Preliminary Information

Application Brief

APPLICATIONS:

- *Steppers and Encoders*
- *Home Appliance Controls Integrated with Voice Control*
- *Smart Appliances*
- *Home Security*
- *Digital Telephone Answering Machine*
- *Engine Management*
- *Power Line Modem*
- *Servo Drives*
- *Automotive Control*
- *Electric Lawn Equipment*
- *Noise Cancellation*
- *Internet Appliances*
- *IP Phone*
- *Modems*
- *Magnetic Card Readers*
- *Security*
- *Digital Speakers*
- *Voice Recognition Systems*
- *"Hands-free" Kits*
- *Digital Cameras*
- *Telecom Test Equipment*
- *Fuel Management Systems*
- *and more. . . . . .*

DSP56824 - Interfacing Two Codecs Without External Glue Logic

Although using only one Codec is most common in DSP applications, in many telephony (and other) applications, we need to interface two Codecs to the digital signal processor. Figure 1 shows an example using only one DSP to filter the signals from the Far End to the Near End and from the Near End to the Far End. Note the DSP is running two Filter Algorithms.

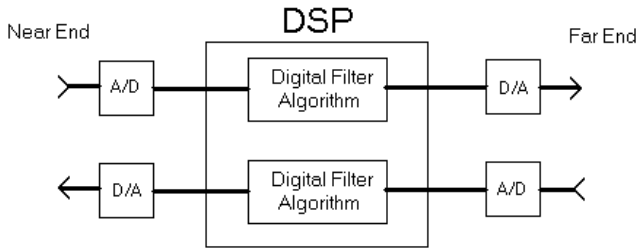


Figure 1. DSP Running Two Digital Filters

But, how can we interface two Codecs to the DSP56824 if it has only one SSI interface? The answer is to use the SSI network mode (please refer to the DSP56824 User's Manual for more details about SSI operation).

In the network mode the SSI is able to receive and/or transmit from 2 to 32 words per sampling frame. The idea basically is to configure the SSI for Network Mode with two slots per frame, doing the data exchanges (read and write) to one Codec on the first slot and the data exchanges to the other Codec on the second slot of the frame.

But, the MC145483 needs a Frame Sync clock to synchronize the input/output serial PCM data! Once the Frame Sync clock generated by the SSI is synchronized to the first slot, if we use the same Frame Sync clock to drive both Codecs, both will transmit and receive data on the first slot; and so, we will have a bus error.

A simple way to solve that conflict is configure the SSI options to generate the Frame Sync clock to Codec2.

The MC145483 FST (frame sync transmit) and FSR (frame sync receive) pins are long and short frame sync compatible (please refer to the MC145483/D technical data for further information); so, we can drive these Codec pins with a one-word length Frame Sync. Figure 6 shows the timing for one-word length Frame Sync clock and two words (slots) per frame.

Note in Figure 2, Codec2 Frame Sync is the negative form of Codec1 Frame Sync Clock. So, all we need to do is use a NOT gate to negate the Frame Sync to Codec2. Better, we can just configure the SSI to generate a Frame Sync active high in the STFS pin to feed Codec1 and a Frame Sync active low on the SRFS pin to feed Codec2. The hardware configuration is shown in Appendix 2.

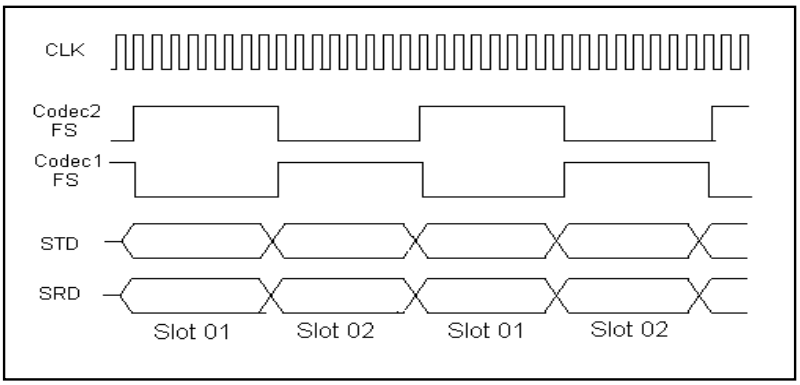


Figure 2. One-word length Frame Sync

The SSI setup (describing the SSI configuration), TX and RX routines are listed in Appendix 1. Note the SSI SRCK pin was left as GPIO in order to be used as Codec mode select.

Also, due to enabling the receiver and the transmitter buffers, the receiver and transmitter interrupt routines perform two read/write operations at each interrupt request attempt.

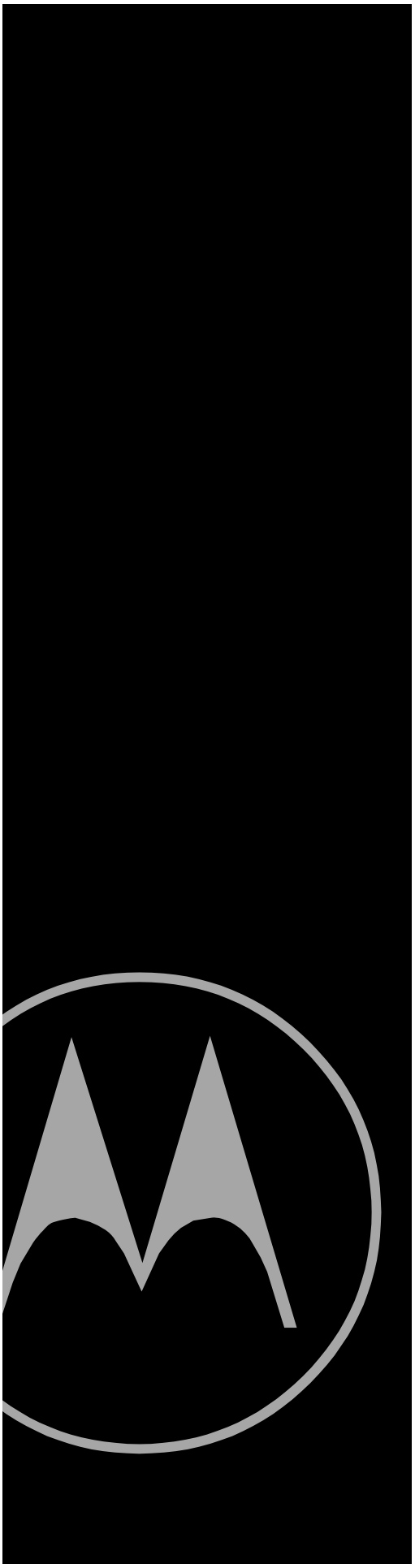
To assure the data to Codec1 is being transmitted in the first slot and the data to Codec2 is being transmitted in the second slot, the transmitter interrupt handler routine checks the SSI TFS flag before choosing the sequence in which the data will be written to the TX buffer.

In a similar way, the receiver interrupt handler routine checks the SSI RFS flag to determine if the first data on the FIFO receiver buffer is from Codec1 or Codec2.

Appendix - 1. Code Example

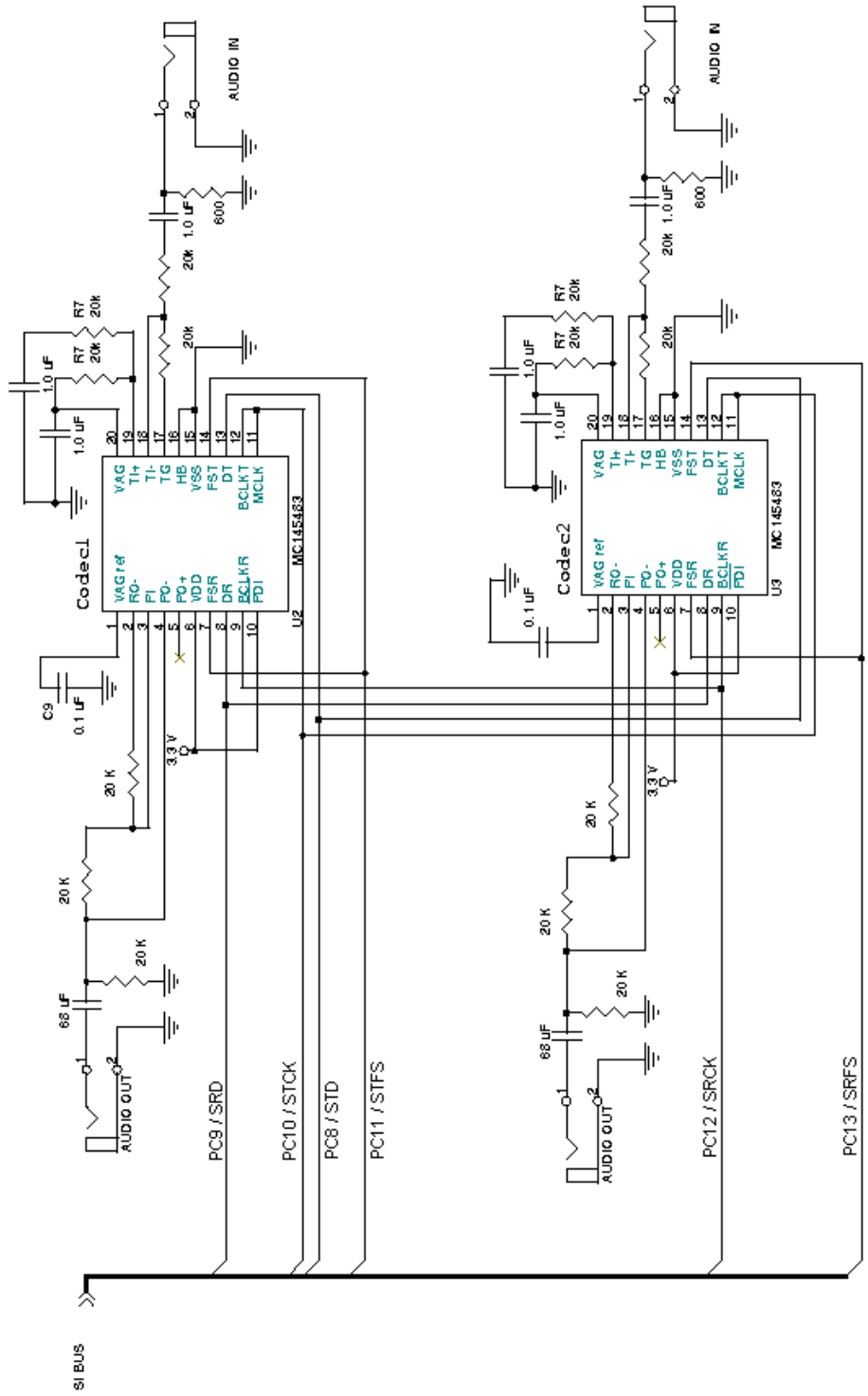
```
/* Global Data */
__fixed__ rx_buffer[2];
__fixed__ tx_buffer[2];

unsigned char SSI_NewData;
/*=====================================================
; Configure the SSI
;=====================================================
;  Asynchonous Mode
;  Network Mode / 2 slots per frame / 16 bits per word
;  256 kHz bit clock / 8 kHz Frame Rate
;
; Core CLK Prescale Bit CLK Word Div Frame Div  Frame CLK
; -------- -------- ------- -------- ---------  ---------
; 32.768MHz =>/128 =>256kHz =>/16    => /2      => 8KHz
;=====================================================*/
asm void SSI_SET_UP () {
  move     #0,y0
  move     y0,SSI_NewData
  move  y0,x:SCR2                  //; disable the SSI
  bfset  #0x2f00,x:PCC   //; SRCK left as GPIO
           move  #0x611f,x:SCRRX  //; PSR=0, WL=3, DC=1, PM=31
  move  #0x611f,x:SCRTX  //; PSR=0, WL=3, DC=1, PM=31
  move  #0xFF08,x:SCR2  //; RIE=1, TIE=1, RE=1, TE=1,
                        //; RBF=1, TBF=1, RXD=1,TXD=1
                        //; SYN=0, SHFD=0,SCKP=0,SSIEN=0,
                       //; NET=1, FSI=0, FSL=0, EFS=0
  move          #0x0400,x:SCSR  //; RSHFD=0,RSCKP=0,RFSI=1, RFSL=0,
                                    //; REFS=0, RDF=0,TDE=0, ROE=0,
                                 //; TUE=0, RFS=0, TFS=0, RDBF=0, TDBE=0
          bfset  #0x0200,x:IPR   //; Enable SSI interrupts
 bfset  #0x0010,x:SCR2  //; Enable SSI
 rts
}
/*=====================================================
; Read Data From SSI – Receive and Rec. with exc. handler
;=====================================================
; Read data from First slot into Fills rx_buffer[0] and data from   ; Second slot
to rx_buffer[1]. Returns SSI_NewData = 1
;=====================================================*\
asm void SSI_RX() {
  lea     (SP)+
  move    y0,x:(SP)+
```

Digital DNA™ from Motorola



```
  move    r0,x:(SP)+
  move    m01,x:(SP)    //; push CPU registers
  move    #rx_buffer,r0 //; points to rx_buffer
  move    #$ffff,m01
  move    x:SCSR,y0     //; read the status reg to clear flags
  brclr  #RFS,y0,RxSecondSlot
  //; read rx_buffer[0] and then rx_buffer[1]
  move    x:SRX,y0
  move    y0,x:(r0)+  //; read the SRX into rx_buffer[0]
  move    x:SRX,y0
  move    y0,x:(r0)   //; read the SRX into rx_buffer[1]
  bra     RxExit
RxSecondSlot:
  //; read rx_buffer[1] and then rx_buffer[0]
  lea     (r0)+      //; r0 points to rx_buffer[1]
  move    x:SRX,y0
  move    y0,x:(r0)-  //; read the SRX into rx_buffer[1]
  move    x:SRX,y0
  move    y0,x:(r0)   //; read the SRX into rx_buffer[0]
RxExit:
  bfset #$0001,SSI_NewData //; set rx flag
  pop     m01
  pop     r0
  pop     y0             //; Pop CPU registers
  rti
}
/*============================================================
; Transmit Data Thru SSI – TRX and TRX with exc. handler
;============================================================
;Transmits tx_buffer[0] on the First Slot and  tx_buffer[1] on
; The second slot.
;=================================================+======*/
asm void SSI_TX () {
  lea     (SP)+
  move    y0,x:(SP)+
  move    r0,x:(SP)+
  move    m01,x:(SP)    //; push CPU registers
  move    #tx_buffer,r0 //; r0 points to tx_buffer
  move    #0xffff,m01
  move    x:SCSR,y0     //; read the status reg to clear flags
  brset    #TFS,y0,TxFirstSlot
  //; Send tx_buffer[0] and then tx_buffer[1]
  move    x:(r0)+,y0
  move    y0,x:STX     //; transfer tx_buffer[0] to STX
  move    x:(r0),y0
  move    y0,x:STX     //; transfer tx_buffer[1] to STX
  bra     TxExit
TxFirstSlot:
  //; Send tx_buffer[1] and then tx_buffer[0]
  lea     (r0)+      //; r0 points to tx_buffer[1]
  move    x:(r0)-,y0
  move    y0,x:STX    //; transfer tx_buffer[1] to STX
  move    x:(r0),y0
  move    y0,x:STX    //; transfer tx_buffer[0] to STX
TxExit:
  pop     m01
  pop     r0
  pop     y0          //; Pop CPU registers
  rti
}
```

Appendix - 2. Schematic



DSP56F80X
CUSTOMER SUPPORT:

Technical Support:
www.motorola.com/semiconductors/dsp/support
dsphelp@dsp.sps.mot.com
1-800-521-6274

Website:
www.motorola.com/semiconductors/dsp

Literature Distribution Center for Motorola:
1-800-441-2447

Other Inquiries:
Contact your Motorola sales representative or authorized distributor

Disclaimer:
This sheet may not include all the details necessary to completely develop this design. It is provided as a reference only and is intended to demonstrate the variety of applications for the device.

Digital DNA™ from Motorola

©2000 Motorola, Inc. Motorola is a registered trademark and OnCE, DigitalDNA and the DigitalDNA logo are trademarks of Motorola, Inc.