TMX320F206 JTAG Based Flash Programmer

This document explains how to use the TMX320F206 JTAG based programmer to program the 'F206 on-chip flash array via an XDS510 connection. The programmer consists of a JTAG based loader which runs on a PC compatible host and flash programming algorithms which run on the target DSP. The current revision of the software (rev 2.0) – including the loader and flash algorithms – is a beta version, and future revisions will be released to give error free operation. The first two sections provide all the information needed to get started. The last section is provided for those users who wish to customize the Programmer.

Environment : IBMPC compatible with Windows 95 only.

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

Program type : DOS window command.

New Features:

* New Windows95 compatible loader.
* Improved algorithms for increased stability.
* 1 to 32K contiguous programming with a single COFF file.
* Recovery method for *depletion*.
* Support for XDS510PP (PGMR20PP.ZIP)

IMPORTANT CONSIDERATIONS FOR VERSION 2.0

* Revision 2.0 of this software is intended for use with PG2.0 'F206 devices. Limited functionality is possible with PG1.0 silicon, but the *erase* and *flash-write* operations will not function as specified in this document.

* The 'F206 **MUST** run at 20 MIPS. That is CLKOUT1 should be 20Mhz/50ns. All of the flash program timings are based on this clock; **DO NOT CHANGE CLKOUT** frequency with these flash programming utilities.

* A directory structure has been included with the ZIP file (PGMR20.ZIP) for Revision 2.0. The **–d option MUST be used with the** unzip utility to restore this directory structure. The included batch files and linker command files use the directory structure and will not work without it.

* The flash algorithms are still under development and TI qualification testing. For this reason, the algorithms are provided only as object files (.obj); **DO NOT** use these files except in the form suggested. When the 'F206 devices are fully qualified (TMS), the source and documentation for the algorithms will be released to facilitate integration with applications.

* The files contained in PGMR20.ZIP are designed to be used with the XDS510 only. For flash programming with the XDS510PP the file PGMR20PP.ZIP must be used instead.

* For any issues on F206 programming, please contact TI. Send an email to DSP hotline with the subject marked as 'F206 - Programming'. Refer to TI's WEB site for future updates on software and device.
  TI DSP hotline address: dsph@ti.com   TI Internet address : www.ti.com/dsps

---

1.0 Required Files

The following files are required to use this tool and are included in the file PGMR20.ZIP. The -d option MUST be used with the unzip utility to extract the directory structure.

PC Executables -Note, additional host files for the XDS510PP are included in the PGMR20PP.ZIP file.
- PRG2XX.EXE   -Windows 95 executable file that invokes the JTAG loader for XDS510.
- EMURST.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 specific DLL used at runtime.

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   -*Clear* operation on flash0 using B0 method.
- BC1.BAT   -*Clear* operation on flash1 using B0 method.
- BE0.BAT   -*Erase* operation on flash0 using B0 method.
- BE1.BAT   -*Erase* operation on flash1 using B0 method.
- BFLW0.BAT   -*Flash-write* operation on flash0 using B0 method.
- BFLW1.BAT   -*Flash-write* operation on flash1 using B0 method.
- BP32K.BAT   -*Program* operation on flash0/1 using B0 method.
- BTEST.BAT   -Batch file to test JTAG connections using B0 method and L20.OUT.

Batch Files for SARAM method ('F206 only).
- SCE0.BAT   -*Clear* and *erase* operations on flash0 using SARAM method.
- SCE0.BAT   -*Clear* and *erase* operations on flash1 using SARAM method.
- SCEP0.BAT   -*Clear*, *erase*, and *program* operations on flash0 using SARAM method.
- SCEP1.BAT   -*Clear*, *erase*, and *program* operations on flash1 using SARAM method.
- SP32K.BAT   -*Program* operation on flash0/1 using SARAM method.
- STEST.BAT   -Batch file to test JTAG connections using SARAM method and L20.OUT.

Sample COFF files
- L20.OUT   -Sample COFF file for testing setup by programming 20 words of flash0.
- L16K0.OUT   -Sample COFF file for testing setup by programming all 16K words of flash0.
- L16K1.OUT   -Sample COFF file for testing setup by programming all 16K words of flash1.
- L32K.OUT   -Sample COFF file for testing setup by programming all 32K words.

Readme file
- README.PDF   -Acrobat version of this file.

   SRC Sub-directory with target control files (.ASM,.CMD,.LST,.MAP,.OBJ)
   - C2XX_BCX.ASM   -Control file for clear using B0 method.
   - C2XX_BEX.ASM   -Control file for erase using B0 method.
   - C2XX_BPX.ASM   -Control file for program using B0 method.
   - C2XX_BFX.ASM   -Control file for flash-write using B0 method.
   - C2XX_SPX.ASM   -Control file for clear/erase/program using SARAM method.
   - C2XX_BTX.ASM   -Control file for test program, linked for B0 or SARAM.
   - SVAR20.H   -Assembly include file for above code.

      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

When the TMX320F206 flash array is erased all bits are read as ones and the *program* operation is used to apply a pattern of zeros. For example, the TMX320F206 devices are shipped from Texas Instruments with some UART code programmed into the first few words of the flash0 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.

The JTAG based flash programmer provides a cost-effective method for programming the TMX320F206 flash. 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 port is provided. Each of the operations described above is performed entirely by the DSP core on the target system. The only function of the JTAG 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 TMX320F206 flash using an XDS510.

Figure 1. TMX320F2XX JTAG Based Programmer Setup.

*Note: Only PC based software is currently supported by TI.
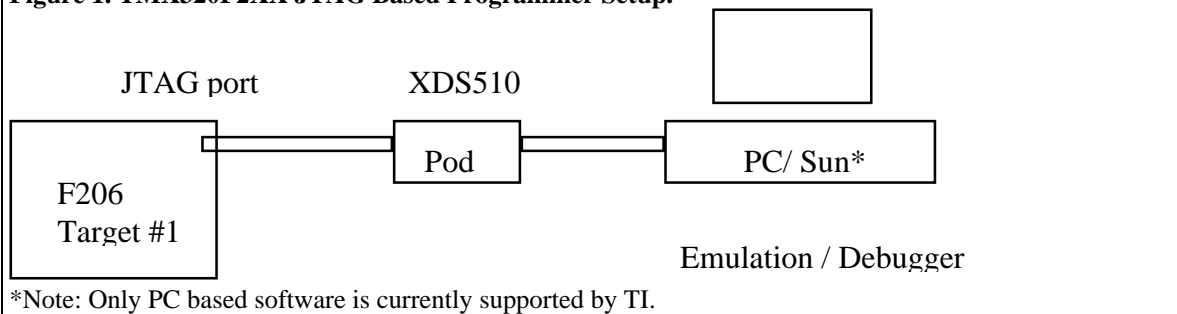
There are two implementations of the target code included with this programmer. One version uses only B0 ram for program storage and B1 ram for data storage while the other implementation ('F206 only) makes use of the 4K SARAM for additional storage. In the SARAM implementation the algorithms for *clear*, *erase*, and *program* are all loaded simultaneously allowing execution of the three operations in a single run of the host program. Although the B0 implementation is less efficient, requiring separate invocations of the host program for each operation, it can be very useful for debug and diagnostics. Also, the B0 implementation is the only option available for devices like the 'F240 that do not include SARAM.

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. In addition to the *clear, erase,* and *program* operations, a new batch file has been included in this release to perform the *flash-write* operation. The *flash-write* operation is used to recover devices that have been over-erased. The condition of a flash array that has been over-erased is known as *depletion* and this condition will inhibit re-programming of the array. The cause of this condition is usually operator error, since *depletion* should not occur under normal use. One mistake that can cause *depletion* is attempting to erase a flash array that is only partially programmed. Remember, the flash array MUST ALWAYS be *cleared* before erasing. The individual batch files for each operation are discussed in the next Subsection.

2.1 Description of Batch Files

As was mentioned earlier, there are two implementations of the target code included with this programmer. One implementation uses only B0 ram for program space and has the limitation of requiring separate invocations of the host program (PRG2XX.EXE) for each of the *clear*, *erase*, and *program* operations. The other implementation uses the 4K SARAM of the 'F206 for additional program and data space and allows multiple operations in one call. Batch files for both methods are included with the programmer and descriptions of each are given below.

Batch Files for Using the B0 Method

	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 0 array
BC1.BAT	; Clear - Flash 1 array
Note:	These files can be combined into one command line that clears both Flash0 and Flash1 by changing the argument of the -s option. For more details see Section 3.1.

	ERASE ALGORITHM
BE0.BAT	; Erase - Flash 0 array
BE1.BAT	; Erase - Flash 1 array
Note:	These files can be combined into one command line that erases both Flash0 and Flash1 by changing the argument of the -s option. For more details see Section 3.1.

	PROGRAM ALGORITHM
BP32K.BAT	; Program - Flash 0 and Flash1 with l32k.out
Note:	The file l32k.out is given as an example which will program both flash modules. It should be replaced with the COFF file to be programmed. A new feature has been added to this function which eliminates the need to provide separate COFF files for flash0 and flash1.

	FLASH-WRITE ALGORITHM
BFLW0.BAT	; Applies flash-write pulses to the Flash 0 array.
BFLW1.BAT	; Applies flash-write pulses to the Flash 1 array.
Note:	These files can be combined into one command line that performs the flash-write operation on both Flash0 and Flash1 by changing the argument of the -s option. For more details see Section 3.1.

Batch Files for Using the SARAM Method ('F206 Only)

	TEST PROGRAM
STEST.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 SARAM method.

	CLEAR and ERASE ALGORITHMS
SCE0.BAT	; Clear and Erase - Flash 0 array
SCE1.BAT	; Clear and Erase - Flash 1 array

Note1:	The SARAM implementation allows a clear and erase operation in one batch file, however if the clear operation fails the loader will exit without executing the Erase algorithm.
Note2:	These files can be combined into one command line that performs a clear and erase on both Flash0 and Flash1 by changing the argument of the -s option. For more details see Section 3.1.

	PROGRAM ALGORITHM
SP32K.BAT	; Program - Flash 0 and Flash1 with l32k.out

Note:	The file l32k.out is given as an example which will program both flash modules. It should be replaced with the COFF file to be programmed. A new feature has been added to this function which eliminates the need to provide separate COFF files for flash0 and flash1.

CLEAR, ERASE, AND PROGRAM ALGORITHM

CAUTION: The following programs should only be used after the individual flash functions are fully functional on the respective F206 hardware.

SCEP0.BAT	; Clear, Erase and Program - Flash 0 array
SCEP1.BAT	; Clear, Erase and Program - Flash 1 array

Note1:	The SARAM implementation allows a clear, erase, and program operation in one batch file, however if the clear or erase operations fail the loader will exit.
Note2:	These files can be combined into one command line that performs a clear, erase, and program on both Flash0 and Flash1 by changing the argument of the -s option. For more details see Section 3.1.
Note3:	The files l16k0.out and l16k1.out are given as examples for programming the arrays individually. These files should be replaced with the COFF files to be programmed. A new feature has been added to this function which eliminates the need to provide separate COFF files for flash0 and flash1.

2.2 Flash Programming Flow

1. Prepare your COFF file (flashcode.out) for programming the flash. **Note, a new feature of rev 2.0 allows both flash0 and flash1 to be programmed using a single COFF file.** The only restriction is that the COFF file must not include anything other then the code that will be programmed in the flash. Never include data sections in the COFF file that will be used to program the flash.

2. Verify that the JTAG connection is working by running one of the test programs – BTEST.BAT, or STEST.BAT. If an error is returned **DO NOT PROCEED** until the problem is resolved. Check the error conditions in Sub-section 2.3 to isolate the problem.

3. Check if the array(s) to be programmed is(are) erased , i.e. all locations contain 0xFFFFh. The 'F2XX samples are shipped from TI with flash1 erased and a boot code programmed in flash0.

4. If the flash array(s) to be programmed IS(ARE) ERASED then skip steps 5 and 6 and proceed to step 7. Else, if array(s) to be programmed IS(ARE) NOT ERASED proceed to step 5.

5. The array(s) must be cleared first before being erased. Run the CLEAR batch file(s) to clear the required array(s). The current version of the CLEAR algorithm will clear the entire flash array0 or 1. If (ERROR 114) occurs then exit to the step labeled **Depletion**. Else, if cleared proceed to step 6.

6. Run the ERASE batch file(s) to erase the required array(s). The current version of the ERASE algorithm will erase the entire flash array0 or 1. If (ERROR 114) occurs then exit to the step labeled **Depletion**. Else proceed to step 7. Note steps 5 and 6 can be combined if the SARAM method is used.

7. Modify one of the programming batch files (BP32K.BAT,SP32K.BAT) to use your specific COFF file (flashcode.out). Run the batch file to program the flash. If (ERROR 112) occurs then the algorithm has failed; Try once more starting at step 1. If no error occurs then the device is programmed. Note steps 5, 6, and 7 can be combined if the SARAM method is used (SCEP0/1.BAT).

    *****DONE*****

Depletion: If you are here then the array is either over-erased (in depletion) or otherwise bad. Try running the recovery batch files – BFLW0/1.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 has been recovered and must under-go the clear/erase/program sequence; proceed to step 5.

Note: if running the recovery algorithm three times does not recover the device then the flash array may be damaged; contact TI.

2.3 Error Messages

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

<u>**System Hangs**</u>
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.

<u>**Errors Related to JTAG Interface**</u>
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.

<u>**Errors Related to File Handling**</u>

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.

<u>**Errors Related to Flash Algorithms**</u>

ERROR 109 "Processor Timeout"
>   Software time-out expires before reaching SWI instruction.
>   - <u>Cause</u> - 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:

- <u>Cause</u> - 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.
- <u>Cause</u> - 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.
- <u>Cause</u> - 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:

- <u>Cause</u> - 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 Subsection 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.
- <u>Cause</u> - 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 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.

<u>**Other Errors**</u>

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 Programmer

Although the 'F206 JTAG Based Flash Programmer provides a complete solution for programming the 'F206 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 'F206 may share the scan chain with other devices. Yet another possible modification would be that the user wishes to incorporate the flash algorithms into the application for reprogramming of a flash module in the field. The information provided in the subsections that follow will give the user a number of options for optimizing the programmer.

3.1 JTAG loader Command format

  PRG2XX   -[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 Subsection 3.2, *Describing the Target System to the Loader*. Optional, default=c200.
- p	*port address*	Specifies IO address for XDS510 card in the PC. 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 0xFFE4.
- 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 PMST register at 0xFFE4, just use -m 0x0006; -i option is not required. Optional, default = no-load.
- o		No flashcode COFF file for programming
- t	*ST1 value*	Initialize ST1 register. Optional, default 0x17FC.
- e		Run PRG_erase function before executing PRG_prog function. Optional, default = not selected.
- v		Run PRG_verify function after executing PRG_prog function. Optional, not used with rev 2.0.
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.
                        See explanation below.

PRG_option

15	14	13 through 3	2	1	0
F1	F0	NOT USED IN REV 2.0	P	E	C

F0/1: Flash Module Select - Used to select which flash to
         perform the specified operation on. A one in these bit
         positions will select the corresponding flash module.
P/E/C: Flash Operation Select: Used for the SARAM
         implementation only and has no effect on the B0
         method. A one in these bit positions will select the
         corresponding operation (clear, erase, or program.)

Example:
The following command line can be used to clear, erase, and
program both flash arrays.
         prg2xx.exe -s C007 c2xx_spx.out flashcode.out

3.2 Describing the Target System to the Loader

In order for the JTAG 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 'F206 and no other devices, you can use the
  board.dat file that was included with the programmer. This file describes to the loader the single 'F2XX
  in the scan path and gives the 'F2XX the name C200. Since the loader automatically looks for the name
  C200 in the board.dat file, you can skip this subsection.
- 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.

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 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 Target Code Structure and Memory Usage

Control Files

The assembly programs contained in the SRC directory are loaded into the on-chip RAM of the 'F2XX and the DSP performs the actual flash operations. The control modules (C2XX_***.ASM) – assembly source provided – are used to call the appropriate algorithm and provide communication between the algorithms and the JTAG loader. The linker command files which are used to link the control modules with the appropriate algorithm are also included and provide information about target memory usage.

The following functions must exist in each control module (C2XX_***.ASM) and are used by the loader during flash operations:

        PRG_init   - Initializes the device with customer specific logic if necessary,
                        before running other functions.
        PRG_program - Default function executed after PRG_init.
        PRG_erase   - Used with -e option, currently used for *CLEAR* and *ERASE*.
        PRG_verify   - Used with -v option, currently not used.
        PRG_stop.   - Exit function with software break point after executing PRG_program

Algorithm Files

The algorithm files for the *clear*, *erase*, *program*, and *flash-write* operations are only given as object files in Rev2.0 of the programmer. The user is asked to use the algorithms only as provided by TI without any modification. The algorithms provided with this revision are still under development and undergoing TI qualification tests. When the 'F2XX devices are qualified as TMS grade products, then the algorithm source files and documentation will be released.

When performing a *clear*, *erase*, or *flash-write* operation, the algorithm is loaded and executed once. The *program* operation is different because it requires loading of the algorithm as well as the COFF file to be programmed. Since the 'F206 JTAG Based Programmer allows programming of all 32K flash locations, the COFF file must be divided into smaller blocks of code, making the JTAG loader a bottle-neck for the *program* operation. During the program operation the JTAG loader (PRG2XX) reads the COFF file to be programmed and loads the code into the target system in blocks of 200 words (B0 implementation) or 3000 words (SARAM implementation). Each time a block of code is loaded the programming algorithm is invoked, and if the block is programmed successfully, then the next block is loaded.

The calling convention described here is valid for Rev2.0 algorithms only. Since this programmer is still under development, TI reserves the right to change this convention in future revisions. The *clear*, *erase*, and *program* algorithm subroutines use the "caller save" software convention. Specifically, if machine state must be maintained the calling code will save and restore the registers. The *clear* and *erase* subroutines require the following global variables to be defined at link time – PROTECT, SEG_ST, SEG_END. Similarly, the *program* subroutine requires the following global variables at link-time – PROTECT, PRG_paddr, PRG_length, and PRG_bufaddr. See the control source files for an example.

The target memory requirements for the two implementations are outlined below.

Target Memory Used for B0 Method:
        B0 Ram:
            0xff00h - 0xffffh in program space for the algorithms (one operation at a time).
        B1 Ram:
            0x300h - 0x30fh as algorithm variable space.
            0x310h - 0x31fh as c2xxprog variable space
            0x320h - 0x3e7h as temporary 200 word buffer for code to be programmed.
        B2 Ram:

0x60h - 0x68h is reserved for PRG2XX.exe

Target Memory Used for SARAM Method ('F206 Only):
    B1 Ram:
        0x300h - 0x30fh as algorithm variable space.
        0x310h - 0x31fh as c2xxprog variable space
    B2 Ram:
        0x60h - 0x68h is reserved for PRG2XX.exe
    SARAM:
        NOTE: Uses SARAM in both program and data space.
        0x8000 - 0x83ff - 1Kx16 Program space for algorithms (all operations).
        0x0c00 - 0x17b8- 3000x16 Data space; 3000 word buffer for code to be programmed.