TEXAS INSTRUMENTS

Interfacing the TLV1572 Analog-to-Digital Converter to the TMS320C203 DSP

Application Report

May 1999

Mixed-Signal Products

SLAA026B

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Contents

List of Figures

List of Tables

Interfacing the TLV1572 Analog-to-Digital Converter to the TMS320C203 DSP

Tom Kugelstadt & Dave Quach

ABSTRACT

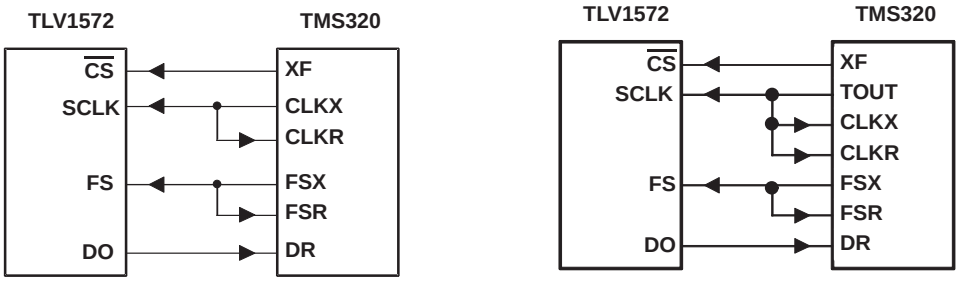
This application report presents a hardware solution for interfacing the TLV1572 10-bit, 1.25 MSPS (Mega Samples Per Second), successive low-power analog-to-digital converter (ADC) to the TMS320C203 16-bit fixed-point digital signal processor (DSP). In addition, a C-callable interface program is shown which supports the data transfer between ADC and DSP.

1 Introduction

The TLV1572 high-speed, 10-bit analog-to-digital converter (ADC), interfaces easily to DSPs and microcontrollers. With a conversion time of 600 ns, the TLV1572 provides a sampling rate of up to 1.25 MSPS with a 5-V supply, and up to 625 KSPS with a 3-V supply. To run at its fastest conversion rate, it must be clocked at 20 MHz at 5 V, or at 10 MHz at 3 V.

Using the TLV1572 with the TMS320C203PZ80 80-MHz DSP demonstrates the power and simplicity of a high-speed ADC-DSP interface. This DSP provides the high-frequency clock rates needed to stretch the TLV1572 to its limits. The DSP either provides a fixed 20-MHz clock at its CLKX output, or a programmable clock rate—adjustable between DC and 20-MHz—through the timer output, TOUT.

In 5-V applications, the the maximum 20-MHz ADC interface clock can be applied directly through the DSP, thereby simplifying the ADC-DSP interface to the one shown in Figure 1a.



a) Interface for 5-V Operation **b) Interface for 3-to-5-V Operation**

Figure 1. ADC-DSP Interface

For 3-V applications however, the ADC clock reduces to 10 MHz, and the DSP on-chip timer must be used as the clock generator. This hardware timer can be programmed to generate clock rates between 0 Hz (dc) and 20 MHz. The interface clock is fed through the timer output, TOUT, into the SCLK pin of the ADC as shown in Figure 1b.

The interfaces are glueless in both cases. To simplify the interface evaluation at 5 V and at 3 V, this report focuses on the second interface method using the DSP timer as a programmable clock source.

2 System Support

Figure 2 shows the system support functions in the form of a linear regulated power supply and an analog input buffer.

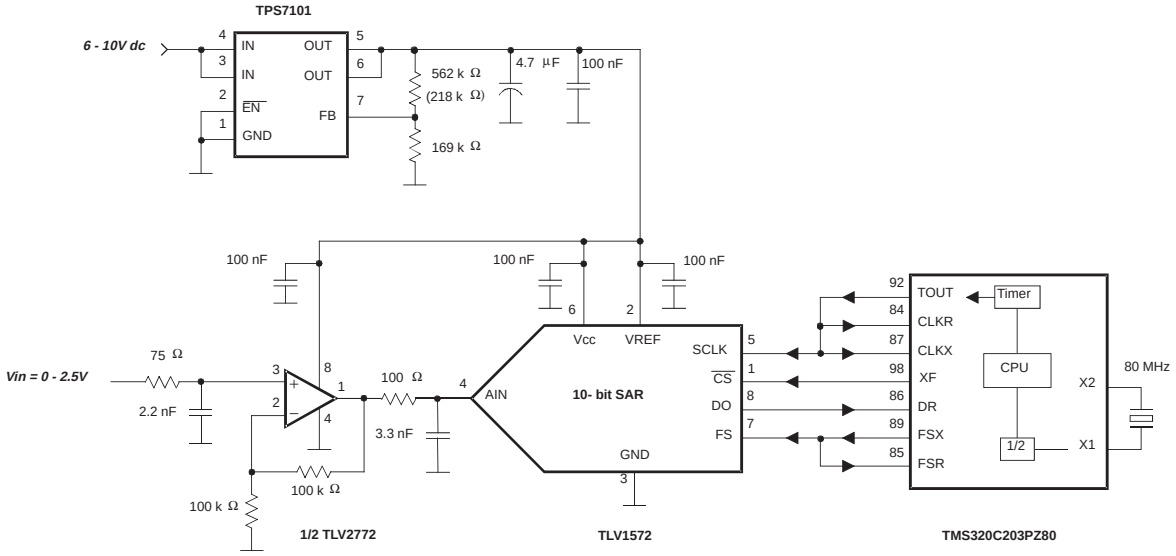


Figure 2. Mini Data Acquisition System

2.1 The Power Supply

The TPS7101 is an adjustable, low-voltage dropout regulator (LDO) with a typical voltage drop of 32 mV per 100-mA load current. The LDO regulates input voltages between 6 and 10 V down to the adjusted output level. The output voltage is adjusted to 5 V by an external voltage divider consisting of a 562-kΩ resistor and a 169-kΩ resistor. For an output voltage of 3 V, the 562-kΩ resistor is replaced by a 218-kΩ resistor.

The following low-ESR 4.7-µF solid tantalum capacitor and the 100-nF high-frequency ceramic capacitor are sufficient to ensure stability, provided that the total ESR is maintained between 0.7 Ω and 2.5 Ω. For more information on the selection and type of low-ESR capacitors, refer to the TPS7101 Data Sheet, literature number SLVS092F.

2.2 The Analog Input Buffer

The TLV2772 is a fast, low-volt, low-noise dual CMOS operational amplifier (op amp). The device operates from 5.5 V down to 2.2 V with a typical slew-rate of 10.5 V/µs and a typical noise density of 17 nV/$\sqrt{\text{Hz}}$ at 1 kHz.

In the configuration above, the op amp is a noninverting amplifier with a gain of two. The analog input signal, in the range of 0 V to Vcc/2, is band limited by the 75-Ω/2.2-nF input low-pass filter, before it is amplified. The 100-Ω/3.3-nF low-pass filter at the output reduces the output signal noise significantly and ensures a signal-to-noise ratio (SNR) > 90 dB at the ADC input.

3 ADC Overview

The TLV1572 is a 10-bit, successive-approximation ADC that operates within a supply voltage range from 2.7 V minimum to 5.5 V maximum. The typical conversion time is ten SCLK cycles with the specified maximum of SCLK = 20 MHz at 4.5 V and SCLK = 10 MHz at 3 V. The TLV1572 interfaces easily to DSPs and microcontrollers through a 4-wire serial interface.

The device features an auto power-down mode that reduces the current consumption to 10 μA when a conversion is not being performed. Figure 3 shows the block diagram of the device.

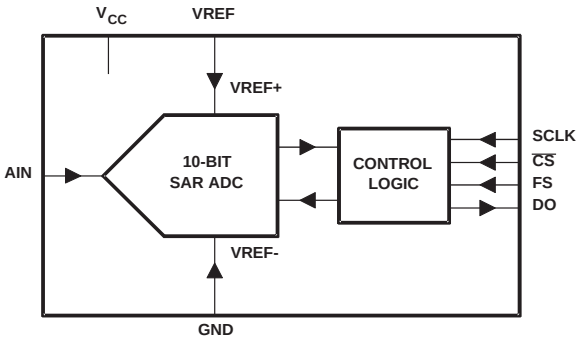


Figure 3. TLV1572 Block Diagram

The converter has a switched-capacitor architecture that performs the successive approximation through charge-redistribution. The internal control logic ensures synchronicity between the serial interface timing and the sampling and conversion process. Table 1 gives an overview of the terminal functions.

Table 1. Terminal Functions

V_{CC}	The device supply of between 2.7-V minimum and 5.5-V maximum is applied to this pin.
V_{REF}	This is the positive reference voltage of the ADC while the negative reference voltage is internally connected to ground. The difference between both reference voltages, in this case between V_{REF} and GND defines the input span of the ADC. The voltage applied to this pin can range from 2.7-V minimum to V_{CC}.
V_{AIN}	The analog input voltage at V_{AIN} can range from GND to a maximum level equals V_{REF}.
SCLK	The serial interface clock, SCLK, synchronizes the data transfer and is also used for internal data conversion. The maximum clock is 20 MHz with a minimum duration of 23-ns for on- and off-time.
$\overline{CS}$	A low-level on the chip select input enables the TLV1572. A high disables the device.
FS	This pin is the frame sync input in DSP mode. The falling edge of an FS-pulse from the DSP starts the serial data frame shifted out of the TLV1572. There is a minimum hold time of 9-ns required between the falling edge of /CS and the rising edge of FS. During this time the ADC checks internally whether it is operating in DSP- or μC-mode.
DO	The A-to-D conversion results are provided at this serial data output pin. The 16-bit format consists of six preceding zeros, followed by the 10-bit conversion result.

Figure 4 shows the interface timing between ADC and DSP. The ADC distinguishes between μC and DSP modes by checking the FS input level at the falling edge of $\overline{CS}$. If FS is low, DSP mode is set, otherwise μC mode is set.

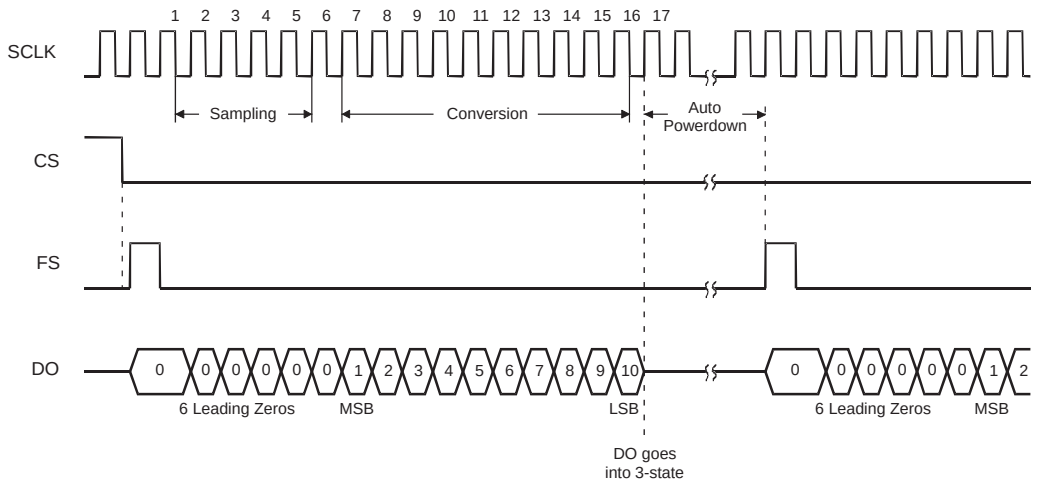


Figure 4. ADC/DSP Interface Timing

The ADC starts transferring data to the DSP with the rising edge of FS. Six zeros precede the 10-bit result to comply with the 16-bit data format of the DSP.

Sampling occurs from the first falling edge of SCLK after FS goes low, until the rising edge when the sixth zero bit is sent out. Thereafter decisions are taken on the rising edges and data is sent out on the rising edges, 1-bit delayed. The DSP samples on the falling edge of SCLK.

DO goes into 3-state on the 17th rising edge and comes out on an FS rising edge. The device goes into auto power down on the 17th falling edge of SCLK. A rising edge of FS pulls it out of power down and the next data transfer begins.

4 The TMS320C203 DSP

The TMS320C203 DSP is a 16-bit fixed point, static CMOS digital signal processor. The combination of an advanced Harvard architecture (separate buses for program memory and data memory), on-chip peripherals, on-chip memory, and a highly specialized instruction set is the basis of the operational flexibility of this device.

The synchronous serial port and the timer are the two on-chip peripherals that are used most in this application. Their block diagrams and operation are described briefly; for detailed information refer to the *TMS320C2xx User's Guide*, literature number: SPRU127B.

4.1 The DSP Serial Port

The serial port provides communication with serial devices such as codecs and serial ADCs. The synchronous serial port offers these features:

- Two four-word-deep FIFO buffers
- Interrupts generated by the FIFO buffers
- A wide range of speeds of operation
- Burst and continuous modes of operation

4.1.1 Signals and Registers

The synchronous serial port consists of the following six basic signals:

CLKX *Transmit clock input or output*. This signal clocks data from the transmit shift register (XSR) to the DX pin. The serial port can be configured for either generating an internal clock, or accepting an external clock.

If the port is configured for generating an internal clock, CLKX becomes an output, transmitting a maximum frequency equal to one half of the CPU clock. If the port is configured to accept an external clock, CLKX changes to an input, receiving the external clock signal.

FSX *Transmit frame synchronization*. FSX indicates the start of a transmission. If the port is configured for generating an internal frame sync pulse, the FSX pin transmits the pulse. If the port is configured for accepting an external frame sync pulse, this pin receives the pulse.

DX *Serial data transmit*. DX transmits serial data from the transmit shift register (XSR).

CLKR *Receive clock input*. CLKR receives an external clock for clocking the data from the DR pin into the receive shift register (RSR).

FSR *Receive frame synchronization*. FSR initiates the reception of data at the beginning of the packet.

DR *Serial data receive*. DR receives serial data, transferring it into the receive shift register (RSR).

4.1.2 Serial Port Operation

Figure 5 shows the block diagram of the synchronous serial port (SSP) with two four-level transmit and receive FIFO buffers. Two on-chip registers allow access to the FIFO buffers and control the operation of the port:

Synchronous data transmit and receive register (SDTR). The SDTR is used for the top of both FIFO buffers (transmit and receive) and is the only visible part of the FIFO buffers.

Synchronous serial port control register (SSPCR). The SSPCR contains bits for setting port modes, indicating the status of a data transfer, setting trigger conditions for interrupts, indicating error conditions, accepting bit input, and resetting the port.

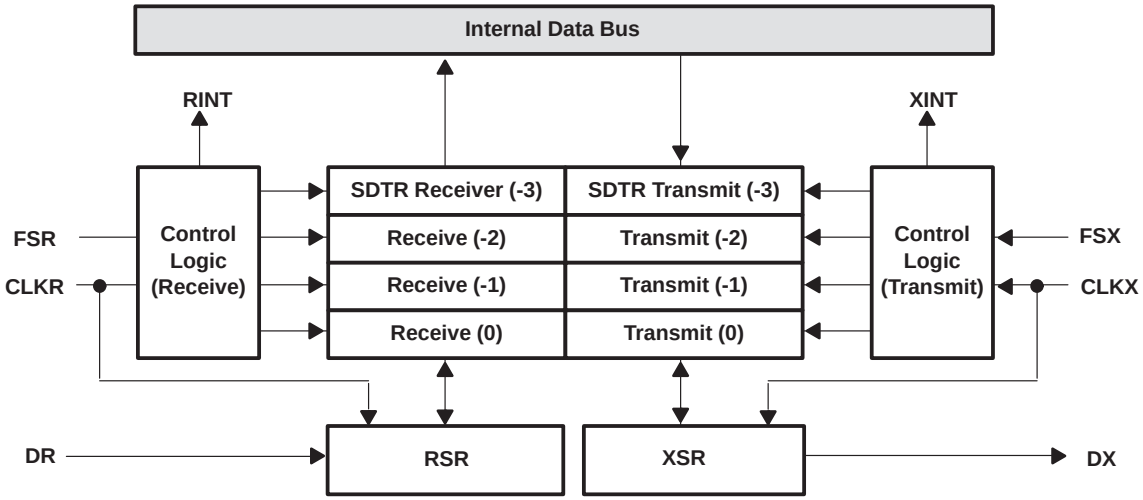


Figure 5. Synchronous Serial Port Block Diagram

Typically, transmitting a word through the serial port follows this five-step process:

1. Configure the serial port through the SSPCR.
2. Write up to four words to the transmit FIFO buffer through the SDTR.
3. The transmit FIFO buffer copies the earliest-written word to the transmit shift register, XSR, when XSR is empty.
4. The XSR shifts the data, bit-by-bit (MSB first), to the DX pin.
5. When the XSR empties, it signals the FIFO buffer, and then:
   – if the FIFO buffer is not empty, the process repeats from step 2,
   – if the FIFO buffer is empty, as specified in the SSPCR, it sends a transmit interrupt, XINT, to request more data and transmission stops.

Receiving a word through the serial port is done as follows:

1. Data from the DR pin is shifted, bit–by–bit (MSB first), into the receive shift register, RSR.
2. When the RSR is full, the RSR copies the data to the receive FIFO buffer.
3. The process then does one of two things, depending upon the state of the receive FIFO buffer:
   – if the receive FIFO buffer is not full, the process repeats from step 1,

– if the FIFO buffer is full (as specified by the SSPCR), it sends a receive interrupt, RINT, to the CPU to request servicing.

4. The processor can read the received data from the receive FIFO buffer through the SDTR.

4.1.3 Transmit and Receive Timing in Burst Mode

In burst mode operation, there is a period of serial port inactivity between packet transmits. Therefore each data packet needs to be marked by a frame sync pulse.

In the transmit direction, after a write to SDTR, a frame sync pulse (at FSX) is generated on the next rising edge of CLKX. On the next falling edge of CLKX, XSR is loaded with the value from DXR. XRDY goes high, generating a transmit interrupt, XINT. On the next rising edge of the CLKX cycle, the first data bit (MSB first) is driven on the DX-pin. With the falling edge of the frame sync pulse, the remaining bits are shifted out. When all bits are transferred, the DX pin enters the high-impedance state.

In the receive direction, the shifting into RSR begins on the falling edge of the CLKX cycle after the frame sync has gone low. After all bits have been received, the content of the RSR is transferred to the SDTR on the falling edge of CLKX. RRDY goes high, generating a receive interrupt, RINT.

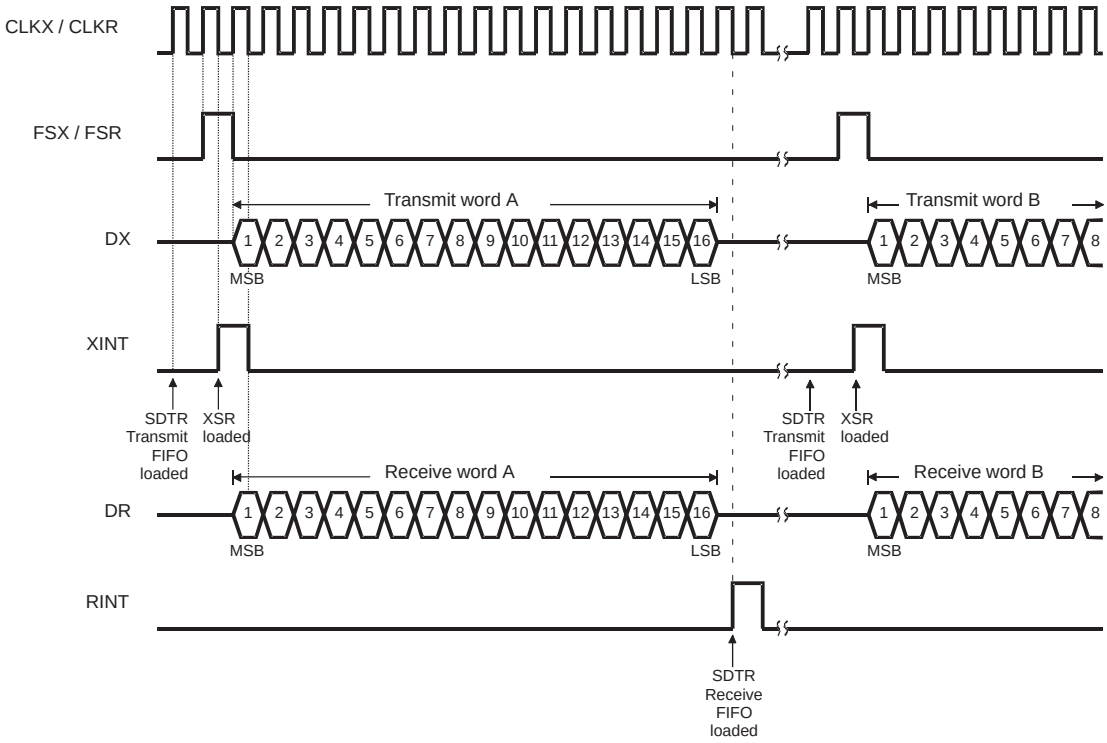


Figure 6. Transmit- and Receive-Operation in Burst Mode

4.2 The Hardware Timer

The hardware timer is the second on-chip peripheral used in this application. When the ADC operates at 3 V or less, the timer serves as the data clock source, providing the necessary low clock rate of 10 MHz.

The hardware timer is a fully programmable down counter, that is configured by the registers, PRD and TCR. The period register, PRD, is a 16-bit, I/O memory-mapped register that specifies the initial period of the timer. The timer control register, TCR, is a 16-bit, I/O memory-mapped register that contains the status and control bits to stop, restart, reset or disable the timer.

The timer consists of two counters in series, a 16-bit main counter, TIM, and a 4-bit prescaler counter, PSC. Figure 7 shows the block diagram of the timer. Each counter is loaded by a preceding register. The PCS is loaded by the timer divide-down register, TDDR, which is a 4-bit field within the TCR. The TIM is loaded by the 16-bit period register, PRD.

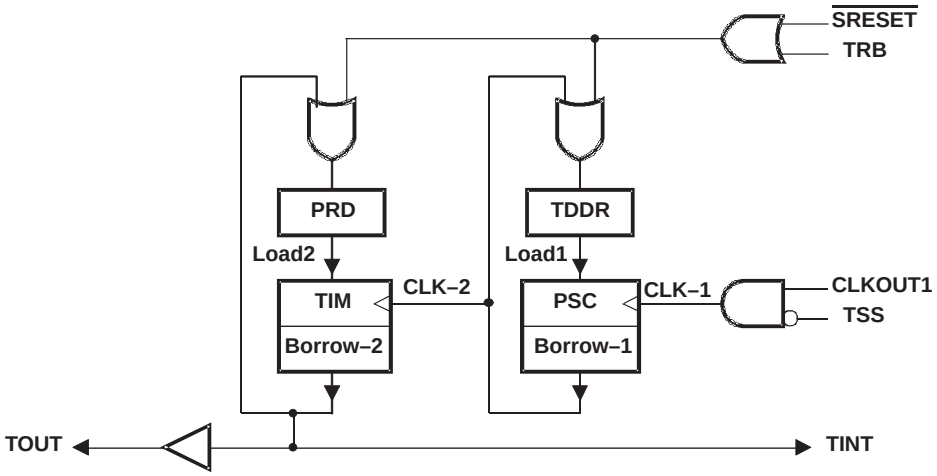


Figure 7. Timer Block Diagram

When the PSC decrements to zero, a borrow-1 signal is generated on the next CLK-1 cycle (with the CLK-1 being identical to the CLKOUT1 cycle). At that time the value of the TDDR is loaded into the PSC, and the TIM decrements by one.

Similarly, when the TIM decrements to zero, a borrow-2 signal is generated on the next CLK-2 cycle (with CLK-2 being identical to the borrow-1 signal). This time, both counters are reloaded. The value of the PRD is loaded into the TIM, and the content of the TDDR is loaded into the PSC. The borrow-2 pulse is sent to the external timer output pin, TOUT, as well as timer interrupt signal, TINT, to the CPU. The duration of a borrow-2 pulse at TOUT is equal to that of a CLKOUT1 cycle.

If the timer is used as an interrupt source, the TINT interval is determined through equation (1):

$$t_{TINT} = t_c \bullet (PRD + 1) \bullet (TDDR = 1) \quad (1)$$

with t_c being the time period of CLKOUT1.

If the timer is used as a clock source, the TINT-signal to the CPU is usually disabled and the clock signal is taken from the external timer output, TOUT. The available clock rate at TOUT is governed by equation (2):

$$TOUT[MHz] = \frac{CLKOUT1[MHz]}{(PRD + 1) \bullet (TDDR + 1)} \quad (2)$$

NOTE: When TDDR = PDR = 0, the timer interrupt rate defaults to ½ CLKOUT1 rate.

5 Software Description

The interface software consists of a C-callable assembler routine (TLV1572.asm) and its corresponding C program (C1572.C). The assembler program executes the data transfer in the following steps:

1. It initializes the DSP and the Synchronous Serial Port,
2. enables the ADC and starts the data transfer,
3. acquires the specified number of data,
4. and disables the ADC and returns to the C program.

The C program enables the user to specify certain data transfer parameters, such as the ADC supply voltage, the memory start address, and the number of samples, without modifying the assembler program.

When the C1572.C routine is called, the global variables in the C file that contain the user specified parameters are loaded into the local variables of the assembler program. Afterwards the assembler routine is started.

In this application the ADC operates from a 2.7-V supply and is configured for transfer clock at SCLK of 10 MHz. Since the only frequency the DSP can provide at CLKX is 20 MHz, the on-chip timer is configured as a programmable clock source providing a 10-MHz clock signal at the timer output, TOUT (Figure 8). Although this timer configuration is required for low-volt operation of the ADC, it permits the clock rate at TOUT to be 20 MHz if the ADC operates at 5.5 V.

The serial port is configured for burst-mode operation, the generation of the frame-sync pulse FS happens on-chip, and the CLKX pin is configured as input.

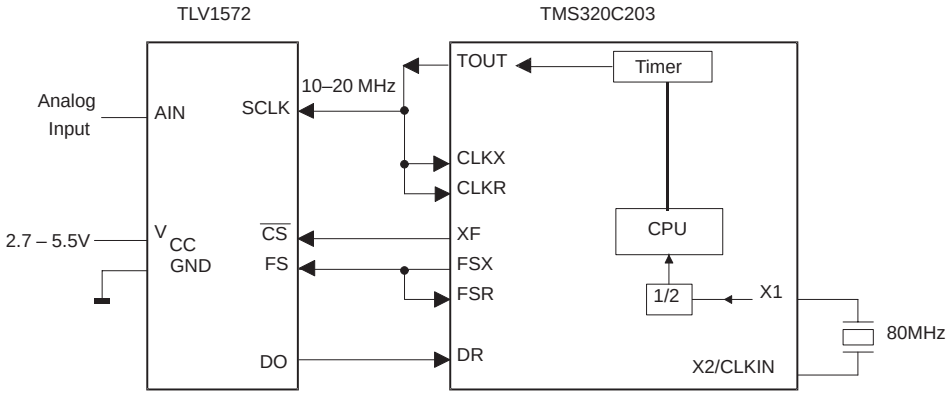


Figure 8. ADC/DSP Interface

The timing diagram in Figure 9 divides a data transfer sequence into three steps:

1. The DSP activates the ADC by taking the $\overline{CS}$ low. Then it sends the frame-sync pulse (FS), to start the data transfer.
2. The ADC sends out the first six zero bits, followed by the 10-bit conversion result.
3. On the 16th cycle of SCLK a receive interrupt, RINT, is generated. The following RINT-interrupt service routine stores the received data in memory and returns to idle mode, waiting for the next FS pulse to occur.

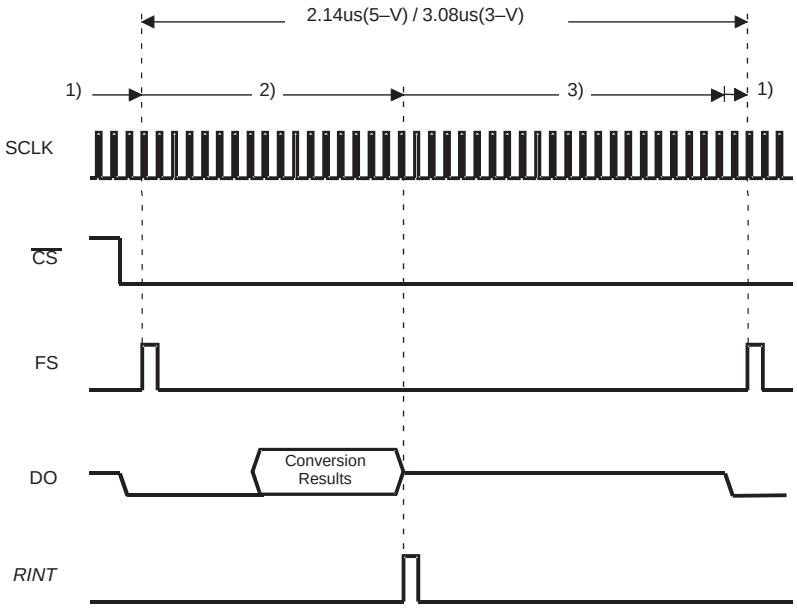


Figure 9. Interface Timing Using RINT

The duration of a transfer period is 2.14 μs at 5 V V_{CC}, and 3.08 μs at 2.7 V V_{CC}.

The following section explains the main assembler programs in detail. For a review of the individual file listings, refer to Appendix A.

5.1 Detailed Assembler Program Description *(Filename: TLV1572.asm)*

5.1.1 TLV1572START

Following the flowchart in Figure 10, the program starts by calling the main routine, '_TLV1572' from the C program. All previously used pointers and registers are saved. These include the following registers:

- The frame and the stack pointer, FP and SP
- The status registers, ST0 and ST1
- The auxiliary registers, AR6 and AR7
- The wait-state register, WSGR
- The interrupt mask register, IMR

During the DSP initialization all interrupts are disabled and the wait states are set to one. Then the serial port is configured for burst-mode operation. The FS signal is programmed to be generated on-chip and the CLKX pin is configured as input. Finally the transmitter and receiver stages are activated.

Now, the user-defined values (global variables in the C program) are loaded into the local variables of the assembler routine. The memory start address is loaded into AR6, the number of samples to be acquired is saved into AR7, and the value for the supply voltage is loaded into the lower byte of the accumulator, ACL.

The following configuration of the timer as clock generator writes a default frequency of 10 MHz into the PRD. Depending upon whether the user selected 3-V or 5-V operation, the content of PRD is either overwritten to 20 MHz, or keeps its default value.

Then the receive interrupt, RINT, is enabled and the local variable, END_BIT, which defines the end of the entire program is set to 1. Before the first data transfer can be initiated, the general output port of the DSP (XF), is driven low to enable the ADC through the chip-select pin, $\overline{CS}$.

The variable START, which contains a random value, is then copied into the serial port data transmit register, SDTR, generating an FS-pulse that starts the data transfer. The CPU then resides in idle-mode and waits for a receive interrupt, RINT, to occur.

5.1.2 RINT (Receive Interrupt Routine)

Upon the 16th clock cycle of SCLK, a RINT is generated that forces the CPU to execute the RINT service routine (RINT-ISR). At the beginning of the RINT routine, the content of the data receive FIFO of the SDTR is stored in data memory, specified in auxiliary register, AR6. Then the memory address is increased by incrementing the content of AR6.

The following decision box decrements the number of samples in AR7 and checks whether all samples have been received. If all samples were received, the ADC is disabled and the END_BIT is set to zero. Then the timer is stopped and RINT is disabled. The program leaves the RINT-ISR and returns to the C program with the EXIT routine,

If more samples need to be acquired, the program clears the RINT flag and returns to idle mode, where it sends the FS pulse and remains idle until the next RINT occurs.

5.1.3 Exit 1572 Program ?

As long as END-BIT is set to one, the CPU diverts to the Start-Data-Transfer box to continue acquiring data. Once END_BIT has been set to zero, all previously saved registers in the Save-Context box are restored. The CPU now exits the interface routine and returns to the C-program.

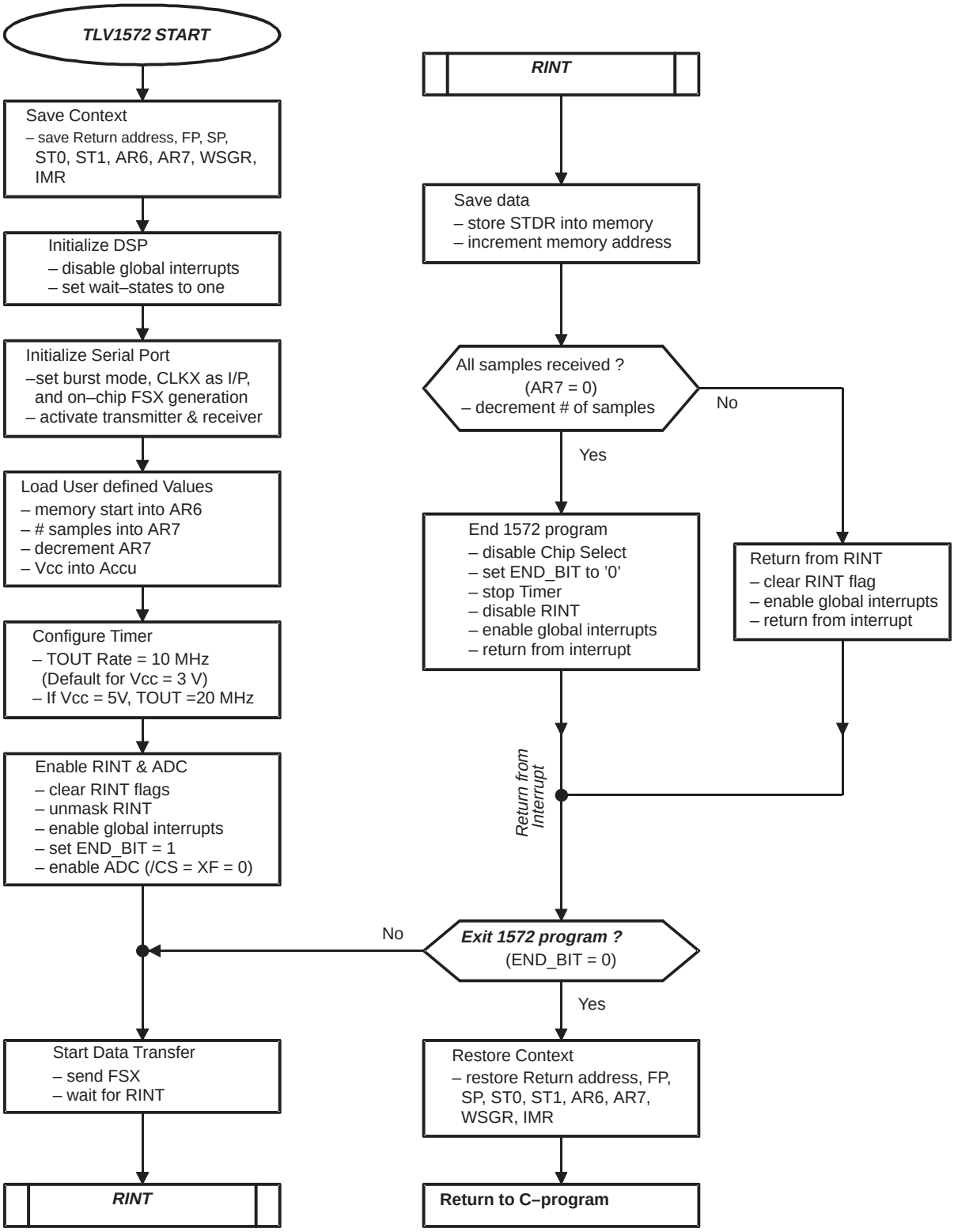


Figure 10. Flowchart of TLV1572.asm

Appendix A TLV1572 Program Files

A.1 Boot Routine: BOOT.ASM

```
*********************************************************************
* TITLE         : TLV1572C ADC boot routine                         *
* FILE          : BOOT.ASM                                          *
* DESCRIPTION   : Boot routine to initialize the DSP and run the    *
*                 C–program ‘main()’ in the C1572.C file.           *
* AUTHOR        : AAP Application Group, Dallas                     *
*                 CREATED 1999(C) BY TEXAS INSTRUMENTS INC.         *
* REFERENCE     : TMS320C2xx User’s Guide, TI 1997                  *
*               : Data Acquisition Circuits, TI 1998                *
*********************************************************************
    .mmregs
*********************************************************************
* Declare the stack.                                                *
*********************************************************************
__stack:        .usect  ”.stack”,0
    .def   _c_int0
    .ref   _main

    .text

_c_int0:
*********************************************************************
* INITIALIZATION BODY                                               *
*********************************************************************
     SETC   INTM           ; disable global interrupts
     LDP    #0             ; load data page 0
*********************************************************************
*  SET UP INITIAL STACK AND FRAME POINTERS                          *
*********************************************************************
     LRLK   AR0,__stack    ; Set up frame pointer for main routine
     LRLK   AR1,__stack    ; Set up stack pointer for main routine
     MAR    *,AR1          ; pointer select to AR1
     B      _main          ; jump to C program at main
```

A.2 C-Program: C1572.C

```
/* TITLE:        TLV1572 ADC C program main routine            */
/* FILE:         C1572.C                                       */
/* DESCRIPTION:  In this c–program file the user specifies the */
/*               Memory Start address, the ADC supply voltage  */
/*               and the number of Samples. This program then  */
/*               calls the TLV1572() interface program to      */
/*               execute it.                                   */
extern Vcc, Samples, MemStart;
extern void TLV1572(void);
main()
{
/* Select Vcc.  Vcc = 5 or 3                                   */
   Vcc = 3;
/* Take 256 samples                                            */
   Samples = 0x100;

/* Define Start Address                                        */
   MemStart = 0x1000;
/* Call TLV1572 Interface Program                              */
   TLV1572();
}
```

A.3 C-Callable Interface Program: TLV1572.ASM

```
*************************************************************************
* TITLE         : TLV1572 ADC C–Callable Interface routine              *
* FILE          : TLV1572.ASM                                           *
* FUNCTION      : _TLV1572                                              *
* DESCRIPTION   : Main routine to transfer data between the C203        *
*                 and TLV1572 ADC via the DSP serial interface. At      *
*                 first it initializes the DSP and enables the ADC,     *
*                 then it transfers data from the ADC to the DSP        *
*                 and stores them within a predefined memory table.     *
*                 The data transfer procedure is supported by the       *
*                 interrupt routine, RINT.                              *
* AUTHOR        : AAP Application Group, Dallas.                        *
*                 CREATED 1999(C) BY TEXAS INSTRUMENTS INC.             *
* REFERENCE     : TMS320C2xx User's Guide, TI 1997                      *
*               : Data Acquisition Circuits, TI 1998                    *
*************************************************************************
          .mmregs
          .sect "vectors"
          .copy "vectors.asm"
          .sect   ".io_reg"
SDTR      .set    0fff0h
SSPCR     .set    0fff1h
TCR       .set    0fff8h
PRD       .set    0fff9h
WSGR      .set    0fffch
* Global variables
          .global _TLV1572
          .global _Samples
          .global _MemStart
          .global _Vcc
          .global _c_int0
* Local variable
          .def    START
          .def    END_BIT
          .def    TEMP
AD_DP     .usect ".variabl",  0    ; AD Data Page for local varibles
START     .usect ".variabl",  1    ; data transfer start
END_BIT   .usect ".variabl",  1    ; end–of–TLV1572–program flag
TEMP      .usect ".variabl",  1    ; temporary data memory
_Samples  .usect ".variabl",  1    ; user defined number of sample
_MemStart .usect ".variabl",  1    ; user defined memory pointer
_Vcc      .usect ".variabl",  1    ; user defined Vcc
LOCALS          .SET   0
                .sect ".text"
```

```
* TLV1572 Start:
_TLV1572:
* Save Context
    POPD    *+                  ; save return address
    SAR     AR0, *+             ; save Frame Pointer (ar0)
    SAR     AR1, *              ; save Stack Pointer (ar1)
    LAR     ar0,*+,ar1          ; set–up new Frame Pointer
    ADRK    #LOCALS             ; set–up new Stack pointer
    SST     #0, *+              ; save status register 0
    SST     #1, *+              ; save status register 1
    SAR     AR6, *+             ; save AR6
    SAR     AR7, *+             ; save AR7
    IN      *+, WSGR            ; save wait state register
    LACL    IMR
    SACL    *+                   ; save IMR
* Initialize DSP
    SETC    INTM                ; disable global interrupts
    LDP     #AD_DP              ; load data page AD_DP
    SPLK    #01h, TEMP          ; WSGR = 1 when run at 80MHz board
    OUT     TEMP, WSGR          ; set wait–state to one
* Initialize Serial Port
    SPLK    #000Ah, TEMP        ; Set Burst Mode, CLKX is input
    OUT     TEMP, SSPCR         ; FSX generated by DSP
    SPLK    #003Ah, TEMP        ; activate transmitter and receiver
    OUT     TEMP, SSPCR
* Load user defined values
    LAR     AR6, _MemStart      ; Store memory start address into AR6
    LAR     AR7, _Samples       ; Store number of Samples into AR7
    MAR     *, AR7              ; ARP = 7
    MAR     *–, AR6             ; (AR7) = _Samples – 1, ARP = 6
    LACL    _Vcc                ; LOAD _Vcc into (ACL)
* Configure Timer
    SPLK    #3, TEMP            ; set timer for 10MHz (Default)
    OUT     TEMP, PRD
    CLRC    SXM                 ; clear sign extension mode bit
    SUB     #3                  ; check if _Vcc = 3V
    BCND    Timer_Start, EQ
    LDP     #AD_DP
    SPLK    #1, TEMP            ; set timer to 20 MHz if _Vcc = 5V
    OUT     TEMP, PRD
Timer_Start:
    LDP     #AD_DP
    SPLK    #0020h, TEMP        ; reload PRD/TDDR and start timer
```

```
    OUT    TEMP,TCR
* Enable RINT & ADC
    LDP     #0                  ; load data page 0
    SPLK    #8h, IFR            ; clear any pending RINT
    SPLK    #8h, IMR            ; unmask RINT
    CLRC    INTM                ; enable global interrupts
    LDP     #AD_DP
    SPLK    #1h, END_BIT        ; END_BIT = 1
    CLRC    XF                  ; enable Chip Select
* Start Data Transfer
Transfer:
    OUT     START,SDTR          ; Send FS to start transfer
    IDLE                        ; power down(IDLE), Waiting for RINT
* Exit 1572 program?
    LDP     #AD_DP
    LACL    END_BIT
    BCND    Transfer, NEQ       ; start new transfer if END_BIT = 1

* Restore Context
    MAR     *, AR1              ; make Stack Pointer (ar1) active
    MAR     *–                  ; decrement Stack Pointer (ar1)
    LACL    *–
    SACL    IMR                 ; restore IMR
    OUT     *–, WSGR            ; restore WSGR
    LAR     AR7, *–             ; restore AR7
    LAR     AR6, *–             ; restore AR6
    LST     #1, *–              ; restore st1
    LST     #0, *–              ; restore st0
    SBRK    #(LOCALS+1)         ; de–allocate frame size
    LAR     AR0, *–             ; restore old Frame Pointer (ar0)
    PSHD    *                   ; push return address on hardware stack
* Return to C–program
    RET                         ; return to C–program

*********************************************************************
* RINT
* The receive interrupt routine (RINT) stores the received data into
* the memory location (AR6), increments the memory pointer and
* decrements the number of samples (AR7).
*********************************************************************
```

```
RINT_IRQ:
* Save Data
    IN     *+, SDTR, AR7        ; Save SDTR into AR6
                                ; Increment memory address (AR6+1)
                                ; Change ARP to point to AR7
* All samples received?
    BANZ    RINT_END, *–, AR6   ; Go to RINT_END if AR7 > 0
                                ; Decrement number of samples (AR7–1)
                                ; Change ARP to point to AR6
* End 1572 program
    SETC       XF               ; disable Chip Select
    LDP        #AD_DP
    SPLK       #0h, END_BIT     ; set END_BIT = 0
    SPLK       #0010h, TEMP     ; stop timer
    OUT        TEMP,TCR
    LDP        #0
    SPLK       #8h,IFR          ; clear RINT flag
    SPLK       #0h,IMR          ; mask RINT
    CLRC       INTM             ; enable global interrupts
    RET                         ; return to”Exit 1572 program ?”
* Return from RINT
RINT_END:
    LDP        #0
    SPLK       #8h, IFR         ; clear RINT flag
    CLRC       INTM             ; enable global interrupts
    RET                         ; return to”Exit 1572 program ?”
   .copy ”boot.asm”
```

A.4 Vector Table: VECTORS.ASM

```
*************************************************************************
* TITLE         : TLV1572 ADC Interface routine                         *
* FILE          : VECTORS.ASM                                           *
* DESCRIPTION   : This file defines the interrupt vector table.         *
*                 If RINT occurs: vector points to RINT_IRQ subroutine*
* AUTHOR        : AAP Application Group, Dallas                         *
*                 CREATED 1999(C) BY TEXAS INSTRUMENTS INCORPORATED.    *
* REFERENCE     : TMS320C2xx User's Guide, TI 1997                      *
*************************************************************************
    .global   _main
RS      b   _c_int0           ;0x00; RESET
INT1    b   INT1              ;0x02; external user interrupt HOLD/#1
INT2    b   INT2              ;0x04; external user interrupt #2 and #3
TINT    b   TINT              ;0x06; internal timer interrupt
RINT    b   RINT_IRQ          ;0x08; Serial Port receive interrupt
XINT    b   XINT              ;0x0A; Serial Port transmit interrupt
TXRXINT b   TXRXINT           ;0x0C; TDM receive interrupt
```