CMD-2001

Single Board Computer for the Motorola MMC2001 MCORE Microcontroller

USERS MANUAL

717 Lingco Dr., Suite 202
Richardson, TX 75081
972) 994-9676 FAX (972) 994-9170
email: Gary@axman.com
web: http://www.axman.com

CONTENTS

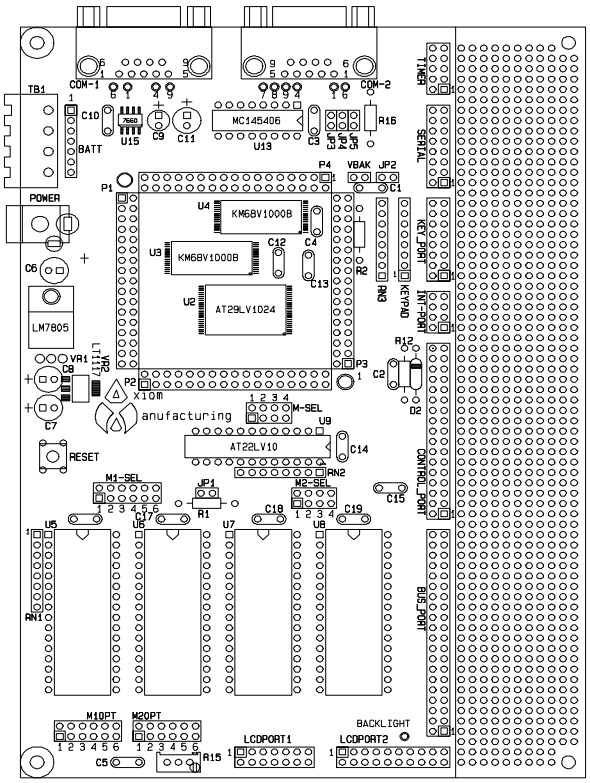
1 CMD-2001 FEATURES

The CMD2001 is a fully configured development system for the Motorola MMC2001 Mcore Microcontroller. The CMD2001 is a Plug and Play system with the supplied PB2001 Controller Module.

The system includes fully assembled development board with Mbug Monitor/Debugger, 64K x 16 Flash EEPROM, 128K x 16 SRAM, DB9 Serial Cable, 9v-300ma Wall Plug, hardware manual and UTL2001 software disk. The UTL2001 software disk includes programming utilities and support software. Free Software is available including the GNU Assembler, C compiler, C++ compiler, Linker and Make utility.

Features include:

- PB2001 MCore Controller Module (installed)
- Standard fixed memory: 64K x 16 Flash EEprom 128K x 16 SRAM
- Two Pairs of Configurable 32pin memory sockets for 32K to 2MByte ROM and 32K to 512KByte SRAM
- COM1 - UART0 w/ RS232 type DB9-S Connection
- COM2 - UART1 w/ RS232 type DB9-P Connection
- 16 Bit Bus support with even/odd 8 bit memory control
- LCD Interface Ports w/ Contrast Adj, Memory Mapped (80 or 160 character)
- Keypad Interface Ports, 16 Key and Full Port
- ISPI Port Connector
- All I/O connectorized to maximize use
- Bus expansion ports with control signals
- Large 1.25 x 7 inch Proto Area
- Easy Power Connection and Tap points
- 8 to 20VDC input to 5 and 3.3V Power Supply
- Operating Power: 100ma @ 3.3V



The Axiom development system provides for low cost software debugging with the use of the MBug Monitor in the MMC2001 Microcontroller. The MBug Monitor allows the user to locate code in the On-Board RAM, set Break Points, Trace, and display or modify registers and memory.

After code is operational the user may relocate the code and reprogram the development board Flash EEPROM for dedicated operation. No additional hardware or software is required. For higher level debugging, the ONCE Debug Port can be used with a background debugger.

2 GETTING STARTED

To get started quickly, perform the following test now to make sure everything is working correctly:

1. Insert the Axiom CMD2001 CD into your CD-ROM drive and open the file called README.TXT.

2. Follow the instructions in this text file to install the utility software and documentation onto your hard drive.

3. Connect one end of the supplied 9-pin serial cable to a free COM port on your PC. Connect the other end of the cable to the COM-1 port on the CMD2001 board.

4. Apply power to the board by plugging in the wall plug power supply that came with the system.

5. Run the terminal program on the CD (or if you prefer you can use your own ASCII terminal program set to 9600,N,8,1).

6. Press then release the RESET button on the CMD2001 board now.

7. After a few seconds delay, if everything is working properly, you should see the MBug prompt in the terminal window. Your board is now ready to use!

```
M     M
MM   MM
M M M M
M  M  M
M     M bug

Version: 1.6
Copyright Motorola Inc., 1993, 1994, 1995, 1997, 1998

Mbug >>
```

If you do not see the monitor/debugger message prompt, or the text is garbage, see the **TROUBLESHOOTING** section of the CMD2001 manual.

2.1 DEVELOPMENT PHILOSOPHY

Software development on the CMD2001 is performed using the Motorola Mbug Debugger in On-Chip ROM to create or assist in creating your program stored in Internal or External RAM (see the Memory Map).

After satisfactory operation running under Mbug your program can be relocated and written to the Internal Flash memory using a utility program running under Mbug. Optionally, you can write your program to External EPROM (U5/6) which frees the Internal Flash for run-time use.

2.2 Tutorial

The following brief tutorial was created to help you become familiar with the software development process quickly.

2.2.1 Using the Mbug Monitor

The Motorola Mbug Debugger is programmed into the MMC2001 On-Chip ROM. It is an interactive Monitor/Debugger that can be accessed via any ASCII terminal program and a serial cable connected to COM-1.

Mbug uses a command line interface where you type commands with parameters to view and modify memory. You can load and execute programs, set breakpoints and examine code, data and registers.

To start Mbug just remove JP1 on the piggyback board and apply power or RESET. Type MENU at the `Mbug >>` prompt to see a list of commands. For complete operating instructions and command descriptions, see the Mbug Users Guide.

When developing software under Mbug, you can locate your code in internal RAM (3000:0000) or external RAM if installed (see the Memory Map).

2.2.2 Creating and executing a program

The compiler tools provided allow you to create programs in either Assembly Language or C. The MCore micro however is optimized for C code, so most users will probably code in C. Here's how to compile, upload and run a simple C program.

Locate the `..\source` directory under the MCORE Software root directory and open the file called `HELLO.C` in a text editor. The file should look something like this:

```
#include "strlib.c"
main(){
      initserial();
      puts("\n Hello World! \n\n");
}
```

If you know C then this should be familiar to you.

Also in the `..\source` directory are example batch files that can be used to make compiling with the GNU C compiler easy. **NOTE**: although these tools use the DOS command line, they require Windows95/NT or newer operating system to run under. At the Windows DOS prompt enter:

```
ram hello
```

`Ram` is the name of the batch file and **`hello`** is the program name.

This will compile and link the hello.c program and create a Motorola .S19 (hex) record called `HELLO.S19`.

Now start your terminal program and reset the CMD2001 board to get the MBug prompt as described in Getting Started.

At the `Mbug >>` prompt type **`dl`** and press enter. This prepares Mbug to receive a Motorola hex file. Use your terminal program to upload the file we just created, `hello.s19`, to the development board.

When the file is finished uploading, you should see the `Mbug >>` prompt again. Type **`go 30000000`** to run the program. That's 30 followed by 6 zero's, which is this program's starting address.

If the program runs successfully you should see the "Hello World!" message on your terminal screen followed by a breakpoint message, which the linker code inserted at the end of main to return control to Mbug.

2.2.3 Configuring where program execution begins

Once your application software is executing satisfactory under the debugger, you will probably want to program it into EEPROM so that it starts when power is applied instead of Mbug. To do this, you can program it into the Internal Flash memory on the board, then change the jumpers so that this memory is used instead of the on-chip ROM.

When power or RESET is applied to the board, program execution will begin either Internally at the reset vector located at 0000:0000 OR Externally at the first address vector located in Chip Select 0 (CS0). This option is controlled by JP1 on the MCORE piggyback board.

If the piggyback JP1 is NOT installed program execution will begin at the RESET vector address `0000:0000` which will jump to the start of the Mbug program located in On-Chip ROM address 0000:0200.

If the piggyback micro jumper JP1 IS installed program execution will begin at memory address `2D00:0000` which is

assigned to **CS0** on POWER-ON / RESET. You can modify which device is mapped to **CS0** using these jumpers:

M-SEL 2 on	Fixed SRAM
M-SEL 4 on	Fixed Flash EEPROM
M1-SEL 3,6 on	External Memory Bank U5/6 (EPROM, EEPROM or RAM)

Chip Select 1 (CS1) is also available for your programs use and is mapped to memory starting at address `2F00:0000.` You can modify which device is mapped to **CS1** using these jumper settings:

M-SEL 1 on	Fixed SRAM
M-SEL 3 on	Fixed Flash EEPROM
M1-SEL 2,5 on	External Memory Bank U5/6 (EPROM, EEPROM or RAM)
M2-SEL 2,4 on	External Memory Bank U7/8 (EPROM, EEPROM or RAM)

See the Memory Map for more information.

NOTE: If using Internal Flash memory, don't forget to install JP1 on the main board to enable writing to it.

2.2.4 Programming the Internal Flash EEPROM

To program your software into internal flash memory, you should first relocate it to the flash memory address space. In this example, we want Internal Flash memory to be CS0, so it's starting address is 2D00:0000. In the previous compiler example, we used the RAM batch file that used the linker script file `RAM.LNK`, which locates the code at address 3000:0000. Now you want to use the **ROM** batch file which locates code at address 2D00:0000 using ROM.LNK. To compile the hello.c program using this file type:

```
rom hello
```

which will produce a new `hello.s19` file located at 0x2D000000. It also adds a default interrupt vector table and chip initialization code by linking the file INIT.S.

To program the new hello.s19 file you can use the supplied MCore utilities program. Start the Mbug program and at the prompt enter **`dl`** to start downloading. Upload the file called **`MCUTIL.S19`** in the **`..\utilities`** directory to the board the same way you loaded hello.s19 into internal RAM. When finished uploading, start it by entering **`go 30000000`**.

You should see the utilities menu. Choose Program Flash EEPROM from the menu and wait for the flash memory to erase. Now upload the new `hello.s19` file. The utility software will program it into the flash memory specified in the ROM.LNK file which is 0x2D000000.

When it is finished programming, all you have to do is install **`JP1`** on the piggyback board and the new program will now run whenever power or reset is applied. To return to Mbug simply remove **`JP1`** again.

If your application requires the Internal Flash memory for data or more code storage, you can program your software into EPROM's and install them in U5 and U6. You can then change the jumpers to make U5/6 assigned to CS0 (see the previous section) and assign Internal Flash memory to CS1, in which case it's memory address will change to 0x2F000000.

To program external EPROM's you need an external EPROM programmer, not included.

2.2.5 Additional Software

Included on the software CD are several example programs to help you get started, including LCD display, Keypad and Serial peripheral functions as well as string manipulation, clock and conversion routines. Feel free to use and distribute any of this software you like.

The GNU C compiler is very flexible and you should be able to find lots of free code examples out there.

All free updates to the software disk can be found on our web site at: **`www.axman.com`**.

3 HARDWARE

3.1 Specifications

Oscillator	32.768 KHz
External Clock	Jumper Selectable: 32, 16 or 8 MHz
Operating temperature	0°C to +70°C
Power requirement	8 - 25V @ 120 ma

3.2 Jumpers and Switches

3.2.1 Miscellaneous Jumpers

JP1	on enables write or programming of onboard Fixed Flash EEPROM
JP2	on enables low voltage backup of onboard fixed SRAM
JP3	on enables RTS0 from COM1
JP4	on enables CTS0 to COM1
JP5	on enables RXD1 from COM2

3.2.2 M-SEL Jumpers

These jumpers control device selection for Onboard Fixed Memory.

1	on enables CS1 to Fixed SRAM (U3 and U4)
2	on enables CS0 to Fixed SRAM (U3 and U4)
3	on enables CS1 to Fixed Flash EEPROM (U2)
4	on enables CS0 to Fixed Flash EEPROM (U2)

3.2.3 M1-SEL Jumpers

1,4	on enables CS2-0 and CS2-1 to M1 devices (U5 and U6)
2,5	on enables CS1-HI0 and CS1-HI1 devices (U5 and U6)
3,6	on enables CS0-H1 to M1 devices (U5 and U6)

3.2.4 M2-SEL Jumpers

1,3	on enables CS2-0 and CS2-1 to M2 devices (U7 and U8)
2,4	on enables CS1-HI0 and CS1-HI1 devices (U7 and U8)

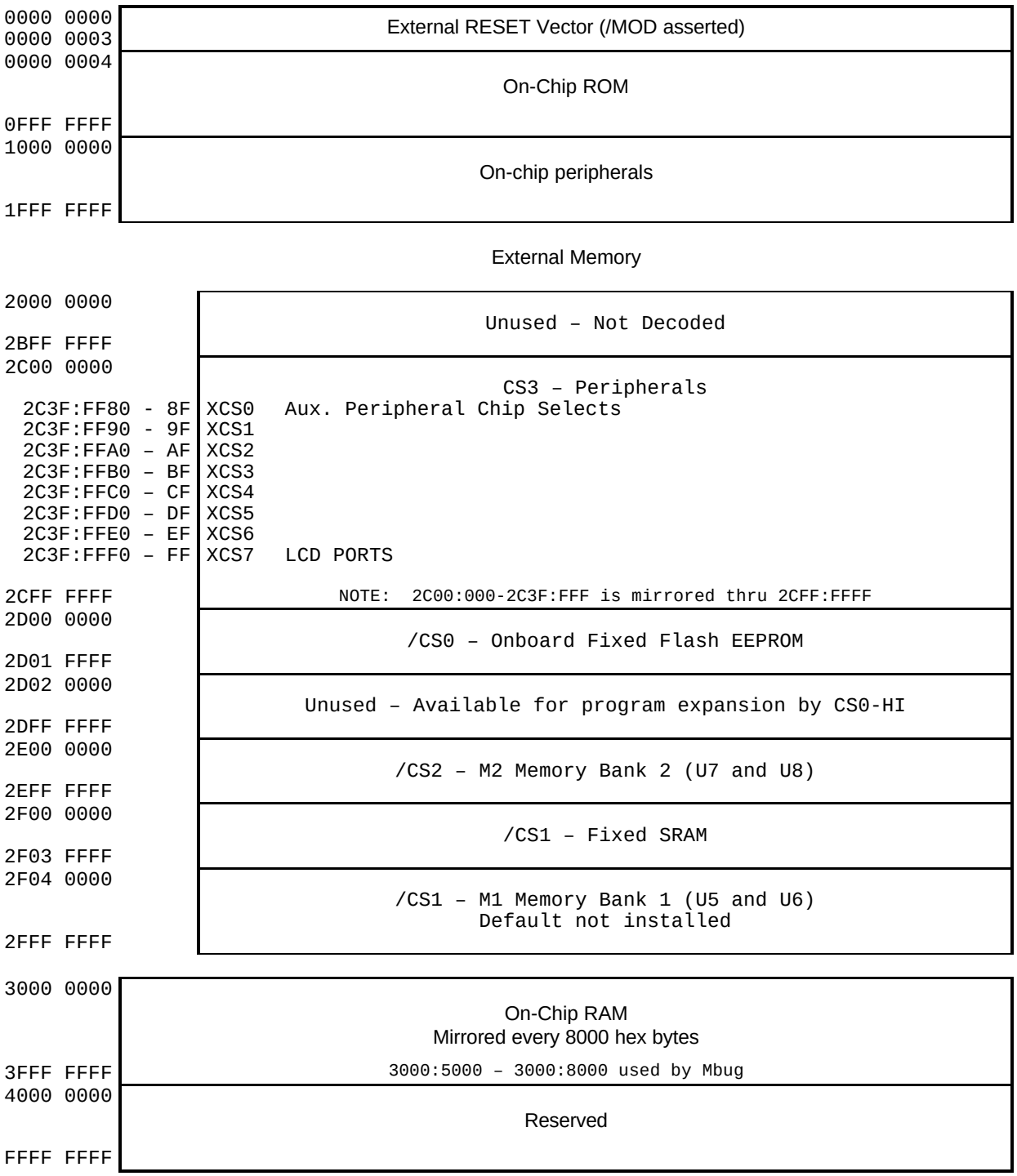
3.2.5 M10PT and M20PT Jumpers

M10PT = Device selection for M1 socket pair (U5 and U6)
M20PT = Device selection for M2 socket pair (U7 and U8)

1	2	3	4	5	6	Device Type
				x		32K Byte EPROMS (27256, low voltage devices)
x					x	32K Byte SRAMS (62256, low voltage devices)
x					x	32K Byte EEPROMS (27256, low voltage devices)
	x			x		128K Byte EPROMS (27010, low voltage devices)
x		x				128K Byte SRAMS (621001, low voltage devices)
x		x				128K Byte EEPROMS (28010, low voltage devices)
	x		x	x		128K Flash 29010

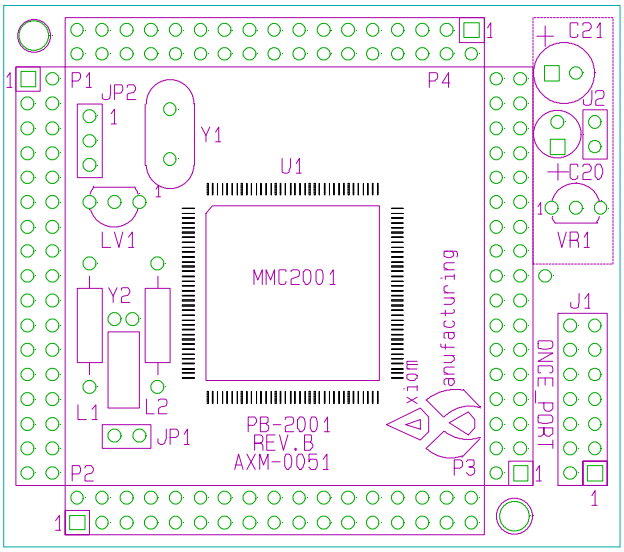
3.3 Memory Map

Following is the **DEFAULT** memory map for this development board:



3.4 PB2001 Controller Module

The PB2001 piggyback board is installed in a socket on the CMD2001 main board to allow replacement if necessary.



3.4.1 PB2001 Jumpers

JP1	on enables Reset in External Memory (see Memory Map)			
JP2	1	2	3	External Clock Speed Selection
				32 Megahertz Clock
	x	x		16 Megahertz
		x	x	8 Megahertz

- **J1** = external voltage input for optional onboard regulator.
- The Clock Oscillator may be disabled for external clock use by connecting P1 pins 15 and 16 together or by applying a ground to pin 16

3.4.2 ONCE Port

Not yet defined

3.4.3 PB2001 Connector Pinouts

The Motorola MMC2001 Microcontroller is attached to four dual row 17 pin connectors (34 pins each) which are configured as follows:

P1

ASIC1	1	1	2	2		ASIC2
CLKOUT	3	3	4	4		CLKIN
/RSTOUT	5	5	6	6		/RSTINX
GND	7	7	8	8		GND
VBATT	9	9	10	10		/LVRSTN
MODE	11	11	12	12		VSTBY
GND	13	13	14	14		GND
XOSC	15	15	16	16		EXOSC
VCC	17	17	18	18		MOD
VCC	19	19	20	20		VCC
D0	21	21	22	22		D1
D2	23	23	24	24		D3
D4	25	25	26	26		D5
D6	27	27	28	28		D7
D8	29	29	30	30		D9
D10	31	31	32	32		D11
GND	33	33	34	34		GND

P2

GND	35	1	2	36		GND
D12	37	3	4	38		D13
D14	39	5	6	40		D15
A0	41	7	8	42		A1
A2	43	9	10	44		A3
GND	45	11	12	46		GND
VCC	47	13	14	48		VCC
A4	49	15	16	50		A5
A6	51	17	18	52		A7
A8	53	19	20	54		VCC
VCC	55	21	22	56		A9
A10	57	23	24	58		A11
A12	59	25	26	60		A13
A14	61	27	28	62		A15
A16	63	29	30	64		A17
A18	65	31	32	66		A19
A20	67	33	34	68		A21

P3

RW	69	1	2	70		OE
/CS0	71	3	4	72		/CS1
GND	73	5	6	74		VCC
/CS2	75	7	8	76		CS3
/EB0	77	9	10	78		/EB1
TD0	79	11	12	80		/DE
TMS	81	13	14	82		TDI
TCK	83	15	16	84		/TRST
TEST	85	17	18	86		DB-RST
GND	87	19	20	88		VCC
INT7	89	21	22	90		INT6
INT5	91	23	24	92		INT4
INT3	93	25	26	94		INT2
INT1	95	27	28	96		INT0
COLUMN7	97	29	30	98		COLUMN6
COLUMN5	99	31	32	100		COLUMN4
COLUMN3	101	33	34	102		COLUMN2

P4

COLUMN1	103	1	2	104		COLUMN0
GND	105	3	4	106		VCC
ROW7	107	5	6	108		ROW6
ROW5	109	7	8	110		ROW4
ROW3	111	9	10	112		ROW2
ROW1	113	11	12	114		ROW0
SPI_MIS0	115	13	14	116		VCC
GND	117	15	16	118		SPI_MOSI
SPI_EN	119	17	18	120		SPI_CLK
SPI_GP	121	19	20	122		TXD0
RXD0	123	21	22	124		/RTS0
/CTS0	125	23	24	126		VCC
GND	127	25	26	128		GND
TXD1	129	27	28	130		RXD1
PWM0	131	29	30	132		PWM1
PWM2	133	31	32	134		PWM3
PWM4	135	33	34	136		PWM5

- Small numbers next to connector pin numbers are MC2001 package pin numbers for reference.
- The PB2001 contains the crystal oscillator and the low voltage RESET generator.
- See the MMC2001 hardware Reference Manual for complete pin information.

3.5 Ports and Connectors

3.5.1 BUS_PORT

/MOD	**1**	**2**	VCC
D0	**3**	**4**	D1
D2	**5**	**6**	D3
D4	**7**	**8**	D5
D6	**9**	**10**	D7
D8	**11**	**12**	D9
D10	**13**	**14**	D11
D12	**15**	**16**	D13
D14	**17**	**18**	D15
A0	**19**	**20**	A1
A2	**21**	**22**	A3
A4	**23**	**24**	A5
A6	**25**	**26**	A7
A8	**27**	**28**	A9
A10	**29**	**30**	A11
A12	**31**	**32**	A13
A14	**33**	**34**	A15
A16	**35**	**36**	A17
A18	**37**	**38**	A19
GND	**39**	**40**	GND

The BUS_PORT supports off-board memory devices as follows:

/MOD
Boot ROM control. Provides the capability of disabling the on-chip ROM and forcing CS0 to be used to select an external boot ROM.

D0 – D15
External 16-Bit Data Bus

A0 – A19
External Memory Address 0-19 for Data Bus access

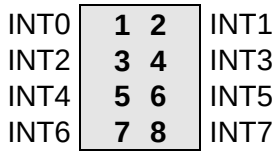
3.5.2 CONTROL_PORT

/CS0	**1**	**2**	/CS1
CS2	**3**	**4**	CS3
GND	**5**	**6**	VCC
XCS0	**7**	**8**	XCS1
XCS2	**9**	**10**	XCS3
XCS4	**11**	**12**	XCS5
XCS6	**13**	**14**	XCS7
GND	**15**	**16**	VCC
/OE	**17**	**18**	/RW
CS1-HI0	**19**	**20**	CS1-HI1
CS2-0	**21**	**22**	CS2-1
CS0-HI	**23**	**24**	P-SEL
EB0	**25**	**26**	/EB1
TD0	**27**	**28**	/DE
TMS	**29**	**30**	TDI
TCK	**31**	**32**	/TRST
GND	**33**	**34**	VCC

The CONTROL_PORT supports off-board memory and peripheral devices.

See the Memory Map for CS and XCS information.
See the MMC2001 Reference Manual for detailed peripheral information.

3.5.3 INT_PORT

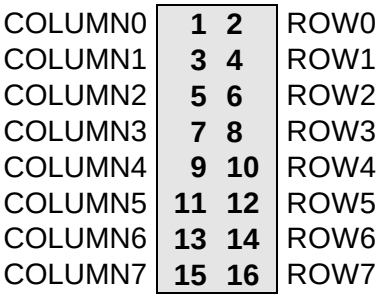


The INT_PORT allows external access to external interrupt sources INT0-INT7. See the MMC2001 Reference Manual for more information.

NOTE: INT7 may be used on PB2001 for external clock disable for low power modes.

3.5.4 KEY_PORT

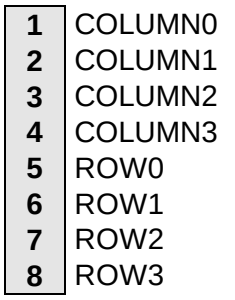
The KEY_PORT connector is a 16-pin connector that can be used to connect a keyboard device, up to an 8 x 8 matrix. The connector is mapped to the MMC2001 KPP keyport I/O lines as follows:



See the MMC2001 Reference Manual, section 14, for a full description of these pins and how to use this port.

3.5.5 KEYPAD Connector

The KEYPAD connector is an 8-pin connector that can be used to connect a 4 x 4 matrix keypad device. The connector is mapped to the MMC2001 KPP keyport I/O lines as follows:



See the MMC2001 Reference Manual, section 14, for a full description of these pins and how to use this port.

See the program KEYPAD.ASM for an example of using this port.

3.5.6 LCDPORT-1

The LCDPORT-1 Display interface is connected to the data bus and memory mapped to locations LCD-CS1 2C3F:FFF0 thru 2C3F:FFF1 (see Memory Map). Address FFF0 is the Command register, address FFF1 is the Data register.

The interface supports all OPTREX™ DMC series displays up to 80 characters and provides the most common pinout. Power, ground, and Vee are also available at the LCDPORT-1 connector. LCD-Vee is supplied by U15 and is adjusted by the R15 Potentiometer (adjustable resister).

See the file **KEYLCD.ASM** for an example program using this LCD connector.

VCC	**2**	**1**	GND
A0	**4**	**3**	LCD-Vee
LCD-CS1	**6**	**5**	/RW
D1	**8**	**7**	D0
D3	**10**	**9**	D2
D5	**12**	**11**	D4
D7	**14**	**13**	D6

Control Register: **2C3F FFF0**
Data Register: **2C3F FFF1**

3.5.7 LCDPORT-2

The LCDPORT-2 Display interface is connected to the data bus and memory mapped to locations LCD-CS1 2C3F:FFF0 thru LCD-CS4 2C3F:FFFD (see Memory Map).

This port allows much larger display addressing space than LCDPORT-1, depending on the type of display device connected. You can address multiple Command and Data registers (see below).

The interface supports all OPTREX™ DMC series displays up to 80 characters and provides the most common pinout. Power, ground, and Vee are also available at the LCDPORT-2 connector. LCD-Vee is supplied by U15 and is adjusted by the R15 Potentiometer (adjustable resister).

D6	**2**	**1**	D7
D4	**4**	**3**	D5
D2	**6**	**5**	D3
D0	**8**	**7**	D1
RW	**10**	**9**	LCD-CS1
LCD-Vee	**12**	**11**	A0
VCC	**14**	**13**	GND
unused	**16**	**15**	LCD-CS2
VCC	**18**	**17**	GND
LCD-CS4	**20**	**19**	LCD-CS3

Control Registers:
LCD-CS1 = **2C3F FFF0**
LCD-CS2 = **2C3F FFF4**
LCD-CS3 = **2C3F FFF8**
LCD-CS4 = **2C3F FFFC**

Data Registers:
LCD-CS1 = **2C3F FFF1**
LCD-CS2 = **2C3F FFF5**
LCD-CS3 = **2C3F FFF9**
LCD-CS4 = **2C3F FFFD**

3.5.8 SERIAL PORT

The SERIAL port connector maps to the MCC2001 ISPI and UART peripherals as follows:

SPI_MISO	**1**	**2**	SPI_MOSI
SPI_EN	**3**	**4**	SPI_CLK
SPI_GP	**5**	**6**	GND
VCC	**7**	**8**	GND
RXD0	**9**	**10**	TXD0
/CTS0	**11**	**12**	/RTS0
RXD1	**13**	**14**	TXD1
VCC	**15**	**16**	GND

3.5.9 COM-1

COM-1 is the default serial interface for the M-Bug Debugger.

The **COM-1** port has a Female DB9 connector that interfaces to the MMC2001 internal UART0 serial port. It uses a simple four wire asynchronous serial interface with hard wired Clear to Send (CTS) and Request to Send (RTS). These two logic level signals are coupled thru a RS232 level shifter to the COM1 connector.

See also Jumpers 3 and 4.

Pins 1,4 and 6 = default DTR, DSR handshake. Pin 7 = Request to send input, Pin 8 = clear to send output.

3.5.10 COM-2

The **COM-2** port has a Male DB9 connector that interfaces to the MMC2001 internal UART1 serial port. It uses a simple four wire asynchronous serial interface with hard wired Clear to Send (CTS) and Request to Send (RTS). These two logic level signals are coupled thru a RS232 level shifter to the COM2 connector.

See also Jumper 5.

Pins 1,4 and 6 = default DTR, DSR handshake. Pin 7 = Request to send input, Pin 8 = clear to send output. These 2 pins are shorted. NOTE also that RXD0 and TXD0 are swapped on this port.

3.5.11 TIMER

The TIMER port connector maps to the MCC2001 PWM and Clock signals as follows:

PWM0	1	2	PWM1
PWM2	3	4	PWM3
PWM4	5	6	PWM5
CLKIN	7	8	CLKOUT
VCC	9	10	GND

See the MMC2001 Reference Manual Section 15 for more information on these pins.

3.5.12 TB1

Vin	1
GND	2
+5v @ 100ma	3
3.3v @ 50ma	4

The TB1 is an Alternate Power connector. This can be used either as an input power supply to the board or to supply power to external devices.

3.5.13 BATT-PORT

1	RSTOUT Reset Out
2	RSTINX Reset Internal
3	LVRSTIN Low Voltage Reset In
4	VBATT (see MCC2001 Reference Manual)
5	VSTBY (see MCC2001 Reference Manual)
6	+Vcc 3.3v DC power supply
7	GND

The BATT-PORT provides connections for RESET and Low Power modes.

4 TROUBLESHOOTING

- If your target board uses EPROM's for code storage, you will need to install RAM in those sockets while debugging. After debugging is finished you can re-install the EPROM's for programming.
- After you add or change memory on the board, make sure you set the correct Expanded mode under the Config / Mode after reset menu, then press the Reset button.
- If you're trying to program flash and it doesn't write anything, be sure JP1 on the board (not the PB) is installed.