CME12-A4

Development Board

Axiom ™
Manufacturing

717 Lingco Dr., Suite 209 • Richardson, TX 75081 • (972) 994-9676 FAX (972) 994-9170
Email: Gary@axman.com • web: http://www.axman.com

CONTENTS

GETTING STARTED

The Axiom CME12-A4 single board computer is a fully assembled, fully functional development system for the Motorola 68HC12A4 Microcontroller, complete with wall plug style power supply and serial cable. To get started quickly, perform the following test now to make sure everything is working correctly:

1. Install the software on your PC:
   - Create a directory on your PC hard drive for the utility software and copy the contents of the UTL12 disk to that directory. **NOTE**: the AX12.EXE utility on this disk requires DOS and will not run under NT.
   - If your board came with the AX12 for Windows software, run the SETUP.EXE program on the setup disk. This software will work with any 32-bit version of windows (95+), including NT.
   - If you have the Background Debug Module, install the BDM software from the floppy disk by running the SETUP.EXE program.
2. Connect one end of the supplied serial cable to a free COM port on your PC. Connect the other end of the cable to the COM1 port on the CME12-A4 board.
3. Apply power to the board by plugging in the wall plug power supply that came with the system.
4. If you have AX12 for Windows:
   - Run the AX12W program. From Edit/Options menu, select the COM port you're using on your PC.
5. If you do not have AX12 for Windows:
   - Change to the directory containing the utility software and execute the program: **AX12.EXE**.
   - Select the PC COM port that you're using and when asked for board type select CME12-A4.
   - From the main menu select "Terminal Window".
6. Press then release the RESET button on the CME12-A4 board now.
7. If everything is working properly, you should see the monitor prompt in the terminal window. Your board is now ready to use!

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

DEVELOPMENT PHILOSOPHY

Software development on the CME12-A4 can be performed using either the DBUG12 monitor utility programmed into EEPROM (U7) or a Background Debug Module (BDM) connected to the DEBUG connector on the PB module. Either of these tools can be used to assist in creating and debugging your program stored in RAM (see the Memory Map).

If you have a BDM, you may want to install RAM in socket U7 while debugging. This will let you locate your code in EEPROM memory space while debugging, for example $8000. You can then use the external RAM in U6 for your application.

For debugging under DBUG12, your program should locate itself above the internal registers and memory, for example $2000.

After satisfactory operation running under a debugger, your program can be written to EEPROM thru the BDM or by relocating its start address to EEPROM space, $8000 for example, then selecting "**Program Code Memory**" from the AX12 (or AX12W) utility program - make sure that bus size = 8. When programming is complete your program will run automatically when the CME12-A4 is powered on or RESET is applied.

To return to debug mode, you can re-program the DBUG12 monitor file: **AX-DB12.S19** or simply re-connect the BDM.

SOFTWARE

There are many useful programs on the UTL12 disk included with the CME12-A4 that can make developing projects easier. You can also download the latest version of this disk free at any time from our web page at: **http://www.axman.com**.

All of the utilities on the UTL12 disk require DOS, or a dos window running under MS Windows™. The main programming interface to the CME12-A4 board is the AX12 program. This program also provides an interface to the assembler and compiler software provided by Karl Lunt. The DOS version is included on the UTL12 disk and there is also a Windows version available from the manufacturer. The DOS version will not work with Windows NT.

Both versions of AX12 communicate with the board via its COM1 port and include a Terminal window for interfacing with other programs running on the CME12-A4, such as the monitor/debugger program DBug12.

In addition to the AX12 terminal, most communications programs will work with the CME12-A4. Even the Terminal program in Microsoft Windows™ will do an adequate job. Communications settings should be set to 9600 baud, 1 start, 1 stop, 8 data, no parity.

See the **README** file on the utility disk for a complete listing of all programs and files on the UTL12 disk.

Axiom also manufacturers a Background Debug Module for this board, which is a powerful real-time source level debugging tool. Contact Axiom for more information.

DOCUMENTATION

All of the documentation for the CME12-A4 hardware and software is available in electronic form. See the MANUAL.TXT file on the UTL12 disk or select HELP from the menu for the AX12 for DOS documentation.

Complete documentation for the AX12 for Windows and BDM Software products are available from the Help menu of these programs.

Owners manuals and Users guides for the M68HC12-A4 micro and programming interface is available in Adobe Acrobat (.pdf) on the Axiom web page **(www.axman.com)**. Here you will also find updates to this document as well as supporting part and schematic documentation. You can also download a free viewer for Acrobat (.pdf) files.

TUTORIAL

This section should help you get started with the specifics of the CME12-A4 development process. Be sure to read the rest of this manual as well as the documentation on the disk if you need further information.

You can develop 68HC12 software with the CME12-A4 board using either:

- your PCs serial port and the DBUG12 monitor utility with AX12 for DOS
- your PCs serial port and the DBUG12 monitor utility with AX12 for Windows
- your PCs Parallel port and a Background Debug Module (BDM) connected to the DEBUG connector on the boards PB module.

The following sections take you thru the complete development cycle of a simple "hello world" program using each of these development methods.

Using AX12 for DOS

Example Program

For this tutorial we'll use an assembly language program on the UTL12 software disk called HELLO.ASM. This is a simple program that just sends a text string to your PC serial port using the HC12 SCI0 (COM1) port. You can substitute your own program here if you wish but, to verify everything is working properly, it's a good idea to start with something simple.

1. If you haven't done so already, verify that the board is connected and operating properly by following the steps under "GETTING STARTED".
2. At your PC's DOS command line prompt, change to your UTL12 software directory.
3. Execute the command: **AX12** ↵
   This will launch the Axiom programming interface for the HC12 development boards.
4. Select "**Assembler**" from the main menu and input the file called HELLO.ASM which is located in the program directory. This will assemble our test source code.
5. If any errors are found, the program listing will be displayed on the screen which contains the errors. Otherwise, you should have a new file called **HELLO.S19** (a Motorola hex object file) in your directory.
6. Select "Terminal Window" from the menu.
7. Make sure JP1 is not installed then press and release the RESET button on the board.
8. You should see the monitor prompt. Press any key to start the debugger.
9. Type **LOAD** ↵
   This will prepare the monitor to receive a program.
10. Press the Page Up key and when prompted for a file name, select the file you just created called **HELLO.S19**
    then select [OK]. Your program will be sent to external RAM.
11. When the program is finished loading, hit the ENTER key to complete the upload.
12. Type **CALL 2400** ↵
    This tells the monitor to execute the subroutine at address $2400, which is the start of our test program.
13. If everything is working properly you should see the message "Hello World" echoed back to your terminal screen then, since we return at then end of our program, lines containing the internal register status followed by the monitor prompt.
14. If you didn't get this message, try going thru this tutorial once more then, if still no go, see the **TROUBLESHOOTING** section in this manual

You can modify the hello program to display other strings or do anything you want. The procedures for assembling your code, uploading it to the board and executing it are the same. The monitor has many powerful features such as breakpoints, memory dump and modify and program trace. Type HELP at the monitor prompt for a listing of commands available.

Programming the External EEPROM

When you're finished with program development, you will probably want to write your program to EEPROM so that it executes automatically when you apply power to the board. The following procedure is the easiest way to accomplish this:

1. Use a text editor to modify HELLO.ASM. You can use the built in "Edit" command in AX12 if you wish.

   - Change the start of your program defined by the , org MONSTRT to org PRGSTRT by commenting out the first (adding a comment character in front of) and removing the comment in front of the second. This change will re-map the code to start at address $8000 instead of $2000 which is the beginning of the external EEPROM.

   - Remove the comment character in front of the **LDS #STACK** line next. This will load the stack pointer when the program starts. It is important that this NOT be done when running your program under D-Bug12 because it must handle the stack.

   - Add a comment character in front of the first **RTS** line. This will allow the program to end gracefully in an endless loop, since it will not be returning to D-Bug12.

2. Select "**Assembler**" from the AX12 menu to re-assemble the modified HELLO.ASM program.. This will prompt you for that file name then execute the batch file DO_ASM.BAT which automates the assembly

process and creates a listing file.

3. Select "**HC12 Utilities**" from the menu. Follow the onscreen instructions for executing the utilities firmware. If you have trouble here, see the TROUBLESHOOTING section.
4. From the utilities menu, select "Program Code Memory" and when prompted for a file name, type:
   HELLO.S19
   then select [OK].
5. Follow the instructions on screen. Be sure to install **JP1** to enter programming mode and **U7_SELECT Jumper 10** to disable write protection.
6. When finished programming select "Auto Start Setup" from the menu. Set Auto Start to "YES" and change the starting address to 8000 hex, which is the new starting address of our HELLO program.
7. Remove JP1 to select HC12 expanded mode (run mode) again.
8. Select "Terminal" from the menu then cycle power or press RESET on the CME12-A4 board. Your new HELLO program should start automatically.

To return to development mode, simply restart the "HC12 Utilities" and turn off Auto Start. Don't forget JP1.

This method works well for small programs (less than 2K). If you want to use the entire onboard EEPROM memory area for your program, you can use "Program Code Memory" to write your program over the top of the debug monitor. You must also change the HC12 reset vector at $FFFE-$FFFF to point to the start of your program instead of the debug monitor.

You can always reprogram the debug monitor to EEPROM, the file is on the disk called **AX-DB12.S19**.

Using AX12 for Windows

Example Program

For this tutorial we'll use a sample assembly language program in the EXAMPLES sub-directory called HELLO.ASM. This small program sends a text string to your PC serial port.

1. If you haven't done so already, verify that the development board is connected and operating properly by following the steps under GETTING STARTED.
2. From the File Menu, select Assemble or press Control-A.
3. Change to the Examples directory and double-click the source file HELLO.ASM. This will launch a DOS window and execute the batch file DO_ASM.BAT to assemble the hello program.
4. If any errors were found they would be displayed on the screen. Press any key to close the DOS window and return to the AX12W program.

You should now have the new file **HELLO.S19** (a Motorola hex object file) in the EXAMPLES directory. The HELLO program is memory mapped to the development boards EXTERNAL RAM, starting at address $2000. Follow these steps to load and execute the program using the D-Bug12 Monitor program:

1. Apply power to the board or press RESET to make sure you have a D-Bug12 Monitor prompt.
2. In the AX12W main terminal window, Type LOAD and press ENTER. This will prepare D-Bug12 to receive a program.
3. From the File Menu select Send, or press Control-U.
4. Select the file named HELLO.S19 in the EXAMPLES sub-directory.
5. The program will be sent to the development board thru the serial port. Press the ENTER key to let D-Bug12 know you're finished uploading. If you're familiar with hex record format, you can see the first S1 record starts at $2000.
6. Type CALL 2000 and press ENTER.
   This tells D-Bug12 to execute the subroutine at address $2000, which is the start of our test program.

If everything is working properly you should see the message "Hello World" echoed back to your terminal screen and, since we return at then end of the hello program, a status message and lines containing the internal register status displayed by D-Bug12. You're then returned to the D-Bug12 prompt.

If you do not get this message try going thru this tutorial once more then, if still no go, see the **TROUBLESHOOTING** section. If the HELLO.ASM file has been modified you'll need to re-load it from the disk, or change it back for this to work.

You can modify the hello program to display other strings or do anything you want. The procedures for assembling your code, uploading it to the board and executing are the same.

D-Bug12 has many powerful features such as breakpoints, assembly/disassembly, memory dump and modify and program trace. Type HELP at the D-Bug12 prompt for a listing of commands.

Programming the External EEPROM

When you're finished with program development you'll probably want to write your program to EEPROM so that it executes automatically when you apply power to the board. There are 2 methods provided by the AX12 software to do this - programming the RESET Vector and Using the Auto-Start feature. If your program is less than 2K, using the Auto-Start feature is simplest because it doesn't over-write D-Bug12, allowing you to switch between it and your program without re-programming.

If your program is greater than 2K or uses interrupt vectors, you'll need to add a RESET vector to your program and must re-program D-Bug12 to use it again. The following tutorial will show you how to program your code using both methods:

1. Use a text editor to modify the HELLO.ASM file used in the previous section as follows:

   A. Change the start of your program defined by the , **org MONSTRT** to **org PRGSTRT** by commenting out the first (adding a comment character in front of) and removing the comment in front of the second. This change will re-map the code to start at address $8000 instead of $2000 which is the beginning of the external EEPROM.

   B. Remove the comment character in front of the **LDS #STACK** line next. This will load the stack pointer when the program starts. It is important that this NOT be done when running your program under D-Bug12 because it must handle the stack.

   C. Add a comment character in front of the first **RTS** line. This will allow the program to end gracefully in an endless loop, since it will not be returning to D-Bug12.

2. Save the modified hello program as HELLO2.ASM.
3. Return to AX12W and press Control-A. Assemble the new HELLO2.ASM program. If there are any errors fix them and try again.
4. Select Utilities from the Tools menu or press the Utilities button. Make sure your board is selected and follow the instructions in the dialog box to configure it for programming mode then press RESET on the board. Note that this board requires the 8-bit bus setting.
5. Select Continue to send the programming utilities to the board. **REMEMBER**: If power is removed or the board is RESET at any time, you MUST select Utilities and re-send these instructions again for the programming tools work. This is necessary because the utilities are executing from RAM.
6. Press the Program button and select your newly assembled HELLO2.S19 program. This will write it to the external EEPROM.
7. The final step is to tell the board to start your program instead of D-Bug12 whenever power is applied. To do this, select the AutoStart button. Check the box on the left and make the auto-start address the beginning of your program, which we changed to 8000 in step 1. Select OK and now the AutoStart is enabled.
8. To test your new program, remove power from the board and remove JP1 to select HC12 expanded mode (run mode) again. Your program should now start whenever power is applied.

The other method of programming your code to EEPROM so that it starts on powerup is exactly the same, except that instead of using the D-Bug12 AutoStart, you simply program the HC12 RESET vector in your code. Here's how to do that:

Use a text editor to modify the HELLO2.ASM file. Remove the comment characters in front of the 2 reset vector lines at the end of the file. Save this new version as HELLO3.ASM.

1. Press Control-A under AX12W and assemble HELLO3.ASM.
2. Select the Utilities button and follow the instructions to re-load the utilities (don't forget JP1).
3. Turn OFF AutoStart by Un-Checking the box in the AutoStart dialog box.
4. Program the new HELLO3.S19 file using the Program button.
5. Remove JP1. Your new program now will run at RESET or power-on in place of D-Bug12.

To restore the D-Bug12 monitor, follow the steps above for loading the utilities and programming a file, substituting the file **AX-DB12.S19** located in the AX12W program directory.

Using the Axiom BDM

If you purchased the Axiom 68HC12 BDM, the easiest way to get started is to become familiar with the BDM software. If you have not done so already, install the BDM software now.

After installing and running the BDM software, you must **Configure** it to work properly with the CME12-A4 board. Select the menu item Config / Configure to see the configuration dialog box.

Make sure the Device select is set to **812A4**. Also the reset Macro file should be selected and point to the directory you installed the BDM software into. There is an example reset macro called RESET.MAC that you should use for now, you can change this later if you like. The macro file will enable the external bus even if it is jumper'd to Single Chip Mode.

To debug code using External RAM set the "Mode after reset" option to Expanded Narrow. Whenever the BDM software is RESET, it will use External Memory addressing and will go fetch the address at $FFFE-$FFFF as the reset vector and set the Program Counter to this address.

After closing the configuration dialog, click the RESET button in the BDM software so the configuration changes will take effect. The memory will most likely change and the program counter will reset to whatever value is in external RAM location $FFFE. If you have RAM installed there, you can modify this value using the data window. It's a good idea to set this word in the application program you're debugging.

Example Program

Included on the software disk is a program called HELLO.ASM. This is a simple program that outputs a text string to the HC12 serial port. It can be modified and assembled using the included free assembler, but for now just load it into the BDM by selecting File / Load S19 or Hex from the menu. The file name is HELLO.S19.

This program starts at address $8000, but since it does not include a RESET vector, you must change the Program Counter manually. Double click the PC address in the Regs window. Type in 8000 and click OK. The Program window should now change to the beginning of the Hello program at address $8000.

Since this program outputs text to the serial port, you need some way of monitoring that port. If you have not done so already, connect a serial port from your PC to the development board as described under Getting Started. Run the AX12 program and select Terminal Window (or any windows terminal program set to 9600,N,8,1) to monitor the serial port.

Switch back to the BDM program and select GO. To see the result of the program switch back to the Terminal, you should see the message "Hello World" displayed there.

You can use AX12 for DOS or Windows to write your program to external EEPROM.

DEBUG MONITOR

The Debug/Monitor software provides an embedded means for uploading your application software to RAM and executing it in a controlled environment using software breakpoints, trace and memory monitoring features. The Debug software itself resides in the on-board EEPROM devices shipped from the factory. The debug/monitor shipped with this board was originally written by Motorola and modified by Axiom and is called D-Bug12. The code for this program is available on the UTL12 disk in the file called **AX-DB12.S19**.

Unfortunately, the D-Bug12 software resides at address $A000-$FFFF starting in the Program Memory Expansion Window Page and is too large to allow complete use of the program page window.

To execute the Debug/Monitor software JP1 must be OFF to select HC12 expanded mode (run mode).

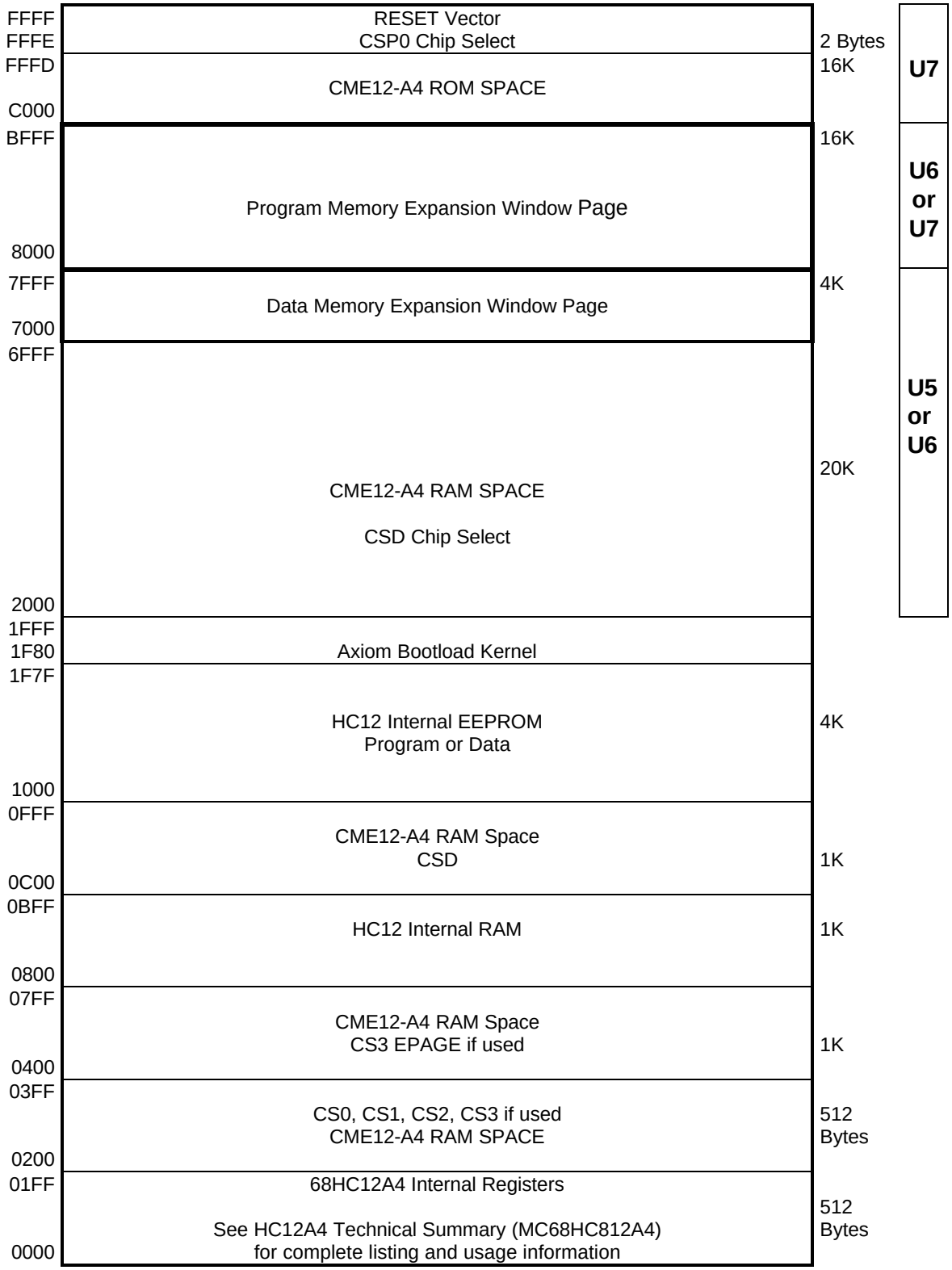
You must also connect the CME12-A4 COM1 serial port to your PC COM port with the supplied serial cable and start the AX12 program "Terminal" window. When power is applied to or RESET pressed on the CMD12-A4 the Debug/Monitor prompt will display in the terminal window.

Type HELP at the > prompt to see a current list of commands. If you are familiar with monitor software, such as the Motorola Buffalo monitor, this should be very familiar.

Command	Parameters	Description
ASM	<Address>	Assemble/Disassemble
	<CR>	Disassemble next instruction
	<.>	Exit assembly/disassembly
BAUD	<baudrate>	Set communications rate for the terminal
BF	<StartAddress> <EndAddress> [<data>]	Fill memory with data
BR	[<Address>]	Set/Display user breakpoints
BULK		Erase entire on-chip EEPROM contents
CALL	[<Address>]	Call user subroutine at <Address>
G	[<Address>]	Begin/continue execution of user code
GT	<Address>	Execute user code to address (temp breakpoint)
HELP		Display this D-Bug12 command summary
LOAD	[<AddressOffset>]	Load S-Records into memory
MD	<StartAddress> [<EndAddress>]	Memory Display Bytes
MDW	<StartAddress> [<EndAddress>]	Memory Display Words
MM	<StartAddress>	Modify Memory Bytes
	<CR>	Examine/Modify next location
	</> or <=>	Examine/Modify same location
	<^> or <->	Examine/Modify previous location
	<.>	Exit Modify Memory command
MMW	<StartAddress>	Modify Memory Words (same subcommands as MM)
MOVE	<StartAddress> <EndAddress> <DestAddress>	Move a block of memory
NOBR	[<address>]	Remove One/All Breakpoint(s)
RD		Display all CPU registers
RM		Modify CPU Register Contents
T	[<count>]	Trace <count> instructions
UPLOAD	<StartAddress> <EndAddress>	ↄ Dump Memory in S19 record format
VERF	[<AddressOffset>]	Verify S-Records against memory contents
PC SP X Y A B D	<Register Value>	Set register contents
S XM H IM N Z V C	<CCR Status>	Set Status Bit (1 or 0)

MEMORY MAP

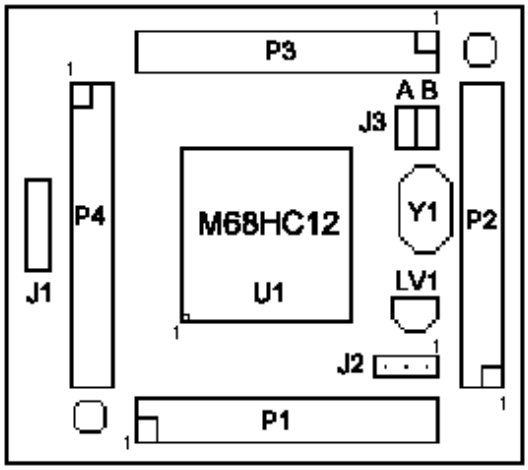
Following is the **DEFAULT EXPANDED** memory map for this development board:



HARDWARE

PB68HC12A4 CONTROLLER MODULE

The PB68HC12A4 is installed on the CME12-A4 board as a standard to allow replacement if necessary. The CME12-A4 will mount the MC68HC812A4 Microcontroller directly, please contact the factory if this option is preferred.



OPTIONS AND CONNECTIONS

The Options Jumpers on the PB68HC12A4 should not be used while installed on the CME12-A4 board. The CME12-A4 performs the necessary option configuration, refer to OPERATING MODES.

J1 SCI0 Header

The J1 header provides access to the HC12 SCI0 serial channel.

Pin 1 HC12 TxD0
Pin 2 HC12 RxD0
Pin 3 +Vdd (5 volts)
Pin 4 Vss (Ground)

J2 Debug Header

The J2 header is an auxiliary Background Debug Port.

Pin 1 RESET active low
Pin 2 Vss Ground
Pin 3 BGND debug pin

J3 Mode Option

J3 should be left open while the PB68HC12A4 is installed on the CME12-A4 board.

P1, P2, P3, P4 I/O Connectors

Refer to Port Connectors section for details.

FEATURES and OPERATION

Y1 Crystal Oscillator

Y1 is 16.00MHz standard. This provides a instruction clock / bus speed of 8mhz to the CME12-A4.

LV1 Reset Generator

LV1 is a voltage detector that will generate an active low RESET state if Vdd is below +4.4 VDC. This duplicates the operation of the CME12-A4 RESET Generator.

Bootloader Firmware

The MC68HC812A4 is pre-programmed with bootload firmware that operates in conjunction with the AX12 Utility software to provide a low cost debugging and programming environment for the M68HC12. The 64 Byte firmware is programmed into the internal HC812A4 EEPROM and becomes operational when the HC812A4 is in Single Chip Mode to support AX12 utility operations.

Firmware Memory

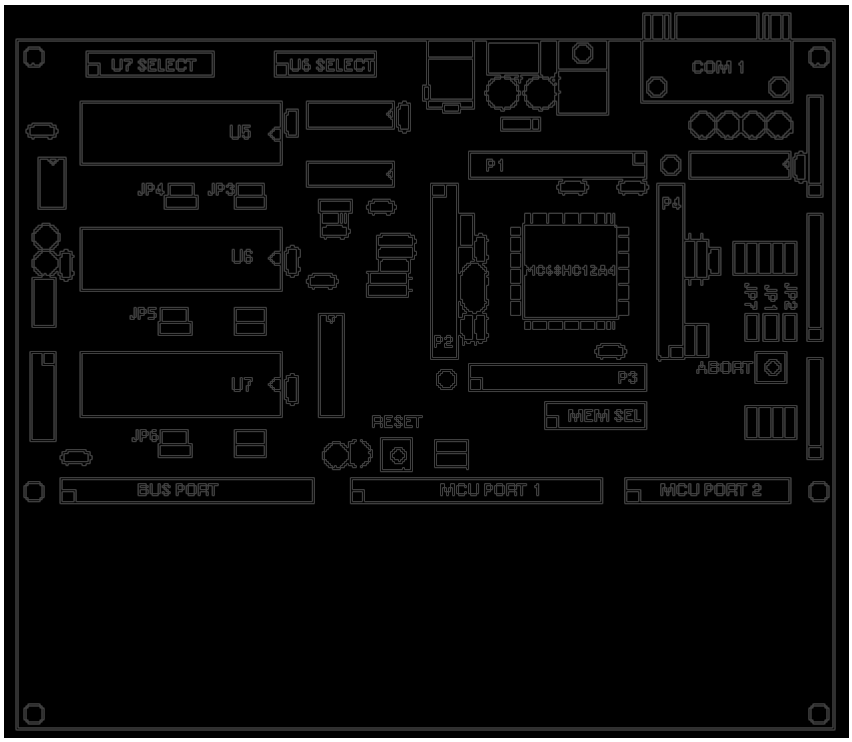
Single Chip Mode (normal)	$FF80 - $FFBF and $FFFE/FFFF = Reset Vector
Expanded Modes (default)	$1F80 - $1FBF and $1FFE/1FFF = $FF80

Caution should be used when programming or erasing the internal M68HC812A4 EEPROM to assure the bootload firmware is not erased or corrupted. The EEPROT Register (Location $00F1 default) bits 0 and 1 should be set high during the users software initialization sequence to protect the bootload firmware.

If the bootloader firmware is erased or corrupted it can be re-installed with the Monitor/Debugger operating in Normal Run Modes by loading the Bootload.S19 file and executing a G $800 instruction. If the Monitor/Debugger has been overwritten in the on-board EEPROMs also, then a EPROM programmer must be used to reprogram the Eprom(s) with the Monitor S19 files to restore the Monitor operation and then reload the Bootloader.

If Single Chip Mode is the desired mode of operation after development, the Interrupt Vector Table will need to be installed, which will require the Monitor/Debugger. The Monitor's Memory Modify (MM) command should be used to perform a Clear of EEPROT register ($F1) and installation of vector addresses ($1FC0 - $1FFC in expanded mode) for the interrupt vectors in use.

CME12-A4 BOARD



MEM_SEL (CHIP SELECT) Option Jumpers

The MEM_SEL Memory Select Option Jumpers enable or disable the 68HC12A4 predefined chip selects for use by the On-Board Memory Controller.

	Position	**Function** ✓= default on
✓	**1**	CSP0 → U7
	2	CSP1 → U7
✓	**3**	CSP0 → U6
	4	CSP1 → U6
	5	CSD → U6
	6	CS3 → U6
✓	**7**	CSD → U5
	8	CS3 → U5

- CSP0 must be enabled for on-board program or the Monitor/Debugger to operate.
- Only one chip select may be enabled for each device.

MEMORY DEVICE SELECTION

Memory Devices installed on the CME12-A4 board are configured for type and size using the JP3 to JP6, U6_SELECT and U7_SELECT option jumpers. Devices installed in the U5 socket may be SRAM from 8K to 128K bytes and are optioned with JP3 and JP4. Devices installed in the U6 socket may be SRAM or EEPROM and are optioned with the U6_SELECT jumpers. The U7 socket may be EPROM, EEPROM, or FLASH EPROM type devices and are optioned with the U7_SELECT jumpers.

The factory setting for the jumpers should be correct for the memory devices that came with your board. If you add or modify the type or size of memory, you must change the following jumpers accordingly.

U5_SEL

JP3 Installed	Write Enable to Device
JP4 Installed	A13 Enable (8K byte option only, installed for larger devices)

U6_SEL

x **=** Jumper Installed 1 = Jumper is WRITE ENABLE

	Device	**Size** (in bytes)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
✓	62256	32K RAM	x					x		1
	28256	32K EEPROM	x					x		1
	62010	128K RAM	x		x			x		1
	28010	128K EEPROM		x		1			x	
	62040	512K RAM	x		x		x			1
	28040	512K EEPROM		x		1	x		x	

You can determine the size of a memory device by reading the label on top of the chip. Memory devices that contain 256 in the part number are usually 32K byte. Those with 512 are usually 64K. If you do not recognize the memory type you can look up the part number in a catalog or device manual. If the chip is by Atmel or XICOR it is probably an EEPROM.

U7_SEL

x = Jumper Installed 1 = Jumper is WRITE ENABLE. Protects EEPROM and FLASH from accidental overwrite if removed after programming.

	Device	Size (in bytes)	1	2	3	4	5	6	7	8	9	10
	27256	32K ROM		x						x	x	
✓	28256	32K EEPROM	x							x		1
	27512	64K ROM		x						x	x	
	29512	64K FLASH		x				1			x	
	27010	128K ROM		x						x	x	
	28010	128K EEPROM	x					1			x	
	29010	128K FLASH		x				1			x	
	27020	256K ROM		x					x		x	
	29020	256K FLASH		x				1	x		x	
	27040	512K ROM		x		x	x		x		x	
	28040	512K EEPROM	x		x			1	x		x	
	29040	512K FLASH		x	x			1	x		x	
	27080	1024K ROM		x		x	x		x		x	

Address Decoding and Expanding Memory

The CME12-A4 ships standard with 64K bytes of on board memory. The 68HC12 Microcontroller can access up to 4 Megabytes of program space (CSP)/CSP1) memory or 1 Megabyte of data space memory (CSD or CS3 EPAGE) using special memory expansion address windows and memory page operations. On-Board RAM (U5) space is accessed with HC12 chip select CSD normally. Access to the RAM is also provided via the CS3 EPAGE or CSP1 chip select if enabled. On-board ROM (U7) space is accessed with HC12 CSP0 normally and CSP1 optionally. The U6 socket is provided to increase either RAM or ROM type memory.

An example to expand memory space is to set the appropriate bits in the HC12 WINDEF, DPAGE, PPAGE, and MXAR registers to enable memory paging with the HC12 chip selects. If CSD expansion is enabled with DPAGE, memory between $7000 - $7FFF is available as 256 x 4K data pages of which the lower 128 pages may be accessed on-board. The DPAGE register contains the data page number currently being accessed. Also available are 256 x 16K program pages from $8000 - $BFFF if the PPAGE is enabled. The PPAGE register contains the program page number.

Off-board memory expansion is possible via the BUS_PORT and MCU_PORT1 Connectors. Section 10 of the Motorola HC12 Reference Manual explains the technique in detail. Also see Section 8 of the MC68HC812A4 Technical Data book for complete configuration register and chip select information.

PORT CONNECTORS

PB68HC12A4 - I/O Connectors

The Motorola M68HC12 Microcontroller is attached to four dual row 14 pin connectors (28 pins each) which are configured as follows:

P1

Vss	1	**1**	**2**	2	Vdd
PJ0	3	**3**	**4**	4	PJ1
PJ2	5	**5**	**6**	6	PJ3
PJ4	7	**7**	**8**	8	PJ5
PJ6	9	**9**	**10**	10	PJ7
A16/PG0	11	**11**	**12**	12	PG1/A17
A18/PG2	13	**13**	**14**	14	Vdd
Vss	15	**15**	**16**	16	PG3/A19
A20/PG4	17	**17**	**18**	18	PG5/A21
BKGD	19	**19**	**20**	20	PD0/D0
D1/PD1	21	**21**	**22**	22	PD2/D2
D3/PD3	23	**23**	**24**	24	PD4/D4
D5/PD5	25	**25**	**26**	26	PD6/D6
D7/PD7	27	**27**	**28**	28	PC0/D8

P2

D9/PC1	29	**1**	**2**	30	PC2/D10
D11/PC3	31	**3**	**4**	32	PC4/D12
D13/PC5	33	**5**	**6**	34	PC6/D14
D15/PC7	35	**7**	**8**	36	PE0/XIRQ
IRQ/PE1	37	**9**	**10**	38	PE2/R/W
LSTR/PE3	39	**11**	**12**	40	/RESET
Vss	41	**13**	**14**	42	Vdd
Vddpll	43	**15**	**16**	44	XFC
Vsspll	45	**17**	**18**	46	EXTAL
XTAL	47	**19**	**20**	48	PE4/ECLK
MODA/PE5	49	**21**	**22**	50	PE6/MODB
ARST/PE7	51	**23**	**24**	52	PB0/A0
A1/PB1	53	**25**	**26**	54	PB2/A2
A3/PB3	55	**27**	**28**	56	PB4/A4

P3

A5/PB5	57	**1**	**2**	58	PB6/A6
A7/PB7	59	**3**	**4**	60	PA0/A8
A9/PA1	61	**5**	**6**	62	PA2/A10
A11/PA3	63	**7**	**8**	64	PA4/A12
A13/PA5	65	**9**	**10**	66	PA6/A14
A15/PA7	67	**11**	**12**	68	PF0/CS0
CS1/PF1	69	**13**	**14**	70	PF2/CS2
CS3/PF3	71	**15**	**16**	72	PF4/CSD
CSP0/PF5	73	**17**	**18**	74	PF6/CSP1
PH0	75	**19**	**20**	76	PH1
PH2	77	**21**	**22**	78	PH3
Vdd	79	**23**	**24**	80	Vss
PH4	81	**25**	**26**	82	PH5
PH6	83	**27**	**28**	84	PH7

P4

V_{RH}	85	**1**	**2**	86	V_{RL}
PAD0	87	**3**	**4**	88	PAD1
PAD2	89	**5**	**6**	90	PAD3
PAD4	91	**7**	**8**	92	PAD5
PAD6	93	**9**	**10**	94	PAD7
V_{DDA}	95	**11**	**12**	96	V_{SSA}
RxD0/PS0	97	**13**	**14**	98	PS1/TxD0
RxD1/PS2	99	**15**	**16**	100	PS3/TxD1
SDI/PS4	101	**17**	**18**	102	PS5/SDO
MOSI/PS6	103	**19**	**20**	104	PS7/SCK
PT0	105	**21**	**22**	106	PT1
PT2	107	**23**	**24**	108	PT3
PT4	109	**25**	**26**	110	PT5
PT6	111	**27**	**28**	112	PT7/PAI

- Small numbers next to Connector pin Numbers are MC68HC812A4 package pin number for reference.
- The PB68HC12A4 contains the crystal oscillator and an additional RESET generator.
- See the M68HC812A4 Technical Manual or PB68HC12A4 Manual for more information.

BUS_PORT

GND	1	2	D11
D10	3	4	D12
D9	5	6	D13
D8	7	8	D14
MA0	9	10	D15
MA1	11	12	MA2
MA10	13	14	MA3
/OE	15	16	MA4
MA11	17	18	MA5
MA9	19	20	MA6
MA8	21	22	MA7
MA12	23	24	MA13
/WR	25	26	Y0
Y1	27	28	Y2
Y3	29	30	Y4
Y5	31	32	/IRQ
+5V	33	34	CS3
R/W	35	36	Y6
ECLK	37	38	Y7
GND	39	40	/ RESET

MCU_PORT 1

+5V	1	2	GND
PA6/A14	3	4	PA7/A15
PG0/A16	5	6	PG1/A17
PG2/A18	7	8	PG3/A19
PG4/A20	9	10	PG5/A21
+5V	11	12	GND
PD0/D0	13	14	PD1/D1
PD2/D2	15	16	PD3/D3
PD4/D4	17	18	PD5/D5
PD6/D6	19	20	PD7/D7
+5V	21	22	GND
PJ0	23	24	PJ1
PJ2	25	26	PJ3
PJ4	27	28	PJ5
PJ6	29	30	PJ7
PH0	31	32	PH1
PH2	33	34	PH3
PH4	35	36	PH5
PH6	37	38	PH7
V_{RH}	39	40	V_{RL}

MCU_PORT 2

+5V	1	2	GND
PAD0	3	4	PAD1
PAD2	5	6	PAD3
PAD4	7	8	PAD5
PAD6	9	10	PAD7
+5V	11	12	GND
PT0	13	14	PT1
PT2	15	16	PT3
PT4	17	18	PT5
PT6	19	20	PT7
+5V	21	22	GND
XIRQ/PE0	23	24	PE7/ARS
	25	26	XFC

The **BUS_PORT** supports off-board memory or peripheral devices. The **MCU_PORTs 1 & 2** provide access to the peripheral features and I/O lines of the HC12.

BUS_PORT Signals

D8 - D15 Data Bus in Narrow Expanded mode or for 8 bit accesses in Wide Mode.

A0 - A13 Memory Address 0 to 13, Driven by Bus Size Switch for bus addressing.

/OE Memory Output Enable signal, Active Low. Valid with ECLK and R/W high.

Y0 - Y7 Auxiliary Chip Selects, 16 Bytes each located at $380 hex to $3FF hex driven by HC12 CS2 chip select. MEM_SEL option jumper 3 to enable.

/WR Memory Write Enable signal, Active Low. Valid with ECLK high and R/W low.

/IRQ HC12 IRQ (PE1) Interrupt Input.

CS3 HC12 chip select CS3 (PF3). BUS_PORT can expand to memory or peripheral devices using the CS3 chip select.

R/W HC12 Read/Write (PE2) control signal.

ECLK HC12 ECLK (PE4) bus clock signal. Stretch should be enabled in software.

/RESET CME12-A4 active low RESET signal.

MCU_PORT 1 Lines

PG0/A16 – PG3/A19 HC12 Port G I/O lines, available if expanded memory space not used.

PH0 – 7 Port H I/O lines. Also used as the CME12-A4 Keypad Port

Vrh/Vrl HC12 A/D Converter Reference Pins. See A/D Reference Section.

PJ0 – PJ7 HC12 Port J lines. Also used as CME12-A4 COM2 and SS_Port control.

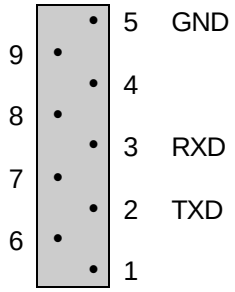
MCU_PORT 2 Lines

PAD0 – PAD7	HC12 Port AD is an input port or the A/D Converter inputs
PT0 – PT7	HC12 Port T is a general purpose port or Timer I/O port
PD0/D0 – PD7/D7	HC12 Port D lines, Low Byte Data Bus in Wide Expanded Mode.
PE7/ARST	HC12 Port E7 is available as a general purpose I/O.
PE0/XIRQ	HC12 Port E0 or XIRQ interrupt input is used as the ABORT Switch while the Debug/Monitor is operating.

SERIAL COM PORT

COM1 interfaces to the HC12 internal SCI0 serial port (I/O PS0 and PS1) and is a simple three wire asynchronous serial interface with hard wired Clear to Send (CTS) and Data Terminal Ready (DTR). These two logic level signals are coupled thru an RS232 level shifter to the COM1 connector.

COM1 DB9S Style Connector



- Permanent jumpers between following pins:
  4 → 1 and 6 (DTR/DSR/DCD)
  7 → 8 (RTS/CTS)
- COM1 is set to connect directly to a PC serial port with a straight thru type of cable (supplied).

J1 Header

The J1 header allows access to the HC12 SCI Serial Channel Pins.

1	HC12 PS1/TxD0
2	HC12 PS0/RxD0
3	HC12 PS2/RxD1
4	U2 RX SPARE OUTPUT
5	HC12 PS3/TxD1 and U2 TX SPARE INPUT
6	U2 TX SPARE OUTPUT (RS232 Level)
7	U2 RX SPARE INPUT (RS232 Level)
8	Vss (Ground)

NOTE: An additional COM2 Port can be attached by connecting J1 pins 4 and 5, and using J1 pins 6,7 and 8 for the serial port connector.

SS_PORT Connector

The Simple Serial connector can be used to communicate with external devices through the SPI Port features of the 68HC12. Up to five separate SPI serial devices can be supported by the SEL lines defining PJ4 - PJ7.

See the files **`SS-AD12.ASM`** and **`SS-DA12.ASM`** for example programs using this connector. Prebuilt Simple Serial devices with software drivers are available from the manufacturer.

			----- SPI ----				---- Select Lines ----			
		MISO	MOSI	SCK	SS	SEL1	SEL2	SEL3	SEL4
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
+5	GND	PS4	PS5	PS6	PS7	PJ4	PJ5	PJ6	PJ7

LCD_PORT Connector

The LCD Display interface is connected to the data bus and memory mapped to locations $3F0 thru $3F3 and controlled by CS2 which must be jumper enabled (see MEM_SEL). Due to bus speed requirements of the LCD Modules the read and write addresses of the LCD are separated. Addresses $3F0 and $3F1 are Command Write and Data Write respectfully. Addresses $3F2 and $3F3 are Command Read and Data Read respectfully. Caution must be used not to Write to the Read addresses as a bus conflict will occur.

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 LCD_PORT connector. The potentiometer, VR1 Vee ADJUST located near the RESET Switch, is used to adjust the contrast of the LCD display by varying Vee from +5 to -5 Volts.

See the file **KEYLCD12.ASM** on the software disk for an example program using this LCD connector.

+5V	**2**	**1**	GND
MA0	**4**	**3**	Vee
LCDCS	**6**	**5**	MA1
D9	**8**	**7**	D8
D11	**10**	**9**	D10
D13	**12**	**11**	D12
D15	**14**	**13**	D14

3F0/3F2 LCD Control Register
3F1/3F3 LCD Data Register

KEYPAD Connector

The Keypad connector is an eight position connector that implements HC12 Port H as a passive keypad interface. This interface is implemented as a software keyscan. Pins PH0 - PH3 are columns which are reset low to read the row condition on PH4 - PH7 which are active low. See the file **KEYLCD12.ASM** on the software disk for an example program using this keypad connector.

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
H0	H1	H2	H3	H4	H5	H6	H7

CME12-A4 OTHER FEATURES

A/D Reference

The V_{RH} and V_{RL} lines from the HC12 are connected to +5v through R2 and to ground through R1 respectively. These two surface mount resistors are on the bottom (solder) side of the CME12-A4 board. The resistors are identified on the silk screen by their reference designators. The appropriate resistor(s) need to be removed in order to apply an external reference to the V_{RH} and/or V_{RL} inputs.

Trace Jumper

The Trace Jumper is not used at this time and should not be installed.

RESET Generation

The CME12-A4 has a voltage level detector (U7) that generates an active low Reset when the supply voltage is below ~4.5VDC. The RESET Switch will apply a ground to the Reset line as long as the switch is depressed.

Power Supply

The CME12-A4 has a regulated power supply that accepts +7 to +25VDC and outputs +5VDC at up to 500ma. Normally input voltages are supplied by the provided wall-plug installed into J6. TB1 provides access to output or optional input to the +V input voltage, Ground and +5VDC system supply. Caution should be used if supplying an external +5VDC system voltage so that a voltage level of +6VDC is not exceeded or permanent damage to components on the CME12-A4 board may occur.

ABORT Switch

The Abort Switch applies a Ground to the HC12 XIRQ interrupt input. It is used by the Debug/Monitor to abort user programs executing under the Monitor.

PHASE-LOCKED LOOP

The CME12-A4 HC12 PLL loop filter components RX1, CX1 and CX2 are not installed. If PLL operation is desired, the user should review the Motorola Application Note and determine the component values to be used for the frequency of the desired operation.

TROUBLESHOOTING

The CME12-A4 board is fully tested and operational before shipping. If it fails to function properly, inspect the board for obvious physical damage first. Ensure that all socketed IC devices are properly seated in the their sockets.

The most common problems are improperly configured communications parameters and attempting to use the wrong COM port on the PC. Verify that your communications port is working by substituting a known good serial device, or by doing a loop back diagnostic. Verify that no other devices are conflicting with the port (such as a mouse, modem, etc.).

Check your hardware configuration jumpers and switches. Verify the power source. You should measure 9 volts between GND and +9V test point pads on the board near J1. If no voltage is found, verify wallplug connections to 115VAC outlet and power connector. Disconnect all external connections to the board except for COM1 to the PC and the wall plug. Follow these steps in the order given:

Troubleshooting Steps

1. Visual Inspection
2. Verify that JP1 is open and all other necessary option jumpers are installed.
3. Verify power by checking for +9 volts between GND and +9V test point pads.
4. Verify the Monitor EEPROM for proper installation (no bent pins) and proper jumper settings for the device used.
5. Re-Check the Host communications parameters.
6. Disconnect any peripheral devices including display and keypad.
7. Make sure that the RESET line is not being held low.
8. Verify presence of 16MHz sine wave on the crystal if possible.
9. Please check off these steps and any others you may have performed before calling so we can better help you.

Tips and Suggestions

Following are a number of tips, suggestions and answers to common questions that will solve most problems users have with the AX12 development system. This information is also available in the AX12 program under Troubleshooting, which will have the most complete, updated information. There also may be a newer version of the AX12 utility software available. You can download the latest version for FREE on our web page at: WWW.AXMAN.COM

AX12 Program

- If you're trying to program memory or start the HC12 Utilities, make sure JP1 is installed.
- AX12 Narrow Data Bus (8bit) must be set in the Utilities Menu Options for correct operation of the CME12-A4 board.
- If you're trying to execute a program, for example running the Debug Monitor from the Terminal window, JP1 should be open.
- If every other byte you read is wrong ($FB for example) then the Utilities is probably set to 16-bit mode and the board is 8-bit mode.
- Be certain that the data cable you're using is bi-directional and is connected securely to both the PC and the board. Also, make sure you are using the correct serial port.
- Make sure the correct power is supplied to the board. You should only use a 9 volt, 300 mA adapter or power supply. If you're using a power strip, make sure it is turned on.
- Make sure you load your code to an address space that actually exists. See the Memory Map if you're not sure.
- If you're running in a multi-tasking environment (such as windows) close all programs in the background to be certain no serial conflict occurs.
- If the Assembler menu option doesn't work properly on your system you can modify it's operation by editing the file DO_ASM.BAT in the AX12 directory. This can also be done from the Options menu.
- You can reset all AX12 configuration options to their original state by deleting the file named AX12.CFG. This file will be re-created the next time you run AX12.

Code Execution

- Make sure ALL jumpers are set correctly according to your board's configuration. Read the hardware manual section on jumpers carefully if you're not sure.
- Always remember to move JP1 to the correct position after programming memory or using Utilities.
- If you're using D-Bug12, Breakpoints are not acknowledged if you use the monitor CALL. You must use one of the GO commands instead. This will be fixed in a later version.
- If you programmed your code into EEPROM memory over the Debug Monitor and it doesn't run check the HC12 reset vector located at FFFEh - FFFFh. These 2 bytes contain the address in the micro. where execution will begin when the unit is powered on.