EVBplus2 68HC11 Development Board

Getting Started Manual

Version 2.27 for Ep2711E9 Rev. C board

Table of Contents

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 easy-to-use and 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.

Because many students are taught with the Motorola BUFFALO, we installed the BUFFALO monitor on the U3 and the board can be booted from the BUFFALO monitor, so students can use the board immediately before leaning how to use the board under Wytec phantom monitor.

The hardware includes a prototype area, 16 extra I/O lines in port F and G, one logic probe, on-board 16X2 LCD display with backlight, 4x4 keypad connector, 8 sensor input 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, 3 pushbutton switches, dual UARTs, RS485 interface, IR transceiver with on-board 38 KHz OSC, on-board 12V programming voltage supply and 60-pin EVB/EVBU-compatible male and female connectors.

The package also includes a 7.5V 300mA wall plug-in power supply and a 6-foot DB9 cable.

The specification of the AC adapter is:

DC input: 110V
DC output: 7.5V to 9V
Current rating: 300 mA to 1A
Type of plug: 2.1mm female barrier plug, center positive

The AC adapter is only available to the countries that use 110V.

If more power is needed in a Robot or other applications, the user should upgrade the AC adapter to 9V 500mA or 9V 1A, otherwise the board could keep resetting itself when the VCC drops below 4.6V.

People often use different terminology. In our product menus, the "Download" means to transfer a file from PC to development board, while the "Upload" means to transfer a file from development board to PC.

Through out the manual, the **left click** means that you click the left button on the mouse, the **right click** means that you click the right button on the mouse.

Un-install old Ep2IDE software:

If you have already installed the old version of Ep2IDE before 05/01/03 on your hard drive, you can un-install it, because it will no longer be needed. The procedure to un-install is:

1. Click the Start Button → Setting → Control Panel → Add/Remove Programs, and then find the Ep2IDE to remove.
2. Rename or delete the folder Ep2IDE

Install new Ep2IDE software from CD:

Make sure to rename the current folder c:\Ep2IDE before installing new software by running the "**SETUP.BAT**" on the CD. The installation will **OVERWRITE** the current folder if you don't rename it.

After software is successfully installed, you can make a shortcut of the AsmIDE.exe. If you do not want to make a shortcut, you have to click the c:\Ep2IDE\AsmIDE.exe every time you invoke the AsmIDE.exe.

It's very important to make a shortcut so that its target location of the shortcut is C:\Ep2IDE, not c:\Windows\desktop or other locations. At first, right click the Start button, then left click the Explorer, left click on C:\Ep2IDE, right click on AsmIDE.exe (an application program), left click the "Send to", and finally left click the "Desktop" (do not click the "COPY"). It will create an icon of the "shortcut to AsmIDE" on the desktop. You can double check the target location by right clicking on the icon, then left clicking on the properties and you should see that the target location is C:\Ep2IDE. If you want to make a shortcut for the AsmIDE on the Desktop, this is the right way to do. If you don't follow this method, your program may have a problem to run. Never drag the AsmIDE.exe to the desktop folder.

The default setting of the AsmIDE for the Ep2711 board is created in a text file named c:\Ep2IDE\AsmIDE.ini. In the future if you get lost with all the changes, you always can copy this file into c:\Ep2IDE.

GETTING STARTED with BUFFALO Monitor

The Memory Map with the BUFFALO monitor:

$0000-$00FF	On-chip 256 byte RAM for the 68HC11A1
$0000-$01FF	On-chip 512 byte RAM for the 68HC11E1
$002D-$00FF	On-chip RAM used by BUFFALO monitor
$1000-$103F	On-chip 64 control registers
$2000-$3FFF	External device, /CS on the pin40 of J1
$4000-$5FFF	8K EEPROM of U3
$6000-$61FF	not used
$6200-$63FF	65C22
$6201	PORTF
$6203	DDRF
$6200	PORTG
$6202	DDRG
$6400-$65FF	68B50
$6400	UCTRL, USTAT
$6401	UART
$6600-$67FF	not used
$6800-$7FFF	not used
$8000-$DFFF	RAM for user code or data
$E000-$FFFF	BUFFALO monitor firmware in U2

Any attempt to write to the locations ($6000-$87FF) will have unpredictable results and must be avoided in the user program.

Because most text books are written for the BUFFALO monitor, the Ep2711E9 board now comes with both the BUFFALO monitor and the Wytec Phantom Monitor installed in the U2. If you have worked with the BUFFALO monitor, you can use the board right away. In fact, if you use the BUFFALO monitor mode with this board, the

board becomes an ordinary 68HC11 EVB board, just like many other EVBs on the market today, except it offers more on-board peripherals.

Before testing the board with the BUFFALO monitor, place a jumper on the middle position (labeled with 'BM' which stands for BUFFALO Monitor) of the J15 and another jumper on the left position (labeled with 'TRACE') of the J22. To test the board, follow the steps 1 through 6 below:

Step 1.

Plug the AC adapter into a wall outlet, and plug the DC plug at the other end into the DC jack on the lower right corner of the Ep2711E9 board. During the power up, the reset LED should blink twice and all other LEDs must be off. If not, turn over to the Questions & Answers section of the user manual.

Step 2.

Plug the DB9 male end of the cable into the DB9 connector P2 on the **upper right** corner of the Ep2711E9 board, and plug the DB9 female end of the cable to the COM1 or COM2 port on the PC. The DB9 connector P3 on the lower right side of the board is the 68HC11 SCI port that can be used by user's application program.

Step 3.

Press the reset button on the Ep2711E9 board, and the reset LED, which is located above the switch, should blink twice. If not, turn over to the Questions & Answers section of the user manual.

Step 4.

To invoke the AsmIDE, right click the Start button, then left click the Explorer, left click on C:\Ep2IDE, left click on AsmIDE.exe. If you have created a shortcut icon on the desktop, just double click the AsmIDE icon on the desktop.

The screen is divided into two windows. The top window is for editing your source code and the bottom window is shared by the **message window** and the **terminal window**.

Step 5:

You only need to use three commands from the AsmIDE for your 68HC11 development work. Use the File command to edit your source code, the Build->Assemble command to assemble your source code and the Build->Download command to communicate with the Ep2I711E9 board.

In the View->Option->Terminal Window Options menu, set the com port 1 or 2 to match your PC com port, set com port options at 9600, N,8,1, and check the "enable the terminal window" box which will disable Wytec hc11 tools.

In the View->Option->Assembler menu, make sure that the chip family is 68HC11 and you also can see that the default assembler name for the hc11 is **myasm11.bat**, not as11.exe. This is the only setup change we have to make from the original AsmIDE. If you would like to use your own assembler, you can make a batch file to replace the myasm11.bat.

If the assembler generates an error, it also will show the error's line number in the file along with an error message. Go to the line to make a correction.

If the terminal options are set correctly, the jumper is installed on the BM position of the J15 and the com port number is correct, you should see the following sign-on message every time the reset button is pressed. If it does not, the bottom window is still a message window. Click the terminal button on the bottom window will enable the terminal window display.

```
EVBplus.com: BUFFALO 3.4 - Bit User Fast Friendly Aid to Logical Operation
```

Press the Enter key, you will get the BUFFALO monitor prompt
>

Step 6

All sample programs are debugged and tested for your convenience and they are located in the folder c:\Ep2IDE\Ex_BUFFL. Here are steps to run your first sample program:

1. Click the File button to load the test.asm from c:\Ep2IDE\Ex_BUFFL to see all instructions on the beginning of this program.

2. Click Build button-> Assemble or click the assembler button on the toolbar to assemble your code and generate the test.s19 file. In order to display assembler messages, the bottom window is switched to the message window.

3. Click the terminal button of the bottom window to activate the terminal window and make sure that the BUFFALO monitor prompt '>' is shown on the terminal window, if not, press the Enter key.

4. At the prompt '>', type "LOAD T" <Enter>.

5. Click the Build->Download button, select the file c:\Ep2IDE\Ex_BUFFL\test.s19 for downloading

6. After the download is done, type G D000 to run the program.

It will run the TEST program in real time. The program will test the switches, scan keypad, send message to the LCD display, emulate an IR proximity sensor, adjust the LED display brightness, generate music and shift number 0 to F on the seven-segment display. At first you can press the PC0 and PC1 pushbutton switches and see the change on the PB0 and PB1 LEDs, then press the PA0 switch, the sound should come out.

To stop the program, you have to press the reset button momentarily.

For more details on all sample programs, please read the readme.txt in the c:\Ep2IDE\EX-BUFFL folder. All example programs are fully debugged, so the assembler won't generate an error report. If you have an error in your program, you must correct it before an s19 file can be generated.

You can try to run a different example program later after finishing reading this manual. You should always press the reset button before download a new program, because download may not work if interrupt was enabled by a previous program.

Software development with the BUFFALO monitor:

We are using the AsmIDE as a terminal program in the following instructions to create your source code. If you are using a different terminal program, the instructions may vary.

The steps to create your source code as follows:

1 Click the **File** button to open an existing file or create a new file.

The memory locations from $00-$2Cfor A family or $00-$2C and $100-$1FF for E family are available as the user DATA RAM. The BUFFALO monitor uses RAM locations at $2D-$FF. The 22K memory locations from $8800 to $DFFF are available to the user program CODE or DATA. In assembly language, you specify the starting address of your CODE by the ORG statement.

You can start DATA RAM at address $00 with the statement ORG 0 followed by RAM variables, such as:

```
        ORG   0
TEMP:   RMB   1         ; reserve one byte of RAM for temp storage
XTEMP:  RMB   2         ; reserve two bytes of RAM for temp storage
```

If your program is small, say less than 4K, you can start your program at address $D000 with the statement ORG $D000 followed by your program, such as:

```
        ORG   $D000
        LDS   #$8FFF    ; initialize stack point
```

It will assemble your source program and generate hex code within 4K locations from $D000 to $DFFF. If your program is larger, you can change the ORG $D000 to ORG $C000, or ORG $A000. You cannot use ORG $E000 or ORG $F000, because the BUFFALO monitor occupies the highest 8K locations ($E000-$FFFF)

After fully debugged your code, you have to relocate it to $F000-$FFFF or $E000-$FFFF and re-assemble it again before programming it into a 68HC711E9. You don't have to do this with the Wytec debugger, WBUG11.

Under the BUFFALO monitor, the Ep2711E9 board cannot stop your program if it's hung in a loop. The only way to stop it is to reset the board. The problem with resetting the board is that you would not know where the 68HC11 was hung. It also leaves a SWI instruction on all breakpoint addresses and you have to re-download your s19 file all over again.

Here is a very simple program, but it's complete. It will flash the PB7 LED when it's running.

For a good programming practice, you should always make the first line of your code with the **LDS** instruction.

```
PB7:        EQU   $80            ; bit 7 of port B
FLS_RATE:   EQU   $8B00          ; change this number will change LED flash rate

            ORG   $D000
START:LDS   #$8FFF
            LDX   #$1000         ; X register points to the register block
BACK:       BSET  0,X PB7        ; turn on the PB7 LED by setting PB7=1
            JSR   DELAY
            BCLR  0,X PB7        ; turn off the reset LED by resetting PB7=0
            JSR   DELAY
            JMP   BACK
DELAY:LDY   #FLS_RATE
DLY:        DEY
            BNE   DLY
            RTS
            END
```

2. Save your file frequently while editing. If you are creating a new file and giving the file a name to save, enter the file name including the file extension, such as "test.asm", not just "test".

3. Click Build button-> Assemble, or click the assembler button on the toolbar to assemble your code and generate an s19 file. If the assembler detects an error, the error message will show the line numbers in your source code that made error. Beware that sometime (not very often, but it does happen) the freeware assembler may indicate a wrong error line in the file and the actual line that caused error may be one line off.

WARNING: The free assembler will generate an error on the BSET, BCLR, BRSET and BSCLR instructions if more than one commas are used in those instructions. For instance, BSET 0,X,$20 or BRSET 0,X,$20,$F000 is illegal, but BSET 0,X $20 or BESET 0,X $20 $F000 is OK. You have to use a space character to separate fields, not a comma.

4. Go to the line and correct the errors and go back to step 3, until there are no errors.

The EVBplus2 board is an ordinary EVB board with the BUFFALO monitor, but it's an easy-to-use and powerful In-circuit Emulator type of development system when it's under the control of the Wytec Symbolic Debugger, WBUG11. If you don't use the WBUG11, you will not be able to exploit all of the features. Once you have used the WBUG11, you will not want to use the BUFFALO again. It's a fact that many of our users would agree.

GETTING STARTED with Wytec Phantom Monitor and WBUG11

The Memory Map with the Wytec phantom monitor:

$0000-$00FF	On-chip 256 byte RAM for the 68HC11A1
$0000-$01FF	On-chip 512 byte RAM for the 68HC11E1
$0800-$0FFF	Wytec monitor firmware in U2
$1800-$1FFF	Wytec monitor firmware in U2
$1000-$103F	On-chip 64 control registers
$2000-$3FFF	External device, /CS on the pin40 of J1
$4000-$5FFF	8K EEPROM of U3
$6000-$61FF	not used
$6200-$63FF	65C22
$6201	PORTF
$6203	DDRF
$6200	PORTG
$6202	DDRG
$6400-$65FF	68B50
$6400	UCTRL, USTAT
$6401	UART
$6600-$67FF	not used
$6800-$7FFF	not used
$8000-$87FF	RAM used by Wytec monitor
$8800-$FFFF	RAM for user code or data

Any attempt to write to the locations ($6000-$87FF) will have unpredictable results and must be avoided in the user program.

Before testing the board with the Wytec monitor, place a jumper on the top position (labeled with 'WYTEC' which stands for Wytec Monitor) of the J15 and another jumper on the right position of the J22. To test the board, follow the steps 1 through 7 below:

Step 1.

Plug the AC adapter into a wall outlet, and plug the DC plug at the other end into the DC jack on the lower right corner of the Ep2711E9 board. During the **initial power up**, the reset LED should blink 4 times and all other LEDs must be off. If not, turn over to the Questions & Answers section of the user manual.

Step 2.

Plug the DB9 male end of the cable into the DB9 connector P2 on the **upper right** corner of the Ep2711E9 board, and plug the DB9 female end of the cable to the COM1 or COM2 port on the PC. The DB9 connector P3 on the lower right side of the board is the 68HC11 SCI port that can be used by user's application program.

Step 3.

Press the reset button on the Ep2711E9 board, and the reset LED, which is located above the switch, should now blink only twice. If not, turn over to the Questions & Answers section of the user manual.

Step 4.

To invoke the AsmIDE, right click the Start button, then left click the Explorer, left click on C:\Ep2IDE, left click on AsmIDE.exe. If you have created a shortcut icon on the desktop, just double click the AsmIDE icon on the desktop.

Step 5:

The default setting of the AsmIDE for the Ep2711 board is created in a text file named c:\Ep2IDE\AsmIDE.ini. In the future if you get lost with all the changes, you always can copy this file into c:\Ep2IDE.

In the View -> Options -> Wytec hc11 tools menu, you must check the 'Wytec Tools Enabled' box to use the Wytec Debugger software, WBUG11. You can change the COM port number, but you do not have to change the baud rate, because the WBUG11 will set it at 38.4K for you. Please note that the default debugger startup batch name is **wbug11.exe**, not mydebug.bat. If you left click the assembler tab under the AsmIDE options menu, you also can see that the default assembler name for the hc11 is **myasm11.bat**, not as11.exe. These are two setup changes we have to make from the original AsmIDE.

Step 6:

All sample programs are debugged and tested for your convenience and they are located in the folder c:\Ep2IDE\Ex_WYTEC. Here are steps to run your first sample program:

1. Click the File button to load the test.asm from c:\Ep2IDE\Ex_WYTEC to see all instructions on the beginning of this program.

2. Click the Build button-> Assemble or click the assembler button on the toolbar to assemble your code and generate the test.s19 file.

3. Click the Build button-> Wytec hc11 tools, or click the debugger button on the toolbar that will bring up a small dialog window. You can see the file to be downloaded is c:\Ep2IDE\Ex_WYTEC\test.s19 **(The current download s19 file name is always updated by the assembler).**

4. Now you **MUST** click the "Select this file" button, which will store the file name test.s19 into a text file named c:\Ep2IDE\Ep2IDE.txt, so the debugger can automatically download the test.s19 by reading the Ep2IDE.txt when it's invoked. This is a very **important step** and you must do it when you want to debug a different program.

5. Click the Debugger button and the debugger will automatically download the test program TEST.s19. The programmer button is used to program the on-chip EPROM of a 68HC711E9 and you can ignore it for the time being.

6. At the prompt Ep6811>, enter g \start (the label is case sensitive) Enter.

It will run the TEST program in real time. The program will test the switches, scan keypad, send message to the LCD display, emulate an IR proximity sensor, adjust the LED display brightness, generate music and shift number 0 to F on the seven-segment display. At first you can press the PC0 and PC1 pushbutton switches and see the change on the PB0 and PB1 LEDs, then press the PA0 switch, the sound should come out.

You can press the ESC key on the PC keyboard to stop the program. If you stop the program, the speaker will generate clicking noise, because the output comparator is still interrupting the 68HC11. To stop the clicking noise, you can press the reset button momentarily.

For more details on all sample programs, please read the readme.txt in the c:\Ep2IDE\Ex_WYTEC folder.

All example programs are fully debugged, so the assembler won't generate an error. If you have an error in your program, you must correct it before an s19 file can be generated.

You can try to run a different example program later after finishing reading this manual. You should always press the reset button before download a new program, because download may not work if interrupt was enabled by a previous program.

Step 7:

To run another program, activate the AsmIDE from the task bar at the bottom of the screen (this will minimize the debugger window), click the Build button-> Wytec hc11 tools, or click the debugger button on the toolbar that will bring up the Wytec hc11 tools dialog window. Use the browser to choose the s19 file you want to download, such as ex2.s19, then **click the "Select this file"** button and close the dialog window (Do not click the Debugger button, because the debugger is already invoked). Now activate the **WBUG11from the task bar** and press the F10 function key and R option, the ex2.s19 will be automatically downloaded. At the prompt Ep6811>, type g \start Enter.

Software development with the Wytec's Phantom Monitor:

The steps to create your source code as follows:

1 Click the **File** button to open an existing file or create a new file.

The memory locations from $00-$FF for A family or $00-$1FF for E family are available as the user DATA RAM. The EVBplus2 board does not use RAM locations at $2D-$FF, as the BUFFALO monitor does. The 30K memory locations from $8800 to $FFFF are available to the user program CODE. In assembly language, you specify the starting address by the ORG statement.

You can start DATA RAM at address $00 with the statement ORG 0 followed by RAM variables, such as:

```
        ORG   0
TEMP:   RMB   1         ; reserve one byte of RAM for temp storage
XTEMP:  RMB   2         ; reserve two bytes of RAM for temp storage
```

If your program is small, say less than 4K, you can start your program at address $F000 with the statement ORG $F000 followed by your program, such as:

```
        ORG   $F000
        LDS   #$FF      ; Initialize the stack point
```

It will assemble your source program and generate hex code within 4K locations from $F000 to $FFFF. If your program is larger, you can change the ORG $F000 to ORG $E000, or ORG $C000. You can use ORG $E000 or ORG $F000, because there is no BUFFALO monitor in the highest 8K locations ($E000-$FFFF)

After fully debugged your code, you do not have to relocate your code and re-assemble it again. You can directly program the s19 file into a 68HC711E9. Your code is final for stand-alone operation after finishing your debugging session, it works just like a real time In-Circuit Emulator.

The Ep2711E9 board can stop your program if it's hung in a loop, when you press the ESC key at the PC keyboard, it will interrupt the Ep2711E9, but the BUFFALO monitor can't stop the program unless you reset the board. The problem with resetting the board is that you would not know where the 68HC11 was hung. It also leaves a SWI instruction on all breakpoint addresses and you have to re-download your s19 file all over again.

Here is a very simple program, but it's complete. The program source code is similar to the tutor2.asm in the directory c:\Ep2IDE\Ex_WYTEC. It will flash the PB7 LED when it's running.

For a good programming practice, you should always make the first line of your code with the **LDS** instruction.

```
PB7:        EQU   $80              ; bit 7 of port B
FLS_RATE:   EQU   $8B00            ; change this number will change LED flash rate
            ORG   $F000
START:LDS   #$FF            ; the top of A1 internal RAM
            LDX   #$1000           ; X register points to the register block
BACK:       BSET  0,X PB7          ; turn on the PB7 LED by setting PB7=1
            JSR   DELAY
            BCLR  0,X PB7          ; turn off the reset LED by resetting PB7=0
            JSR   DELAY
            JMP   BACK
DELAY:LDY   #FLS_RATE
DLY:        DEY
            BNE   DLY
            RTS
            ORG   $FFFE
            FDB   START            ; reset vector
            END
```

2. Save your file frequently while editing. If you are creating a new file and giving the file a name to save, enter the file name including the file extension, such as "test.asm", not just "test".

NOTE: Since the debugger is written in DOS, the filename should not be longer than 8 characters.

3. Click Build button-> Assemble, or click the assembler button on the toolbar to assemble your code. IF your code has no errors, it will generate an s19 file, a listing file and a symbol file. If the assembler detects an error, the error message will show the line numbers in your source code that made error. Beware that sometime (not very often, but it does happen) the freeware assembler may indicate a wrong error line in the file and the actual line that caused error may be one line off.

4. Correct the errors and go back to step 3, until there are no errors.

5. After your code is successfully assembled, you click the Build button-> Wytec hc11 tools-> "select this file" button to update the file c:\Ep2IDE\Ep2IDE.txt for the debugger.

6. Close the dialog window, activate the **WBUG11 from the task bar** and press the F10 function key and R option, the debugger will automatically download 3 files, namely YOUR_FILENAME.S19, YOUR_FILENAME.SYM and YOUR_FILENAME.PAR. The S19 file is the hex code, the SYM file is the symbol file, so you can use symbols in commands instead of hex numbers. The PAR file is the parameter file that includes the EEPROM programming enable/disable flag, INIT, TMSK2, OPTION and BPROT register values, the breakpoint addresses, and the memory display addresses.

The assembler does not make a PAR file, but the Debugger does. When you first invoke the debugger, there would be no PAR file exists. When you exit the debugger, the debugger will create one by saving the current settings. **The PAR file is automatically saved after you exit the debugging session.**

After that the par file will always exist. Also the Write command of the debugger can make a PAR file. After invoking the Write command in the debugger, it will prompt you to enter a choice among U, S and P options, choose the P option and give a full file name such as test.par, it will make a PAR file for you.

The most useful feature of the PAR file is to remember the INIT register value. Suppose you relocate the 68HC11 control registers to a different memory block by changing the value of the INIT register in the beginning of your source program (the INIT register can only be changed in the first 64 E cycles in expanded mode), you have to use the Init command to change the INIT register to match that value before running your program under the WBUG11, otherwise your program will not run. When the PAR file is loaded, it will generate a system reset and force the 68HC11 to take the new INIT register value from the PAR file. For more information, see the Questions & Answers section of the user manual.

7. When download is finished and at the prompt Ep6811>, you can run your program by entering go \start where the start is the label of the starting address in your source code, or enter Go F000 if you know the starting address is $F000, but do not enter command too earlier. After download, the PC will reset the board, during the reset; there would be some communication between the PC and the board. If you enter the command too earlier, it could disrupt the communication and you will get an error message.

In the command line, **ALL NUMBERS ARE HEXADECIMAL** and the $ sign is not needed. Also you should notice that the label \start has the back slash in the front. **All hex numbers may be substituted by symbols starting with the backslash '\'.** For more information read the Command format section in the user manual.

8. During the debugging session, if you want to modify your source code, you don't have to exit the debugger. You can activate the AsmIDE to edit your code and save your new code in the same file name, then click the Build button-> Assemble or click the assemble button on the toolbar to assemble the code. After the new S19 file is generated, because you did not change the file name, you don't have to **click the "Select this file"** button again. Only thing you have to do is to click the "**WBUG11**" button at the task bar at the **bottom line of the screen** to activate debugger window, then enter the F10 key and the R option to download your files and start to debug again.

Single Board Computer application:

The Memory Map in Single Board Computer application:

$0000-$00FF	On-chip 256 byte RAM for the 68HC11A1
$0000-$01FF	On-chip 512 byte RAM for the 68HC11E1
$0800-$0FFF	Wytec monitor in U2, some subroutines are callable from user programs
$1800-$1FFF	Wytec monitor in U2, some subroutines are callable from user programs
$1000-$103F	On-chip 64 control registers
$2000-$3FFF	External device, /CS on the pin40 of J1
$4000-$5FFF	not used
$6000-$61FF	not used
$6200-$63FF	65C22
$6201	PORTF
$6203	DDRF
$6200	PORTG
$6202	DDRG

$6400-$65FF	68B50
$6400	UCTRL, USTAT
$6401	UART
$6600-$67FF	not used
$6800-$7FFF	not used
$8000-$DFFF	RAM for user data
$E000-$FFFF	User code in U3 (8K EEPROM)

Program the 8K EEPROM in the U3 and 512 byte on-chip EEPROM:

You can program the 8K EEPROM in the U3 and the 512-byte on-chip EEPROM ($B600-B7FF) under control of the Wytec debugger WBUG11. Use the Load command as described on the page 10. The instructions will be shown on the screen when the L command is entered.

If you want to use the board as a SBC, you can program your application s19 file into the U3 (8K EEPROM) and place a jumper on the bottom position of the J15 (at the left side of the speaker). It will completely bypass both the Wytec's proprietary phantom monitor and the BUFFALO monitor. The address range of the U3 is changed from $4000-$5FFF to $E000-$FFFF and the address range of the U4 is changed from $8000-$FFFF to $8000-$DFFF. The program code will auto start from the U3 after reset or power up.

You can only program the U3 (8K EEPROM) under control of the Wytec debugger WBUG11 and your application s19 file must be assembled in the address range of $E000-$FFFF. Here are the steps to program the U3 with the file test.s19:

1. Move the jumper to the right position on the J21 to enable EEPROM write.
2. Enter Load command and select the E option to download an s19 file into the U3.
3. Enter test.s19 as the file name to be downloaded.
4. When the programming cycle starts, the PB0-PB7 LEDs will act as a binary counter.
5. The programming is done when LEDs stop counting.
6. Move the jumper to the left position on the J21 to write-protect the U3.
7. Move the jumper on the J15 to the bottom position (labeled with the 'U3').
8. Press the reset button, your application program should run.

Before programming you have to make sure that the jumper on the J21 is on the right side to enable EEPROM write, otherwise the programming software will not continue. After programming place the jumper on the left side of the J21 to write-protect the EEPROM, otherwise the EEPROM could lose data quite easily after running a bad program during a debugging session. The 28-pin DIP socket for the U3 is not built for a high volume production programmer. It will not last much longer than 50 insertions.

The procedures to program the 512-byte on-chip EEPROM ($B600-B7FF) are the same above steps except the step 2. Select the B option instead of the E option after the Load command id entered.

About AsmIDE:

The AsmIDE is written by Eric Engler and it's simple and easy to use. It offers all basic features that you need to learn 68HC11 programming with this board. Some time a simple IDE may not be a bad idea; at least students don't need to spend too much time to learn how to use the IDE, so they will have more time to focus on learning the 68HC11 which is their main objectives of taking the course. If you spend a lot of time to master a bloated IDE and then you don't use it for a while, you probably would forget how to use it anyway, so why spend time on something that you are going to lose it anyway? On the contrary, a simple, easy to use IDE saves your time and you will remember how to use it even you don't use it for a long period of time.

The most valuable asset of the EVBplus2 board is the WICE debugger, **WBUG11**. Some boards that you can buy today may have a bloated IDE, but you probably would get a BUFFALO type monitor for debugging.

Using your own assembler or editor for WBUG11:

If you would prefer to use your own assembler or editor instead of the AsmIDE, you can use the AsmIDE for launching the WBUG11. The full path name of the file that you work on must agree to the file name shown on the Wytec hc11 tools dialog window, otherwise the WBUG11 would not be able to locate your s19 file to be downloaded.

Make your own monitor:

If you want to make your own monitor to install a high language source level debugger, you can treat your monitor as an application and program it into the U3 (8K EEPROM) according the instruction shown on the previous page.

Using the board as a 68HC711E9 programmer:

If you need to program a 68HC711E9, you can use this board as a 68HC711E9 programmer, but only under control of the Wytec debugger WBUG11. At first, you must make sure that the program is fully debugged. To activate the programmer, click the Build button -> Wytec hc11 tools, or click the debugger button on the toolbar that will bring up a small dialog window. If the current download s19 file is correctly shown on the dialog window, click the "Select this file" button and then click the "**programmer"** button to program the 68HC711E9. The programming is done in the bootstrap mode and the programming instructions will be displayed on the screen step by step. The memory addresses range for the 68HC711E9 is from $D000 to $FFFF. If your S19 file contains addresses outside of this range, an error message will be generated and the chip will not be programmed.

During the programming, the data will be automatically verified.

The 52-pin PLCC socket is not built for a high volume production programmer. It will not last much longer than 100 insertions. When the contacts of the PLCC socket are worn, you can use a fine dental tool to pry up the contacts a little bit to extend its life considerably.

ON-BOARD HARDWARE

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 emulated by the PRU chip 68HC24. The PRU stands for port replacement unit.

The port B is an output port and each port B line is monitored by a LED. The port C is a bi-directional I/O port and it is connected to an 8-position DIPswitch. The DIPswitch is connected to GND via eight 4.7K resistors, so it's not dead short to GND. When the port C is programmed as an output port, the DIPswitch setting is ignored.

The PA0 switch is used as a general input switch, except during the initial power up. During the initial power up, press and hold the PA0 switch while press the RESET button will force the 68HC11 to enter test mode. In the test mode, the configure register can be modified. The PA3 and PA4 headers are the outputs of the Output Comparator 5 and 4, and they can be used to drive robot servos.

The Port E is an 8-bit ADC or a general input port. The trimmer VR1 is hard wired to the PE7 input of the ADC port via the J10, but the trace can be cut if the PE7 must be used by target circuit for a different purpose.

An on-board logic probe LED is hard wired to the pin 55 of the female socket connector P1F and can be used to monitor high or low status at any point of the circuit as a logic probe.

The U16, LTC1262 or MAX662, provides a 12 V programming voltage for the 68HC711E9.
The U18, 74HC14, generates the 38.4K baud for the U5, 68B50, and it also provides the 38KHz square wave to the IR transmitter.

The U9, SN75176, converts the TTL signal from the SCI to the RS485 differential signals and visa versa.
The two RJ12 jacks, JK1 and JK2, can be used to daisy chain many Ep2711E9 boards together for a network application. The connections on the JK1 and Jk2 are identical, so either one can be input or output.

Two I/O ports, port F and port G are added through the U1, VIA 65C22. These are bi-directional ports. The port F and port G are the port A and port B of the 65C22, respectively. The address locations for the ports are as follows:

```
PORTF   $6201
DDRF    $6203    ; 1=OUTPUT, 0=INPUT
PORTG   $6200
DDRG    $6202    ; 1=OUTPUT, 0=INPUT
```

The CA2 output from the 65C22 is used to control the direction of the RS485 communication. If the CA2=0, the RS485 port, U9 DS75176, is set for receiver port; If the CA2=1, the RS485 port, U9 DS75176, is set for transmitter port.

```
CA2=0    RS485 receiver port
CA2=1    RS485 transmitter port
```

For users' convenience, there are two subroutines added to control the CA2. User can call RS485_RECV at $0800 to reset CA2 to 0, or RS485_XMIT at $0803 set CA2 to 1.
The following are all I/O subroutines in Wytec monitor that are callable from user's application programs:

```
                ORG     $0800    ; EVBplus2 Rev. C board I/O routines

RS485_RECV:     RMB     3        ; enables RS485 receiving mode
RS485_XMIT:     RMB     3        ; enables RS485 transmitting mode
GET_DATE:       RMB     3        ; gets current date from PTC
GET_TIME:       RMB     3        ; gets current time from PC
OUTSTRG00:      RMB     3        ; outputs a string terminated by 0
LCD_INI:        RMB     3        ; initializes the 16x2 LCD module
LCD_LINE1:      RMB     3        ; displays 16 char on the first line
LCD_LINE2:      RMB     3        ; displays 16 char on the second line
SEL_INST:       RMB     3        ; selects instruction before writing the LCD module
SEL_DATA:       RMB     3        ; selects data before writing the LCD module
WRT_PULSE:      RMB     3        ; generates a write pulse to the LCD module
```

The circuit is designed in such way that the value of all resistors and capacitors are not critical, they can be off -50% or +100%.

How to use the port F:

The port F is an 8-bit bi-directional port. Its primary usage is for a LCD display module. If the port is not used for LCD display, it can be used as a general-purpose I/O port that can be accessed via J2 or J3.

The pinouts of the J3 is as follows:

Pin 1	GND	
Pin 2	VCC (5V)	
Pin 3	Via a 100 Ohm resistor to GND	
Pin 4	PF0	RS pin for LCD module
Pin 5	GND	
Pin 6	PF1	EN pin for LCD module
Pin 7	Not used	
Pin 8	Not used	
Pin 9	PF2	
Pin 10	PF3	
Pin 11	PF4	DB4 pin for LCD module
Pin 12	PF5	DB5 pin for LCD module
Pin 13	PF6	DB6 pin for LCD module
Pin 14	PF7	DB7 pin for LCD module
Pin 15	Via an 18 Ohm resistor to VCC	LED backlight for LCD module
Pin 16	GND	

The pinouts of the J2 is as follows

Pin 1	PF7	Pin 2	PG7
Pin 3	PF6	Pin 4	PG6
Pin 5	PF5	Pin 6	PG5
Pin 7	PF4	Pin 8	PG4
Pin 9	PF3	Pin 10	PG3
Pin 11	PF2	Pin 12	PG2
Pin 13	PF1	Pin 14	PG1
Pin 15	PF0	Pin 16	PG0
Pin 17	VCC	Pin 18	VCC
Pin 19	GND	Pin 20	GND

How to use the port G:

The port G is an 8-bit bi-directional port. Its primary usage is for a 4X4 keypad interface. If the port is not used for keypad application, it can be used as a general-purpose I/O port that can be accessed via J2 or J6.
The pinouts of the J6 is as follows:

Pin 1	PG0	connects ROW0 of the keypad
Pin 2	PG1	connects ROW1 of the keypad
Pin 3	PG2	connects ROW2 of the keypad
Pin 4	PG3	connects ROW3 of the keypad
Pin 5	PG4	connects COL0 of the keypad
Pin 6	PG5	connects COL1 of the keypad
Pin 7	PG6	connects COL2 of the keypad
Pin 8	PG7	connects COL3 of the keypad

Keypad interface

PG0 connects ROW0 of the keypad via pin 1 of the 8-pin keypad header
PG1 connects ROW1 of the keypad via pin 2 of the 8-pin keypad header
PG2 connects ROW2 of the keypad via pin 3 of the 8-pin keypad header
PG3 connects ROW3 of the keypad via pin 4 of the 8-pin keypad header

PG4 connects COL0 of the keypad via pin 5 of the 8-pin keypad header
PG5 connects COL1 of the keypad via pin 6 of the 8-pin keypad header
PG6 connects COL2 of the keypad via pin 7 of the 8-pin keypad header

PG7 connects COL3 of the keypad via pin 8 of the 8-pin keypad header

The PG0-PG7 has a 100K pull-up resistor in each line.

The keypad scan routine sets PG3 low, PG0,PG1,PG2 high, and then test the PG4-PG7
If no key is down, PG4-PG7 remain high.
If PG7 = low, the key 15 is down.
If PG6 = low, the key 14 is down.
If PG5 = low, the key 13 is down.
If PG4 = low, the key 12 is down.

The keypad scan routine then sets PG2 low, PG0,PG1,PG3 high then test the PG4-PG7
If no key is down, PG4-PG7 remain high.
If PG7 = low, the key 11 is down.
If PG6 = low, the key 10 is down.
If PG5 = low, the key 9 is down.
If PG4 = low, the key 8 is down.

The keypad scan routine then sets PG1 low, PG0,PG2,PG3 high then test the PG4-PG7
If no key is down, PG4-PG7 remain high.
If PG7 = low, the key 7 is down.
If PG6 = low, the key 6 is down.
If PG5 = low, the key 5 is down.
If PG4 = low, the key 4 is down.

The keypad scan routine then sets PG0 low, PG1,PG2,PG3 high then test the PG4-PG7
If no key is down, PG4-PG7 remain high.
If PG7 = low, the key 3 is down.
If PG6 = low, the key 2 is down.
If PG5 = low, the key 1 is down.
If PG4 = low, the key 0 is down.
SPI port pinouts are as follows:

Pin 1	VCC (5V)	Pin 2	VCC (5V)
Pin 3	PF2 (LOAD)	Pin 4	PD2 (SPI DATA IN)
Pin 5	PF3 (STROBE)	Pin 6	PD3 (SPI DATA OUT from 68HC11)
Pin 7	not used	Pin 8	PD4 (CLOCK)
Pin 9	GND	Pin 10	GND

All on-board jumpers:

J1 40 pin logic analyzer connector, Motorola 68HC11 EVM compatible
J2 Port F and Port G, total 16 bits, the left side is port F and the right side is port G
J3 LCD port
J4 SPI connector
J5 Mode selector. The letter J5 is not printed on the PCB. It's a minor mistake.
J6 4 X 4 keypad interface
J7 RS485 direction control, hard wired
J8 HC11 SCI receiver source selector (numbering from top to bottom)
1= SCI PD0 receives signal from your target system via the P1 or P1F
2= SCI PD0 receives signal from the P3, DB9 connector for RS232 input
3= SCI PD0 receives signal from the JK1 or JK2 (RJ12 jacks) for RS485 input
4= SCI PD0 receives signal from the on-board IR receiver

J9 The PRG/RUN jumper is used for selecting the operating mode.
When it's on the right position, the board is used for programming the 68HC711E9 chip.
When it's on the left position, it's for debugging your code.

J10 Connects the VR1 trimmer pot to the PE7 of the ADC, it is hard-wired.

J11 Analog voltage reference selector
When it's on the right position, the on-board 5V DC is the reference voltage
When it'son the left position, the user target provides the ADC reference voltage
It's hard-wired to the on-board 5V DC reference voltage

J12 Clock selector.
When it's on the right position (labeled with 'INT'), the clock is provided by the on-board crystal.
When it's on the left position, (labeled with 'EXT'), the user target provides a HC compatible clock source.

J13 Clock output. It's not installed. If it's installed, the clock output of the 68HC11(pin 8) can be a clock source of user target board.

J14 Enables speaker. The speaker is driven by the PA5, Output Comparator 3 through the jumper J14. It's hard-wired.

J15 Monitor selector. Place the jumper on the top position for the Wytec monitor, the middle position for the BUFFALO monitor and the bottom position for auto-start the program in the U3.

J16 Connects the HC11 SCI's PD1 to all communication hardware (RS232, RS485 and IR transceiver) on this development board. It is hard wired.

J17 IR transceiver control source selector.

When the jumpers are on the upper position (labeled with 'SCI'), the HC11's PD1drives the IR transmitter and the HC11's PD0 receives the data from the IR receiver. The PD0 and PD1 can be general I/O lines or the SCI UART.
When the jumpers are on the lower position (labeled with 'PA26'), the HC11's PA6 drives the IR transmitter and PA2 receives the data from the IR receiver.

J18 Enables the 7 segment LED display driver U11, 74HC367. If the 7 segment LED display is not needed in an application, remove this jumper to turn off the display and save the power.

J19 Enables LCD backlight. If the LCD display is not needed in an application, remove this jumper to turn off the backlight and save the power.

J20 12V output from the U16 (5V-12V converter).

J21 U3 EEPROM write protect, place the jumper on the left position to disable (write-protect) EEPROM programming. Place the jumper on the right position to enable EEPROM programming.

J22 Enables BUFFALO trace function and the PA3 is connected to the XIRQ to enable the single-stepping operation when the jumper is on the left position. When it's on the right position, the PA3 is disconnected from the XIRQ.

Sensor port: The top line is the signal, the middle line is VCC and the bottom line is GND. Digital sensors can be connected to the PA0,PA1,PA2 and PA7. The analog sensor can be connected to the PE0-PE3. There is a 100K pullup resistor on each sensor input.

The MODEA and MODEB jumpers: They are not used in debugging sessions. They can be used for programming the 68HC711E9 OTP part in bootstrap mode.

The P3 DB9 female connector is configured as a **DCE** device and it can be directly connected to the PC 's COM port.

Application Circuit Corner (ACC) and Solderless Breadboard:

The footprints of four popular 68HC11 applications are laid out on the upper left corner of the PC board. The **ACC** consists of a DS1302 Real Time Clock with battery backup, a DS1620 Digital Thermometer and Thermostat, a 12V DPDT relay and a L293D Motor driver. In the future we will offer each circuit in a separate kit. With these application circuits, this board can easily be transformed to a complete platform for Robot, Home Automation or other useful applications.

The lower left corner is a small solderless breadboard with the similar pad pattern underneath. All I/O signals are available on the 60 pin female connector P1F. The 60 pin male header P1 can be used to connect all user

I/O boards made for the original Motorola 68HC11 EVB board. Use the breadboard for prototyping or solder the components directly to the board if you remove the breadboard. The EVBplus2 board is not complete without a small breadboard.

The schematic for the PC board is divided into sch1.jpg, sch2.jpg and sch3.jpg. The ACC schematic is in the sch_acc.jpg.

New RF transceiver and phone jacks:

The Rev C board incorporates circuit layout under the breadboard for a RF transceiver, a RJ12 jack and a RJ45 jack. In the future we will offer a kit for your experiment. The schematic is in RF_jack.jpg.

BUFFALO I/O routines.

Many 68HC11 books have example programs that access BUFFALO I/O routines. The BUFFALO I/O routines located at $FFA0-$FFCF. When using with the Wytec debugger WBUG11 with this board, the BUFFALO I/O functions are duplicated at $0FA0-$0FCF.

Following are the BUFFALO I/O routines' function description and jumper table:

```
                ORG    $0FA0

UPCASE          convert the character in A to uppercase
WCHEK           test the character in A for white space and returns with the Z bit set if A is a
                white space  (Space, comma, tab)
DCHEK           test the character in A for white space and returns with the Z bit set if A is a
                carriage return or white space  (Space, comma, tab)
INIT            initialize SCI, is not needed with the EP2711E9 board
INPUT           reads PC keyboard input
OUTPUT          writes the character in the A to CRT display
OUTLHLF         converts 4 most Significant Bit of A to ASCII and Writes it to CRT display
OUTRHLF         converts 4 most Significant Bit of A to ASCII and Writes it to CRT display
OUTA            output ASCII character in A to CRT display
OUT1BYT         converts the binary byte that is pointed to by X register to 2 ASCII bytes and
                write them to CRT display
OUT1BSP         it's OUT1BYT followed by sending a space to the CRT display
OUT2BSP         converts the binary word (2 bytes) that is pointed to by X register to 4 ASCII
                bytes and write them followed a space to CRT display
OUTCRLF         write carriage return, line feed to CRT display.
OUTSTRG         it's OUTCRLF followed by writing the ASCII string that is pointed to by
                X register to CRT display and until character is $04
OUTSTRG0        it's OUTSTRG without writing leading carriage return & line feed to CRT display
INCHAR          waits for an ASCII character from keyboard and put it in accumulator A
```

Jump Table

```
$0FA0           UPCASE
$0FA3           WCHEK
$0FA6           DCHEK
$0FA9           INIT
$0FAC           INPUT
$0FAF           OUTPUT
```

$0FB2	OUTLHLF
$0FB5	OUTRHLF
$0FB8	OUTA
$0FBB	OUT1BYT
$0FBE	OUT1BSP
$0FC1	OUT2BSP
$0FC4	OUTCRLF
$0FC7	OUTSTRG
$0FCA	OUTSTRG0
$0FCD	INCHAR

We have added several I/O routines in the beginning of our Phantom Monitor.
Following are the Phantom Monitor's I/O routines' function description and jumper table starting at $0800:

```
                ORG    $0800
```

RS485_RECV	sets the RS485 port to receiver mode.
RS485_XMIT	sets the RS485 port to transmitter mode
GET_DATE	X register points to a 10 byte RAM block before calling this subroutine, it returns the date information of host PC in the format of MM-DD-YYYY.
GET_TIME	X register points to an 11 byte RAM block before calling this subroutine, it returns the time information of host PC in the format of HH:MM:SS AM or HH:MM:SS PM.
OUTSTRG00	It's OUTSTRG0 except the ending character is $00, instead of $04.
LCD_INI	initialize a 16x2 LCD display module
LCD_LINE1	displays 16 characters on the first line of a 16X2 LCD display module
LCD_LINE2	displays 16 characters on the second line of a 16X2 LCD display module
SEL_INST	selects instruction before writing a LCD module
SEL_DATA	selects data before writing a LCD module
WRT_PULSE	generates a write pulse for LCD module
SEVEN_SEGMENT:	converts Accu A to its segment pattern, bit 7 of the Accu A = Decimal Point of the 4 digit LED display module.

Jump Table

$0800	RS485_RECV
$0803	RS485_XMIT
$0806	GET_DATE
$0809	GET_TIME
$080C	OUTSTRG00
$080F	LCD_INI
$0812	LCD_LINE1
$0815	LCD_LINE2
$0818	SEL_INST
$081B	SEL_DATA
$081E	WRT_PULSE
$0821	SEVEN_SEGMENT

The subroutine SEVEN_SEGMENT is a conversion routine that converts a hex numbers to its seven segment pattern. The hex number range is from $00 to $20.

At entry, A= hex #	At exit, A=segment pattern	The letter or number to be formed by the segment pattern
A=$00	A=$3F	0
A=$01	A=$06	1
A=$02	A=$5B	2
A=$03	A=$4F	3
A=$04	A=$66	4

A=$05	A=$6D	5
A=$06	A=$7D	6
A=$07	A=$07	7
A=$08	A=$7F	8
A=$09	A=$6F	9
A=$0A	A=$77	A
A=$0B	A=$7C	B
A=$0C	A=$39	C
A=$0D	A=$5E	D
A=$0E	A=$79	E
A=$0F	A=$71	F
A=$10	A=$3D	G
A=$11	A=$76	H
A=$12	A=$74	h
A=$13	A=$1E	J
A=$14	A=$38	L
A=$15	A=$54	n
A=$16	A=$63	o
A=$17	A=$5C	o
A=$18	A=$73	P
A=$19	A=$50	r
A=$1A	A=$78	t
A=$1B	A=$3E	U
A=$1C	A=$1C	u
A=$1D	A=$6E	Y
A=$1E	A=$08	_
A=$1F	A=$40	--
A=$20	A=$00	blank

IMPORTANT NOTES

The following are some important notes that you should know and they may save your time:

1. Things to do if the board does not work.

Many little mistakes can cause a big problem, especially for the new beginners. For instance, you debug your code in expanded mode, but the MODEB and MODEA are set for bootstrap, you know it won't work. If the jumper on the J19 is missing, the LCD backlight won't work and if the jumper on the J18 is missing, the 7-segment display won't be lit.

Before troubleshooting the board, you must apply the power to the board. Press the reset button, the LED should blink twice. If it does not happen, the board may not have 5V DC. Use a DMM to check voltages at the VCC test point. Sometimes it may be caused by a bad AC adapter or the AC adapter is not even plugged in.

Sometime the Config register has a wrong value, but the 68HC11 chip is still good.

To determine if the board malfunctions, you can restore the board jumper settings to the original default settings when you receiving the board. The default settings are as follows:

J5 Mode selector. No jumper installed.

J8 HC11 SCI receiver source selector (numbering from top to bottom)
The jumper is on the bottom position (labeled with 'IR')

J9 The PRG/RUN jumper is used for selecting the operating mode.
It's on the left position for debugging your code.
J12 Clock selector.
It's on the right position (labeled with 'INT'), the clock is provided by the on-board crystal.
J15 Monitor selector. It's on the middle position for the BUFFALO monitor.

J17 IR transceiver control source selector.

The jumpers are on the upper position (labeled with 'SCI'), the HC11's PD1drives the IR transmitter and the HC11's PD0 receives the data from the IR receiver.
J18 Enables the 7 segment LED display driver U11, 74HC367. The jumper is installed on this header.
J19 Enables LCD backlight. The jumper is installed on this header.
J22 Enables BUFFALO trace function. The jumper is on the left position.

If all above settings are correct and when you press the reset button, the reset LED should blink twice. If not, press the reset button, while holding down the PA0 switch to enter test mode. Sometime the Config. register has a wrong value, but the 68HC11 chip is still good and it will work in test mode. In the test mode you can use the F8 function key to re-program the value of the Config. register to $0D. If you cannot enter the test mode, the 68HC11 is **defective**.

The crystal is not soldered to the board and it's held by two machine pins. If you want to change the crystal, just unplug it and replace it with a new frequency, such as 9.8304 MHz. Don't cut a crystal leads too short, if it's shorter than ¼", its metal case could short the two machine pins.

2. Always reset the board before downloading a new program.

If the previous application program that you ran was aborted, then you may need to reset the board before downloading a new application program. The reset action will disable the interrupt that was enabled by the previous application. If the interrupt was caused by a timer and is not disabled, the timer interrupt will continue even it's not called for in your new application program. The result will be unpredictable.

3. Disconnect LCD module from the bottom of the main PCB first.

The LCD display module has 2 supporting nylon spacers. They fit tighter with the LCD module PCB than with the main PCB. So if you need to remove the LCD module from the board, the spacers should stay with the LCD module. That means you should use a pair of pliers to push up the spacers from the bottom of the main PCB to separate the spacers from the main PCB. Once the spacers are loose from the main PCB, you can easily unplug the LCD module.

.