully loaded Block, Port, Value upon
        Returning (TRUE),(1),(SUCCESS).

        The PCD Driver failed to load Block, Port, Value upon
        Returning (FALSE),(0),(FAILURE).

Description:
         This function currently supports one Block and two Ports.
         A modification will be made to output pins only.
         Port 0 Currently Not Supported
**********************************************************************/


BOOL PCDSendSerialPort(PUCHAR pSourceBytes,     //(unsigned char *) to
                                                //array of characters or
                                                //unsigned char string
                                                //name

ULONG NumberOfBytes);                           //unsigned long number of
                                                //bytes to send to serial
                                                //port
/**********************************************************************



Pre:    PCD must be Open.
        PCD must be Initialized

Post:
        The PCD Driver successfully sent user defined number of bytes and
        load them from the pSourceBytes Buffer upon Returning
        (TRUE)(1)(SUCCESS)
        The PCD Driver failed to send user defined number of bytes and
```

```
        load them from the pSourceBytes Buffer upon Returning
        (FALSE)(0)(FAILURE)

Description:
        This function gets a user-defined number of data bytes from
        the PCD serial port and puts the data into the pTargetBuffer.



**********************************************************************/


BOOL PCDGetSerialPort(PUCHAR pTargetBuffer,     //(unsigned char *) to
                                                //array of characters
                                                //or unsigned char
                                                //string name

ULONG BufferLength);                            //unsigned long number of
                                                //bytes to receive from
                                                //serial port

/**********************************************************************

Pre:    PCD must be Open.
        PCD must be Initialized

Post:
        The PCD Driver successfully received user defined number of bytes and
        loaded them into the PTargetBuffer upon Returning
        (TRUE)(1)(SUCCESS)

        The PCD Driver failed to receive user defined number of bytes and
        load them into the PTargetBuffer upon Returning
        (FALSE)(0)(FAILURE)


Description:
        This function gets a user-defined number of data bytes from
        the PCD serial port and puts the data into the
        pTargetBuffer.

**********************************************************************/


BOOL PCDSetPWMPort(int Channel,         //Channels      0-15

int Frequency,                          //Frequency     0-100 (kHz)
int DutyCycle,                          //DutyCycle     0-100 (%)
int Delay);                             //Delay         0-65,535 (Milliseconds)


/**********************************************************************
Pre:    PCD must be Open.
        PCD must be Initialized

Post:
         upon Returning (TRUE)(1)
        (SUCCESS)

         upon Returning (FALSE)(0)
        (FAILURE)

Description:
        This function sets the Frequency, Duty Cycle, Delay for the
        Channel PWM
**********************************************************************/
```

```
char getNB(void);
/*********************************************************************
non blocking getchar function.
*********************************************************************/


BOOL PCDSend2LCD(PUCHAR pSourceBytes, ULONG  NumberOfBytes);
/*********************************************************************
Pre:    PCD must be Open.
        PCD must be Initialized

Post:
         upon Returning (TRUE)(1)
        (SUCCESS)

         upon Returning (FALSE)(0)
        (FAILURE)

Description:
        This function sends a formatting string to the LCD of the
pcd-e12. it supports the following special characters

'\b' backspace 1
'\a' advance 1
'\n' linefeed
'\r' carriage return
'\v' end of line
'\t' tab

*********************************************************************/
```