

Appendix E:

FAQs 1
PIC Assembler/Compiler FAQs 2
ClearView Mathias FAQs 5
TechTools Design Environment FAQs 6

FAQs

This appendix contains common questions and issues that customers encounter with our products. The text is divided by product, so finding the information you need should be relatively easy.

Please consider reading this appendix before calling for technical support. This is not to say that we don't want to hear from you. However, for common questions that have a simple answer, you'll receive your answers more quickly here. By reading the questions and their answers, you may also avoid problems in the future.



PIC Assembler/Compiler FAQs

The following FAQs deal with known issues regarding TechTools CVASM16 assembler as well as popular PIC compilers.

1. When I assemble my program, I get a long list of “undefined symbol” errors for standard symbols (W, RTCC, etc).

a) This is caused by an ill-formed or missing DEVICE directive. CVASM16 does not know any of the standard PIC symbols. Instead, it relies on a symbol table, which may be different for different PICs. Without this directive, the compiler does not know whether to generate 12 or 14 bit code and has no idea what all of the predefined symbols mean.

Verify that the DEVICE directive:

- Is at the top of the program
- Occurs before the first executable line of CODE
- and Occurs before any INCLUDE files

b) The DEVICE directive is correct, but starts in the first character of the line. This causes the assembler to attempt to interpret it as a LABEL rather than a directive. Use at least one space or TAB before the DEVICE directive.

2. Why do I get “ only file registers 05h through 07h are allowed “ when I try to do something like ‘!PORTD = 0;’ ?

a) The !PORTx syntax is only valid for the register range 5 - 7. This is actually 5X code and 5-7 are the only valid addresses (there were no D or E registers in this family). We allow its use in the XX family to make it easier to port code from the 5X chips to the XX chips.

In the XX family of chips, you should use the TRISx registers to control the tri-state enables. For example:

```
setb RP0          ; switch to bank 1
mov TRISD, #00h   ; Set PORTD as lower nibble outputs, upper ; nibble inputs.
mov TRISC, #00h   ; Set PORTC as all outputs
clrb RP0          ; switch to bank 0
mov PORTC, #55h   ; Write 0x55 to PORTC
mov PORTD, #0ah   ; Set PORTD, bits 0-3 to '1010'
mov PORTD, #0fah  ; Same as prev., upper nibble does not reach pins.
```

3. My Parallax programmer works with my SPASM generated files but not your CVASM16 files. Why the change?

A) CVASM16 generates HEX files that are IDENTICAL with those generated by SPASM version 4.7. The problem is probably related to the fact that SPASM 4.7 (and CVASM16) generate HEX files which are NOT compatible with earlier versions of SPASM and SPEP. SPASM4.7 and CVASM16 generate files that are compatible with SPEP 4.7. The different formats are related to the extra data records encoded in the HEX file. You can use a /S directive on the command line to suppress these records. This would eliminate the incompatibility, but would require that you manually set the configuration data in SPEP.

E

4. Why do the instructions that involve TWO file registers seem to operate inconsistently?

a) When using any of the instructions that involve two file registers, BOTH registers must be in the CURRENT file register bank. If you need to compare file register from different banks, you should :

1. set the file register bank bits for the first register,
2. move the first file register into the W register,
3. set the file register bank bits for the second file register,
4. and then do the operation between the second register and the W register.

b) This works of course because the W register is accessed at the same offset within all banks.

Questions & Answers

5. How do I use the OPTION directive?

a) CVASM16 supports the use of the OPTION directive on 5x parts exactly like Microchip defines it. CVASM16 also allows more flexible (XX style) access to the OPTION register through mov instructions. The only difference is that OPTION is a DIRECTIVE and !OPTION is the name of the OPTION REGISTER. The following examples should clarify how to use each form:

***** VALID formats *****

```
mov !OPTION,W
```

```
mov !OPTION,#55h ; TechTools instruction (uses W)
```

```
OPTION ; Microchip instruction to move W to OPTION (obsolete)
```

***** INVALID formats *****

```
mov OPTION,w
```

```
mov OPTION,#55h
```

```
!OPTION
```

ClearView Mathias FAQs

The following FAQs deal with known issues regarding the ClearView Mathias hardware and optional modules.

1. I purchased ClearView Mathias with the modules to support the 16C74. Now, I need to develop code for a 16C5x part.

a) ClearView Mathias has two different “family” modules: one for 16C5x and another for 16Cxx devices. The 16C5x module supports the 52, 54, 55, 56, 57, and 58. The 16Cxx module supports all other 16Cxx devices.

b) There are also “member” modules, which cause the emulator to emulate a specific PIC. For 16C5x devices, there is only one member module. For 16Cxx devices, there are various member modules.

c) To support 16C5x devices, you’ll need to purchase the 16C5x family module and the 16C5x member module.

2. I’d prefer to use my Mathias from DOS, rather than Windows.

a) ClearView Mathias only works with Windows software (TDE).

TechTools Design Environment FAQs

The following FAQs deal with known issues regarding the TechTools Design Environment (TDE). Last update: 8/1/98

1. Why does my WATCH window not track my FILE REGISTERS correctly?

a) SPASM and pre-5.1 versions of CVASM16 do not encode enough information for TDE to determine which register bank the variables are supposed to be in. TDE mistakenly assumes register bank 0. The WATCH window will actually show the contents of the variable's offset within bank 0. The solution is to use TDE version 3.1 or later and CVASM16 version 5.1 or later. NOTE: CVASM16 is fully compatible with the Parallax PASM and SPASM assemblers. Also note that this issue involves ONLY the watch window, not the SPECIAL register window.

b) Some register bits are automatically cleared when certain registers are read. In particular, the SPI receive full and PSP full bits are cleared when their corresponding DATA registers are read. The emulator READS these registers each time the file register window and/or special register window are refreshed. Of course this happens when the emulator breaks, is halted, steps or animates. They are also refreshed when any register is modified. The ACT OF READING these registers to display them clears the bits of interest. You should close the special register and file register windows while stepping through this type of code. Another option is to avoid stepping through the code (run full speed to a breakpoint following the code. Another approach is to have the code itself copy the status register to another file register location, and then place a watch on that location.

c) Note that many of the chips with A/D converters default at power-up with port A configured as ANALOG INPUTS. This causes these pins to always read as 0. Mathias properly emulates this behavior. You must disable A/D before you can use PortA for Digital I/O.

d) If you have Version 5.1 or later of CVASM16, and you are working with code that was created under an earlier version, verify the ORG statements for your variables. The current assembler will accept the old relative addressing format used by the older versions, but you will not eliminate the issues mentioned in a) until you start using absolute addressing for the file registers. For example, variables declared at address A0h WERE ORGed at 20h (offset 20 in some bank) in older releases. NOW, you should ORG it at A0h (the absolute address).

2. Why don't the PORT bits change when I edit them?

- a) First note that the emulator accurately emulates the PIC chip. The PIC architecture updates the data output latch at the end of an instruction. Mathias freezes the system at the end of the instruction. The port PIN will accurately reflect the new value. HOWEVER, the PIC architecture does not read back the port pin until the beginning of the next instruction cycle. As soon as you STEP once, you will see the input value you expected.
- b) Mathias reads the PIN states; not the data output latch (exactly like the PIC processor it is emulating). This means that Mathias will read and display the same values your program would read. Remember to verify that the TRIS bits are correctly configured to allow the written data to reach the pins. Also note that your target can affect the levels on these pins.
- c) If experiencing a problem with Port A pin 4 (RA4), check the Data sheet for the device you are emulating. In most PIC devices this pin is an open drain and may require an external pullup, depending on your application.
- d) See 1.C above about A/D converters

3. My program modifies specific file registers, but the expected file register does not change in the FILE REGISTER or in the WATCH window. What could cause this?

- a) The most common cause for this is forgetting to set or clear the appropriate register bank bit(s) in the STATUS byte. This causes the processor to access the wrong file register. This is not a bug in the assembler or Mathias. The PIC architecture only encodes the file register OFFSET within its OPCODEs. This allows it to perform most operations with a single OPCODE fetch within a single instruction cycle. Of course this is what leads to the 'RISC' architecture. It also burdens the programmer with remembering which register bank a particular variable is in, and then presetting/clearing the correct register bank select bits before accessing a file register. The assembler can not predict which path your program will follow so it can not know at compile time whether these bits need to be updated. It would be terribly inefficient for the assembler to blindly set these bits before EVERY file register operation.
- b) Another possible cause is a mismatch between the processor targeted by your assembler or compiler, and the processor Mathias is configured to emulate. We have added considerable error checking where possible to reduce the likelihood that these get out-of-sync. However, it is

Questions & Answers

possible to fool Mathias. Particularly if you are using something other than CVASM16. If things are looking a little unbelievable, check the project-info settings to insure that you have selected the proper processor to emulate. Also check you compiler/assembler directives/settings to insure it is targeting the correct processor.

c) See number 1 above.

4. Why does my program JMP to the wrong location?

a) The PIC architecture uses a paged code space. Similar to the file register banking scheme, it depends on you to select the proper code page before jumping or calling into a different code page. CVASM16 provides some relief for this headache: LJMP. This instruction actually generates one or two bitset/clr instructions before the JMP instruction to automate this process. To save code and processor cycles, you can selectively use LJMPs and LCALLs to code in other code pages and use standard call and jmp instructions to code within the current page. If efficiency in code space or cycles is not an issue, you could just use the LONG versions of these all of the time and forget about which code page the target address is in. The code generated by the LJMP and LCALL instructions is no less efficient than the code you would generate yourself. It is also much easier to read, understand and maintain code written with the LONG versions.

b) See 3.b above about mis-configured device types.

5. Why does the green ‘current location’ bar disappear from my source window when I STEP sometimes?

a) This is an indication that Mathias stopped at an address that could not be referenced back to a source line. This can happen if the code is “off in the weeds”. It can also happen if your compiler linked in libraries and you happened to HALT the processor while it was executing code in one of these routines. You can open the CODE window and see the address and disassembly of where it is executing. You can continue to single step from the CODE window until you hit a source line (as evidenced by the green bar reappearing in the source window).

b) See 4.a above about landing on the wrong code page.

c) See 3.b above about mis-configured device types.

6. Why does my code seem to jump back to the start while I am stepping sometimes? or Why does my code refuse to reach a breakpoint that I am sure that it must eventually reach?

a) Verify that the watchdog timer has not be enabled by mistake. If it times-out unexpectedly, it will reset the processor. Of course Mathias accurately emulates this. It is possible that the processor continues to be reset by the watchdog before it reaches the intended breakpoint. Place a breakpoint at the start-up location. This should stabilize things by letting you know for sure that the processor was reset. Normally, one will do ALL debugging without the watchdog enabled. At the very end of the debugging cycle, you then analyze your worst case paths and decide where you should “kick the dog” to insure that it never times-out unless the processor went stupid. At this point, of course, you will do considerable testing with the watchdog enabled. In this case, set a breakpoint on the initial start-up location to insure that you reliably detect every reset. Remember that if you are running the emulator at a clock speed other than the final system clock speed that the watchdog time out values may need to be adjusted to compensate. If you have an XX Family Module, you can alternately enable the 'Break on Watchdog Timeout' option.

E

7. Why does the Watch window display only eight Bits of my 16 or 32 Bit variables?

a) TDE gets its variable type information from the symbol data in the COD file. Some compilers do not make full use of the COD format. They encode all variable types as 8bit variable types. TDE faithfully displays these as 8 bit values. TDE does not (and should not) attempt to second-guess the compiler. If the compiler lies about the data type, TDE is fooled into displaying the data in the wrong format.

TDE allows you to change the displayed length of watches. You can override the default supplied by the compiler in the 'Add Watch' dialog. You can also change the length of 1 or more after they are added. Simply highlight the watches, right-click in the Watch window and select 'Change Length'.

Questions & Answers

8. Why do some of my TRIS bits change on their own sometimes?

a) Some TRIS bits are AUTOMATICALLY re-configured when you enable/disable certain peripherals or peripheral modes. These are documented in the PIC references for each particular peripheral. Other peripherals REQUIRE specific TRIS settings to function properly, but do NOT automatically force them. Mathias faithfully duplicates the real PIC device in both cases.

9. Why do I get an “Invalid Line” message when I try to set a breakpoint?

a) You have double-clicked on a source line that does not generate executable code and, therefore, TDE complains that it can not place a breakpoint there.

b) If you have been editing the file and then attempt to set a breakpoint before rebuilding, TDE will be using symbol and line information from the last build. The line you are attempting to set a breakpoint on may NOW be a valid executable line, but if it WAS invalid, TDE complains. A fresh build brings the source, object and symbols in sync.

10. I’m running on an older 80286-based PC. Can I run the TDE software on my system?

a) TDE will run under Windows 3.1, Win 95 and Win NT. Windows 3.1 will work on most 80286-based computers. However, we strongly recommend that TDE be run under Windows 95 or NT, which requires at least an 80386-based PC.

11. TDE reports the following error: “Mathias not found”

a) The serial port is set incorrectly, or

b) Two devices are trying to use the same serial port, or

c) The serial port is already open by another program, or

d) There is an interrupt conflict on the desired serial port.

e) Check the serial port and make sure that it’s functioning. If the serial port appears to be okay, but TDE cannot communicate with Mathias, then you will probably need to contact our tech support department.

12. TDE reports the following error: “Communication error”

- a) See remedies for #11.

13. Why doesn't my Data Breakpoint work when the qualified conditions seem to be valid?

- a) Data Breakpoints operate on Data Read/Write cycles generated by your program. Peripherals do not use read/write cycles to update registers. Therefore data breakpointing circuits will not sense register changes initiated by peripherals. For example; if you setup a data breakpoint for the value of 55h in TIMER0, a break will not be generated when TIMER0 increments from 54h to 55h. It WILL break however, if your program reads TIMER0 when its value is 55h.