CME-11A

Development Board

xiom
anufacturing
™ 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 3

SUPPORT SOFTWARE 3
SOFTWARE DEVELOPMENT 4

TUTORIAL 4

CREATING SOURCE CODE 4
RUNNING THE EXAMPLE PROGRAM 5

MEMORY CONFIGURATION 6

ADDRESS DECODING 6
MEMORY SELECTION JUMPERS 7
MEMORY MAP 8

HARDWARE 9

PORTS AND CONNECTORS 9
SERIAL PORT 9
SS:KEYPAD Connector 9
LCD_PORT Connector 9
PARALLEL PORTS 10
MISCELLANEOUS OPTION JUMPERS 11
MODE SELECT 11
Programming Jumpers 11
A/D REFERENCE 11
BUFFALO TRACE 11
EXPANSION BUS 12

TROUBLESHOOTING 12

AX11 Program 13
Code Execution 14
Basic11 14
ImageCraft C 14
SMALLC Compiler 14

GETTING STARTED

The Axiom CME11 single board computer is a fully assembled, fully functional development system for the Motorola 68HC11 Micro controllers, 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. Create a directory on your PC hard drive for the utility software and copy the contents of the UTL11 disk to that directory. NOTE: it is possible to run the utility software from the floppy disk but not recommended.
2. Connect one end of the supplied 9pin serial cable to a free COM port on your PC. Connect the other end of the cable to the COM port on the CME11 board.
3. Apply power to the board by plugging in the wall plug power supply that came with the system.
4. Change to the directory containing the utility software and execute the program: **AX11.EXE**. Select the PC's COM port that you are using.
5. From the AX11 menu, select the option: "Terminal". This should bring up a blank terminal window.
6. Press then release the RESET button on the CME11 board now.
7. If everything is working properly, you should see a Buffalo Monitor prompt similar to that below in the Terminal window. Press the ENTER key and you should see a prompt which is the > character.

```
BUFFALO 3.4AX - CMD-11A User Fast Friendly Aid to Logical Operation
>_
```

8. Your board is now ready to use!

If you do not see the buffalo message prompt like that above, or if the text is garbage, see the **TROUBLESHOOTING** section at the end of this manual.

NOTE: The Buffalo monitor is loaded at the factory and the jumpers set to run the program when the board is shipped. If you did not receive the board from Axiom, or if it has been used previously, the Buffalo Monitor program may have been overwritten. You can re-program it easily from the file named BUF34E.S19. located on the UTL11 disk.

SUPPORT SOFTWARE

There are many useful programs on the UTL11 floppy disk included with the board that can make developing projects on the CME11 easier. You can also download the latest version of this disk free at any time from our web page at: **http://www.axman.com**.

The main programming interface to the CME11 board is the AX11 program. This program communicates with the board via its COM1 port and includes a Terminal window for interfacing with other programs running on the CME11, such as the Buffalo Monitor or the Basic11 interpreter. It is also useful for displaying information from your own programs that send output to the serial port.

In addition to the AX11 terminal, most communications programs will work with the CME11. Even the Terminal program in Microsoft Windows™ will do an adequate job. If using another terminal program, communications settings should be set to 9600 baud, 1 start, 1 stop, 8 data, no parity.

See the **README** file on the UTL11 disk for a complete listing of all programs and files available.

SOFTWARE DEVELOPMENT

Software development on the CME11 is performed using either the Buffalo Monitor or Basic utilities installed in U7 to create or assist in creating your program that is stored in RAM on U5. During this "debug" phase your program should locate itself just above the internal register block, for example $2000 (see the **Memory Map** section for details).

After satisfactory operation under the monitor or basic environment, your program can be written to the EEPROM in U7 by relocating it to start at $E000 then selecting “Program Code Memory” in the AX11 utility. When programming is complete your program will run automatically when the CME11 is powered on or RESET is applied.

Using the external EEPROM this way is very convenient since you can re-write the same memory chips over and over many times without any special UV erasing, which would require chip removal.

To return to Monitor or Basic development mode, run AX11 and select “Program Code Memory” then select one of the following files you wish to program:

BUF34X.S19 the Buffalo Monitor program.
BASIC11.S19 the Basic11 interpreter

If you have the 32K upgrade to the CME11 you can use U6 for external data storage or relocate your code to $8000 and use the extra memory for larger program space. If you do this, be sure to change the HC11 reset vector at $FFFE - $FFFF to point to the new beginning of your code ($8000 for example) instead of the default $E000.

TUTORIAL

This section was written to help you get started with the specifics of the CME11 software development process. Be sure to read the rest of this manual as well as the documentation on the disk if you need further information. Also, you should have a good reference manual for the 68HC11 Micro controller.

The following sections take you through the complete development cycle of a simple "hello world" program.

Creating source code

You can write source code for the CME11 board using any language that compiles to Motorola 68HC11 instructions. Included on the software disk is a free Assembler and also a freeware C compiler. Inexpensive or free compilers are also available for Basic and Forth languages. See the www.axman.com web page for more information.

You should write your source code using a standard ASCII text editor. Many powerful code editors are available or you can use the free EDIT or NOTEPAD programs that come with your computer. Once your source code is written and saved to a file, you can assemble or compile it to a Motorola S-Record (hex) format. This type of output file usually has a .MOT or .HEX or .S19 file extension and is in a format that can be read by the programming utilities to be programmed into the CME11 board.

IT IS IMPORTANT to understand your development board's use of Memory and Addressing when writing source code so you can locate your code at valid addresses. For example, when in "debug" mode, you should put your Program CODE in External RAM. In assembly language, you do this with ORG statements in your source code. Any lines following an ORG statement will begin at that ORG location, which is the first number following the word ORG, for example: ORG $2000.

You must start your DATA (or variables) in a RAM location unused by your program, for example: ORG $1040. When finished debugging, you must change these ORG statements so that your program is moved to a valid EEPROM area - somewhere after hex E000 (for the 8K board or 8000 if the U6 upgrade is installed). Do this by putting an ORG $E000 in front of your Program CODE. Data may remain where it is or be moved down to internal RAM starting at ORG $0000. You must also program the STACK register somewhere at the top of your available RAM, for example hex 1FF. Do this with this instruction as the first instruction in your program code: LDS #$01FF.

A look at the example programs on the disk can make all of this clearer. If you're using a Compiler instead of an assembler, consult the compiler documentation for methods used to locate your code and data.

Running the example program

1. If you haven't done so already, verify that the CME11 is connected and operating properly by following the steps under "GETTING STARTED".
2. At your PC's DOS command line prompt, change to the directory the UTIL11 software was copied to.
3. Execute the command: **AS11 HELLO.ASM** ↵
   This will assemble our test source code.
4. If any errors were found they would be displayed on the screen, otherwise, you should have the new file **HELLO.S19** (a Motorola hex object file) in your directory.
5. Execute the command: **AX11** ↵
   This will launch the Axiom programming interface for the CME11.
6. Select Terminal from the menu to see the terminal window.
7. Press and release the RESET button on the CME11 board. You should see the Buffalo Monitor message. Hit the return key ↵ to get the monitor prompt.
8. Type **LOAD T** ↵
   This will prepare buffalo to receive a program.
9. Press the Page Up key and when prompted for a file name, type: **HELLO.S19**
   then select [OK]. Your program will be sent to the CM11A8 board.
10. When the file finishes loading you will see the > prompt again.
    Type **CALL 2400** ↵
    This tells buffalo to execute the subroutine at address $2400, which is the start of our test program.
11. 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, a line containing the internal register status displayed by buffalo and the buffalo prompt.
12. If you do not 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 remain the same. Buffalo has many powerful features such as breakpoints, assembly/disassembly, memory dump and modify and program trace. Type HELP at the buffalo prompt for a listing of commands or consult the buffalo text file on the utility disk for more information.

When you are 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 will accomplish this:

1. Use a text editor to modify HELLO.ASM. Change the start of the program, **ROMBS**, to **$E000** which is the beginning of the EEPROM in U7.
2. Remove the comment * character before the first line after **START**. This will initialize the stack pointer which is necessary when running outside of buffalo but should not be done while running under buffalo since buffalo must handle the stack.
3. At the DOS prompt, execute the command: **AS11 HELLO.ASM** ↵
   to reassemble the program. **Note:** you can also select "Assembler" from the AX11 main menu, which calls the batch file DO_ASM.BAT which automates this process and creates a listing file.
4. Start the programming interface: **AX11** ↵
5. Select "Program Code Memory" and when prompted for a file name, type: **HELLO.S19**
   then select [OK].
6. Follow the instructions on screen. When finished programming, remove jumpers 1,2 and 10.

7. Select “Terminal” from the menu then cycle power or press RESET on the CME11 board. Your new program should start automatically.

Note: you should probably replace the return at the end of main with an endless loop or something since returning from nowhere will have unknown results when your program finishes.

```
eloop:  nop
        bra     eloop
```

To return to development mode, repeat the above steps starting with number 4, substituting BUF34X.S19 instead of HELLO.S19.

MEMORY CONFIGURATION

ADDRESS DECODING

Address decoding is accomplished using a GAL16V8 programmable logic device. Address lines A<8:15>, AS (address strobe), R/W (read/write), and E (clock) are processed to provide the memory control signals as shown below by default. Custom configurations, differing from that shown below, are also possible. Contact the factory for assistance in redefining the memory map if required.

OE Output enable to U5, U6, and U7
WR Write enable to U5 direct, and to U6 and U7 through jumpers JP6 and JP10 respectively.
M1 Chip select to U5 active from 0 to 8K (with mirrored mapping) , or 0 to 32k depending on the status of JP3. See U5 JP3 selection for more information.
M2 Chip select to U6 active for the 24k between 8000 and DFFF, with the exception of B580 through B7FF inclusive. B580 to B5FF is used by P. B600 to B7FF is taken by the HC11 internal EEPROM.
M3 Chip select to U7 active for the 8k between E000 and FFFF.
P Peripheral Access CS0 - CS7. B580 through B5FF.

All of these signals except P are active low. P is active high. Signal line M2 is also connected to the BUS_PORT expansion connector allowing M2 to work in conjunction with the CS and Address lines to implement off board, page banked memory. When M2 is used in this manner, U6 must be removed from the board.

U5 is intended to be either an 8k or a 32k RAM. U6 can accommodate RAM, EEPROM, or ROM. U7 is to be used primarily for ROM but it can also accommodate EEPROM. Jumpers JP3 through JP10 determine how U5, U6, and U7 are used. The chart on the next page defines these options.

Peripheral Access 'P' is used in conjunction with A<4:6>, and AS to generate CS<0:7>. Each of these eight chip selects controls sixteen bytes in the memory map from B580 through B5FF. CS<5:0> are brought out to the BUS_PORT where they can be used to control peripherals external to the development board. See the Memory Map for further clarification.

Memory Selection 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. All jumpers are two-pin jumpers and are installed vertically.

JP3 for **32K U5** device **Closed**

JP3 for **8K U5** device **If Open** 8K from 0000 hex to 1FFF hex and mirrored at 2000 - 3FFF, 4000 - 5FFF, and 6000 - 7FFF. This position will allow CPU internal Ram and I/O ports to segment the 8K address space.

If Closed 8K from 2000 hex to 3FFF hex and mirrored at 6000 - 7FFF. Recommended position for Buffalo Monitor and Small C operation. No segmentation occurs.

JP6 write protects **U6** memory device when open.

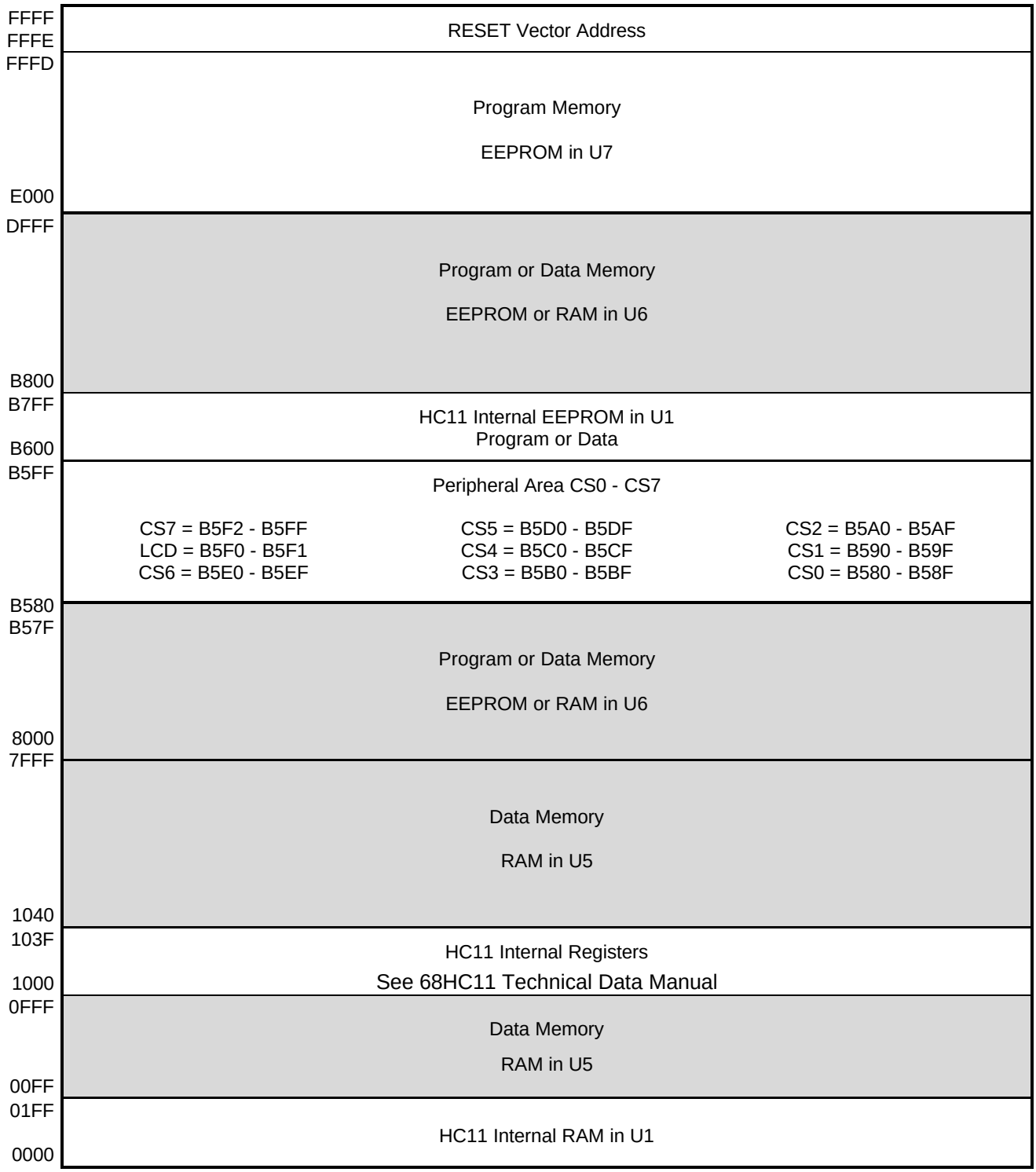
JP4	JP5	JP7	JP6	MEMORY DEVICE
open	open	open	closed	8k RAM/EEPROM.
open	open	open	open	8k EPROM.
open	closed	closed	open	32k EPROM.
closed	open	closed	closed	32k RAM/EEPROM.

JP10 write protects **U7** memory device when open.

JP8	JP9	JP10	MEMORY DEVICE
open	open	closed	8k EEPROM.
open	open	open	8k EPROM.
open	closed	open	32k EPROM.
closed	open	closed	32k EEPROM.

MEMORY MAP

The following memory map is for a 68HC11A8 as used in this development board. Other HC11 devices in the A series may also be used, as well as devices in the E and F series. These optional devices differ in the amount of internal RAM, ROM, EEPROM available and the factory default value of the CONFIG register. Consult the technical reference for the specific device you are using for additional information.



HARDWARE

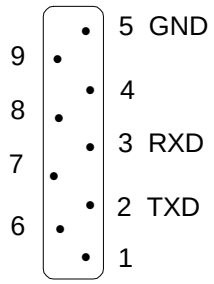
PORTS and CONNECTORS

SERIAL PORT

The onboard serial port COM1 is a simple, three wire asynchronous serial interface with hard wired Clear to Send (CTS) and Data Terminal Ready (DTR). It is driven by the HC11 internal SCI port using I/O pins PD0 and PD1. These two logic level signals are coupled through a RS232 level shifter to the COM1 connector.

COM1 is the default serial interface for the Buffalo Monitor and programming software.

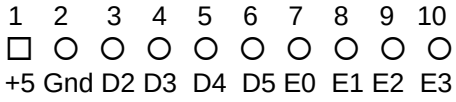
COM1 DB9S Style Connector



- Cut-Away jumpers between the 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 through type cable (supplied).

SS:KEYPAD Connector

The SS:Keypad Connector is an ten position connector that implements 4 bits of Port D and 4 bits of Port E as a simple serial or keypad interface. This interface provides connection for the SPI port on Port D as a Simple Serial interface or may be implemented as a software keyscan for a passive Keypad. Keypad Connector pinout follows:



LCD_PORT Connector

The LCD Display interface is connected to the data bus and memory mapped to locations $B5F0 through $B5F3. Addresses $B5F0 and $B5F1 are the Command and Data registers respectfully. The LCD 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.

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

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

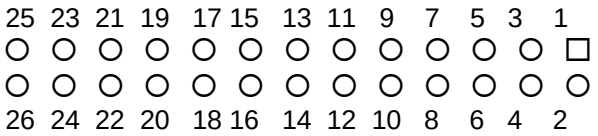
B5F0 LCD Command Register
B5F1 LCD Data Register

PARALLEL PORTS

The 68HC11 is configured for expanded/multiplexed mode. It uses Port B and Port C for address and data buss to external memory and memory mapped I/O devices. This leaves CPU Port D, Port A, and Port E to provide all other parallel I/O from the controller. CPU port lines are mixed as input only, output only, and some are input or output. All CPU port lines serve dual functions with internal CPU peripherals such as the timer subsystem and port A, the A/D converter on port E, and the SPI or SCI on port D.

MCU_PORT CONNECTOR

The CPU_PORT connector is a dual row 13 pin Berg-style connector (26 pins total) which is configured as follows:



PIN	FUNCTION	PIN	FUNCTION
1	PD0/RXD0	2	PD1/TXD0
3	PD2/MISO	4	PD3/MOSI
5	PD4/SCLK	6	PD5/SS/SEL0
7	PA0	8	PA1
9	PA2	10	PA3
11	PA4	12	PA5
13	PA6	14	PA7
15	PE7	16	PE3
17	PE6	18	PE2
19	PE5	20	PE1
21	PE4	22	PE0
23	VRL	24	VRH
25	GND	26	+5

PD0 and PD1 are used by the HC11 SCI to implement COM1. PD<2:5> are used by the HC11 SPI to implement the SIMPLE_SERIAL : KEYPAD port. These port D lines can also be used for parallel I/O, but then they will not be available for COM1 and the SIMPLE_SERIAL : KEYPAD port. Use caution when assigning port D lines to functions other than COM1 and the SIMPLE_SERIAL : KEYPAD port (Identified as SS : KEYPAD on the schematic and silk screen).

MISCELLANEOUS OPTION JUMPERS

MODE SELECT

The MODA and MODB pins of the HC11 are pulled high by two 10k resistors. This is the normal EXPANDED MODE configuration. A shunt on JP1 will take MODA to ground. A shunt on JP2 will take MODB to ground. These two jumpers allow selection of any of the following modes of operation:

JP1	**JP2**	
MODA	**MODB**	**MODE OF OPERATION**
closed	closed	Special Bootstrap
closed	open	Special Test
open	closed	Normal Single Chip
open	open	Normal Expanded (default)

Programming Jumpers

Programming the CME11 external EEPROM memory is performed by installing **JP1** and **JP2** mode jumpers and **JP10** (or JP6 if programming U6) WRITE Enable jumper. Initiate AX11 program menu item PROGRAM CODE MEMORY and follow instructions supplied by the program. After programming, remove JP10 before applying RESET or removing power from the board to guarantee retention of the new program.

A/D REFERENCE

The VRH and VRL lines from the HC11 are connected to +5v through R3 and to ground through R2 respectively. These two surface mount resistors are on the bottom (solder) side of the circuit board. The resistors are indentified on the silk screen by their reference designators and the dashed line box that surrounds them. The appropriate resistor(s) need to be removed in order to apply an external reference to the VRH and/or VRL inputs.

BUFFALO TRACE

The Buffalo Monitor Trace and Single Step functions can be enabled by installing option jumper not present on this board. Attach a wire from MCU_PORT PIN 10 (Port A3 Output Compare 5) to the XIRQ pad for nonmaskable interrupt service for the Buffalo Trace functions. Use caution when installing this jumper that no other connections are made to the XIRQ or Port A3 I/O pins of the 68HC11.

EXPANSION BUS

The BUS_PORT supports off-board devices. Power (+5V), ground, address lines, data lines, and control lines are brought out to this 40 pin connector. Pin assignments are shown on page 2 of the schematic and are listed below for convenience.

BUS_PORT			PIN	FUNCTION	PIN	FUNCTION
1	☐ ○	2	1	GND	2	D3
3	○ ○	4	3	D2	4	D4
5	○ ○	6	5	D1	6	D5
7	○ ○	8	7	D0	8	D6
9	○ ○	10	9	A0	10	D7
11	○ ○	12	11	A1	12	A2
13	○ ○	14	13	A10	14	A3
15	○ ○	16	15	OE	16	A4
17	○ ○	18	17	A11	18	A5
19	○ ○	20	19	A9	20	A6
21	○ ○	22	21	A8	22	A7
23	○ ○	24	23	A12	24	A13
25	○ ○	26	25	W/R	26	CS0
27	○ ○	28	27	CS1	28	CS2
29	○ ○	30	29	CS3	30	CS4
31	○ ○	32	31	CS5	32	IRQ
33	○ ○	34	33	+5	34	M2
35	○ ○	36	35	R/W	36	CS6
37	○ ○	38	37	E	38	CS7
39	○ ○	40	39	GND	40	RESET

TROUBLESHOOTING

The 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 IC devices in sockets are properly seated. Verify the communications setup as described on page three under GETTING STARTED.

The most common problems are improperly configured communications parameters, and attempting to use the wrong COM port (on the PC AND on the development board). Verify that your communications port is working by substituting a known good serial device, or by doing a loop back diagnostic.

Check your hardware configuration jumpers. Make sure JP1 and JP2 are NOT jumpered. 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 J1 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 below:

Hardware Troubleshooting Steps

1. Visual Inspection
2. Verify that mode jumpers (JP1 and JP2) are not installed
3. Verify power by checking for +9 volts between GND and +9V test point pads.
4. Verify the presence of +5 volts between GND test point and pin 1 of the SS : KEYPAD connector.
5. Verify U7 EEPROM for proper installation (no bent pins) and proper jumper settings for the device used.
6. Re-Check the communications parameters.
7. Disconnect any peripheral devices including display, and keyboard.
8. Make sure that the RESET line is not being held low. Check for this by measuring across S1.
9. Verify presence of 8MHz sine wave on the crystal if possible.

Please check off these steps and list any others you 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 AX11E development system. This information is also available in the online help under Troubleshooting, which will have the most complete and updated information. There also may be a newer version of the software available. You can download the latest software from the Support section of our web page at:

www.axman.com

AX11 Program

- If you're trying to execute Buffalo, Basic, or your own code from EEPROM, make sure jumpers 1 and 2 are NOT installed.
- 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 9volt, 300mA adapter or power supply. If you're using a power strip, make sure it is turned on.
- If the configuration file loads (the first 100 bytes or so), but you get a time-out error when the program section begins to download, make sure the HC11 is internally configured correctly by selecting Configure Processor from the main menu. ROMON must be set to off. During program development the only bit that should be on is EEON.
- Make sure you load your code to an address space that actually exists. The 8k boards (those with part numbers ending in -8) have EPROM's beginning at address E000h. The 32k boards can be programmed starting at 8000h.
- If you are running in a multi-tasking environment (such as Windows™) close all programs in the background to be certain no serial conflict occurs.
- If programming is slow, run the batch file PAGE.BAT in the ax11 directory then see if programming is faster. If programming doesn't work following this, return to normal operation by running the batch file BYTE.BAT.
- If the Assembler or Small C compiler menu options do not work properly on your system, you can modify their operation by editing the files DO_SC.BAT (for the C compiler) and DO_ASM.BAT (for the assembler). These are the batch files that are run when their menu items are selected. Try putting a PAUSE statement at the end of the batch files. This will halt the batch file before returning to AX11 so you can see any error messages they may be giving you.

- You can reset all AX11 configuration options to their original state by deleting the file named AX11.CFG. This file will be re-created the next time you run AX11.

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 remove jumpers 1, 2 and 10 (jumper 6 on the CMM board) after programming the code memory.
- If the board has the trace jumper, make sure it's not installed when running code from ROM or when using interrupts.
- If you programmed your code into EEPROM memory and it doesn't run, check the HC11 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. The default is E000h, which is the beginning of the 8k program address space (or the middle of the 32k program address space).

Basic11

- After programming Basic11.S19, you must remove JP1, JP2, JP10 AND JP3
- In some older versions of the Basic11 manual, the word AUTOEE was misused in place of the correct command AUTOST.
- If you want to autostart your Basic code don't forget to add a RETURN command somewhere in your programs main loop to get back to the basic interpreter. Otherwise, you'll have to remove the external EEPROM to get BASIC11 to autostart again.
- If you're getting errors or your program isn't uploading to basic11 properly from the AX11 Terminal program, try adding pacing delays. 10 or 20 (ms) should be enough.

ImageCraft C

- Your make or build should create a .MAP file. At the top of this file should be a label __START. This is where you should CALL or GO to when debugging in buffalo. Do not use your main label.

SMALLC Compiler

- If you're programming to ROM, delete the first line of the S19 record generated, since SMALL.C adds data there.
- Make sure you put the CODE, DATA, and STACK at the correct locations per your memory configuration. DATA may need to be at 0000 if you're burning a ROM, and STACK should have plenty of room.
- Don't forget to use the -R option if you're compiling for ROM.
- If Internal Registers have been re-mapped (default at 0x1000) be sure to change the #define REG_BASE accordingly.
- If you're having trouble with the shift operations (>> or <<) this is probably because the small c compiler uses a rotate thru carry instruction here. Try using inline assembly code to accomplish the shift.
- You may have trouble returning any values larger than 1 byte from functions (i.e. char functions). You will have to use global integer values instead.
- Small C does not support structures or unions.
- Small C is a free "un-supported" public domain compiler. If you're doing more than just experimenting consider purchasing a commercial quality C compiler. ImageCraft sells a good compiler for around $50, call them or Axiom Mfg. to order.