TMS320F241/3 On-chip Flash Programming Using XDS510 Emulator

This document explains how to program the TMS320F241/3 on-chip flash memory using a PC host and an XDS510 or XDS510PP emulator. The programming environment consists of a XDS510 based loader which runs on a PC compatible host and flash programming algorithms which run on the target DSP.
The programming of 'F241 and 'F243 is identical. Both are referred to by the name TMS320F243 in this document.
Environment : IBM PC compatible with Microsoft Windows 95.

TI tools : XDS510 or XDS510PP emulator and the software described in this document.

Program type : DOS window command.

The files described in this document can be downloaded from the TI web page. Note that the executable files for the loader differ depending on the emulator used—XDS510 or XDS510PP. Be sure to download the appropriate files for your setup. The first two chapters of the document provide all the information needed to get started. The last chapter is provided for those users who wish to customize the programming utility.

TI Web Page URL *http://www.ti.com*

IMPORTANT CONSIDERATIONS FOR VERSION 2.1

- The TMS320F243 instruction rate is expected to be 20 MIPS. With the internal PLL multiplication fixed at x4, an input clock frequency of 5 Mhz is required. If a different clock rate is used, then the algorithms must be adjusted accordingly. See Chapter 3 of this document for more detail.
- A directory structure is required by the included batch files and linker command files. Be sure to restore the directory structure after downloading the files. For PKUNZIP use –d for this purpose.

1.0 Required Files

The following files are **required** for the programming utility. These files are available for download from the TI web page. Note, the directory structure shown here is **required** for proper operation.

PC Executables
- PRG2XXPP.EXE -Windows 95 executable file that invokes the loader.
- EMURSTPP.EXE -Emulator reset utility.
- COMPOSER.EXE -Utility that translates BOARD.CFG to a binary conditioned format.
- EMU2XXDM.DLL -'C2XX specific Dynamically linked library (DLL) used at runtime.
- SMG510W.DLL -XDS510 or XDS510PP specific DLL used at runtime.

Additional PC Executables for XDS510PP only
- SMCMODE.EXE -SMC port configuration utility.
- NSCMODE.EXE -National port configuration utility.
- PORTCHK.EXE -Bi-directional Parallel Port Detection Utility.
- XDS510PP.INI -Initialization file for XDS510PP

Target configuration files
- BOARD.CFG -Text file that describes the target system to the JTAG loader.
- BOARD.DAT -Binary version of BOARD.CFG.

Batch Files for B0 method
- BC0.BAT - Batch file to run clear algorithm.
- BE0.BAT - Batch file to run erase algorithm.
- BFLW0.BAT - Batch file to run flash-write algorithm.
- BP8K.BAT - Batch file to run programming algorithm.
- BTEST.BAT - Batch file to test JTAG connections using L20.OUT.

Sample COFF files
- L20.OUT -Sample COFF file for testing setup by programming 20 words of flash0.
- L8KN.OUT -Sample COFF file for testing setup by programming all 8k words of flash.

Readme file
- README2.PDF -Acrobat version of this file.
- README.PP -Text file describing XDS510PP usage.

SRC Sub-directory with target control files (.ASM,.CMD,.LST,.MAP,.OBJ)
- C2XX_BCX.ASM -Control file for clear.
- C2XX_BEX.ASM -Control file for erase.
- C2XX_BPX.ASM -Control file for program.
- C2XX_BFX.ASM -Control file for flash-write.
- C2XX_BTX.ASM -Control file for test program, linked for B0 or SARAM.
- SVAR20.H -Assembly include file for above code.
- MKALL.BAT -Batch file for assemble/link for all algorithms, interactive version.
- MKNOPAUS.BAT -Batch file for assemble/link for all algorithms, non-interactive.

ALGOS Sub-directory with object files for flash algorithms.
- SERA20.OBJ -Algorithm for *erase* operation.
- SCLR20.OBJ -Algorithm for *clear* operation.
- SPGM20.OBJ -Algorithm for *program* operation.
- SFLW20.OBJ -Algorithm for *flash-write* operation.
- SUTILS20.OBJ -Subroutines used by algorithms.

2.0 Programming Basics

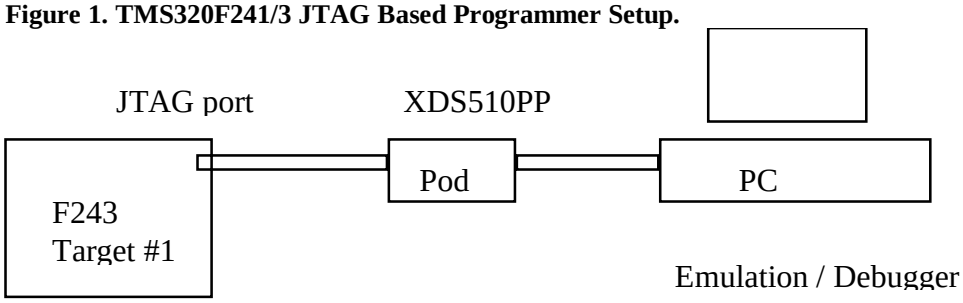
This section provides a brief description of the operations that modify the contents of the flash memory, for a detailed discussion of these operations, the user is referred to the *TMS320F20x/F24x DSP Embedded Flash Memory Technical Reference*, TI Literature Number SPRU282. When the TMS320F243 flash array in an erased state all bits are read as ones and the *program* operation is used to apply a pattern of zeros. For example, the TMS320F243 devices are shipped from Texas Instruments with a boot-loader code programmed into the first few words of the flash array. If the preprogrammed code is not needed, the flash array must be erased before it can be re-programmed. Note that before erasing the array it is very important that all bits be programmed to zero. This procedure of changing all bits to zeros is known as a *clear* operation. Considering this logic, the flash array must be programmed using the following sequence.

clear - make all bits zero 0.
erase - make all bits one 1.
program - make selected bits 0.

In addition to the *clear*, *erase*, and *program* operations, the *flash-write* operation is sometimes required when erasing the flash memory. The *flash-write* operation is used to recover devices that are over-erased. The condition of a flash array that is over-erased is known as *depletion mode* and this condition inhibits re-programming of the array. When the above sequence is followed, over-erasure rarely occurs. However, if the erase algorithm is not preceded by the clear algorithm or if the clock rate of the DSP is not the same as expected by the algorithms, the chances of over-erasing the flash into depletion mode are high.

The only hardware required is a PC host with an XDS510 and a target board with a working JTAG connector. In fact, the actual application board can be used as the target board as long as a JTAG header is provided. Each of the operations described above is performed entirely by the DSP core on the target system. The only function of the XDS510 based loader running on the host PC is to load the required DSP code into the target and return information about programming status. Figure 1 illustrates the setup for programming the TMS320F243 flash using an XDS510.

Figure 1. TMS320F241/3 JTAG Based Programmer Setup.



*Note: Only PC based software is currently supported by TI.

The host PC program which drives the XDS510 offers a large number of command line options. With this in mind, batch files for each of the operations have been provided to get the user started and to give examples of the options.

2.1 Description of Batch Files

TEST PROGRAM

BTEST.BAT ; Used to test the JTAG interface and target connection.

Note: None of the flash operations will function properly if this batch file returns an error. Always use this test before performing any flash operations with the B0 method.

CLEAR ALGORITHM

BC0.BAT ; Clear flash memory array

ERASE ALGORITHM

BE0.BAT ; Erase flash memory array

PROGRAM ALGORITHM

BP8K.BAT ; Program - Flash 0 with L8KN.OUT

Note: This batch file uses a command line argument to specify the name of the file to be programmed. Example, if the following is entered on the command line:

PROGRAM.BAT L8KN.OUT

All 8K words of flash are programmed with the contents of the L8KN.OUT COFF file. The file L8KN.out is given as an example. It should be replaced with the COFF file to be programmed. Rename the file to e.g. MYPROG.BAT and replace the L8KN.OUT by MYCOFF.OUT

FLASH-WRITE ALGORITHM

BFLW0.BAT ; Executes the flash-write algorithm on flash, for depletion recovery.

2.2 Flash Programming Flow

1. Prepare your COFF file (flashcode.out) for programming the flash. This COFF file can contain up to 8K words for the 'F241/3. The only restriction is that the COFF file must not include anything other then the code to be programmed in the flash. Never include data sections in the COFF file that will be used to program the flash. For more information on COFF and working with sections, refer to *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide*, TI literature Number SPRU018D.

2. Verify that the host can properly communicate with the target via the XDS510, by running the test batch file – BTEST.BAT. If an error is returned **DO NOT PROCEED** until the problem is resolved. Check the error conditions in Section 2.3 to isolate the problem.

3. If the flash array to be programmed IS ERASED (i.e. all locations contain 0xFFFFh) then skip steps 3 and 4 and proceed to Step 5. Else, if the array to be programmed IS NOT ERASED, run the BC0.BAT batch file to clear the flash array. If an error occurs, see Section 2.3 for a description of the error, and repeat the entire procedure after correcting the problem.

4. Run the BE0.BAT batch file to erase the flash array. If (ERROR 114) occurs then exit to the step labeled **Depletion**. If any other error occurs, see Section 2.3 for a description of the error, and repeat the entire procedure after correcting the problem.

5. Run the BP8K batch file to program the flash. This batch file uses a command line argument to specify the COFF file to program. To program your specific COFF file from Step 1, copy this batch file BP8K.BAT to another batch file MYPROG.BAT, and run it. Change L8KNOUT to USRCOFF.OUT if USRCOFF is your COFF file, to be programmed into the flash. If an error occurs, see Section 2.3 for a description of the error, and repeat the entire procedure after correcting the problem.

   ******DONE******

Depletion: If (ERROR114) occurs while executing the erase algorithm, the device may be in depletion mode. However, since the programming utility doesn't distinguish between a depletion test failure and other erase algorithm failures, the error may not be caused by depletion mode. If the programming utility is used as described in this manual, the chances of a device being over-erased into depletion mode are very low, and depletion fails can be recovered with the flash-write algorithm. Also, if a device is over-erased into depletion mode, (ERROR114) will be returned whenever a clear or erase is attempted, and the problem should be corrected before attempting to program any other device.

Certain problems in the programming setup can cause depletion failures to occur more frequently. The first is executing the erase algorithm when the flash is not cleared. The second is executing the erase algorithm, with the improper CLKOUT frequency on the target device. If the CLKOUT frequency is not 20Mhz, correct the problem immediately by changing the CLKOUT1 frequency or adjusting the delays in the algorithms. For instructions on adjusting the delays of the algorithms to match the CLKOUT1 frequency, see Section 3.3 of this document.

After the problem is corrected, the affected device can be recovered using the flash-write algorithm. Run the recovery batch file – BFLW0.BAT – to recover the device from depletion. The recovery algorithm will apply flash-write pulses until the device passes the depletion-test or until a maximum number of pulses is reached. If the device does not pass the depletion test then (ERROR 114) will be returned again. If no error is returned then the device is recovered and must under-go the clear/erase/program sequence; proceed to step 3.

2.3 Error Messages

The following is a list and descriptions of the possible error conditions for the host loader PRG2XX.EXE.

System Hangs

If the PC host hangs indefinitely after invoking PRG2XX.EXE, the following conditions may exist:

- Wrong port address specified with the -p command line option.
- Emulator cannot detect target power (i.e. cable not connected, target not powered, improper JTAG connection on target board.)
- External RESET asserted on target.

Errors Related to JTAG Interface

The following errors are all associated with a problem in JTAG communication. As mentioned before, this flash programmer depends on a fully functional JTAG connection. Check the TCK_RET signal at the JTAG connector of the target system; also make sure that no external sources for NMI or RESET are active while programming the flash. A description is given for each error.

ERROR 100 "Processor Initialization"

Target power detected, but scan path is not functional.

- Wrong device name used with -n option.
- The target Vdd level maybe lower than expected.
- One or more JTAG pins may have an opens or shorts fail.

ERROR 101 "Processor Reset"

Emulator is unable to reset the target system.

ERROR 102 "Processor Register Write"

Emulator is unable to initialize the ST1 register.

ERROR 103 "Processor Memory Write"

Emulator is unable to write to memory locations specified by algorithm code.

ERROR 104 "Processor Memory Read"

Emulator is unable to read from memory locations specified by algorithm code.

ERROR 105 "Processor Memory Fill Not Allowed"

Emulator unable to write to expected memory locations on target system.

ERROR 106 "Processor Run"

Target system will not execute from the address specified by the PC register.

ERROR 107 "Processor Halt"

Emulator is unable to halt the target system.

ERROR 108 "Processor Status"

Target processor status is undefined.

Errors Related to File Handling

ERROR 110 "File Open"

One or more files specified on the command line cannot be found. Check the path and filename.

ERROR 111 "COFF Load"

The file specified for programming is not recognized as a COFF file. Re-verify proper command line format; Re-link and check for linker error.

Errors Related to Flash Algorithms

ERROR 109 "Processor Timeout"

Software time-out expires before reaching SWI instruction.

- Cause - The CPU clock-rate is not 20MHz. If the CPU rate is too fast, the software delays used in the programming algorithm will be shortened and the algorithm may not terminate before the time-out period. Check the CLKOUT frequency using an oscilloscope; If the wrong frequency, correct the problem and re-program the flash.

ERROR 112 "Verify"

This error will only occur with the -v option. Not used in Rev2.0.

ERROR 113 "Program"

This error will occur when the programming algorithm fails. Possible causes are:

- Cause - Flash was not fully erased when the program operation was attempted. For example, the COFF file being used to program may extend beyond the end address of flash0, in which case both arrays must be erased before programming. Considering this, retry the *clear*, *erase*, and *program* sequence.
- Cause - The CPU clock-rate is not 20MHz. If the CPU rate is too fast, the software delays used in the programming algorithm will be shortened and the algorithm may reach it's maximum number of retries with no effect. Check the CLKOUT frequency using an oscilloscope; If the wrong frequency, correct the problem and re-program the flash.
- Cause - The wrong control file was used. Use the included batch files as examples.

ERROR 114 "Erase"

This error is shared by the *clear* and *erase* algorithms. Possible causes are:

- Cause - If the error occurs during *clear* and *erase*. Flash was not *cleared* (all locations programmed) when the erase operation was attempted. One or more bits in the array may be in *depletion*.

Follow recovery sequence outlined in Section 2.2 using the *flash-write* batch files. Note, the device may be permanently damaged if more than three tries using the *flash-write* algorithm does not recover the array.

- Cause - If error occurs during *clear*, but not during *erase* or *flash-write*. The array must not be in *depletion*, since the *depletion* check is only performed in the *erase* and *flash-write* algorithms. The CPU clock-rate is not 20MHz. If the CPU rate is too fast, the software delays used in the clear or programming algorithm will be shortened and the algorithm may reach it's maximum number of retries with no effect. Check the CLKOUT frequency using an oscilloscope; If the wrong frequency, correct the problem and re-program the flash. If this does not correct the problem then the array maybe permanently damaged.

Other Errors

ERROR 115 "Missing symbol"

This error is caused if a symbol passed from the host loader is not defined in the target code. This will never occur if the target code is used as provided. If the code has been modified, use the control files as examples to verify that all variables of the form (PRG_***) have been defined.

3.0 Customizing the Programming Utility

Although the XDS510 Based Flash Programming Utility provides a complete solution for programming the ‘F243 device, a user may wish to customize the setup for a particular application. For example, once the user becomes familiar with the setup he or she may wish to modify the command line options to perform multiple tasks simultaneously, or to use another host program to invoke the loader. Another possibility is that the user may wish to program the flash in the final application board, in which the ‘F243 may share the scan chain with other devices. Yet another possible modification would be to use the XDS510 based loader to perform some other system initialization tasks by downloading the appropriate code to the ‘F241/3. For example, the target code can be modified for programming of an external flash device connected to the DSP. The information provided in the following sections give the user a number of options for customizing the programming utility.

3.1 Command Line Format for the Loader

prg2xx.exe [-options] c2xxprog.out [flashcode.out]

Option	**Description.**
-h	Help, Lists the options.
-n *Device Name*	Identifies the processor to program if multiple devices are connected to the scan-chain. For more details on this option see Section 3.2, *Describing the Target System to the Loader*. Optional, default=c200.
- p *port address*	Specifies IO address for XDS510 card in the PC. Not required for the XDS510PP. Optional, default = 240.
- w *time-out*	Specifies time-out limit (1-6) for the host while programming. Necessary to match fast and slow PCs. Optional, default = 1.
- i *I/O register*	I/O address to be initialized before program loading. Used to initialize the SARAM mapping in the PMST register at 0xFFE4h. Optional, default 0xFFE4h.
- m *I/O value*	Value to be written in the I/O address specified by the -i option. For flash programming the value 0x0006 should be written in the PMST register. To initialize the PMST register at 0xFFE4, just use -m 0x0006; -i option is not required. Optional, default = no-load.
- o	No flashcode COFF file for programming. If this option is used, the programming algorithm will not be executed.
- t *ST1 value*	Initialize ST1 register. Optional, default 0x17FCh.
- e	Run PRG_erase function before executing PRG_prog function. Optional, default = not selected.
- v	Run PRG_verify function after executing PRG_prog function.

default — If neither -e or -v options are specified, the loader will run PRG_init first followed by PRG_prog and PRG_stop. These functions are defined in the c2xxprog.out COFF files.

- s *PRG_option* — The 16bit HEX operand is used to initialize the PRG_option variable in target ram. This command line option can be used to pass a flag to the target device, for adding custom features. See Section 3.4 for more details.

3.2 Describing the Target System to the Loader

In order for the XDS510 based loader to understand how you have configured your target system, you must supply a configuration file for the loader to read.

- If you're using an emulation scan path that contains only one 'F243 and no other devices, you can use the *board.dat* file that was included with the programming utility. This file describes to the loader the single 'F243 in the scan path and gives it the name C200. Since the loader automatically looks for the name C200 in the board.dat file, you can skip this Section.
- If you plan to use a target system with multiple devices in the scan-chain, you must follow the procedure described below.

Step 1: Modify the Board Configuration Text File (board.cfg)

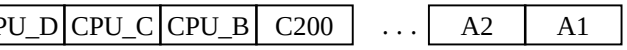
The file consists of a series of entries, each describing one device on the scan path. Each entry in the file consists of at least two fields – the device name enclosed in double quotes, and the device type. The device name specified in the configuration file must begin with an alphabetic character and can consist of up to eight alphanumeric characters including the underscore. This is the same name that is used with the -n command-line option of the loader. The 'F2XX JTAG based loader supports the following two device types:

- TI320C2xx — Describes a device in the 'C2xx/'F2xx family.
- BYPASS## — Describes a none 'C2xx/'F2xx device to be bypassed, where the ## is the hexadecimal number of bits in the device's JTAG instruction register.

The order in which the devices are listed is important. The loader scans the devices assuming that the data from one device is followed by the data of the next device on the chain. The devices should be listed in the order in which their data reaches the JTAG loader. So the device with TDO pin connected directly to the TDO pin of the emulation header should be first on the list. An example of a multiple device scan chain and the corresponding board.cfg file is given below in Figure 3.2.

Figure 3.2, An example of a ‘F2xx device chain.

a) A sample ‘F2XX device chain.



b) A sample board.cfg file.

DEVICE NAME	DEVICE TYPE	COMMENTS
“A1”	BYPASS08	;First device (8bits) nearest TDO. ;(test data output)
“A2”	BYPASS10	;Next device (16bits).
“C200”	TI320C2xx	;the first C2XX.
“CPU_B”	TI320C2xx	
“CPU_C”	TI320C2xx	
“CPU_D”	TI320C2xx	;The last ‘C2xx in chain. ;(test data in)

Step 2: Translate the File to the Binary Conditioned Format (board.dat)

Once the text file has been modified use the composer.exe utility to generate the special binary format (board.dat). Note the board.cfg file must be in the same directory as the composer.exe utility.

Step3: Use the Command Line Option to Specify the Device to Program

If there are multiple ‘F2XX devices on the scan chain and the device to be programmed is not named C200, then the device name must be specified using the -n option. Note that the ‘F2XX JTAG loader always looks for the file board.dat in the working directory, so there is no need to specify the filename.

3.3 Adjusting Clock Frequency

The programming algorithms for the ‘F241/3 on-chip flash include software delays that must be adjusted according to the instruction rate of the target device. This chapter describes how to modify the programming utility for use with different clock frequencies.

WARNING! If the design will be using a variable CLKOUT, (i.e. CLKOUT will be varied by the application) then the flash should be erased at the highest possible CLKOUT rate. This is important to insure adequate read-back margin through-out the life of the application. If for instance the CLKOUT may be any of 5,10,20 MHz, then the flash must be erased using the CLKOUT at 20Mhz. Refer to the *TMS320F20x/F24x DSP Embedded Flash Memory Technical Reference*, TI Literature Number SPRU282, for more information on this requirement.

3.3.1 Adjusting the algorithm delays: sys-clock frequencies <> 20Mhz.

If the instruction rate (sys-clock frequency) of the target system is not 20Mhz, than the algorithm delays must be adjusted accordingly. Follow these steps to adjust the delays for the XDS510/PP based programming utility:

1) Locate the SVAR20.H and SUTILS20.ASM files in the ALGOS sub-directory. See Section 1.0 for more details on the directory structure. Using a text editor, modify the DLOOP constant in the

SVAR20.H file according to the CLKOUT frequency of the target board. Use the following equation to modify the DLOOP constant:

DLOOP = (0.000005/tCLKOUT)-6 ;Round down to the next integer.

Examples
---
fCLKOUT=20Mhz, tCLKOUT=50ns, DLOOP=94
fCLKOUT=16.384Mhz, tCLKOUT=61ns, DLOOP=75
fCLKOUT=15Mhz, tCLKOUT=67ns, DLOOP=69
fCLKOUT=10Mhz, tCLKOUT=100ns, DLOOP=44
fCLKOUT=9.8304Mhz, tCLKOUT=102ns, DLOOP=43
fCLKOUT=5Mhz, tCLKOUT=200ns, DLOOP=19
fCLKOUT=3Mhz, tCLKOUT=333ns, DLOOP=9

Save the modified SVAR20.H file in the ALGOS sub-directory.

2) Using the TMS320C1x/C2x/C2xx/C5x Assembler, re-assemble the SUTILS20.ASM file in the ALGOS sub-directory. Note, the modified include file from Step 1 must be in the same directory as SUTILS20.ASM.

3) Locate the target control files—C2XX_BCX.ASM, C2XX_BEX.ASM, C2XX_BPX.ASM, C2XX_BFX.ASM, C2XX_BTX.ASM—in the SRC sub-directory. See Section 1.0 for more details on the directory structure. Using the TMS320C1x/C2x/C2xx/C5x Linker, re-link the target control files (C2XX_***.ASM) via the corresponding linker command files (C2XX_***.CMD) in the SRC directory.

4) The MKALL.BAT file does all these operations for the user.

The programming utility can now be used with the CLKOUT frequency of the target.

3.4 Target Code Structure

This section describes the part of the programming utility which executes on the target DSP. The loader used in this programming utility (prg2xx.exe) provides a means of communication between the PC host, and the target DSP via the XDS510. This communication link is used for programming the on-chip flash of the ‘F241/3, however it is not restricted to this function. For example, the loader can be used to direct the DSP to initialize some external components, or to run self-diagnostic tests during production. The information provided in this section can be used to modify the current target files for custom applications.

3.4.1 Control Files

When the loader (prg2xx.exe) executes on the PC host, it communicates with the target device via the XDS510 in the following way:

Example command line entry— prg2xx.exe [-options] c2xxprog.out [flashcode.out]

The file, c2xxprog.out is a COFF file that controls execution of the programming algorithms. The control modules included in the SRC sub-directory – assembly source provided – are all examples of this (e.g. C2XX_BCX.OUT, C2XX_BEX.OUT, C2XX_BPX.OUT, C2XX_SPX.OUT, etc.)

The c2xxprog.out file must contain a set of specifically named functions. The loader directs the DSP to branch to these locations, as described below. Each of these functions should end with a BE90h opcode,

rather than the standard return instruction. The BE90h opcode is a special software interrupt which tells the XDS510 or XDS510PP that the target device is ready to be scanned. The c2xxprog.out file must contain the following functions:

PGR_init	- This is where any device initialization code should be placed.
PRG_program	- This is where the code for programming should be placed.
PRG_erase	- This is where the code for erasing should be placed.
PRG_verify	- This is where the code for verifying should be placed.

Additionally, the c2xxprog.out file must contain the following constants:

PRG_bufaddr	- This is the start address of the on-chip buffer used for programming data.
PRG_bufsize	- This is the defines the maximum size of the onchip buffer used for programming data.
PRG_devsize	- This defines the maximum size, in 16bit words, of the programmable device.
PRG_page	- This defines whether the programmable device is mapped in page 0 (program space), or in page 1 (data space).

Additionally, the file must contain the following variables:

PRG_status	- This is used by the target to communicate algorithm status to the host (0 = pass, 1 = fail).
PRG_paddr	- This is initialized by the host with the start address of the programmable device that should be programmed with the current buffer data.
PRG_length	- This is initialized by the host with the length, in 16bit words, of the filled portion of the buffer.

If any of these symbols are not defined, then the host program will return error#115. Note, that in addition to the above variables, the PRG_options variable will be initialized by the -s option. This variable isn't required (i.e. won't cause ERROR115 if missing) but provides a way to pass a flag to the 'F241/3 via the command line. See *Section 3.1, Command Line Format for the Loader*, for details on how it is used in the standard utility.

3.4.2 Sequence of events that occur when PRG2XX.EXE is run.

When the following command line is entered on the PC host,

prg2xx.exe [-options] c2xxprog.out [flashcode.out]

communication between the host and target DSP follows these steps:

Step1:
******
The host program, loads the COFF file c2xxprog.out into the 'F241/3 addresses specified at link time. In other words, if c2xxprog.out is linked at B0, then it will be loaded at that address. The files c2xx_bpx.asm, and c2xx_btx.asm are implemented this way. Alternatively, if c2xxprog.out is linked to SARAM, and SARAM is mapped in both program and data space (PMST=6), than it will be loaded into the SARAM. The file c2xx_spx.asm is implemented this way.

Step2:
******
The host then directs the 'F241/3 to begin executing the code from c2xxprog.out. The host forces the ‘F241/3 to branch to one of the functions defined above, then regains control when the ‘F241/3 executes the reserved software interrupt (opcode = BE90h). Upon regaining control, the host checks the PRG_status variable before proceeding to the next operation. The following execution sequences are used:

- If neither the -e nor the -v option is used then execution order will be
  PRG_init -> PRG_program.
- Otherwise, if the -e option is used execution order will be
  PRG_init -> PRG_erase -> PRG_program.
- Finally, if the -v option is used execution order will be
  PRG_init -> PRG_program -> PRG_verify.

Further, the -v and -e options are independent, so that if both are used, the execution flow will be
PRG_init -> PRG_erase -> PRG_program -> PRG_verify.

The execution of the PRG_program and PRG_verify functions follows a special sequence. If the -o option is not used, then the host program will load the COFF file flashcode.out into the 'F241/3 data memory at the address defined by the PRG_bufaddr constant. The host will try to fill the entire buffer as defined by the PRG_bufsize constant, but if the file flashcode.out is larger then the PRG_bufsize variable, the host will program or verify the file in blocks. Each time the buffer is filled, or a COFF section ending is encountered, the host initializes the PRG_paddr variable with the start address of the block to be programmed or verified, and it initializes the variable PRG_length with the number of words in the buffer, then it directs the DSP to execute the PRG_program or PRG_verify function. The host continues to direct execution to these functions, until all the data in flashcode.out has been buffered and successfully programmed or verified.

3.4.3 Algorithm Files

The control files described above are linked with the standard algorithms for performing the various operations on the embedded flash memory of the ‘F241/3 DSP. These algorithms are in the ALGOS sub-directory. Refer to the *TMS320F20x/F24x DSP Embedded Flash Memory Technical Reference*, TI Literature Number SPRU282, for a detailed description of the standard algorithms, and the calling conventions used.