TMX320LC206/TMX320C206 Bootloader

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

This document describes the bootloader options available on the TMX320LC206 and TMX320C206. Several bootload options are available on these devices. The user can choose the option required by external pin configurations and an 8-bit word input from I/O address 0000h. The bootloader gives the flexibility of loading any executable code into the program memory of the DSP. User code can be transferred to the DSP program memory from any one of the following external sources:

- 8/16-bit transfer through the 'Synchronous Serial Port' (SSP)
- 8-bit transfer through the 'Asynchronous Synchronous Serial Port' (ASP)
- 8/16-bit EPROM
- 8/16-bit parallel port mapped to I/O space address 0001h of the DSP

In addition to this, a warm-boot is also supported.

2. Bootload options

The main function of the boot loader is to transfer user code from an external source to the program memory at power-up. The TMX320LC/'C206 provides several ways to "download" code to accommodate varying system requirements. To ensure compatibility, the 'C206 bootloader supports the original 'C203 bootload mode. The *EXT8* pin (pin 1) of the 'C206/LC206 is sampled during startup to determine whether the 'C203 bootload or the enhanced 206 bootload options are to be performed. Unlike the 'C203 bootloader , the 'C206 bootloader is capable of loading multiple sections of user code in different segments of memory. In all bootload modes, the processor automatically branches to the beginning of the user code, once bootloading is complete.

There are two possible scenarios for the TMS320LC/'C206 during startup based on the condition of the *EXT8* pin:

***EXT8* = low :** This invokes the original 'C203 style bootloader, which bootloads from an external 8-bit EPROM.

***EXT8* = high :** This invokes the enhanced 'C206 bootloader which supports the following bootload options:

- Synchronous Serial Port, 8/16 bit
- UART/Asynchronous Serial Port, 8 bit
- External Parallel EPROM, 8/16 bit
- Parallel I/O boot, 8/16 bit using $\overline{BIO}$ and *XF* for handshaking
- Warm-boot

The option to be executed is determined by reading the word at I/O address 0000h. The lower 8-bits of the word specify which bootloader option is to be executed.

3. Bootloader operation

If the $MP/\overline{MC}$ pin is sampled low during a hardware reset, execution begins at location 0000h of the on-chip ROM. This location contains a branch instruction to the start of the bootloader program. The level of the *EXT8* pin is read via bit 3 (LEVEXT8) in the PMST register (FFE4h in I/O space). If *EXT8* pin is read high, the bootloader will check the boot selection word at location 0000h in I/O space and determine which booting method to execute. If *EXT8* pin is read low, control will pass by default to 8 bit EPROM boot ('C203 style bootloader). This will allow upward compatibility from TMS320C203.

Figure 1. Program memory status register (PMST) register

15	14 - 4	3	2	1	0
FRDN	RESERVED	LEVEXT8	DON	PON	$MP/\overline{MC}$
R/W	0	R	R/W	R/W	R/W

Figure 2. PMST register – bit descriptions

Bit	Name	Value at Reset	Description
0	$MP/\overline{MC}$	x	Bit 0 latches in the state of MP/$\overline{MC}$ pin at reset. This bit can also be written to switch between Microprocessor (1) or Microcontroller (0) modes. This pin must be low to activate the bootloader.
1	PON	1	Bit 1 and 2 configure the SARAM mapping either in Program memory, data memory or both. At reset these bits are 11. DON (bit 2) PON (bit 1) 0 0 - SARAM not mapped, address in external memory 0 1 - SARAM in program memory at 0x8000h 1 0 - SARAM in data memory at 0x800h 1 1 - SARAM in progam and data memory (reset value)
2	DON	1	See above.
3	LEVEXT8	x	Bit 3 latches in the state of EXT8 pin at reset. It is a read only bit . If low, the on-chip bootloader uses 'C203 style bootload. If high, the enhanced 'C206 bootloader is used.
14 - 4	RESERVED	0	These bits are not used.
15	FRDN	0	At reset, this bit is 0, which enables enhanced RDN signal. If high, the inverted RNW is active. (Refer '206 datasheet for more details)

Figure 3. Bootloader-pin configuration

MP / $\overline{MC}$	EXT8	Option	Mode(s)
0	0	use ‘C203 style bootloader	1
0	1	use ‘C206 enhanced bootloader	2 to 9
1	0	EXT8 has no effect	-
1	1	EXT8 has no effect	-

The bootloader sets up the CPU status registers as follows:

- On–chip DARAM block B0 is mapped into program space (CNF = 1).
- On–chip SARAM block is mapped into program and data space (PON = 1, DON=1).

Note that both B0-DARAM and SARAM memory blocks are enabled in program memory space; this allows you to transfer code to on–chip program memory.

At reset, interrupts are globally disabled (INTM = 1).Entire program and data memory spaces are enabled with seven wait states.

Figure 4 lists the available bootload options and the corresponding values for the boot routine selection word at I/O address 0000h. This word could be set by a DIP-switch.

4. ‘C206 Enhanced bootloader (EXT8 high – Modes 2 to 9**)**

The bootloader reads the I/O port address 0000h by driving the I/O strobe ($\overline{IS}$) signal low. The lower eight bits of the word read from I/O port address 0000h specify the mode of transfer; the higher eight bits are ignored. This ‘boot routine selection’ (BRS) word determines the boot mode. The BRS word uses a 6-bit source address field (SRCE_AD) in parallel EPROM mode and a 6-bit entry address field (ADDR_bb) in warm-boot mode to arrive at the starting address of the code.

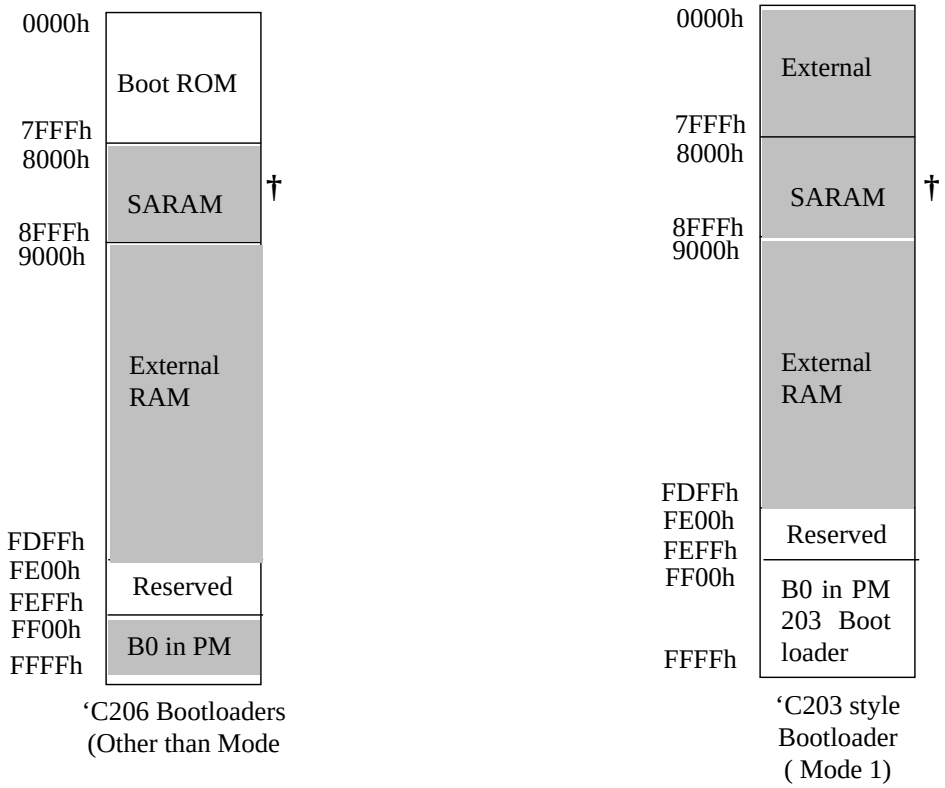
Figure 4. Enhanced ‘C206 Bootloader options

BRS word @ I/O 0000h			Boot load option	Mode
xxxxxxxx	xxx0	0000	8-bit serial SSP, ext FSX, CLKX	2
xxxxxxxx	xxx0	0100	16-bit serial SSP, ext FSX,CLKX	3
xxxxxxxx	xxx0	1000	8-bit parallel I/O	4
xxxxxxxx	xxx0	1100	16-bit parallel I/O	5
xxxxxxxx	xxx1	0000	8-bit ASP /UART	6
xxxxxxxx	SRCE	AD01	8-bit EPROM	7
xxxxxxxx	SRCE	AD10	16-bit EPROM	8
xxxxxxxx	ADDR	bb11	warm-boot	9

Figure 5 provides the memory map of program address spaces that are accessible through the boot loader. For modes other than 1, memory locations from 0000h to 7FFFh are not available

for loading user code since that space is occupied by ROM. However this limitation can be overcome by the user by modifying the memory map in the user's own boot code.

Figure 5. Destination address space for programs in program memory



Legend : - Memory locations available for bootloading user code.

† Caution : Locations 8000h – 807Fh in SARAM are reserved for the second interrupt vector table as mentioned in Section 5. Exercise caution while moving code into this area.

Note: The remapped interrupt vectors are not uniformly spaced and some interrupt vectors have additional space embedded between them. This is to enable the user to locate an 'Interrupt service routine' within the 'Secondary interrupt vector table (SIVT)' itself. This eliminates the need for an additional "branch" instruction in the SIVT, thereby reducing the interrupt latency.

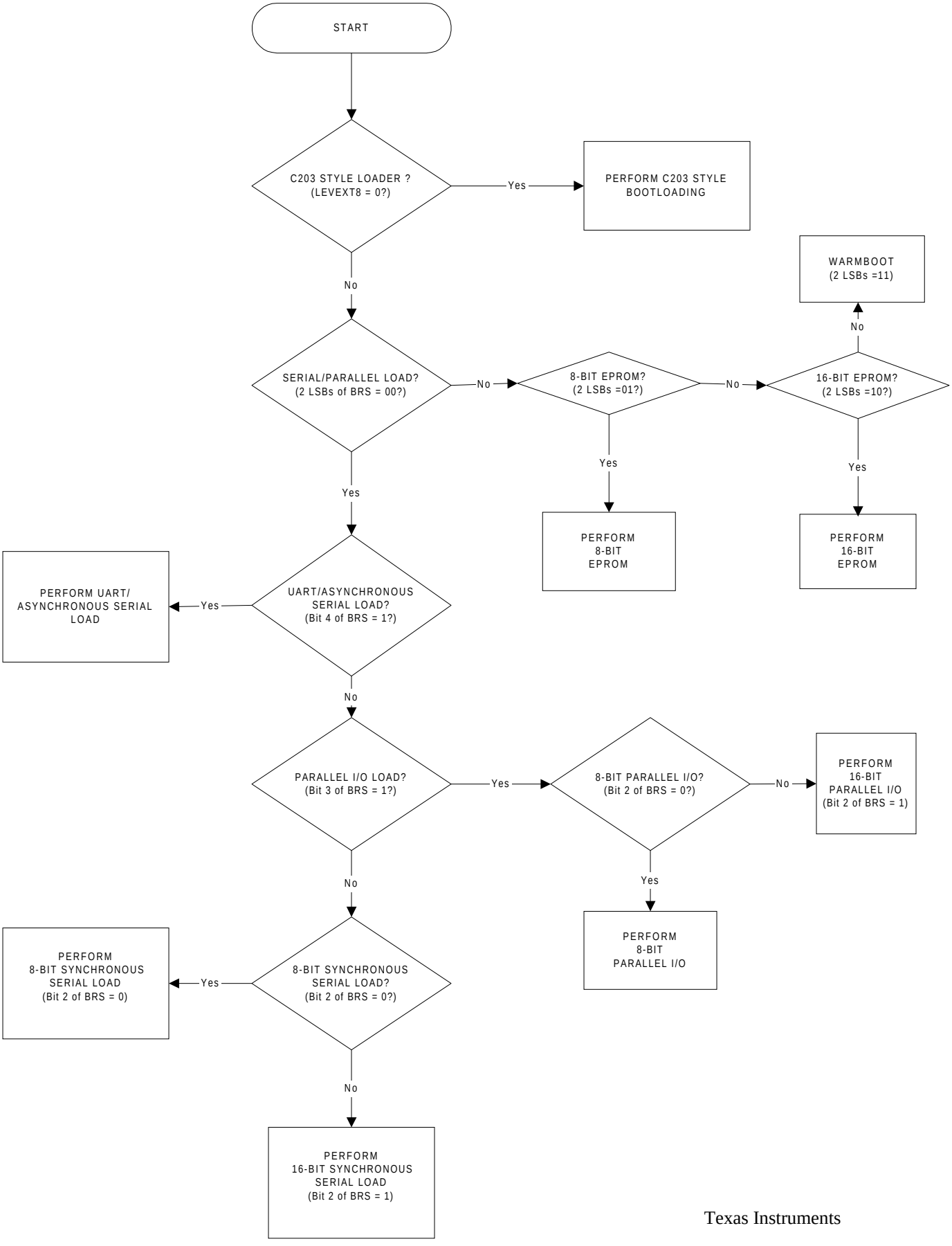
5. Interrupt Vectoring

Interrupt vectors stored in the on-chip ROM have hard coded addresses to the on-chip SARAM starting at address 8000h in program space. When an interrupt occurs, a branch is made to the corresponding interrupt vector located in the on chip ROM at addresses (0000h- 0040h). A branch instruction then transfers program control to the second interrupt vector table in the on-chip SARAM. The second interrupt vector table must be initialized by the user. This is used to allow remappable interrupt vectors. See below for interrupt vectors in the SARAM.

Remapped interrupt vectors for TMS320C206, TMS320LC206

```
int1_holdv      .set   8000h     ; User maskable interrupt #1
int2_3v         .set  8002h     ; User maskable interrupts #2 & #3
tintv           .set   8004h     ; Timer interrupt vector
rintv           .set   801Ah     ; SSP receive interrupt vector
xintv           .set   8032h     ; SSP transmit interrupt vector
txrxintv        .set  804Eh     ; UART port Tx/Rx interrupt vector
trapv           .set   8050h     ; Software trap vector
nmiv            .set   8052h     ; Non-maskable interrupt vector
swi8v           .set   8054h     ; Software interrupt vectors begin...
swi9v          .set   8056h
swi10v          .set  8058h     ; (Note:If these interrupts are unused
swi11v          .set  805Ah     ; these memory locations may be
swi12v          .set  805Ch     ; used for other purposes.)
swi13v          .set  805Eh
swi14v          .set  8060h
swi15v          .set  8062h
swi16v          .set  8064h
swi20v          .set  8066h
swi21v          .set  8068h
swi22v          .set  806Ah
swi23v          .set  806Ch
swi24v          .set  806Eh
swi25v          .set  8070h
swi26v          .set  8072h
swi27v          .set  8074h
swi28v          .set  8076h
swi29v          .set  8078h
swi30v          .set  807Ah
swi31v          .set  807Ch
reserved        .set   807Eh
```

Figure 6 Bootload flowchart



6. Synchronous serial port (SSP) boot mode

In this mode, the ‘Synchronous serial port control register’ (SSPCR) is configured for 16–bit or 8-bit word transfer. The data shift clock and frame sync must be supplied by the external device to the ’LC/’C206.

6.1 16–Bit Word Serial Transfer (Mode 3)

If the 16–bit word transfer is selected , the first 16–bit word received by the ‘C206 from the serial port specifies the destination address ($Destination_{16}$) of code in program memory. The next 16–bit word specifies the length ($Length_{16}$) of the actual code that follows. These two 16–bit words are followed by N number of code words to be transferred to program memory. Note that the number of 16–bit words specified by the parameter N does not include the first two 16–bit words received ($Destination_{16}$ and $Length_{16}$). After the specified number of code words are transferred to program memory, the ’C206 checks to see if there are any more sections to be transferred. If there are additional sections to be transferred, the bootloader proceeds to transfer them in exactly the same way as the first section. After transferring all the sections, the ‘C206 branches to the first destination address. The length N is defined as:

$$N = (\text{Number of 16-bit words}) - 1$$

If, after transferring all the N words of a section, the ‘C206 receives a 0000, it signals the end of user code. If any word other than 0000 is read, it indicates that one or more sections is following and the word read is treated as the destination address of the next section. Refer figure 7 for the format of data transfer in 16-bit mode.

Figure 7. 16–Bit Word Transfer

$DESTINATION_1$
LENGTH of first section (N_1)
CODE(1) of length N_1
$DESTINATION_2$
LENGTH of second section (N_2)
CODE(2) of length N_2
$DESTINATION_N$
LENGTH of N^{th} section (N_N)
CODE(N) of length N_N
0000 to end program

Legend:

$Destination_{16}$	16–bit destination address
$Length_{16}$	16–bit word that specifies the length of the code (N) that follows
$Code(N)_{16}$	N number of 16–bit words to be transferred (actual code)

6.2 8–Bit Word Serial Transfer (Mode 2)

If the 8–bit word transfer is selected , a higher–order byte and a lower–order byte form a 16–bit word. The first 16–bits received by the ‘C206 from the serial port specifies the destination address ($Destination_h$ and $Destination_l$) of code in program memory. The next 16–bits specify the length ($Length_h$ and $Length_l$) of the actual code that follows. These two 16–bit words are followed by N number of code words to be transferred to program memory. Note that the number of 16–bit words specified by the parameter N does not include the first four bytes (first two 16–bit words) received (Destination and Length). After the specified number of code words are transferred to program memory, the ’C206 checks to see if there are any more sections to be transferred. If there are additional sections to be transferred, the bootloader proceeds to transfer them in exactly the same way as the first section. After transferring all the sections, the ‘C206 branches to the first destination address. The length N is defined as:

N = (Number of 16-bit words) – 1
or
N = (Number of bytes to be transferred/2) – 1

If, after transferring all the N words of a section, the ‘C206 receives a 0000, it signals the end of user code. If any word other than 0000 is read, it indicates that one or more sections is following and the word read is treated as the destination address of the next section. Refer figure 9 for the format of data transfer in 8-bit mode. Figure 8 shows the connection details for SSP bootload option.

Figure 8 Host-’C206 interface for SSP bootload option

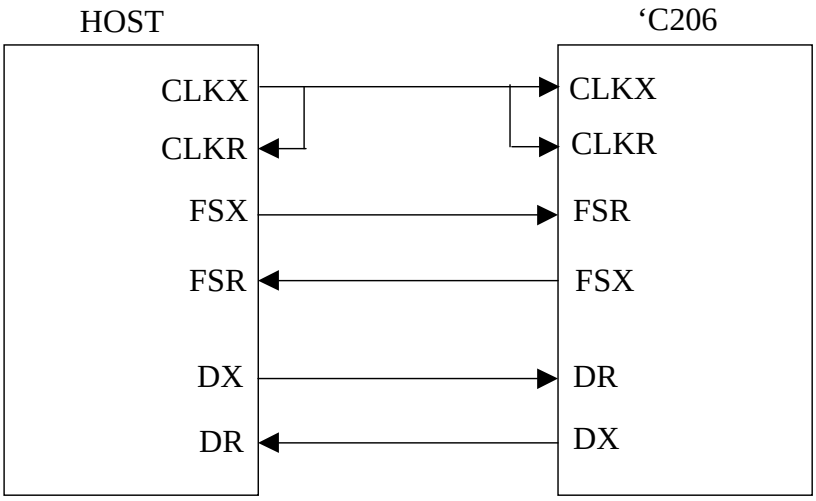


Figure 9. 8–Bit Word Transfer

$DESTINATION_{1h}$
$DESTINATION_{1l}$
$LENGTH_h$ of first section (N_{1h})
$LENGTH_l$ of first section (N_{1l})
$CODE(1)_h$
$CODE(1)_l$
$DESTINATION_{2h}$
$DESTINATION_{2l}$
$LENGTH_h$ of second section (N_{2h})
$LENGTH_l$ of second section (N_{2l})
$CODE(2)_h$
$CODE(2)_l$
$DESTINATION_{Nh}$
$DESTINATION_{Nl}$
$LENGTH_h$ of N^{th} section
$LENGTH_l$ of N^{th} section
$CODE(N)_h$
$CODE(N)_l$
0000 to end program

Legend:	$Destination_h$	High byte of destination address
	$Destination_l$	Low byte of destination address
	$Length_h$	High byte that specifies the length of the code (N) that follows
	$Length_l$	Low byte that specifies the length of the code (N) that follows
	$Code\ (N)_h$	High byte of N number of 16–bit words to be transferred
	$Code\ (N)_l$	Low byte of N number of 16–bit words to be transferred

7. UART/Asynchronous serial port (ASP) boot mode (Mode 6)

This mode is extremely useful to transfer user code to the ‘206 through an asynchronous serial port such as the RS-232 port available in personal computers. The data packet format in this mode is similar to that of Synchronous serial port (SSP) boot mode, with the exception that only 8-bit transfers are supported. The DSPHEX utility is used to convert the COFF file (*.out) of the user to a hex file suitable for UART bootloading. For more information about the DSPHEX utility, refer to *TMS320C1x/C2x/C2xx/C5x Assembly Language Tools – User’s Guide (SPRU018D)*

The ‘206 is capable of sensing the baud rate of the incoming data and automatically update its baud-rate register. To make this happen, the host must transmit the ASCII character “a” (or “A) in the very beginning of data transfer. ‘C206 boot code will lock to the incoming baud rate and then prepare itself to receive user code. The DSPHEX utility does not automatically add the ASCII value of the character “a” in the hex file it creates. This can be easily done by the user

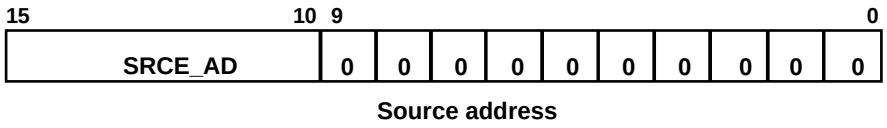
with the help of any ASCII editor. While editing the hex file, the user should also verify that the last word of the file is 0000h in order to transfer control to the user code after bootloading. The options for the DSPHEX utility can be either specified on the command line or with he help of a command file. A sample command file for the DSPHEX utility is given below:

```
/* DSPHEX command file to generate hex file from .out file */
/* suitable for UART bootloader                            */


usercode.out     /* Replace with the actual name of user code */
-a               /* ASCII- hex format                         */
-o usercode.hex  /* Replace with the reqd. name of user code  */
-byte            /* default                                   */
-order MS        /* default                                   */
-memwidth 8
-romwidth 8

SECTIONS
{ .text   : boot }
```

8. Parallel EPROM boot mode

The parallel EPROM boot mode is used when code is stored in EPROMs (8–bit or 16–bit wide). The code is transferred from external global data memory (starting at the source address) to program memory (starting at the destination address). The EPROMS can be mapped anywhere in global memory, but their starting address should lie on a 400 word boundary. The six MSBs of the source address are specified by the SRCE_AD field of the boot routine selection word. A 16–bit source address is formed with the help of this SRCE_AD field as shown in Figure 10. The bootload code will initialize the GREG register to external global data memory space 8000h-0FFFFh. The ’LC/’C206 transfers control to the source address after disabling global data memory.

Figure 10. 16–Bit Source Address for Parallel EPROM Boot Mode



Legend: SRCE_AD = 6–bit page address

8.1 16–Bit EPROM transfer (Mode 8)

If the 16–bit mode is selected, boot code will be read in 16–bit words starting at the source address. The source address is incremented by 1 after every read operation. The first 16–bit word read from the source address specifies the destination address ($Destination_{16}$) of code in program memory. The next 16–bit word specifies the length ($Length_{16}$) of the actual code that follows. These two 16–bit words are followed by N number of code words to be transferred to program memory. Note that the number of 16–bit words specified by the parameter N does not include the first two 16–bit words received ($Destination_{16}$ and $Length_{16}$). After the specified number of code words are transferred to program memory, the ’C206 checks to see if there are any more sections to be transferred. If there are additional sections to be transferred, the bootloader proceeds to transfer them in exactly the same way as the first section. After transferring all the sections, the ‘C206 branches to the first destination address. The length N is defined as:

$$N = (\text{Number of 16-bit words}) - 1$$

If, after transferring all the N words of a section, the ‘C206 receives a 0000, it signals the end of user code. If any word other than 0000 is read, it indicates that one or more sections is following and the word read is treated as the destination address of the next section. Refer figure 7 for the format of data transfer in 16-bit mode.

Note: There is at least a 4–instruction–cycle delay between a read from the EPROM and a write to the destination address. This delay ensures that if the destination is in external memory (for example, fast SRAM), there is enough time to turn off the source memory (for example, EPROM) before the write operation is performed.

8.2 8–Bit EPROM Transfer (Mode 7)

If the 8–bit mode is selected, two consecutive memory locations (starting at the source address) are read to form a 16–bit word. The high–order byte of the 16–bit word is followed by the low–order byte. Data is read from the lower eight data lines, ignoring the higher byte on the data bus. The first 16–bit word specifies the destination address ($Destination_h$ and $Destination_l$) of code in program memory. The next 16–bit word specifies the length ($Length_h$ and $Length_l$) of the actual code that follows. These two 16–bit words are followed by N number of code words to be transferred to program memory. Note that the number of 16–bit words specified by the parameter N does not include the first four bytes (first two 16–bit words) received (Destination and Length). After the specified number of code words are transferred to program memory, the ’C206 checks to see if there are any more sections to be transferred. If there are additional sections to be transferred, the bootloader proceeds to transfer them in exactly the same way as the first section. After transferring all the sections, the ‘C206 branches to the first destination address. The length N is defined as:

$$N = (\text{Number of 16-bit words}) - 1$$
$$\text{or}$$
$$N = (\text{Number of bytes to be transferred}/2\ ) - 1$$

If, after transferring all the N words of a section, the ‘C206 receives a 0000, it signals the end of user code. If any word other than 0000 is read, it indicates that one or more sections is following

and the word read is treated as the destination address of the next section. Refer figure 9 for the format of data transfer in 8-bit mode.

Note: There is at least a 4–instruction–cycle delay between a read from the EPROM and a write to the destination address. This delay ensures that if the destination is in external memory (for example, fast SRAM), there is enough time to turn off the source memory (for example, EPROM) before the write operation is performed.

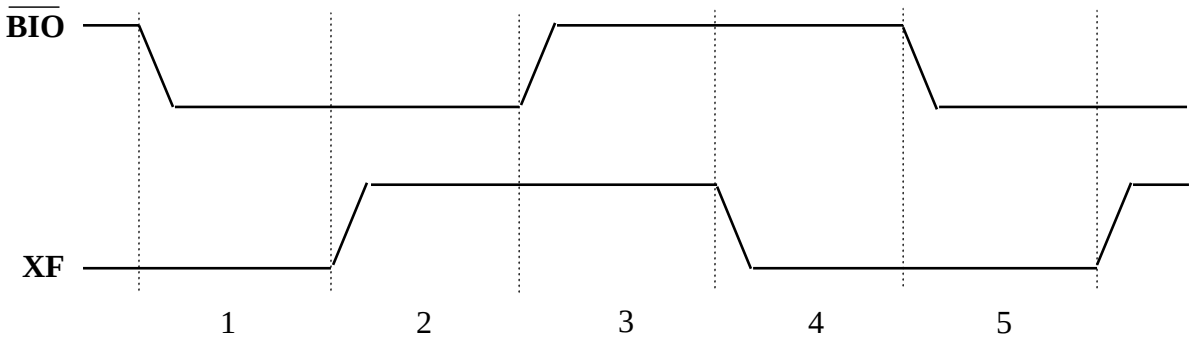
9. Parallel I/O boot mode (Mode 4 – 8 bit, Mode 5 – 16 bit)

The parallel I/O boot mode asynchronously transfers code from I/O port at address 0001h to internal or external program memory. Each word can be 16 bits or 8 bits long and follows the same sequence outlined in parallel EPROM mode. The ’LC/’C206 communicates with the external device using the BIO and XF lines for handshaking. This allows a slower host processor to communicate with the ’LC/’C206 by polling/driving the XF and BIO lines. The handshake protocol shown in Figure 11 must be used to successfully transfer each word via I/O port 0001h.
If the 8–bit boot mode is selected, two consecutive 8–bit words are read to form a 16–bit word. The high–order byte of the 16–bit word is followed by the low–order byte. Data is read from the lower eight data lines of I/O port 0001h, ignoring the higher byte on the data bus.

A data transfer is initiated by the host, driving the BIO pin low. When the BIO pin goes low, the ‘C206 inputs the data from I/O address 0001h, drives the XF pin high to indicate to the host that the data has been received and then writes the input data to the destination address. The ‘C206 then waits for the BIO pin to go low before driving the XF pin low. The low status of the XF line can then be polled by the host for the next data transfer.

There is at least a 4–instruction–cycle delay between the XF rising edge and a write operation to the destination address. This delay ensures that if the destination is in external memory (for example, fast SRAM), the host processor has enough time to turn off the data buffers before the write operation is performed. The ’C206 accesses the external bus only when XF is high.

Figure 11. Handshake Protocol

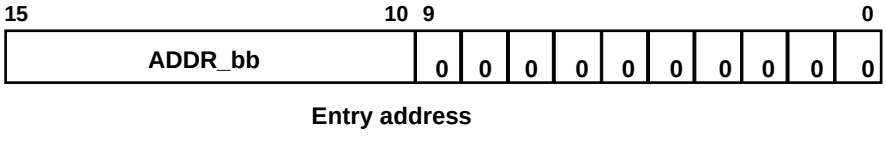


Legend : 1 – Host requests data transfer to 'C206 by making $\overline{\text{BIO}}$ low.
2 – 'C206 reads in the data through I/O port 1 and makes XF high. Bootloader program loops until $\overline{\text{BIO}}$ becomes high.
3 – After $\overline{\text{BIO}}$ is made high, bootloader acknowledges by making XF low indicating that it is ready for new data.
4 – Bootloader program loops until $\overline{\text{BIO}}$ becomes low. XF continues to be low.
5 – When $\overline{\text{BIO}}$ becomes low, it signals the host request for the transmission of the next word and the whole sequence repeats until all words are transferred.

10. Warm-boot mode (Mode 9)

The warm-boot operation does not move any code. It is useful to branch to the user code if the code has already been transferred to internal or external program memory by other bootload methods. This mode is used only if a "warm" device reset is required. Since warm-boot mode can be invoked only in the microcontroller mode, the first section of user code can reside only from 8000h onwards in program memory as 0000h to 7FFFh is occupied by ROM. The six MSBs of the entry address are specified by the ADDR_bb field of the boot routine selection word (Figure 4). A 16–bit entry address is defined by this ADDR_bb field as shown in Figure 12. Since bits 0 – 9 are zero, the starting address must lie on 400 word boundaries (x000h, x400h, x800h and xC00h). The address of the user code must be specified in the BRS word for warm-boot to function. The 'LC/'C206 transfers control to the entry address after disabling global data memory.

Figure 12. 16–Bit Entry Address for Warm-boot Mode



Legend: ADDR_bb = 6–bit page address in 400h word boundaries

11. ‘C203 style bootloader (EXT8 low - Mode 1)

The ‘C206 bootloader supports the ‘C203 style bootloader when the EXT8 pin is tied low. However, there are some differences between the original ‘C203 device bootloader and the ‘C203 style bootloader option supported in the ‘C206. (This paragraph applies to the ‘C203 device bootloader only). The bootloader option in the TMS320C203 device has a fixed destination address for the user code. This address must be 0000h, as the interrupt vector table must be modified first. The reset vector (0000h in program memory) must be initialized to point to the beginning of the user code. Other interrupt vectors may need to be setup depending on the user application. After the user code is bootloaded (i.e. application code transferred to external program memory), *INTR 0* instruction is invoked by the bootloader. This transfers program control to user code. The boot source address (the address at which user code is stored in external non-volatile memory) is fixed at 8000h in global memory space.

The bootloader in TMX320LC/’C206 devices features an 8-bit boot option from external non-volatile memory (EPROM) to external SRAM or internal memory at reset if $MP/\overline{MC}$ pin is sampled low and EXT8 pin is tied low during a hardware reset. This mode is similar in operation to the original ‘C203 device bootloader except during the final branch. There is no *INTR 0* instruction, rather program control branches to the address specified by the accumulator.

The maximum size of the EPROM can be 32K x 8 bits to yield 16K x 16 bits of program memory. However, users could boot in their own bootloader, which would perform a function as desired. The bootloader will begin loading from a fixed source address 8000h in external global data space and begin transferring to the destination address in program space defined by the user. This destination address is defined by the first two bytes stored in the EPROM. The destination address is not constrained to be 0000h as in the case of ‘C203 device and can be any valid program address. However, the interrupt vector table may need to be modified by the user.

At reset, interrupts will be globally disabled, INTM = 1, B0 will be mapped to program space, CNF = 1, and seven wait states will be selected for program and data spaces. The bootload code will initialize the GREG register to external global data memory space 8000h-FFFFh. The operation of this mode is similar to 8-bit EPROM transfer (‘C206 boot mode – 7). The status registers ST0 and ST1 do not contain their reset values when the boot loader transfers program control to the user code. The users must initialize these registers at the beginning of their code with suitable values appropriate for their applications.

12. µ-law, A-law table

µ-law and A-law are encoding formats used for the quantization and digitization of analog signals into Pulse Code Modulation (PCM) signals and recovery of the original analog signals. This companding scheme is extensively used in telecommunication applications. This scheme specifies the parameters for compression and re-expansion of the signals during signal transmission and processing. The look-up table for performing the µ-law, A-law conversion is stored in the ROM of ‘C206. The A-law table is stored at 400h in the on-chip ROM. The µ-law table follows the A-law table and starts at 500h.