TEXAS INSTRUMENTS

Interfacing the TLV1544 Analog-to-Digital Converter to the TMS320C50 DSP

Application Report

December 1998

Mixed-Signal Products

SLAA025A

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.

Copyright © 1998, Texas Instruments Incorporated

Contents

List of Figures

List of Tables

Interfacing the TLV1544 Analog-to-Digital Converter to the TMS320C50 DSP

Thomas Kugelstadt and David Quach

ABSTRACT

This application report presents two hardware solutions for interfacing the TLV1544 10-bit low-power analog-to-digital converter (ADC) to the TMS320C50 16-bit fixed-point digital signal processor (DSP). The report describes the interface hardware and shows three C-callable software routines which support the data transfer. In addition, it provides useful hints on the design of a typical system power supply and shows two input buffers for the analog inputs of the ADC.

1 Introduction

The TLV1544/48 is a 10-bit, low-power, successive approximation analog to digital converter (ADC) with a conversion time of $t_{conv} \leq 10\ \mu s$. The TLV1544 has four analog inputs; the TLV1548 has eight.

The device operates from a maximum supply voltage of V_{DD} = 5.5 V down to a minimum supply voltage of V_{DD} = 2.7 V, thus making it suitable for portable, low-power applications.

With a 5.5-V supply, a maximum ADC interface clock (I/OCLK) of 10 MHz is possible. With a 2.7-V supply, the maximum clock is 2.89 MHz.

The DSP serial interface port standard configuration provides only a 5-MHz clock and it is applied when the ADC operates at 5 V.

A second serial port configuration uses the DSP on-chip timer to generate a programmable clock rate between 0 Hz and 5 MHz. This configuration is used to clock the interface at 2 MHz when the ADC operates at 2.7 V.

In addition, this solution allows increasing the data throughput by a factor of two. The interface configuration possibilities are discussed in detail in Section 3.2.

2 The System

While this report focuses on the interface between the ADC and DSP (see Figure 1), it includes informative hints on the power supply section and on the buffering of the ADC inputs.

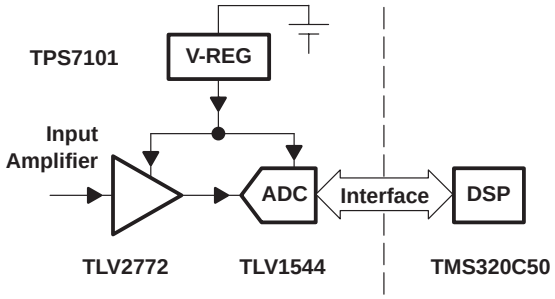


Figure 1. Data Acquisition System Using the TLV1544 ADC

2.1 Standard ADC-DSP Interface

Figure 2 shows the standard ADC/DSP interface for a fixed data clock rate at CLKX. The regulated supply voltage of V_{DD} = 5 V is applied to the ADC V_{CC} pin. A 4.7-µF bulk capacitor keeps the entire circuit supply stable against any significant current changes during ADC operation. The 0.1-µF bypass capacitors keep the ADC supply, as well as the positive reference voltage and the unused digital inputs, ripple-free. The bypass capacitors should be as close as possible to the individual pins.

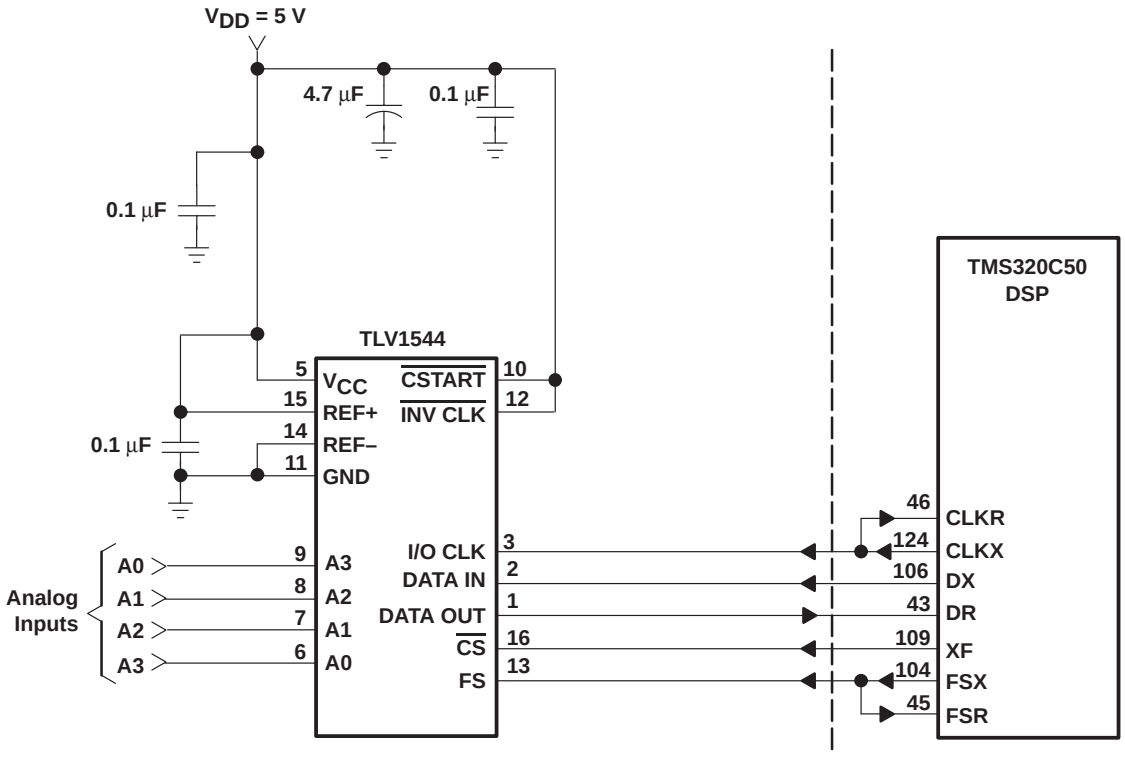
The positive reference voltage (REF+), is tied to V_{CC}; the negative reference voltage, REF–, is connected to GND. This defines the analog conversion range of the ADC and specifies the maximum input signal level at the analog inputs, A0 to A3.

The unused, low active, digital control pins are tied to V_{CC}.

In the application with fixed data clock, the interface between the TLV1544 ADC and the TMS320C50 DSP requires no additional control logic.

- The DSP sends an initial chip-enable signal via the XF pin to the $\overline{CS}$ pin of the ADC.
- The transmit clock output of the DSP, CLKX, provides a fixed data clock into the I/O CLK input of the ADC and into the receive clock input, CLKR.
- The transmit frame-sync output (FSX), initializes every data transfer by sending a frame-sync pulse to the ADC FS input as well as to the receive frame-sync input, FSR.
- The DSP initializes the ADC by transferring 4-bit control words from the DX output into the DATA IN input of the ADC.
- The ADC clocks digital conversion results out at DATA OUT into the DR-pin of the DSP.

Section 3.1 gives detailed information on signal sequence, word-length and clock rates.



NOTE: Fixed data clock is only possible with V_{DD} = 5 V.

Figure 2. Standard ADC/DSP Interface for a Fixed Data Clock at CLKX

2.2 Power Supply Circuit

Figure 3 shows a typical voltage regulator schematic using the adjustable low-dropout regulator (LDO) TPS7101. The LDO has a very low dropout voltage of 32 mV at an output current of I_{OUT} = 100 mA.

Its very low typical quiescent current of 285 µA remains independent of output loading over the full range of output current, 0 mA to 500 mA.

The TPS7101 regulates an input voltage in the range of 6 to 10 V dc down to the adjusted output level. In this application the output is adjusted to 5.1 V through the voltage divider R1 and R2.

The equation governing the output voltage is:

$$V_O = V_{REF} \times \left(1 + \frac{R1}{R2}\right) \text{ with Vref = reference voltage of 1.178 V typ.} \quad (1)$$

Resistors R1 and R2 should be chosen for approximately 7 µA divider current. The recommended value for R2 is 169 kΩ with R1 adjusted for the desired output voltage. Smaller resistors can be used, but offer no inherent advantage and consume more power. Larger values of R1 and R2 should be avoided, as leakage currents at the FB pin will introduce an error. Solving Equation 1 for R1 yields a more useful equation for choosing the appropriate resistance:

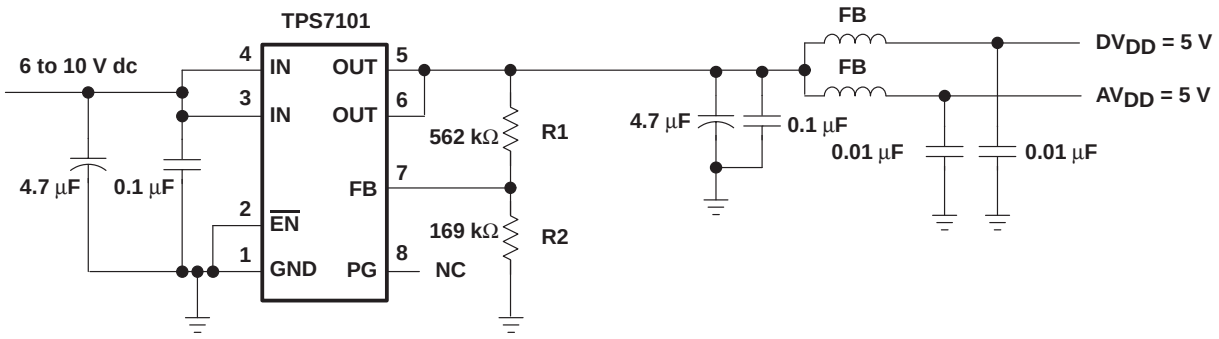
$$R1 = \left(\frac{V_O}{V_{ref}} - 1\right) \times R2 \qquad (2)$$

Calculating R1 for V_{OUT} = 5.1 V and R2 = 169 kΩ results in:

$$R1 = \left(\frac{5.1\ V}{1.178\ V} - 1\right) \times 169\ k\Omega = 562\ k\Omega$$

In the low-power application the TLV1544 ADC operates at the minimum supply voltage of V_{CC} = 2.7 V. Therefore, the value of R1 changes to:

$$R1 = \left(\frac{2.7\ V}{1.178\ V} - 1\right) \times 169\ k\Omega = 218\ k\Omega$$



NOTES: A. R1 = 562 kΩ for V_{DD} = 5.1 V and 218 kΩ for V_{DD} = 2.7 V
B. Bypass capacitors should be placed as close to the device pins as possible. The total ESR of the regulator output capacitors must maintain between 0.7 Ω min and 2.5 Ω max.
C. FB is the Fair-Rite™ #27-44-044447 or equivalent.
Fair-Rite is a trademark of Fair–Rite Products Corporation.
D. NC = no internal connection.

Figure 3. Typical Voltage Regulator Schematic

The input capacitors shown are usually not required; however, the 0.1-µF ceramic bypass capacitor improves load transient response and noise rejection if the TPS7101 is located more than a few inches from the power supply. A higher capacitance electrolytic capacitor may be necessary if large load transients with fast rise times are anticipated.

The chosen output capacitors are required for stability. A low-ESR 4.7-µF solid tantalum capacitor and a 0.1-µF high-frequency ceramic capacitor, connected from the regulator output to ground, are sufficient to ensure stability, provided that the total ESR is maintained between 0.7 Ω and 2.5 Ω.

Two additional low-pass filters, each consisting of a ferrite bead, FB, and a 0.01-µF capacitor, block the digital noise and transients on the digital supply line, DV_{DD} , from the analog supply, AV_{DD}.

For more information on the type of ferrite beads and the selection and type of low-ESR capacitors, refer to the TPS7101 Data Sheet, literature number SLVS092F and the TLV1544 EVM User's Guide, literature number SLAU014.

2.3 Analog Input Buffer Circuit

Figure 4 and Figure 5 show the schematics of typical analog input buffers using the TLV2772 dual operational amplifier.

The TLV2772 combines high slew rate and bandwidth, rail-to-rail output swing, high output drive and excellent dc precision. The device provides 10.5 V/μs of slew rate and 5.1 MHz of bandwidth while only consuming 1 mA of supply current per channel.

This ac performance is much higher than current competitive CMOS amplifiers. The rail-to-rail output swing and high output drive make it a good choice for driving the analog input or reference of analog-to-digital converters. The device also has low distortion while driving a 600-Ω load for use in telecom systems.

This amplifier has a 360 μV input offset voltage, a 17 nV/Hz input noise voltage, and a 2 pA input bias current for measurement, medical, and industrial applications.

The device operates from a 2.5 V to 5.5 V single supply voltage. Its low power consumption makes it a good solution for portable applications.

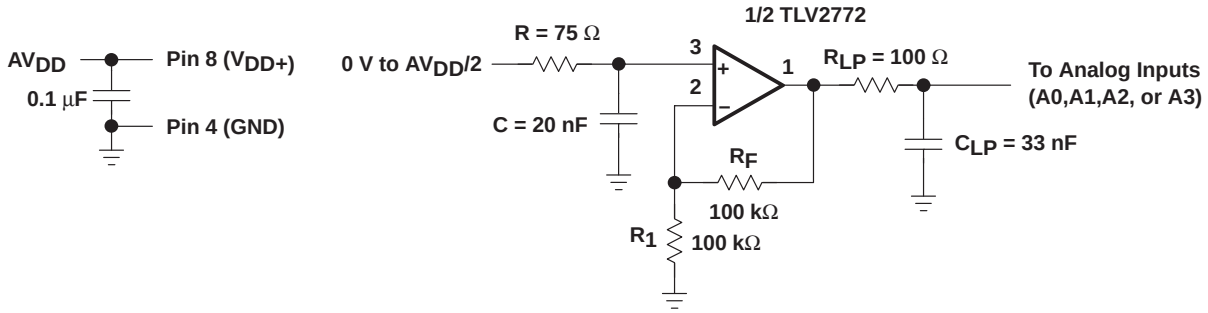


Figure 4. Schematic of a Noninverting Analog Input Buffer

In this configuration the TLV2772 works as a noninverting amplifier with a closed loop gain of two (as shown in Equation 3).

$$Gain = \frac{V_{OUT}}{V_{IN}} = 1 + \frac{R_F}{R_1} = 1 + \frac{100\ k\Omega}{100\ k\Omega} = 2 \quad (3)$$

The input voltage range is therefore limited to between 0 V and $AV_{DD}/2$. The input low-pass filter consists of an RC circuit with a corner frequency of 106 kHz. While the filter is optional, it can be useful when operating in a noisy environment.

The low-pass filter at the op-amp output (RLP and CLP), has a corner frequency of

$$fc = \frac{1}{2\pi \times R_{LP} \times C_{LP}} = \frac{1}{2\pi \times 100\ \Omega \times 33\ nF} = 48\ kHz \quad (4)$$

This filter limits the input signal bandwidth, and with it the operational amplifier inherent noise level which is fed into the ADC, thus improving the signal-to-noise ratio of the system significantly.

An additional 0.1-μF bypass capacitor, connected between V_{DD+} (pin 8) and GND (pin 4) of the device, ensures a noise-free supply for the operational amplifier.

Figure 5 shows another input buffer circuit that allows ±10-V input signals into the 5-V analog inputs of the ADC.

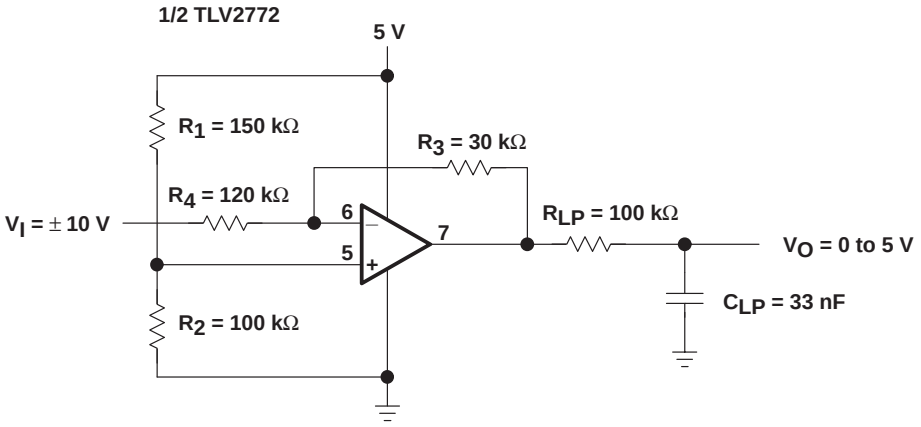


Figure 5. Schematic of an Inverting Input Buffer for Larger Input Signals

In this configuration the TLV2772 operates as an inverting amplifier with a closed loop gain of 0.25 (as shown in Equation 4).

$$Gain = \frac{V_{OUT}}{V_{IN}} = -\frac{R_3}{R_4} = -\frac{30\ k\Omega}{120\ k\Omega} = -0.25 \quad (5)$$

Resistors R_1 and R_2 bias the output of the operational amplifier to a 2.5-V operating point. A maximum input voltage of ±10-V is amplified by –0.25 and results in an output voltage of ±2.5 V. The negative sign in the gain factor represents a phase shift of 180° between input- and output-signal.

3 ADC Overview

The TLV1544 and TLV1548 are CMOS 10-bit switched capacitor successive approximation (SAR) ADCs. The TLV1544 has four analog inputs; the TLV1548 has eight. Figure 6 shows a functional block diagram of the devices.

The TLV1544 operates from a minimum supply of 2.7-V to a maximum supply of 5.5-V and allows high-speed data transfer from the host of up to 10-MHz maximum. In addition to the on-chip multiplexer that can select any one of the analog inputs or any one of the three internal self-test voltages, the ADC provides a versatile control capability. Through the DATA IN pin, 4-bit control words initialize the device for:

- Any one of the analog input channels
- Power-down mode
- Slow or fast conversion rate
- Any one of the self-test voltages.

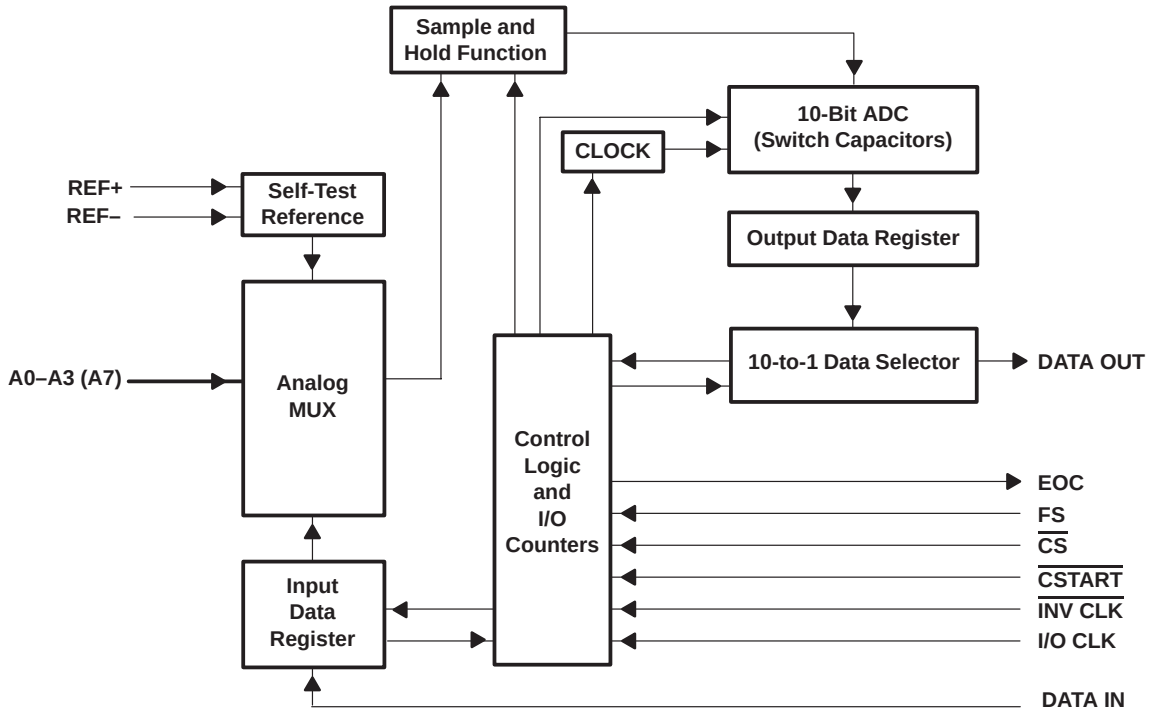


Figure 6. Functional Block Diagram of the TLV1544 and TLV1548

A 4-wire serial interface (SPI ™, QSPI ™) allows data transfer to a microprocessor or DSP. When interfacing to a TMS320 DSP, an additional frame sync (FS) signal indicates the start of a serial data frame. A high at the chip-select pin $\overline{CS}$ activates the device. The data clock at I/O CLK determines the data rate between ADC and host. Through DATA IN, 4-bit control words initialize the ADC for the desired operation mode and select the analog input. The EOC pin indicates the end of a conversion and DATA OUT provides the conversion results in a 10-bit serial format.

SPI and QSPI are trademarks of Motorola, Inc.

A high at the $\overline{\text{INV CLK}}$ pin allows for I/O clock phase adjustment of 180°.When operating in the extended sampling mode, the $\overline{\text{CSTART}}$ pin controls the sampling period of the sample-and-hold circuit and starts the conversion. In this application, $\overline{\text{INV CLK}}$ and $\overline{\text{CSTART}}$ are not used and are therefore tied high.

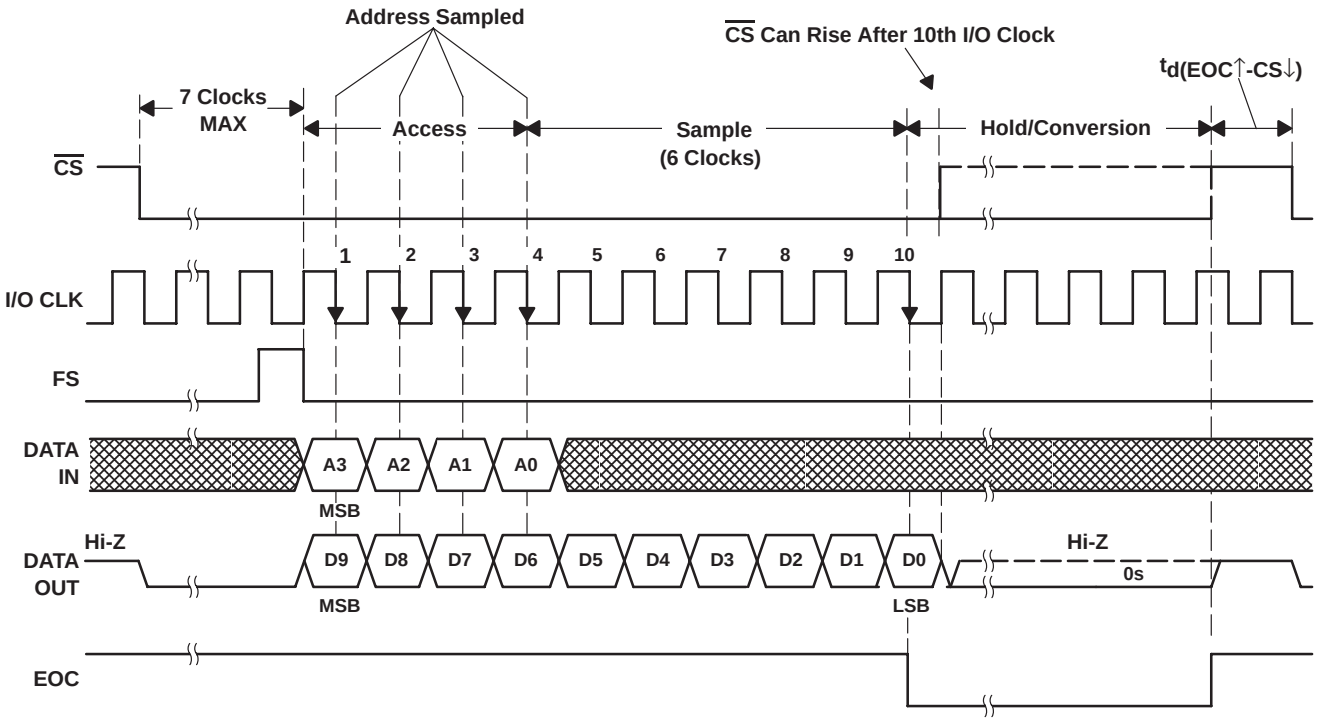
3.1 Signal Sequence

Figure 7 shows the timing diagram of a data transfer between the TLV1544 ADC and the TMS320C50 DSP through the DSP serial port. The ADC serves as a slave device and the DSP operates as the master, supplying the frame-sync signal, FS, and the data transfer clock, I/O CLK. Initially, with $\overline{\text{CS}}$ high and the ADC being inactive, the inputs, DATA IN and I/O CLK, are disabled. DATA OUT is in the high-impedance state and EOC is high.

When the DSP activates the ADC by taking $\overline{\text{CS}}$ low and providing the data clock to I/O CLK, the data transfer sequence begins. I/O CLK and DATA IN are enabled and DATA OUT is removed from the high-impedance state to logic low. The DSP then sends out a FS pulse on the FS line, indicating the start of a data frame.

With the falling edge of FS, the DSP provides the 4-bit control word to DATA IN (see Table 1 for ADC control words) starting with the most significant bit (MSB). At the same time, the ADC provides a 10-bit conversion result (from the previous conversion) at DATA OUT, beginning with MSB.

The input data selects a different mode or selects different analog input channels. In the case of the control word being a channel address, the selected analog input channel is accessed during the first four I/O clock cycles after the falling edge of FS. Starting with the falling edge of the fourth I/O cycle, the sample-and-hold (S&H) circuit samples the selected analog input.



NOTES: E. The falling edge of FS has to appear within 7 I/O clock cycles from the falling edge of $\overline{CS}$.

F. The TLV1544/48 data sheet specifies a minimum time of t_d = 100-ns after the rising edge of EOC before the next falling edge of $\overline{CS}$.

Figure 7. DSP Interface Timing (16-Clock Transfer, Normal Sample Mode, $\overline{INV\ CLK}$ = High)

At the tenth falling edge of I/O CLK, the sample is held and the analog-to-digital conversion starts. At the same time the EOC output changes from high to low, indicating the conversion start. The DSP initiates the end of a data transfer sequence by taking $\overline{CS}$ high, which disables DATA IN, I/O CLK and DATA OUT. This can happen between the end of the tenth I/O clock and the rising edge at EOC, indicating end-of-conversion.

If $\overline{CS}$ is taken high immediately after the tenth clock, DATA OUT goes into high-impedance state and, following the 10-bit conversion result, random signal levels are clocked into the DSP for the next six clock cycles. If $\overline{CS}$ changes to high somewhere between the end of the tenth clock and EOC going high, the 10-bit result is padded with a maximum of six zeros to complement the 16-clock cycle of the DSP.

In any case, the six bits following the conversion result are useless information and should be ignored by the DSP interface software. The entire data transfer sequence is complete when EOC returns to high.

A delay time of $t_{d(EOC\uparrow - CS\downarrow)}$ = 100 ns after the rising edge of EOC is required before $\overline{CS}$ can change to low again to start a new data transfer.

Table 1. TLV1544/TLV1548 ADC Control Words

FUNCTION SELECT	INPUT DATA BYTE A3 – A0		COMMENT
	BINARY	HEX	
Analog channel A0 for TLV1548 Selected	0000b	0h	Analog channel 0 for TLV1544
Analog channel A1 for TLV1548 Selected	0001b	1h	
Analog channel A2 for TLV1548 Selected	0010b	2h	Analog channel 1 for TLV1544
Analog channel A3 for TLV1548 Selected	0011b	3h	
Analog channel A4 for TLV1548 Selected	0100b	4h	Analog channel 2 for TLV1544
Analog channel A5 for TLV1548 Selected	0101b	5h	
Analog channel A6 for TLV1548 Selected	0110b	6h	Analog channel 3 for TLV1544
Analog channel A7 for TLV1548 Selected	0111b	7h	1001b
Software power down set	1000b	8h	No conversion result (cleared by any access)
Fast conversion rate (10 μs) set	1001b	9h	No conversion result (cleared by setting to fast)
Slow conversion rate (40 μs) set	1010b	Ah	No conversion result (cleared by setting to slow)
Self-test voltage ($V_{ref+} – V_{ref–}$)/2 selected	1011b	Bh	Output result = 200h
Self-test voltage $V_{ref–}$ selected	1100b	Ch	Output result = 000h
Self-test voltage V_{ref+} selected	1101b	Dh	Output result = 3FFh
Reserved	1110b	Eh	No conversion result
Reserved	1111b	Fh	No conversion result

3.2 The ADC/DSP Interface

Following the signal sequence description in Section 3.1 and recalling that the DSP is the master and the ADC the slave device, the following interconnections always stay the same:

XF – The XF output drives the chip-enable signal into the $\overline{CS}$-pin of the ADC.

FSX/FSR – The frame-sync pulse is generated by the DSP. This signal is sent from the FSX output into the FS input of the ADC and into the FSR input of the DSP to synchronize the transmitter and receiver stages within the serial port.

DX – Control data, which initialize the TLV1544, are driven from the DSP data transmit output, DX into DATAIN of the ADC.

DR – Conversion results from the ADC are fed into the data receive input, DR.

This section focuses mainly on the different options for generating the I/O clock and detecting the end-of-conversion that are possible with the TLV1544/48.

Figure 8 shows the standard interface between a TLV1544/48 ADC and a TMS320C50 DSP with a glueless interface.

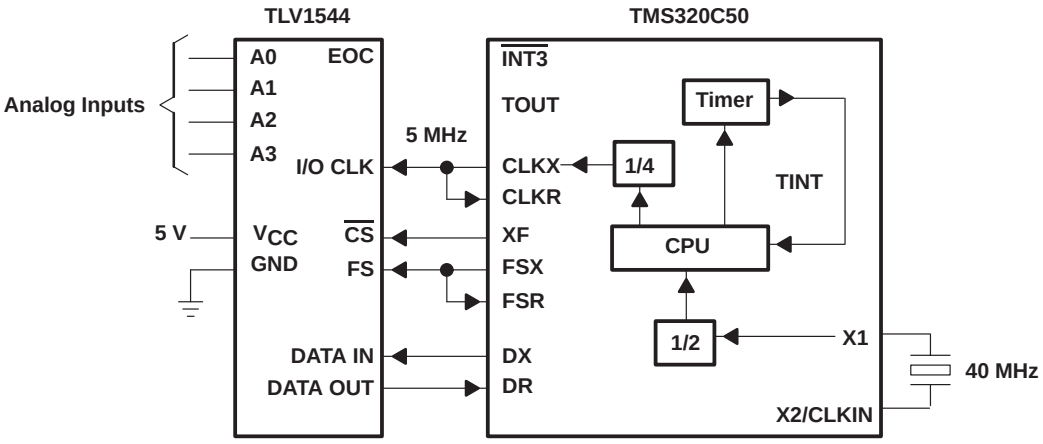


Figure 8. Glueless ADC/DSP Interface

The DSP provides an on-chip oscillator, which is enabled by connecting a crystal, with a fundamental frequency of 40 MHz, across X1 and X2/CLKIN. An internal divide-by-two clock option provides the CPU clock of 20 MHz, which translates to an instruction cycle time of 50 ns.

The generation of the I/O clock happens with the initialization of the serial port, which provides an additional divide-by-four function and sets the data transfer clock at CLKX to 5 MHz.

The information on the end-of-conversion status is important, since a previous conversion must be completed before a new conversion can start. The application in Figure 8 does not detect the end-of-conversion signal, but waits for the ADC maximum conversion time to pass. The TLV1544 data sheet specifies the typical and the maximum conversion times for fast and slow conversion, as shown in Table 2.

Table 2. Conversion Times for Fast and Slow Conversion

PARAMETER		LAB	DATA SHEET	UNIT
Conversion time, t_{conv}	Fast conversion from V_{CC} = 5.5 V to 3.3 V	5	10	µs
	Slow conversion from V_{CC} = 5.5 V to 2.7 V	13	25	

In the interface software, the DSP on-chip timer-counter is programmed with the maximum conversion time for the selected conversion mode and causes a timer interrupt (TINT) once the counter value reaches zero. The following interrupt routine can then initiate a new data transfer sequence.

Table 2 compares the conversion times, measured on a series of TLV1544 evaluation modules (EVMs) with the maximum values specified in the data sheet. The lab data differ by a factor of two from the maximum possible conversion times. It is obvious that, even with a glueless interface, waiting for the maximum conversion time to pass is highly time-inefficient. To maximize the data throughput over the interface, the end of a conversion must be detected as soon as it occurs. Only then is it possible to start a new data transfer with minimum latency. The circuit in Figure 9 accomplishes this by connecting the EOC output of the ADC through an inverter to the external interrupt input, $\overline{INT3}$ of the DSP.

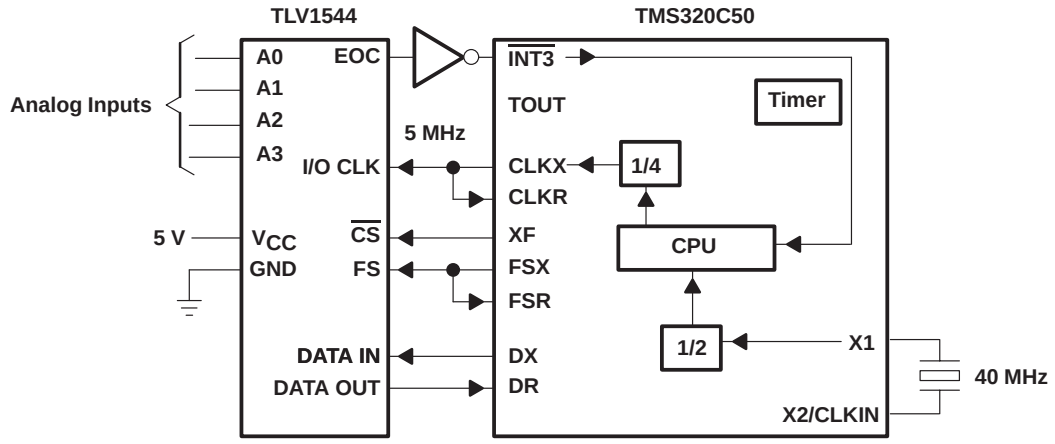


Figure 9. Maximum Data Throughput via Immediate EOC Detection

The end-of-conversion signal (rising edge at EOC) is inverted into a high-to-low transition at $\overline{\text{INT3}}$ and causes an external interrupt. The initiated interrupt service routine can now start a new data transfer sequence.

The two previous scenarios describe the interface for 5-V operation. For low-power applications with supply voltages of 3 V and less, the TLV1544 data sheet defines lower data clock frequencies at I/O CLK, when compared to 5-V operation.

Table 3. Clock Frequencies at I/O CLK for 5.5-V and 2.7-V Operation

PARAMETER		MIN	TYP	MAX	UNIT
Clock frequency at I/O CLK, f_{CLK}	V_{CC} = 5.5 V	0.1	6	10	MHz
	V_{CC} = 2.7 V	0.1	2	2.81	

The circuit in Figure 10 shows the ADC operating from a 2.7-V supply. The required data clock at I/O CLK is 2 MHz nominal. Since the only frequency the DSP can provide at CLKX is one fourth of the internal CPU clock (CLKX = 5 MHz), the on-chip timer is initialized as a programmable clock source.

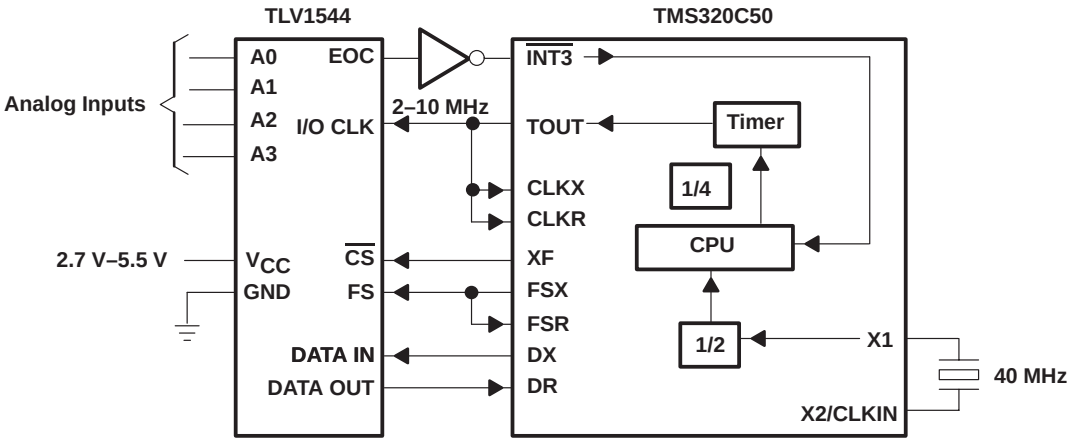


Figure 10. Serial Interface for Low-Volt Operation and High Data Throughput

The two period registers of the timer, PRD and TDDR, are programmed to divide the CPU clock of 20 MHz by 10. The timer now generates periodic interrupts at a rate of 2 MHz. While the interrupt requests are masked to the CPU, the 2-MHz signal is still available as data clock at the timer output, TOUT. For more information on the timer functions see Section 4.3. A further advantage of this adjustable clock generator is that it can be programmed for clock rates of up to one half of the CPU clock, to TOUT = 10 MHz.

Since the timer serves here as a clock generator, it is no longer available as an interrupt source to indicate the end-of-conversion (EOC) state of the TLV1544/48. As in the previous application, this circuit detects the EOC by inverting the EOC signal from the ADC into a high-to-low transition at $\overline{\text{INT3}}$. The external interrupt request to the CPU starts the interrupt service routine which initiates a new data transfer sequence. Figure 11 shows that when the external inverter is removed, the interface becomes glueless again.

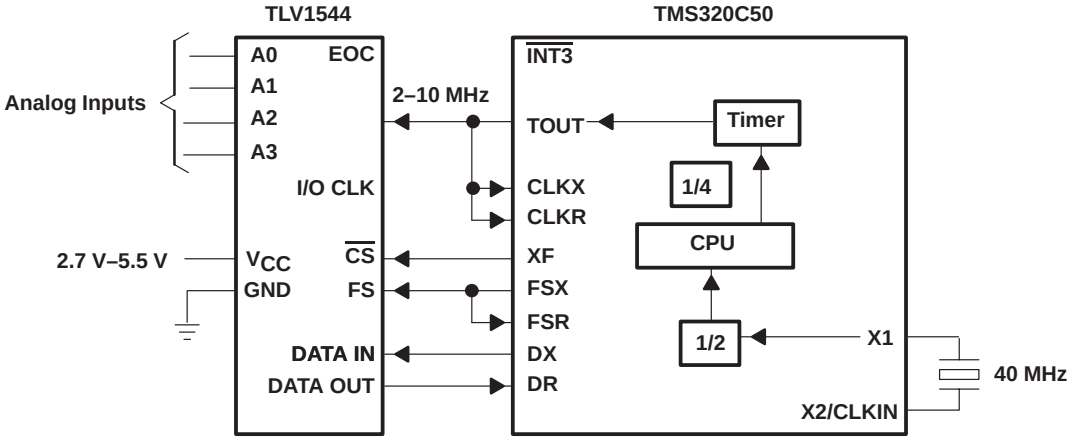


Figure 11. Glueless Interface for Low-Volt Operation

Since neither the inverter nor the timer is available for EOC detection, additional software is required to accomplish this task. A few lines of software code, forming a delay loop that waits for the ADC maximum conversion time to pass, need to be added to the interface program. The additional wait loop consumes DSP programming power and reduces the data throughput significantly. Applying this technique to a low-power application, which already operates in slow conversion mode and at a low data transfer rate, reduces the data throughput significantly. Depending on the requirements of the final application, the simplicity of a glueless interface might outweigh the benefits of a higher data throughput.

Table 4 summarizes the pros and cons of the interface circuits, shown in Figure 8 through Figure 11.

Table 4. Benefits of the Individual Interface Circuits

FIGURE	PROS	CONS
8	Glueless interface for 5-V operation at fixed data clock	Low data throughput due to waiting for maximum conversion time to pass
9	Factor 2 higher data through put for standard 5-V operation	No glueless interface due to external inverter
10	– Flexible interface for a supply range from 2.7 V to 5.5 V – Flexible data clocks from 2 MHz to 10 MHz – Higher data throughput over whole supply range	No glueless interface due to external inverter
11	– Glueless interface – Flexible interface for a supply range from 2.7 V to 5.5 V – Flexible data clocks from 2 MHz to 10 MHz	Low data throughput due to waiting for maximum conversion time to pass

The software description in Section 5 provides three examples of interface programs for the circuits in Figure 8 through Figure 11.

4 The TMS320C50 DSP

The TMS320C50 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.

4.1 Key Features

The ’C50 offers the following key features:

- **2K x 16-bit On-Chip ROM**
  This on-chip, maskable, programmable memory is used for booting from slower, external ROM or EPROM of program to fast on-chip or external SRAM.
- **1056 x 16-bit On-Chip Data RAM**
  The data RAM can be accessed twice per machine cycle (dual-access RAM). This block of memory is primarily intended to store data values but, when needed, can be used to store programs as well as data.
- **9K x 16-bit On-Chip Program/Data RAM**
  This memory is software configurable as program and/or data memory space. Code can be booted from an off-chip, nonvolatile memory and then executed at full speed, once it is loaded into this RAM.
- **On-Chip Memory Security**
  The ’C50 has a maskable option to protect the contents of on-chip memories. When the related bit is set, no externally originating instruction can access the on-chip memory spaces.
- **16 Address-Mapped Software Wait-State Generator**
  The device incorporates a software wait-state logic for interfacing with slower off-chip memory and I/O devices. This circuit consists of 16 wait-state generating circuits and is user programmable to operate 0, 1, 2, 3, or 7 wait states. For off-chip memory access, these wait state generators can be mapped on 16K-word boundaries in program memory, data memory, and to the I/O ports.
- **User-Maskable Interrupts**
  The ’C50 has four external-interrupt lines. These lines are internally latched so that asynchronous interrupt operations can be performed by the DSP. Also the device possesses five internal interrupts: the timer interrupt, TINT, and four serial port interrupts.
- **64K Parallel I/O Ports**
  The ’C50 has a total of 64K I/O ports, sixteen of which are memory-mapped in data memory space. These ports can be addressed by the IN instruction or the OUT instruction. The memory-mapped I/O ports can be accessed with any instruction that reads or writes data memory.
- **2 Serial I/O Ports**
  The two high-speed serial ports are capable of operating at up to one-fourth the machine cycle rate (CLKOUT1). One of the two circuits is a synchronous, full-duplex serial port. Its transmitter and receiver are double buffered and

individually controlled by maskable, external interrupt signals. Data is framed either as bytes or as words. The second circuit is a full-duplex serial port that can be configured either for synchronous or for time-division multiple (TDM) access operations.

- **Hardware Timer**
  The on-chip timer consists of a 16-bit counter and a 4-bit prescaler. This timer clocks between one-half and one-thirty-second the machine rate, depending upon the programmable divide-down ratio. The timer can be stopped, restarted, reset, or disabled by specific status bits.

From the list of key features, the serial port and the hardware timer are the two on-chip peripherals that are mainly used in this application. Their block diagrams and operation are, therefore, explained in more detail in the following sections.

4.2 The DSP Serial Port

The serial port provides communication with serial devices such as codecs and serial ADCs. Receive and transmit operations are double-buffered, thus allowing a continuous communication stream in either 8- or 16-bit data packets.

In the internal clock mode, the maximum transmission rate for transmit and receive operations is the CPU clock divided by four or CLKOUT1 rate/4. Therefore, the maximum rate is 5 megabits/s for a 20-MHz (50 ns) device.

Two modes of operation are provided to support a wide range of applications.

- Continuous mode provides operation that requires only one FS pulse to transmit several packets at maximum frequency.
- Burst mode allows transmission of a single 16-bit word following an FS pulse.

4.2.1 Signals and Registers

The 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 fourth 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).

The serial port operates through the three memory–mapped registers SPC, DXR, and DRR and two other registers, XSR and RSR, that are not accessible but permit double-buffering capability. These five registers are listed in Table 5.

Table 5. Serial Port Registers

REGISTERS	DESCRIPTION
SPC	Serial port control register
DXR	Data transmit register
DRR	Data receive register
XSR	Transmit shift register
RSR	Receive shift register

4.2.2 Serial Port Operation

Figure 12 shows how the pins and registers are configured on the serial port and how the double buffering is implemented.

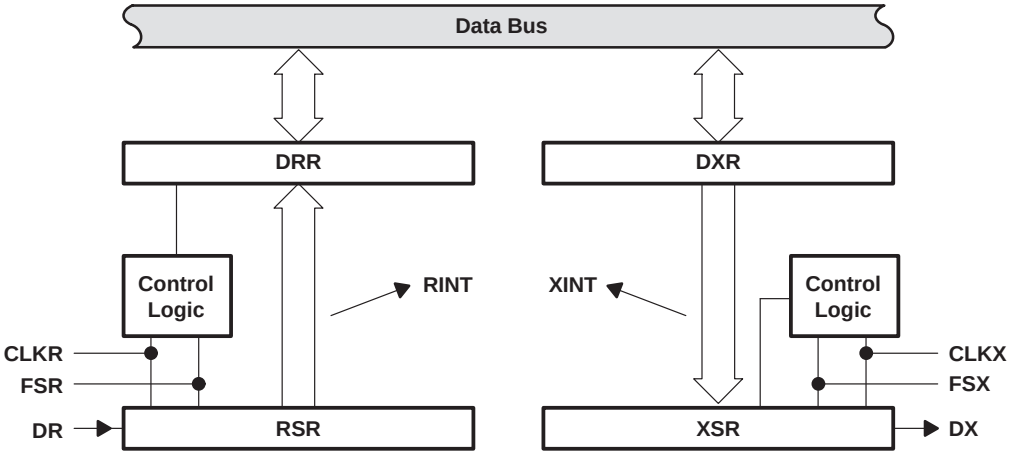


Figure 12. Serial Port Block Diagram

Transmit data is written to the DXR, while received data is read from the DRR.

A transmit is executed by writing data to the DXR, which copies the data to the XSR. The XSR manages the shifting of the data to the DX pin, thus allowing another write to DXR as soon as the DXR-to-XSR copy is completed.

Upon completion of the DXR-to-XSR copy, a 0-to-1 transition occurs on the transmit-ready bit, XRDY, in the SPC, and generates a transmit interrupt, XINT, that signals to the CPU that DXR is ready for a new word. The process is similar on the receive side.

Data from the DR pin is shifted into the RSR, which copies it to the DRR from which it may be read. Upon completion of the RSR-to-DRR copy, a 0-to-1 transition occurs on the receive ready bit, RRDY, in the SPC, and generates a receive interrupt, RINT.

Thus, the serial port is double-buffered because data can be transferred to or from DXR or DRR while another transmit or receive is being performed.

In burst mode, the frame sync pulse synchronizes the transfer timing as described in section 4.2.4 transmit and receive operations in burst mode.

4.2.3 Serial Port Configuration

While the data registers are mainly responsible for shifting and buffering data, the SPC configures the entire serial port and is therefore the most important register used in the interface programs. Figure 13 shows the 16-bit memory-mapped SPC. Some of the bits are read-only while others are read/write.

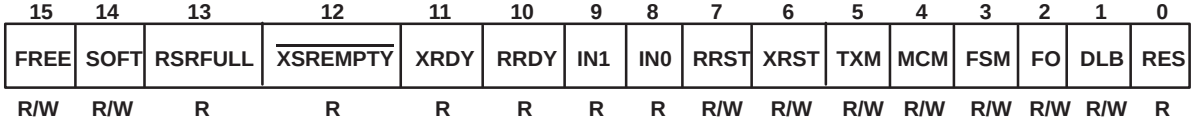


Figure 13. Serial Port Control Register

Table 6 gives an overview of the bit definition of the SPC. Only the bits which are important to this application are explained. For detailed information on the bit functions refer to the TMS320C5x User's Guide.

Table 6. Serial Port Control Register Bits Summary

BIT	NAME	FUNCTION
0	Reserved	Always read as zero.
1	DLB	The digital loop-back mode bit allows to connect the transmitter output (DX and FSX) to the receiver input (DR and FSR). *(This bit is not used in the application and set to zero.)*
2	FO	The format bit specifies the word length of the transmitter and receiver. FO = 0, word length is 16 bit FO = 1, word length is 8 bit
3	FSM	The frame synch mode bit specifies when a frame sync pulse is needed. FSM = 1, burst mode is selected (an FS-pulse is used for each word) FSM = 0, continuous Mode is selected (only one start pulse is required)
4	MCM	The clock mode bit specifies the clock source for CLKX. MCM = 1, on-chip clock source is used with CLKX = 1/4 CLKOUT1 MCM = 0, external clock source is chosen
5	TXM	The transmit mode bit specifies the source for FSX-pulse generation TXM = 1, FSX is generated on–chip and synchronized to CLKX TXM = 0, FSX needs to be applied from extern
6 7	XRST RRST	The transmit and receive reset signals activate and deactivate the transmitter and receiver of the serial port. XRST/ RRST = 1, transmitter and receiver are active XRST/ RRST = 0, activity halts
8 9	IN0 IN1	The input 0 bit and input 1 bit reflect the levels of the CLKX and CLKX pins. IN0 and IN1 are read only bits.
10 11	RRDY XRDY	The receive and transmit ready bits. Upon the completion of a RSR–to–DRR copy or a DXR-to-XSR copy, these bits perform a 0–1 transition, indicating a receive interrupt (RINT or a transmit interrupt (XINT) respectively.
12	$\overline{\text{XSREMPTY}}$	The transmit shift register empty flag indicates whether the transmitter has experienced underflow. $\overline{\text{XSREMPTY}}$ is a read only bit.

Table 6. Serial Port Control Register Bits Summary (Continued)

BIT	NAME	FUNCTION
13	RSRFULL	The receive shift register full flag indicates whether the receiver has experienced overrun. RSRFULL is a read only bit.
14	SOFT	The SOFT bit is an emulation bit that aborts transmission when a breakpoint is encountered in the high-level language debugger. It is enabled when the FREE bit is 0. SOFT = 1, stop after word completion SOFT = 0, immediate stop (This bit is effective only in emulation mode, otherwise it is set to zero).
15	FREE	The FREE bit selects the free run of CLKX. FREE = 1, CLKX runs free FREE = 0, STOP bit is enabled (This bit is effective only in emulation mode, otherwise it is set to zero).

When interfacing to the TLV1544/48 ADC, the serial port of the TMS320C50 DSP must be configured as shown in Table 7.

Table 7. SPC Configuration for Interfacing to the TLV1544/48 ADC

REQUIRED CONFIGURATION	AFFECTED BITS
The DSP needs to be set-up as the master device, generating the necessary frame sync pulse to start a data transfer.	TXM = 1
Because of serial port inactivity during the conversion process of the ADC, the serial port must operate in burst mode.	FSM = 1
The communication between ADC and DSP must use the 16-bit word format to support the 10-bit format of the ADC conversion result.	FO = 0
When the ADC operates at 5 V, the DSP on-chip clock source can be selected to provide CLKX as the data transfer clock to the I/O CLK input of the ADC. When the ADC operates at 2.7 V, the CLKX pin of the DSP needs to change to an input, to receive the clock signal from the timer output, TOUT.	MCM = 1 MCM = 0
When the SPC is to be modified to reconfigure the serial port, transmitter and receiver need to be reset. After the modification is complete, transmitter and receiver need to be activated.	XRST / RRST = 0 XRST / RRST = 1

With the exception of the MCM bit, which needs to comply with the specific ADC clock requirements at 5-V and 2.7-V supply, the remaining bits of the SPC stay the same.

When reconfiguring the SPC, two instructions are required. The first instruction resets the transmitter and receiver and configures the SPC. The second instruction reactivates the transmitter and receiver.

Figures 14 and 15 show the binary format and the hex code of the assembler instructions used to configure and to activate the serial port.

		FREE	SOFT	RSRFULL	XSREMPTY	XRDY	RRDY	IN1	IN0	RRST	XRST	TXM	MCM	FSM	FO	DLB	RES
Configure Port	SPLK #0038h, SPC	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0
		0				0				3				8			
Activate Port	SPLK #00F8h, SPC	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
		0				0				F				8			

Figure 14. Configuring and Activating the Serial Port for 5-V Operation of the ADC

		FREE	SOFT	RSRFULL	XSREMPTY	XRDY	RRDY	IN1	IN0	RRST	XRST	TXM	MCM	FSM	FO	DLB	RES
Configure Port	SPLK #0028h, SPC	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
		0				0				2				8			
Activate Port	SPLK #00E8h, SPC	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0
		0				0				E				8			

Figure 15. Configuring and Activating the Serial Port for 2.7-V Operation of the ADC

4.2.4 Transmit and Receive Operations 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 transmit direction, after a write to DXR, a FSX pulse 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 fall 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 receive direction, the shifting into RSR begins on the falling edge of the CLKX cycle after the FX has gone low. After all bits have been received, the content of the RSR is transferred to the DRR on the falling edge of CLKX. RRDY goes high, generating a receive interrupt, RINT.

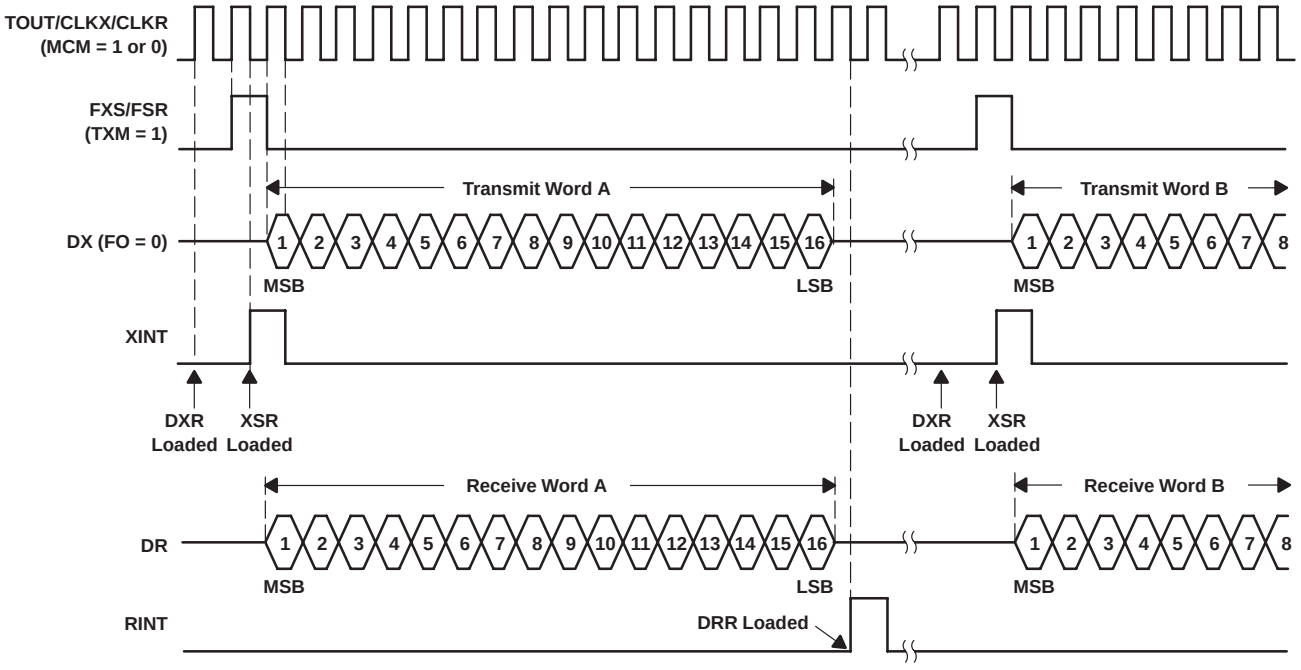


Figure 16. Transmit and Receive Operation in Burst Mode

4.3 The Hardware Timer

The second on-chip peripheral used in this application is the hardware timer. When the ADC operates at 3V or less, the timer serves as the data clock source, providing the necessary low clock rate of 2 MHz.

In applications where a slow data clock is not required, the timer can be programmed as an EOC indicator.

4.3.1 Timer Operation

The hardware timer is a fully programmable down counter. It can be stopped, restarted, reset, or disabled by specific status bits. The timer consists of a 16-bit main counter, TIM, and a 4-bit prescaler counter, PSC.

Figure 17 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, and the TIM is loaded by the period register, PRD.

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. In addition, the borrow-2 pulse is sent as a TOUT-pulse to the external timer output pin, TOUT, and as a timer interrupt, TINT, to the CPU.

The TINT request sets the TINT flag in the interrupt flag register, IFR, and can be masked or unmasked in the interrupt mask register, IMR.

The duration of a borrow-2 pulse at TOUT is equal to that of a CLKOUT1 cycle.

The timing diagram in Figure 18 gives a typical sequence of events for the two timer values, PSC = 4 and TIM = 2.

- An initial load-pulse, at Load1 and Load2, copies the TDDR value of 4 into the PSC, and the PRD value of 2 into the TIM.
- The PSC decrements on each succeeding CLK-1 cycle (CLKOUT1) until it reaches zero.
- On the next CLK-1 cycle, a borrow-1 is generated. The TDDR loads the new divide-down count into the PSC, and the TIM decrements by one.
- The PSC and the TIM continue to decrement in the same way until the TIM decrements to zero.
- On the next CLK-2 cycle, a borrow-2 pulse is generated. A timer interrupt, TINT, is sent to the CPU, and the borrow-2 pulse is sent to the TOUT-pin. Both counters, TIM and PSC, are reloaded by their corresponding registers, PRD and TDDR, and the entire sequence can be repeated.

The TIM decrements by one, every (TDDR + 1) CLKOUT1 cycles. When either one of the two period registers, or both registers, PRD and TDDR, are nonzero, the interrupt rate is defined by equation 5 and the interrupt interval is given by equation 6.

$$TINT[MHz] = \frac{CLKOUT1[MHz]}{(TDDR + 1) \times (PRD + 1)} \quad (6)$$

$$t_{TINT} = t_C \times (PRD + 1) \times (TDDR + 1) \quad (7)$$

For the example in Figure 18, the TINT rate results in:

$$TINT[MHz] = \frac{CLKOUT1[MHz]}{(4 + 1) \times (2 + 1)} = \frac{CLKOUT1[MHz]}{15}$$

That means the CPU clock, CLKOUT1, is divided by 15, or every 15 CPU clock cycles, a TINT pulse is generated.

NOTE: When both registers, PRD and TDR, are zero, the timer interrupt rate automatically sets to CLKOUT1/2.

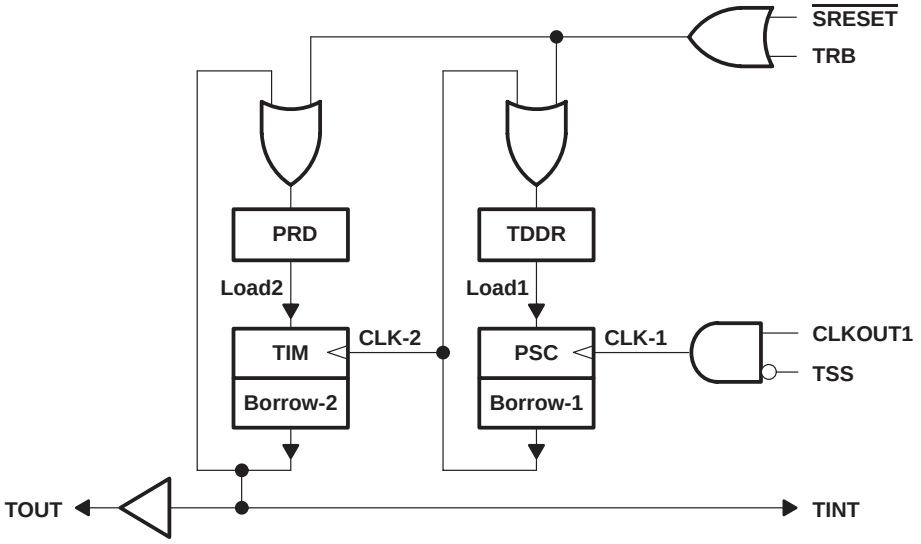


Figure 17. Timer Block Diagram

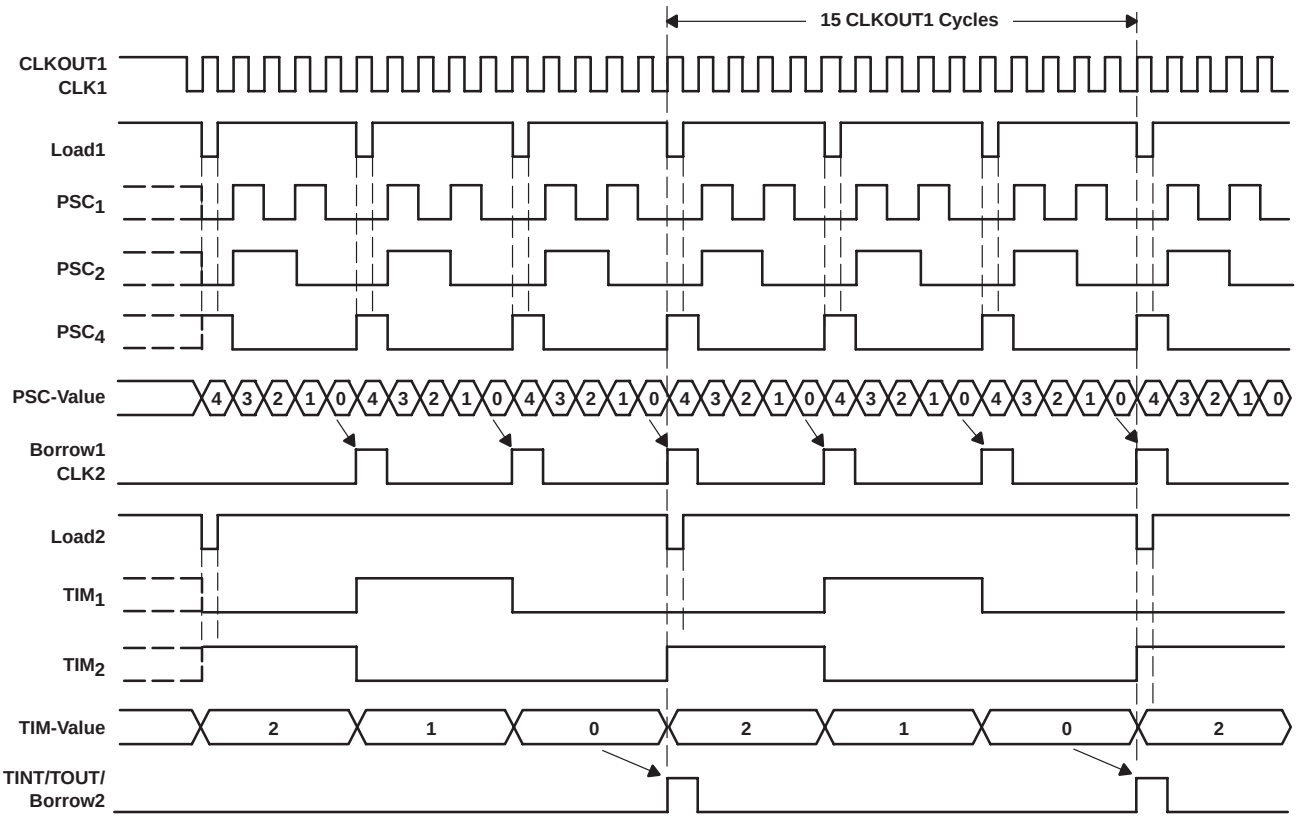


Figure 18. Timing Diagram for TDDR = 4 and PRD = 2

4.3.2 Programming the Timer

The on-chip timer is configured by the period register, PRD, and the timer control register, TCR. The PRD is a 16-bit, memory-mapped register that specifies the initial period of the timer. The TCR is a 16-bit, memory-mapped register that contains the status and control bits to operate the timer. Figure 19 shows the status and control bits within the TCR, and Table 8 gives an overview of the bit definition of the TCR.

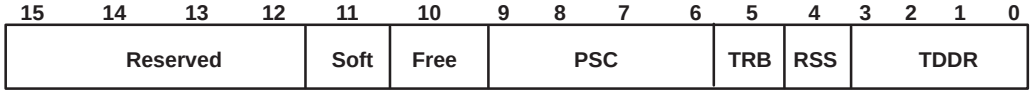


Figure 19. Timer Control Register

Table 8. Timer Control Register Bits Summary

BIT	NAME	FUNCTION
0 – 3	TDDR	The timer divide-down register bits define the divide-down ratio for the timer.
4	TSS	The timer stop status bit starts or stops the timer. TSS = 0, Timer starts TSS = 1, Timer stops
5	TRB	The timer reload bit resets the timer. When the TRB is set, the TIM is reloaded with the value in the PRD, and the PSC is loaded with the value in the TDDR. *(This bit is always read as zero).*
6 – 9	PSC	The prescaler counter bits hold the current prescale count for the timer. The PSC is reloaded with the contents of the TDDR one cycle after it has reached zero, or whenever the TRB is et by software.
10	FREE	The FREE bit selects the free run of the timer. FREE = 1, Timer runs free FREE = 0, STOP bit is enabled (This bit is effective only in emulation mode, otherwise it is set to zero).
11	SOFT	The SOFT bit is an emulation bit that aborts transmission when a breakpoint is encountered in the high–level language debugger. It is enabled when the FREE bit is 0. SOFT = 1, Timer stops after completion of transmission SOFT = 0, Timer stops immediately (This bit is effective only in emulation mode, otherwise it is set to zero).
12 – 15	Reserved	Always read as zero.

When programming the timer interrupt rate using equation 6, it is common to specify the initial timer period within the PRD and to keep the TDDR value within the TCR at zero. By programming the PRD from 0 to 65536, it is possible to generate timer interrupts, TINT, every 2 to 65536 CPU cycles. If a lower interrupt rate is required, an additional divide-down value can be programmed into the TDDR that extends the interrupt interval by the desired factor.

In the software description in Section 5, Program 1 uses the on-chip timer as an end-of-conversion indicator for the ADC. Thus the timer has to generate only one interrupt, TINT, for the length of 10 μs (which is the maximum conversion time of the ADC). With a CPU clock of 20 MHz, or a clock cycle of t_C = 50 ns, the interrupt period needs to be 200 CPU cycles.

Solving equation 7 for PRD and keeping TDDR = 0, results in:

$$PRD = \frac{t_{TINT}}{t_C \times (TDDR + 1)} - 1 = \frac{10\ \mu s}{50\ ns \times (0 + 1)} - 1 = 199$$

The following instructions configure the timer in program 1:

```
SPLK #199, PRD    ; load timer period of 10 μs into PRD
SPLK #0030h, TCR  ; set TRB = 1 to reload PRD/TDDR and set TSS = 1 to stop timer
SPLK #0008h, IFR  ; clear any pending timer interrupt (TINT)
```

Program 2 uses the on-chip timer as a programmable clock generator to generate a 2-MHz interface clock. With a CPU clock of 20 MHz and an interrupt rate of 2 MHz, the PRD value needs to be:

$$PRD = \frac{CLOCKOUT1}{TINT\text{–}rate \times (TDDR + 1)} - 1 = \frac{20\ MHz}{2\ MHz \times (0 + 1)} - 1 = 9$$

The following instructions configure the timer in program 2:

```
SPLK #9, PRD      ; load timer period of 10 µs into PRD
SPLK #0020h, TCR  ; set TRB = 1 to reload PRD/TDDR and set TSS = 0 to run timer
```

NOTE: Since the timer operates as a clock generator, the timer interrupt, TINT, is masked for the entire program. Therefore, an additional instruction to clear any pending interrupts is not required.

5 Software Description

The interface software consists of three C-callable assembler programs and three corresponding C-programs. The user can specify certain data transfer parameters—such as the analog input channel, the conversion mode, the memory start address, and the number of samples—through the global variables of the .C file without modifying the corresponding assembler program.

When a specific .C interface routine is called, the global variables in the specific C file are loaded into the local variables of the assembler program.

Table 9 shows the global and local variables and their corresponding files.

Table 9. Local and Global Variables and Corresponding Programs

PROGRAM	TYPE	VARIABLE	DESCRIPTION
All C–files	Global	_Samples	User defined number of samples to be acquired
	Global	_MemStart	User defined memory pointer start
	Global	_Channel	User defined channel number to be sampled
C1544T.C only	Global	_ControlMode	User defined conversion mode
C1544C.C only	Global	_Vcc	User defined ADC voltage supply
All Assembler files	Local	ADWORD	ADC Control word
	Local	ADCOUNT	Sample counter
	Local	ADMEM	Memory pointer
	Local	RINT_COUNT	Number of Receive Interrupts
	Local	END_BIT	End-of-program bit
	Local	TEMP	Temporary register for control mode and receive data

Each of the three assembler programs executes the following steps:

1. Initialize the DSP and the serial port
2. Load the user defined values
3. Activate and initialize the ADC
4. Acquire the specified number of data
5. Disable the ADC
6. Return to the C program

The above sequence shows that the assembler routines are used solely to acquire data. It also shows that every time the interface routine is called, the ADC is enabled for the data acquisition process and disabled before the routine returns to the C program. Therefore neither of the self-test modes nor the power-down instruction, which are available in the list of TLV1544 control words, are used in these routines.

All programs are differentiated by the use of the on-chip timer as shown in Table 10.

- Program 1 uses the timer as an interrupt source and is called C1544**TIN**.ASM.
- Program 2 uses the timer as a clock generator and is called C1544**CLK**.ASM. The end-of-conversion is detected by using the EOC output of the ADC.
- Program 3 is similar to Program 2, but uses a delay loop, which waits for the ADC maximum conversion time to pass. This wait-loop executes a specified

number of no-operation-instructions, NOPs, to generate the necessary delay, hence the program name is C1544**NOP**.ASM.

Table 10. Timer Application and Corresponding Filenames

TIMER USED AS		
Interrupt source to detect EOC	Clock source, EOC used from ADC	Clock source, EOC indicated after wait-loop
C1544**TIN**.asm (Program–1)	C1544**CLK**.asm (Program–2)	C1544**NOP**.asm (Program–3)
C1544**T**.c	C1544**C**.c	C1544**N**.c

The following sections explain the main assembler program in detail. For a review of the individual file listings, refer to Appendix A for Program-1, Appendix B for Program-2 and Appendix C for Program-3.

5.1 Program-1 *(Filename: C1544TIN.asm ⇒ Timer as Interrupt-Source)*

Program 1 supports the glueless DSP-to-ADC interface in Figure 20. The serial port is programmed for burst-mode operation. The interface clock, CLKX (5 MHz), and the frame-sync signal, FS, are generated on chip. The on-chip timer is programmed to generate an interrupt, TINT, after the maximum conversion time of the ADC (10 μs) has elapsed.

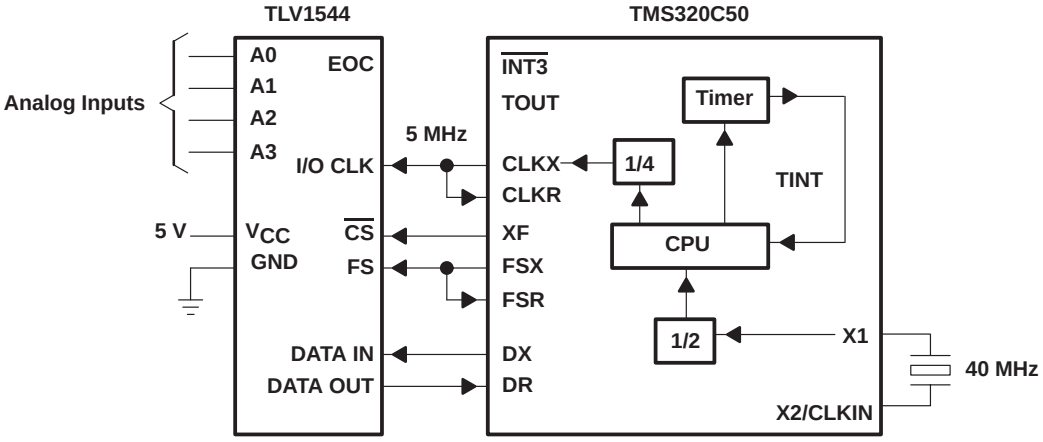


Figure 20. Timer Interrupt, TINT, Indicates EOC in a Glueless Interface

The timing diagram in Figure 21 divides a data transfer sequence into four steps:

1. The DSP activates the ADC by taking the $\overline{CS}$ low. Then the control word to initialize the ADC is loaded into the DSP serial port.
2. The actual data transfer happens. While the DSP sends control data to configure the ADC operation mode, the ADC transmits conversion results to the serial port receiver.
3. On the 16th clock cycle of I/O CLK (CLKX) after FS has gone low, a receive interrupt, RINT, is generated automatically. The following interrupt service routine, RINT-ISR, disables the ADC and starts the on-chip timer. Then it stores the received data into memory and returns from interrupt to idle mode.
4. The CPU continues idling during the entire conversion time of the ADC.

After the timer period has elapsed, a timer interrupt, TINT, is generated. The following interrupt service routine, TINT-ISR, stops the timer and loads the latest ADC control word into the serial port to prepare for a new data transfer.

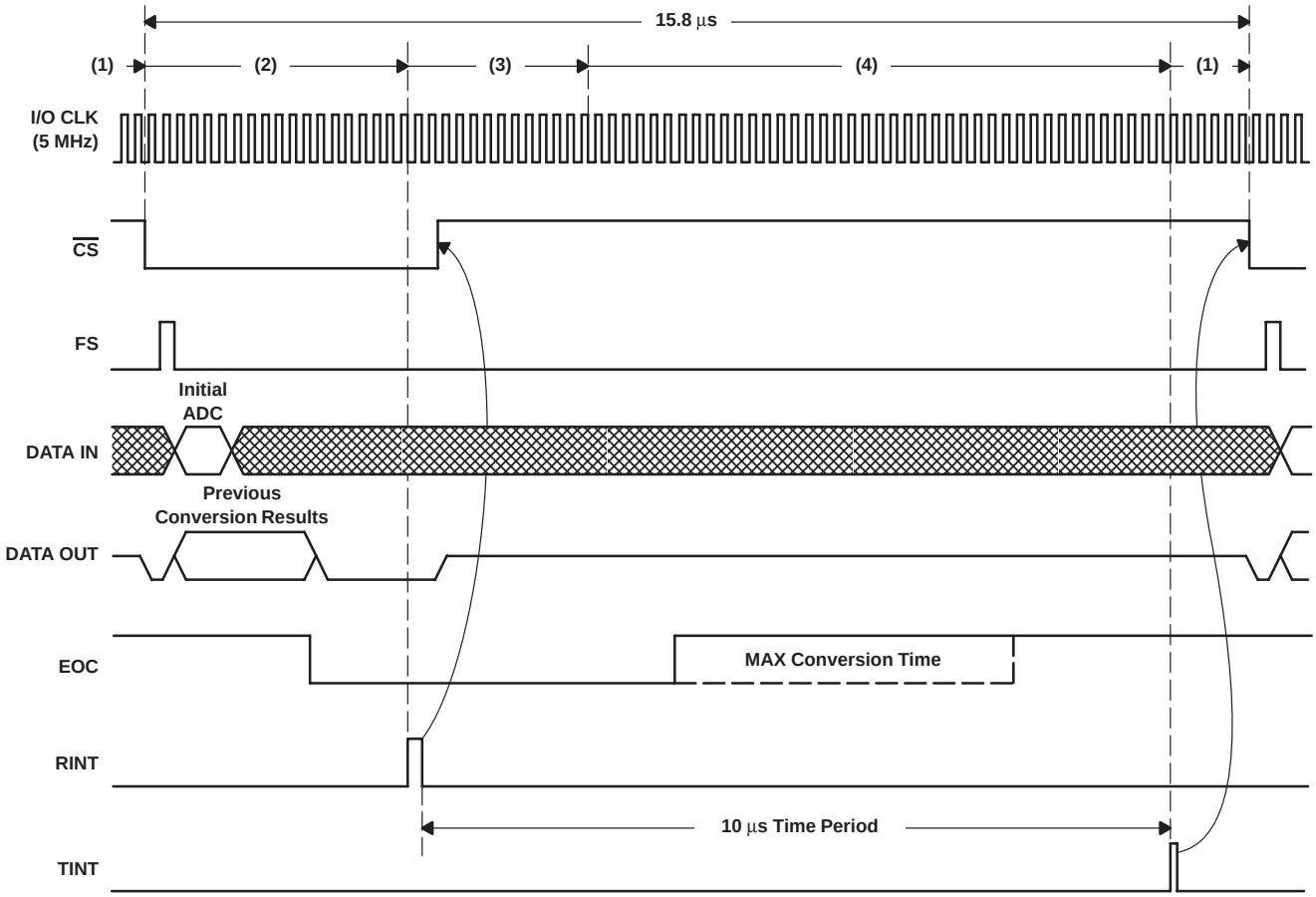


Figure 21. Data Transfer Sequence When Using TINT and ADC in Fast Conversion Mode

One data transfer takes 15.8 μs when the ADC operates in fast conversion mode, and 30.8 μs when in slow conversion mode.

5.1.1 Assembler Program-1 Description

Following the flowchart in Figure 22, this section explains the assembler program in detail.

> **NOTE:** Each task box in the flowchart appears as a header in the assembler program listing in Appendix A.

TLV1544START

The program starts with the call of the main routine _C1544TIN from the C program. At first, 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 registers, PDWSR and CWSR
- The processor mode status register PMST
- The index register INDX

During the DSP initialization all interrupts are disabled, the PMST is updated, and the wait states are set to zero. Then the serial port is configured for burst-mode operation. The FS signal and the CLKX are programmed to be generated on-chip. 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 channel number is stored into TEMP. The memory start address is loaded into ADMEM, and the number of samples to be acquired is saved into ADCOUNT. The control mode, which can be fast or slow conversion is stored into ADWORD.

The following timer configuration writes a default period of 10 μs into the period register (PRD). Depending upon whether the user selected slow or fast conversion, the content of PRD is either overwritten to 25 μs, or keeps its default value. After that, the timer start/stop bit (TSS), in the timer control register (TCR), is set to 1 to halt the timer. Both DSP internal interrupts, RINT and TINT, are now enabled.

Before the initialization of the ADC begins, the variable RINT_COUNT is set to 0. RINT_COUNT specifies the number of receive interrupts that must occur before the ADC can provide valid conversion results.

The local variable, END_BIT, which defines the end of the entire program, is set to 1. The general output port (XF), of the DSP is driven low to enable the ADC through the chip-select pin, $\overline{CS}$.

The variable ADWORD, which contains the conversion mode, is then copied into the serial port data transmit register (DXR), and sent to the ADC. Subsequently the content of TEMP, which specifies the channel number, is loaded into ADWORD.

The CPU then resides in idle mode and waits for a receive interrupt (RINT), to occur.

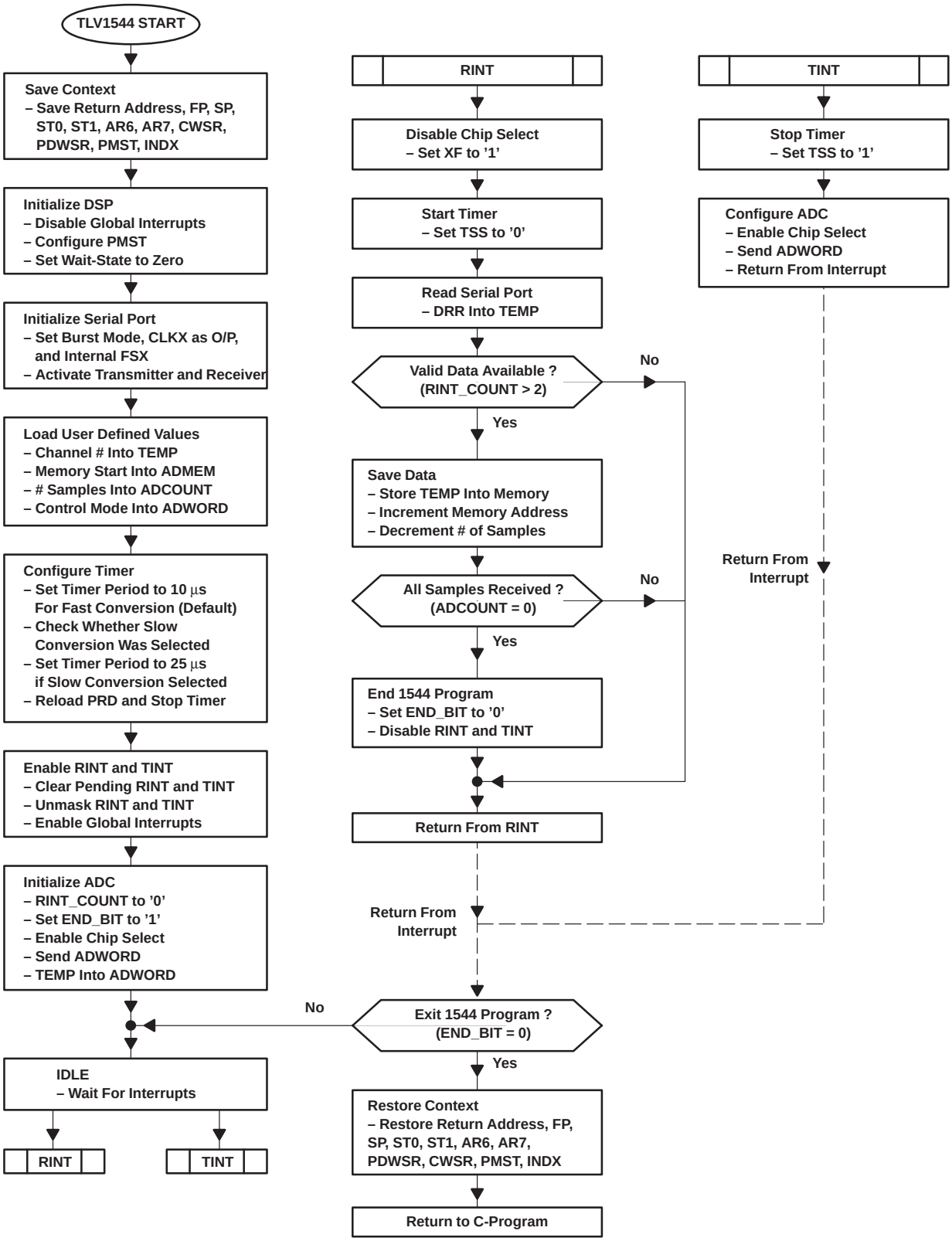


Figure 22. Flowchart of C1544IN.asm (Timer as Interrupt Source)

RINT (Receive Interrupt Routine)

With the initialization of the ADC into fast-conversion mode, the DSP starts the first data transfer of control data in a 16-bit data frame. Therefore, upon the 16th clock cycle of CLKX, RINT is generated that forces the CPU to execute the RINT service routine (RINT-ISR). At the beginning of the RINT routine, the XF output of the DSP is taken high, disabling the ADC. Then the TSS bit in the TCR is set to zero to start the timer. After that the content of the data receive register (DRR), is stored into the temporary (TEMP) register.

The following decision box investigates the receive data for validity by checking the content of RINT_COUNT for a value higher than two. Since it takes two data transfers to configure the ADC completely—one to specify the conversion mode and a second one to select the analog input channel—the conversion results from the ADC are invalid for both transfers. With each data transfer generating a receive interrupt, the number of occurring RINTs (which is stored in RINT_COUNT) must be three to indicate that valid data are available.

If no valid data are available, the CPU leaves the RINT-ISR to configure the ADC channel number via the timer interrupt service routine (TINT-ISR).

If valid data are available, the latest receive data, stored in TEMP, are saved into memory. Subsequently the memory address is increased by incrementing the content of ADMEM, and the number of samples is decreased by decrementing the content of ADCOUNT.

A second decision box checks whether all samples have been received. If all samples were received, the END_BIT is set to zero and both interrupts, RINT and TINT, are disabled. The program leaves the RINT-ISR and returns to the C program through the EXIT routine.

If all samples were not received, the CPU returns from the RINT-ISR into the idle mode and waits for a timer interrupt to occur.

TINT (Timer Interrupt Routine)

The TINT routine loads the serial port with transmit data and initiates a new data transfer. Once the timer period has elapsed, a timer interrupt is generated that causes the CPU to enter the TINT service routine. At the beginning the TSS is set to one, which stops the timer immediately. Then the DSP initiates a new data transfer by taking the XF output low to enable the ADC. The channel number stored in ADWORD is loaded into the serial port and sent to the ADC as the new control word. Afterwards the program returns from interrupt and the CPU resides in idle mode until the next RINT occurs.

Exit 1544 program ?

This decision box determines whether the entire data transfer program is terminated. As long as END-BIT is one, the CPU diverts to the idle mode 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.

5.2 Program-2 *(Filename: C1544CLK.asm ⇒ Timer as Clock-Source)*

Program 2 supports the non-glueless interface in Figure 23. In this application the ADC operates from a 2.7 V supply and is configured for slow conversion mode. The required transfer clock at I/O CLK is 2 MHz. Since the only frequency the DSP can provide at CLKX is 5 MHz (1/2 of the CPU clock), the on-chip timer is configured as a programmable clock source providing a 2-MHz clock signal at the TOUT. Although this timer configuration is required for low-volt operation of the ADC, it allows the clock rate at TOUT to be increased to 10 MHz if the ADC operates at 5.5 V. For supply voltages above 3.3 V the ADC can operate in fast conversion mode, for voltages below 3.3 V, the ADC must be operated in slow conversion mode.

Because the timer serves here as a clock generator, it is no longer available as an interrupt source to indicate the EOC state of the TLV1544. Instead, the EOC signal of the ADC is inverted and fed into the external interrupt input, $\overline{\text{INT3}}$, of the DSP. The remaining configuration is the same as for program 1: the serial port is configured for burst-mode operation, and the generation of the FS pulse happens on chip.

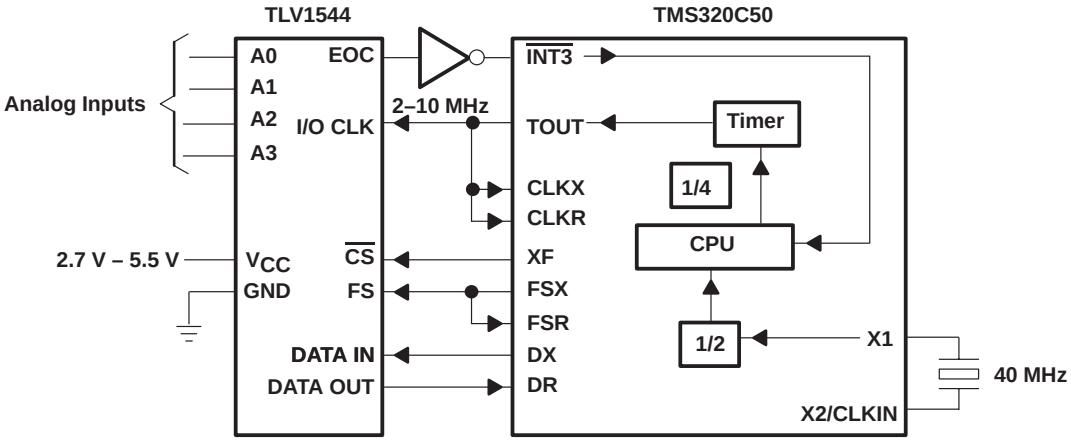


Figure 23. External Interrupt, INT3, Detects EOC Via External Inverter

The timing diagram in Figure 24 divides a data transfer sequence into four steps:

1. The DSP activates the ADC by taking the $\overline{\text{CS}}$ low. Then the control word to initialize the ADC is loaded into the DSP serial port.
2. Actual data transfer happens while the DSP sends control data to configure the ADC operation mode, the ADC transmits conversion results to the serial port receiver.
3. On the 16th clock cycle of I/O CLK after FS has gone low, RINT is generated. The following RINT-ISR disables the ADC, stores the received data into memory, and returns from interrupt to idle mode.
4. The CPU idles for the remaining ADC conversion time.

At the end of a conversion, the EOC signal of the ADC causes an external interrupt through the DSP $\overline{\text{INT3}}$ input. The following interrupt service routine, INT3-ISR, enables the ADC for a new data transfer and loads the latest control word into the serial port.

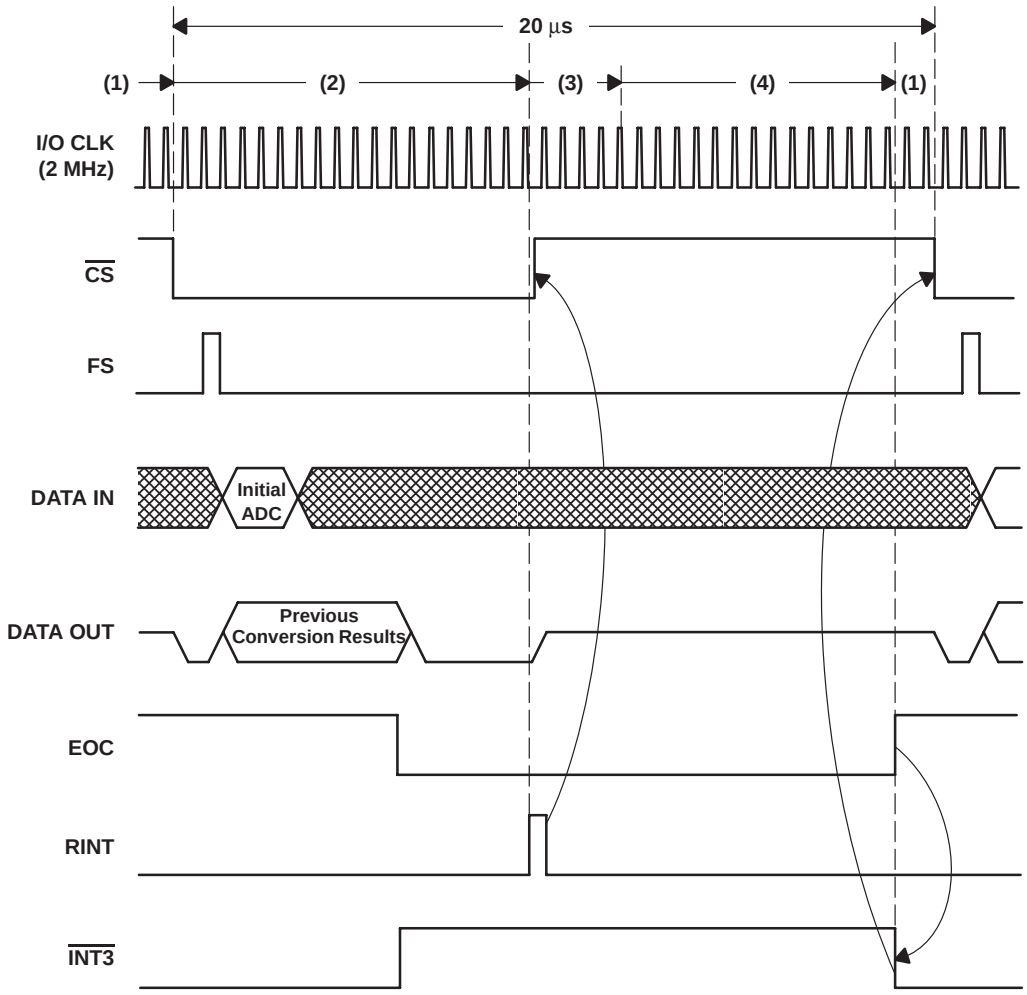


Figure 24. Data Transfer Sequence When Using $\overline{\text{INT3}}$ and ADC in Slow Conversion

A data transfer takes 20 µs when the ADC operates at 2.7 V in slow conversion mode. I/OCLK = 5 MHz. When operating at 5.5 V and in fast conversion mode, a data transfer takes 8.8 µs at I/O CLK = 5 MHz and 7.7 µs at I/O CLK = 10 MHz.

5.2.1 Assembler Program-2 Description

The flowchart in Figure 25 shows the structure of Program 2. The sequence of tasks is very similar to Program 1. The major difference lies in the use of the timer as clock generator, and the EOC detection through the external interrupt pin, $\overline{\text{INT3}}$.

TLV1544START

The program starts with the call of the main routine, C1544CLK from the C program. All previously used pointers and registers are saved and the DSP is initialized.

The following initialization of the serial port now configures CLKX as an input to receive the interface clock from the timer output (TOUT). The setup for Burst-mode operation and on-chip frame-sync generation remains the same.

One of the user-defined-values, the control mode, has been replaced by the new variable, V_{CC}. The user can now select between 3 V (2.7 V) and 5 V (5.5 V) supply voltage.

During the timer and ADC configuration, two default values are defined for low-volt operation. ADWORD is loaded with the code for slow conversion and the period register of the timer is programmed to provide an interface clock of 2 MHz at TOUT. If the user selected a V_{CC} of 5 V, then the content of both registers is overwritten. The content of ADWORD changes to fast conversion, and PRD is reloaded with a value that generates a 5-MHz clock at TOUT. If V_{CC} = 3 V is selected, the default values remain unchanged. Following the configuration, the timer is started to provide the selected interface clock at TOUT.

Interrupts RINT and INT3 are enabled and the ADC is initialized. The CPU goes into idle mode until, on the 16^{th} clock cycle after FS has gone low, a receive interrupt occurs.

RINT (Receive Interrupt Routine)

At the beginning of the RINT-ISR, the ADC is disabled by the XF output of the DSP. The receive data are loaded into TEMP and, depending on their validity, stored into memory. The memory pointer is incremented, and the sample counter is decremented.

A check is performed on whether all data have been received. If the answer is yes, the CPU exits the RINT-routine and returns to the C program. If more samples need to be acquired, the CPU returns from RINT into idle mode, where it resides until the INT3 occurs.

INT3 (External Interrupt Routine)

The INT3–ISR loads the serial port with transmit data and initiates a new data transfer. Once the EOC output of the ADC changes from low to high, an external interrupt is generated that causes the CPU to enter the INT3 service routine.

Immediately the DSP initiates a new data transfer by enabling the ADC through the XF pin. Afterwards, the channel number stored in ADWORD is sent to the ADC as the new control word. Then the CPU returns from interrupt into idle mode and waits until the next RINT occurs.

Exit 1544 program ?

As long as END-BIT is set to one, the CPU diverts to the idle mode 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 to return to the C program.

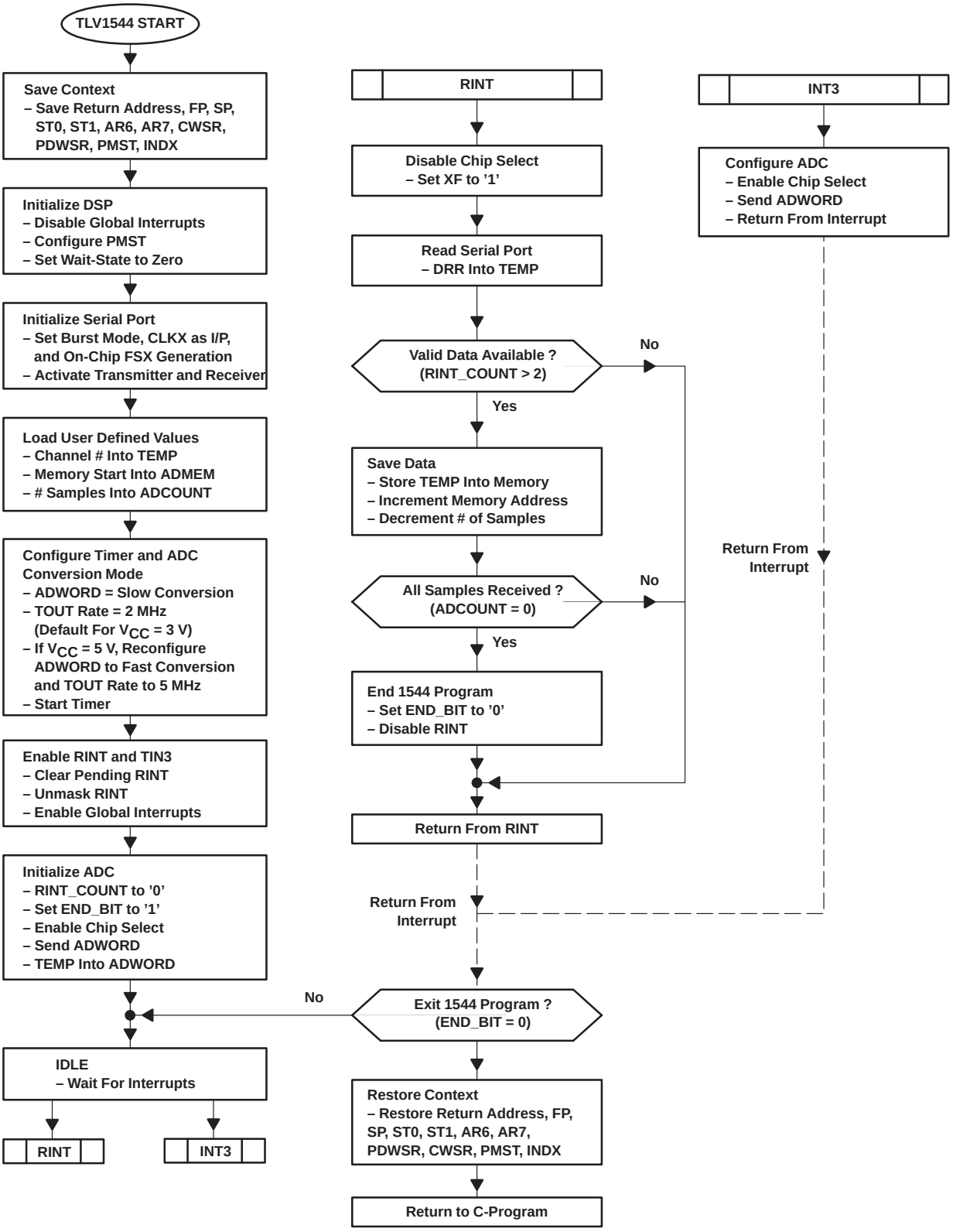


Figure 25. Flowchart of C1544CLK.asm (Timer as Clock Source)

5.3 Program-3 *(Filename: C1544NOP.asm ⇒ Timer as Clock-Source + NOP delay-loop)*

Program 3 supports a glueless interface at low supply voltage only. Figure 26 shows that this circuit is almost identical to the one in Figure 23, except this one does not need the external inverter.

The ADC operates from a 2.7 V supply and is configured for slow conversion mode only. The DSP on-chip timer provides the required interface clock of 2 MHz at TOUT. The configuration of the serial port is the same as for program 2: the serial port is configured for burst-mode operation, CLKX is programmed as inputs, and the generation of the FS happens on-chip.

To achieve a glueless interface, the external inverter, previously used for EOC indication, is removed. Instead, a delay loop of 400 no-operation (NOPs) instructions is implemented, that waits for the maximum conversion time to pass.

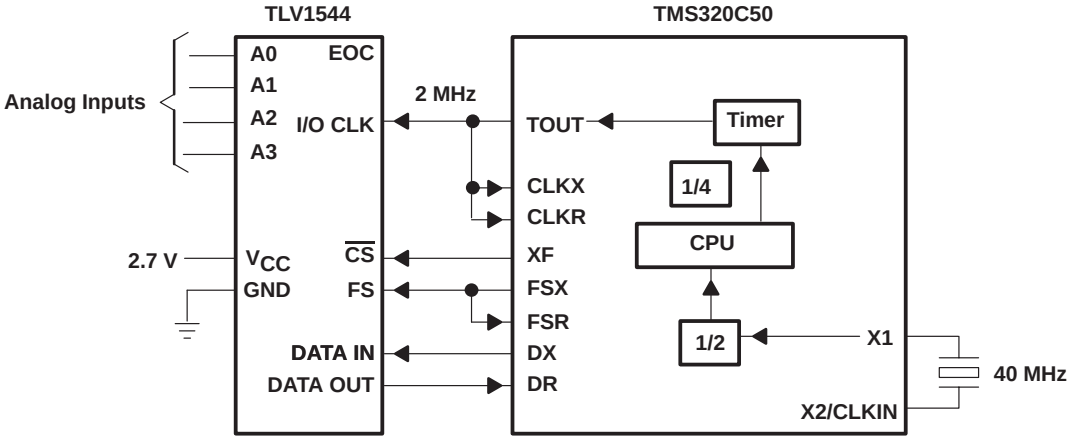


Figure 26. A Wait-loop Indicates EOC in a Glueless Interface for Low-Volt Application

The timing diagram in Figure 27 divides a data transfer sequence into two steps.

1. Step 1 is the actual data transfer, where the DSP sends control data and the ADC transmits conversion results.
2. Step 2 represents the execution phase of the entire RINT routine. Since RINT is the only interrupt used in this program, its task is not limited to finishing a transfer sequence and saving the data, but also includes waiting for the maximum conversion time to pass and initiating the next data transfer.

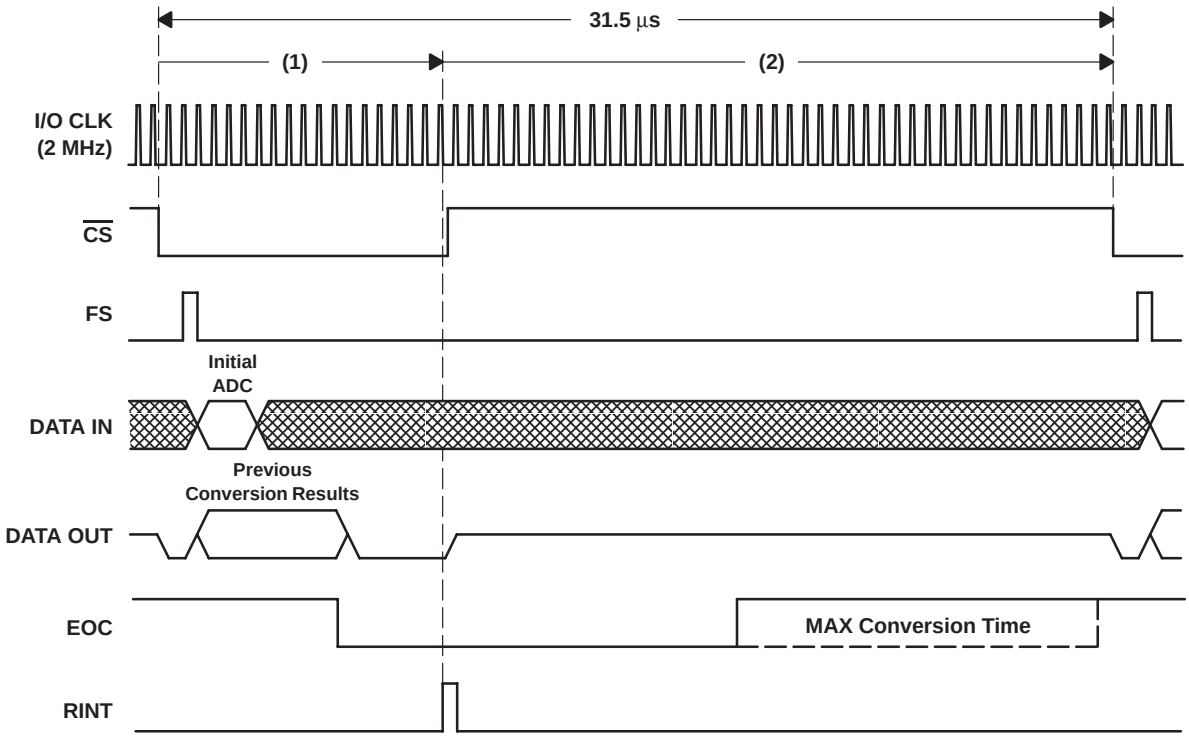


Figure 27. Data Transfer Sequence When Using a Wait-Loop for a Low-Volt Application

Since the ADC works from a 2.7-V supply, the device needs to operate in slow conversion mode. The data sheet specifies the maximum conversion time with 25 μs, thus expanding the time for one data transfer to 31.5 μs.

5.3.1 Assembler Program-2 Description

The flowchart in Figure 28 shows the structure of Program 3, which is almost identical to Program 2. The major difference is that only one interrupt routine is used to end the previous data transfer and to initiate the next one.

TLV1544START

The program starts with the call of the main routine, C1544NOP from the C program. The DSP and its serial port are initialized as in Program 1.

Since this program supports only a glueless, low-volt interface, the conversion mode and the interface clock are fixed values. User-defined variables are reduced to three: the channel number, the memory start address, and the number of samples.

The conversion mode is set to slow conversion, and the time is programmed to provide a 2-MHz interface clock through TOUT. After enabling the receive interrupt and initializing the ADC, the CPU goes into idle mode and waits for the next RINT to occur.

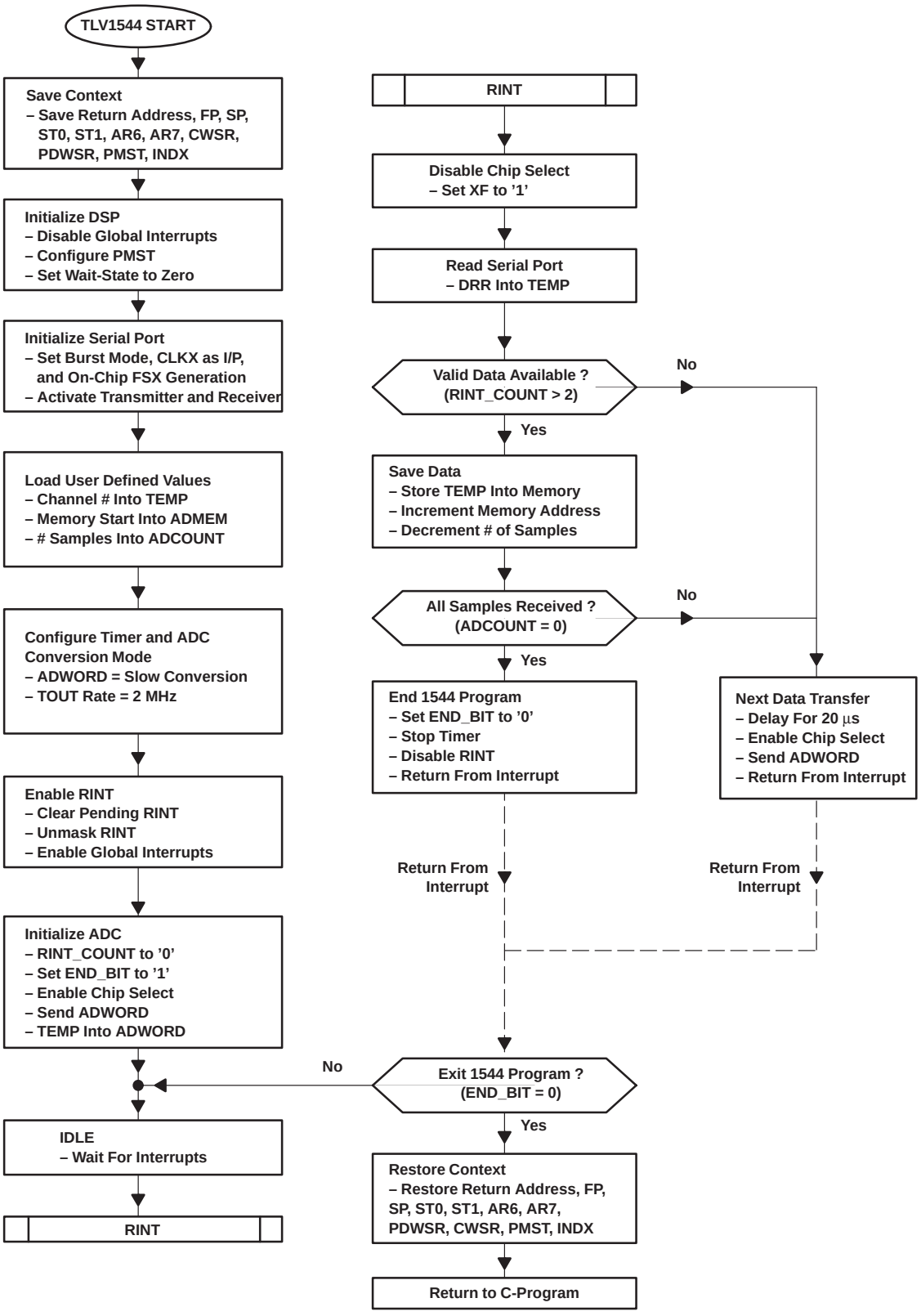


Figure 28. Flowchart of C1544NOP.asm (Timer as Clock Source + NOP delay)

RINT (Receive Interrupt Routine)

The root path of the RINT-ISR is identical to Program 1. The difference is that any further data transfers are not initiated by a second interrupt routine, but by a bypass task within the RINT routine itself.

In the case of *no valid data available* or *not all samples received*, the program diverts to a bypass task, called *Next Data Transfer*. In this task, the wait loop, consisting of 400 NOPs instructions generates a delay that is sufficient for the maximum conversion time to pass.

The next data transfer is initiated by enabling chip select and sending the channel number to the ADC. Then the CPU returns to idle mode and waits for the next RINT to occur. During the entire data acquisition process, the CPU jumps in and out of the RINT-ISR.

Once all samples are received, the CPU exits the interrupt routine and returns to the C program in the same way as in Program1.

6 Summary

This report discussed all components of a low-cost, low-power data acquisition system as shown in Figure 29. The low-dropout voltage regulator, TPS7101, is programmable through an external voltage divider to provide the required supply voltages of 5 V or 2.7 V.

The low-noise, dual operational amplifier, TLV2772, builds the signal conditioning interface to the TLV1544 ADC. One stage works as a noninverting amplifier with a closed-loop gain of two to interface small input signals between 0 and 2.5 V to the ADC. The second stage operates as an inverting amplifier with a gain of 0.25, allowing large input signals of ±10 V to be interfaced to the ADC.

The TLV1544 is a 10-bit, ADC with four analog inputs. The device interfaces easily with the TMS320C50 DSP to build a simple data acquisition system used in battery powered applications as well as in industrial control systems.

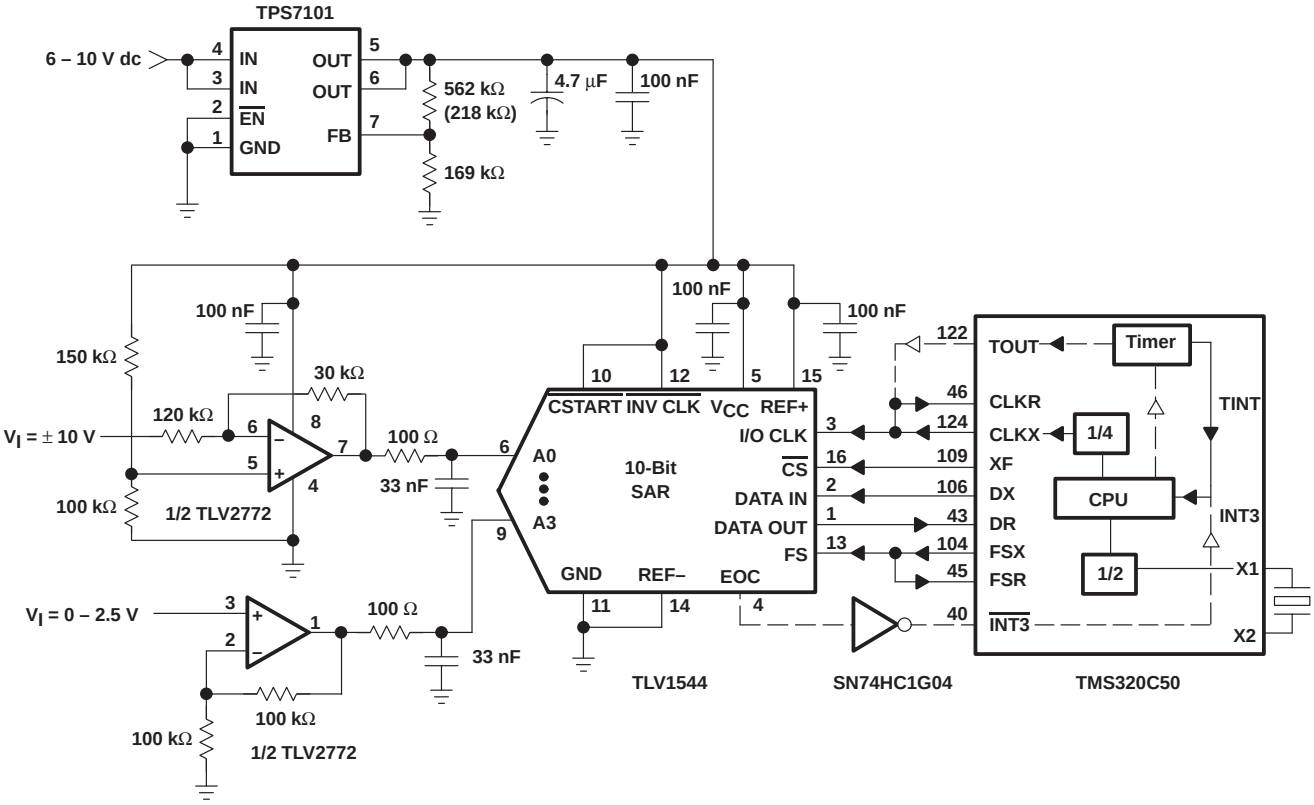


Figure 29. Interfacing the TLV1544 ADC with the TMS320C50 DSP

The report also includes three C-callable interface programs, which support the different hardware configurations of the serial interface between ADC and DSP. Depending on the supply voltage of the ADC, the interface clock can vary between 2 MHz and 10 MHz, and the conversion mode must be adjusted to either fast or slow conversion.

The ADC interfaces directly to the DSP and does not require additional control logic. However, the data throughput can be doubled if an external inverter is used between the EOC output of the ADC and one of the external interrupt inputs (INT0–INT3) of the DSP. Figure 30 shows the different periods between two high-to-low transitions of the ADC chip-select pin.

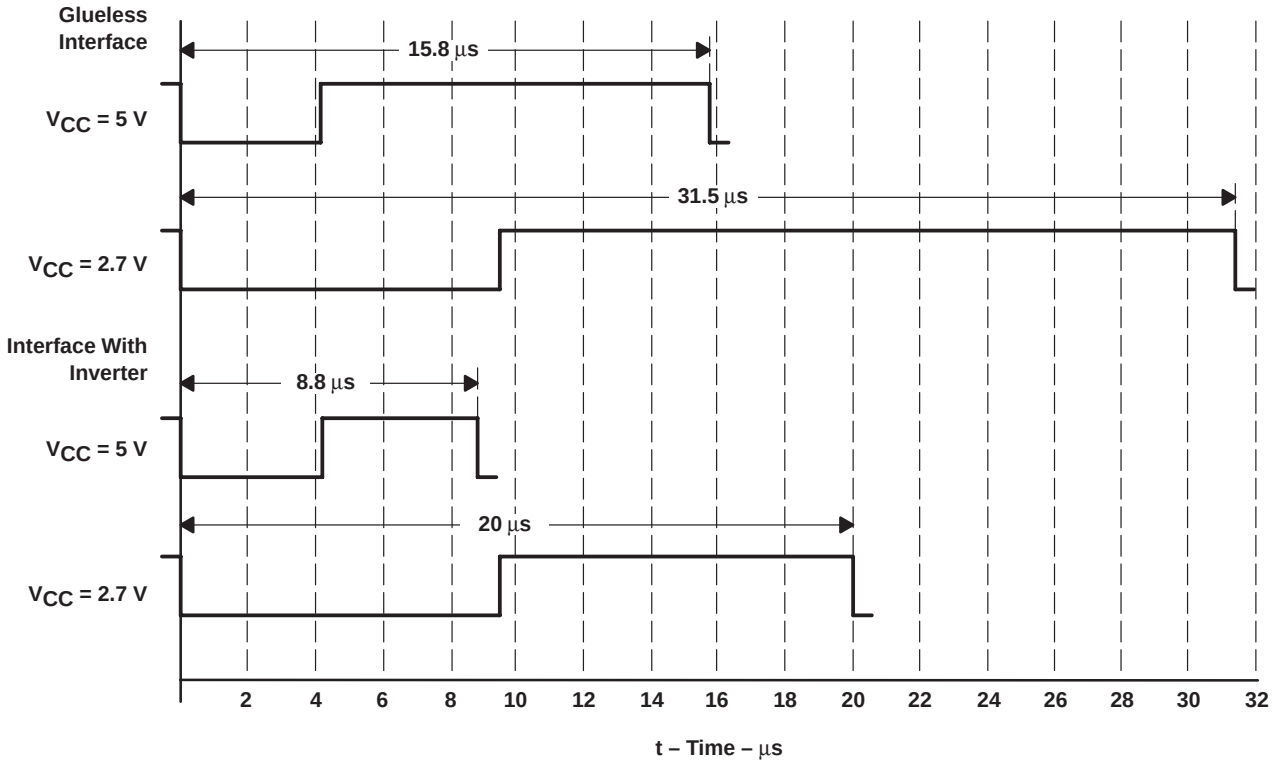


Figure 30. Time Periods Between Two Consecutive Data Transfers for Different Applications

At V_{CC} = 5 V, the ADC can operate in the fast conversion mode with an interface clock rate of I/O CLK = 5 MHz. At V_{CC} < 3.3 V, the ADC must work in the slow conversion mode. Also the interface clock is limited to a maximum of I/O CLK = 2.89 MHz.

Since the actual conversion time of an individual device can differ from the specified maximum by a factor of two, the overall data transfer period and data throughput are directly affected.

Using a glueless interface means that no direct EOC indication is provided from the ADC. Therefore, the interface program must wait for the maximum conversion time to pass before initiating a new data transfer. If the ADC operates at V_{CC} = 5 V and uses the fast conversion mode, the time between two consecutive data transfers takes 15.8 µs. This time increases to 31.5 µs when operating the ADC at V_{CC} = 2.7 V in the slow conversion mode.

To increase the data throughput, the end of the actual conversion time needs to be detected immediately. An external inverter, that connects the EOC output of the ADC with one of the interrupt inputs of the DSP, provides direct feedback on the EOC status. This way the activated interrupt routine can immediately initiate the next data transfer. The duration of a data transfer period is now reduced to 8.8 μs when operating the ADC at 5 V, and takes only 20 μs when operating at 2.7 V.

7 References

For additional information and reference, see the following related documents:

- *TMS320C5x User's Guide*, literature number SPRU056
- *TMS320C5x Data Sheet*, literature number SPRS030
- *TLV1544 Data Sheet*, literature number SLAS139
- *TLV1544 EVM Manual*, literature number SLAU014
- *TLV2432 Data Sheet*, literature number SLOS168B
- *TPS7101 Data Sheet*, literature number SLVS092F
- *SN74AHC1G04 Data Sheet*, literature number SCLS318G
- *Switched-Capacitor Analog Input Calculations Application Report*, literature number SLAA036

Appendix A TLV1544 Program Files: Timer as Interrupt Source)

A.1 Boot Routine: BOOTIN.ASM

```
*************************************************************************
* TITLE         : TLV1544 ADC boot routine (timer as interrupt source)*
* FILE          : BOOTTIN.ASM                                           *
* DESCRIPTION   : Boot routine to initialize the DSP and run the        *
*                 c–program 'main()' in the C1544T.C file.              *
* AUTHOR        : AAP Application Group, Dallas                         *
*                 CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.    *
* REFERENCE     : TMS320C5x 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                                                   *
*************************************************************************
*
* DSP INITIALIZATION
       SETC    INTM            ; disable global interrupts
       LDP     #0              ; load data page 0
       OPL     #0038h, PMST    ; configure PMST
*************************************************************************
* 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: C1544T.C

```
/*  TITLE:       TLV1544 ADC C program main routine      */
/*               (timer as interrupt source)             */
/*  File:        C1544T.C                                */
/* Description:  In this c–program file the user selects */
/*               the input channel, the conversion mode, the  */
/*               memory start address, and the number    */
/*               of samples.This c–program then calls the     */
/*               C1544TIN() interface program to execute it.  */
/* ------------------------------------------------------*/
/* Programmed Operation Mode:                            */
/*                                                       */
/* Channel:                   Value:                     */
/* Select TLV1544 channel 0      0x0000                  */
/* Select TLV1544 channel 1      0x2000                  */
/* Select TLV1544 channel 2      0x4000                  */
/* Select TLV1544 channel 3      0x6000                  */
/* (Vreg+ – Vreg–)/2             0x0B000                 */
/* Vreg–                         0x0C000                 */
/* Vreg+                         0x0D000                 */
/*                                                       */
/* ControlMode:                                          */
/* Software Power Down           0x08000                 */
/* Fast Conversion Rate          0x09000                 */
/* Slow Conversion Rate          0x0A000                 */
/*------------------------------------------------------*/
extern ControlMode, Channel, Samples, MemStart;
extern void C1544TIN(void);
main()
{
/* Fast Conversion Mode                                  */
   ControlMode = 0x09000;                                 */
/* Select channel 1                                      */
   Channel = 0x2000;

/* Take 256 samples                                      */
   Samples = 0x100;

/* Memory Start Address                                  */
   MemStart = 0x1000;
/* Call TLV1544 Interface Program                        */
   C1544TIN();
}
```

A.3 C-Callable Interface Program: C1544TIN.ASM

```
* TITLE         : TLV1544 ADC C–Callable Interface routine            *
*                 (timer as interrupt source)                         *
* FILE          : C1544TIN.ASM                                        *
* FUNCTION      : _C1544TIN                                           *
* DESCRIPTION   : Main routine to transfer data between the C50 DSP   *
*                 and TLV1544 ADC via the DSP serial interface. At    *
*                 first it initializes the DSP and the ADC, then      *
*                 transfers data from the ADC to the DSP and stores   *
*                 them within a pre-defined memory table.             *
*                 This program uses the timer as EOC indicator.       *
*                 The timer waits for the ADC conversion time to pass,*
*                 and then sends a timer interrupt (TINT) to the CPU. *
*                 The data transfer procedure is supported by the two *
*                 interrupt routines, RINT and TINT.                  *
* AUTHOR        : AAP Application Group, Dallas.                      *
*                 CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.  *
* REFERENCE     : TMS320C5x User’s Guide, TI 1997                     *
*               : Data Acquisition Circuits, TI 1998                  *
************************************************************************
    .mmregs
    .sect ”.vectors”
    .copy ”VECTIN.asm”

LOCALS    .SET   0
* Global Variables
    .global        _C1544TIN
    .global       _Samples
    .global       _MemStart
    .global       _Channel
    .global       _ControlMode
    .global       _c_int0
* Local Variables
    .def          ADWORD
    .def          ADCOUNT
    .def          ADMEM
    .def          RINT_COUNT
    .def          END_BIT
    .def          TEMP
AD_DP           .usect ”.variabl”, 0    ; AD data page for local variable
ADWORD          .usect ”.variabl”, 1    ; control word for the ADC
ADCOUNT         .usect ”.variabl”, 1    ; counter for the samples
ADMEM           .usect ”.variabl”, 1    ; memory pointer during interrupts
```

```
RINT_COUNT      .usect ".variabl", 1    ; the number of receive interrupts
END_BIT         .usect ".variabl", 1    ; end-of-data-transfer-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
_Channel        .usect ".variabl", 1    ; user defined channel
_ControlMode    .usect ".variabl", 1    ; user defined control mode
    .sect ".text"
* TLV1544 START
_C1544TIN:
* 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
    LAMM  PDWSR             ; save P/D wait-sate register
    SACL  *+
    LAMM  CWSR              ; save wait-state control register
    SACL  *+
    LAMM  PMST              ; save PMST register
    SACL  *+
    LAMM  INDX              ; save INDX register
    SACL  *+
* Initialize DSP
    SETC  INTM              ; disable global interrupts
    LDP   #0                ; load data page 0
    OPL   #0038h, PMST      ; Configure PMST
    APL   #0000h, PDWSR     ; set wait-states to zero
    APL   #00F0h, CWSR      ; set wait-states to zero
* Initialize Serial Port
    SPLK  #0038h,SPC        ; set Burst Mode, CLKX = 1/4 CLKOUT1
                            ; FSX generated by DSP
    SPLK  #00F8h,SPC        ; activate transmitter and receiver
* Load User defined Values
    LDP   #AD_DP            ; load data page  AD_DP
    LACL  _Channel          ; LOAD _Channel into TEMP
    SACL  TEMP              ;
    LACL  _MemStart         ; LOAD _MemStart into ADMEM
```

```
    SACL  ADMEM             ;
    LACL  _Samples          ; LOAD _Samples into ADCOUNT
    SACL  ADCOUNT           ;
    LACL  _ControlMode      ; LOAD _ControlMode into ARWORD
    SACL  ADWORD            ;
* Configure Timer
    LDP   #0                ; load data page 0
    SPLK  #199,PRD          ; load timer period for 10us for fast mode
    CLRC  SXM               ; clear sign–extension mode bit
    SUB   #0A000h           ; compare if is slow mode
    BCND  TIMER_INI, NEQ
    LDP   #0                ;
    SPLK  #499, PRD         ; load timer period for 25 us (slow mode)

TIMER_INI:
    LDP   #0                ; load data page 0
    SPLK  #00030h,TCR       ; reload PRD/TDDR and stop timer

* Enable RINT and TINT
    OPL   #00018h,IFR       ; clear any pending RINT and TINT
interrupt
    OPL   #00018h,IMR       ; unmask RINT and TINT
    CLRC  INTM              ; enable global interrupts
* INITIALIZE ADC
    LDP   #AD_DP            ; load data page AD_DP
    SPLK  #0h, RINT_COUNT   ; RINT_COUNT = 0
    SPLK  #1h,END_BIT       ; END_BIT = 1
    LACL  ADWORD            ; load ADWORD (Conv.Mode) into (ACL)
    CLRC  XF                ; enable Chip Select
    SAMM  DXR               ; Send ADWORD
    LACL  TEMP              ; load TEMP (Channel #) into ADWORD
    SACL  ADWORD            ;
* Idle
Idle_Mode:
    IDLE                    ; power down (IDLE), waiting for interrupts
* Exit 1544 program ?
    LDP   #AD_DP
    LACL  END_BIT
    BCND  Idle_Mode,NEQ     ; If END_BIT <> 0, go to Idle_Mode
* Restore Context
    MAR   *, AR1            ; make Stack Pointer (ar1) active
    MAR   *–                ; decrement Stack Pointer (ar1)
    LACL  *–
```

```
    SAMM  INDX              ; restore INDX register
    LACL  *–
    SAMM  PMST              ; restore PMST register
    LACL  *–
    SAMM  CWSR              ; restore wait–state control register
    LACL  *–
    SAMM  PDWSR             ; restore P/D wait–state register
    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) disables the Chip Select signal
* and stores the received data into the memory location specified by
* ADMEM. It increments the memory pointer and decrements the number of
* samples being transferred.
*************************************************************************
RINT_IRQ:
* Disable Chip Select
    SETC  XF                ; disable Chip Select (XF = 1)
* Start Timer
    LDP   #0                ; load data page 0
    SPLK  #20h,TCR          ; start the timer
* Read Serial Port
    LACC  DRR,10            ; load content of DRR in ACCh and shift 10
    LDP   #TEMP             ; load data page #TEMP
    SACH  TEMP              ; ACCH = TEMP
* Valid data available ?
    LACL  RINT_COUNT        ; ACCL = RINT_COUNT
    ADD   #1                ; increment 1
    SACL  RINT_COUNT        ; RINT_COUNT += 1
    SUB   #2                ; ACCL –= 2
    BCND  RINT_END,LEQ      ; skip saving data until RINT_COUNT > 2
* Save data
    LDP   #ADMEM            ; load data page #ADMEM
    MAR   *,AR7             ; select AR7
    LAR   AR7, ADMEM        ; AR7 = ADMEM
```

```
    LACL  TEMP              ; load TEMP into (ACL)
    SACL  *+                ; save sample in the memory
    SAR   AR7, ADMEM        ; increment memory address (ADMEM)
    LACL  ADCOUNT           ; load ADCOUNT into (ACL)
    SUB   #1                ; decrement ADCOUNT
    SACL  ADCOUNT           ; ADCOUNT –= 1
* All samples received ?
    BCND  RINT_END,NEQ      ; if ADCOUNT not 0 jump to DISABLE_CS
* End 1544 program
    LDP   #END_BIT          ; load data page #END_BIT
    SPLK  #0h,END_BIT       ; set END_BIT = 0
    LDP   #0                ; load data page 0
    SPLK  #0010h, TCR       ; stop timer
    SPLK  #0018h,IFR        ; clear any pending RINT & TINT
    APL   #0FFE7h,IMR       ; mask RINT and TINT
* Return from RINT
RINT_END:
    RETE                    ; return from RINT–ISR to ”Exit 1544 program ?”
*************************************************************************
* TINT
* The interrupt routine for the internal interrupt, TINT, is initiated
* by the timer. This routine enables the Chip Select signal and sends the
* current control word, defined in ADWORD, to the ADC.
*************************************************************************
TINT_IRQ:
* Stop Timer
    LDP   #0                ; load data page 0
    SPLK  #00010h,TCR       ; stop timer
* Configure ADC
    LDP   #ADWORD           ; load data page #ADWORD
    LACL  ADWORD            ; load ADWORD into (ACL)
    CLRC  XF                ; enable Chip Select
    SAMM  DXR               ; move ADWORD into DXR
    RETE                    ; return from TINT–ISR to ”Exit 1544 program ?”
   .copy  ”BOOTTIN.asm”
```

A.4 Vector Table: VECTIN.ASM

```
*************************************************************************
* TITLE       : TLV1544 ADC Interface Vector routine                    *
*               (timer as interrupt source)                             *
* FILE        : VECTIN.ASM                                              *
* DESCRIPTION : This file defines the interrupt vector table.           *
*               If TINT occurs: vector points to TINT_IRQ subroutine.   *
*               If RINT occurs: vector points to RINT_IRQ subroutine.   *
* AUTHOR      : AAP Application Group, Dallas                           *
*               CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.      *
* REFERENCE   : TMS320C5x User's Guide, TI 1997                         *
*************************************************************************
        .global     _main
RS      b   _c_int0           ;0x00; RESET
INT1    b   INT1              ;0x02; external user interrupt #1
INT2    b   INT2              ;0x04; external user interrupt #2
INT3    b   INT3              ;0x06; external user interrupt #3
TINT    b   TINT_IRQ          ;0x08; internal timer interrupt
RINT    b   RINT_IRQ          ;0x0A; Serial Port receive interrupt
XINT    b   XINT              ;0x0C; Serial Port transmit interrupt
TRNT    b   TRNT              ;0x0E; TDM receive interrupt
TXNT    b   TXNT              ;0x10; TDM transmit interrupt
INT4    b   INT4              ;0x12; external user interrupt #4
        .space      14*16     ;0x14–0x21; reserved area
TRAP    b   TRAP              ;0x22; trap instruction vector
NMI     b   NMI               ;0x24; non–maskable interrupt
```

Appendix B TLV1544 Program Files: Timer as Clock Source

B.1 Boot Routine: BOOTCLK.ASM

```
;***********************************************************************
* TITLE         : TLV1544 ADC boot routine (timer as clock source)    *
* FILE          : BOOTCLK.ASM                                         *
* DESCRIPTION   : Boot routine to initialize the DSP and run the      *
*                 c–program 'main()' in the C1544C.C file.            *
* AUTHOR        : AAP Application Group, Dallas                       *
*                 CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.  *
* REFERENCE     : TMS320C5x 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                                                 *
;***********************************************************************
* DSP INITIALIZATION
       SETC    INTM            ; disable global interrupts
       LDP     #0             ; load data page 0
       OPL     #0038h, PMST    ; configure PMST
;***********************************************************************
* 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
```

B.2 C-Program: C1544C.C

```
/* TITLE:       TLV1544 ADC C program main routine            */
/*              (timer as clock source)                       */
/* File:        C1544C.C                                      */
/* Description: In this c–program file the user selects the   */
/*              input channel, the conversion mode, the memory*/
/*              start address, and the number of samples.     */
/*              This c–program then calls the C1544CLK()      */
/*              interface program to execute it.              */
/*--------------------------------------------------------------*/
/* Programmed Operation Mode:                                 */
/*                                                            */
/* Channel:                          Value:                   */
/* Select TLV1544 channel 0             0x0000                */
/* Select TLV1544 channel 1             0x2000                */
/* Select TLV1544 channel 2             0x4000                */
/* Select TLV1544 channel 3             0x6000                */
/* (Vreg+ – Vreg–)/2                    0x0B000               */
/* Vreg–                                0x0C000               */
/* Vreg+                                0x0D000               */
/*                                                            */
/* ControlMode:                                               */
/* Software Power Down                  0x08000               */
/* Fast Conversion Rate                 0x09000               */
/* Slow Conversion Rate                 0x0A000               */
/*--------------------------------------------------------------*/

extern Vcc, Channel, Samples, MemStart;
extern void C1544CLK(void);
main()
{
/* Select Vcc, Vcc = 3 or Vcc = 5                             */
   Vcc = 3;
/* Select channel 1                                           */
   Channel = 0x2000;

/* Take 256 samples                                           */
   Samples = 0x100;

/* Memory Start Address                                       */
   MemStart = 0x1000;
/* Call TLV1544 Interface Program                             */
   C1544CLK();
}
```

B.3 C-Callable Interface Program: C1544CLK.ASM

```
* TITLE:         TLV1544 ADC C–Callable Interface routine             *
*                (timer as clock source)                              *
* FILE:           C1544CLK.ASM                                       *
* FUNCTION:       _C1544CLK                                          *
* DESCRIPTION:    Main routine to transfer data between the C50      *
*                 DSP and TLV1544 ADC via the DSP serial             *
*                 interface.At first it initializes the DSP and      *
*                 the ADC, then transfers data from the ADC to       *
*                 the DSP and stores them within a pre-defined       *
*                 memdory table. This program uses the EOC signal    *
*                 of the TLV1544 ADC. The data transfer procedure    *
*                 is supported by the two interrupt routines, RINT   *
*                 and INT3.The DSP timer is configured as I/O        *
*                 clock source to provide two different clock        *
*                 rates of 2 MHz at 3V Vcc,and 5 MHz at 5V           *
* AUTHOR:         AAP Application Group, Dallas.                     *
*                 CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED  *
* REFERENCE:      TMS320C5x User’s Guide, TI 1997                    *
*                 Data Acquisition Circuits, TI 1998                 *
*********************************************************************
    .mmregs
    .sect ”.vectors”
    .copy ”VECCLK.asm”

LOCALS    .SET   0
* Global Variables
    .global        _C1544CLK
    .global       _Samples
    .global       _MemStart
    .global       _Channel
    .global       _Vcc
    .global       _c_int0
* Local Variables
    .def          ADWORD
    .def          ADCOUNT
    .def          ADMEM
    .def          RINT_COUNT
    .def          END_BIT
    .def          TEMP
AD_DP           .usect ”.variabl”, 0    ; AD data page for local variable
ADWORD          .usect ”.variabl”, 1    ; control word for the ADC
ADCOUNT         .usect ”.variabl”, 1    ; counter for the samples
```

```
ADMEM           .usect ".variabl", 1    ; memory pointer during interrupts
RINT_COUNT      .usect ".variabl", 1    ; the number of receive interrupts
END_BIT         .usect ".variabl", 1    ; end-of-data-transfer-program flag
TEMP            .usect ".variabl", 1    ; temporary memory for DRR data
_Samples        .usect ".variabl", 1    ; user defined number of sample
_MemStart       .usect ".variabl", 1    ; user defined memory pointer
_Channel        .usect ".variabl", 1    ; user defined channel
_Vcc            .usect ".variabl", 1    ; user defined Vcc
    .sect ".text"
* TLV1544 START
_C1544CLK:
* 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
    LAMM  PDWSR             ; save P/D wait-sate register
    SACL  *+
    LAMM  CWSR              ; save wait-state control register
    SACL  *+
    LAMM  PMST              ; save PMST register
    SACL  *+
    LAMM  INDX              ; save INDX register
    SACL  *+
* Initialize DSP
    SETC  INTM              ; disable global interrupts
    LDP   #0                ; load data page 0
    OPL   #0038h, PMST      ; Configure PMST
    APL   #0000h, PDWSR     ; set wait-states to zero
    APL   #00F0h, CWSR      ; set wait-states to zero
* Initialize Serial Port
    SPLK  #0028h,SPC        ; set Burst Mode, CLKX set as input pin
                            ; FSX generated by DSP
    SPLK  #00E8h,SPC        ; activate transmitter and receiver
* Load User defined Values
    LDP   #AD_DP            ; point to AD Data Page
    LACL  _Channel          ; LOAD _Channel into (ACL)
    SACL  TEMP              ; TEMP = (ACL)
```

```
    LACL  _MemStart          ; LOAD _MemStart into (ACL)
    SACL  ADMEM              ; ADMEM = (ACL)
    LACL  _Samples           ; LOAD _Samples into (ACL)
    SACL  ADCOUNT            ; ADCOUNT = (ACC)
    LACL  _Vcc               ; LOAD _Vcc into (ACL)
* Configure Timer and ADC conversion mode
    SPLK  #0A000h, ADWORD    ; set ADC for slow conversion
    LDP   #0
    SPLK  #9, PRD            ; set Timer for 2MHz
    CLRC  SXM                ; clear sign extension mode bit
    SUB   #3                 ; check if _Vcc = 3V
    BCND  Timer_Start, EQ
    LDP   #0                 ; load data page 0
    SPLK  #3, PRD            ; set timer for 5MHz
    LDP   #AD_DP             ; load data page #AD_DP
    SPLK  #09000h, ADWORD    ; set ADC for fast conversion mode
Timer_Start:
    LDP   #0
    SPLK  #00020h,TCR        ; reload PRD/TDDR, start timer
* Enable RINT and INT3
    OPL   #0014h,IFR         ; clear RINT and INT3
    OPL   #0014h,IMR         ; unmask RINT and INT3
    CLRC  INTM               ; enable global interrupts
* Iinitialize ADC
    LDP   #AD_DP
    SPLK  #0h, RINT_COUNT    ; RINT_COUNT = 0
    SPLK  #1h,END_BIT        ; END_BIT = 1
    LACL  ADWORD             ; Conversion Mode into ACC
    CLRC  XF                 ; enable Chip Select
    SAMM  DXR                ; Send Conversion Mode to ADC
    LACL  TEMP               ; Load Channel number into ACC
    SACL  ADWORD             ; Save ACC into ADWORD
* Idle
Idle_Mode:
    IDLE                     ; power down (IDLE), waiting for
interrupts
* Exit 1544 program ?
    LDP   #AD_DP
    LACL  END_BIT            ; Load END_BIT into (ACL)
    BCND  Idle_Mode,NEQ      ; If END_BIT <> 0, go to Idle_Mode
* Restore Context
    MAR   *, AR1             ; make Stack Pointer (ar1) active
    MAR   *–                 ; decrement Stack Pointer (ar1)
```

```
    LACL  *–
    SAMM  INDX              ; restore INDX register
    LACL  *–
    SAMM  PMST              ; restore PMST register
    LACL  *–
    SAMM  CWSR              ; restore wait–state control register
    LACL  *–
    SAMM  PDWSR             ; restore P/D wait–state register
    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) disables the Chip Select signal
* and stores the received data into the memory location specified by
* ADMEM. It increments the memory pointer and decrements the number of
* samples being transferred.
*************************************************************************
*
RINT_IRQ:
* Disable Chip Select
    SETC  XF                ; disable Chip Select (XF = 1)
* Read Serial Port
    LDP   #0                ; load data page 0
    LACC  DRR,10            ; load content of DRR in ACCh; shift 10
    LDP   #TEMP             ; load data page #TEMP
    SACH  TEMP              ; ACCH = TEMP
* Valid data available ?
    LACL  RINT_COUNT        ; ACCL = RINT_COUNT
    ADD   #1                ; increment 1
    SACL  RINT_COUNT        ; RINT_COUNT += 1
    SUB   #2                ; ACCL –= 2
    BCND  RINT_END,LEQ      ; skip saving data until RINT_COUNT > 2
* Save data
    LDP   #ADMEM
    MAR   *,AR7             ; select AR7
    LAR   AR7, ADMEM        ; AR7 = ADMEM
```

```
    LACL  TEMP              ; load sample into ACC
    SACL  *+                ; save sample in the memory
    SAR   AR7, ADMEM        ; increment memory address (ADMEM)
    LACL  ADCOUNT           ; ACCL = ADCOUNT
    SUB   #1                ; decrement ADCOUNT
    SACL  ADCOUNT           ; ADCOUNT –= 1
* All samples received ?
    BCND  RINT_END,NEQ     ; if ADCOUNT not 0 jump to RINT_END
* End 1544 program
    LDP   #END_BIT
    SPLK  #0h,END_BIT       ; set END_BIT = 0
    LDP   #0                ; load data page 0
    SPLK  #0010h, TCR       ; stop timer
    SPLK  #0014h, IFR       ; clear pending RINT and INT3
    APL   #0FFEBh,IMR       ; mask RINT and INT3
* Return from RINT
RINT_END:
    RETE                    ; return from RINT–ISR to ”Exit 1544 program ?”
*************************************************************************
* INT3
* The interrupt routine for the external interrupt, INT3, is initiated
* by the End–Of–Conversion signal sent from the ADC. This routine enables
* the Chip Select signal and sends the current control word, defined
* in ADWORD, to the ADC.
*************************************************************************
INT3_IRQ:
* Configure ADC
    LDP   #ADWORD
    LACL  ADWORD            ; load ADWORD into ACC
    CLRC  XF                ; enable Chip Select
    SAMM  DXR               ; move ADWORD into DXR
    RETE                    ; return from INT3–ISR to ”Exit 1544 program ?”
   .copy  ”BOOTCLK.asm”
```

B.4 Vector Table: VECCLK.ASM

```
*************************************************************************
* TITLE       : TLV1544 ADC Interface Vector routine                    *
*               (timer as clock source)                                 *
* FILE        : VECCLK.ASM                                              *
* DESCRIPTION : This file defines the interrupt vector table.           *
*           If INT3 occurs: vector points to INT3_IRQ subroutine.       *
*           If RINT occurs: vector points to RINT_IRQ subroutine.       *
* AUTHOR   : AAP Application Group, Dallas                              *
*        CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.             *
* REFERENCE   : TMS320C5x User's Guide, TI 1997                         *
*************************************************************************
        .global     _main
RS      b   _c_int0           ;0x00; RESET
INT1    b   INT1              ;0x02; external user interrupt #1
INT2    b   INT2              ;0x04; external user interrupt #2
INT3    b   INT3_IRQ          ;0x06; external user interrupt #3
TINT    b   TINT              ;0x08; internal timer interrupt
RINT    b   RINT_IRQ          ;0x0A; Serial Port receive interrupt
XINT    b   XINT              ;0x0C; Serial Port transmit interrupt
TRNT    b   TRNT              ;0x0E; TDM receive interrupt
TXNT    b   TXNT              ;0x10; TDM transmit interrupt
INT4    b   INT4              ;0x12; external user interrupt #4
        .space      14*16     ;0x14–0x21; reserved area
TRAP    b   TRAP              ;0x22; trap instruction vector
NMI     b   NMI               ;0x24; non–maskable interrupt
```

Appendix C TLV1544 Program Files: NOPs for Wait-Loop

C.1 Boot Routine: BOOTNOP.ASM

```
;***********************************************************************
* TITLE         : TLV1544 ADC boot routine (NOPs for wait–loop)       *
* FILE          : BOOTNOP.ASM                                         *
* DESCRIPTION   : Boot routine to initialize the DSP and run the      *
*                 c–program 'main()' in the C1544N.C file.            *
* AUTHOR        : AAP Application Group, Dallas                       *
*                 CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.  *
* REFERENCE     : TMS320C5x 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                                                 *
;***********************************************************************
*
* DSP INITIALIZATION
       SETC    INTM            ; disable global interrupts
       LDP     #0                ; load data page 0
       OPL     #0038h, PMST    ; configure PMST
;***********************************************************************
* 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
```

C.2 C-Program: C1544N.C

```
/* TITLE:         TLV1544 ADC C program main routine            */
/*                (NOPs for wait–loop)                          */
/* File:          C1544N.C                                      */
/* Description:   In this c–program file the user selects the   */
/*                input channel, the memory start address and   */
/*                the number of samples.                        */
/*                This c–program then calls the C1544NOP()      */
/*                interface program to execute it.              */
/*--------------------------------------------------------------*/
/* Programmed Operation Mode:                                   */
/*                                                              */
/* Channel:                        Value:                      */
/* Select TLV1544 channel 0              0x0000                 */
/* Select TLV1544 channel 1              0x2000                 */
/* Select TLV1544 channel 2              0x4000                 */
/* Select TLV1544 channel 3              0x6000                 */
/* (Vreg+ – Vreg–)/2                    0x0B000                */
/* Vreg–                                  0x0C000               */
/* Vreg+                                  0x0D000               */
/*                                                              */
/* ControlMode:                                                 */
/* Software Power Down                   0x08000                */
/* Fast Conversion Rate                  0x09000                */
/* Slow Conversion Rate                  0x0A000                */
/*--------------------------------------------------------------*/

extern Channel, Samples, MemStart;
extern void C1544NOP(void);
main()
{
/* Select channel 1                                             */
   Channel = 0x2000;

/* Take 256 samples                                             */
   Samples = 0x100;

/* Memory Start Address                                         */
   MemStart = 0x1000;
/* Call TLV1544 Interface Program                               */
   C1544NOP();
}
```

C.3 C-Callable Interface Program: C1544NOP.ASM

```
* TITLE         : TLV1544 ADC C–Callable Interface routine          *
*                 (timer as clock source)                           *
* FILE          : C1544NOP.ASM                                      *
* FUNCTION      : _C1544NOP                                         *
* DESCRIPTION   : Main routine to transfer data between the C50     *
*                 DSP andTLV1544 ADC via the DSP serial interface.  *
*                 At first it initializes the DSP and the ADC, then *
*                 transfers data from the ADC to the DSP and stores *
*                 them within a pre-defined memory table.           *
*                 This program uses a delay loop as EOC indicator.  *
*                 The delay loop waits for the ADC conversion       *
*                 time to pass,before staring a new data transfer.  *
*                 The DSP timer is configured to provide a 2 MHz    *
*                 I/O clock The data transfer procedure is          *
*                 supported by the receive interrupt routine only.  *
* AUTHOR        : AAP Application Group, Dallas.                    *
*                 CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.*
* REFERENCE     : TMS320C5x User’s Guide, TI 1997                   *
*               : Data Acquisition Circuits, TI 1998                *
*********************************************************************

    .mmregs
    .sect ”.vectors”
    .copy ”VECNOP.asm”

LOCALS    .SET   0
* Global Variables
    .global        _C1544NOP
    .global       _Samples
    .global       _MemStart
    .global       _Channel
    .global       _c_int0
* Local Variables
    .def          ADWORD
    .def          ADCOUNT
    .def          ADMEM
    .def          RINT_COUNT
    .def          END_BIT
    .def          TEMP
AD_DP           .usect ”.variabl”, 0    ; AD data page for local
variable
ADWORD          .usect ”.variabl”, 1    ; control word for the ADC
ADCOUNT         .usect ”.variabl”, 1    ; counter for the samples
```

```
ADMEM           .usect ".variabl", 1    ; memory pointer during
interrupts
RINT_COUNT      .usect ".variabl", 1    ; the number of receive
interrupts
END_BIT         .usect ".variabl", 1    ; end-of-data-transfer-program
flag
TEMP            .usect ".variabl", 1    ; temporary memory for DRR data
_Samples        .usect ".variabl", 1    ; user defined number of sample
_MemStart       .usect ".variabl", 1    ; user defined memory pointer
_Channel        .usect ".variabl", 1    ; user defined channel
    .sect ".text"
* TLV1544 START
_C1544NOP:
* 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
    LAMM  PDWSR             ; save P/D wait-sate register
    SACL  *+
    LAMM  CWSR              ; save wait-state control register
    SACL  *+
    LAMM  PMST              ; save PMST register
    SACL  *+
    LAMM  INDX              ; save INDX register
    SACL  *+
* Initialize DSP
    SETC  INTM              ; disable global interrupts
    LDP   #0                ; load data page 0
    OPL   #0038h, PMST      ; Configure PMST
    APL   #0000h, PDWSR     ; set wait-states to zero
    APL   #00F0h, CWSR      ; set wait-states to zero
* Initialize Serial Port
    SPLK  #0028h,SPC        ; set Burst Mode, CLKX set as input pin
                            ; FSX generated by DSP
    SPLK  #00E8h,SPC        ; activate transmitter and receiver
* Load User defined Values
    LDP   #AD_DP            ; point to AD Data Page
    LACL  _Channel          ; LOAD _Channel into TEMP
```

```
    SACL  TEMP
    LACL  _MemStart          ; LOAD _MemStart into ADMEM
    SACL  ADMEM
    LACL  _Samples           ; LOAD _Samples into ADCOUNT
    SACL  ADCOUNT
* Configure Timer and ADC conversion mode
    SPLK  #0A000h, ADWORD    ; set ADC for slow conversion
    LDP   #0
    SPLK  #9, PRD            ; set Timer for 2MHz
    CLRC  SXM                ; clear sign extension mode bit
    SPLK  #00020h,TCR        ; reload PRD/TDDR, start timer
* Enable RINT and INT3
    OPL   #0010h,IFR         ; clear RINT
    OPL   #0010h,IMR         ; unmask RINT
    CLRC  INTM               ; enable global interrupts
* Iinitialize ADC
    LDP   #AD_DP
    SPLK  #0h, RINT_COUNT    ; RINT_COUNT = 0
    SPLK  #1h,END_BIT        ; END_BIT = 1
    LACL  ADWORD             ; Conversion Mode into ACC
    CLRC  XF                 ; enable Chip Select
    SAMM  DXR                ; Send Conversion Mode to ADC
    LACL  TEMP               ; Load Channel number into ACC
    SACL  ADWORD             ; Save ACC into ADWORD
* Idle
Idle_Mode:
    IDLE                     ; power down (IDLE), waiting for
interrupts
* Exit 1544 program ?
    LDP   #AD_DP
    LACL  END_BIT            ; Load END_BIT into (ACL)
    BCND  Idle_Mode,NEQ      ; If END_BIT <> 0, go to Idle_Mode
* Restore Context
    MAR   *, AR1             ; make Stack Pointer (ar1) active
    MAR   *–                 ; decrement Stack Pointer (ar1)
    LACL  *–
    SAMM  INDX               ; restore INDX register
    LACL  *–
    SAMM  PMST               ; restore PMST register
    LACL  *–
    SAMM  CWSR               ; restore wait–state control register
    LACL  *–
    SAMM  PDWSR              ; restore P/D wait–state register
```

```
    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) disables the Chip Select signal
* and stores the received data into the memory location specified by
* ADMEM. It increments the memory pointer and decrements the number of
* samples being transferred and initiates the next data transfer.
*************************************************************************
RINT_IRQ:
* Disable Chip Select
    SETC  XF                ; disable Chip Select (XF = 1)
* Read Serial Port
    LDP   #0                ; load data page 0
    LACC  DRR,10            ; load content of DRR in ACCh; shift 10
    LDP   #TEMP             ; load data page #TEMP
    SACH  TEMP              ; ACCH = TEMP
* Valid data available ?
    LACL  RINT_COUNT        ; ACCL = RINT_COUNT
    ADD   #1                ; increment 1
    SACL  RINT_COUNT        ; RINT_COUNT += 1
    SUB   #2                ; ACCL –= 2
    BCND  NEXT_DATA,LEQ     ; skip saving data until RINT_COUNT > 2
* Save data
    LDP   #ADMEM
    MAR   *,AR7             ; select AR7
    LAR   AR7, ADMEM        ; AR7 = ADMEM
    LACL  TEMP              ; load sample into ACC
    SACL  *+                ; save sample in the memory
    SAR   AR7, ADMEM        ; increment memory address (ADMEM)
    LACL  ADCOUNT           ; ACCL = ADCOUNT
    SUB   #1                ; decrement ADCOUNT
    SACL  ADCOUNT           ; ADCOUNT –= 1
* All samples received ?
    BCND  NEXT_DATA,NEQ     ; if ADCOUNT not 0 jump to NEXT_DATA
```

```
* End 1544 program
    LDP   #END_BIT
    SPLK  #0h,END_BIT        ; set END_BIT = 0
    LDP   #0                 ; load data page 0
    SPLK  #0010h, TCR        ; stop timer
    SPLK  #0010h, IFR        ; clear pending RINT
    APL   #0FFEFh,IMR        ; mask RINT
    RETE                     ; return from RINT–ISR to ”Exit 1544 program ?”
* Next Data Transfer
NEXT_DATA:
    RPT   #18Fh              ; 400 NOPs x 50 ns = 20 us delay
    NOP
    LDP   #ADWORD
    LACL  ADWORD             ; load ADWORD into Accu
    CLRC  XF                 ; enable Chip Select
    SAMM  DXR                ; send ADWORD
    RETE                     ; return from RINT–ISR for next data
   .copy  ”BOOTNOP.asm”
```

C.4 Vector Table: VECNOP.ASM

```
*************************************************************************
* TITLE       : TLV1544 ADC Interface Vector routine                    *
*               (NOPs for wait–loop)                                    *
* FILE        : VECNOP.ASM                                              *
* DESCRIPTION : This file defines the interrupt vector table.           *
*               If RINT occurs: vector points to RINT_IRQ subroutine.   *
* AUTHOR      : AAP Application Group, Dallas                           *
*               CREATED 1998(C) BY TEXAS INSTRUMENTS INCORPORATED.      *
* REFERENCE   : TMS320C5x User's Guide, TI 1997                         *
*************************************************************************
        .global     _main
RS      b   _c_int0           ;0x00; RESET
INT1    b   INT1              ;0x02; external user interrupt #1
INT2    b   INT2              ;0x04; external user interrupt #2
INT3    b   INT3              ;0x06; external user interrupt #3
TINT    b   TINT              ;0x08; internal timer interrupt
RINT    b   RINT_IRQ          ;0x0A; Serial Port receive interrupt
XINT    b   XINT              ;0x0C; Serial Port transmit interrupt
TRNT    b   TRNT              ;0x0E; TDM receive interrupt
TXNT    b   TXNT              ;0x10; TDM transmit interrupt
INT4    b   INT4              ;0x12; external user interrupt #4
        .space      14*16     ;0x14–0x21; reserved area
TRAP    b   TRAP              ;0x22; trap instruction vector
NMI     b   NMI               ;0x24; non–maskable interrupt
```