CMM-332

An enhanced version of the Motorola MC68332 Business Card Computer

xiom
anufacturing

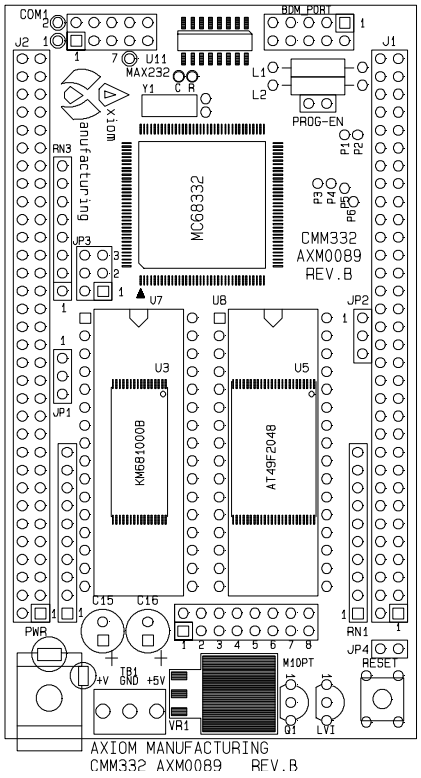
2813 Industrial Ln. • Garland, TX 75041 • (972) 926-9303 FAX (972) 926-6063
email: Gary@axman.com • web: www.axman.com

CONTENTS

Features

The CMM0332 is an enhanced version of the Motorola MC68332 BCC (Business Card Computer). With the addition of power supply, RS232 COM, and flash memory, the CMM0332 offers a versatile stand-alone or system plug-in control board. As a development system for the Motorola MC68332 Microcontroller, the CMM0332 is shipped with CPU32BUG installed in EPROM and has the standard BDM port. The system is Plug and Play with the supplied CPU32BUG Monitor/Debugger in the on board EPROM, 128K x 16 Flash EEPROM, 128K x 16 SRAM, Serial Cable, 9v 300ma Wall Plug, printed hardware manual and the UTL332 software disk with Assembler, programming utilities, support software.

- MC68332 Controller (16MHz)
  - CPU32 Core (32 bits)
  - 16M Byte address space
  - 16 Bit Data Bus
  - Programmable Chip Selects
  - 16 channel TPU
  - 32Khz crystal w/ PLL Clock (16MHz max)



- Standard fixed memory:
  - 128Kx16 (256K bytes) Flash EEPROM
  - 128Kx16 (256K bytes) SRAM
  - (Optional Larger Memory – contact factory)

- Two Configurable 32pin memory sockets for 32K to 2MByte ROM (CPU32BUG Installed)

- COM1 – 332 SCI w/ RS232 type header connection

- Back Ground Debug (BDM) Port

- Two 60 pin DIN I/O connectors

- Easy Power Connection and Tap points

- 7 to 25VDC input to 5 Power Supply

- Operating Power: 120ma @ 5V

The Axiom development system provides for low cost software debugging with the use of the Motorola CPU32BUG Monitor installed in PROM. Operation allows the user to locate code in the On-Board RAM, set Break Points, Trace, and display or modify registers or memory. After code is operational the user may relocate the code and program the development board Flash EEPROM for dedicated operation of new software. No additional hardware or software is required. For higher level debug, the BDM Port is available to connect a background debugger. Board is compatible with software compilers that provide an integrated debug interface.

GETTING STARTED

This section assumes you have just received your board from the manufacturer. If this is not the case then jumpers and switches may have been changed so that the board may not function as expected. In this case, see the "Jumpers and Switches" section of this manual and return everything to their "default" positions before proceeding.

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

1. 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 connector on the board.
2. Run a standard ANSI terminal communications program set to 9600 baud, N,8,1. Any terminal program will work, including the simple terminal that comes with MS Windows.
3. Apply power to the board by plugging in the 9V power transformer wall plug that came with the system.
4. If everything is working properly, you should see the utilities menu on in your terminal, similar to the following:

```
    ------------------------------
     CMM332 AXM-0089 Utilities
    ------------------------------

  d. Debug Monitor - CPU32Bug
  f. Flash EEPROM Utilities
  t. Test Hardware

  Select:
```

Your board is now ready to use. If you do not see the menu, press then release the RESET button on the board.

Software Development

The example utility software initializes the system clock to run this board at 16 MHz on power-up. You can modify this by changing the PLL register configuration in your software.

Software development on the CMM332 is best performed using a BDM tool connected to the BDM-PORT connector on the board. This provides real-time access to all hardware, peripherals and memory on the board. BDM software is also available for high-level source code debugging. The SDS Single-Step debugging software was used in this boards development and an example register configuration file for that software is provided on the utility disk, called `68332.CFG`.

When a BDM is not available, software development can be done using the CPU32Bug monitor software to upload your code to RAM and execute it. Type HELP at the monitor prompt for a list of commands. See the CPU32Bug manual for more information.

In either case, you can uploaded your application program to internal or external RAM and executed it from there during development, then program it into Flash to execute on power-up. You can program the onboard flash by selecting (f) from the utility menu. You will be prompted to send your S-Record (hex) file to the board. See the Tutorial section of this manual for more information.

TUTORIAL

The following tutorial sections were created to help you become familiar with the utilities and support software provided with this board.

Using the CPU32Bug Monitor

The CMM-332 ships with a modified version of the Motorola CPU32Bug Debugger/Diagnostics software programmed into the U7 and U8 EPROMS. See "Getting Started" for startup instructions.

Using this monitor program and any serial communications terminal program, you can display or modify memory and registers from any PC as well as upload and run programs in RAM with simple debugging functions such as trace and breakpoints. This is done by typing commands into the terminal program on the PC. Type help and hit <enter> for a complete list of available commands or see the CPU32Bug manual.

You can experiment with some of the commands like reading (MD) and modifying (MM) memory. Make sure you are modifying memory that is mapped to valid addresses or the monitor program will "hang" or throw an exception. You may then have to RESET the board to get the monitor back. See the Memory Map for The provided utility maps the External RAM starting at 0x0000 (with JP1 jumper set to 2-3).

NOTE: CPU32Bug uses the first 12K of external RAM (0000-2FFF) for internal stack, variables and vector space. You should locate your program above address 0x2FFF when using CPU32Bug.

You can upload and execute a program into RAM memory using CPU32Bug. A simple "hello world" program is provided for you on the disk as an example. Follow these steps to load and execute it from RAM:

1. At the CPU32Bug prompt > type lo and hit <enter>.

2. Select the send text file (or upload) command from your terminal program and locate the file named "HELLO_R.S19" included on the software disk. Send this file to the board.

3. If your terminal is set to Echo characters sent, you should see a each line of the S-Record file sent. The last line of the file starts with S7 and in this case contains the START address of the hello program, which is 3002. (note the last 2 characters are always the checksum).

4. If you do not receive a prompt back, hit the ENTER key several times.

5. Type go a000 at the prompt. You should see the phrase " Hello World! " echoed back to the screen, which is all this simple test program does. Hit any key and it displays the string again.

6. To return to the monitor program, press the RESET button on the board.

This example program was compiled with the Diab Data compiler. The source files scripts used are included on the utility disk as follows:

HELLO.C	Main source code file
SERIAL.C	Serial port I/O and ASCII text conversion routines
HELLO_R.DLD	Diab Linker script - specifies the programs memory organization
HELLO_R.BAT	DOS batch file used to build the program

Using a BDM

A BDM real-time debugger allows you to download to and modify internal and external memory and peripheral registers on the board in a completely user controlled state. You must configure the BDM software correctly to recognize this board. In your bdm software set the target processor setting to 68332.

The BDM software writes to a number of CPU registers before your application software is loaded, after it resets and establishes initial communication with the board. The values these registers are set to is the source of many BDM debugging problems.

BDM software saves these initialization values in a configuration file. The SDS software, for example, uses a file with the CFG extension under it's program \CMD directory (68332.CFG for this board). You can change the values written to these registers by manually editing this file or in the SDS Debug window, under the Target Configuration tab.

The Metrowerks CodeWarrior BDM software uses the "BDM/JTAG Configuration File" listed under "EPPC Target Settings" for register configuration. This text file can be modified in a text editor. NOTE: the manufacturer has not yet tested this board with the CodeWarrior product.

The following table contains the default register settings recommended by the board manufacturer which can be used as an example to debug and execute software from External RAM with a BDM. This assumes JP1 is set to its default position 2-3 which is CS0 and the program you are debugging is in the 0x0000:0000 - 0x0003:FFFF address range.

If you have trouble loading or executing the example software, you should first check these values against the current settings in your BDM software and verify the memory jumpers on the board are set correctly. Remove the BDM from the board then apply power to the board. Now try to re-connect the BDM to the BDM-PORT connector. Be sure the red stripe on the bdm pod cable lines up with pin 1 on the bdm connector.

ADDRESS	NAME	VALUE
0xFFFA00	SIMMCR	0x604F
0xFFFA04	SYNCR	0x7F01
0xFFFA21	SYPCR	0x33
0xFFFA44	CSPAR0	0x3FFF
0xFFFA46	CSPAR1	0x0155
0xFFFA48	CSBARBT	0x0C05
0xFFFA4A	CSORBT	0x78F0

ADDRESS	NAME	VALUE
0xFFFA4C	CSBAR0	0x0005
0xFFFA4E	CSOR0	0x78F0
0xFFFA50	CSBAR1	0x0405
0xFFFA52	CSOR1	0x78F0
0xFFFA54	CSBAR2	0x2003
0xFFFA56	CSOR2	0x7871

Using SDS Single Step

When the Single Step software starts, you should see the debug status window. All the settings in this window are automatically saved when you exit the program. Follow these steps to configure the software to debug the example "hello world" program on this development board.

1. Select the Connection tab. Make sure the settings match those used to connect the BDM to your PC. Set the delay to 0, unless you're having connection problems, in which case try 3.

2. Select the Processor tab and choose 68F375 or 68332 with no coprocessor.

3. Select the Registers tab and choose "Default.."

4. Select the Options tab and check all boxes under "Loading" and un-check the rest for now.

5. Select the Target Configuration tab. The Category should be General, the Processor should be set to the same as Processor. Each of the registers listed in this window are written by the BDM whenever it connects to the development board. It is important that these values match your hardware/software configuration in order to debug your software. If you're not sure what these registers must be set to, double-click on each of them and verify that they match with the tables above. Alternatively, you can manually edit the file in the programs \CMD directory, which will have the same name as the processor, with the .CFG extension. However, if you make changes in the Target Configuration tab, this file will be overwritten.

6. Select the File tab again. The "debug without a file" box should be un-checked. Select Browse and locate the file named "HELLO_R.ELF" on the utilities disk.

7. Make sure the BDM is connected to the board and it is powered.

The SingleStep program will first create its own debugging files based on the ELF file. It will then attempt to connect to the development board through the BDM. If this does not work, disconnect the BDM and power from the board, re-apply power to the board, press the RESET button, then re-connect the BDM to the board. You can get the debug window back from the File menu.

Once connection has been made with the board, the software will execute the .CFG script then start downloading the HELLO_R program to the board. When finished, close the debug status window. If you did not disable the "execute till main" option it will attempt to execute your

program until the main() function is reached, otherwise you should see an assembly listing, which contains startup code the compiler links at the front of your program.

Start your serial terminal program and make sure the serial cable is attached, as described under "Getting Started". Select the Green GO button or press F5. You should now see the " Hello World! " message in the terminal window. Press any key to repeat it. Some versions of SingleStep will insert a breakpoint at the end of the main loop, which you can delete. You can single step, examine and modify memory and other debugging stuff.

Programming the Flash Memory

After testing your program running out of RAM you will probably want to program it into Flash Memory so that it starts whenever power is applied to the board. To do this you must first change the starting address of your program to match that of the memory device being programmed. You must also initialize ALL of the CPU registers that you will be using, since your program will be running the show from power-up.

A version of the example "Hello World" program which does this for you is provided on the utilities disk. This version is called HELLO_A. It uses the same HELLO.C and SERIAL.C source code, but uses the following unique files:

INITA.S	Example Boot initialization assembly source file
HELLO_A.DLD	Diab Linker script - specifies the programs memory organization
HELLO_A.BAT	DOS batch file used to build the program

The output files produced from this build are called HELLO_A.S19 and HELLO_A.ELF. Notice that this version locates the code starting at address 0x0000:0000. This is important since the flash programming utility will relocate the code when programming it, and expects the program to start at 0, which is the start of the 68332 memory map.

To program the onboard flash, the jumpers should be set to their default positions.

1. JP3 should be set to 3, PROG_EN should be ON and JP2 set to 2-3.

2. If a BDM is connected to the board, disconnected it.

3. Reset the board to get the Utilities menu (see Getting Started for more information).

4. Select F. for flash programming

5. Select E. to erase the onboard flash chip

6. Select P. to program the flash. Select O for offset programming. Send the file named HELLO_A.S19 to the board using your terminal program (be sure to send in text mode if given the option).

7. The programming utility will program each data word as it is received. When finished, you will be returned to the programming menu.

To test your program to see if it boots, you need only remove JP3 and move JP2 to 1-2.

Remove power from the board then power the board back up. You should see the " Hello World! " message each time you hit a key. This should also happen if you press the RESET button on the board.

To return to the Utilities software (booting U7/U8), move JP3 back to position 3 and JP2 to 2-3.

Although this is a simple program example, you can use the same procedure for programming your own application using the CMM-332 board.

MEMORY MAP

Following is the **DEFAULT** memory map for the CMM-332 board as configured by the utility software in U7/U8. Chip Select Address Registers can be changed by your software to map memory any way you like. See the boot utility source code for this example. The Internal 68332 memory map is documented in the 68332 Users Manual starting at Paragraph 3.6.

0000 0000	**CS0** – External **SRAM** U3/4 see JP1 - RAM Chip Select Jumper	128K x 16 (256K byte) device installed 00:0000 - 03:FFFF
0004 0000	**CS1** – External **Flash** U5 see JP2 - Flash Chip Select Jumper	128K x 16 (256K byte) device installed 04:0000 - 07:FFFF
0008 0000	**CS2** – unused can be used by U7/8 EPROM, see JP3	
000C 0000	**CSBOOT** – U7/8 External **EPROM** Utilities / CPU32Bug software installed see JP3 Option Jumper	2) 128k x 8 devices installed (256K bytes) C:0000 - 0F:FFFF
0010 0000	**CS3** – unused can be used by U3/4 External SRAM, see JP1	
0014 0000	not mapped	
007F F000	On-chip Control and Status Registers see 68332 Users Manual	

Hardware Specifications

Oscillator	32.768 KHz Crystal
Clock	16 MHz Maximum
Operating temperature	0°C to +70°C
Power requirement	7 - 25V @ 120 ma Typical

Jumper Options

Memory Device Selection and Configuration

The CMM-332 board provides two standard and an optional external memory bank:

- Standard Low Power Static Ram 1 (U3 and U4), 128K x 16
- Standard Flash (U5), 128K x 16
- Standard Memory Sockets (U7 and U8), 32K Byte to 1M Byte EPROM's, EEPROM's, RAM, Flash

Each memory bank can be configured individually to operate from the MC68332 chip selects. Caution should be used not to place more than one memory bank on the same chip select.

JP1 - RAM Chip Select

The JP1 jumper selects which chip select accesses the on-board Static Ram memory bank (U3, 4). If no jumper is installed then the memory bank is disabled and idle. This memory bank is 128K x 16 bits (256K bytes) and operates at 70ns access time. This memory bank may be useful for emulating program operation prior to programming into external flash memory. Absolute addressing of the memory bank is determined by programming the associated chip select register. Default operation is CS0 with address base 0x00:0000.

Position 1-2 = CS3 Alternate chip select

Position 2-3 = CS0* **Default Position for monitor use**

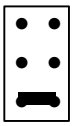
JP2 – Flash Chip Select

The JP2 jumper selects which chip select accesses the on-board Flash memory bank (U5). If no jumper is installed then the memory bank is disabled and idle. This memory bank is standard 128K x 16 (256K bytes) and operates at 120ns access maximum. The memory bank is available to the user to provide additional program code or data space and optional Boot start memory space. Absolute addressing of the memory bank is determined by programming the chip select register.

1 Position 1-2 = CSBOOT Position to Boot from user application in Flash.
2
3 Position 2-3 = CS1* **Default Position for Flash Programming Utility**

JP3 - Option Memory Chip Select

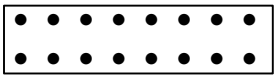
The JP3 jumper selects which chip select accesses the on-board Option Memory bank (U7, U8). If no jumper is installed then the memory bank is disabled and idle. This memory bank is 16 bits wide and requires two 8 bit memory devices for 16 bit (Word) wide access. If EPROM's are to be used, they should be programmed even and odd split with the even EPROM installed in the U7 socket. Devices from 32K byte to 128K byte can be used in the U7 and U8 sockets for 256K byte total space. If larger devices are used, the code size should be limited to 256K Bytes maximum and should be located in the upper memory area of the larger devices for proper operation. The memory bank may be used for Boot Start program code from Power On Reset of the board. The board ships with a CPU32BUG plus utility Monitor program in EPROM's installed in the sockets. Absolute addressing of the memory bank is determined by programming the chip select register. See MEM Option jumpers for device selection information.



1 Position 1 = CS2
2 Position 2 = CS1
3 Position 3 = CSBOOT* **Default Position to Boot Utilities**

MEMORY OPTION Jumpers

The MEMORY OPTION jumper block selects the TYPE of devices installed in the 32 pin U7 and U8 optional memory sockets. RAM, Flash, EEPROM, and EPROM may be installed. Following is a list of the jumper signal options and a chart showing standard jumper positions for typical devices. Note that U7/8 device pins without an option jumper installed are pulled to a high level. The option jumpers are setup in pairs for device pins so that Only One jumper is installed for each pair 1/2, 3/4, 5/6, 7/8, for a maximum of 4 jumpers.

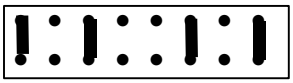


Position 1 = A15 to U7/8 pin 3
Position 2 = A16 to U7/8 pin 3
Position 3 = A16 to U7/8 pin 31
Position 4 = WE0 to U7/8 pin 31 * removed for write protection
Position 5 = A15 to U7 pin 29
Position 6 = WE0 to U7 pin 29 * removed for write protection
Position 7 = A15 to U8 pin 29
Position 8 = WE1 to U8 pin 29 * removed for write protection

32K / 128K Byte SRAM and 32K byte EEPROM Jumper Options

Manufacturer device types: KM68256, KM681000, TC551001, AT28C256 and others

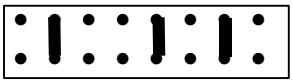
1 2 3 4 5 6 7 8



Standard device types: 62256, 68256, 681000, 551001, 28C256
Note: Positions 6 and 8 can be removed to write protect devices

32K, 64K, 128K, 256K Byte EPROM Jumper Options

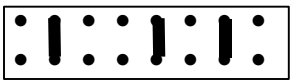
1 2 3 4 5 6 7 8



Standard device types: 27C256, 27C512, 27C010
Manufacturer device types: NM27C512 and others

64K Byte Flash, 128K Byte EEPROM / Flash Jumper Options

1 2 3 4 5 6 7 8



Standard device types: 28C010, 29C512, 29C010, 49F010
Manufacturer device types: AT28C010 and others

Note: Position 5 can be removed to write protect devices, write is 16 bits wide.

JP4 - Address A19 – A23 Enable

The JP4 option jumper provides Reset configuration selection option for the high order address lines and CS6 – 10 chip selects. Default position is installed. JP4 installed will select operation of A19 to A23 on the 68332 address bus. JP4 open or removed will select operation of CS6 to CS10 on the 68332 address bus. This selection can be changed by software but if chip select operation is needed, the jumper should be removed.

PROG-EN Jumper

The PROG-ENable option jumper provides write protection to the on-board flash memory bank. Jumper is open or off by default. The on-board flash memory cannot be written to if the PROG-EN jumper is not installed. For programming operations the jumper should be installed. The jumper prevents unscheduled writes to the on board flash memory.