MOTOROLA
Semiconductor Application Note

Order by AN1781/D
(Motorola Order Number)
Rev. 0 , 11/98

Booting DSP563xx Devices Through the Serial Communication Interface (SCI)

By Tina M. Redheendran

The DSP563xx bootloader code allows the DSP to load an application program and data through the SCI to X, Y, and P memory and begin executing the downloaded program upon reset of the DSP. The bootloader also allows you to program various DSP control registers before executing the downloaded program. The bootloader is necessary for applications in which the user must program control registers or download data to X or Y memory upon reset of the DSP.

This application note applies to all DSP56300 family devices. It describes the steps in using the bootloader code: resetting the DSP, downloading the bootloader, and running the bootloader. This document also contains overviews of the SCI, the DSP mode pins and operation modes, and the internal bootstrap ROM code.

Contents

1 SCI Setup

The bootloader code loads the application program to the DSP through the SCI. Thus, the SCI must be set up correctly for the bootloader code to operate correctly. SCI setup involves connecting the SCI pins and programming the SCI control registers. **Figure 1** shows how to connect the SCI pins. The SCI Receive Data Pin (RXD) is an input to the DSP and receives data from the external device. This pin must connect to the transmit pin of the external device. The SCI Transmit Data Pin (TXD) is an output from the DSP and transmits data to the external device. This pin must connect to the receive pin of the external device. The SCI Serial Clock Pin (SCLK) is an input or an output to the DSP (depending on the control register setting) and provides the serial bit rate clock for the SCI. For this application, the SCLK pin functions as an input to the DSP. This pin must connect to the transfer clock pin of the external device.

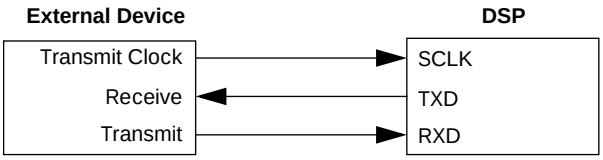

Figure 1. SCI Connections

The SCI is programmed by two 24-bit control registers:

- SCI Control Register (SCR) — The SCR controls the main SCI operation. The external device must be programmed to interface with the SCI as defined by the SCR settings. The SCR is set to $000302 for the bootstrap and bootloader code. Bits 0-2, Word Select (WDS), are set so that each SCI transfer contains 10 bits: 1 start bit, 8 data bits, and 1 stop bit (no parity). Bit 3, Shift Direction (SSFTD), is set so that the SCI transfers occur least significant bit first. Bits 8 and 9 are set to enable the SCI transmitter and receiver. The remaining bits of the SCR are not important for this application and are cleared.
- SCI Clock Control Register (SCCR) — The SCCR controls the SCI clock operation. The external device must be programmed to generate the appropriate clock for the transfers. The SCCR is set to $00C000 for the bootstrap and bootloader code. Bits 14 and 15, Receiver and Transmitter Clock Mode Source (RCM and TCM), determine the clock source for the SCI. Since both of these bits are set, the clock source for the SCI transfers is external to the SCI and is obtained from the SCLK pin. The clock frequency must be 16x the transfer baud rate. The remaining bits of the SCCR are not important for this application (because the internal SCI clock logic is not used) and are cleared.

2 Booting Steps

As **Figure 2** shows, booting the DSP with the bootloader code occurs in three steps:

1. The DSP is reset.
2. The internal bootstrap code runs on the DSP, and the bootloader is downloaded from the external device, which can be any device, including a PC or another DSP. The bootstrap code is contained in the DSP's 192-word boot ROM and automatically runs when the DSP is reset. This code loads a program segment from one of a variety of sources including the SCI (depending on how the mode pins are set) and then begins executing the downloaded program segment.
3. The bootloader runs on the DSP. The bootloader code can load data to X or Y memory, load an application program segment to P memory, or begin executing a downloaded program segment.

3 Resetting the DSP

In the first step of using the bootloader code, the external device resets the DSP, as **Figure 2** shows. The external device asserts the $\overline{\text{RESET}}$ pin of the DSP, placing the DSP in the reset state. Then the external device deasserts the $\overline{\text{RESET}}$ signal. In response, the DSP leaves the reset state and the values of the mode pins, MODA, MODB, MODC, and MODD, are loaded into bits MA, MB, MC, and MD of the Operating Mode Register (OMR). These bit settings determine the DSP operating mode and the bootstrap code option the DSP uses to start up. Finally, the DSP automatically jumps to the bootstrap code entry point, which is also determined by the mode pin settings.

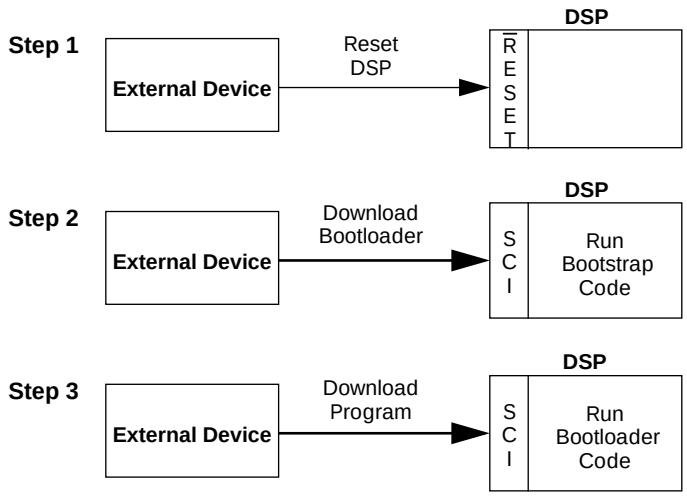

Figure 2. Booting Steps

For this application note, the mode pins must be set in the configuration shown in **Table 1**, which directs the DSP to boot through the SCI. This configuration also directs the DSP to jump to the bootstrap ROM beginning at program memory location $FF0000.

Table 1. Mode Pin Settings for Booting Through the SCI

Mode Pin	Value
MODA	1
MODB	0
MODC	1
MODD	0

4 Downloading the Bootloader

In the second step of using the bootloader code, the external device downloads the bootloader to the DSP running the bootstrap code, as shown in **Figure 2**. This step begins when the DSP starts executing the bootstrap code. The bootstrap code reads the MA, MB, MC, and MD pins in the OMR and determines which section of the bootstrap code to execute. If the mode pins are set as described in **Section 3,** "Resetting the DSP," the DSP jumps to the section that loads the program memory through the SCI interface.

The SCI bootstrap code expects to receive 3 bytes for each word: least significant byte first followed by the middle byte and then the most significant byte. The first word the bootstrap code receives is the number of program words to load, NUMBER. The second word is the starting address, START. Then the bootstrap expects to receive the program words. The bootstrap code receives NUMBER program words and places them in program memory starting at START. After the DSP receives each byte, DSP, it is echoed back to the external device through the SCI. When all program words are loaded (all NUMBER of them), the bootstrap code begins executing the downloaded program starting at START.

The number of program words in the bootloader code is 160 (or $A0). Thus, for this application, the external device must first transmit $0000A0 (least significant byte first). Then, the external device must transmit three bytes for the starting address of the bootloader code. The starting address can be anywhere in program memory as long as it does not interfere with the location of the application program to be downloaded. Finally, the external device must transmit three bytes for each of the 160 program words in the bootloader program. The bootloader program words can be derived from the bootloader code shown in the Appendix. When all 160 program words are loaded, the bootstrap code begins executing the bootloader code.

5 Running the Bootloader

In the final step of using the bootloader code, the external device downloads the application program to the DSP running the bootloader code, as shown in **Figure 2**. This step begins when the DSP begins executing the bootloader code. **Figure 3** shows the flow of the bootloader program. First, the bootloader receives one word and transmits that word back to the external device. The actual value of this word is irrelevant. The purpose of this word is to initiate the transfers. The bootloader transmits the version number of the bootloader code and the value of the DSP's Identification Register (IDR). Then, the bootloader begins receiving values for the DSP control registers. It receives (and echoes back) values for the following registers and writes the received values to the appropriate registers:

- PLL Control Register (PCTL)
- Operating Mode Register (OMR)
- Status Register (SR)
- Address Attribute Registers 0-3 (AAR0-3)
- Bus Control Register (BCR)
- SCI Clock Control Register (SCCR)

Again the bootloader receives one word that is echoed back, the actual value of which is irrelevant. The purpose of this word is to re-initiate the transfers after the control registers are written. Next, the bootloader receives (and echoes back) three direction words that tell the bootloader what to do next. The first word describes what action to take. This word should be a number between zero and three. The action for each of these values is as follows:

- 0 = Download to P Memory
- 1 = Download to X Memory
- 2 = Download to Y Memory
- 3 = Run the code

The second word the bootloader receives is the number of words to be downloaded, NUMBER. This word is ignored if the previous word instructed the bootloader to run the code (action word = 3). The third word the bootloader receives is the starting address, START. This is the memory address where the bootloader begins downloading the data or running the code.

If one of the download actions is chosen, the program begins receiving the data without echoing the data back to the external device. The bootloader receives NUMBER words and places them in the appropriate memory space beginning at START. When all words are downloaded (all NUMBER of them), a 24-bit checksum is transmitted back to the external device. Then, the bootloader jumps back to the section where the action is received, so more data can be downloaded or the program can be run.

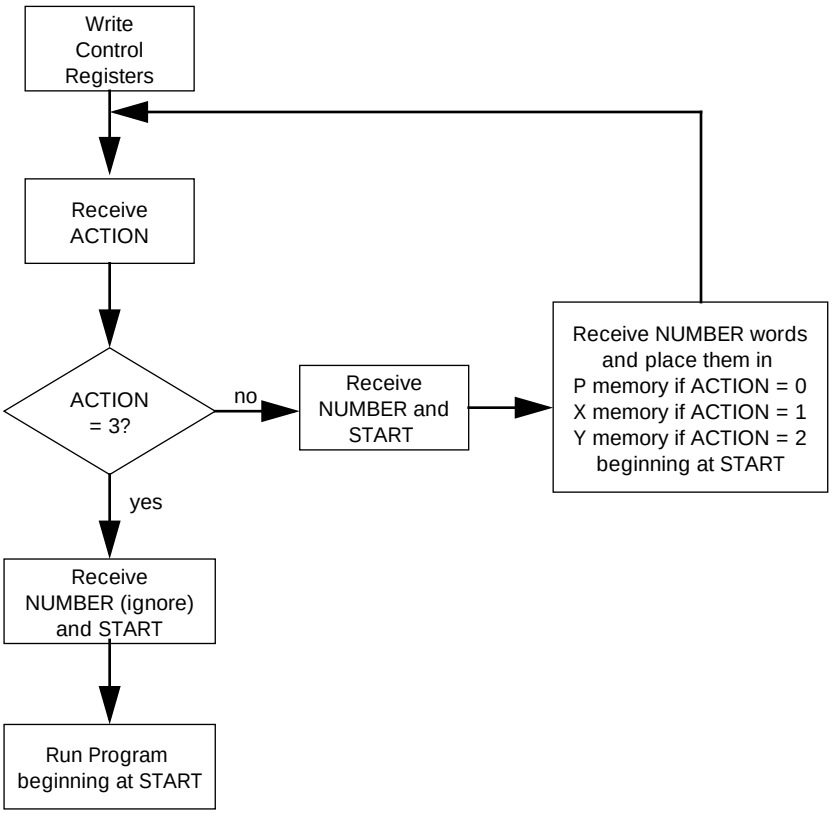

Figure 3. Bootloader Program Flow

If the run code action is chosen, the bootloader begins executing the downloaded program beginning at START. If the received action word is not a number between zero and three, the bootloader jumps back to the section where the action is received and waits for another three direction words. The external device must transmit the correct sequence of words to the bootloader code running on the DSP. The proper sequence of words depends on which memory spaces need to be loaded and where the application code should begin running.

Appendix: Bootloader Code

```
;------------------------------------------------------------------------
; bootload.asm
;------------------------------------------------------------------------
        opt cc,mu
;------------------------------------------------------------------------
;       Copyright (C) 1998 Motorola
;
;       Wireless Infrastructure Systems Division
;       Networking & Computing Systems Group,
;       Semiconductor Products Sector
;
;       This code was originally written by
;       Signals & Software Ltd
;       3 Jardine House
;       Harrovian Business Village
;       Bessborough Road
;       Harrow
;       Middx HA1 3EX
;       United Kingdom
;       Tel: 0181 423 6469
;       Fax: 0181 869 1182
;       WWW: www.sasl.com
```

```
;       E-mail: Charles.Cox@sasl.com
;
;-------------------------------------------------------------------------

        nolist
        include 'ioequ.asm'
        list

        section Bootload

;-------------------------------------------------------------------------
; External Variable Definitions
;-------------------------------------------------------------------------
; definitions of variables that are accessible by other routines external
; to this section
         xdef   Bootload                    ;Program start address

;-------------------------------------------------------------------------
; Local defines
;-------------------------------------------------------------------------
VERSION         equ     $000100

;-------------------------------------------------------------------------
; Local Scratch Variable Definitions
;-------------------------------------------------------------------------
                org     Y:0

Version         ds      1                  ;Storage for program version number
T_Pctl          ds      1                  ;Storage for control register values
T_Omr           ds      1
T_Sr            ds      1
T_Aar0          ds      1
T_Aar1          ds      1
T_Aar2          ds      1
T_Aar3          ds      1
T_Bcr           ds      1
T_Sccr          ds      1

;-------------------------------------------------------------------------
; Local Routine
;-------------------------------------------------------------------------
        org     P:$3000
Bootload:
        movep   #$2800,x:M_TCSR0
        ori     #3,mr                      ;Disable interrupts
        move    #VERSION,x0                ;Store version number
        move    x0,y:Version

        movep   #$000302,x:M_SC            ;Configure SCI, to use external
        movep   #$00C000,x:M_SCCR          ;clock, 8 bits/word, 1 stop
        movep   #7,x:M_PCRE                ;Same as bootloader in ROM

        bsr     RxWordEcho                 ;Read word and echo back

        move    y:Version,a                ;Transmit Version number
        bsr     TxWord

        move    x:M_IDR,a                  ;Transmit Part ID
        bsr     TxWord

        bsr     RxWordEcho                 ;Receive control register values
        move    a1,y:T_Pctl
        bsr     RxWordEcho
        move    a1,y:T_Omr
        bsr     RxWordEcho
        move    a1,y:T_Sr
        bsr     RxWordEcho
        move    a1,y:T_Aar0
        bsr     RxWordEcho
        move    a1,y:T_Aar1
        bsr     RxWordEcho
        move    a1,y:T_Aar2
        bsr     RxWordEcho
        move    a1,y:T_Aar3
        bsr     RxWordEcho
        move    a1,y:T_Bcr
        bsr     RxWordEcho
        move    a1,y:T_Sccr
        brclr   #0,X:M_SSR,*               ;Wait until TX complete

        move    y:T_Pctl,r0                ;Write values to control registers
        move    r0,x:M_PCTL
```

```
        move    y:T_Omr,r0
        move    r0,omr
        move    y:T_Sr,r0
        move    r0,sr
        move    y:T_Aar0,r0
        move    r0,x:M_AAR0
        move    y:T_Aar1,r0
        move    r0,x:M_AAR1
        move    y:T_Aar2,r0
        move    r0,x:M_AAR2
        move    y:T_Aar3,r0
        move    r0,x:M_AAR3
        move    y:T_Bcr,r0
        move    r0,x:M_BCR
        move    y:T_Sccr,r0
        move    r0,x:M_SCCR

        bsr     RxWordEcho                  ;Read word and echo back
        bra     StartDownload

Checksum:
        move    b1,a                        ;Check received data by transmitting
        bsr     TxWord                      ;checksum

StartDownload:
        bsr     RxWordEcho                  ;Read action to complete
        move    a1,b
        bsr     RxWordEcho                  ;Read length to download
        move    a1,y1
        bsr     RxWordEcho                  ;Read address to download to
        move    a1,r0

        tst     b                           ;Test action to complete
        beq     DownloadP
        sub     #1,b
        beq     DownloadX
        sub     #1,b
        beq     DownloadY
        sub     #1,b
        beq     RunCode
        bra     StartDownload

DownloadP:                                  ;Download to P Memory
        clr     b
        dor     y1,_loopP
        bsr     RxWord
        move    a,p:(r0)+
        add     a,b
_loopP:
        bra     Checksum

DownloadX:                                  ;Download to X Memory
        clr     b
        dor     y1,_loopX
        bsr     RxWord
        move    a1,x:(r0)+
        add     a,b
_loopX:
        bra     Checksum

DownloadY:                                  ;Download to Y Memory
        clr     b
        dor     y1,_loopY
        bsr     RxWord
        move    a,y:(r0)+
        add     a,b
_loopY:
        bra     Checksum

RunCode:                                    ;Run code
        brclr   #0,X:M_SSR,*                ;Wait until TX complete
        jmp     (r0)                        ;Start running code

;*************************************************************************
RxWord:                                     ;Receive word routine
        clr     a
        dor     #3,_rxloop
        brclr   #2,X:M_SSR,*
        movep   x:M_SRXL,a2
        asr     #8,a,a
_rxloop:
        rts
```

```
;*************************************************************************
RxWordEcho:                                     ;Receive word and echo back routine
        clr     a
        dor     #3,_rxloopecho
        brclr   #2,X:M_SSR,*
        movep   x:M_SRXL,a2
        brclr   #1,X:M_SSR,*
        movep   a2,x:M_STXL
        asr     #8,a,a
_rxloopecho:
        rts

;*************************************************************************
TxWord:                                         ;Transmit word routine
        dor     #3,_txloop
_waittx:
        nop
        nop
        brclr   #1,X:M_SSR,_waittx
        movep   a1,x:M_STXL
        asr     #8,a,a
_txloop:
        rts

        endsec
;-------------------------------------------------------------------------------
;       Copyright (C) 1998 Motorola
;-------------------------------------------------------------------------------
```


How to reach us:

USA/Europe/Locations Not Listed:
Motorola Literature Distribution
P.O. Box 5405
Denver, Colorado 80217
1 (800) 441-2447
1 (303) 675-2140

Motorola Fax Back System (Mfax™):
TOUCHTONE (602) 244-6609
1 (800) 774-1848
RMFAX0@email.sps.mot.com

Asia/Pacific:
Motorola Semiconductors H.K. Ltd.
8B Tai Ping Industrial Park
51 Ting Kok Road
Tai Po, N.T., Hong Kong
852-26629298

Technical Resource Center:
1 (800) 521-6274

DSP Helpline
dsphelp@dsp.sps.mot.com

Japan:
Nippon Motorola Ltd
SPD, Strategic Planning Office141
4-32-1, Nishi-Gotanda
Shinagawa-ku, Japan
81-3-5487-8488

Internet:
http://www.motorola-dsp.com/