ZAP

Source Level Cross Debugger

Installation and Setup Guide

ZAP Monitor Configuration for
Motorola 68HC11

PC/Windows 95/98/NT

Document Version 3.2 December 1999

Copyright © COSMIC Software Inc 1994, 1996, 1999

All Trademarks are the property of their respective owners

Installation Guide for ZAP MON 6811

This chapter describes how to install the COSMIC *ZAP C Cross Debugger and Monitor for the MC68HC11* on your host system.

- Preparing For Installation
- Monitor And Hardware Configuration
- Monitor Setup
- Using ZAP MON with Interrupts
- Modifying Monitor OperationVerifying Installation
- Compiling Programs for Cross Debugging
- Modifying and Saving the ZAP Environment

Preparing For Installation

Your ZAP package consists of the ZAP Debugger, installation script files, tutorial files, and message files. To use ZAP, the following is required:

- A PC with an 80386 or better microprocessor.
- Microsoft Windows 95/98 or Windows NT 4.0 operating system.
- 3-1/2” 1.44 Mb diskette drive.
- Hard disk drive with at least 5 Mbytes of free space.
- Minimum of 8 Mbytes of Extended RAM

Each software diskette in the package has a printed label identifying the product, the product version number and the license serial number. In the installation instructions that follow, we assume that your floppy disk drive is designated by A: and your hard disk partition by c:. If your system uses different device names for your disks, you should adjust the installation instructions accordingly.

Installation Process

ZAP is installed by an Installshield setup program. Throughout the installation procedure, there is an assumed default directory in which ZAP will be installed. This directory is c:\cosmic\11zapmon11. If you install ZAP in a different directory or on a different hard disk drive, you must substitute your specified location wherever you see C:\cosmic\11zapmon11.

Running the Installation Program

1. Insert the diskette marked “Disk 1 of ...” in your disk drive.
2. Open the Windows Explorer and select the appropriate drive letter by double clicking on it.

3. Double click on Setup.exe in the Windows Explorer window containing ZAP disk #1 to run the ZAP installation and setup program. Alternatively, you can use the RUN command from the start menu and type a:\setup to run the installation program.

An Installshield window should appear. The installation program, allows you to select the hard disk drive and directory where ZAP will be installed. Follow the on-screen instructions to complete the installation. If you do not specify your own installation directory, ZAP and its associated programs will be installed in the c:\cosmic\11zapmon11 directory. The installation program will also create a Cosmic Tools program group and ZAP program items.

Setting Communications Parameters

ZAP uses two Windows environment variables to determine which comport (serial port) and speed is to be used to communicate to the board. The environment variable ZAPPORT specifies which comport is to be used and ZAPSPEED specifies the baud rate for communication. ZAP supports the following com ports and communication speeds:To use the ZapPort and ZapSpeed variables, edit your zap

Ports	Baud Rates
COM1	**9600, 19200**
COM2	

initialization (.ini) file to include the following lines, and substitute your preferred serial port and baud rate from the table above. After the initial setup you can change the baud rate at any time by selecting a new baud rate from the menu “Target->Baud Rate”. e.g.

```
[Options]
ZapPort=COM2
ZapSpeed=28800
```

Line Delay

When downloading the Cosmic monitor to the board using the Terminal window you may need to add a delay to the ini file. Typically, PC’s running over 150 Mhz will require a small delay after sending each line of the monitor. The delay is specified in milliseconds. If the EVSbug monitor downloads part of the S-record and then returns with an error “INSUFFICIENT ILLEGAL ENTRY” this usually means you need to increase the delay. To change the delay edit the file zapr11.ini which is located in your windows directory.

```
[Terminal]
```

Delay=10

NOTE

1. To enable any changes you made to your “zapr11.ini” file, including setting environment variables you will need to restart ZAP.
2. The EVB board’s baud rate is fixed at 9600 baud, so you must change the ZAPSPEED variable to 9600.
3. The EVS board’s default speed is 19200. If you want to change to a lower speed you must change the baud rate in the I/O routines of the monitor and rebuild the monitor (i.e., IOEVS.S or IOEVSF1.S). See modifying Monitor operation later on in this chapter.

Monitor And Hardware Configuration

After installing ZAP you should make sure that the evaluation board is connected properly to your PC via a direct RS-232 serial cable between a serial port on your PC and the Terminal/PC port of the evaluation board.

To ensure that you have connected the board properly you should refer to and follow the installation and testing instructions supplied with your board. Once this is complete you are ready to configure your board to run ZAP.

ZAP uses a RAM-based monitor to communicate with the evaluation board. The monitor performs all the low-level tasks required by ZAP such as reading and writing to memory and CPU registers, single stepping, and setting breakpoints. There are three different board configurations and several versions of the chips available for each board. Since each board communicates with the PC in a slightly different way it is necessary to have a different monitor and hardware configuration for each board. It is also necessary to use a different monitor to reflect differences among chip versions. For example the default location for the internal registers is different between the K4 and the E series. Please refer to the appropriate section for hardware setup and monitor configuration pertaining to your specific board and chip version. ZAP currently supports the following Motorola evaluation boards with corresponding chip versions.

- EVB Chip Versions Ax, Ex
- EVM Chip Versions Ax, Ex
- EVS Chip Versions Ax, Ex, F1, K4, D3, PH8

EVB Configuration

The Motorola EVB board uses the monitors MXEVB.SR and

MEVB.SR. These are supplied prebuilt in the Motorola S-record format. Both EVB monitors are linked to run at address 0x6000 and use approximately 550 bytes of user RAM. The monitors assume the UART (MC6850) is available at address 0x9800. ZAP uses the UART to communicate with the board so it must be available at address 0x9800. If you have modified this address you must modify IO.s, reassemble IO.s and rebuild the monitor using BUILDEVB.BAT OR BLDXEVB.BAT for the non-maskable configuration. If you want to relink the monitor to run at an address other than 0x6000, you must modify the text bias (-b0x6000) address given in the file MEVB.LKF or MXEVB.LKF., then rebuild the monitor.

If your application needs to use TOC5 then you must modify RUN.S to use another Timer Output Compare function, such as TOC3. Reassemble RUN.S and rebuild the monitor The file RUN.S is commented to help you do this. RUN.S also assumes that 1 CPU cycle is equal to 1 timer tick and loads the value of 48 plus the value currently stored in TCNT in order to time out the preprocessor after the minimum number of cycles needed to execute 1 instruction. This feature is used by ZAP to single step the processor by one assembly language instruction. RUN.S also assumes that the 68HC11 control register map is located in it's default location of 0x1000. If you have mapped it to another address you must modify the define BASE=1000H in RUN.S to match this address and reassemble RUN.S. You must also rebuild the monitor at this point. The monitor occupies about 750 bytes of code/data.

The module RUN.S assumes that two resources are available:

a) The SWI interrupt is available for single stepping at the C level.

b) The TOC5 interrupt is available for single stepping in the assembly language level.

ZAP requires the following hardware modifications to the EVB board. You must choose whether you want to run the Non-Maskable configuration or the Maskable configuration. We recommend the non-maskable configuration unless you need to use the XIRQ

interrupt and you don't need to debug your interrupt service routines.

Standard EVB Board Modifications

Non-Maskable Configuration

- Wire XIRQ to PA3 (TOC5). This can be done on the MCU I/O port connector P1, use monitor MXEVB.SR.

Maskable Configuration

- Wire XIRQ and IRQ to any +5v line so they won't interfere with the TOC5 interrupt used for single stepping. This can be done on the MCU I/O Port connector P1, use monitor MEVB.SR.

EVM Configuration

The monitors for the 68HC11A0,1,2,3,E0,1,2,9 series are supplied prebuilt in the Motorola S-record format in the MEVM.SR & MXEVM.SR files. The EVM monitors are linked to run at 0xFD00 and use approximately 510 bytes of user RAM. If you wish to load the monitor at a different address you must modify the MEVM.LKF file or MXEVM.LKF file for the non-maskable version to set the new code and data bias addresses to match the new load addresses. The standard EVM board configures user RAM as read-only memory. In order to use ZAP you need to make a small modification to the board to enable user RAM as read/write memory. You also need to remove the serial enable header (jumper J18) to allow ZAP to communicate to the monitor through the DUART (MC2681). The EVM monitors assume the external DUART is always available at 0xA200. If operating the EVM board in stand alone mode (no application board attached) or single chip mode the DUART is always available so you need only make the standard modifications. However, in expanded mode the external DUART is NOT always available due to addressing conflicts with the expanded bus buffers. So if you want to run the board in expanded mode you must make an additional

hardware modification as described in the “Expanded Mode Hardware Modifications” section.

The same configuration remarks for RUN.S as for the EVB board noted above, apply to the EVM board.

Standard EVM Hardware Modifications

1. Cut U27-4. See diagram below.
2. Tie U27-4 to ground to enable user emulation RAM as read/write.
3. Remove J18 to enable external DUART (MC2681).

Non-Maskable Configuration

4. Wire XIRQ to PAC (TOC5). This can be done on either J1 or J3, depending on the mode. With this configuration you are free to use IRQ in your application. Use monitor MXEVM.SR.

Maskable Configuration

5. Wire XIRQ and IRQ to any +5v line so they won’t interfere with the TOC5 interrupt used for single stepping. This can be done on either J1 or J3, depending on the mode of operation. Use monitor MEVM.SR.

Expanded Mode Hardware Modifications

These modifications to run the EVM board in expanded mode resolve a problem which is peculiar to this board only. In this case you must provide another debugging serial port either from your target board or the 68HC11 on-chip SCI if your hardware does not use it and you provide the necessary hardware buffering.You must also modify the file IOEVM.S to reflect your new serial port addressing, reassemble it and rebuild the monitor using BUILEVM.BAT or BLDXEVM.BAT. Alternatively, you can install yet another hardware patch to your EVM board to get around the problem.

6. Cut pin 9 on U10, this is a delicate operation because the tracks cross the pin. The pin must be bent up to allow a wire to be soldered to it.
7. Add a wire between pin 9 of U10 and pin 8 of U38.
8. Jumper J11 should be removed.

NOTE

This additional modification will disable the pseudo EEPROM (NCN signal) so you must use the internal EEPROM of the chip instead of the pseudo eeprom on the board.

EVS Configuration

The EVS monitors are linked to run at 0xFD00 and use approximately 560 bytes of user RAM. The monitors assume that the DUART (R65C52) is addressed at 0x7C00. Unlike the EVM board, the DUART is available in all modes of operation.

The same configuration remarks for RUN.S as for the EVB board noted above, apply to the EVS board unless noted in the following section.

NOTE

There is a different monitor pair for each of the different versions of the emulator board as shown below:

EVS Ax, Ex: Use either MEVS.SR or MXEVS.SR.

EVS F1: Use either MEVSF1.SR or MXEVSF1.SR.

EVS D3: Use either MEVSD3.SR or MXEVSD3.SR

EVS K4 Use either MEVSK4.SR or MXEVSK4.SR

EVS PH8 Use MEVSPH8.SR

The RUNK4.S assumes that the 68HC11K4 control register block is mapped at address 0x0000. If you have modified its location you BASE=0000 in RUNK4.S. This module also assumes that the SWI and the TOC4 are available for debugging purposes. If you application uses TOC4 you must modify and reassemble RUNK$.s and rebuild the monitor using BLDEVSK4.BAT or BLDXEVK4.BAT. RUNK4 also assumes that 1 CPU cycle is equal to 2 timer ticks and loads the value of 86 plus the value currently being stored in TCNT in order to time out the processor after the minimum number of cycles needed to execute 1 instruction. This feature is used by ZAP to single the processor by one assembly instruction.

As for the 68HC11F1, the K4 monitor assumes that the DUART is at address 7C00. Unlike the EVM board, the DUART(R65C52) is accessible in all modes of operation including expanded multiplexed mode.

EVS board users must make sure they have jumpered the board to enable read/write memory. Zap requires this to allow patching users code at the start of every C source line with the SWI instruction.

Standard EVS Board Modifications (Platform Board PFB)

1. Remove J1 and J2 to disable the segmented emulation RAM which overlays the chips internal RAM at 0x000 and the internal register block at 0x1000.
2. Remove J8 to disable segmented memory used by the DUART at 0x7C00.
3. Remove J9 and J10 to enable use of the Rockwell R65C52 DUART in user mode.
4. Move J14 to enable (pins 1-2) to enable the 32K of unsegmented emulation RAM.
5. Move J15 to 0000/7FFF (pins 1-2) to map the segmented emulation RAM from 0x000 through 0x7FFF and unsegmented emulation RAM from 0x8000 through 0xFFFF.
6. Move J16 and J17 to disable (2-3) to disable write protection for the emulation RAM.

Non-Maskable configuration

7. Wire XIRQ to PA3 (TOC5) This can be done on the MCU connector on the emulator board. With this configuration you are free to use IRQ in your application. Use either MXEVS.SR, MXEVSF1.SR, MXEVSK4.SR. or MXEVSD3.SR

Maskable configuration

8. Wire XIRQ and IRQ to any +5v line so they won't interfere with the TOC5 interrupt used for single stepping. This can be done on the MCU connectors on the emulator board. Use either MEVS.SR, MEVSF1.SR, MEVSK4.SR., MEVSD3.SR. or MEVSPH8.SR

Monitor Setup

ZAP uses a small monitor program to communicate with the evaluation board. This monitor, which is provided with ZAP, must be downloaded and started each time you power up the board. You can download the monitor using any communications program. However, the easiest way to download the monitor is to use the terminal window provided by ZAP. Upon startup, ZAP tries to communicate with the monitor. If it is unable to establish communication with the monitor ZAP will automatically open up a terminal window. You can also use the Windows' HyperTerminal program to download the monitor. This method is also described in the next section.

Using the ZAP Terminal Window

The ZAP Terminal Window opens automatically when you startup ZAP if the monitor is not already running. The terminal window will use the parameters specified by ZAPPORT and ZAPSPEED in the ZAP.INI file to try to establish communications. You must make sure these parameters are set to appropriate values for your board. See the section "Configuring Your System" earlier in this chapter for details.

ZAP Terminal Window

1. Press the Master Reset Button on the Evaluation Board and type a couple of carriage returns. You should receive the boards BUG monitor prompt, which should look something like the following:

```
EVSbug11 Rev 2.0
P=FFFF   Y=FFFF   X=FFFF  A=FF   B=FF  C=D0   SX.I....   S=00FF
>
```

2. Click the mouse in the Terminal Window and Type *load t* at the bug prompt.

```
>load t <ENTER>
```

3. Click on *Load* in the Menu bar to open the Monitor *Load* browser window.

Browse and select the appropriate S-record monitor file and click on OK. You should see the S-Records scroll across the Terminal Window. When ZAP is finished downloading the monitor you need to start the monitor running.

Monitor Load Browser Window

Motorola Evaluation Board Monitor Support		
Evaluation Board Type	**Maskable Configuration Monitors**	**Non-maskable Configuration Monitors**
EVB Ax, Ex	**MEVB.SR**	**MXEVB.SR**
EVM Ax, Ex	**MEVM.SR**	**MXEVM.SR**
EVS Ax, Ex	**MEVS.SR**	**MXEVS.SR**
EVS F1	**MEVSF1.SR**	**MXEVSF1.SR**
EVS K4	**MEVSK4.SR**	**MXEVSK4.SR**
EVS D3	**MEVSD3.SR**	**MXEVSD3.SR**
EVS PH8	**MEVSPH8.SR**	

To start the monitor for the **EVM and EVS** boards, press the **<User Reset>** switch on the board.

5. To start the Monitor for the **EVB** board type:

```
>call 6000
```

6. Once the monitor is running, you can test it by pressing the space bar a couple times. You should see the monitor prompt (>) echoed each time. If the monitor prompt does not appear, go back to step one and try downloading the monitor again. Once you get the monitor prompt, exit the terminal program by closing the window at the control prompt. You must exit the Terminal Window not just switch applications because the Terminal Window will not release the comport until it exits. Once you exit the

Terminal Window ZAP is ready to load an object file. Go to the section “Verifying Installation” for details.

Using Windows HyperTerminal Program

Locate the HyperTerminal program in the Accessories program group of the Windows Program manager and Start HyperTerminal by double clicking on it.

1. Choose *New Connection* from the File menu and create a new connection.
2. Select the desired comport from the pull down labeled “Connect Using:” and click OK. (e.g. Direct to Com1)
3. Set the comport and baud rate to the desired parameters. Note: Make sure these values match the values of the ZAPPORT and ZAPSPEED parameters in ZAP.ini.
4. Set the rest of the options to: Data Bits=8; Stop Bits=1; Parity=None; Flow control=XON/XOFF. Click on OK to close.
5. Select *Properties* from the File menu and select the *Settings* tab. Click on the *ASCII Setup* Button and enter 10 for the line delay and check “Wrap Lines.” all other boxes should NOT be checked. Click OK.

Press the Master Reset Button on the Evaluation Board and type a couple of carriage returns. You should receive the boards BUG monitor prompt, which should look something like the following:

```
EVSbug11 Rev 2.0
P=FFFF   Y=FFFF   X=FFFF   A=FF   B=FF   C=D0   SX.I....   S=00FF
>
```

6. Type *load t* at the bug prompt.

```
>load t <ENTER>
```

7. Choose *Transfer-> Send Text File* from the Main menu and Browse and select the appropriate S-record monitor file.

8. When Terminal is finished downloading the monitor, you need to start the monitor. For the **EVM and EVS** boards, press the **<User Reset>** switch on the board. To start the **EVB** monitor type:

```
>call 6000
```

Once the monitor is started, you can test it by pressing the space bar a couple times. You should see the monitor prompt (>) echoed each time. If the monitor prompt does not appear, go back to step one and try downloading the monitor again. Once you get the monitor prompt, exit the terminal program. You must now exit the HyperTerminal to release the comport for ZAP. You are now ready to start up ZAP and verify the installation and setup.

Using ZAP MON with Interrupts

The user must be aware that ZAP uses two of the HC11 interrupt vectors in order to single step (TOC5/XIRQ) and set breakpoints (SWI). Therefore, care must be taken when defining user interrupts. The user should NOT link in their own application vector table when using ZAP without adding the vectors used by the monitor. To avoid this problem, you need to add the vector addresses for the monitor to your vector table using the appropriate monitor map file. A map file has been provided for each of the monitors as supplied.

NOTE

If you rebuild the monitors you may have to adjust the addresses in the vector tables. The new interrupt service routine addresses may be obtained from the symbol section of the map file. (i.e., XIRQ=address of “toc5int()” and SWI=address of “sweienter ()”),

For maskable map files use mevm.map, mevs.map, mevsf1.map, mevsk4.map, mevsd3.map or mevsph8.map.

For non-maskable map files use mxevm.map, mxevs.map, mxevsf1.map, mxevk4.map ormxevd3.map.

Refer to the last segment of the map file labeled SYMBOLS to find the start addresses for the interrupt service routines _swienter and _toc5int.

Symbols:

```
__memory    0000ff13   defined in command file section .bss  used in evmcrts.o
_exit       0000fd06   defined in evmcrts.o section .text
_getchar    0000fe51   defined in ioevs.o section .text used in evmonit.o
_inchar     0000fe5e   defined in ioevs.o section .text used in evmonit.o
_irun       0000fe18   defined in run.o section .text used in evmonit.o
_main       0000fd08   defined in evmonit.o section .text  used in evmcrts.o
_monitsp    0000ff00   defined in evmonit.o section .bss used in run.o
_putchar    0000fe44   defined in ioevs.o section .text used in evmonit.o
_run        0000fe2e   defined in run.o section .text  used in evmonit.o
```

_swienter 0000fe3c defined in run.o section .text used in evmonit.o evmres.o
_toc5int 0000fe36 defined in run.o section .text used in evmonit.o evmres.o

Modify your vector table to include the addresses of _swienter and _toc5int.

The following is an example of a maskable user vector table with the addresses of the monitor interrupts hard coded.

INTERRUPT VECTORS TABLE

```
 #include <stdlib.h>
extern void _stext();            /* startup routine */
extern void recept();            /* character receive handler */

void (* const _vectab[])() = {
recept,                          /* SCI            */
NULL,                            /* SPI            */
NULL,                            /* Pulse acc input  */
NULL,                            /* Pulse acc overf  */
NULL,                            /* Timer overf     */
(void (*) (void) ) 0xFE36,       /* Output compare 5 */
NULL,                            /* Output compare 4 */
NULL,                            /* Output compare 3 */
NULL,                            /* Output compare 2 */
NULL,                            /* Output compare 1 */
NULL,                            /* Input capture 3  */
NULL,                            /* Input capture 2  */
NULL,                            /* Input capture 1  */
NULL,                            /* Real time       */
NULL,                            /* IRQ            */
NULL,                            /* XIRQ  use _toc5int          */
(void (*) (void) ) 0xFE3C,       /* SWI    use _swienter        */
NULL,                            /* illegal         */
NULL,                            /* cop fail        */
NULL,                            /* cop clock fail   */
_stext,                          /* RESET           */
};
```

The following is an example of a **non-maskable** user vector table with the addresses of the monitor interrupts hard coded.

***INTERRUPT VECTORS TABLE**

```
 #include <stdlib.h>
extern void _stext();            /* startup routine */
```

```
extern void recept();           /* character receive handler */

void (* const _vectab[])() = {
recept,                         /* SCI            */
NULL,                           /* SPI            */
NULL,                           /* Pulse acc input  */
NULL,                           /* Pulse acc overf  */
NULL,                           /* Timer overf     */
NULL,                           /* Output compare 5 */
NULL,                           /* Output compare 4 */
NULL,                           /* Output compare 3 */
NULL,                           /* Output compare 2 */
NULL,                           /* Output compare 1 */
NULL,                           /* Input capture 3  */
NULL,                           /* Input capture 2  */
NULL,                           /* Input capture 1  */
NULL,                           /* Real time       */
NULL,                           /* IRQ            */
(void (*) (void) ) 0xFE36,      /* XIRQ  use _toc5int          */
(void (*) (void) ) 0xFE3C,      /* SWI    use _swienter        */
NULL,                           /* illegal         */
NULL,                           /* cop fail        */
NULL,                           /* cop clock fail   */
_stext,                         /* RESET           */
};
```

Modifying Monitor Operation

The 68HC11 processor defines 2 types of interrupts:

- Maskable interrupts - Those that can be masked using the I bit in the CCR. (i.e. Masking an interrupt prevents it from interrupting the processor.)
- Non-maskable interrupts which can not be masked and which always cause the CPU to interrupt

Monitor Operation

ZAP uses TOC5 to allow you to trace at the assembly language level. If you use the maskable monitor and the function you wish to trace is an interrupt service routine entered as the result of an interrupt, you will find that ZAP hangs because the processor has disabled all maskable interrupts until the current interrupt is serviced. Thus TOC5 interrupts will not be acknowledged, therefore stopping ZAP from stepping at the assembly language level (i.e., ZAP will just hang). Therefore, to debug with interrupts you must use a special version of the monitor which uses the non-maskable interrupt XIRQ and you must wire TOC5 output to the XIRQ input(i.e., connect the XIRQ input to PA3 output). The monitor will use the PORTA output to drive the interrupt line, and bit 3 of PORTA will be driven by the Output Compare 5 output (TOC5). With this connection established, the monitor first sets the PA3 bit to 1 to set up the interrupt line to the state of NO INTERRUPT (this input is sensitive to a low level [zero]). Each time ZAP requests to step one assembler instruction, it calls the irun() function. This function generates a TOC5 interrupt after one machine instruction is executed.

The prebuilt monitors which support XIRQ interrupts are included with this distribution as specified in the non-maskable configuration portion of the Monitor and Hardware Configuration section. These monitors have names whose second letter is X (e.g., MXEVB.SR).

If your project requires the use of TOC5 then you must modify all the

necessary monitor files to use another timer output compare and make the appropriate change to the hardware jumper between XIRQ and the output compare. For example, if you want to use TOC4 instead of TOC5 (EVS Ax, Ex Non-maskable configuration):

1. Change toc5int and TOC5 to toc4int and TOC4 in the files evxmonit.c, evmres.s, and xruns.s

2. Modify the file xrun.s as follows:

   a) Change TOC5 to TOC4
   b) Add the line .define TOC4=1CH to the top of the file.
   c) Replace the line to clear TOC5:

      ```
      bc1r      1,TCTL1,x
      ```

      with the appropriate mask to clear TOC4:

      ```
      bc1r      4,TCTL1,x
      ```

   d) Change the function to toc4int in XRUN.S to reflect the change:

      ```
      toc4int:
      ldx     #BASE
      bset     4,TCLT1,x      ; reset XIRQ
      bset     10H, CFORC.x; clear FOC4
      ```

3. Change XIRQ jumper from PA3 to PA4.

Rebuilding the Monitors

The monitors must be built totally separate from any user applications. Special DOS batch files are supplied for this purpose. The user must be aware that ZAP uses two of the HC11 interrupt vectors in order to single step and set breakpoints. Therefore care must be taken when defining user interrupts. The user should NOT link in a vector table when using ZAP without adding the vectors used by the monitor. To avoid this problem you can either add the monitor to your project link or simply stuff the vectors you wish to use directly without using a vector table. For more information see the “Using

ZAP MON with Interrupts" section of this chapter.

DOS batch files to rebuild the monitor programs have been included in the monit11 directory. To rebuild a particular monitor:

Run the appropriate batch file for the monitor you want to build. Each build batch file invokes a COSMIC linker command file to generate the executable format for the monitors, and then generates a Motorola S-record file by invoking the CHEX utility. The building process requires access to the library file LIBM.H11 which, by default, is assumed to be in the C:\COSMIC\CX32\LIB directory. If you did not install the compiler in this default location, you will need to modify the appropriate .LKF file in order to specify the correct path for LIBM.H11.

SUMMARY OF CONFIGURATIONS						
Board	**EVB**	**EVM**	**EVS**	**EVS**	**EVS**	**EVS**
Processor	**68HC11 AX/EX**	**68HC11 AX/EX**	**68HC11 AX/EX**	**68HC11 F1**	**68HC11 K4**	**68HC11 D3**
ZAP Monitor Maskable	**MEVB.SR**	**MEVM.SR**	**MEVS.SR**	**MEVSF1.SR**	**MEVSK4.SR**	**MEVSD3.SR**
ZAP Monitor Nonmaskable	**MXEVB.SR**	**MXEVM.SR**	**MXEVS.SR**	**MXEVSF1.SR**	**MXEVSK4.SR**	**MXEVSD3.SR**
DEFAULT SETTINGS						
Monitor Link Address	**0x6000**	**0xFD00**	**0xFD00**	**0xFD00**	**0xFD00**	**0xFD00**
DUART	**MC6850**	**MC2861**	**R65C52**	**R65C52**	**R65C52**	**R65C52**
DUART Address	**0x9800**	**0xA200**	**0x7C00**	**0x7C00**	**0x7C00**	**0x7C00**
Required Interrupts (Maskable)	**SWI TOC5**	**SWI TOC5**	**SWI TOC5**	**SWI TOC5**	**SWI TOC4**	**SWI TOC5**
Required Interrupts (Non-Maskable)	**SWI TOC5 XIRQ**	**SWI TOC5 XIRQ**	**SWI TOC5 XIRQ**	**SWI TOC5 XIRQ**	**SWI TOC4 XIRQ**	**SWI TOC5 XIRQ**
Register Base Address	**0x1000**	**0x1000**	**0x1000**	**0x1000**	**0x0000**	**0x0000**
Timer Ticks Per CPU Cycle	**1**	**1**	**1**	**1**	**2**	**1**
E Clock Rate	**2MHz**	**2MHz**	**2MHz**	**2MHz**	**4MHz**	**4MHz**

ZAP Monitor Sources Files to Change Defaults

Maskable Configuration

ZAP Monitor Sources Files to Change Defaults						
Monitor Link Address	**MEVB.LKF**	**MEVM.LKF**	**MEVS.LKF**	**MEVSF1.LKF**	**MEVSK4.LKF**	**MEVSD3.LKF**
DUART Address	**IO.S**	**IOEVM.S**	**IOEVS.S**	**IOEVSF1.S**	**IOEVS.S**	**IOEVS.S**
Available Interrupts	**RUN.S**	**RUN.S**	**RUN.S**	**RUN.S**	**RUNK4.S**	**RUND3.S**
Base Register Address	**RUN.S**	**RUN.S**	**RUN.S**	**RUN.S**	**RUNK4.S**	**RUND3.S**
Timer Ticks per CPU Cycle	**_irun in RUN.S**	**_irun in RUN.S**	**_irun in RUN.S**	**_irun in RUN.S**	**_irun in RUNK4.S**	**_irun in RUND3.S**
Non Maskable Configuration						
Monitor Link Address	**MXEVB.LKF**	**MXEVM.LKF**	**MXEVS.LKF**	**MXEVSF1.LKF**	**MXEVSK4.LKF**	**MXEVSD3.LKF**
DUART Address	**IO.S**	**IOEVM.S**	**IOEVS.S**	**IOEVSF1.S**	**IOEVS.S**	**IOEVS.S**
Available Interrupts	**XRUN.S**	**XRUN.S**	**XRUN.S**	**XRUN.S**	**XRUNK4.S**	**XRUND3.S**
Base Register Address	**XRUN.S**	**XRUN.S**	**XRUN.S**	**XRUN.S**	**XRUNK4.S**	**XRUND3.S**
Timer Ticks per CPU Cycle	**_irun in XRUN.S**	**_irun in XRUN.S**	**_irun in XRUN.S**	**_irun in XRUN.S**	**_irun in XRUNK4.S**	**_irun in XRUND3.S**

Verifying Installation

Make sure the Monitor is properly configured and running (See the Monitor Setup section of this chapter). Select Cosmic Tools-> ZAP 6811 MON from the Start Menu to open the ZAP debugger or close the Terminal Window if ZAP is already open. (i.e. There should be no error messages when you start ZAP).The main ZAP window should now be active.

1. Choose Load from the file menu.
2. Select the file c:\cosmic\11\examples\demo11.h11 by clicking on it and then clicking on OK. Note: The demo example may need to be relinked for use with some processors. Compare demo11.map with the processor map if download fails.

ZAP should load the file and you should see either the source (if compiled in debug) or the disassembly of the startup routine.

3. Choose Step from the Debug menu

The source file **CRTSI.S** should appear in the source window and the first line should be highlighted.

4. Choose Registers from the Show menu to open the register window.
5. Choose *StepInst* from the debug menu or click on the single foot in the button bar to step one assembly instruction at a time. Instruction step a couple of times and you should see the program counter (PC) in the register window incrementing. The blue highlight which denotes the current assembly instruction should also move down one instruction at a time in the source and disassembly windows.
6. Choose *Go to Function Entry* from the debug menu, select main and click OK. The source window and the PC should move to the first line of the main function.

7. Click on the Source Step button in the button bar a couple of times and you should see the blue highlight for the current line of C code trace through execution one C line at a time.

If you observe the behavior described above then ZAP should be configured correctly. and the monitor is running properly. The only problem that would not appear in this verification is if you are using the maskable hardware configuration and/or the maskable version of the monitor and you try to debug an interrupt service routine in your application. This will cause the debugger to hang if an interrupt occurs.

Compiling Programs for Cross Debugging

The COSMIC ZAP debugger requires your application to be built with the COSMIC C cross compiler or its equivalent, version 4.0 or higher. To compile your C programs for full C source level cross debugging with *ZAP*, you must specify the **+debug** compiler option when you compile your program. The command line will have the form: where `<options>` are any other compiler options you wish to

```
cx6811 -v +debug <options> <file1.c> <file2.c>
<file3.c>
```

specify, `<file1.c>`, `<file2.c>` and `<file3.c>` are the C source files that will make up part of the linked executable.

When you specify **+debug**, the compiler includes all of the necessary cross debugging information required by ZAP. The debug information is kept in a separate, hidden section on the host system. You should not attempt to link the debug section explicitly.

To build an assembly file for assembly source level debug you must use the -x assembly option. The following is an example command line for the assembler:

```
ca6811 -vl -x <file.s>
OR
cx6811 -vl -ax <file.s>
```

Modifying and Saving the ZAP Environment

The installation script creates the file **zapr11.ini** in the **Windows** directory. This file is used to save configuration and initialization information for ZAP. The asterisk (*) is a comment delimiter in the initialization file. Standard C comments are also accepted.

The following environment variables under [Options] are recognized by ZAP. All of the variables under [Options] can be set using the **Setup** menu and are saved when the configuration is saved except for StartFile.

StartFile

Set this environemtn variable to have ZAP load the specified input command file automatically on startup.

DefIntFormat

This environment variable specifies the default display format for the variable browser window. This variable can also be set from the **Setup** menu (Default Int Format) and accepts binary, decimal, octal or hexadecimal.

VarFormat

The environment variable VarFormat is used to specify what information will be displayed in the variable window. VarFormat accepts the keywords “Brief”, “Standard” and “Full”. The “Brief” format lists the variable name and type. The “Standard” format lists the variable name, type and value. The “Full” format displays the variable name, type, value and address.

Tile

The Windows environment variable Tile is used to specify the layout of the sub windows on the main desktop. The Tile variable accepts the keywords Free, Cascade, Horizontal Tile and vertical Tile. These options are also found under the **Windows** menu.

Free Allows you to place and size all windows by hand

with new windows opening with the default size and on top of other windows.

Cascade Displays all windows cascaded from top left to bottom right with new windows cascaded on top as the last window.

Horizontal Tile - Displays all windows in a wider horizontal size. Each window is proportionally sized to fill the entire main the window without overlapping. New windows are added from the top left corner and push the other windows down and then up to the next column to keep the display proportional.

Vertical Tile - Displays all windows in a taller vertical size. Each window is proportionally sized to fill the entire main window without overlapping. New windows are added from the top left corner and push the other windows to the right and then down to the next row to keep the display proportional.

Path

The Path environment variable is used to define the search path to the source files. By default, ZAP searches the directory where the debugger examples are installed. Paths are searched from left to right and separated by a vertical bar (|). You can also use the PATH editor to setup the search path. See the Path Editor section of the “Using ZAP” chapter for details.

Example

Options section of ZAP Initialization file.

```
[Options]

StartFile=c:\test\commandfile.in
```

```
DefIntFormat=Decimal      /* Default Int Format */

VarFormat=BRIEF

Tile=FREE

Path=C:\TEST\|C:\CX\SRC
ZapPort= Com2
ZapSpeed=19200Delay=10
```

NOTE

To enable any changes you made to the initialization file (**.ini**) file, including setting environment variables you will need to restart ZAP.