10 HARDWARE INTERFACING AND EXAMPLES

Overview

This chapter contains two major sections: *Interfacing to DSP Processors* and *Interfacing Examples.* The *Interfacing to DSP Processors* section provides detailed information about interfacing ADSP-218x family processors to analog-to digital converters (ADCs), digital-to-analog converters (DACs) and coder/decoders (codecs). The *Interfacing Examples* section provides some simple examples of interfacing ADSP-218x family processors to ADCs, DACs, and codecs.

Interfacing to DSP Processors

Current technology offers highly integrated DSPs that contain on-chip ADCs and DACs, as well as the DSP itself. These integrated DSPs eliminate most of the interface problems of separate components. Additionally, stand-alone ADCs and DACs are now available with interfaces especially designed for DSP chips, thereby minimizing or eliminating external interface support or glue logic.

High performance sigma-delta ADCs and DACs are currently available separately or in the same package (called a codec). Some examples of codecs include the AD73311 and AD73322. These products are also designed to require minimum glue logic when interfacing to the most common DSP chips. This section discusses the various data transfer and timing issues associated with ADCs, DACs, and codecs.

Parallel Interfacing to DSP Processors

Interfacing an ADC or a DAC to a fast DSP via a parallel interface requires an understanding of how the DSP processor reads data from a memory-mapped peripheral (the ADC) and how the DSP processor writes data to a memory-mapped peripheral (the DAC). We will first consider some general timing requirements for reading and writing data. It should be noted that the same concepts presented here regarding ADCs and DACs apply equally when reading and writing from/to external memory.

Reading Data from Memory-Mapped ADCs

Figure 10-1 provides a block diagram for a typical parallel DSP interface to an external ADC. This diagram has been greatly simplified to show only those signals associated with reading data from an external memory-mapped peripheral device.

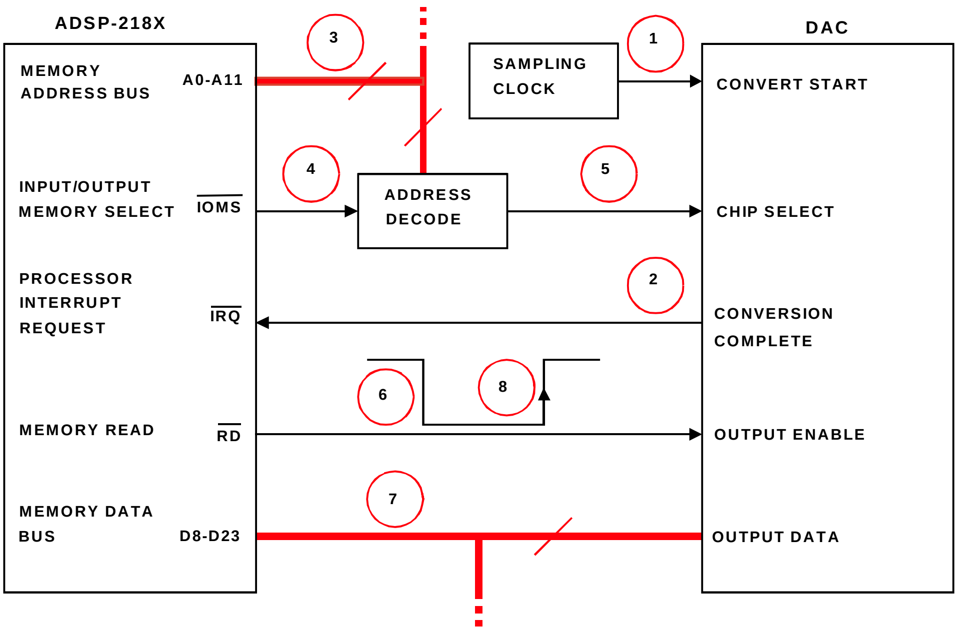

Figure 10-1. ADC to ADSP-218x DSP Parallel Interface

Figure 10-2 shows the timing diagram for the ADSP-218x read-cycle.

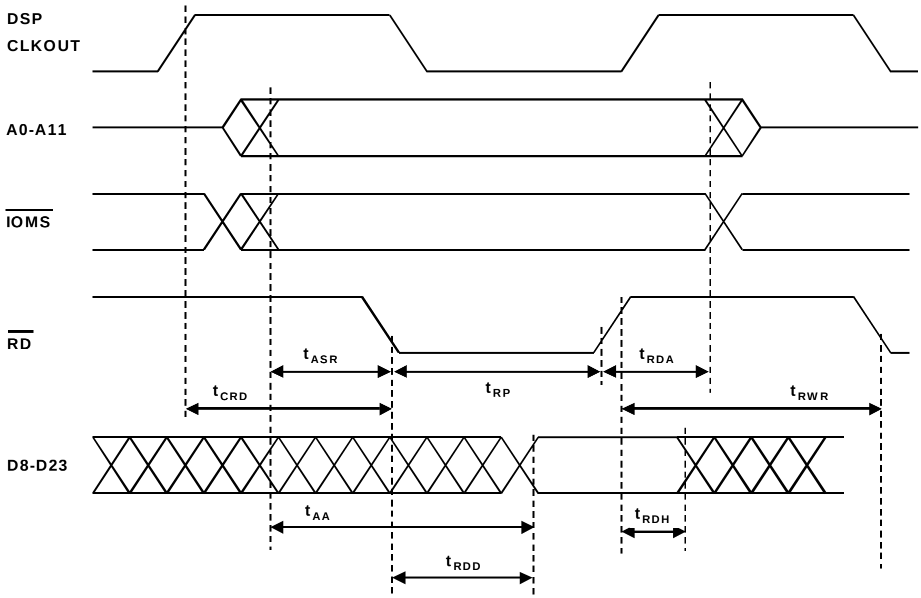

Figure 10-2. ADSP-218x DSP Memory Read Timing

In this example, it is assumed that the ADC is sampling at a continuous rate which is controlled by the external sampling clock, not the internal DSP clock. Using a separate clock for the ADC is the preferred method, since the DSP clock may be noisy and introduce jitter in the ADC sampling process, thereby increasing the noise level.

Assertion of the sampling clock at the ADC *convert start* input initiates the conversion process (step 1). The leading (or trailing) edge of this pulse causes the internal ADC sample-and-hold to switch from the sampling mode to the hold mode so that the conversion process can take place.

When the conversion is complete, the *conversion complete* output of the ADC is asserted (step 2). The read process thus begins when this signal is applied to the processor interrupt request line ($\overline{IRQ}$) of the DSP. The processor then places the address of the peripheral initiating the interrupt request (the ADC) on the memory address bus (A13- A0) (step 3). At the same time, the processor asserts a memory select line ($\overline{IOMS}$ is shown here) (step 4).

The two internal address buses of the ADSP-218x (Program Memory address bus and Data Memory address bus) share a single external address bus, and the two internal data buses (program memory data bus and data memory data bus) share a single external data bus. The boot memory select ($\overline{BMS}$), data memory select ($\overline{DMS}$), program memory select ($\overline{PMS}$), and input/output memory select ($\overline{IOMS}$) signals indicate which memory space the external buses are being used for. These signals are typically used to enable an external address decoder as shown in Figure 10-1. The output of the address decoder drives the chip select input of the peripheral device (step 5).

The memory read ($\overline{RD}$) is asserted t_{ASR} ns after the $\overline{IOMS}$ line is asserted (step 6). The sum of the address decode delay plus the peripheral chip select setup time should be less than t_{ASR} in order to take full advantage of the $\overline{RD}$ low-time. The $\overline{RD}$ line remains low for t_{RP} ns. The memory read signal is used to enable the three-state parallel data outputs of the peripheral device (step 7). The $\overline{RD}$ line is connected to the appropriate pin on the peripheral device usually called output enable or read. The rising edge of the $\overline{RD}$ signal is used to clock the data on the data bus into the DSP processor (step 8). After the rising edge of the $\overline{RD}$ signal, the data on the data bus must remain valid for t_{RDH} ns, the data hold time. In the case of most members of the ADSP-218x family, this specification value is 0 ns.

The following list provides the key requirements for a parallel peripheral device read interface. Values are given for the ADSP-2189M DSP operating at 75 MHz.

- Peripheral device data outputs must be three-state compatible.
- Address decode delay plus peripheral chip select setup time must be less than address and memory select setup time t_{ASR} (0.325 ns minimum for an ADSP-2189M DSP).
- For zero wait-state access, the time from a negative-going edge of read signal ($\overline{RD}$) to output data valid must be less than t_{RDD} (1.65 ns maximum for an ADSP-2189M DSP operating at 75 MHz). Otherwise, software wait states must be added or processor clock frequency reduced.
- Output data from the peripheral must remain valid for t_{RDH} from the rising edge of read signal ($\overline{RD}$) (0 ns for the ADSP-2189M).
- The peripheral device must accept minimum output enable pulse width of t_{RP} (3.65 ns for ADSP-2189M operating at 75 MHz). Otherwise, software wait states must be added or processor clock frequency reduced.

The DSP t_{RDD} specification determines the peripheral device data access time requirement. In the case of the ADSP-2189M, t_{RDD} = 1.65 ns minimum at 75 MHz. If the access time of the peripheral is greater than this, wait states must be added or the processor speed reduced. This is a relatively common situation when interfacing external memory or ADCs to fast DSPs.

The following equations provide the relationship between these timing parameters for the ADSP-2189M (these specifications are dependent on the DSP clock frequency):

- t_{CK} = Processor Clock Period (13.3 ns)
- t_{ASR} = Address and Memory Select Setup before Read Low = $0.25t_{CK}$ – 3 ns minimum
- t_{RDD} = Read Low to Data Valid = 0.5 t_{CK} – 5 ns + # wait states x t_{CK} maximum
- t_{RDH} = Data Hold from Read High = 0 ns minimum
- t_{RP}= Read Pulse Width = 0.5 t_{CK} – 3ns + # wait states x t_{CK} minimum

The ADSP-2189M processor can be interfaced easily to slow peripheral devices, using its programmable wait state generation capability. Three registers control wait state generation for boot, program, data and I/O memory spaces. You can specify 0 to 15 wait states for each parallel memory interface. Each added wait state increases the allowable external data memory access time by an amount equal to the processor clock period (13.3 ns for the ADSP-2189M operating at 75 MHz). In this example, the I/O memory address, $\overline{\text{IOMS}}$, and $\overline{\text{RD}}$ lines are all held stable for an additional amount of time equal to the duration of the wait states.

The AD7854/AD7854L is an example of ADCs that operated in parallel mode. It is a 12 bit, 200/100 KSPS ADC. It operates on a single +3 V to +5.5 V supply and dissipates only 5.5 mW (+3 V supply, AD7854L). An automatic powerdown after conversion feature reduces this to 650 µW.

Figure 10-3 shows a functional block diagram of the AD7854/AD7854L. The AD7854/AD7854L uses a successive approximation architecture, which is based on a charge redistribution (switched capacitor) DAC. A calibration mode removes offset and gain errors

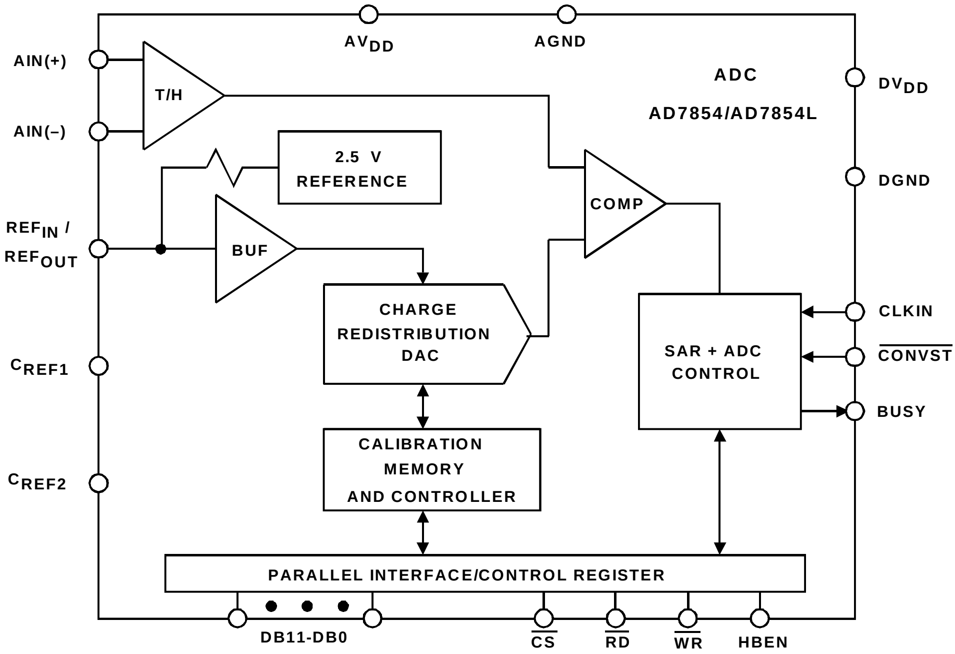

Figure 10-3. AD7854/AD7854L Functional Block Diagram

The key interface timing specifications for the AD7854/AD7854L and the ADSP-2189M are compared in Table 10-1. Specifications for the ADSP-2189M are given for a clock frequency of 75 MHz.

Table 10-1. Parallel Read Interface Timing Specification Comparison between the ADSP-2189M and AD7854/AD7854L

ADSP-2189M Processor (75 MHz)	AD7854/AD7854L ADC
t_{ASR} (Data Address Memory Select Setup Time before $\overline{RD}$ Low) = 0.325 ns minimum	t_5 ($\overline{CS}$ to $\overline{RD}$ Setup Time) = 0 ns minimum (Must add Address Decode Time to this value)
[1]t_{RP} ($\overline{RD}$ Pulse Width) = 3.65 ns + # wait states x 13.3 ns minimum = 70.15 ns minimum	t_7 ($\overline{RD}$ Pulse Width) = 70 ns minimum
t_{RDD} ($\overline{RD}$ Low to Data Valid) = 1.65 ns + # wait states x 13.3 ns minimum = 68.15 ns minimum	t_8 (Data Access Time after $\overline{RD}$) = 50 ns maximum
t_{RDH} (Data Hold from $\overline{RD}$ High) = 0 ns minimum	[2]t_9 (Bus Relinquish Time after $\overline{RD}$) = 5 ns minimum/40 ns maximum

1 Adding 5 wait states to the ADSP-2189M DSP increases t_{RP} to 70.15 ns, which is greater than t_7 (70 ns) and meets the t_8 (50 ns) requirement.

2 t_9 maximum (40 ns) may cause bus contention if a write cycle immediately follows the read cycle.

Examining the timing specifications shown in Table 10-1reveals that for the timing between the devices to be compatible, 5 software wait states must be programmed into the ADSP-2189M. This increases t_{RDD} to 68.15 ns, which is greater than the data access time of the AD7854/AD7854L (t_8 = 50 ns maximum). The read pulse, t_{RP}, is likewise increased to 70.15 ns, which meets the ADC's read pulse width requirement (t_7 = 70 ns minimum). Unless the memory-mapped peripheral has an extremely short access time, wait states are generally required, whether interfacing to ADCs, DACs, or external memory.

A simplified interface diagram for the two devices is shown in Figure 10-4. The conversion complete signal from the AD7854/AD7854L corresponds to the BUSY output pin. Notice that the configuration allows the DSP to write data to the AD7854/AD7854L parallel interface control register. This is needed in order to set various options in the AD7854/AD7854L and perform the calibration routines. In normal operation, however, data is read from the AD7854/AD7854L as described above. Writing to external parallel memory-mapped peripherals is discussed in the next section, "Writing Data to Memory-Mapped DACs",

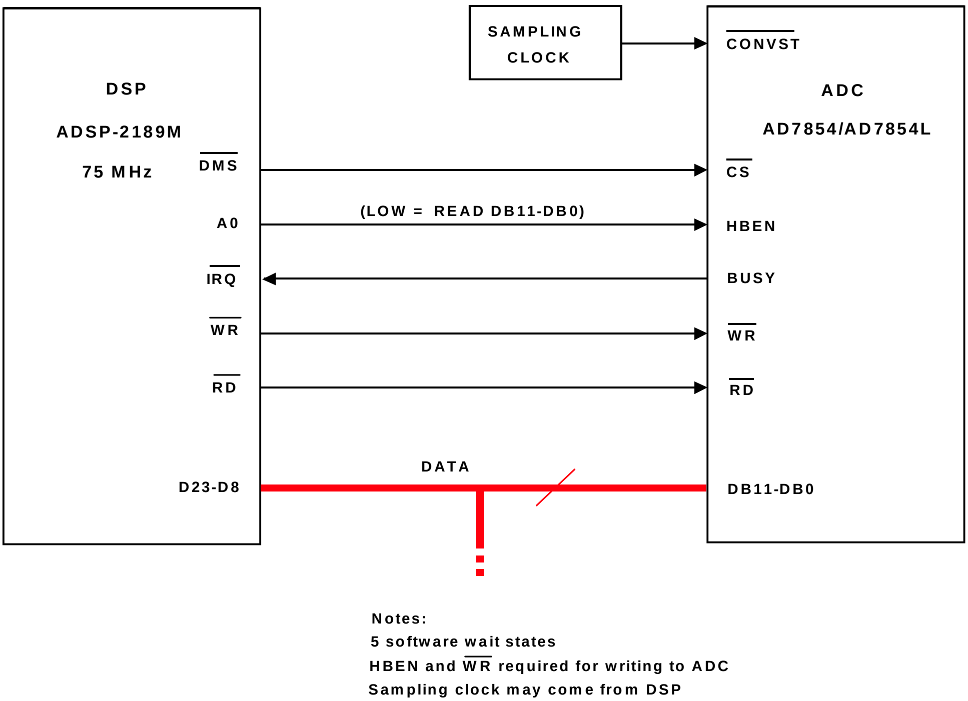

Figure 10-4. AD7854/AD7854L ADC Parallel Interface to ADSP-2189M

Parallel interfaces between other DSP processors and external peripherals can be designed in a similar manner by carefully examining the timing specifications for all appropriate signals for each device. The data sheets for most ADCs contain sufficient information in the application section to interface them to the DSPs.

Writing Data to Memory-Mapped DACs

Figure 10-5 shows a simplified block diagram of a typical DSP interface to a parallel peripheral device (such as the DAC shown in this figure).

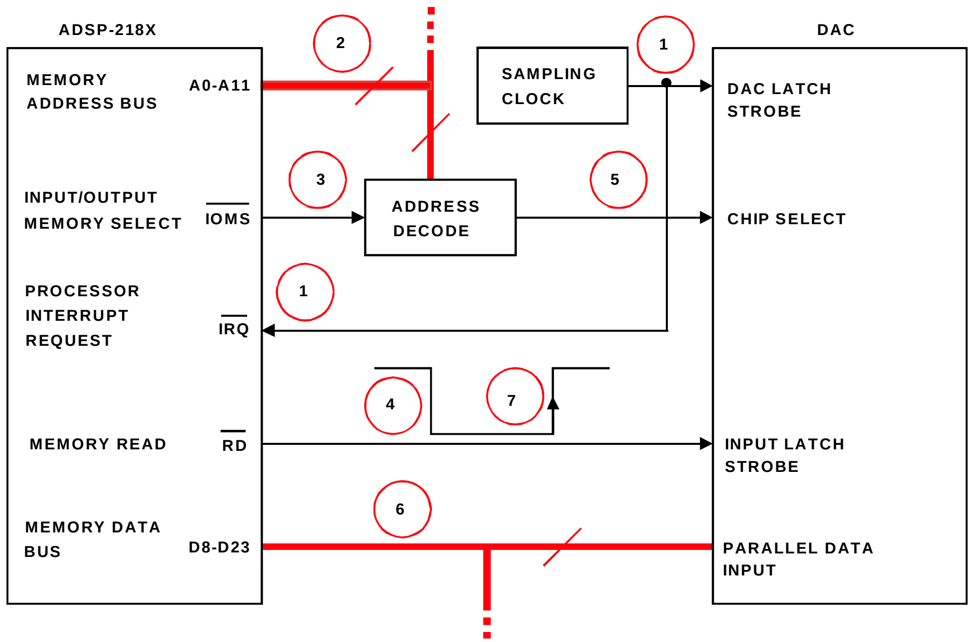

Figure 10-5. DAC to ADSP-218x DSP Parallel Interface

Figure 10-6 shows the memory-write cycle timing diagram for the ADSP-21xx-family.

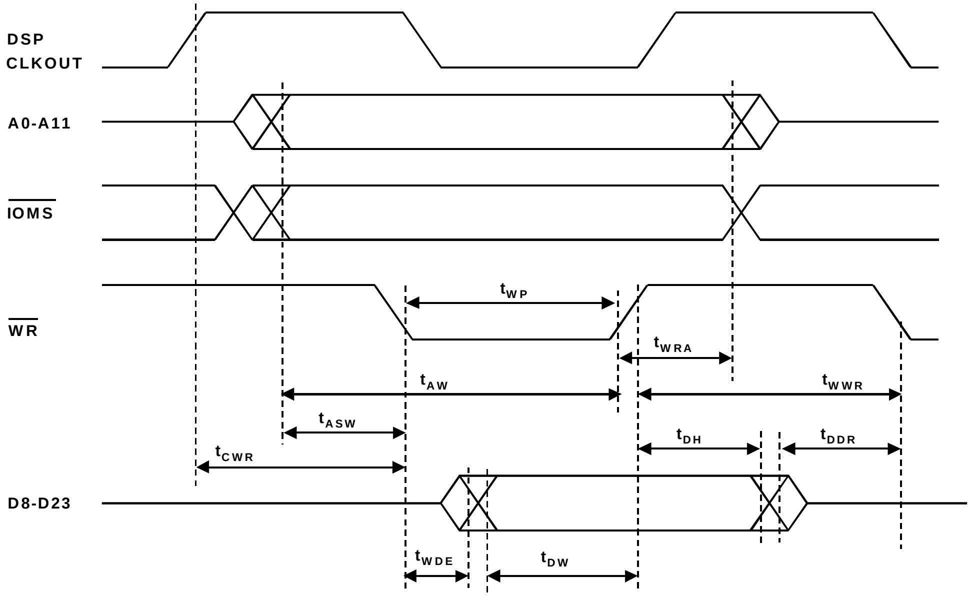

Figure 10-6. ADSP-218x DSP Memory Write Timing

In most real-time applications, the DAC is operated continuously from a stable sampling clock. Most DACs for these applications have double buffering: an input latch to handle the asynchronous DSP interface, followed by a second latch (called the DAC latch), which drives the DAC current switches. The DAC latch strobe is derived from an external stable sampling clock. In addition to clocking the DAC latch, the DAC latch strobe is also used to generate a processor interrupt to the DSP, which indicates that the DAC is ready for a new input data word.

The write process is thus initiated by the peripheral device asserting the DSP interrupt request line. This indicates that the peripheral device is ready to accept a new parallel data word (step 1). The DSP then places the address of the peripheral device on the address bus (step 2) and asserts a memory select line ($\overline{DMS}$ is shown here) (step 3). This causes the output of the address decoder to assert the chip select input to the peripheral (step 5). The write ($\overline{WR}$) output of the DSP is asserted t_{ASW} ns after the negative-going edge of the $\overline{DMS}$ signal (step 4). The width of the $\overline{WR}$ pulse is t_{WP} ns. Data is placed on the data bus (D) and is valid t_{DW} ns before the $\overline{WR}$ line goes high (step 6). The positive-going transition of the $\overline{WR}$ line is used to clock the data on the data bus (D) into the external parallel memory (step 7). The data on the data bus remains valid for t_{DH} ns after the positive-going edge of the $\overline{WR}$ signal.

The following is a list of key requirements for a parallel peripheral device write interface. The key specification is t_{WP}, the write pulse width.

- Address decode delay plus peripheral chip select setup time must be less than the address and memory select setup time t_{ASW} (0.325 ns for the ADSP-2189M processor operating at 75 MHz).
- For zero wait-state access, input data setup time must be less than t_{DW} (2.65 ns for the ADSP-2189M processor operating at 75 MHz). Otherwise, software wait states must be added or processor clock frequency reduced.
- Input data hold time must be less than t_{DH} (2.325 ns for the ADSP-2189M processor operating at 75 MHz).
- The peripheral device must accept input write clock pulse width t_{WP} (3.65 ns minimum for the ADSP-2189M processor operating at 75 MHz). Otherwise, software wait states must be added or processor clock frequency reduced.

All but the fastest peripheral devices require wait states to be added due to their longer data access times.

The following equations show the key timing specifications for the ADSP-2189M. Note that they are all related to the processor clock frequency.

- t_{CK} = Processor Clock Period (13.3 ns)
- t_{ASW} = Address and Memory Select Setup before $\overline{WR}$ Low = $0.25t_{CK}$ – 3 ns minimum
- t_{DW} = Data Setup before $\overline{WR}$ High = 0.5 t_{CK} – 4 ns + # wait states x t_{CK}
- t_{DH} = Data Hold after $\overline{WR}$ High = 0.25 t_{CK} – 1 ns
- t_{WP}= $\overline{WR}$ Pulse Width = 0.5 t_{CK} – 3ns + # wait states x t_{CK} minimum

Another parallel device is the AD5340. It is a 12-bit 100 KSPS DAC with a parallel data interface. It operates on a single +2.5 V to +5.5 V supply and dissipates only 345 µW (+ 3V supply). A powerdown mode further reduces the power to 0.24 µW.

The AD5340 incorporates an on-chip output buffer that can drive the output to both supply rails. The AD5340 allows the choice of a buffered or unbuffered reference input. The device has a poweron reset circuit that ensures that the DAC output powers on at 0 V and remains there until valid data is written to the device. Figure 10-7 shows a block diagram of the AD5340.The input is double buffered.

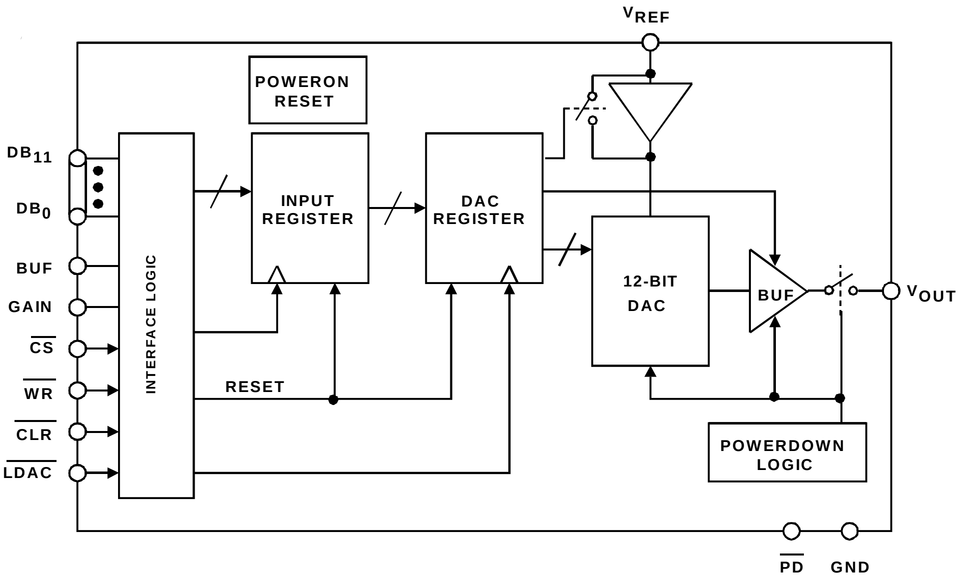

Figure 10-7. AD5340 Parallel Input DAC

Table 10-2 compares the key interface timing specifications for the ADSP-2189M DSP and the AD5340 DAC. Specifications for the ADSP-2189M are given for a clock frequency of 75 MHz.

Table 10-2. Parallel Write Interface Timing Specification Comparison between the ADSP-2189M DSP and AD5340 DAC

ADSP-2189M Processor (75 MHz)	AD5340 DAC
t_{ASW} (Address and Data Memory Select Setup Time before $\overline{WR}$ Low) = 0.325 ns minimum	t_1 ($\overline{CS}$ to $\overline{RD}$ Setup Time) = 0 ns minimum
t_{WP} ($\overline{WR}$ Pulse Width) = 3.65 ns + # wait states x 13.3 ns minimum =30.25 ns minimum	t_3 ($\overline{WR}$ Pulse Width) = 20 ns minimum
t_{DW} (Data Setup before $\overline{WR}$ High) = 2.65 ns + # wait states x 13.3 ns minimum = 29.25 ns minimum	t_4 (DataValid to $\overline{WR}$ Setup Time) = 5 ns maximum
t_{DH} (Data Hold after $\overline{WR}$ High) = 2.325 ns minimum	t_5 (DataValid to $\overline{WR}$ Hold Time) = 4.5 ns minimum

Note: Adding 2 wait states to the ADSP-2189M DSP increases t_{WP} to 30.25 ns and t_{DW} to 29.25 ns, which is greater than t_3 (20 ns) and t_4 (5 ns), respectively.

Examining the timing specifications shown in Table 10-2 reveals that for the timing between the devices to be compatible, two software wait states must be programmed into the ADSP-2189M processor. This increases the width of $\overline{WR}$ to 30.25 ns, which is greater than the minimum required by the AD5340 write pulse width (20 ns). The data setup time of 5 ns for the AD5340 is also met by adding two wait states. A simplified interface diagram for the two devices is shown in Figure 10-8.

Parallel interfaces with other DSP processors can be designed in a similar manner by carefully examining the timing specifications for all appropriate signals for each device.

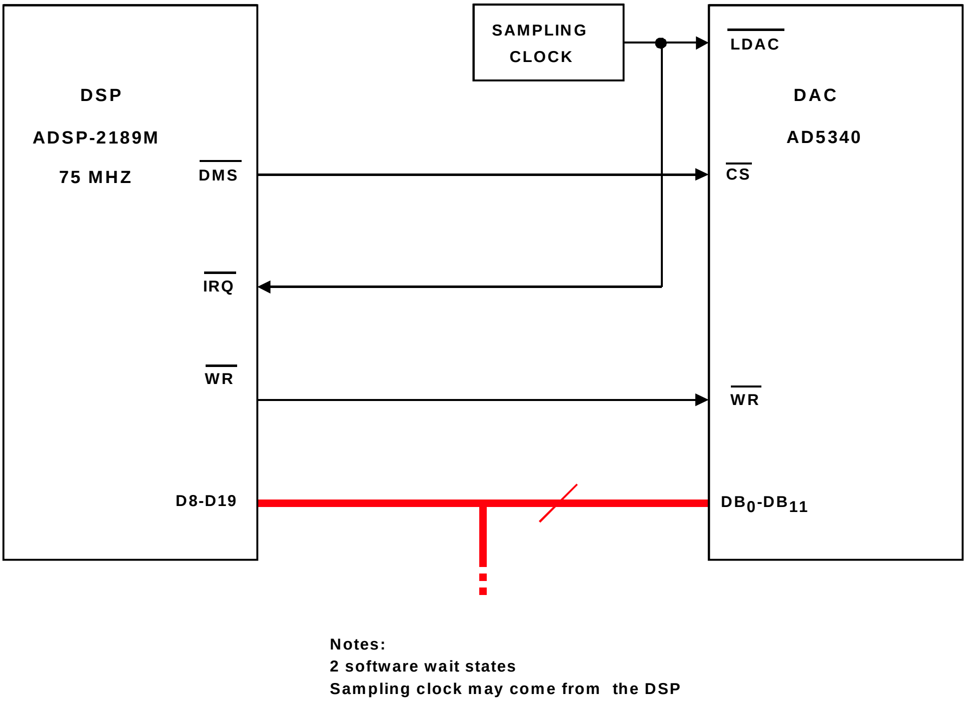

Figure 10-8. AD5340 DAC Parallel Interface to ADSP-2189M

Serial Interfacing to DSP Processors

DSP processors that have serial ports, such as the ADSP-218x family, provide a simple interface to peripheral ADCs and DACs. Use of the serial port eliminates the need for using large parallel buses to connect the ADCs and DACs to the DSP.

In order to understand serial data transfer better, we will first examine the serial port operation of the ADSP-218x series. A block diagram of one of the two serial ports (SPORTs) of the ADSP-218x is shown in Figure 10-9.

The transmit (TX) and receive (RX) registers are not memory mapped, but they are identified by name in the ADSP-218x assembly language. For SPORT0, the transmit and receive registers are named TX0 and RX0, respectively. For SPORT1, these registers are named TX1 and RX1, respectively.

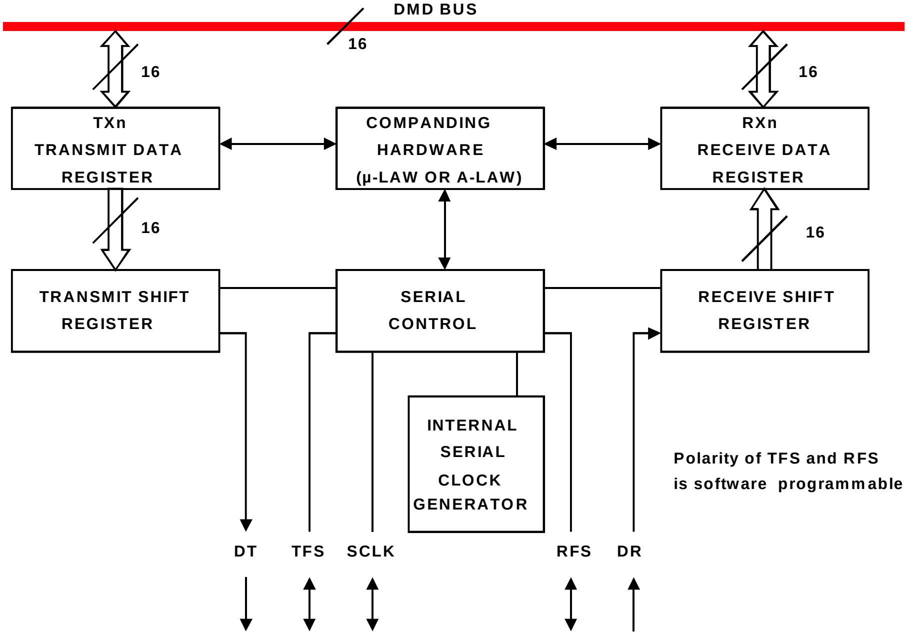

Figure 10-9. ADSP-218x Family Serial Port Block Diagram

In the receiving portion of the serial port, the receive frame sync (RFS) signal initiates reception. The serial receive data (DR) from the external device (ADC) is transferred into the receive shift register one bit at a time. The negative-going edge of the serial clock (SCLK) is used to clock the serial data from the external device into the receive shift register. When a complete word has been received, it is written to the receive data register (RX), and the receive interrupt for that serial port is generated. The receive data register is then read by the processor.

Writing to the transmit data register (TX) readies the serial port for transmission. The transmit frame sync (TFS) signal initiates transmission. The value in the TX register is then written to the internal transmit shift register. The data in the transmit shift register is sent to the peripheral device (DAC) one bit at a time, and the positive-going edge of the serial clock (SCLK) is used to clock the serial transmit data (DT) into the external device. When the first bit has been transferred, the serial port generates the transmit interrupt. The transmit data register can then be written with new data, even though the transmission of the previous data is not complete.

In the normal framing mode, the frame sync signal (RFS or TFS) is checked at the falling edge of SCLK. If the framing signal is asserted, data is available (transmit mode) or latched (receive mode) on the next falling edge of SCLK. The framing signal is not checked again until the word has been transmitted or received.

In the alternate framing mode, the framing signal is asserted in the same SCLK cycle as the first bit of a word. The data bits are latched on the falling edge of SCLK, but the framing signal is checked only on the first bit. Internally-generated framing signals remain asserted for the length of the serial word.

The alternate framing mode of the serial port in the ADSP-218x is normally used to receive data from ADCs and transmit data to DACs.

The serial ports of the ADSP-218x family are extremely versatile. The TFS, RFS, or SCLK signals can be generated from the ADSP-218x clock (master mode) or generated externally (slave mode). The polarity of these signals can be reversed with software, thereby allowing more interface flexibility. The port also contains µ-law and A-law companding hardware for voice-band telecommunications applications.

Serial ADC to DSP Interface

Figure 10-10 shows a timing diagram of the ADSP-2189M serial port operating in the receive mode (alternate framing). The first negative-going edge of the SCLK to occur after the negative-going edge of the RFS signal clocks the MSB data from the ADC into the serial input latch. The process continues until all serial bits have been transferred into the serial input latch.

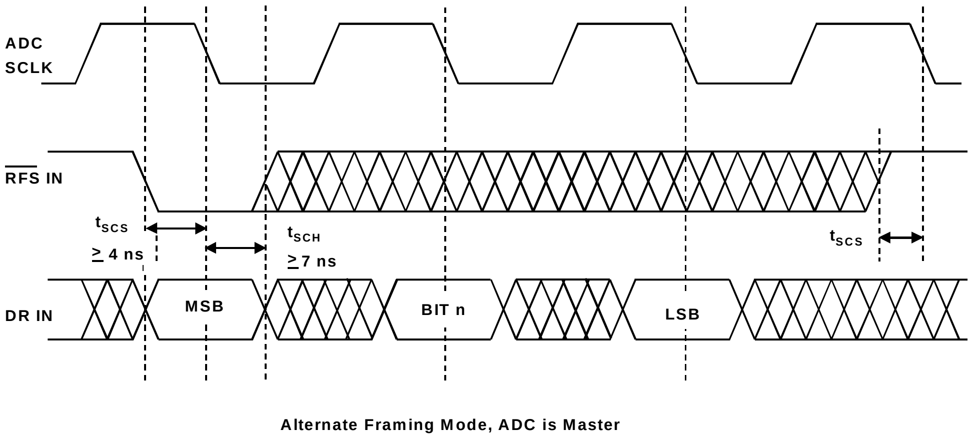

Figure 10-10. ADSP-2189M Serial Port Receive Timing

The key timing specifications of concern are the serial data setup (t_{SCS}) and hold times (t_{SCH}) with respect to the negative-going edge of the SCLK. In the case of the ADSP-2189M, these values are 4 ns and 7 ns, respectively. The latest generation ADCs with high speed serial clocks will have no trouble meeting these specifications, even at the maximum serial data transfer rate.

The AD7853/AD7853L is a 12 bit, 200/100 KSPS ADC which operates on a single +3 V to +5.5 V supply and dissipates only 4.5 mW (+3 V supply, AD7853L). After each conversion, the device automatically powers down to 25 µW. The AD7853/AD7853L is based on a successive approximation architecture and uses a charge redistribution (switched capacitor) DAC. A calibration feature removes gain and offset errors.

Figure 10-11 shows a block diagram of the AD7853/AD7853L.

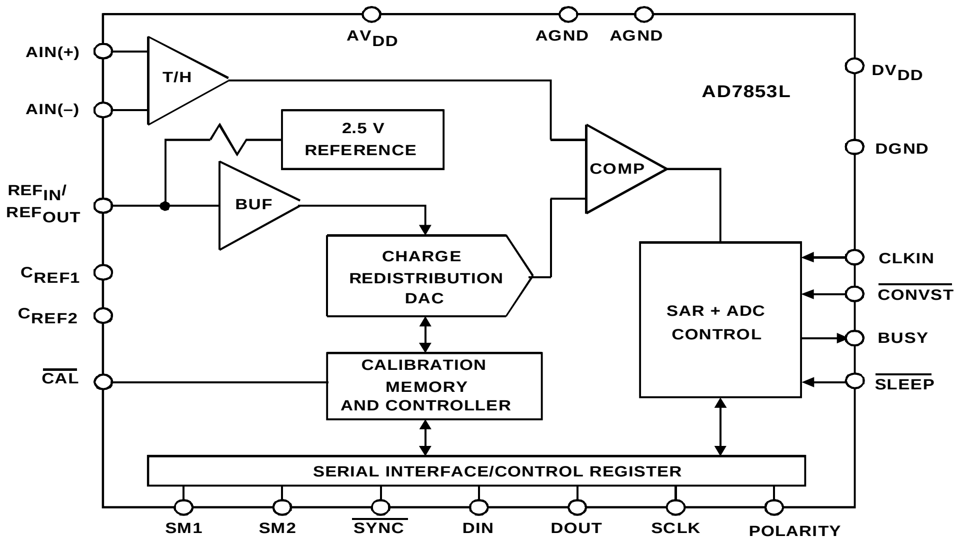

Figure 10-11. AD7853/AD7853L ADC Serial Output

The AD7853 operates on a 4 MHz maximum external clock frequency. The AD7853L operates on a 1.8 MHz maximum external clock frequency. Figure 10-12 shows the timing diagram for AD7853L.

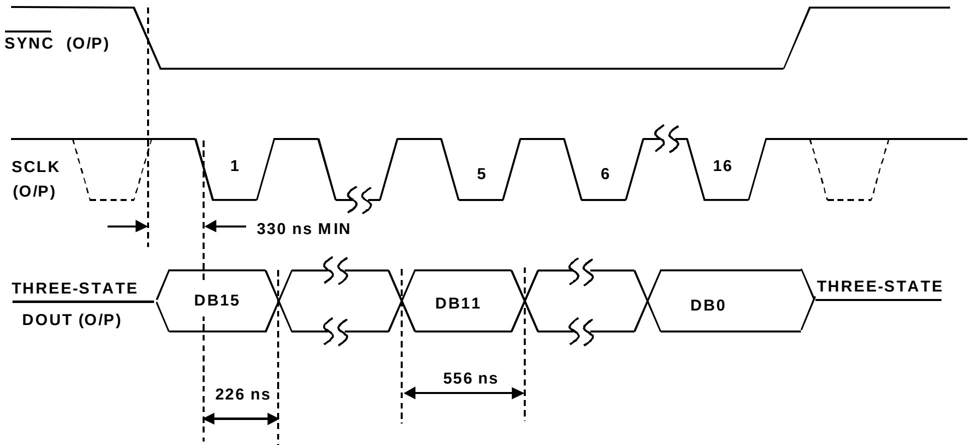

Figure 10-12. AD7853L ADC Serial Output Timing

The AD7853/AD7853L ADCs have external mode pins, SM1 and SM2, which configure the $\overline{\text{SYNC}}$ and SCLK signals as inputs or outputs. In the examples shown in Figure 10-11 and Figure 10-12, the $\overline{\text{SYNC}}$ and SCLK signals are generated internally by the AD7853L.

The AD7853L serial clock operates at a maximum frequency of 1.8 MHz (556 ns period). The data bits are valid 330 ns after the positive-going edges of SCLK. This allows a setup time of approximately 330 ns minimum before the negative-going edges of SCLK, which easily meets the ADSP-2189M 4 ns t_{SCS} requirement.

The hold-time after the negative-going edge of SCLK is approximately 226 ns, which again easily meets the ADSP-2189M 7 ns t_{SCH} timing requirement. These simple calculations show that the data and RFS setup and hold requirements of the ADSP-2189M are met with considerable margin.

Figure 10-13 shows the AD7853L interfaced to the ADSP-2189M and connected in a mode to transmit data from the ADC to the DSP (alternate/master mode).

The AD7853/AD7853L contains internal registers that can be accessed by writing from the DSP to the ADC via the serial port. These registers are used to set various modes in the AD7853/AD7853L as well as to initiate the calibration routines. These connections are not shown in the diagram.

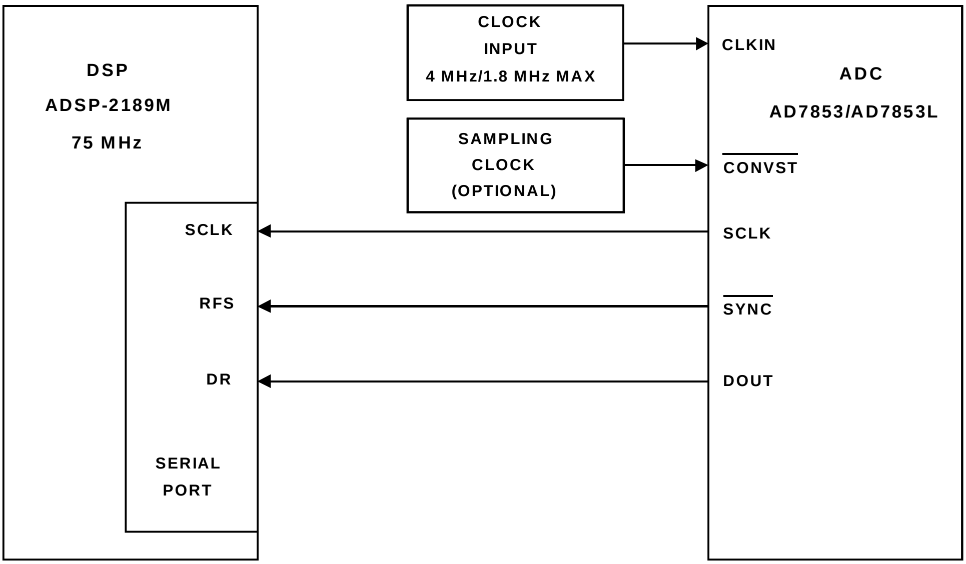

Figure 10-13. AD7853/AD7853L ADC Serial Interface to ADSP-2189M

Serial DAC to DSP Interface

Interfacing serial input DACs to the serial ports of DSPs, such as the ADSP-218x family, is also relatively straightforward and similar to the previous discussion regarding serial output ADCs. The details will not be repeated here, but a simple interface example will be shown.

The AD5322 is a 12-bit, 100 KSPS dual DAC with a serial input interface. It operates on a single +2.5 V to +5.5 V supply. Figure 10-14 shows a block diagram of the AD5322.

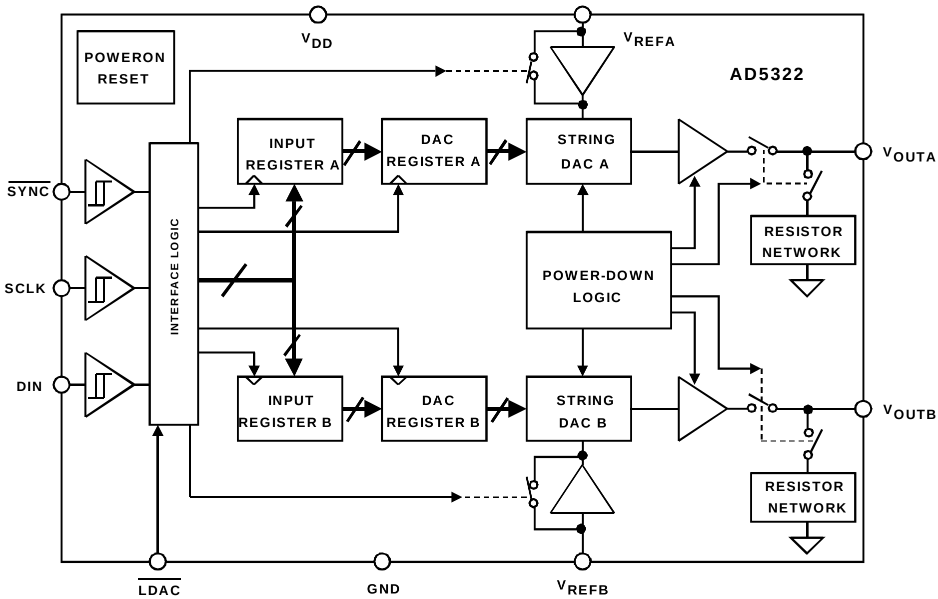

Figure 10-14. AD5322 Dual DAC

Power dissipation on a +3 V supply is 690 µW. A powerdown feature reduces this to 0.15 µW. Total harmonic distortion is greater than 70 dB below full scale for a 10 kHz output.

The references for the two DACs are derived from two reference pins (one per DAC). The reference inputs may be configured as buffered or unbuffered inputs. The outputs of both DACs may be updated simultaneously using the asynchronous $\overline{\text{LDAC}}$ input. The device contains a poweron reset circuit that ensures that the DAC outputs power up to 0 V and remain there until a valid write to the device takes place.

Data is normally input to the AD5322 via the SCLK, DIN, and $\overline{\text{SYNC}}$ pins from the serial port of the DSP. When the $\overline{\text{SYNC}}$ signal goes low, the input shift register is enabled. Data is transferred into the AD5322 on the falling edges of the following 16 clocks. Figure 10-15 shows a typical interface between the ADSP-2189M and the AD5322.

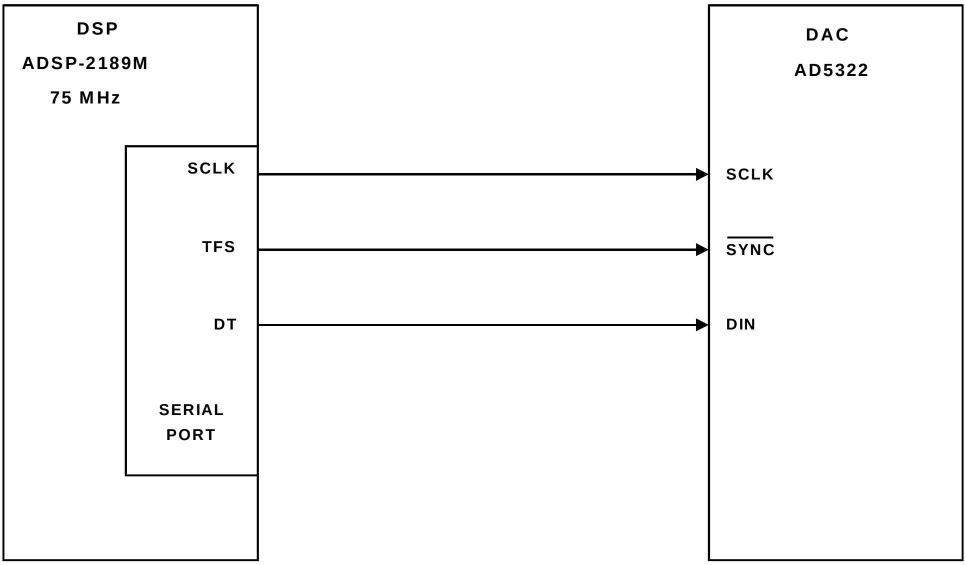

Figure 10-15. AD5322 DAC Serial Interface to ADSP-2189M

Notice that the clocks to the AD5322 are generated from the ADSP-2189M clock. It is also possible to generate the SCLK and $\overline{\text{SYNC}}$ signals externally to the AD5322 and use them to drive the ADSP-2189M. The serial interface of the AD5322 is not fast enough to handle the ADSP-2189M maximum master clock frequency. However, the serial interface clocks are programmable and can be set to generate the proper timing for fast or slow DACs.

The input shift register in the AD5322 is 16 bits wide. This 16-bit word consists of four control bits followed by 12 bits of DAC data. The first bit loaded determines whether the data is for DAC A or DAC B. The second bit determines if the reference input will be buffered or unbuffered. The next two bits control the operating modes of the DAC (normal, powerdown with 1 kΩ to ground, powerdown with 100 kΩ to ground, or powerdown with a high impedance output).

Interfacing I/O Ports, Analog Front Ends, and Codecs

Since most DSP applications require both an ADC and a DAC, I/O Ports and codecs have been developed that integrate the two functions on a single chip as well as provide easy-to-use interfaces to standard DSPs. These chips also go by the name of analog front ends (AFEs).

An example of an analog front end is the AD73322. This device is a dual analog front end with two 16-bit ADCs and two 16-bit DACs and is capable of sampling at 64 KSPS. It is designed for general purpose applications, including speech and telephony using sigma-delta ADCs and sigma-delta DACs. Each channel provides 77 dB signal-to-noise ratio over a voiceband signal bandwidth.

Figure 10-16 shows a functional block diagram of the AD73322.

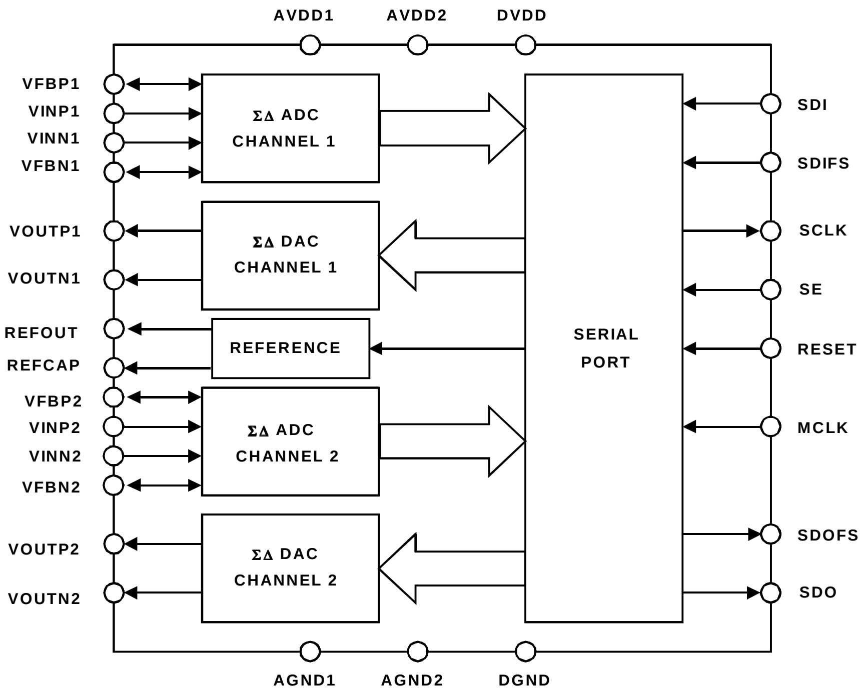

Figure 10-16. AD73322 Codec with Serial Interface

The ADC and DAC channels feature programmable input/output gains with ranges of 38 dB and 21 dB, respectively. An on-chip voltage reference is included to allow single supply operation on +2.7 V to +5.5 V. Power dissipation is 73 mW with a +3 V supply.

The sampling rate of the codecs is programmable with four separate settings of 64 kHz, 32 kHz, 16 kHz, and 8 kHz when operating from a master clock of 16.384 MHz.

The serial port allows easy interfacing of single or cascaded devices to industry standard DSP engines, such as the ADSP-218x family. The SPORT transfer rate is programmable to allow interfacing to both fast and slow DSP engines. Figure 10-17 shows the AD73322 interface to the ADSP-218x family.

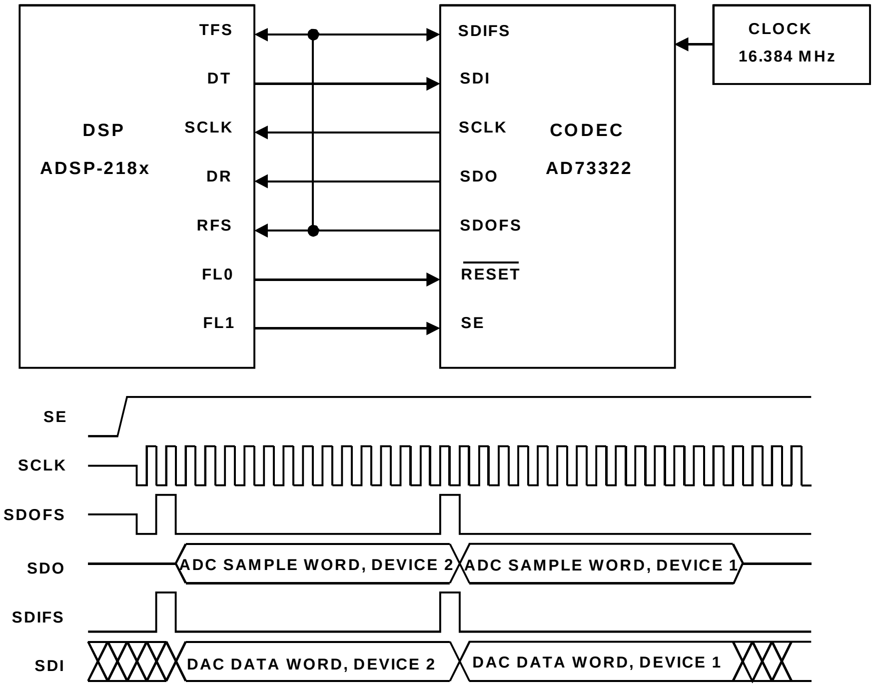

Figure 10-17. AD73322 Interface to ADSP-218x Family Processors

The SE pin (SPORT enable) may be controlled from a parallel output pin or a flag pin, such as FL1; or, where SPORT powerdown is not required, it can be permanently strapped high using a suitable pull-up resistor. The RESET pin may be connected to the system hardware reset, or it may be controlled with another flag bit.

In the program mode, data is transferred from the DSP to the AD73322 control registers to set up the device for desired operation. Once the device has been configured by programming the correct settings to the various control registers, the device may exit the program mode and enter the data mode. The dual ADC data is transmitted to the DSP in two blocks of 16-bit words. Similarly, the dual DAC data is transmitted from the DSP to the AD73322 in two blocks of 16-bit words. Simplified interface timing is also shown in Figure 10-17.

The AD73422 is the first product in the dspConverter™ family of products that integrate a dual analog front end (AD73322) and a DSP (52 MIPS ADSP-2185L/ADSP-2186L). The entire functionality of the dual-channel codec and the DSP fits into a small, 119-ball 14 mm by 22 mm plastic ball grid array (PBGA) package. The obvious advantage of this size package is the saving of circuit board real estate. ADC and DAC signal-to-noise ratios are approximately 77 dB over voiceband frequencies.

The AD74222-80 integrates 80 K bytes of on-chip memory configured as 16 K words (24-bit) of program RAM, and 16 K words (16-bit) of data RAM. The AD73422-40 integrates 40 K bytes of on-chip memory configured as 8 K words (24-bit) of program RAM, and 8 K words (16-bit) of data RAM. Powerdown circuitry is also provided to meet the low power needs of battery operated portable equipment. The AD73422 operates on a +3 V supply and dissipates approximately 120 mW with all functions operational.

The following summarizes the features of the ADSP73422 dspConverter™:

- Complete dual codec (AD73322) and DSP (ADSP-2185L/ADSP-2186L)
- 14 mm by 22 mm BGA package
- +3 V single-supply operations, 73 mW power dissipation
- Powerdown mode

- Codec
    - Dual 16-bit sigma-delta ADCs and DACs
    - Data rates: 8, 16, 32, 64 KSPS
    - 77 dB SNR
- DSP
    - 52 MIPS
    - ADSP-218x code compatible
    - 80 K byte and 40 K byte on-chip memory options

High-Speed Interfacing

With the advent of ever faster DSP clock rates and newer architectures it has become possible to acquire and process high speed signals. The programmability of DSPs makes it possible to run different algorithms on the same hardware while providing different system functionality.

An example of high-speed ADC is the AD9201. It is a dual-channel, 10-bit, 20 MSPS ADC that operates on a single +2.7 V to +5.5 V supply and dissipates only 215 mW (+3 V supply). The AD9201 offers closely matched ADCs needed for many applications, such as I/Q communications. Input buffers, an internal voltage reference and multiplexed, digital, output buffers make interfacing to the AD9201 very simple.

The companion part to the AD9201 ADC is the AD9761 DAC. The AD9761 is a dual, 10-bit, 20 MSPS per channel DAC operating on a single +2.7 V to +5.5 V supply and dissipating only 200 mW (+3 V supply). A voltage reference, digital latches and 2x interpolation make the AD9761 useful for I/Q transmitter applications.

Figure 10-18 shows a simplified ADSP-218x system connected to the AD9201 ADC and the AD9761 DAC. The ADC and DAC both have parallel interfaces connected to the external port of the DSP.

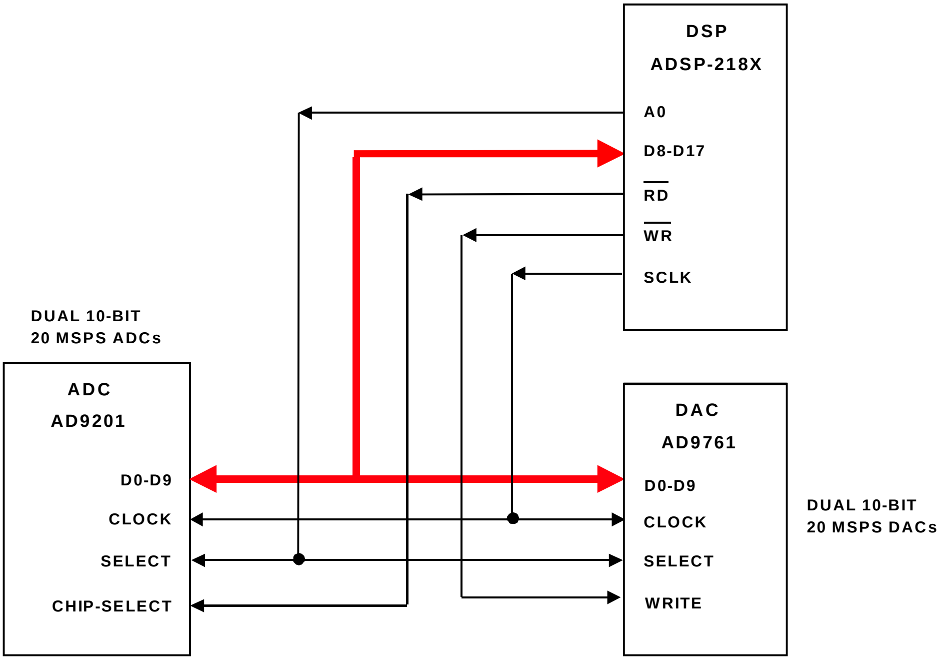

Figure 10-18. AD9201 ADC and AD9761 DAC Interface to ADSP-218x

Due to the simple interface between the DSP processor and the AD9201 and AD9761, shown in Figure 10-18, a memory select signal is not required. When performing reads from the ADC, only the $\overline{\text{RD}}$ signal is required to assert the chip select of the AD9201. Writes require only the use of the $\overline{\text{WR}}$ signal to the AD9761. If additional peripherals are to be interfaced to the DSP's external bus, some external decoding logic would be required.

DSP System Interface

Figure 10-19 shows a simplified ADSP-2189M system using the full memory mode configuration with two serial devices, a byte-wide EPROM, and optional external program and data overlay memories.

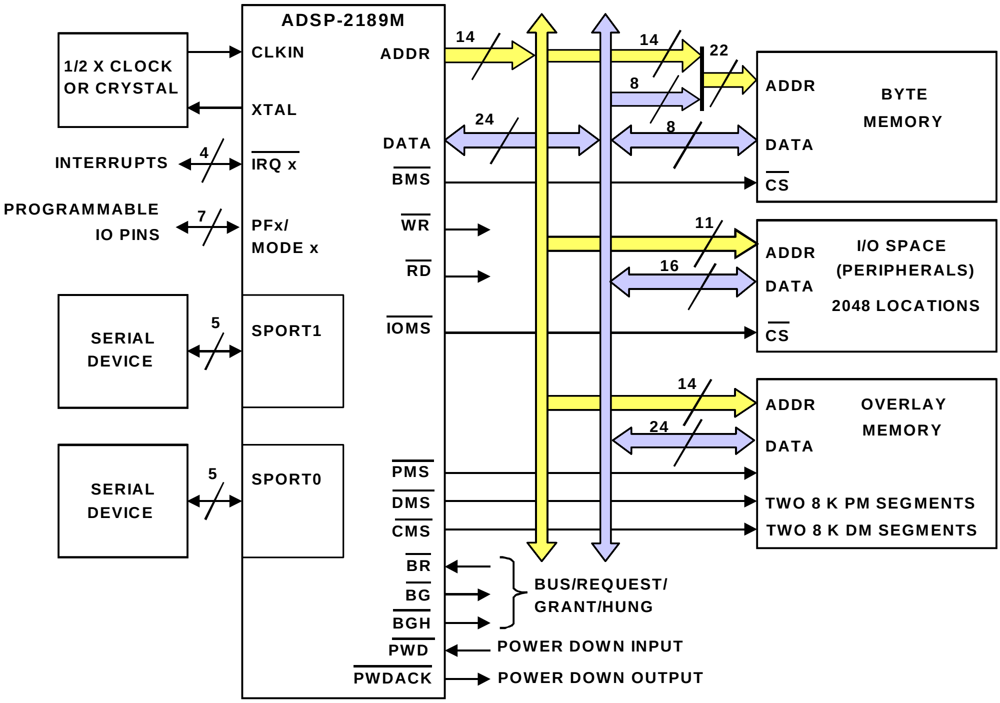

Figure 10-19. ADSP-2189M System Interface (Full Memory Mode)

Programmable wait state generation allows the fast processor to connect easily to slower peripheral devices. The ADSP-2189M also provides four external interrupts, seven general-purpose input/output pins, and two serial ports.

SPORT1 can alternately be configured as two additional interrupts ($\overline{\text{IRQ0}}$ and $\overline{\text{IRQ1}}$), a general-purpose input pin (FI), a general purpose output pin (FO), and the serial clock (SCLK). This alternate configuration provides a total of six external interrupts (excluding the non-maskable powerdown interrupt signal, which can also be used as an external interrupt), eight programmable I/O pins, one dedicated input pin, one dedicated output pin, and one serial port.

The ADSP-2189M can also be operated in the Host Memory mode, which allows access to the full external data bus but limits addressing to a single address bit. Additional system peripherals can be added in the Host Memory mode through the use of external hardware to generate and latch address signals.

Interfacing Examples

This section provides some hardware examples of circuits that can be interfaced to the ADSP-218x DSP serial ports or DMA ports. As with any hardware design, it is important that timing information be carefully analyzed. Therefore, the appropriate ADSP-218x processor data sheet should be used in addition to the information presented in this chapter.

Serial Port to Codec Interface

The ADSP-218x family processors may be interfaced, via the serial ports, to most common codec's. An example is shown Figure 10-20, using the AD73311 codec. Up to eight, AD73311 codec's, may be connected in a cascade configuration to obtain multiple channel operation.

When two or more codec's are connected in a cascade configuration, it is necessary to synchronously enable the serial ports and bring all codecs out of reset simultaneously to ensure correct operation of the serial interface. Figure 10-20 illustrates this process.

For a single AD73311 codec, the D-latches are not required. Therefore, the codec $\overline{\text{RESET}}$ input could be connected to the system $\overline{\text{RESET}}$ signal, and the serial port enable could be connected directly to an output flag pin (FLn) on the DSP.

Please refer to the AD73311 data sheet for further application details.

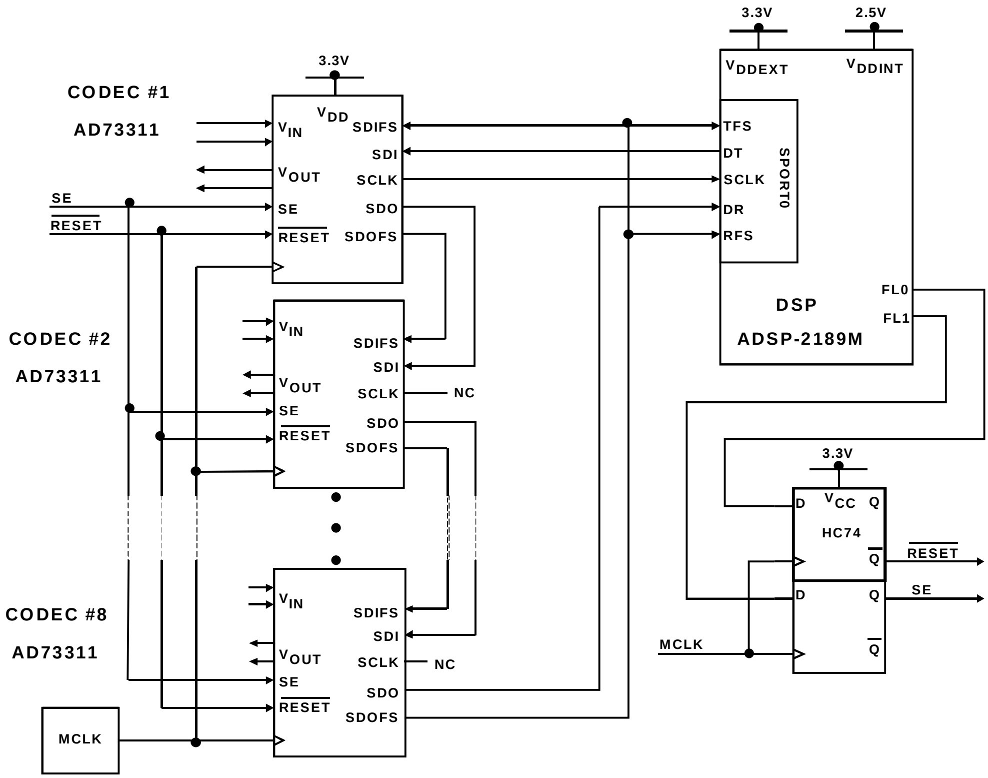

Figure 10-20. AD7311 Codec(s) to ADSP-218x DSP Serial Interface

Serial Port to ADC Interface

This section provides the following two examples of a serial port to ADC interface:

- ADSP-218x DSP SPORT to AD7475/95 ADC interface
- ADSP-218x DSP SPORT to AD7888 ADC interface

ADSP-218x DSP to AD7475/95 ADC Interface

The ADSP-218x DSP SPORT can be interfaced to the AD7475/95 ADC, as shown in Figure 10-21. Note that the RFS pin is configured as an output on the DSP and is used to initiate conversion in the ADC. It is also used to determine the sample rate.

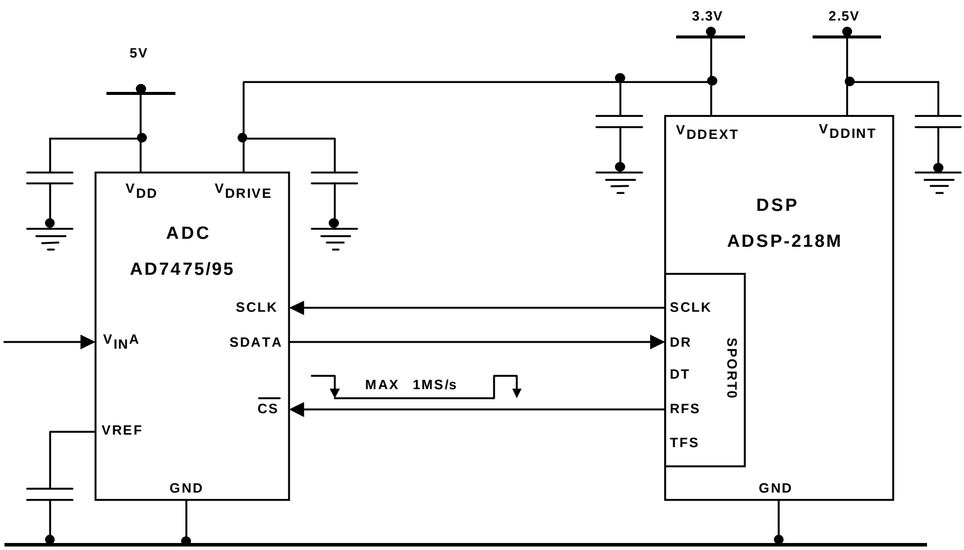

Figure 10-21. ADSP-218x DSP to AD7475/95 ADC Serial Interface

Figure 10-22 shows the serial timing for the ADSP-218x DSP interface to the AD7475/95 ADC.

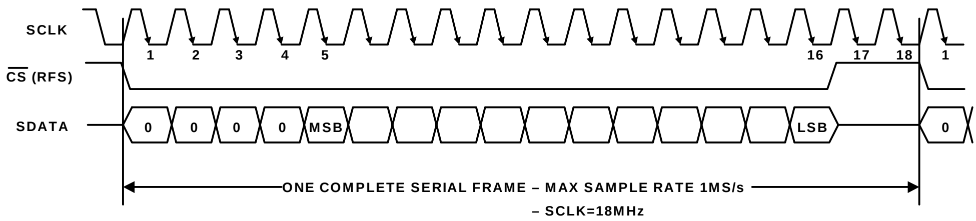

Figure 10-22. ADSP-218x DSP to AD7475/95 ADC Serial Interface Timing

Using a DSP `CLKIN` frequency of 36 MHz, the `CLKOUT` frequency will be 72 MHz. For the correct serial interface timing, the DSP SPORT Control registers should be set up as follows:

SPORTn Control Register, DM (0x3FF6):	0x7DCF
SPORTn SCLKDIV Register, DM (0x3FF5):	0x0001
SPORTn RFSDIV Register, DM (0x3FF4):	0x0011

Note that the DSP `RFS` frame sync output is used to initiate conversion and set the sample rate. For some applications, it may be desirable to use a more stable frequency source, such as an independent clock. In this case, the external clock would be an input to both the $\overline{\texttt{CS}}$ pin on the ADC and the DSP `RFS` frame sync pin.

ADSP-218x DSP to AD7888 ADC interface

Using the 8-channel AD7888 ADC, it is possible to sample eight independent analog inputs. Figure 10-23 shows the serial interface used. Note that there is also a data line from the DSP to the ADC. This data line is used to send channel select and power management information to configure internal ADC registers.

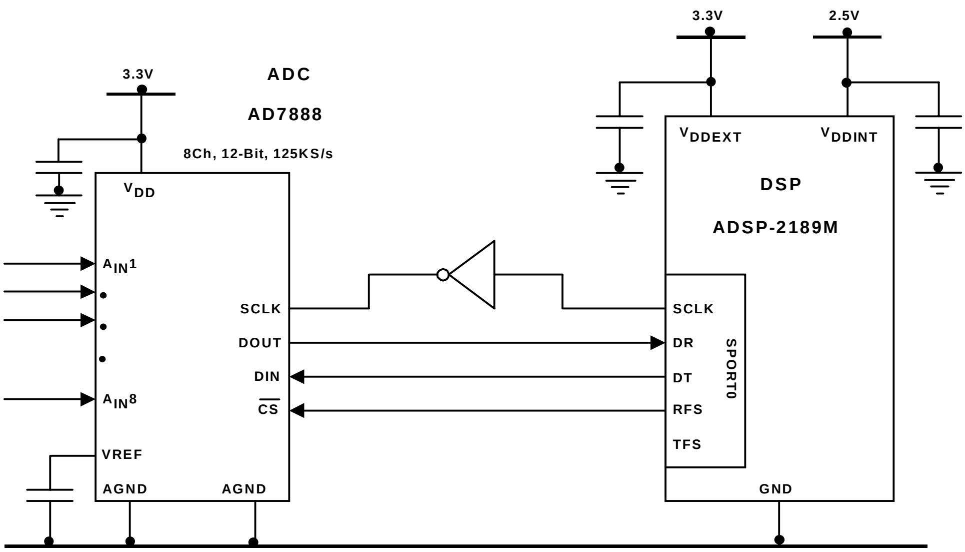

Figure 10-23. ADSP-218x DSP to AD7888 ADC Serial Interface

Figure 10-24 shows the serial interface timing for the ADSP-218x DSP to AD7888 ADC Interface.

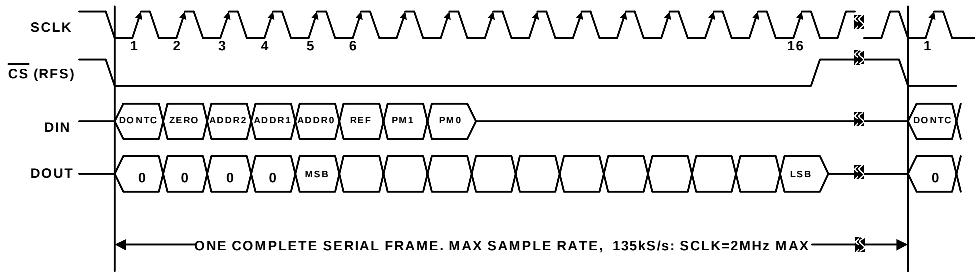

Figure 10-24. ADSP-218x DSP to AD7888 ADC Serial Interface Timing

Using a DSP CLKIN frequency of 36 MHz, the CLKOUT frequency will be 72 MHz. For the correct serial interface timing, the DSP SPORT Control registers should be set up as follows:

SPORTn Control Register, DM(0x3FF6):	0x7DCF
SPORTn SCLKDIV Register, DM (0x3FF5):	0x0011
SPORTn RFSDIV Register, DM (0x3FF4):	0x000F

Note that the DSP RFS frame sync output is used to initiate conversion and set the sample rate. For some applications, a more accurate clock may be needed to set the sample rate and minimize jitter. In this case, the external clock would be an input to both the $\overline{\text{CS}}$ pin on the ADC and the DSP RFS frame sync pin.

Parallel Port to ADC Interface

The ADSP-218x DSPs allow you to interface an ADC to a parallel port. Figure 10-25 shows an interface between the ADSP-218x DSP and the AD7899 ADC.

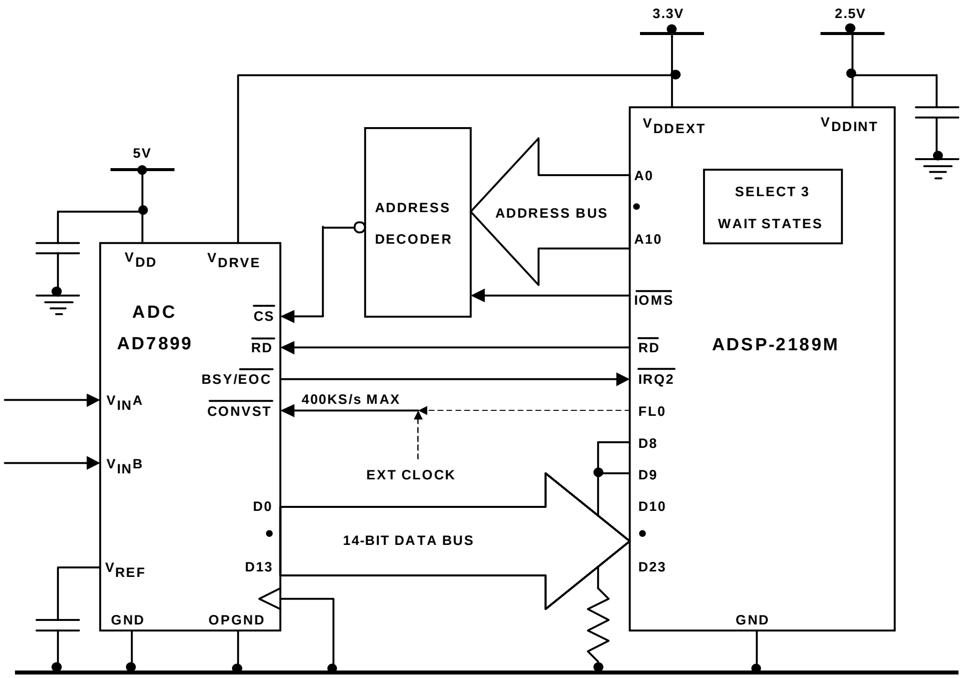

Figure 10-25. ADSP-218x DSP to AD7899 ADC Parallel Interface

The CONVST signal can be generated by the ADSP-218x DSP or from an external clock source. Figure 10-25 shows the ADC CS being generated by a logical decode of the IOMS and the ADSP-218x DSP address bus. The AD7899 ADC is mapped into the 2K IO space of the ADSP-218x DSP.

The AD7899 BSY/EOC line provides an interrupt to the ADSP-218x DSP when the conversion is completed. The converted digital word can be read from the AD7899 using a read operation.

Please note that in this example the 14-bit data from the AD7899 is MSB aligned on the 16-bit external data bus to preserve the sign information of the input data. Therefore, data pins D8 and D9 are pulled to ground since they are unused.

The DSP should be programmed to provide the minimum number of wait states required by the AD7899, three in this example.

The AD7899 is read using the following instruction:

```
MR0 = dm(ADC);
```

Where MR0 is the ADSP-218x MR0 register and ADC is the AD7899 IO address.

Serial Port to DAC Interface

Figure 10-26 shows an example of how to connect a three-wire serial interface between the ADSP-218x DSP SPORT and a typical DAC (AD5320).

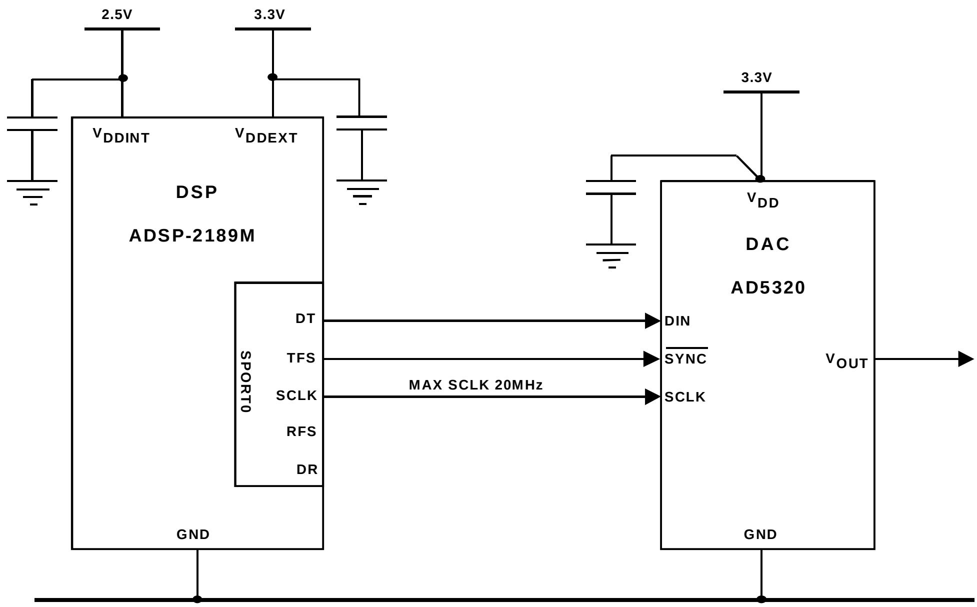

Figure 10-26. ADSP-218x DSP to AD5320 DAC Serial Interface

The associated timing diagram, shown in Figure 10-27, is very similar to the Motorola SPI interface, except that it has been extended to a 16-bit word size to accommodate the AD5320 DAC. The maximum SCLK rate supported by the AD5320, when using a 3.3 V supply, is 20 MHz.

By adding the minimum inactive time for the SYNC pulse of 33 ns, a sample rate over 1 MS/s can be supported.

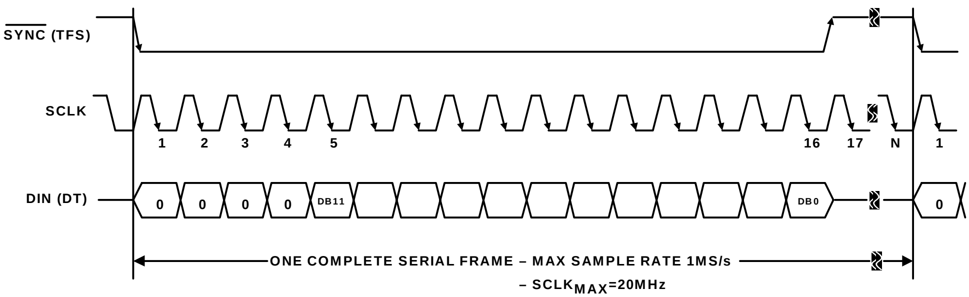

Figure 10-27. ADSP-218x DSP to AD5320 DAC Serial Interface Timing

The SPORT0 interface control registers in the ADSP-218x DSP should be programmed with the following data:

SPORT0 Control Register, DM(0x3FF6):	0x4E8F
SPORT0 SCLKDIV Register, DM(0x3FF5):	0x0001
System Control Register, DM(0x3FFF):	Set Bit-12 to enable SPORT0

This data programs the SCLK to 18.75 MHz, assuming that the ADSP-218x DSP is operating with a CLKOUT frequency of 75 MHz. The data also sets the TFS to alternate inverted mode (active low TFS signal) and the word length to 16-bits. The sample rate is set by the frequency at which data is written to the transmit buffer, but in no case should the rate exceed –1.1 MHz.

IDMA Interface to a Host Processor

The ADSP-218x family processors are ideal candidates for use in co-processing systems.Their extensive DMA and peripheral interface features allow the ADSP-218x processors to function with minimal external support circuitry. In order to realize the highest possible performance in a co-processor system, efficient host-DSP communication is vital.

This section shows an example hardware and software interface between the ADSP-218x processor's Internal DMA (IDMA) port and a microcontroller. As each specific system design has its own requirements and challenges, this section does not presume to provide the only possible solution. Rather, it is meant to provide the system designer a flexible framework of ideas that can be tailored to meet individual system requirements.

The devices selected for this example are the ADSP-2189 processor and the Motorola M68300 family of microcontrollers. The ADSP-2189 is ideal in such a system because of its 192 K bytes of on-chip RAM (configured as 32 K words of on-chip Program Memory RAM and 48 K words of on-chip Data Memory RAM) and its IDMA interface. For a lower cost system, an ADSP-218x family member with less internal memory could be used. The popular Motorola M68300 family of microcontrollers is a good choice as a host because it provides a powerful and flexible bus interface that is easily adaptable to a coprocessing system.

IDMA Operation

External devices can gain access to the internal memory of any of the ADSP-218x family members through the DSP's IDMA port. Host processors accessing the ADSP-218x through IDMA can treat the DSP as a memory-mapped slave peripheral. They have access to all of the DSP's internal Data Memory (DM) and Program Memory (PM) except for the 32 memory-mapped control registers, which reside at addresses DM(0x3FE0) through DM(0x3FFF).

The IDMA port consists of a 16-bit multiplexed address /data bus (IAD16:0), a select line ($\overline{IS}$), address latch (IAL), read ($\overline{IRD}$), write ($\overline{IWR}$), and acknowledge ($\overline{IACK}$) signals. The host processor is responsible for initiating all data transfers. A typical transfer sequence is shown in Figure 10-28.

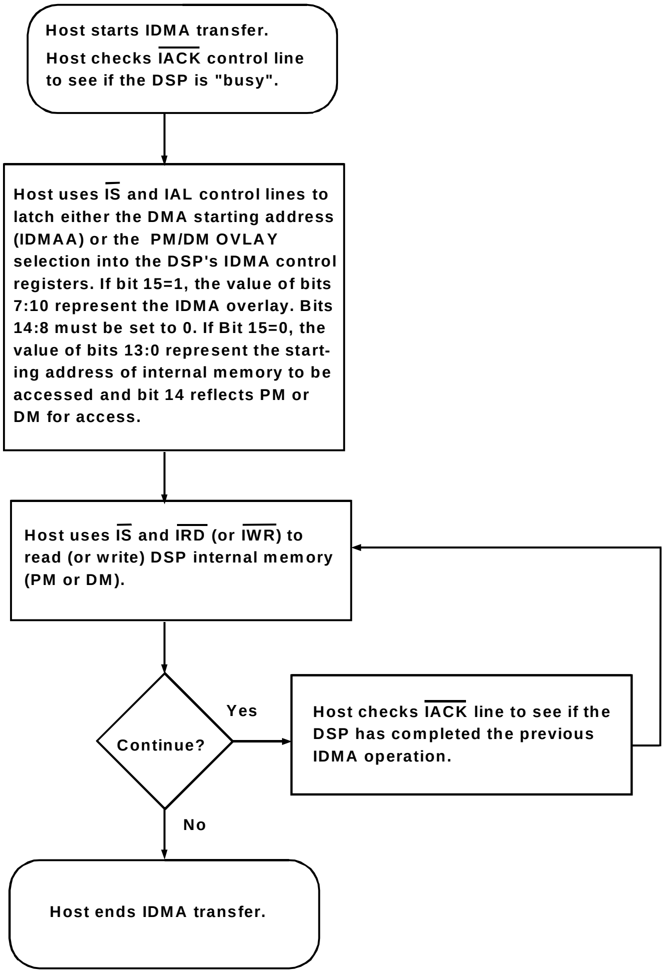

Figure 10-28. IDMA Transfer Sequence

The DSP memory address and destination memory type bit field is loaded into the IDMA Control register, shown in Figure 10-29.

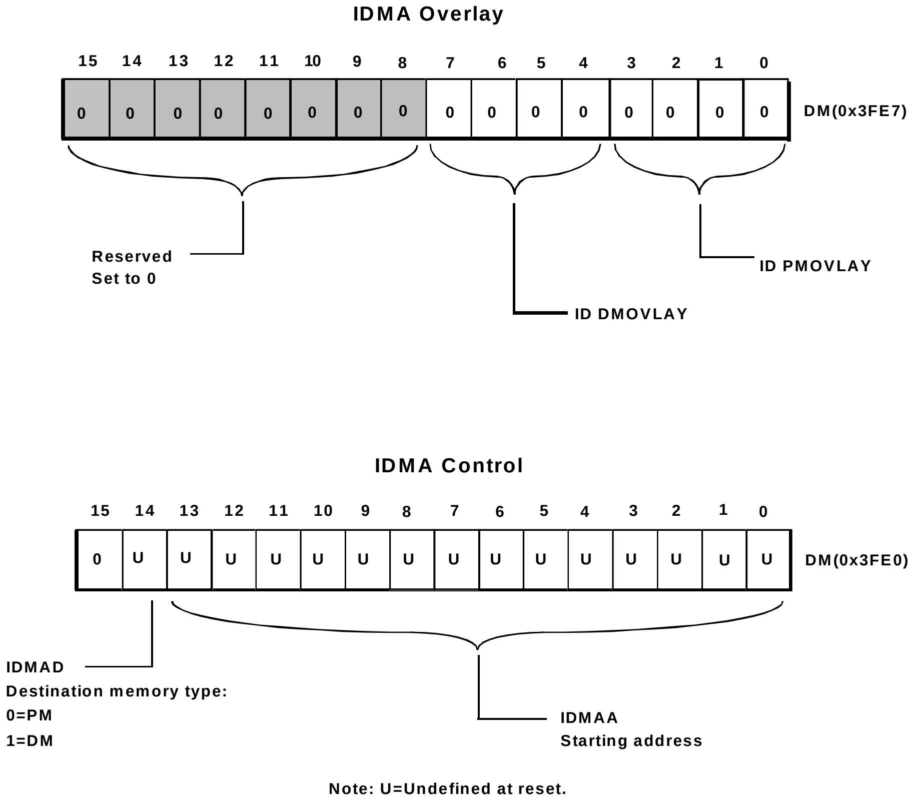

Figure 10-29. IDMA Control Registers

This register contains the 14-bit internal memory address, along with a bit to specify the type of transfer: 24-bit Program Memory opcodes or 16-bit Data Memory data. The IDMAA starting address can be initialized by either the DSP or by a host processor.

The host can initialize the IDMAA starting address by performing an address latch cycle. An address latch cycle is defined by the host asserting the ALE signal and then transferring a 15-bit (14 address bits plus 1 destination memory type bit) value on the IAD pins. If Bit 15 is set to 0, IDMA latches the address. If Bit 15 is set to 1, IDMA latches into the IDMA Overlay register. (Note that the host cannot read the latched address (IDMAA) back.)

The IDMA Overlay register, as shown in Figure 10-29 on page 10-44, is memory mapped at address DM (0x3FE0).

ⓘ The IDMA Overlay register does not apply to the ADSP-2181, ADSP-2183, ADSP-2184, ADSP-2185, and ADSP-2186 processors due to their smaller amounts of on-chip memory.

To streamline the transfer of large segments of opcodes or data, an address latch cycle does not need to be performed for each IDMA access. Instead, once latched, the address is automatically incremented after every IDMA word transfer. Since the IDMA port has a 16-bit bus, 24-bit transfers require two host accesses. The first access transfers the most significant 16 bits; the second access transfers the least significant 8 bits, right justified, with a zero-filled upper byte. IDMA address increments occur after the entire 24-bit word has been transferred. (For more information about the IDMA port see the Data Sheet for the selected DSP.)

Host Interface Hardware Design

The IDMA port of the ADSP-218x processor is mapped into two locations in the microcontroller's external memory space: one location is used by the microcontroller to set the DSP memory address it wishes to access; the other location is used when transferring data and instruction information.

Motorola MC6833x Overview

The Motorola MC6833x Family of microprocessors use a System Integration Module (SIM) to communicate to parallel peripherals. The SIM incorporates separate address and data busses, along with multiple memory select lines and strobe lines. The SIM is common (with minor changes) to all MC6833x processors, and material presented in this section should apply to all processors in the family.

Schematic Explanation

Figure 10-30 provides a schematic showing the glue logic between a Motorola MC68332 processor and the ADSP-2189M processor using address decoding. Figure 10-31 provides a schematic showing the glue logic between a Motorola MC68332 processor and the ADSP-2189M processor using a chip select.

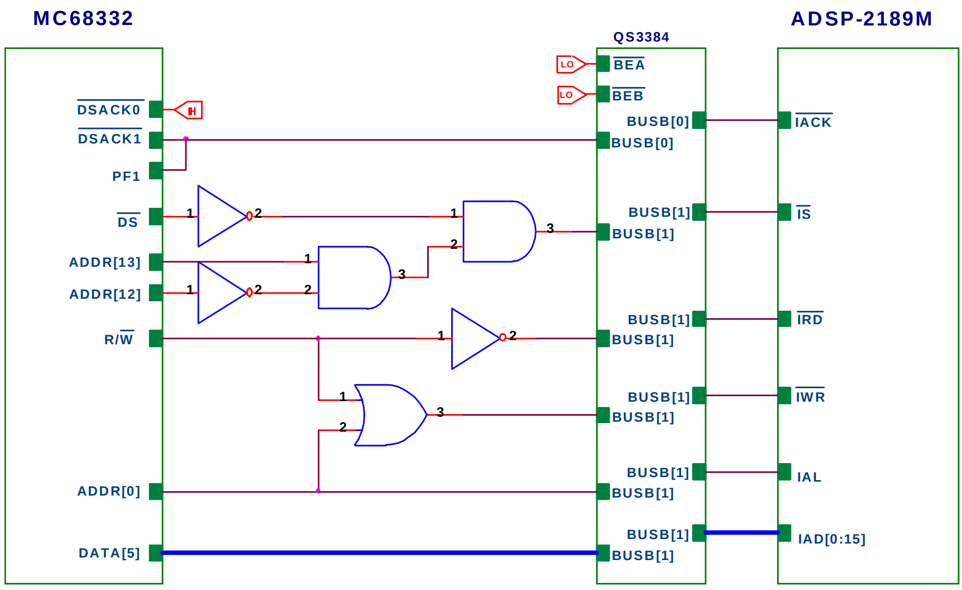

Figure 10-30. Glue Logic between the MC68332 and the ADSP-2189M Using Address Decoding

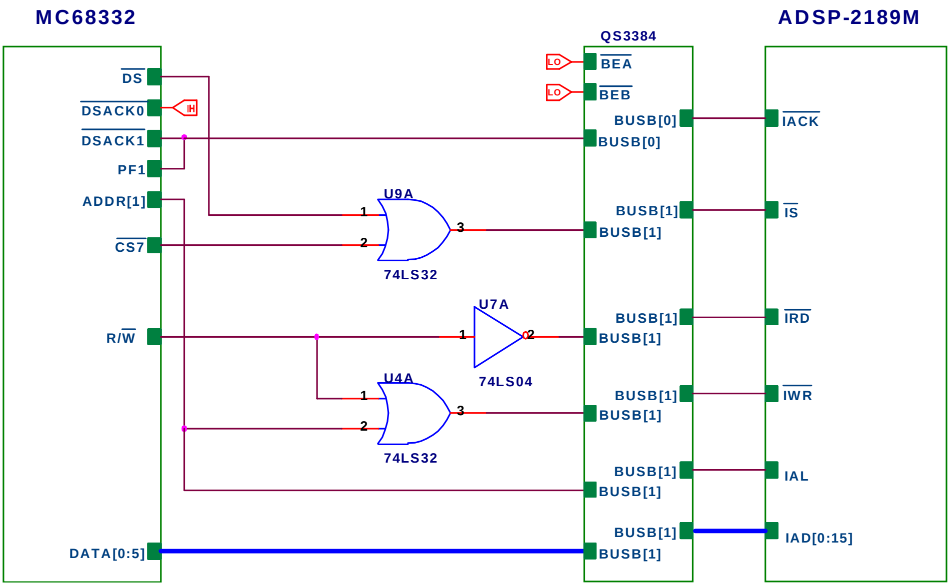

Figure 10-31. Glue Logic Between the MC68332 and the ADSP-2189M Using a Chip Select

Minimal logic is required to connect the external bus of the MC6833x to the IDMA port. All logic necessary for this interface is programmed into a single GAL20V8B programmable logic device. The 16 data lines from the MC6833x are connected via a logic level translator to the ADSP-2189's IAD pins. The MC6833x uses this bus to transmit the DSP memory address, as well as, transfer data to and from the DSP processor.

The IACK signal from the DSP is routed to both the DSACK1 pin and a programmable flag pin on the MC6833x. The DSACK1 pin signals the end of a memory transfer cycle for the MC6833x, while the programmable flag pin is used by the MC6833x to check IACK status prior to initiating a transfer.

Listing 10-1 on page 10-55 shows the microcontroller downloader code, which checks for a low level of the flag prior to any transfer.

The microcontroller's address pin A1 is connected directly to the ALE pin of the IDMA port. To begin a transfer, the microcontroller must first initialize the DSP's IDMAA register through an address latch cycle. This is accomplished by writing the DSP memory address that the microcontroller wants to access to address 0xbbb2 in the microcontroller's memory space.

Address pin A1 is used because it is the least significant address pin used by the microcontroller during 16-bit word transfers. You can assign the base address at which the ADSP-2189 IDMA port resides (in the MC6833x's external memory map) in two different ways:

- Using the MC6833x's address lines A12 and A13 in conjunction with the microcontroller's DS signal
- Using one of the MC6833x Chip Select pins

These signals (A12, A13, $\overline{DS}$, $\overline{CS7}$) are logically combined so that the IDMA port's $\overline{IS}$ signal is asserted (low) when the MC6833x's $\overline{DS}$ pin is asserted (low), A12 is low and A13 is high. With this combination, the IDMA port can be accessed in the microcontroller's memory space at addresses 0x2xxx, 0x6xxx, 0xaxxx, and so on. In the example shown in Figure 10-30 on page 10-46, we use address 0x2000 for data transfers and 0x2002 for IDMA address transfers. Tighter assignment of addresses can be accomplished through the use of additional address lines in the $\overline{IS}$ logic.

The final IDMA control lines that need to be driven by the MC68332 are $\overline{IRD}$ (IDMA Read) and $\overline{IWR}$ (IDMA Write). Since the microcontroller has only a single, multiplexed R/$\overline{W}$ (Read/Write) line, the R/$\overline{W}$ line is inverted and then routed to $\overline{IRD}$ to generate the IDMA read signal. The IDMA write signal, $\overline{IWR}$, is the OR'ed combination of the microcontroller's R/$\overline{W}$ line, and address line 2. This logic is necessary to insure that $\overline{IWR}$ stays high during an IDMA address latch cycle.

System Design Issues

The physical hardware interface between the microcontroller and DSP is just the enabling step in a DSP-based co-processing system. System start-up and host-DSP communication issues must be planned for ahead of time and adequate provisions for these issues should be included into both the microcontroller's and the DSP's firmware.

Booting the DSP

The IDMA port on the DSP can be used to boot load the DSP on powerup. This eliminates the need for a separate EPROM for the DSP. On the ADSP-2189, booting is controlled through the use of the `Mode [A,B,C,D]` pins. Booting through the IDMA port is enabled by holding the `Mode B`, `D` pin low, and the `Mode A`,`C` pin high. With this signal combination, upon deassertion of the $\overline{\texttt{RESET}}$ signal, the DSP does not activate its external address bus to access an EPROM. Instead, the DSP expects a host to begin IDMA transfers to fill its internal Data Memory and Program Memory. This process consists of the host performing standard IDMA instruction and data transfers.

Booting is terminated when the DSP restart vector at DSP Address PM(0x0000) is written. An efficient boot loading sequence would consist of the host filling the DSP's internal Program Memory, starting at location PM(0x0001), and using the automatic address increment feature on the IDMA port to speed the transfer of code block in ascending address order. The host can then initialize Data Memory.

When all initialization is complete, the host should then initialize the DSP's restart vector. Then, the DSP program execution commences. This booting process is shown in Figure 10-32.

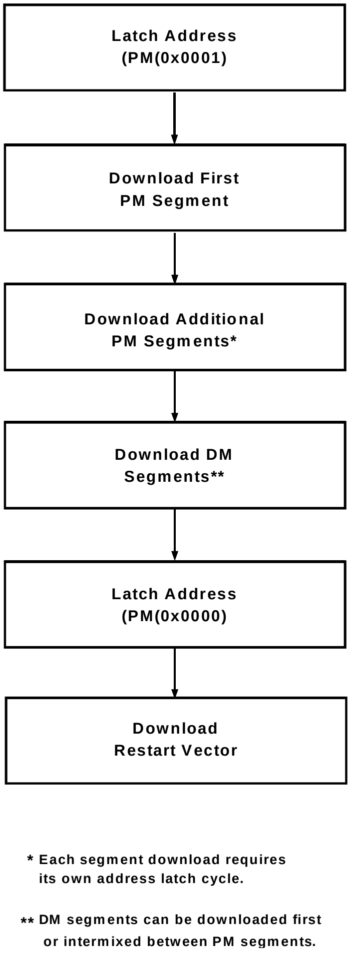

Figure 10-32. IDMA Booting Process

Generating Boot Code

The ADSP-218x processors operate on 24-bit instruction opcodes. The IDMA port can only accept 16-bit values. To transfer instruction opcodes through the IDMA port, the most significant 16 bits are transferred first.

The DSP's IDMA boot files are produced by the ADSP-218x family PROM Splitter, `elfspl21.exe`. The PROM Splitter command line switch, `-idma`, is used to generate an ASCII text output file that is suitable for booting an ADSP-2181, ADSP-2183, ADSP-2184, ADSP-2185, or ADSP-2186 processor. An additional command line switch, `-218x`, enables support for IDMA booting for the ADSP-2187, ADSP-2188, and ADSP-2189 processors, which have additional on-chip memory overlay regions for booting via the IDMA port.

The output file contains a series of IDMA transfer records, each starting with a count (of 16 bit words), an address (consisting of the 14 bit internal address (`IDMAA`) and the 1 bit `IDMAD`), to be written to the IDMA Control register. When using the PROM Splitter's `-218x` command line switch, an additional address word, which represents the IDMA overlay page, is included in the IDMA image file, immediately after the IDMA control word. Each word is expressed as four characters, which represent a 16-bit value in hexadecimal format. The data is displayed as one word per line, as follows:

```
00A8 <—— count value
0001 <—— IDMA control word
8000 <—— IDMA OVERLAY control word (218x)
0001 <—— First Opcode (16 bit MSB), (count -2)
0002 <—— First Opcode (8 bit LSB), (count -3)
0001 <—— Second Opcode (16 bit MSB), (count -4)
0002 <—— Second Opcode (8 bit LSB), (count -5)
: :
: :
: :
: :
5678 <—— Last Opcode (16 bit MSB), (count =1)
```

```
0090 <—— Last Opcode (8 bit LSB), (count =0)
: :
: : <—— additional PM or DM Segments
: :
FFFF <—— End-of-module specifier
```

Host Code Generation Downloading Issues

In order to utilize the data file produced by the PROM Splitter program, the microcontroller needs to be programmed to understand the given format. The PROM Splitter program produces a IDMA image file that can be initialized somewhere in the microcontroller's memory space. Figure 10-33 shows the format of this image file.

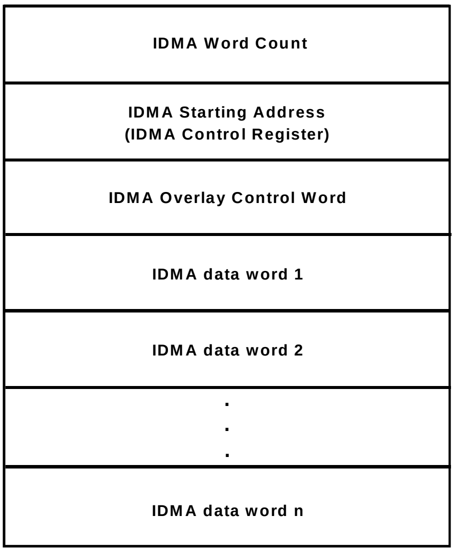

Figure 10-33. IDMA Image File Format

The first element read from the image file is the number of 16-bit words (IDMA Word Count) to be transferred to the DSP (remember that each 24-bit PM opcode counts as two 16-bit words). This value is placed in a data register and can be used as a loop counter to control the download function.

The next value in the IDMA image file is the DSP starting address (IDMA Starting Address), which is the 15-bit value that represents the starting address of the code or data segment that will be transferred during the IDMA access. This starting address value should be written from the host processor into the DSP's IDMA Control register. The DSP's starting address is then followed by the IDMA Overlay Control word, which is used to assign the proper internal DSP overlay memory region that will be accessed during the IDMA transfer.

Please keep in mind that the IDMA Overlay register applies only for the ADSP-2187, ADSP-2188, and ADSP-2189 processors.

The next values are the data or instruction values (IDMA data word 1…n) that need to be transferred. When the microcontroller has transferred the proper number of items (as determined by the count), it gets the next count value from the buffer, the next DSP address, and so on.

The download process stops when the microcontroller encounters a count value of 0xffff. This download process is shown in Figure 10-34. MC68332 assembly code to implement this download process is presented in Listing 10-1 on page 10-55.

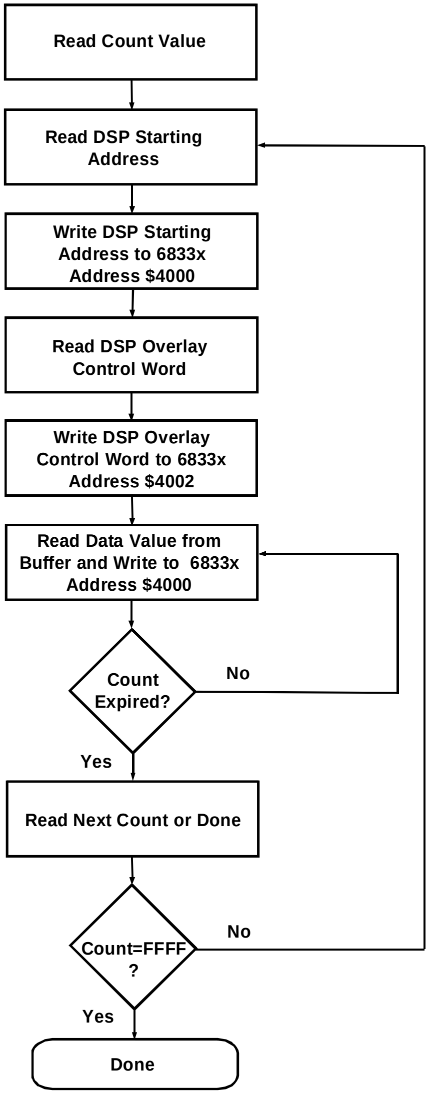

Figure 10-34. MC6833x Download Flow Process

Listing 10-1. Downloading Code and Data to ADSP-2189 IDMA Port Interface Example (MC6833x Assembly Code)

```
; download.asm
;
; This code runs on an MC6833x processor and is used to
; download code and data segments to an ADSP-2189  IDMA port
; interface.
; Note: The ADSP-2189 is a 3.3V device in order to avoid damage
; use 5V to 3.3V logic level Voltage translator (e.g. QS 3384)
;
SCDR    EQU $fffc0e  ;SCI Data Register
SCCR0   EQU $fffc08  ;SCI Control Register 0
SCCR1   EQU $fffc0a  ;SCI Control Register 1
QMCR    EQU $fffc00  ;QSM Configuration Register
SCSR    EQU $fffc0c  ;SCI Status Register
SRAMBAH EQU $fffb44  ;SRAM Base Address Register High Word
SRAMMCR EQU $fffb40  ;SRAM Module Configuration Register
FYPCR   EQU $fffa21  ;SCIM System Protection Control Register
SIMMCR  EQU $fffa00  ;SCIM Configuration Register
CSPAR0  EQU$fffa44   ;Chip Select Pin Assignment Register 0
CSPAR1  EQU $fffa46  ;Chip Select Pin Assignment Register 1
CSBAR0  EQU $fffa4c  ;Chip Select Base Register 0
CSOR0   EQU $fffa4e  ;Chip Select Option Register 0
PORTF0  EQU $fffa18  ;Port F Data Register

; 6833x MEMORY MAP:

; $000000-$0003FF    Interrupt Vector Table {TRAM}
; $000400-$000DFF    Code Space {TRAM}
; $010000-$0101FF    Variables (left blank) {SRAM}
; $0101FF-Downward   Stack Space {SRAM}
; **********************************************************
; Variables
; DSP Code and Data will be placed here
; **********************************************************

org $010000

; Opcode and data information for DSP download should be
; included here

org $000400
```

```
; **********************************************************
; Init: Beginning of the CODE segment
; **********************************************************

Init:
   move.b #$0,(FYPCR).L      ; Turn off watchdog timer
   move.l #$101FE,a7         ; Stack at location $101FE
   move.w #$0001,(SRAMBAH).L ; Move SRAM to $10000
   move.w #$0000,(SRAMMCR).L ; Turn on SRAM (Variables/Stack)
   move.w #$0040,(SIMMCR).L  ; Enable User Mode
   move.w #$3FFF,(CSPAR0).L  ; Enable Chip Selects 0-5
   move.w #$03FF,(CSPAR1).L  ; Enable Chip Selects 6-10
   move.w #$0000,(CSBAR0).L  ; Use Chip Select 0
   move.w #$3822,(CSOR0).L   ; Assert Chip Select 0

top:
   move.w (PORTF0).l,d1      ; Check PF1 to see if IACK low
   and.w #$0002,d1           ; before proceeding
   bne top
   move.l #$002002,a4        ; initialize a4 with Address
                             ; Latch address
   move.l #$002000,a3        ; initialize a3 with data port
                             ; address
   move.l #$010000,a2        ; initialize a2 to start of DSP
                             ; code/data
   move.w (a2)+,d2           ; load count value into d2

tx_rx_loop:

   move.w (PORTF0).l,d1      ; check PF1 to see if IACK low
   and.w #$0002,d1
   bne t x_rx_loop
   move.w (a2)+,(a4)         ; write starting address to IDMAA
   move.w (a2)+,(a4)         ; write IDMA OVERLAY register
                             ; (218x)
sub.w #$1,d2                 ; decrement count

tx_dat a:

   move.w (a2)+,(a3)         ; transfer next instruction
```

```
wait_data:
   move.w (PORTF0).l,d1      ; check PF1 to see if /IACK low
   and.w #$0002,d1
   bne wait_data
   dbf d2,tx_data            ; decrement count to see if at end
                             ; of module
   move (a2),d4              ; get next count value
   sub.w #$ffff,d4           ; check if end of all modules
   beq done_data             ; if at end, send Restart vector
                             ; if booting, done otherwise
   move (a2)+,d2             ; get next module count
   bra tx_rx_loop            ; go back to transferring DSP
                             ; information

done_data:
   bra done_data             ; data file is completed.
```

Host-DSP Message Transfers

In addition to boot-loading the DSP, many systems require continuous interaction between a host microcontroller and the DSP computation engine. The IDMA port of the ADSP-2189 processor was designed so that there does not need to be any DSP core involvement with host microcontroller transfers. The host processor is expected to manage the data flow to and from the DSP.

No DSP interrupts are generated during IDMA accesses, and IDMA transfers occur asynchronously to DSP operation. Therefore, the system designer must allocate DSP internal memory resources and arbitrate host accesses so that there is no conflict between host access and DSP access of DSP internal memory resources. For data transfers, one could allocate an area of internal memory for "messages" and constrain the host to access this area only. For code transfers other than booting, a software flag set in this "message" area could be used to signal the host that the DSP is available for transfer.

Advanced Topics

This section discusses some issues that the system designer may find helpful when using the Motorola MC68332 for more complex systems.

Multiple Processors

In this hardware example, we focused on connecting a single ADSP-2189M DSP to a Motorola MC68332 microprocessor. This scheme can easily be expanded to support multiple DSP processors, without additional glue logic. In a multiple DSP system, multiple $\overline{\text{IS}}$ lines are needed to select each individual DSP processor. The multiple $\overline{\text{IACK}}$ signals from each DSP can be bussed together in a "wired-OR" configuration to create a single $\overline{\text{IACK}}$ signal to the host processor. The 100-pin ADSP-218x processors (all ADSP-218x processors except for the ADSP-2181 and ADSP-2183) support this "wired-OR" $\overline{\text{IACK}}$ logic configuration when their `Mode C` and `Mode D` pins are set to a logic high. In this configuration, an external pulldown resistor is needed, since the $\overline{\text{IACK}}$ signal is driven from an open-drain PMOS transistor.

For our system design, each DSP processor requires two of the Motorola 6833x's memory locations: one memory location is used to perform an IDMA address latch sequence; the other is used for transmitting or receiving IDMA data. Both memory location addresses are used to assert the appropriate $\overline{\text{IS}}$ signal of the specific DSP processor in the system that the host processor wishes to access. In this manner, each DSP processor can be accessed individually.

Hardware Signaling

In many instances, it may be desirable for the host and DSP processors to have additional avenues of communication. The host can use one of its programmable flags as an output attached to a hardware interrupt on the DSP. With this method, the host can alert the DSP before a transfer occurs or inform the DSP that a transfer has been completed. This method can be especially useful since there is no interrupt associated with IDMA operation on the ADSP-2189. The DSP can likewise use a programmable flag as an output to signal the host if there is new data for the host to use or if new code is required for download.

References

The following is a list of references for materials used in developing this chapter and for materials that provide additional information. Please note that many of these materials can be found on Analog Devices' Web pages at `www.analog.com`.

- Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing,* Second Edition, 1999, California Technical Publishing, P.O. Box 50240, San Diego, CA 92150.
- C. Britton Rorabaugh, *DSP Primer,* McGraw-Hill, 1999.
- Richard J. Higgins, *Digital Signal Processing in VLSI,* Prentice-Hall,1990.
- *DSP Designer's Reference (DSP Solutions)* CDROM, Analog Devices,1999.
- *DSP Navigators: Interactive Tutorials about Analog Devices' DSP Architectures (ADSP-218x family):*
- DSP Training and Workshops:

- *ADSP-2100 Family EZ-KIT Lite Reference Manual.*
- *ADSP-2100 Family DSP Applications, Vol. 1 and Vol. 2.*
- *M68300 Family CPU32 Reference Manual,* Motorola, Inc. (reference number CPU32RM/AD)
- *Modular Microcontroller Family SIM Reference Manual*, Motorola, Inc. (reference number SIMRM/AD)
- *MC68F333 User's Manua*l, Motorola, Inc. (reference number MC68F333UM/AD)
- *68F333 Development Kit User's Manual, Revision 1.00*, P&E Microcomputer Systems, Inc.