Order this document by
AN1855/D
Revision 0

MOTOROLA

AN1855

Application Note

‘Download’ and ‘Checksum’ Programs for Use with the DSP5636x Family

This application note is designed to help customers use the DSP5636x Family of digital signal processors in their digital audio applications. In particular, customers who use them to decode home theatre digital surround sound signals, such as Dolby Digital (AC3) and Digital Theatre Sound (DTS).

The example in this document uses a DSP56362 and an external EPROM to store the coefficients that are required for DTS decoding. In many applications, the EPROM is not fully utilized because the coefficients do not use all of the EPROM space. The remaining space can be used to store excess program information, such as Post Programming Phases (PPPs), which could not fit into the host controller’s memory. The program can be adapted for other applications with minor additions, such as the start address of the EPROM.

When data is being transferred from one memory array to another, it is important to verify that the data was stored correctly in the EPROM and that is was transferred correctly to the other memory array. The user can perform these tasks with the checksum program, which can be used in the same applications as the bootloader.

The DSP5636x Family has an embedded Software Architecture (SA) that is designed specifically for digital audio applications. This means that the listed programs must be intialized and used differently than other DSP56300 devices, although the code could be adapted for these applications.

DOWNLOAD OVERVIEW

The Download program downloads a program from external memory (in this case, a DTS EPROM) to the DSP’s internal memory (either P, X or Y RAM). How the data is stored in the external EPROM is crucial to successful operation because it determines where the data is stored in the internal RAM. In this program, the only ‘variable’ information that has to be given to the DSP is the start address of the EPROM data. All other information is stored within the data block itself.

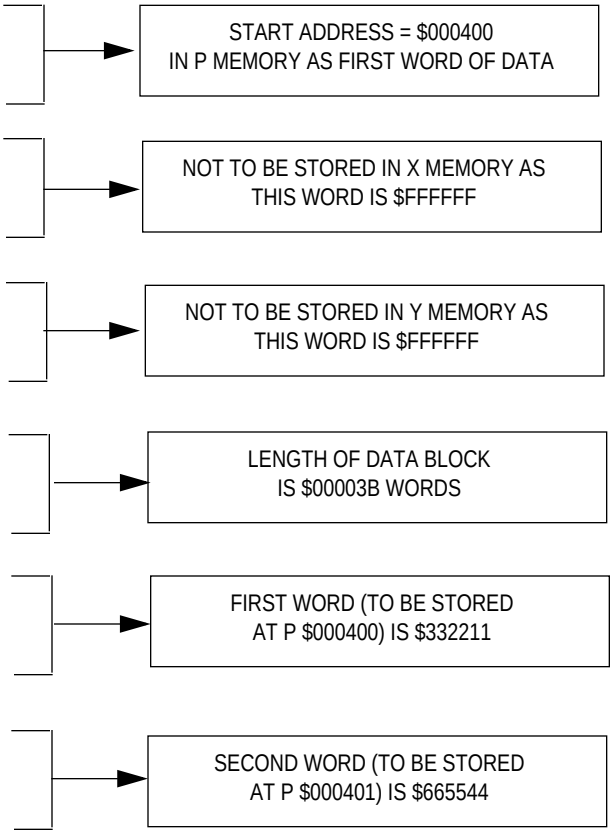
The DTS EPROM in audio applications is usually 8-bits (byte) wide, as opposed to the DSP56362’s 24-bit wide architecture. The program handles this by converting three successive bytes in the EPROM to one complete 24-bit word in the DSP.

The first four words (12 bytes) in the particular EPROM block of data determines the start address where the data is to be stored, the length of the program/data and the type of DSP RAM that the data will be copied to (i.e. P, X or Y). The user can do this by comparing the first three 24 bit words (consisting of three successive bytes) with the value $FFFFFF. If it is not equal to this value, then it represents the start address of the data. The position of this non-$FFFFFF word determines whether it is to be stored in P, X or Y RAM. The fourth word (or 10th, 11th and 12th bytes) determine the length of code that is to follow. This operation is illustrated in the following examples.

EXAMPLE 1

EXTERNAL EPROM VALUES IN HEX

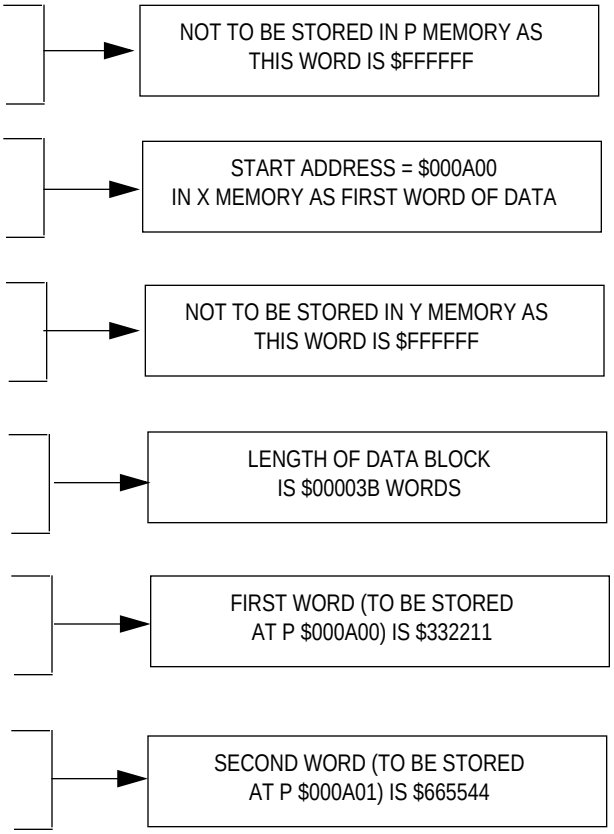
ADDRESS	DATA
300	00
301	04
302	00
303	FF
304	FF
305	FF
306	FF
307	FF
308	FF
309	3B
30A	00
30B	00
30C	11
30D	22
30E	33
30F	44
310	55
311	66
312	77



EXAMPLE 2

EXTERNAL EPROM VALUES IN HEX

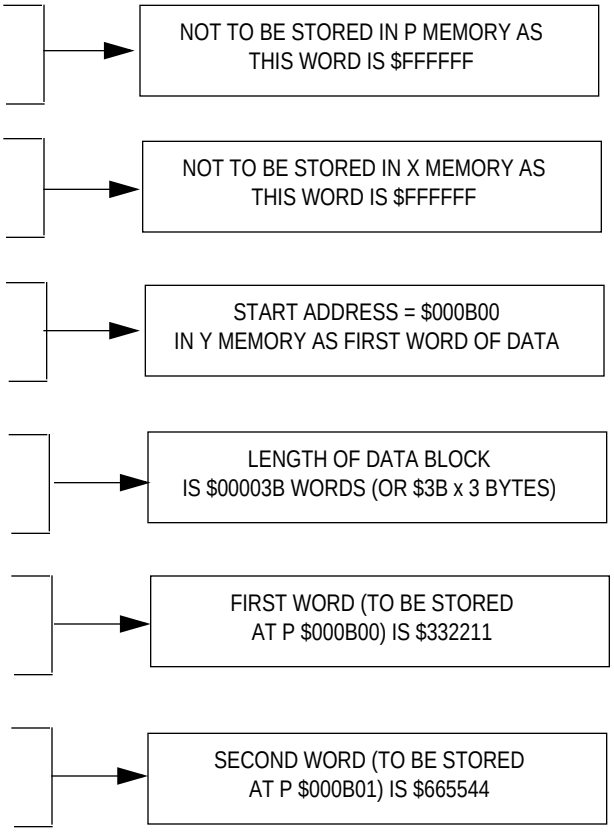
ADDRESS	DATA
300	FF
301	FF
302	FF
303	00
304	0A
305	00
306	FF
307	FF
308	FF
309	3B
30A	00
30B	00
30C	11
30D	22
30E	33
30F	44
310	55
311	66
312	77



EXAMPLE 3

EXTERNAL EPROM VALUES IN HEX

ADDRESS	DATA
300	FF
301	FF
302	FF
303	FF
304	FF
305	FF
306	00
307	0B
308	00
309	3B
30A	00
30B	00
30C	11
30D	22
30E	33
30F	44
310	55
311	66
312	77



Note: The program can be run as many times as necessary to load all the ‘blocks’ stored in the external EPROM.

CHECKSUM OVERVIEW

The Checksum program takes a value (in Y memory) that defines the start of data and a value (in Y memory) that defines the end of data, adds up all of the data between and including these values, and stores the result in Y memory. This program is good for checking the integrity of the external EPROM used for DTS coefficients (and any further data stored in it) and the connections between the EPROM and DSP. It can also be used to check data after it has been downloaded to the DSP.

Note: An ‘AND’ operation is required when checking an 8-bit wide EPROM to mask off the 16 MSBs of the 24-bit word that the DSP expects.

SOFTWARE ARCHITECTURE (SA) CONSIDERATIONS

Since this program is designed to work within the DSP56362 device (although it may be adapted for other Motorola DSPs), the user must account for the onboard ROM and SA. Specifically, the Download and Checksum programs must be run before the High Level eXecutive (HLX) is envoked. The user can do this by loading the ‘cld’ file in the intialization routine and running it before the HLX is run. This routine would be carried out by the host controller, which could be a microcontroller (e.g. Motorola HC08 or HC11) or a PC. This is shown in the following DSP56362 Evaluation Board initialization routine. Communication between the board and the host PC is via the P&E Microcomputer Systems PPI cable.

```
rem EVB Version
rem InitAV4 routine

rem  set PLL
CMD $C50001 $FFFFFD $0F0009

rem  set IPRP
CMD $C50001 $FFFFFE $000187

rem  set AAR0, AAR1, AAR2
CMD $C50003 $FFFFF7 $040639 $080539 $0C063D

rem  set BCR
CMD $C50001 $FFFFFB $0005E1

rem  test Mu
CMD $C00048 $FF27B6
CMD $C00049 $400000

rem load 'Checksum' cld
LOADC:\Checksum.cld

rem run Checksum routine
rem Pstart = $200
CMD $C00048 $000200
CMD $C00049 $400000
```

Note: The host controller could now use the result stored in Y memory to check EPROM validity.

```
rem load 'Download' cld
LOADC C:\Download.cld

rem run Download routine
rem Pstart = $200
```

CMD $C00048 $000200
CMD $C00049 $400000

Note: The host controller could perform another Checksum to verify that the data has been downloaded successfully or it could perform another Download for another block of data.

rem Init Mu
CMD $C00048 $ff2459
CMD $C00049 $400000

rem Run Mu
CMD $C00048 $FF2475
CMD $C00049 $800000

CODE LISTINGS

Download

```
;OPERATION
;Will download a program from external memory to internal
;memory.  The first byte represents the start address of the
;data.
;
;The actual 'bodies of code' will be loaded into either P, X, or Y
;depending on which of the three first words does NOT have $FFFFFF.
;The value it does have will be the starting address in the relevant
;memory space.  The length of code is located in the 4th word of the ;block

;*****************************************
        TITLE 'Download';Name of Program
;*****************************************
      SECTION     y_memory;    Y memory start

      org         y:           ;Defined in Control File



      GLOBALBlockStartAddress; Declare variables globally

BlockStartAddress dc    $0A0000;    Declare where the block starts
                                    ;(external EPROM)

      ENDSEC                  ;End of Y memory
;*****************************************

      SECTION           p_memory;    P memory start
      org               p:

Start:                        ;Start of program

;Stage 1 - determine which type of memory it is

      move              #$FFFFFF,a   ;If number is $FFFFFF,
                                          ;there is none of that type
                                          ;Any other number signifies the
                                          ;start address of that type of data
```

```
      move        y:BlockStartAddress,r0  ;Set up pointer to block
      jsr         MakeWord                ;Get value from EPROM
      cmp         x0,a
      jne         Pdata                   ;Pdata found
;---------------------------------------------
      jsr         MakeWord                ;Get value from EPROM
      cmp         x0,a
      jne         Xdata                   ;Xdata found
;---------------------------------------------
      jsr         MakeWord                ;Get value from EPROM
      cmp         x0,a
      jne         Ydata                   ;Ydata found
;---------------------------------------------
      jmp         End                     ;Error as all had #$FFFFFF
;*************************************************************

Pdata:   ;Pdata with start address already in x0

      move        x0,r1       ;use r1 as pointer to Pram

      move        #6,n0
      move        y:(r0)+n0,x0 ;Need to increment r0 6 times to get to
                               ;body length

      jsr         MakeWord    ;Get value from EPROM

      do          b1,EndofP   ;Repeat for whole 'body length'

      jsr         MakeWord    ;Get value from EPROM

      move        b1,p:(r1)+  ;Store 24-bit value into P memory
      nop                     ;Need this or do loop doesn't work

EndofP:
      jmp         End               ;Finished downloading P
;*************************************************************

Xdata:   ;Xdata with start address already in x0

      move        x0,r1       ;use r1 as pointer to Xram

      move        #3,n0
      move        y:(r0)+n0,x0 ;Need to increment r0 3 times to get to
                               ;body length

      jsr         MakeWord    ;Get value from EPROM

      do          b1,EndofX   ;Repeat for whole 'body length'

      jsr         MakeWord    ;Get value from EPROM

      move        b1,x:(r1)+  ;Store 24-bit value into X memory
      nop                     ;Need this or do loop doesn't work
```

```
EndofX:
      jmp         End               ;Finished downloading X
;*************************************************************

Ydata:   ;Ydata with start address already in x0

      move        x0,r1       ;use r1 as pointer to Yram

      jsr         MakeWord    ;Get value from EPROM

      do          b1,EndofY

      jsr         MakeWord    ;Get value from EPROM

      move        b1,y:(r1)+  ;Store 24-bit value into Y memory
      nop                     ;Need this or do loop doesn't work

EndofY:
      jmp         End               ;Finished downloading Y
;*************************************************************
;*****************************************

; Subroutine which takes 3 8-bit values
; from y:(r0),y(r0+1),y(r0+2) and makes
; up a 24-bit value in b1
; Format is LSB-MSB-USB

MakeWord:

      do          #3,EndofWord      ;3bytes in every word
      move        y:(r0)+,b2        ;Move byte into b2 first
      asr         #8,b,b            ;Then shift right 8 and repeat

EndofWord:                          ;Result stored in b1

      move  b1,x0                   ;Store in x0 for comparison with $FFFFFF

      rts                           ;Return from subroutine
;*****************************************
;***********************************

End:
      clr   a     ;Signal OK status
      rts         ;Exit
;***********************************
      ENDSEC                        ;End of P memory
;-------------------------------------------------------------------
```

Checksum

```
;OPERATION
;Will perform a checksum on an external memory area (EPROM)
;to check data integrity, result is stored in y:$302
;Result should be $802FD8 for an external DTS EPROM
```

```
;*****************************************************
      TITLE 'Checksum';      Name of Program
;*****************************************************

      SECTION           y_memory

      org   y:    ;Declare start address in control file

      GLOBALStartOfCode,EndOfCode,Checksum

StartOfCode       dc    $080000     ;Declare where the start of code/data
                                    ;to be checked is

EndOfCode         dc    $090000     ;Declare where the last word of code/data
                                    ;is (+1)

Checksum          dc    0           ;This is where the checksum will be stored

      ENDSEC
;*****************************************
      SECTION           p_memory

      org         p:                ;start of PPP code, declare in control file

Start:
      clr         a
      clr         b                 ;Ensure accumulators are clear first
      move        y:StartOfCode,r0  ;use r0 as an indexed pointer

Next:
      move        y:(r0)+,b
*1    move        #>$0000FF,y0      ;Mask off the 16MSBs (only 8 bit value)
*1    and         y0,b
      move        b1,x0             ;get the value into x0
      add         x0,a              ;Add value to running total in a
      move        r0,y0             ;Check where we are in the EPROM
      move        y:EndOfCode,b
      cmp         y0,b              ;Check if at end of code
      jeq         Getsum            ;Reached the end of code
      jmp         Next              ;Get Next opcode in memory

Getsum:
      move        a1,y:Checksum     ;Store result in y:Checksum

;***************
      clr         a                 ;Signal OK status
      rts                           ;Exit
;***************

      ENDSEC

NOTE - '*1' code only required if checking an external EPROM which is 8 bits wide

;_____________________________________________________________________
```

Control File

```
; This is required as the code was written to be re-locatable
; The sections below may have to be changed to suit different ROMs

;******************   Motorola DSP56362   ******************
;***              memory configuration file          ***
;*********************************************************

START             Start             ;define entry point

; allocate sections in memory

SECTION     y_memory    y:$300      ;allocate Y memory
SECTION     p_memory    p:$200      ;allocate program space

;_____________________________________________________________________
```

Assemble and Link Commands

Assemble.cmd *(creates a '.cln' file from the '.asm')*

```
asm56300 -b -l Download.asm > AssembleErrors.txt
```

Link.cmd *(creates '.map' and '.cld' files from the '.ctl' and '.cln' files)*

```
dsplnk -mDownload.map -rControlFile.ctl -bDownload.cld Download.cln >
LinkErrors.txt
```

NB – Motorola development tools are available from
http://www1.motorola-dsp.com/software/software.taf?type=updates

Mfax is a trademark of Motorola, Inc.

How to reach us:

USA/EUROPE/Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217.

1-303-675-2140 or 1-800-441-2447

JAPAN: Motorola Japan Ltd.; SPS, Technical Information Center, 3-20-1, Minami-Azabu, Minato-ku, Tokyo 106-8573 Japan. 81-3-3440-3569

ASIA/PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong. 852-26668334

Customer Focus Center: 1-800-521-6274

Mfax‰: RMFAX0@email.sps.mot.com - TOUCHTONE 1-602-244-6609

Motorola Fax Back System- US & Canada ONLY 1-800-774-1848

- http://sps.motorola.com/mfax/

HOME PAGE: http://motorola.com/sps/

AN1855/D