



TMS320LF240x(A) On-Chip Flash Programming

with the XDS510, XDS510PP, and XDS510PP_PLUS Emulators

This document provides a guide to programming the on-chip Flash ROM on the TMS320LF240x DSP Controllers. The TMS320LF240x parts are referred to by the generic name and specific part numbers are used wherever an exception needs to be made.

The environment for these flash programming tools is a Microsoft Windows 95/98 PC, with a Texas Instruments XDS510(PP) JTAG Emulator.

The files described in this document are available for download from the Texas Instruments' Website at <http://www.ti.com>

Important Notes

1. The programming algorithms included in the TMS320LF240x(A) Flash Programming Utilities are different from the 'F24x devices ('F240, 'F241 and 'F243). Do not intermix these, as this will not function as intended.
2. The tools are configured to work with a particular directory structure: Make sure that the unzip utility you use restores the directory structure.
3. The utilities as distributed are configured to work with a default input frequency of 7.5 MHz, and are set up to multiply the input with a multiplier of 4, resulting in a CPU Clock frequency of 30MHz for the LF240x utilities. For the LF240x-A these are configured for a default input frequency of 10MHz input and 40MHz CLKOUT. If your input clock frequency is other than the default, the utilities will need to be reconfigured before use. See section 2.3 for details. For the LF240xA parts the maximum CPUCLK limit is 40MHz, for the LF240x this is 30MHz.

Date Released 01/04/2001



Contents

| | | |
|--------|---|----|
| 1.0 | Flash Programming Basics | 3 |
| 1.1 | Batch files for Flash Operations | 4 |
| 1.2 | Flash Programming Flow | 5 |
| 1.3 | Error Messages | 7 |
| 1.3.1. | System Hangs | 7 |
| 1.3.2. | Errors Related to JTAG Interface | 7 |
| 2.0 | Customizing the Programming Utility | 10 |
| 2.1 | Command Line Format for the Loader | 10 |
| 2.2 | Describing the Target System to the Loader | 11 |
| 2.3 | Adjusting Clock Frequency | 13 |
| 2.3.1. | Configuring the PLL Multiplier Ratio | 13 |
| 2.3.2. | Scaling the Timing Parameters | 13 |
| 2.3.3. | Generating a timing set | 13 |
| 2.3.4. | Re-building the algorithms with the new timing parameters | 14 |
| 2.4 | Sector Clear and Erase Operations | 14 |
| 2.5 | Target Code Structure | 14 |
| 2.5.1. | Control Files | 14 |
| 2.5.2. | Sequence of events that occur when PRG2XX.EXE is run. | 15 |
| 2.5.3. | Algorithm Files | 16 |

Figures

| | |
|--|----|
| Figure 1. TMS320LF24xx JTAG Based Programmer Setup. | 3 |
| Figure 2. An example of a 'F2xxx device chain. | 12 |

1.0 Flash Programming Basics

This section provides a brief description of the operations that modify the contents of the flash memory. The Flash Cells in the Flash Memory can be in two states in normal operation. It can contain either a valid 'zero' level or a valid 'one' level. When the TMS320LF240x/240xA flash array is in an erased state all bits are read as ones. When the array contains all zeros then it is said to be in a cleared state.

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 zero (0).

The only hardware required is a PC host with an XDS510 and a target board with a working JTAG connector. 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 TMS320LF240x flash using an XDS510.

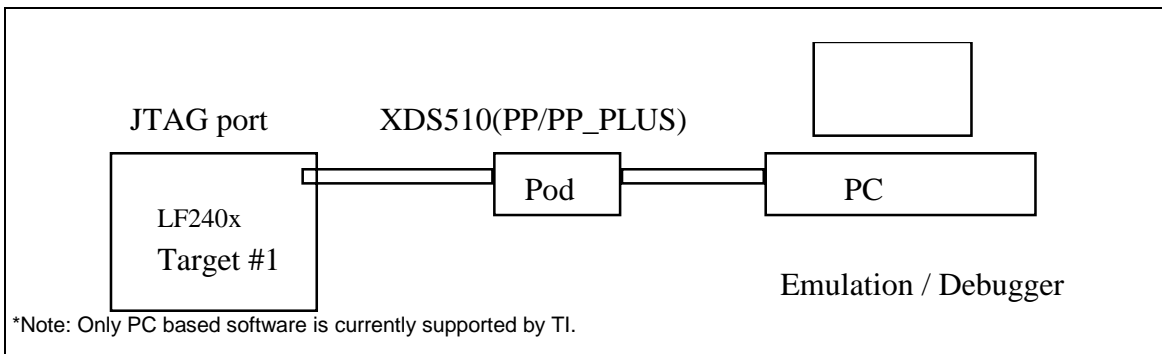


Figure 1. TMS320LF24xx JTAG Based Programmer Setup.

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



1.1 Batch files for Flash Operations

JTAG Connection Test Program

BTEST.BAT Used to test the integrity of the JTAG connection. This script must be successfully executed before attempting any other flash operations. This loads a dummy algorithm in the DSP Controller and checks to see if the target DSP is responding correctly. (Once you are sure of your JTAG connection, this may be skipped, without damaging the device).

Clear Operation

CLR_ALL.BAT This batch file invokes the clear algorithm, which preconditions the flash prior to erase operations. This step **MUST** be executed each and every time before the Flash is erased. Skipping this step can cause PERMANENT damage to your DSP controller flash memory.

This clears(preconditions) all four sectors of the flash array.

To clear Sector 0 , leaving the other sectors untouched, run CLR_S0.BAT. For clearing sectors 1,2 and 3 run the batch files CLR_S1.BAT , CLR_S2.BAT, CLR_S3.BAT respectively.

For more information on how to clear sectors individually see section 2.4.

Erase Operation

ER_ALL.BAT This batch file invokes the erase algorithm, which erases the flash, setting all the bits in the flash memory to a 'one' state.

This erases all the four sectors in the array.

To erase Sector 0 , leaving the other sectors untouched, run ERA_S0.BAT. For erasing sectors 1,2 and 3 run the batch files ERA_S1.BAT , ERA_S2.BAT, ERA_S3.BAT respectively.

For information on how to erase sectors individually see section 2.4.

Program Operation

BP32K.BAT Invokes the programming algorithm. A sample file with 32K words of dummy code is provided, for illustration. Copy this batch file to another batch file e.g. MYPROG.BAT and then substitute your COFF file name for the I32kn.out.

```
prg2xxw95 -p 240 -w 6 program\c2xx_bpX.out L32kn.out
```

should be changed to

```
prg2xxw95 -p 240 -w 6 program\c2xx_bpX.out mycode.out
```

Un Locking / Locking Operations (TM(X/S) 320LF240x A only)



LOCK.BAT Programs the password into the password locations in the flash array.

UNLOCK.BAT Reads the password from the key file and unsecures the flash array, by writing the password into the key registers.

1.2 Flash Programming Flow

1. Prepare your COFF file (flashcode.out) for programming the flash. This COFF file can contain up to 32K words for the 'LF2407. See datasheet for other devices. The only restriction is that the COFF file must not include anything other than the code to be programmed in the flash. **Never include sections mapped to other memory (other than the Flash) in the COFF file that will be used to program the flash. This will interfere with the programming of the flash.** For more information on COFF and working with sections, refer to the *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools User's Guide*.
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 1.3 to isolate the problem.
3. For LF240x units proceed to the step 5. For LF240xA units the flash must be unsecured before programming operations are attempted. To do this, perform the unlock operation as described in step below. If the unit does not contain a password in the password locations (i.e. the password locations are all 0xffff or 0x0000) then the unlock is not needed.
4. Locate the key.asm file in the key subdirectory.
Typically this is in c:\specdig\prg2xx\lf2407a\key. Open this file in a text editor and key in the password. For example the key.asm lines shown below show the password 0x3453, 0x2848, 0x2373, and 0x3498 used for the password locations 0x40,0x41,0x42,0x43.

```

;-----
; Sample Key file.
; This key is used in conjunction with the program routine to
; 'lock' the LF2407A, and with the unlock algo to unlock the
; lf2407A device.
;-----
        .text

        .word    3453h
        .word    2848h
        .word    2373h
        .word    3498h

        .end

```

Once the key.asm file contains the password, save the file and then run the batch file key.bat to create the COFF file key.out.

Now run unlock.bat. This will copy the passwords to the password locations and then unsecure the device by using the password match flow for the device. See SPRU357A for details on the password match flow.

5. If the flash array to be programmed IS ERASED (i.e. all locations contain 0xFFFF) then skip steps 3 and 4 and proceed to Step 5. Else, if the array to be programmed IS NOT ERASED, run the CLR_ALL.BAT batch file to clear the flash array. **THIS INCLUDES THE PASSWORD**



LOCATIONS. If any password is programmed the array must be cleared before it is erased.

If an error occurs, see Section 1.3 for a description of the error, and repeat the entire procedure after correcting the problem.

6. Run the ER_ALL.BAT batch file to erase the flash array. If any error occurs, see Section 1.3 for a description of the error, and repeat the entire procedure after correcting the problem.
7. Run the BP32K 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 BP32K.BAT to another batch file MYPROG.BAT, and run it. Change L32KNOUT to MYCODE.OUT where MYCODE.OUT is your COFF file, to be programmed into the flash. If an error occurs, see Section 1.3 for a description of the error, and repeat the entire procedure after correcting the problem.

CAUTION

Note: Running this batch file puts in a password, if the COFF file contains data at addresses 0x40-0x43. e.g. the L32KN.OUT has 0x0040, 0x0041, 0x0042, 0x0043 as 'password' data.

8. Securing the flash after programming.

To secure the flash the password must be programmed into the password locations. There are two ways to do this:

- i. Include the password as part of the COFF file programmed into the device. In this case simply turning the device off and back on will activate the security logic per the description in the device user guide (SPRU357A).
- ii. In case the password locations are not programmed as part of the COFF file, and it is to be done as a separate step, then set up key.out per step 4 (above).

Then run the lock.bat batch file. This will program the password into the password locations.

One major source of errors in C/E/P operations is executing the algorithms with the improper CLKOUT frequency on the target device. If the CLKIN frequency is other than 7.5Mhz, correct the problem immediately by changing the CLKIN frequency or adjusting the delays in the algorithms. For instructions on adjusting the delays of the algorithms to match the CLKIN frequency, see Section 2.3 of this document. The defaults are 7.5MHz and multiplier of 4. For the 'Lf240x-A the defaults are a CLKIN of 10MHz and a PLL multiplier of 4.



1.3 Error Messages

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

1.3.1. 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.

1.3.2. 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"

- Scan path not functional – check setup as in section 1.3.1
- 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 differs from the rate used to configure the utility. See section 2.3 for information to configure the utility. 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.

ERROR 113 "Program"

(Note: The flash errors described below are defined only in the newer embeddable algos. The older algos generate error code 1 for all the error conditions.)

Flash Error 3: This error will occur when the programming algorithm fails to program bits in the flash array despite attempting maximum number of pulses to the location. Possible causes are:

- Cause (LF240xA ONLY) The flash is secured. Follow the instructions to unsecure it in section 1.2
- Cause The COFF file may contain stray data. For example, the COFF file being used to program may extend beyond the end address of sector(s) erased. Considering this, retry the *clear*, *erase*, and *program* sequence, erasing all needed sectors. Also make sure that the COFF file does not exceed the ENTIRE array.
- Cause - The CPU clock-rate is not correct per the utility configuration. 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.
- Cause – The flash array may be permanently damaged.

Flash Error 4: This error will happen only in a certain circumstance: when a bit in flash is a zero,



when its intended value as defined by the COFF file is a one. Typically this is caused by attempting to program a word that has not been erased, and the COFF file seeks to put in a 1 in a location where there is already a zero from a previous programming cycle.

ERROR 114 "Erase"

(Note: The flash errors described below are defined only in the newer embeddable algos. The older algos generate error code 1 for all the error conditions.)

Flash error 1: This error is generated by *clear* algorithm. Possible causes are:

- Cause (LF240xA ONLY) The flash is secured. Follow the instructions to unsecure it in section 1.2
- Cause The flash algorithms could not set the bits to a zero. This can happen due to the following reasons:
 1. The CPU clock-rate is not correct as per utility configuration. 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 its 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.
 2. Vccp is not at 5V – Flash Operations are not defined under these conditions. Ensure the correct voltage on the Vccp pin.

Flash Error 2: This error is generated by *erase* algorithm. Possible causes are:

- Cause (LF240xA ONLY) The flash is secured. Follow the instructions to unsecure it in section 1.2
- Cause The flash algorithms could not set the bits to a one. This can happen due to the following reasons:
 1. The CPU clock-rate is not correct as per utility configuration. 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 its 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.
 2. Vccp is not at 5V – Flash Operations are not defined under these conditions. Ensure the correct voltage on the Vccp pin.

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.



2.0 Customizing the Programming Utility

Although the XDS510 Based Flash Programming Utility provides a complete solution for programming the 'LF240x 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 'LF240x 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 'LF240x. 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.

2.1 Command Line Format for the Loader

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

| Option | Description. |
|------------------------|---|
| -h | Help, Lists the options. |
| -n <i>Device Name</i> | Identifies the processor to program if multiple devices are connected to the scan-chain. For more details on this option see Section 3.2, <i>Describing the Target System to the Loader</i> . Optional, default=c200. |
| -p <i>port address</i> | Specifies IO address for XDS510(PP/PP_PLUS/card) in the PC. Optional, default = 240. The prg2xxw95 selects which configuration from the SDOPTS.CFG to use depending upon this address. |
| -w <i>time-out</i> | Specifies time-out limit (1-6) for the host while programming. Necessary to match fast and slow PCs. Optional, default = 1. |
| -i <i>I/O register</i> | 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>I/O value</i> | 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 <i>ST1 value</i> | 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 2.4 for more details.

On the TMS320LF240x devices, the flash has four sectors. The operations clear and erase are run on a per-sector basis depending on the four least significant bits of the -s parameter.

See section 2.4 for details.

2.2 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 'LF240x 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 'LF240x 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. Note that the 'LF240x devices share the same core with the F24x devices, so this name is still C200.
- 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.



a) A sample 'F2XX device chain.

TDI

| | | | |
|-------|-------|-------|------|
| CPU_D | CPU_C | CPU_B | C200 |
|-------|-------|-------|------|

 ...

| | |
|----|----|
| A2 | A1 |
|----|----|

 TDO

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 F/LF/C2XX. |
| "CPU_B" | TI320C2xx | |
| "CPU_C" | TI320C2xx | |
| "CPU_D" | TI320C2xx | ;The last 'F/LF/C2xx in chain. ;(test data in) |

Figure 2. An example of a 'F2xx device chain.

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.



2.3 Adjusting Clock Frequency

The programming algorithms for the 'LF240x 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 ensure adequate read-back margin throughout 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.

The current version of the flash programming utilities is shipped to program the flash memory based on the assumption that the device will be clocked at a CLKIN of 7.5 MHz. Based on this assumption the flash programming utilities configure the PLL at a multiplier of 4, thus the resulting CLKOUT is 30MHz.

If the frequency of CLKOUT will be different for some reason, e.g. the CLKIN is not 7.5 MHz, then it is necessary to re-configure the flash programming utilities to take this into account. To do this following things must be accomplished.

- a. **PLL Multiplier Configuration**
- b. **Scaling the Timing Parameters.**
- c. **Generating a timing set.**
- d. **Re-building the algorithms.**

2.3.1. Configuring the PLL Multiplier Ratio

To configure the PLL multiplier ratio, locate the VAR.H file in the include sub-directory. In this file locate the STEP 1 Section. Uncomment the PLL_RATIO_CONST for your selection. For example, assume a system running a 25MHz clock off a 12.5 MHz external source. CLK_RATIO_CONST for a multiplication ration of 2 is chosen, and uncommented. All other settings MUST be commented out.

2.3.2. Scaling the Timing Parameters

Timing parameters are selected based on the clock frequency the CPU is running at. A few timing sets are distributed with the tools. Each set of timings is contained in a file called timings.xx. For example the file containing loop timings calibrated at 25MHz is called timings.25. In STEP 2 in VAR.H choose the appropriate timing set.

2.3.3. Generating a timing set

Lets assume that a system needs timings to be calibrated at 27 MHz. For this purpose the Excel worksheet timings.xls is provided. To generate a new set of timings open this file in Microsoft Excel, and see the instructions in the worksheet.

2.3.4. Re-building the algorithms with the new timing parameters

Once the timing parameters are chosen correctly, run the batch file buildall.bat to re-assemble and link the algorithms.

2.4 Sector Clear and Erase Operations

The TMS320LF240x devices support sector wide operations. This means that instead of clearing and erasing the entire flash memory at once, it is possible to clear erase and re-program just part of the memory. To do this the '-s' parameter in the command line is initialized with a sector command. The four least significant bits of the -s parameter are decoded to indicate which sectors are to be operated on. For example in the clear batch file the command line has -s as follows: -s 0x000a.

This means that the bits 3 and 1 are set while the bits 0 and 2 are not. The clear algorithm will now clear out sectors 1 and 3 while the sectors 0 and 2 are left unmodified.

The erase algorithm works in the same manner, with a sector being erased if the corresponding bit is set, otherwise leaving it untouched.

A sector must be pre-conditioned using the clear operation before using the erase operation on it. Failure to observe this pre-caution could permanently damage the device and have other unforeseen consequences.

Batch files to clear and erase the sectors individually are provided.

Clr_s0.bat clears sector 0. Clr_s1.bat, Clr_s2.bat, Clr_s3.bat clear sectors 1,2,3 respectively. Er_s0.bat erases sector 0. Er_s1.bat, Er_s2.bat, Er_s3.bat erases sectors 1,2,3 respectively.

For the TMS320LF2401A device, the flash has only two sectors. So batch files are provided only for sectors 0 and 1.

2.5 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 'LF240x, 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.

2.5.1. 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 sub-directories – assembly source provided – are all examples of this structure.

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 op-code, rather than the standard return instruction. The BE90h op-code 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 'F24xx 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.

2.5.2. 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 'LF240x 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.

Step2:

The host then directs the 'LF240x to begin executing the code from `c2xxprog.out`. The host forces the 'LF240x to branch to one of the functions defined above, then regains control when the 'LF240x 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 'LF240x 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 than 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. It also initializes the variable `PRG_length` with the number of words in the buffer, and 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.

2.5.3. 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 'LF240x DSP. These algorithms are in the `algos` subdirectories.



TI Contact Numbers

INTERNET

TI Semiconductor Home Page

www.ti.com/sc

TI Distributors

www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas

Phone +1(972) 644-5580

Fax +1(972) 480-7800

Email sc-infomaster@ti.com

Europe, Middle East, and Africa

Phone

Deutsch +49-(0) 8161 80 3311

English +44-(0) 1604 66 3399

Español +34-(0) 90 23 54 0 28

Français +33-(0) 1-30 70 11 64

Italiano +33-(0) 1-30 70 11 67

Fax +44-(0) 1604 66 33 34

Email epic@ti.com

Japan

Phone

International +81-3-3344-5311

Domestic 0120-81-0026

Fax

International +81-3-3344-5317

Domestic 0120-81-0036

Email pic-japan@ti.com

Asia

Phone

International +886-2-23786800

Domestic

Australia 1-800-881-011

TI Number -800-800-1450

China 10810

TI Number -800-800-1450

Hong Kong 800-96-1111

TI Number -800-800-1450

India 000-117

TI Number -800-800-1450

Indonesia 001-801-10

TI Number -800-800-1450

Korea 080-551-2804

Malaysia 1-800-800-011

TI Number -800-800-1450

New Zealand 000-911

TI Number -800-800-1450

Philippines 105-11

TI Number -800-800-1450

Singapore 800-0111-111

TI Number -800-800-1450

Taiwan 080-006800

Thailand 0019-991-1111

TI Number -800-800-1450

Fax 886-2-2378-6808

Email tiasia@ti.com

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.



IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.

Copyright © 1999 Texas Instruments Incorporated