LIMITED WARRANTY

EVBplus.com provides the warranty on its EVBplus2 board against defects in materials and workmanship for a period of one year from the purchase date.

The product returned postpaid to EVBplus.com during this period will, at our option, be either replaced or repaired and returned to the sender free of charge.

If the defective unit has been modified without EVBplus.com's consent or it is the result of misuse, abuse or misapplication, EVBplus.com has no obligation to repair or replace the defective unit. If any charge to you is involved, the replacement unit will be sent by C.O.D. This warranty does not cover damage due to shipping, accident, unauthorized repair, abuse, or misuse.

EVBplus.com does not assume any responsibility for the use of this product. No EVBplus.com products sold under this warranty may be used in life support applications or situations wherein malfunction or failure of the product or its software could cause bodily injury or property damage. EVBplus.com does not assume any liability arising out of the use or application of this product; neither does it convey any license under its patent rights or the rights of others.

The PC driver software is provided "as is" without warranty of any kind, either expressed or implied, including warranties of merchantability and fitness for a particular purpose. Should the software prove defective, you (and not EVBplus.com) assume the entire cost of all necessary servicing, repair or correction. In no event will EVBplus.com be liable to you for any damages, including any lost profits, lost savings or other incidental or consequential damages arising out of the use or inability to use such product even if EVBplus.com has been advised of the possibility of such damages or for any claim by any other party.

Some states do not allow the limitation or exclusion of liability for incidental or consequential damages, some states do not allow the exclusion of implied warranties, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have rights which vary from state to state.

You acknowledge that you have read this limited warranty, understand it and agree to be bound by its terms and conditions. You further agree that it is the complete and exclusive statement between us and supersedes any proposal or prior agreement, oral or written, and any other communications between us relating to the subject matter of this agreement.

Return policy:

Before returning a defective unit, you must obtain a Return Merchandise Authorization number. The RMA number must be prominently displayed on outside of the returned package, otherwise the package will not be accepted.

EVBplus2 68HC11 Development Board

User Manual

Version 2.20 for Ep2711E9 Rev. C board

Table of Contents

INTRODUCTION

The Ep2711E9, a low cost and high performance development board, provides real time emulation for the Motorola 68HC11microcontroller A and E families. It offers all useful features of the Motorola EVB board with the Buffalo monitor and adds numerous enhancements at an extremely low cost. It combines a complete 68HC11 development system, an advanced trainer, a reliable 711E9 programmer and a versatile SBC into one package. For engineers, it's a WICE in-circuit emulator development system and Motorola EVB, EVM, EVS replacement, a convenient prototype platform, and a low cost single board computer. For students, it's a user-friendly microcontroller trainer. It is as powerful as a high priced real-time in-circuit emulator, but it's affordable as a low cost single board computer.

Ep2711E9 includes user-friendly IDE software which runs under Windows® 95, 98, 2000 and XP. It offers fast file transfer, single-stepping, breakpoints, data watch for memory and registers, symbolic debugging compatibility with most assemblers and compilers, and user program termination with the <Esc> key.

Our exclusive ***phantom monitor***™ technology preserves all interrupt vectors including RESET and All on-chip RAM ($00-$1FF), EEPROM, and 30K external emulation RAM ($8800-$FFFF) available for user applications - there is no pre-empted chip memory.

The hardware includes a prototype area, 16 extra I/O lines form ports F and G, one logic probe, on-board 16X2 LCD display with backlight, 4x4 keypad connector, 8 sensor I/O port, SPI port, speaker, 4-digit LED display, potentiometer, 8 LED status indicators for port B, one 8-position DIP switch connected to port C and three pushbutton switches, dual UART, RS485 interface, IR transceiver with on-board 38 KHz OSC, on-board 12 programming voltage supply and 60-pin EVB/EVBU-compatible connector.

SYSTEM DESCRIPTIONS

The basic operation of the Ep2711E9 is divided into two modes: the MONITOR mode and the RUN mode. In the MONITOR mode, the user can exercise all commands as described later in this manual, and all user interrupts are temporarily suspended in this mode. In the RUN mode, it will run a user program in full speed, and all interrupts are enabled for the user program.

As the RS-232C handshake lines are not used, the handshaking is done by XON and XOFF.

The board will use 9 bytes on the stack during debugging. You should allocate enough memory locations for the stack to cover these 9 bytes. The resident MCU is running in expanded mode even it emulates single chip mode. The port B and port C are used for address and data buses, and they are not available as I/O ports during debugging session in the expanded mode, but they are replaced by the 68HC24, the port replacement unit.

STANDARD FEATURES

User interface

Memory window, monitors user programmable memory locations

Register window, monitors all CPU registers

Stack window, monitors stack locations

Breakpoint window, monitors breakpoints

Interactive command window with novice and expert modes for command entry

On-line help menu and function key selection

Displays memory in hexadecimal, ASCII, Motorola S-record, Intel Hex or Tektronix Hex formats

Echo debugging session to log file and/or printer

DOS shell facility

Displays internal CPU registers and control bit assignment map using standard Motorola labels

Math function for verifying program results

Memory and register operation

Memory manipulation

Display memory block

Examine and modify memory

Compare memory block

Search memory block for pattern

Move memory block

Fill memory block with pattern

Register manipulation

Examine and modify CPU registers

Breakpoints

Set or clear breakpoint address

Assembler and disassembler

Symbolic disassembler and single line symbolic assembler

Automatically disassembles 16 lines of code during assembling

Scrolls display back and forth by page or line during disassembling

Execution Control

Execute program

Single step over all instructions, including the RTI instruction

Trace execution

File management

File transfer

Download file from host to memory

Upload memory to host file

Automatically download files during startup

Upload/download of files in Motorola S record,

Intel Hex and Tektronix Hex formats

File formats

Motorola S-record, Intel Hex, Tektronix Hex

Symbol formats

Motorola, 2500AD, Microtek, Zax, Wintek, Introl, Avocet, Archimedes, Whitesmiths, IAR, P&E systems

System

Configure INIT, TMSK2, OPTION, BPROT registers

EEPROM erasing and programming

CONFIG register programming in test mode

Scope or logic analyzer trigger output 40 pin logic analyzer connector

SYMBOLIC DEBUGGER

Predefined names

All internal CPU registers can be referenced by their name, such as porta PORTA, portb, PORTB, ddrd, DDRD, etc.

Symbols on the command entry

For a command in which an absolute number can be replaced by a symbol, the symbol must start with a '\', such as \BEGIN, \loop. The maximum number of characters in a symbol is 32, and symbol is CASE SENSITIVE. Although symbols can be 32 character long, but the disassembler can only display 10 characters maximum for a symbol. In your source code, it would be better make symbols shorter.

Symbol file format

Symbolic debugger reads symbol files which are generated by the Motorola cross assembler (a public domain program), the 2500AD, Microtek, Zax assemblers, other popular assemblers and C compilers, including the Wintek, Introl, Avocet, Archimedes, Whitesmiths, and IAR C compilers.

When the Archimedes C compiler is used, you should specify the symbol format by the -Fmpds-m in the xcl file. If you have an assembler or C compiler whose symbol format is not supported by the Ep2711E9 board, we will modify our software to add your symbol format.

COMMAND REFERENCE

Command format

The command line format is as follows:

Ep6811->> [Command] [Parameters]

Where: **Ep6811->>** is the Ep2711E9 prompt.
[Command] is the command mnemonic, may be entered in an uppercase or a lowercase letter. Only the first character of the command is needs to be typed.
[Parameters] are memory addresses or symbols. **ALL NUMBERS ARE HEXADECIMAL.** All hex numbers may be substituted by symbols starting with the backslash '\'.

Comment:

The fields can be separated by a space or the CR, but a command must be terminated by the CR only.

Errors may be corrected by backspacing or by the ESC key to abort the command. The ESC character will abort the command any time.

Only a **valid key** will be accepted on the command line and any invalid keys will NOT be echoed to the CRT display.

Command reference

The command reference is listed in alphabetical order.

Assemble

Syntax:

Assemble [address]

Description:

[address] is the starting address for the assembler operation. Each source line is converted into the proper machine language code and is stored in memory, overwriting previous data on a line-by-line basis at the time of entry.

Caution:

When a new source line is assembled, the assembler overwrites what was previously in the memory. If the assembler detects an error in the new source line, it will output an error message, then re-open the same address location, and the contents of the memory will remain unchanged.

Any attempt to assemble at locations of the monitor will be ignored.

Syntax rules:

1. All numerical values are assumed to be hexadecimal, therefore, no base designators are allowed.
2. Operands must be separated by the space character.

Addressing modes are designated as follows:

1. Immediate addressing is designated by preceding the address with the # sign.
2. Indexed addressing is designated by the ",X". The comma must be preceded by an offset byte (e.g., LDAA 0,X).
3. Direct and extended addressing are specified by the value of the address operand, a value of less than 256 specifying direct addressing, otherwise the extended addressing is implied.
4. Relative offsets for branch instructions are computed by the assembler. Therefore the valid operand for any branch instructions is the branching address, not the relative offset.

Subcommands:

ASM	[address]	starts assemble at a new location.
FCB	[byte].....	allocates a hex byte for each location.
FCC	[byte].....	allocates an ASCII byte for each location.
BYTE	[byte].....	as same as FCB
DB	[byte].....	as same as FCB
ASCII	[byte].....	as same as FCC

The assembler's subcommands are available only when an assemble command is executed. For the FCB or FCC subcommand, a maximum of 32 bytes may be entered on a command line.

Example:

```
Assemble F000       ; This command starts assemble at the address location $F000.
ldaa    #10         ; load Accumulator A with hex #$10, not decimal #10
ldab    2,x         ; load Accumulator B with the value of the address that is 2 plus the value of the
                    ; X register
staa    5           ; store the value of the Accumulator A in the direct RAM location 5
staa    1200        ; store the value of the Accumulator A in the extended RAM location $1200
bra     f000        ; branch to the location $f000
```

Breakpoint

Syntax:

Breakpoint [address]

Description:

The breakpoint is set at the location specified by the address. The breakpoint can only be set at instruction addresses. A maximum of eight breakpoints may be set.

All breakpoints are software breakpoints.

Comment:

During initial power up all breakpoints are automatically cleared.

Example:

Breakpoint F010
This command sets a breakpoint at the address $F010.

Compare

Syntax:

Compare [address1] [address2] [address3]

Description:

[address1] is the starting address of the 1st memory block.

[address2] is the ending address of the 1st memory block.

[address3] is the starting address of the 2nd memory block.

The Compare command allows the user to compare a memory block with another. If the address3 is not specified, the block of memory residing from address1 to address2 will be compared with the block memory residing from address1-1 to address2-1.

Example:

Compare F000 F03F E000

This command compares 64 bytes of the memory block1 from $F000 to $F03F with the memory block2 from $E000 to $E03F.

Dump

Syntax:

Dump [address1] [address2]

Description:

[address1] is the starting display address. It defaults to the address which equals the ending address from the previous memory dump plus 1.

[address2] is the ending display address. It defaults to the address which equals the address1 plus

$3F(in hex).

The Dump command allows the user to display a block of memory beginning at the address1 and continuing to the address2. If the address2 is not entered before the Carriage Return is entered, 64 bytes of memory are displayed beginning at the address1.

Example:

Dump F000 F0FF
This command displays a block of memory beginning at $F000 and continuing to $F0FF.

Enter

Syntax:

Enter [address]

Description:

[address] is the memory location at which to start examining and/or modifying.

The Enter command allows the user to examine and/or modify contents in memory at specified locations in an interactive manner. Once entered, the enter command has several subcommands that allow modification and verification of the memory contents.

Following are the subcommands that are recognized by this command:

```
<Enter>     ; examine/modify the next location.
<->         ; examine/modify the previous location.
<ESC>       ; exit from the Enter mode.
```

Caution:

Any attempt to modify at address locations of the monitor will be ignored.

Example:

Enter F000

```
F000 00/      ; display data at the address $F000.
F000 00/AA    ; change data at $F000 from 00 to AA.
F001 11/      ; display data at the next address $F001.
```

Fill

Syntax:

Fill [address1] [address2] [pattern]

Description:

[address1] is the lower limit of the memory block. It defaults to the address which equals the ending address from the previous DUMP command plus 1.

[address2] is the upper limit of the memory block. It defaults to the address which equals the address1 plus $3F (in hex).

[pattern] is the data pattern in hexadecimal values. It defaults to 00 by entering the Carriage Return.

The Fill command allows the user to repeat a specific pattern (up to 16 bytes) throughout a determined memory range. The data bytes must be separated by the space character. The Carriage Return will end this command.

Caution:

Any attempt to fill memory locations of the monitor will be ignored.

Example:

Fill F000 FFFF 00

This command fills locations $F000 through $FFFF with 00's.

Go

Syntax:

Go [address]

Description:

[address] is the starting address where the program execution begins.
The G command allows the user to initiate the program execution (free run in real-time). The user may optionally specify a starting address where execution is to begin. The program execution starts at the current program counter address location, unless a starting address is specified. The program execution continues until a breakpoint is encountered, or the ESC key, or the reset switch is depressed.

Example:

Go F000

This command starts execution from the address $F000.

Init

Syntax:

Init [value1] [value2] [value3] [value4]

Description:

[value1] is the new value for the INIT register.
[value2] is the new value for the TMSK2 register.
[value3] is the new value for the OPTION register.
[value4] is the new value for the BPROT register.

The user will be prompted to enter all 4 values, press the CR to skip the modification or the ESC key to cancel the modification. The INIT register, the BPROT register, the bit 0,1,4,5 of the OPTION register, and the bit 0,1 of the TMSK2 register can be modified only within first 64 E cycles after a system reset.

The Init command will automatically generate a reset signal on the 68HC11 in order to force the 68HC11 to take new values for these 4 control registers.

Caution:

Any other commands, such as Enter, Fill or Move, will not reset the 68HC11, and will not have any effect on the INIT register, the BPROT register, the bit 0,1 of the TMSK2 register and bit 0,1,4,5 of the OPTION register.

Before executing a user program the value of the above four registers must be set correctly. If a user program modifies these registers in its initialization routine, the user must use the I command to modify these registers with the same values.

Kill breakpoint

Syntax:

Kill breakpoint [address]

Description:

The breakpoint is cleared at the location specified by the address. If the address is not specified, all breakpoints will be cleared.

Example:

Kill breakpoint F010

This command clears the breakpoint address location $F010.

Load

Syntax:

Load [filename] [offset]

Description:

The Load command will prompt the user to enter the type of the file to be downloaded. This command can download the user file into the 512 byte on-chip EEPROM ($B600-$B7FF), or the 8K on-board EEPROM ($4000-$5FFF), or the 30K on-board RAM ($8800-$FFFF). It also can download the symbol file, or the system parameter file. The parameter file includes the EEPROM programming enable/disable flag, the INIT, TMSK2,OPTION and BPROT registers, the breakpoint addresses, the memory display addresses

If the type of the file is a user file which is in the Motorola S record, or the Intel hex, or the Tektronix Hex format. The user will be prompted with the following parameters:

[filename] is the filename to be download from the host PC.
[offset] is the offset to be added with the download addresses.

If the type of the file is the symbol file or the parameter file the user would be prompted to enter file name.
The Load command downloads user files in the Motorola S record, or Intel hex or Tektronix hex format from the PC into the Ep2711E9 board. Before downloading a user file, the user will be prompted to input an offset value which will be added to the download addresses of the user file to generate the absolute addresses for the Ep2711E9 board. If the offset is not needed, just press the Carriage Return. By using this offset feature, a file can be downloaded into different memory blocks.

Caution:

Always download the symbol file first, so the symbols in a parameter file will have their values.
Any attempt to download data from a file into the address of the monitor will be ignored.

Example:

Load test.s19 8000

This load command downloads the file named test.s19 from the host PC into the Ep2711E9 board. It will add $8000 with all memory addresses of the file. For instance, the data at the address $1000 of the file will be downloaded into the location $9000 instead of $1000, because every download address is added with $8000.

Move

Syntax:

Move [address1] [address2] [address3]

Description:

[address1] is the starting address of the source memory block.
[address2] is the ending address of the source memory block.
[address3] is the starting address of the destination memory block.

The Move command moves the contents of the source memory block to the destination memory block. If the destination memory block is not specified, all data residing from the address1 to the address2 will be moved up one location.

Caution:

Any attempt to move data to the address locations of the monitor will be ignored.

Example:

Move F000 F03F E000

This move command moves 64 bytes of data in the memory block between $F000 and $F03F to the memory block between $E000 and $E03F.

Next step (single-step)

Syntax:

Next step

Description:

The N command allows the user to monitor program execution on an instruction-by-instruction basis. It will execute one instruction (including the RTI instruction) whose location is pointed to by the program counter. After a single-step, the PC will display all CPU register values and the next instruction to be executed.

Program memory watch

Syntax:

Program

Description:

The P command allows the user to select 3 blocks of 2 contiguous memory locations and 3 blocks of 8 contiguous memory locations for memory watching purpose. These locations will always be updated after using the Assemble, Enter, Fill, Move, Go, or Next Step commands.

All six starting addresses can be substituted by symbols. For example, \DATA is 0005 if the value of the symbol \DATA is 0005, \message is E100 if the value of the symbol \message is E100.

Register modify

Syntax:

Register modify

Description:

The R command allows the user to display and modify:
PC - program counter,
CC - condition code register,
B - Accumulator B,
A - Accumulator A,
X - Index register X,
Y - Index register Y,
SP - stack pointer.
After modifying the value of a register, press the CR to complete modification and press the ESC key to exit from the command.

Caution:

The stack pointer should be modified with a great care. It should only be modified when nothing was pushed onto the stack, otherwise the return address and flag register that were pushed onto the old stack must be manually moved to the new stack area to prevent a crash.

Search

Syntax:

Search [address1] [address2] [hex data pattern]

Description:

[address1] is the starting address of the memory block to be searched.

[address2] is the ending address of the memory block to be searched. It defaults at the address which equals the address1 plus $3F(in hex).

[pattern] is the data pattern to be searched for.

The Search command allows the user to search for a specific pattern (up to 16 bytes) throughout a specified memory range. The data bytes must be separated by space characters. The Carriage Return will end this command.

Example:

Search F000 FFFF 01 02 03 04 05 06 07 08

This command will search for the pattern of 8 hex numbers, which are 01 02 03 04 05 06 07 08, in the memory block between $F000 and $FFFF.

Trace

Syntax:

Trace [number of steps]

Description:

The Trace command allows the user to trace program execution for a specified number of steps on an instruction-by-instruction basis. During the trace, the PC will display all register values and the next instruction to be executed after each step.

Unassemble

Syntax:

Unassemble [address]

Description:

[address] is the starting address for disassemble operation. If no address is specified, the starting address is the one that followed the last instruction unassembled by the previous U command.

The U command will disassemble 16 lines of instructions. All valid op codes are converted to the assembly language mnemonics, and all invalid op codes are replaced by the FCB's. The PgUp, PgDn , , Home keys can be used to scroll display back and forth.

Write

Syntax:

Write [filename] [address1] [address2] [M/I/T]

Description:

The Write command will prompt the user to enter the type of the file to be saved. This command can upload the user file, or the symbol file, or the system parameter file. The parameter file includes the EEPROM programming enable/disable flag, the INIT, TMSK2, OPTION and BPROT registers, the breakpoint addresses, and the memory display addresses.

If the type of the file is the user file which is in the Motorola S record, or the Intel hex, or the Tektronix Hex format, the user will be prompted with the following parameters:

[filename] is the filename to be saved in the host PC.

[address1] is the starting address of the memory block to be saved.

[address2] is the ending address of the memory block to be saved.

[M/I/T] is the file format in which the user file is saved. Enter M for the Motorola S record, I for the Intel Hex format or T for the Tektronix Hex format.

If the type of the file is the symbol file or the parameter file, the user would be prompted to enter the file name only

Command summary

Command	Syntax	Function
Assemble	A	Single line assembler
Breakpoint	B	Set breakpoint
Compare	C	Compare memory block
Dump	D	Display memory block

Enter	E	Examine and modify memory
Fill	F	Fill memory with user pattern
Go	G	Execute program
Init	I	Configure INIT, TMSK2, OPTION & BPROT registers
Kill	K	Clear breakpoint
Load	L	Download file from host
Move	M	Move memory block
Next step	N	Single step
Program	P	Program memory watch
Register modify	R	Examine and modify CPU registers
Search	S	Search memory range for user pattern
Trace	T	Trace execution
Unassemble	U	Disassemble
Write	W	Upload memory block to host file

FUNCTION KEY ASSIGNMENT

F1: A brief on-line help.

F2: Emulation memory map.

F3: Not used.

F4: Dump memory contents in the Motorola S record, or the Intel Hex format or the Tektronix Hex format.

F5: This key allows the user to echo CRT output to a printer and/or a log file on the disk.

F6: This key displays all 68HC11 control registers and control bit assignments.

F7: This function evaluates a numerical operand, or performs arithmetic and logical operations on a simple expression, giving the result in Hex, Decimal, Octal, Binary, valid ASCII character, one's complement and valid negative number. The arithmetic operations are addition (+), subtraction (-), multiplication (*), and division (/). The logical operations are AND (&), OR (|), Exclusive OR (^), left shift (<) and right shift (>). The default notation is HEX, and a decimal, octal, or binary number must be followed by the letter T, O, and Q. The result in the first line is HEX, Decimal, Octal, Binary and ASCII character if the value of the result is between 0x20 to 0x7F. The result in the second line is one's complement, negative number if the value of the result is between 0x80 to 0xFF in 8 bit format, 0x8000 to 0xFFFF in 16 bit format, and 0x80000000 to 0xFFFFFFFF in 32 bit format.

F8: This key allows the user to erase or program the 512 byte on-chip EEPROM.

F9: This key allows the user to display and edit the symbol table. Use the Load command to read symbol files which are generated by the Motorola cross assembler (a public domain), 2500 AD, Microtek, Zax assemblers, Wintek, Introl, Avocet, Archimedes, Whitesmiths, IAR C compilers or Wytec format directly. No conversion is needed.

F10: This key allows the user to do:

1. Terminate the emulation and exit from the debugger, or
2. Re-start the emulation without terminating. It is often called warm boot and is useful to initialize the CRT screen. For instance, when the PC reads a file from a floppy disk drive for downloading into the Ep2711E9 board, the PC will print out an error message that will garble the monitor screen, if the door of the drive is left open. This key will restore the screen. It also can be used to automatically download files.

ALT+F1:

This key combination toggles between the novice and the expert modes. In the novice mode the user will be prompted to enter each parameter on the command line.

QUESTIONS & ANSWERS

This chapter contains the most commonly asked questions about the Ep2711E9 board. They are divided into three sections.

1. Hardware
2. Miscellaneous questions

Hardware

Visually check all solder joints and also if there is any solder bridge. We fully tested every board before shipping it out, but still cannot guarantee that we wouldn't make a mistake.

Q. When I turn on the Ep2711E9 board, how come the LED does not flash?

A. After the initial power up, the RESET LED should blink 4 times in Wytec monitor mode, but twice in BUFFALO monitor mode, otherwise check the VCC. It should have a reading of about 5V DC. If the 5V DC is not present, check if the polarity of DC output of the AC adapter is correct. The center is positive on the DC plug.

The board is running in expanded mode, even it's emulating the 711E9 OTP part. You must also remove the MODEA and MODEB jumpers for the expanded mode if they were installed for programming the 711E9 OTP chip in bootstrap mode.

If the problem still exists, try starting it up in the test mode by pressing and holding the PA0 data switch down while resetting the board. If the RESET LED does not flash, then the 68HC11 chip is bad and you have to replace it. If the RESET LED flashes, the chip is good and most likely the COP function is enabled. It will work after re-programming the CONFIG register in the test mode to disable the COP function.

If you have a watchdog circuit with the reset signal, you should disable it during debugging, only test your watchdog function after your code is completely debugged. The LED may not flash when the COP is enabled.

Finally check the EPROM U2 DIP socket for any loose contact.

Q. The RESET LED flashes during power up or whenever the reset switch is depressed but why does the Ep2711E9 board not communicate with the host PC?

A. If the LED blinks and there is no response from the Ep2711E9 board, check the RS-232C cable connection and if the COM port number is configured correctly. Also make sure that you only open the debug window once. If the first debugger is running, you cannot invoke second debugger before closing the first one.

Q. My program works with another SBC but why does it not work when I use the Ep2711E9?

A. The values of the INIT register (shown on the top of the screen) and the TMSK2, OPTION, and BPROT registers must be set correctly by the Init command before running your program. Because these registers can only be modified in normal modes within the first 64 E cycles, any modification to these registers in the beginning of your program will not change their values when your target program is executed by the GO command, because the 64 E cycles period was already passed. The Init command allows you to modify these registers to the correct values before running your target program, because the I command will automatically reset the 68HC11E1, so the modification can be done.

If you don't relocate the control register, the value of the INIT register should be 01. Always watch this value on the top of the screen. If this value is corrupted by running a bad program, you have to re-program it to 01 by the Init command, or power cycle the board. During the power up, its value is always = 01.

Q. My program works with the Ep2711E9 board but it does not work in another SBC, why?

A. The stack point in your program may not be set in the beginning of your program. The Ep2711E9 sets the stack point during power up, you should do the same thing in your program. If your resident 68HC11 microcontroller on the Ep2711E9 board is a 68HC11E1, but the 68HC11 on your target is a 68HC11A1, you should be careful to use the PA3 of the PORTA, it is an output line in the A1 controller but a bi-directional line in the E1 controller. You have to program the bit 3 of the PACTL register if you want the PA3 to be an output line. It defaults as an input line during power up.

Q. After executing the program by the GO command why can't the ESC key stop the Ep2711E9 board?
A. In most cases the ESC key can stop the user program, if this is not the case, there could be many reasons for this problem. Since you cannot use the ESC key to stop the 68HC11, the only thing you can do is to reset the Ep2711E9 board by depressing and releasing the reset button. The reset button of the Ep2711E9 board should be pressed whenever the Ep2711E9 board does not seem to work properly, such as a system crash after running a bad program. Once it is pressed, it will reset the program counter to the reset vector fetched from locations $FFFE and $FFFF, the stack pointer to $00FF, and update the INIT register.

Check if the jumper on the PRG/RUN header is on the RUN position. If the jumper is set at the PRG position, then you cannot stop program and also the single step function will not work.

Carefully check the user program to determine if there is any attempt to write data into the memory locations at $6800-$x87FF. These locations are used by monitor firmware. Any memory write to these locations must be avoided in the user program.

Q. I want to modify the CONFIG register, how do I enter the test mode?
A. The CONFIG register can only be modified in the test mode. In order to enter the test mode, press and hold the PA0 switch, then press and release the RESET button and it will start the test mode. By checking the status line on the top of the screen you can tell if you are in the test mode. The Ep2711E9 board cannot emulate the user program in the test mode and the test mode is only used for modifying some registers (such as the CONFIG register) that cannot be modified in both single chip mode and expanded mode.
The status line on the top of the screen indicates the operating mode. The 'EXP' stands for the expanded mode and the 'TST' stands for the test mode. The status line also displays the value of the INIT register, which indicates the locations of the on-chip direct RAM and the 64 control registers.
In the test mode, press the F8 key and follow the instructions to modify the CONFIG register.

Q. Can I modify the CONFIG register in my program after a system reset?
A. The CONFIG register can only be modified in the test mode. This is a safety feature of the 68HC11. Your program cannot modify it unless your target hardware is set for the test mode.

Miscellaneous questions

Q. What is the parameter file, and how do I create this file?

A. The parameter file includes the EEPROM programming enable/disable flag, the INIT, TMSK2, OPTION and BPROT registers, the breakpoint addresses, the memory display addresses. These parameters can be saved for a particular user program, they should be loaded before running the user program so the program will be executed in a consistent manner. You do not have to make this file by yourself. You can use the W command to save all parameters into a parameter file.

Q. Why does the disassembler display wrong symbols for some instructions, especially the instructions with 8 bit values?

A. Many cross assemblers do not have any attribute bytes for their symbols. When too many symbols have the same value, the disassembler cannot display the correct symbol for that value. The disassembler will display first two labels it found. Occasionally, when too many symbols have the same value, the disassembler may display a wrong symbol for the value.