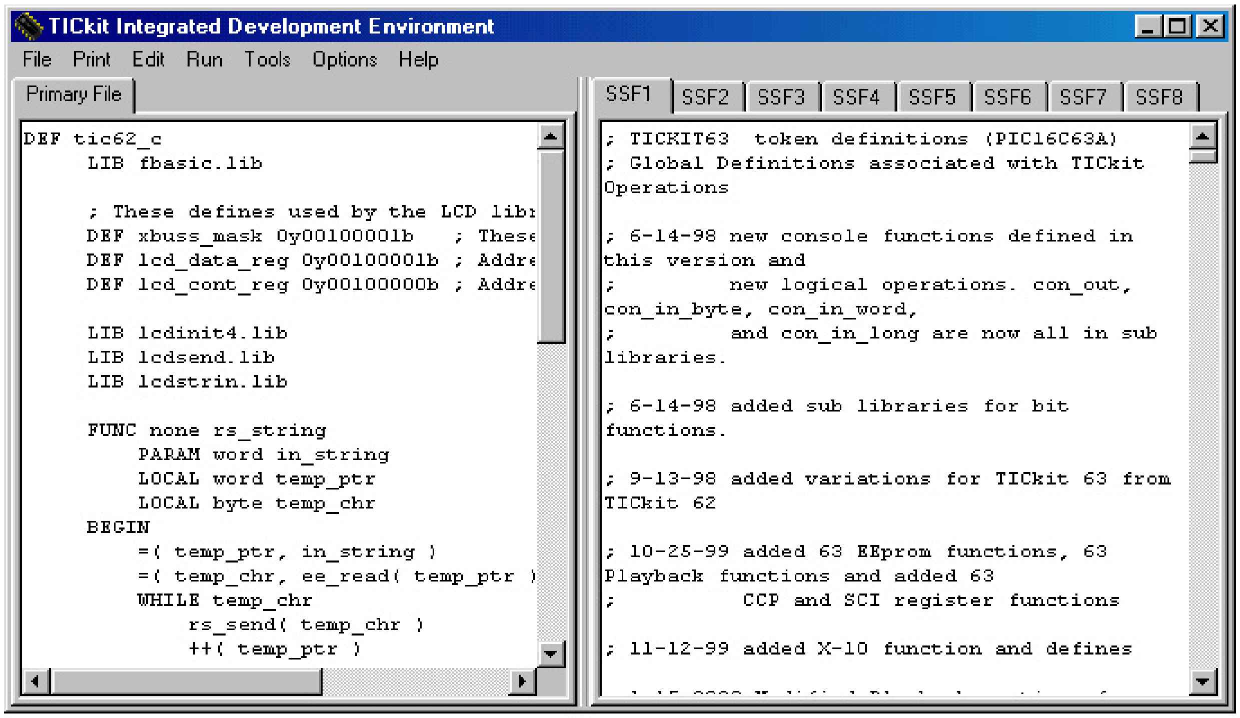
6 The TICkit IDE Program

6.1 What is the TICkit IDE?

The TICkit IDE is an Integrated Development Environment. This means that it is a single program with all the tools necessary to write, compile, and debug programs for TICkit devices. Using a PC running Windows 95 or later operating systems, you use the TICkit IDE to develop your program for the TICkit device. Then, using the download functions of the IDE, you place the developed program into the TICkit. At this point the TICkit can execute the program without any need of the PC. However, you can use the IDE's powerful debugging capabilities to monitor program operation, inspect the values of variables, and use the IDE console as an output device for the TICkit. Once you are satisfied with the operation of the TICkit program, the TICkit can be removed from the download cable and placed into operation free of the PC. The program is remembered by the TICkit even when power is removed, so the TICkit is now operating as a completely independent small computer.

6.2 The Main TICkit IDE Edit Window



```
DEF tic62_c
     LIB fbasic.lib

     ; These defines used by the LCD libr
     DEF xbuss_mask 0y00100001b   ; These
     DEF lcd_data_reg 0y00100001b ; Addre
     DEF lcd_cont_reg 0y00100000b ; Addre

     LIB lcdinit4.lib
     LIB lcdsend.lib
     LIB lcdstrin.lib

     FUNC none rs_string
         PARAM word in_string
         LOCAL word temp_ptr
         LOCAL byte temp_chr
     BEGIN
         =( temp_ptr, in_string )
         =( temp_chr, ee_read( temp_ptr )
         WHILE temp_chr
             rs_send( temp_chr )
             ++( temp_ptr )
```

```
; TICKIT63  token definitions (PIC16C63A)
; Global Definitions associated with TICkit
Operations

; 6-14-98 new console functions defined in
this version and
;         new logical operations. con_out,
con_in_byte, con_in_word,
;         and con_in_long are now all in sub
libraries.

; 6-14-98 added sub libraries for bit
functions.

; 9-13-98 added variations for TICkit 63 from
TICkit 62

; 10-25-99 added 63 EEprom functions, 63
Playback functions and added 63
;          CCP and SCI register functions

; 11-12-99 added X-10 function and defines
```

The first window you see after starting the TICkit IDE is refered to as the "edit window" and is pictured above. This window is sizeable and contains two distinct areas that can be resized as well. The left side is refered to as the "Primary Source" edit area. This area is used to prepare the main file of the program. Typically, this file contains the main body of the program including the function "main" which is the starting point for FBasic programs. The right side of the edit window is refered to as the "Secondary Source" edit area. This area has 8 notebook tabs for editing up to 8 files at once. Many programs do not require secondary source files at all (SSFs) but often programs can be developed better or maintained easier when a program is broken into different files. For example, you may have a large array of bytes which are samples for the audio playback to say "hello". If you place this array initialization into a secondary source file named "hello.inc", not only will your primary file be much smaller and easier to view, any new program can refer to this data simply by using the following line in the top part of the program:

```
INCLUDE hello.inc
```

You may also wish to put functions for specific peripherals into secondary source files called library files. For example, the release disk contains a file called "ltc1298.lib". This library file has prewritten functions for controlling a Linear

Technologies 1298 12bit A/D converter. Any program needing to control such a device and simply add the following line to the top of the program and refer to the functions later:

```
LIBRARY ltc1298.lib
```

You may also want to use a secondary source edit area for looking at other programs. Since the primary and secondary edit areas are side by side, you can page through both edit areas comparing source code line by line to find and examine differences.

6.3 The Main Edit Window Menu

The menu selections for the maid edit window provide all the options necessary for editing and maintaining files for a program. Also, the menu contains access to other tools, and areas of the program necessary for setting up the IDE, compiling programs, and debugging programs, and using utility programs.

The first three menu items on this window (File, Print, Edit) are used to prepare your program. The Run menu item is used to compile and download programs into a TICkit device. The Options menu selection controls the behavior of the IDE. The Help selection provides access to the complete user manual for the TICkit as well as revision information and release information.

The **File** menu item pulls down selections used to create, open, save or close files in the IDE. Primary program files determine the file names of the token and symbol files. The selections are:

1. **New Primary**: Creates a skeleton FBasic program in the Primary edit area. Use when writing a program from scratch.
2. **Open Primary:** Is used to open an existing program file. Use this to view or modify a program.
3. **Save Primary:** Saves the program currently in the Primary edit area to the file.
4. **Save Primary As:** Saves the program currently in the Primary edit area to a different file name.
5. **Close Primary:** Clears the Primary edit area and discards any changes.
6. **New Secondary:** Creates a skeleton of an FBasic library program in the selected Secondary Source area.
7. **Open Secondary:** Is used to open an existing program file in the selected Secondary Source edit area.
8. **Save Secondary:** Saves the program currently in the selected Secondary Source edit area to the file.
9. **Save Secondary As:** Save the program currently in the selected Secondary Source edit area to a different file.
10. **Close Secondary:** Clears the selected Secondary Source edit area and discards any changes.
11. **Save All:** Saves all edit areas that have been modified to their respective files. If an edit area is new and not associated with any file name, the user will be prompted for a file name.
12. **Close All:** Closes all edit areas, and discards all changes.
13. **Exit:** Closes all edit areas and exits the TICkit IDE.

The **Print** menu item pulls down selections used to print one or all edit areas and to configure the printer. The selections are:

1. **Print Primary:** Prints the text of the Primary edit area.
2. **Print Secondary:** Prints the text of the selected Secondary Source area.
3. **Print All:** Prints the text of both the Primary and all Secondary edit areas.
4. **Printer Setup:** Selects the printer to use and allows selection of printer options.

The **Edit** menu item pulls down selections used to manipulate the text of an edit area. The selections are:

1. **Undo:** Restores the edit area to the condition before the last text change.
2. **Cut:** Cuts text from the selected edit area and places it in the clipboard.
3. **Copy:** Copies text from the selected edit area and places it in the clipboard.

4. **Paste:** Places the text in the clipboard into the selected edit area at the point of the cursor.
5. **Delete:** Removes the text from the selected edit area.
6. **Goto Line:** Displays the position of the cursor in the selected edit area and optionally moves the cursor to another line.
7. **Find:** Searches the selected edit area for a pattern of text.
8. **Replace:** Searches the selected edit area for a pattern of text, and replaces the text with the specified replacement pattern. **Replace All,** will interactively prompt for replacement, or not.

The **Run** menu item pulls down selections used to run your program on a TICkit device. The selections are:

1. **Compile and Debug:** Compiles the program in the Primary edit area, and, provided no errors are found, starts a debug dialog. If errors are revealed, they are reported, and you are returned to the main IDE edit window.
2. **Debug Only:** Starts the debug dialog window used for downloading and debugging TICkit programs. Use this selection when working with prewritten programs. Use the Download From File option of the debug dialog for distributed token files.
3. **Compile Only:** Compiles the program in the Primary edit area. Any errors are reported and the user is always returned to the main edit window.

The **Tools** menu item pulls down selections of utility modules required for special features of the TICkit. The Selections area:

1. **WAV to INC Conversion:** This utility converts a standard WAV format audio file into data arrays in one or more secondary source edit areas. Typically, these arrays are saved in INC files (include) and are used with the audio playback functions of the TICkit. NOTE: This utility can only convert one channel, 8 bit, and PCM files sampled at 11025 or 8000 samples per second.
2. **Serial Terminal:** Not implemented in this release.

The **Options** menu item opens a dialog for setting up the COM port (serial port) and protocol conventions of the TICkit you are using. Normally, the protocol is determined automatically, however older token files or older TICkits may require manual protocol settings. The Revision prompting and Automatic Version Control features of the IDE are not implemented in this release.

The **Help** menu contains selections of documents meant to be helpful. What you are reading is one of them. The TICkit Manual, DOS Tools Manual, and Release notes are all Adobe's PDF format documents. This allows you to print them out in a formatted maner, add sticky notes to them for your reference, and search the documents for words or phrases. Adobe's PDF viewer is freely distributed and a copy of the program is available on the TICkit release disk if you need to install it on your computer. Because the Manual is in electronic form, you can actually highlight sample code, copy it to the clipboard, and paste it into the TICkit IDE edit areas to experiment with the example code. The help selections are:

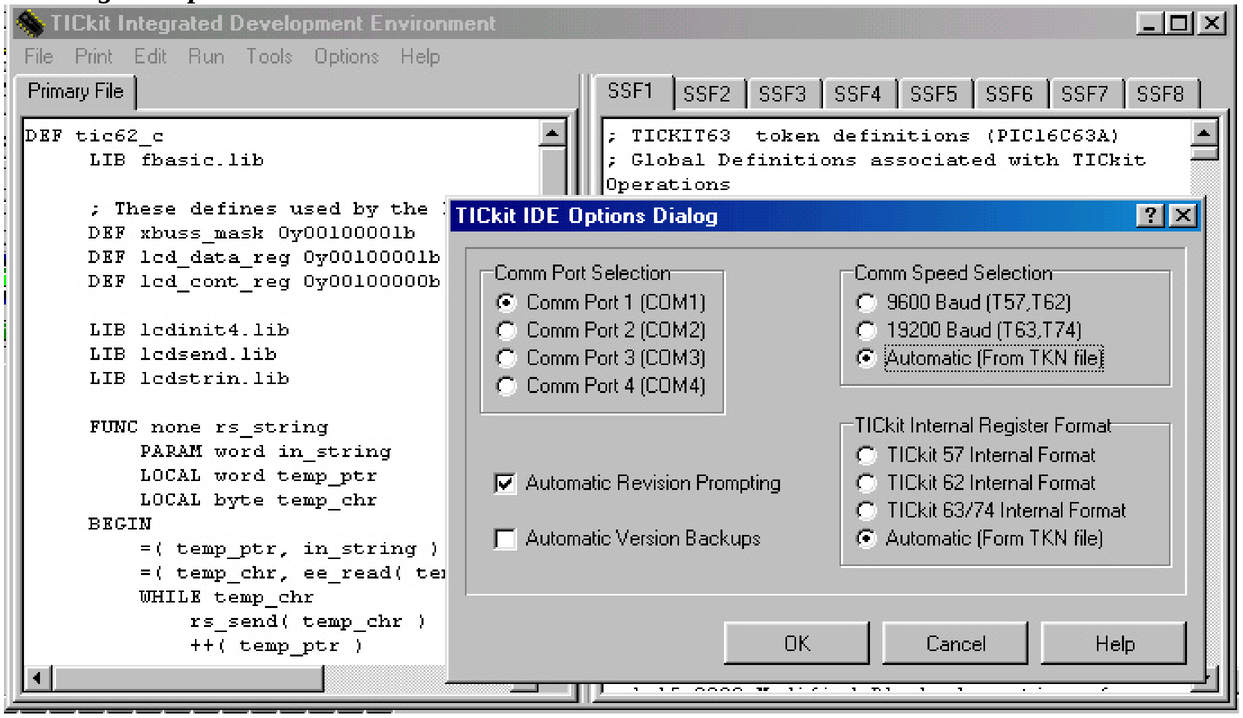
1. **About:** Displays information about this release of the TICkit IDE.
2. **TICkit Manual:** Provides information about the TICkit IDE, the TICkit hardware, the FBasic language, etc.
3. **DOS Tools Manual:** Provides information about the older DOS development tools.
4. **Release Notes:** Provides information specific to a release. May include errata, or new feature documentation.
5. **Intall Acrobat 4.0** Provides instruction for installating Adobe's Acrobat reader.
6. **Using This Program:** An abreviated explination of the edit window menu options

6.4 Using the Pop-Up Menus

In addition to the main edit window menus at the top of the window, you may click on the right button of your mouse when the mouse pointer is over an edit area to pup up another menu. This shorter menu provides options specific to the edit area and may be more intuative to use.

Note: When use Find or Replace or Goto Line menu functions, the position of the cursor is improtant. You must make the edit area active by clicking on it and place the cursor at the beginning of the search area.

6.5 Setting the Options


TICkit Integrated Development Environment
File Print Edit Run Tools Options Help
Primary File
SSF1 SSF2 SSF3 SSF4 SSF5 SSF6 SSF7 SSF8
TICkit IDE Options Dialog
OK
Cancel
Help

```
DEF tic62_c
     LIB fbasic.lib

     ; These defines used by the
     DEF xbuss_mask 0y00100001b
     DEF lcd_data_reg 0y00100001b
     DEF lcd_cont_reg 0y00100000b

     LIB lcdinit4.lib
     LIB lcdsend.lib
     LIB lcdstrin.lib

     FUNC none rs_string
         PARAM word in_string
         LOCAL word temp_ptr
         LOCAL byte temp_chr
     BEGIN
         =( temp_ptr, in_string )
         =( temp_chr, ee_read( te
         WHILE temp_chr
             rs_send( temp_chr )
             ++( temp_ptr )
```

```
; TICKIT63  token definitions (PIC16C63A)
; Global Definitions associated with TICkit
Operations
```

Comm Port Selection
- [✓ Comm Port 1 (COM1)]
- Comm Port 2 (COM2)
- Comm Port 3 (COM3)
- Comm Port 4 (COM4)

Comm Speed Selection
- 9600 Baud (T57,T62)
- 19200 Baud (T63,T74)
- [✓ Automatic (From TKN file)]

[✓ Automatic Revision Prompting]

Automatic Version Backups

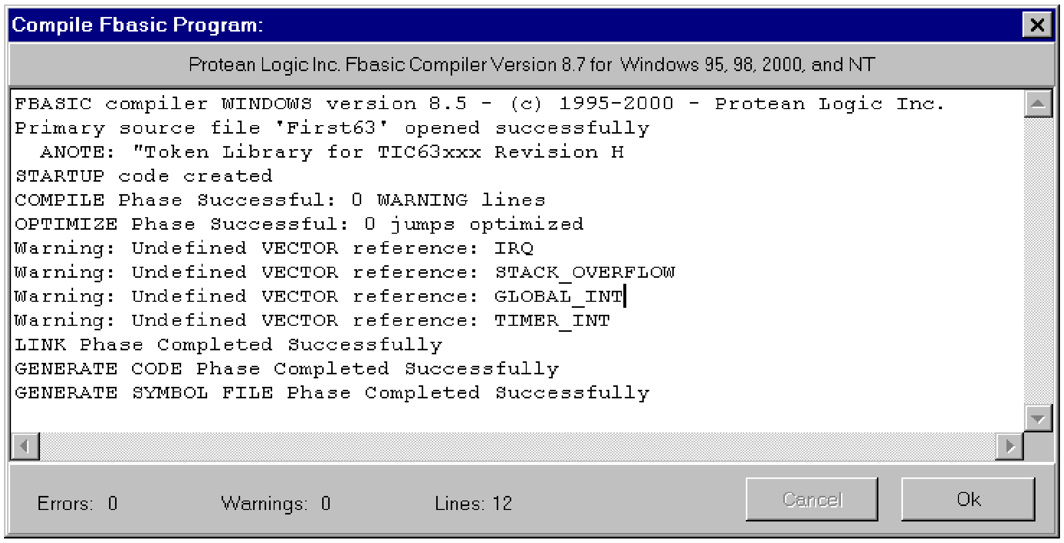
TICkit Internal Register Format
- TICkit 57 Internal Format
- TICkit 62 Internal Format
- TICkit 63/74 Internal Format
- [✓ Automatic (Form TKN file)]

Shown above is the options window. This window is used to designate the serial port that the TICkit device connects to. It also can be used to force a protocol speed or register format. Normally, the speed and format designation is specified in the token file for a program, but you may receive a token file for a project that was compiled under an older FBasic compiler. When this happens, you will need to set the protocol parameters manually.

You will also notice some check boxes in the options window. These are features not yet implemented in the TICkit IDE, but will eventually aid you in commenting your program and breaking versions for easy version roll-back.

6.6 Compiling and Debugging

Once your program is written, use the RUN menu item to compiled and/or debug the program. The Compile window is shown below. This the area for the compiler to report errors. When the compilation is complete, note any errors so you can fix them, then click Okay to return to the edit window or to enter the Debug Dialog window.


Compile Fbasic Program:
Protean Logic Inc. Fbasic Compiler Version 8.7 for Windows 95, 98, 2000, and NT
Errors: 0
Warnings: 0
Lines: 12
Cancel
Ok

```
FBASIC compiler WINDOWS version 8.5 - (c) 1995-2000 - Protean Logic Inc.
Primary source file 'First63' opened successfully
  ANOTE: "Token Library for TIC63xxx Revision H
STARTUP code created
COMPILE Phase Successful: 0 WARNING lines
OPTIMIZE Phase Successful: 0 jumps optimized
Warning: Undefined VECTOR reference: IRQ
Warning: Undefined VECTOR reference: STACK_OVERFLOW
Warning: Undefined VECTOR reference: GLOBAL_INT
Warning: Undefined VECTOR reference: TIMER_INT
LINK Phase Completed Successfully
GENERATE CODE Phase Completed Successfully
GENERATE SYMBOL FILE Phase Completed Successfully
```

Once your program has compiled correctly, you can use the Debug function to download the program into a TICkit device. The next chapter deals with the details of downloading a program and using the debug window to test your program.

6.7 The WAV to INC conversion utility

The TICkit 63 and 74 have audio playback capabilities. To use audio playback, you will need to create an array of bytes initialized with all the audio samples of the playback fragment. The TICkit IDE has a utility for converting standard audio WAV format files into FBasic source code as initialized arrays of bytes. First you need a WAV file to convert you can use any of a number of audio recording programs to generate these files. The most common is the "Sound Recorder" program of Windows 95/98/2000. This program is usually in the Entertainment section of Accessories under Program Files. If you cannot locate the program on your system, use the Windows Setup Tab of Add/Remove Programs under Control Panel under Settings to make sure the Sound Recorder program was installed. Record your audio sample, or save an existing sample as a single channel, 8bit, PCM format file. Sampling at 11025 or 8000 samples per second yeild the best playback results.

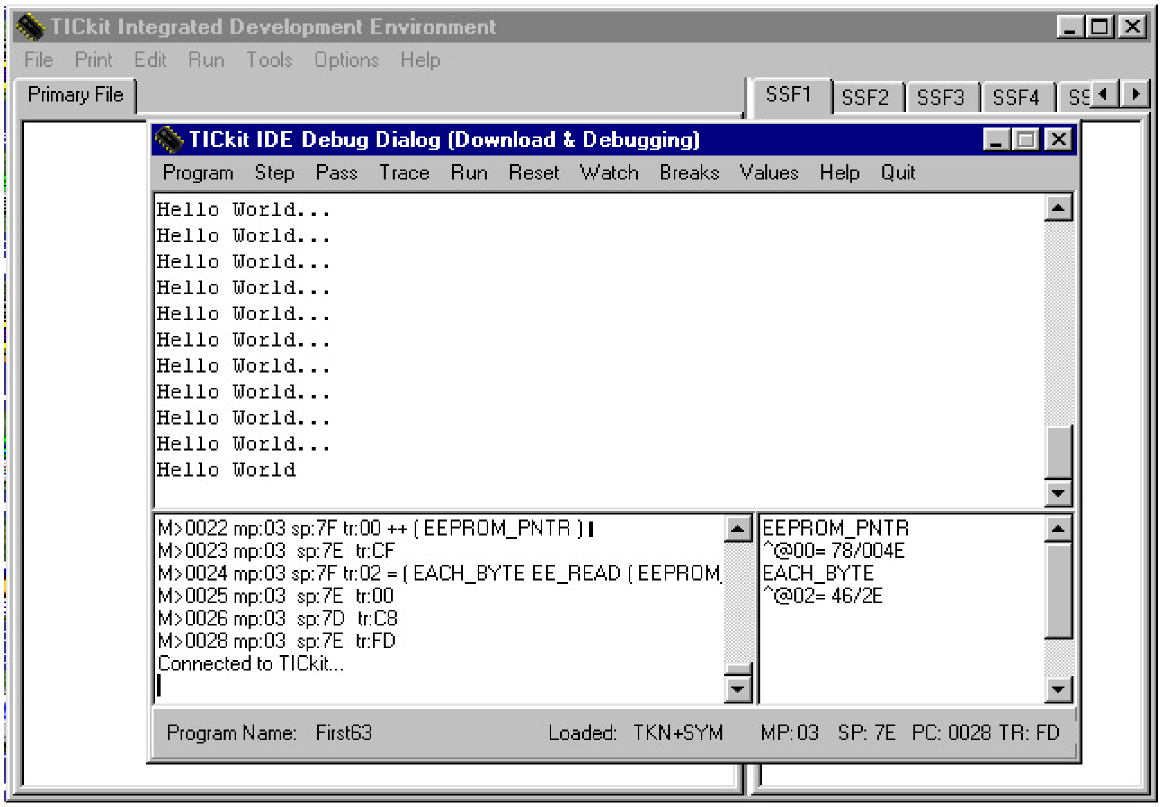
Once your WAV file has been generated, Click the WAV to INC selection under Tools in the TICkit IDE. Select the WAV file you want to convert then specify the name of the array to contain the data and a symbolic name to be used for the size of this array. Lastly, specify a secondary source edit area to receive the source code. Click convert and the program source is created. The secondary source can then be saved as an include file or treated like any other FBasic source code.

6.8 TICkit IDE Program Distribution

Protean Logic Inc. makes the TICkit IDE available for immediate download from the protean web site. Also, you may distribute copies of the program to others who will need to download token files or who want to evaluate the TICkit IDE. The TICkit IDE is NOT free or public domain however. Once you or any person you distribute the program to, starts developing software for the TICkit and utilizes the TICkit IDE for something beyond evaluation of program compilation and download, the TICkit IDE program should be registered. This provides the revenue necessary for Protean Logic to continue to improve the IDE and assures that the TICkit product will be supported in the future.

7 The TICkit IDE Debug Window

7.1 The Elements of Debugging and the Debug Window


TICkit Integrated Development Environment
File Print Edit Run Tools Options Help
Primary File
SSF1 SSF2 SSF3 SSF4
TICkit IDE Debug Dialog (Download & Debugging)
Program Step Pass Trace Run Reset Watch Breaks Values Help Quit

```
Hello World...
Hello World...
Hello World...
Hello World...
Hello World...
Hello World...
Hello World...
Hello World...
Hello World...
Hello World...
Hello World
```

```
M>0022 mp:03 sp:7F tr:00 ++ ( EEPROM_PNTR )
M>0023 mp:03  sp:7E  tr:CF
M>0024 mp:03 sp:7F tr:02 = ( EACH_BYTE EE_READ ( EEPROM
M>0025 mp:03  sp:7E  tr:00
M>0026 mp:03  sp:7D  tr:C8
M>0028 mp:03  sp:7E  tr:FD
Connected to TICkit...
```

```
EEPROM_PNTR
^@00= 78/004E
EACH_BYTE
^@02= 46/2E
```

Program Name: First63 Loaded: TKN+SYM MP:03 SP: 7E PC: 0028 TR: FD

The Debug window consists of three areas. The top area is the Console area and is used by TICkit programs as an output device. The bottom left area is the command box or area and provides a history of commands as well as displays token and source line information as the program progresses. The command box is also used to display status and error messages about the debug process.

NOTE: The protocol used to communicate between the PC and the TICkit is relatively complex and is time sensitive. When the PC running the debug program is busy doing other things, or if you enter the debug window while the TICkit is running, you may likely see "Illegal Protocol Character" messages. This is normal and presents no problem as the Debug program will re-synchronize with the TICkit automatically.

The box area on the bottom left is the watch point area and is used to display information about variables inside the TICkit as the program progresses.

Debugging involves various methods of using the information provided by the debug window to determine if a TICkit program is running as intended. If the program is not behaving as intended, the debug window can be used in various ways to determine exactly where the errors in logic are within a program.

Debugging methods include simple console I/O methods where the program sends messages to the debug console when it reaches crucial points in execution. The console might also be used as a means of altering program execution or pausing execution while the user is prompted for a valule.

More sophisticated debug methods can use watch points and break points to cause the program to execute slower and constantly update key status information as the program runs. The user may also wish to step through sections of a program one function, source line, or token at a time to see how the program operates. At any point, the user can examine memory or variable contents and alter the contents if necessary.

So, debugging consists of a lot more than just downloading and executing a program. The time it takes to make even complex programs function properly can be greatly speeded by clever use of the debugging capabilities of the debug window.

7.2 The Debug Window Menu Selections

The **Program** menu item pulls down selections used to download a program to a TICkit or to compare the contents of a TICkit with a program to verify its version and accuracy. The selections are:

1. **Download Current** Downloads the program's tokens currently loaded in the IDE into the TICkit device.
2. **Compare Current:** Compares the program's tokens currently loaded in the IDE with the TICkit device's EEprom contents
3. **Download from File:** Asks for a program token file, loads the tokens into the IDE, then downloads the tokens into the TICkit Device. Use this download option when you are using a TICkit with prewritten code and you only have the program's token file distributed to you.
4. **Compare with File:** Asks for a program token file, loads the tokens into the IDE, then compares the tokens with those inside the TICkit device's EEprom.

The **Step** menu item instructs the TICkit to execute exactly one line from your FBasic program.

The **Pass** menu item instructs the TICkit to execute one line from your FBasic program. If the current line is a call to another FBasic function, the entire function is executed.

The **Trace** menu item instructs the TICkit to execute exactly one token of your FBasic program. There are usually many tokens in a single FBasic program line, so this low level of execution shows exactly how the stack and memory usage change as a single program line progresses.

The **Run** menu item pulls down selections used to start or manage program execution within the TICkit device. The selections are:

1. **Execute**: Execute the program in the TICkit at full speed. Suspend debug protocol as necessary.
2. **Monitor Verbose:** Execute the program in the TICkit, but maintain the debug protocol. All source lines and watch point information is displayed as the program progresses.
3. **Monitor Silent:** Execute the program in the TICkit, but maintain the debug protocol. Source lines and watch point information is NOT displayed when monitoring silently.
4. **Monitor Stop (Zap):** Halts program execution if the TICkit is running in a monitor mode. Execution can resume from the point of the halt.

The **Reset** menu item instructs the TICkit device to internally reset. If you have a deluxe download cable connected, this menu item will also externally reset the TICkit device in the same way that pressing the reset button will. If you have the standard download cable, this menu item will have no effect unless the protocol is connected.

The **Watch** menu item opens a window to maintain the selection of watch points. Watch points are references to variables used by the TICkit's FBasic program. When a watch point has been set, every Step, Pass, Trace, or Monitor command will display the variables contents in the watch point area.

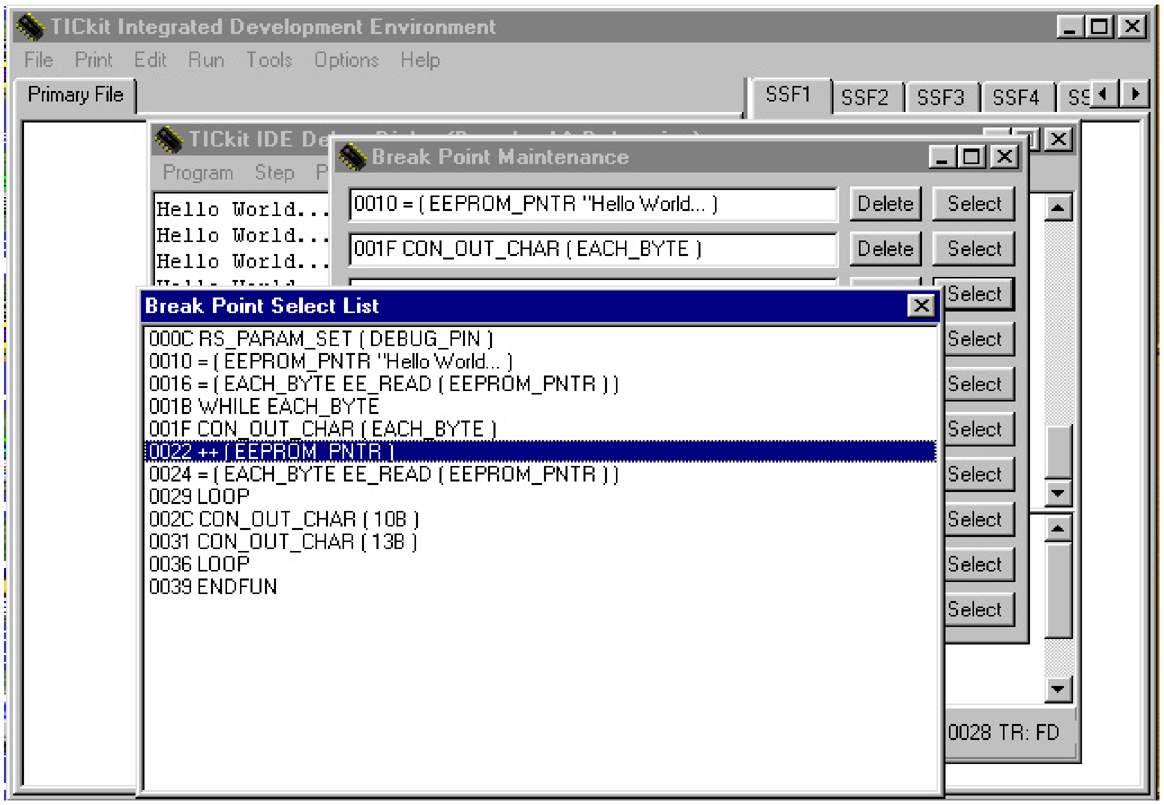
The **Breaks** menu item opens a window to maintain break points. Break points are references to places in the TICkit's FBasic program. When the TICkit is running in Monitor mode, execution will stop when a break point is encountered.

The **Values** menu item opens a window that allows individual variables or memory locations to be viewed and changed inside the TICkit device.

The **Help** menu item displays an abreviated explanation of the debug menu items.

The **Quit** menu item closes the debug connection and debug dialog window. If a program is contained inside the TICkit, it will begin executing at full speed once the debug connection is closed.

7.3 The Break Point Selection Window



```
Hello World...
Hello World...
Hello World...
```

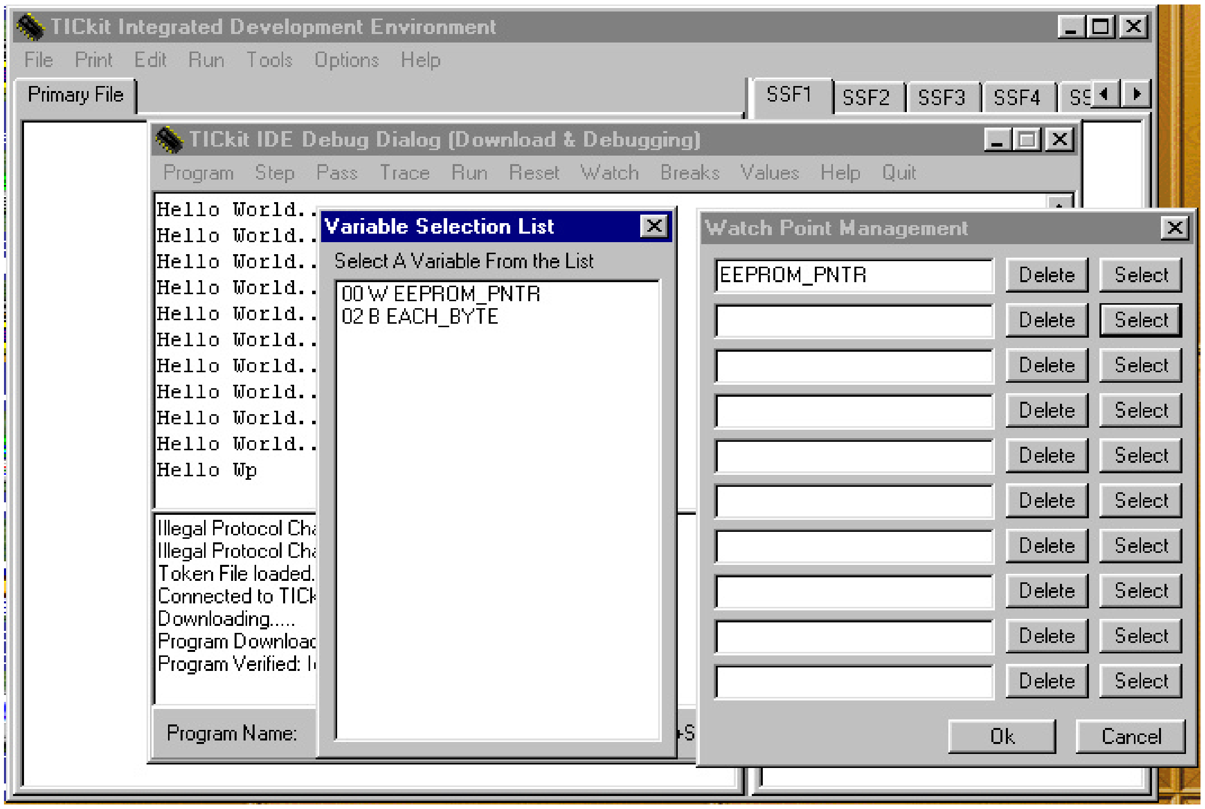
```
0010 = ( EEPROM_PNTR "Hello World... )
001F CON_OUT_CHAR ( EACH_BYTE )
```

```
000C RS_PARAM_SET ( DEBUG_PIN )
0010 = ( EEPROM_PNTR "Hello World... )
0016 = ( EACH_BYTE EE_READ ( EEPROM_PNTR ) )
001B WHILE EACH_BYTE
001F CON_OUT_CHAR ( EACH_BYTE )
0022 ++ ( EEPROM_PNTR )
0024 = ( EACH_BYTE EE_READ ( EEPROM_PNTR ) )
0029 LOOP
002C CON_OUT_CHAR ( 10B )
0031 CON_OUT_CHAR ( 13B )
0036 LOOP
0039 ENDFUN
```

Use the break point selection window to pick an execution stop point from the lines of your program . Once set, a break piont will interrupt any monitored execution of the TICkit. The selection window is shown above. Use the up and down arrows to move the lines of your program accross the selection window, then click on the line you want to be the break point. When executing, the TICkit will stop immediately before executing the line selected as a break point. The TICkit IDE allows up to 10 breakpoints at a time.

NOTE: A break point will not suspend execution if the TICkit has suspended the debug link. Therefor, when you want to execute up to a break point, choose the MONITOR VERBOSE or MONITOR SILENT run methods.

7.4 The Watch Point Selection Window


TICkit Integrated Development Environment
File Print Edit Run Tools Options Help
Primary File
SSF1 SSF2 SSF3 SSF4
TICkit IDE Debug Dialog (Download & Debugging)
Program Step Pass Trace Run Reset Watch Breaks Values Help Quit
Variable Selection List
Select A Variable From the List
Watch Point Management
Delete
Select
Ok
Cancel
Program Name:

```
Hello World..
Hello World..
Hello World..
Hello World..
Hello World..
Hello World..
Hello World..
Hello World..
Hello World..
Hello World..
Hello Wp
```

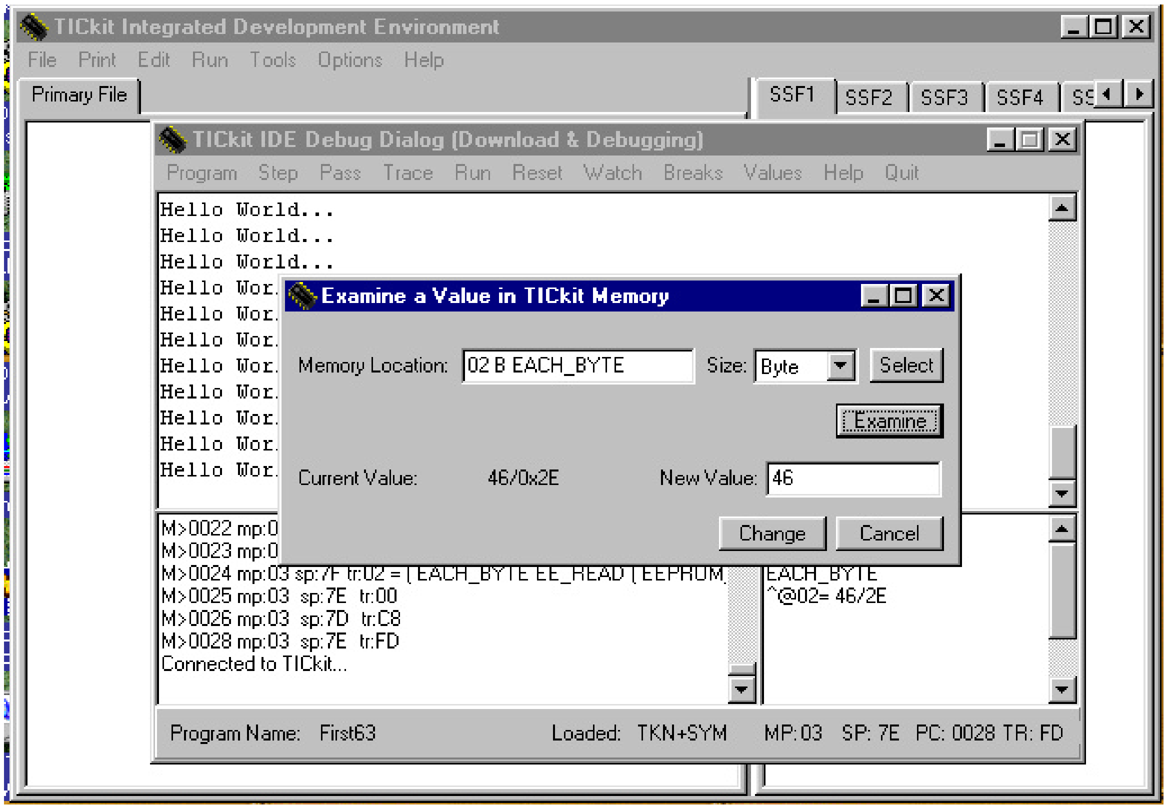
```
Illegal Protocol Ch
Illegal Protocol Ch
Token File loaded.
Connected to TICk
Downloading.....
Program Download
Program Verified: I
```

```
00 W EEPROM_PNTR
02 B EACH_BYTE
```

EEPROM_PNTR

The window shown above is used to select watch points. Watch points are simply named variables that display in the watch point area of the debug dialog. This release of the TICkit IDE only allows watch points on Global Variables. Click one of the SELECT buttons to pull up the Variable Selection List. The Variable Selection List will display all the Global variables in a program. Click on the variable to make it a watch point. Up to 10 watch points are allowed at one time in the TICkit IDE.

7.5 Using the Value Examine and Modify Window


TICkit Integrated Development Environment
File Print Edit Run Tools Options Help
Primary File
SSF1 SSF2 SSF3 SSF4
TICkit IDE Debug Dialog (Download & Debugging)
Program Step Pass Trace Run Reset Watch Breaks Values Help Quit
Examine a Value in TICkit Memory
Select
Examine
Change
Cancel

```
Hello World...
Hello World...
Hello World...
Hello Wor.
Hello Wor.
Hello Wor.
Hello Wor.
Hello Wor.
Hello Wor.
Hello Wor.
Hello Wor.
```

Memory Location: 02 B EACH_BYTE Size: Byte
Current Value: 46/0x2E New Value: 46

```
M>0022 mp:0
M>0023 mp:0
M>0024 mp:03 sp:7F tr:02 = [ EACH_BYTE EE_READ [ EEPROM
M>0025 mp:03  sp:7E  tr:00
M>0026 mp:03  sp:7D  tr:C8
M>0028 mp:03  sp:7E  tr:FD
Connected to TICkit...
```

```
EACH_BYTE
^@02= 46/2E
```

Program Name: First63 Loaded: TKN+SYM MP:03 SP: 7E PC: 0028 TR: FD

The window shown above is used to examine and modify variables or memory locations. Type in the name, decimal address, or hexidecimal address of the variable or memory you want to examine. If you type a name or use the SELECT button to select a variable, the SIZE field will automatically adjust to the variable's size. If using a memory address, use the SIZE list box to select the appropriate variable size. Click the EXAMINE button to read the specified location and size from the TICkit Device's memory. The value will display in the field labeled, "Current Value". If you want to modify the value, type the new value in decimal, hexidecimal or floating point format in the field labeled, "New Value" then press the CHANGE button. This dialog is modeless, so you can keep the window active while the program executes and while you do other things in the TICkit IDE. This allows you to repeatedly examine the same memory without having to re-enter the location or size.

This window only examines RAM address locations, not EEprom locations. To examine EEprom locations, a small FBasic program is required. This limitation was imposed to prevent easy copying of commercial trade secret applications.

7.6 Debug Strategies

There are many different strategies for testing and debugging programs. Simple rules to follow are:

1. Do not change very many things in the program at once.
2. Make copies of your program whenever operational milestones are acheived.
3. Use the tools to diagnose the problem. (In other words, after looking at source code with no clear indication of a problem, try to discover clues as the program executes).
4. Monitor Stack Usage.
5. Check to see if any warnings generated by the compiler might be related to a problem being observed.
6. Re-use code that is known to work when possible.
7. Use symbolic names for values. This enables easy find and replace modifications, and prevents accidental scews of assumptions.

You will likely come up with some more rules of your own. The TICkit devices can support very large programs. Programs which can easily exceed a programmers ability to keep all the information inside his or her head. Therefore, use the debugging tools and heavily document code when programs start getting larger. Also, use the encapsulation and blocking capabilities of FBasic to keep code blocks small and concise in function. By using functions and LOCAL values, accidental re-use of variables is completely eliminated and your functions can be tested in smaller chunks.