TMS320 Hardware Applications

APPLICATION REPORT: SPRA119

Jon Bradley
Digital Signal Processor Products
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software 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 application 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, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

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

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI 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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

TMS320 Hardware Applications

Abstract

This chapter discusses how to use the TMS320C30s interfaces to connect to various external devices, including the implementation of parallel interface to devices with/without wait states, and the use of general I/O.

Major topics discussed include:

- System configuration options overview
- Primary bus interface
  - Zero wait interface to RAMs
  - Ready generation
  - Bank switching techniques
- Expansion bus interface
  - A/D converter interface
  - D/A converter interface
- System control functions
  - Clock oscillator circuitry
  - Reset signal generator
- Serial port interface
  - XDS1000 target design considerations

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. New users must register with TI&ME before they can access the data sheet archive. TI&ME allows users to build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to dsph@ti.com. Questions receive prompt attention and are usually answered within one business day.

Introduction

The TMS320C30 is a high-speed, floating-point, digital signal processor. The TMS320C30s advanced interface design allows it to be used to implement a wide variety of system configurations. Its two external buses and DMA capability provide a parallel 32-bit interface to byte- or word-wide devices, while the interrupt interface, dual serial ports, and general purpose digital I/O provide communication with a multitude of peripherals.

This application report describes how to use the TMS320C30s interfaces to connect to various external devices. Specific discussions include implementation of parallel interface to devices with and without wait states, use of general purpose I/O, and system control functions. All interfaces shown in this report have been built and tested to verify proper operation.

Major topics discussed in this report are as follows:

- System Configuration Options Overview
- Primary Bus Interface
  - Zero Wait Interface to RAMs
  - Ready Generation
  - Bank Switching Techniques
- Expansion Bus Interface
  - A/D Converter Interface
  - D/A Converter Interface
- System Control Functions
  - Clock Oscillator Circuitry
  - Reset Signal Generator
- Serial Port Interface
- XDS1000 Target Design Considerations

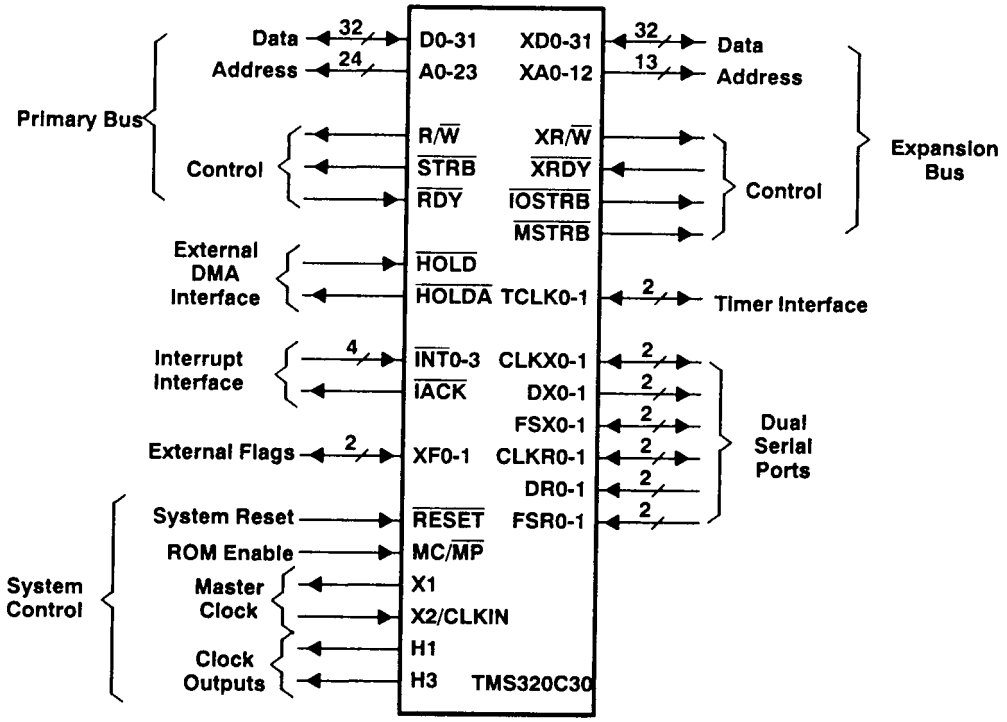
System Configuration Options Overview

The various TMS320C30 interfaces allow connections to a wide variety of different device types. Each of these interfaces is tailored to a particular family of devices.

Categories of Interfaces on the TMS320C30

The interface types on the TMS320C30 fall into several different categories depending on the devices to which they were intended to be connected. Each interface comprises one or more signal lines that transfer information and control its operation. Shown in Figure 1 are the signal line groupings for each of these various interfaces.

Figure 1. External Interfaces on the TMS320C30



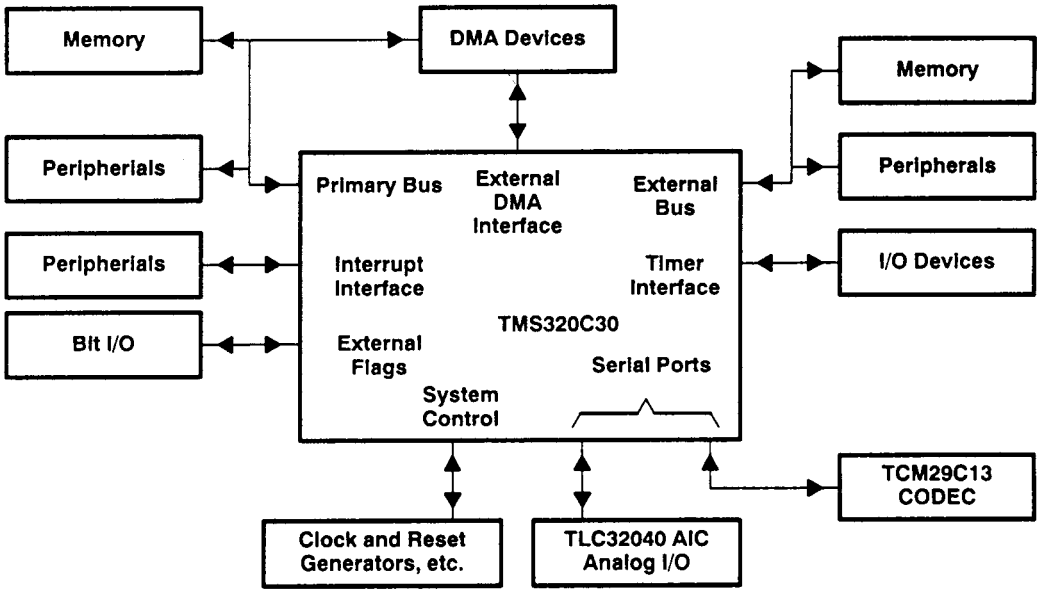
All of the interfaces are independent of one another and different operations may be performed simultaneously on each interface.

The Primary and Expansion buses implement the memory mapped interface to the device. The external DMA interface allows external devices to cause the processor to relinquish the Primary bus and allow direct memory access.

Typical System Block Diagram

The devices that can be interfaced to the TMS320C30 include memory, DMA devices, and numerous parallel and serial peripherals and I/O devices. Figure 2 illustrates a typical configuration of a TMS320C30 system showing different types of external devices and the interfaces to which they are connected.

Figure 2. Possible System Configurations



This block diagram constitutes essentially a fully expanded system. In an actual design, any subset of the illustrated configuration may be used.

Primary Bus Interface

The primary bus is used by the TMS320C30 to access the majority of its memory mapped locations. Therefore, typically when a large amount of external memory is required in a system, it is interfaced to the primary bus. The expansion bus (discussed in the next section) actually comprises two mutually exclusive interfaces, controlled by the $\overline{\text{MSTRB}}$ and $\overline{\text{IOSTRB}}$ signals respectively. Cycles on the expansion bus controlled by the $\overline{\text{MSTRB}}$ signal are essentially equivalent to cycles on the primary bus, with the exception that bank switching is not implemented on the expansion bus. Accordingly, the discussion of primary bus cycles in this section applies equally to $\overline{\text{MSTRB}}$ cycles on the expansion bus.

Although both the primary bus and the expansion bus may be used to interface to a wide variety of devices, the devices most commonly interfaced to these buses are memories. Therefore, detailed examples of memory interface will be presented in this section.

Zero Wait State Interface To Static RAMs

For full speed, zero-wait state interface to any device, the TMS320C30 requires a read access time of 30 ns from address stable to data valid. Because, for most memories, access time from chip select is the same as access time from address, it is theoretically possible to use 30 ns memories at full speed with the TMS320C30. This, however, dictates that there be no delays present between the processor and the memories. This is usually not the case in practice, due to interconnection de-

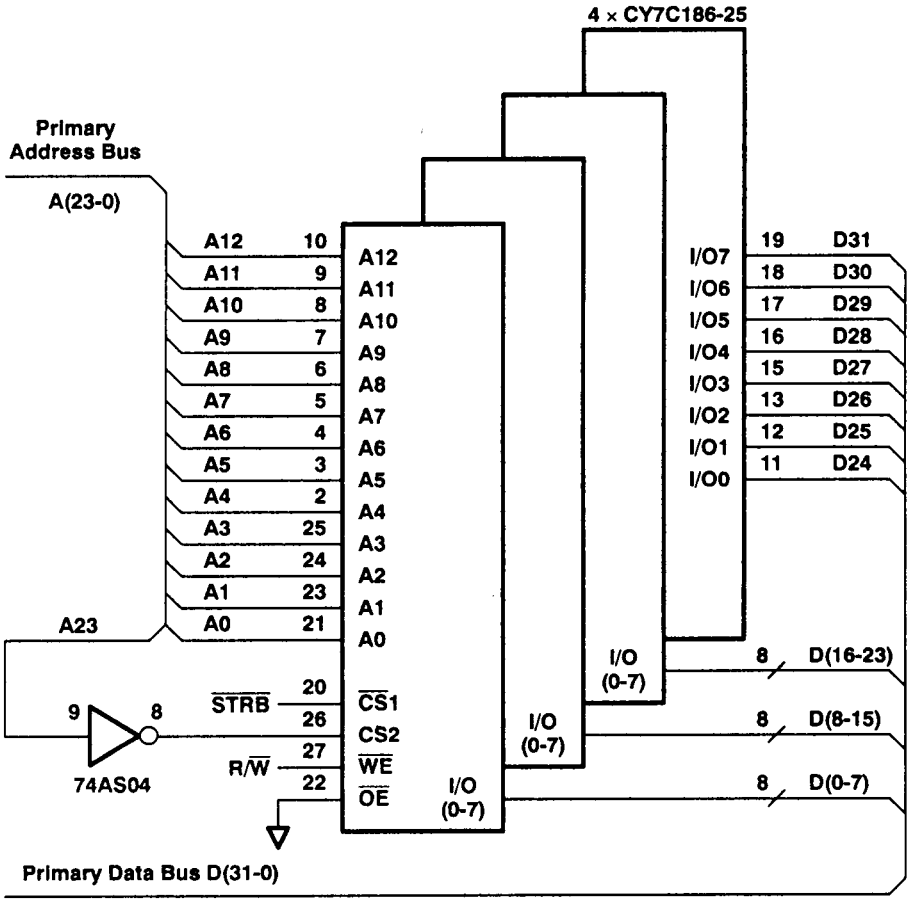
lays and the fact that typically some gating is required for chip select generation. Therefore, slightly faster memories are generally required in most systems. If one level of reasonably high-speed (below 10 ns in propagation delay) gating is used to generate chip select for the memories, 20 ns devices may be used.

Among currently available RAMs, there are two distinct categories of devices with different interface characteristics. These two categories are RAMs without output enable control lines ($\overline{OE}$), which include the 1-bit wide organized RAMs and most of the 4-bit wide RAMs, and those with $\overline{OE}$ controls, which include the byte wide and a few of the 4-bit wide RAMs. Many of the fastest RAMs do not provide $\overline{OE}$ control, and use chip select ($\overline{CS}$) controlled write cycles to insure that data outputs do not turn on for write operations. In $\overline{CS}$ controlled write cycles, the write control line ($\overline{WE}$) goes low prior to $\overline{CS}$ going low, and internal logic holds the outputs disabled until the cycle is completed. Using $\overline{CS}$ controlled write cycles is an efficient way to interface fast RAMs without $\overline{OE}$ controls to the TMS320C30 at full speed.

In the case of RAMs with $\overline{OE}$ controls, the use of this signal can provide added flexibility in many systems. Additionally, many of these devices can be interfaced using $\overline{CS}$ controlled write cycles with $\overline{OE}$ tied low, in the same manner as with RAMs without $\overline{OE}$ controls. There are, however, two requirements for interfacing to $\overline{OE}$ RAMs in this fashion. First, the RAMs $\overline{OE}$ input must be gated with chip select and $\overline{WE}$ internally so that the device's outputs do not turn on unless a read is being performed. Second, the RAM must allow its address inputs to change while $\overline{WE}$ is low, which some RAMs specifically prohibit.

The circuit shown in Figure 3 shows an interface to Cypress Semiconductor's CY7C186 25 ns 8K × 8-bit CMOS static RAMs with the $\overline{OE}$ control input tied low and using a $\overline{CS}$ controlled write cycle.

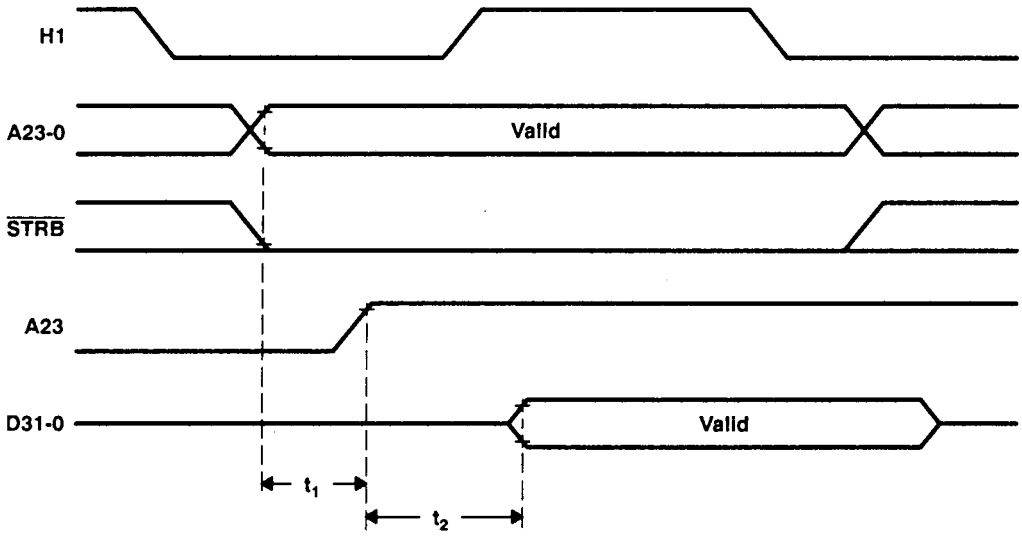
Figure 3. TMS320C30 Interface to Cypress Semiconductor CY7C186 CMOS SRAM



In this circuit, the two chip selects on the RAM are driven by $\overline{\text{STRB}}$ and $\overline{\text{A23}}$, which are ANDed together internally. The use of $\overline{\text{A23}}$ locates the RAM at addresses 00000h through 03FFFh in external memory and $\overline{\text{STRB}}$ establishes the $\overline{\text{CS}}$ controlled write cycle. The $\overline{\text{WE}}$ control input is then driven by the TMS320C30 R/$\overline{\text{W}}$ signal, and the $\overline{\text{OE}}$ input is not used, and is therefore connected to ground.

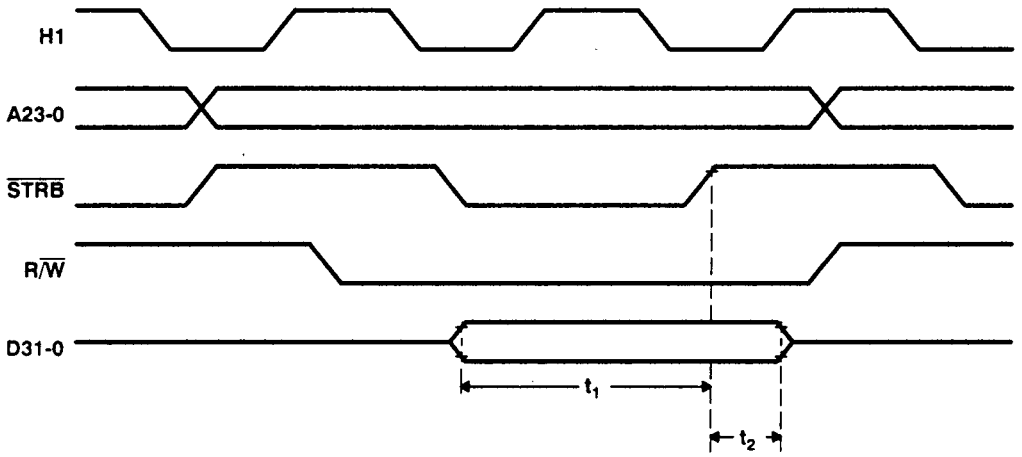
The timing of read operations, shown in Figure 4, is very straightforward since the two chip select inputs are driven directly. The read access time of the circuit is therefore the inverter propagation delay added to the RAMs chip select access time or $t_1 + t_2 = 5 + 25 = 30$ ns. This access time therefore meets the TMS320C30s specified 30 ns requirement.

Figure 4. Read Operations Timing



During write operations, as shown in Figure 5, the RAMs outputs do not turn on at all, due to the use of the chip select controlled write cycles. The chip select controlled write cycles are generated by the fact that R/$\overline{W}$ goes active (low) before the $\overline{STRB}$ term of the chip select input. Because the RAMs output drivers are disabled whenever the $\overline{WE}$ input is low (regardless of the state of the $\overline{OE}$ input) bus conflicts with the TMS320C30 are automatically avoided with this interface. The circuit's data setup and hold times (t_1 and t_2 in the timing diagram) of approximately 50 and 20 ns, respectively, also easily meet the RAMs timing requirements of 10 and 0 ns.

Figure 5. Write Operations Timing



If more complex chip select decode is required than can be accomplished in time to meet zero-wait state timing, wait states or bank switching techniques (discussed in a later section) should be used.

It should be noted that the CY7C186's $\overline{OE}$ control is gated internally with $\overline{CS}$, therefore the RAMs outputs are not enabled unless the device is selected. This is critical if there are any other devices connected to the same bus; if there are no other devices connected to the bus, then $\overline{OE}$ need not be gated internally with chip select.

RAMs without $\overline{OE}$ controls can also be easily interfaced to the TMS320C30 using a similar approach to that used with RAMs with $\overline{OE}$ controls. If there is only one bank of memory implemented, and no other devices are present on the bus, the memories' $\overline{CS}$ input may often be connected to $\overline{STRB}$ directly. If several devices must be selected, however, a gate is generally required to AND the device select and $\overline{STRB}$ to drive the $\overline{CS}$ input to generate the chip select controlled write cycles. In either case, the $\overline{WE}$ input is driven by the TMS320C30 R/$\overline{W}$ signal. Provided sufficiently fast gating is used, 25 ns RAMs may still be used.

As with the case of RAMs with $\overline{OE}$ control lines, this approach works well if only a few banks of memory are implemented where the chip select decode can be accomplished with only one level of gating. If many banks are required to implement very large memory spaces, bank switching can be used to provide for multiple bank select generation while still maintaining full speed accesses within each bank. Bank switching is discussed in detail in a later section.

Ready Generation

The use of wait states can greatly increase system flexibility and reduce hardware requirements over systems without wait state capability. The TMS320C30 has the capability of generating wait states on either the primary bus or the expansion bus and both buses have independent sets of ready control logic. Ready generation is discussed in this subsection from the perspective of the primary bus interface, however, wait state operation on the expansion bus is similar to that of the primary bus, therefore these discussions pertain equally well to expansion bus operation. Thus, ready generation will not be included in the specific discussions of the expansion bus interface.

Wait states are generated on the basis of the internal wait state generator, the external ready input ($\overline{RDY}$), or the logical AND or OR of the two. When enabled, internally generated wait states effect all external cycles, regardless of the address accessed. If different numbers of wait states are required for various external devices, the external $\overline{RDY}$ input may be used to tailor wait state generation to specific system requirements.

If the logical OR (or electrical AND since the signals are true low) of the external and wait count ready signals is selected, the earlier of either of the two signals will generate a ready condition and allow the cycle to be completed. It is not required that both signals be present.

The OR of the two ready signals can be used to implement wait states for devices that require a greater number of wait states than are implemented with external logic (up to seven). This feature is useful, for example, if a system contains some fast and some slow devices. In this case, fast devices can generate a ready signal externally with a minimum of logic, and slow devices can use the internal wait counter for larger numbers of wait states. Thus, when fast devices are accessed, the external hardware responds promptly with a ready signal that terminates the cycle. When slow devices are accessed, the external hardware does not respond, and the cycle is appropriately terminated after the internal wait count.

The OR of the two ready signals may also be used if conditions occur that require termination of bus cycles prior to the number of wait states implemented with external logic. In this case, a

shorter wait count is specified internally than the number of wait states implemented with the external ready logic, and the bus cycle is terminated after the wait count. This feature may also be used as a safeguard against inadvertent accesses to nonexistent memory that would never respond with ready and therefore lock up the TMS320C30.

If the OR of the two ready signals is used, however, and the internal wait state count is less than the number of wait states implemented externally, the external ready generation logic must have the ability to reset its sequencing to allow a new cycle to begin immediately following the end of the internal wait count. This requires that, under these conditions, consecutive cycles must be from independently decoded areas of memory and that the external ready generation logic be capable of restarting its sequence as soon as a new cycle begins. Otherwise, the external ready generation logic may lose synchronization with bus cycles and therefore generate improperly timed wait states.

If the logical AND (electrical OR) of the wait count and external ready signals is selected, the later of the two signals will control the internal ready signal, and both signals must occur. Accordingly, external ready control must be implemented for each wait state device in addition to the wait count ready signal being enabled.

This feature is useful if there are devices in a system that are equipped to provide a ready signal but cannot respond quickly enough to meet the TMS320C30s timing requirements. In particular, if these devices normally indicate a ready condition and, when accessed, respond with a wait until they become ready, the logical AND of the two ready signals can be used to save hardware in the system. In this case, the internal wait counter can be used to provide wait states initially, and become ready after the external device has had time to send a not ready indication. The internal wait counter then remains ready until the external device also becomes ready, which terminates the cycle.

Additionally, the AND of the two ready signals may be used for extending the number of wait states for devices that already have external ready logic implemented but require additional wait states under certain unique circumstances.

In the implementation of external ready generation hardware, the particular technique employed depends heavily on the specific characteristics of the system. The optimum approach to ready generation varies depending on the relative number of wait state and non-wait state devices in the system and the maximum number of wait states required for any one device. The approaches discussed here are intended to be general enough for most applications, and are easily modifiable to comprehend many different system configurations.

In general, ready generation involves the following three functions:

1) Segmentation of the address space in some fashion to distinguish fast and slow devices.
2) Generating properly timed ready indications.
3) Logically ORing all of the separate ready timing signals together to connect to the physical ready input.

Segmentation of the address space is required so that a unique indication of each of the particular areas within the address space that require wait states can be obtained. This segmentation is commonly implemented in a system in the form of chip select generation. Chip select signals may be used to initiate wait states in many cases, however, occasionally chip select decoding considerations may provide signals that will not allow ready input timing requirements to be met. In this case, coarse address space segmentation may be made on the basis of a small number of address lines, where simpler gating allows signals to be generated more quickly. In either case, the signal indicating that a particular area of memory is being addressed is normally used to initiate a ready or wait state indication.

Once the region of address space being accessed has been established, a timing circuit of some sort is normally used to provide a ready indication to the processor at the appropriate point in the cycle to satisfy each device's unique requirements.

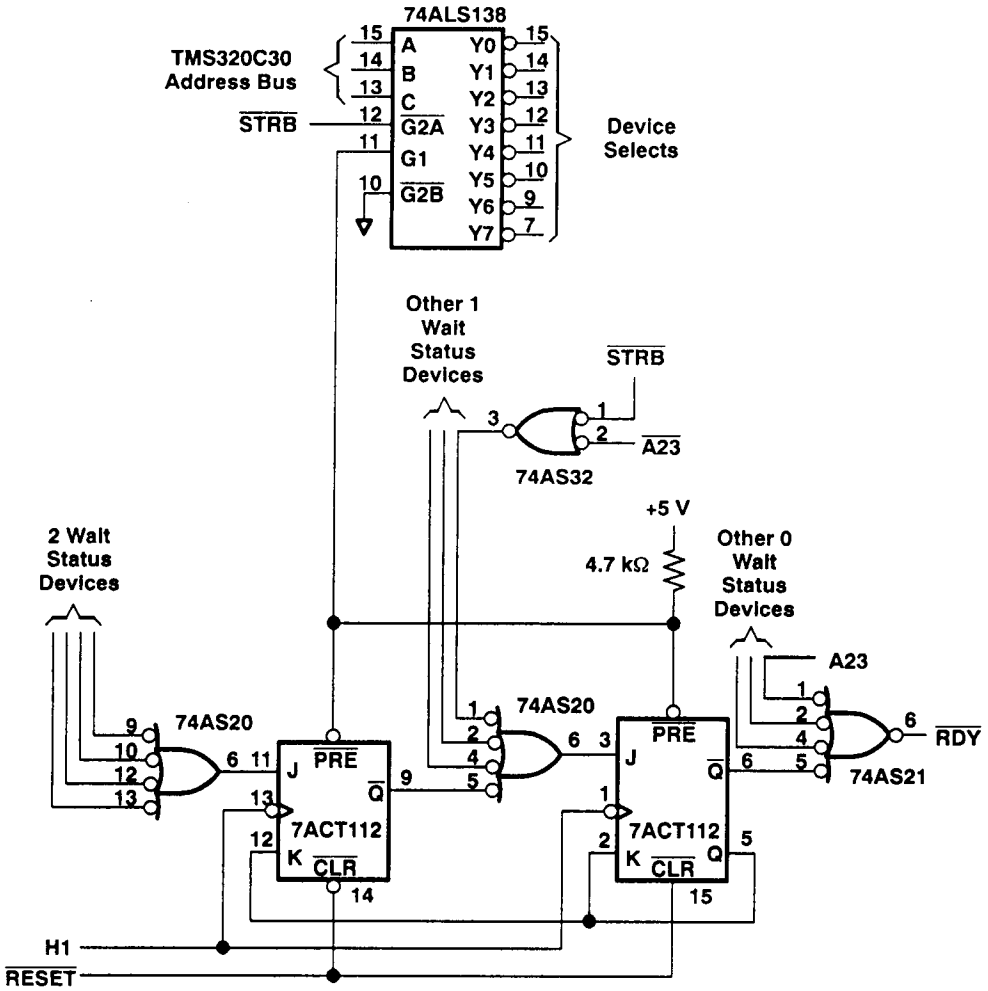
Finally, since indications of ready status from multiple devices are typically present, the signals are logically ORed using a single gate to drive the $\overline{\text{RDY}}$ input.

One of two basic approaches may be taken in the implementation of ready control logic depending upon the state in which the ready input is to be between accesses. If $\overline{\text{RDY}}$ is low between accesses, the processor is always ready unless a wait state is required; if $\overline{\text{RDY}}$ is high between accesses, the processor will always enter a wait state unless a ready indication is generated.

If $\overline{\text{RDY}}$ is low between accesses, control of full speed devices is straightforward; no action is necessary since ready is always active unless otherwise required. Devices requiring wait states, however, must drive ready high fast enough to meet the input timing requirements. Then, after an appropriate delay, a ready indication must be generated. This can be quite difficult in many circumstances since wait state devices are inherently slow and often require complex select decoding.

If $\overline{\text{RDY}}$ is high between accesses, zero wait state devices, which tend to be inherently fast, can usually respond immediately with a ready indication. Wait state devices may simply delay their select signals appropriately to generate a ready. Typically, this approach results in the most efficient implementation of ready control logic. Figure 6 shows a circuit of this type which can be used to generate 0, 1, or 2 wait states for multiple devices in a system.

Figure 6. Circuit For Generation of 0, 1, or 2 Wait States for Multiple Devices



In this circuit, full speed devices drive ready directly through the '74AS21, and the two flip-flops delay wait state devices' select signals one or two H1 cycles to provide 1 or 2 wait states.

Considering the TMS320C30's ready delay time of 8 ns following address, zero wait state devices must use ungated address lines directly to drive the input of the '74AS21, since this gate contributes a maximum propagation delay of 6 ns to the $\overline{RDY}$ signal. Thus, zero wait state devices should be grouped together within a coarse segmentation of address space if other devices in the system require wait states.

With this circuit, devices requiring wait states may take up to 36 ns from a valid address on the TMS320C30 to provide inputs to the '74AS20s inputs. Typically, this allows sufficient time for any decoding required in generating select signals for slower devices in the system. For exam-

ple, the 74ALS138 driven by address and $\overline{\text{STRB}}$, can generate select decodes in 22 ns, which easily meets the TMS320C30s timing requirements.

With this circuit, unused inputs to either the 74AS20s or the 74AS21 should be tied to a logic high level to prevent noise from generating spurious wait states.

If more than 2 wait states are required by devices within a system, other approaches may be employed for ready generation. If between three and seven wait states are required, additional flip-flops may be included, in the same manner as shown in Figure 6, or internally generated wait states may be used in conjunction with external hardware. If greater than seven wait states are required, an external circuit using a counter may be used to supplement the internal wait–state generator's capabilities.

Bank Switching Techniques

The TMS320C30's programmable bank switching feature can greatly ease system design when large amounts of memory are required. This feature is used to provide a period of time during which all device selects are disabled that would not normally be present otherwise. During this interval, slow devices are allowed time to turn off before other devices have the opportunity to drive the data bus, thus avoiding bus contention.

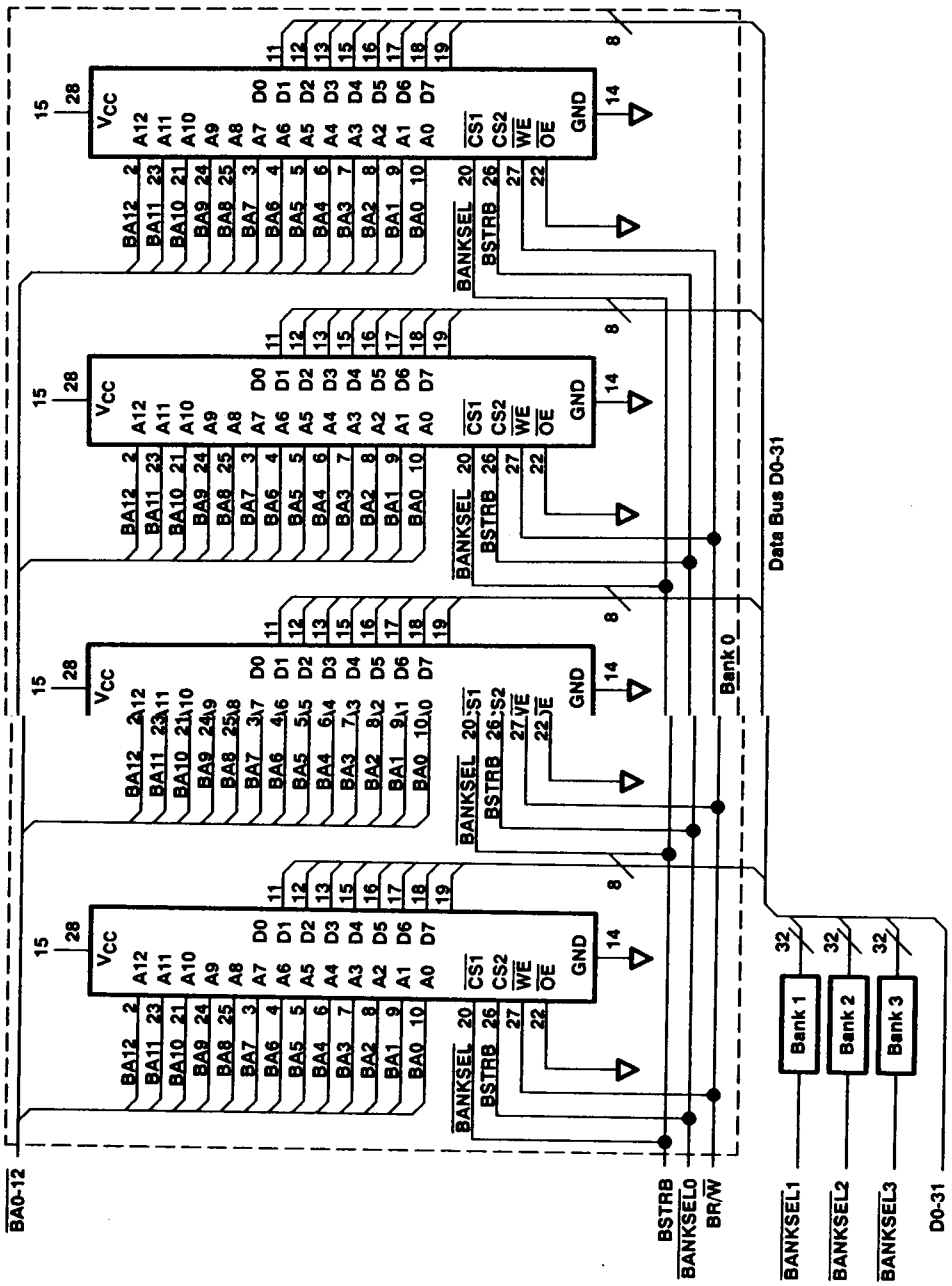
When bank switching is enabled, any time a portion of the high order address lines change, as defined by the contents of the BNKCMPR register, $\overline{\text{STRB}}$ goes high for one full H1 cycle. Provided $\overline{\text{STRB}}$ is included in chip select decodes, this causes all devices to be disabled during this period. The next bank of devices is not enabled until $\overline{\text{STRB}}$ goes low again.

Bank switching is not required during writes since these cycles always exhibit an inherent one-half H1 cycle setup of address information before $\overline{\text{STRB}}$ goes low. Thus, when using bank switching for read/write devices, a minimum of half of one H1 cycle of address setup is provided for all accesses. Therefore, large amounts of memory can be implemented without wait states or extra hardware required for isolation between banks. Also, note that access time for cycles during bank switching is the same as that of cycles without bank switching, and accordingly, full speed accesses may still be accomplished within each bank.

When using bank switching to implement large multiple-bank memory systems, an important consideration is address line fanout. Besides parametric specifications for which account must be made, AC characteristics are also crucial in memory system design. With large memory arrays which commonly require large numbers of address line inputs to be driven in parallel, capacitive loading of address outputs is often quite large. Because all TMS320C30 timing specifications are guaranteed up to a capacitive load of 80 pF, driving greater loads will invalidate guaranteed AC characteristics. Therefore it is often necessary to provide buffering for address lines when driving large memory arrays. AC timings for buffer performance may then be derated according to manufacturer specifications to accomodate a wide variety of memory array sizes.

The circuit shown in Figure 7 illustrates the use of bank switching with Cypress Semiconductor's 'CY7C185 25 ns 8K × 8 CMOS static RAM. This circuit implements 32K 32-bit words of memory with one wait-state accesses within each bank.

Figure 7. Bank Switching r Cypress Semiconductors CY7C185

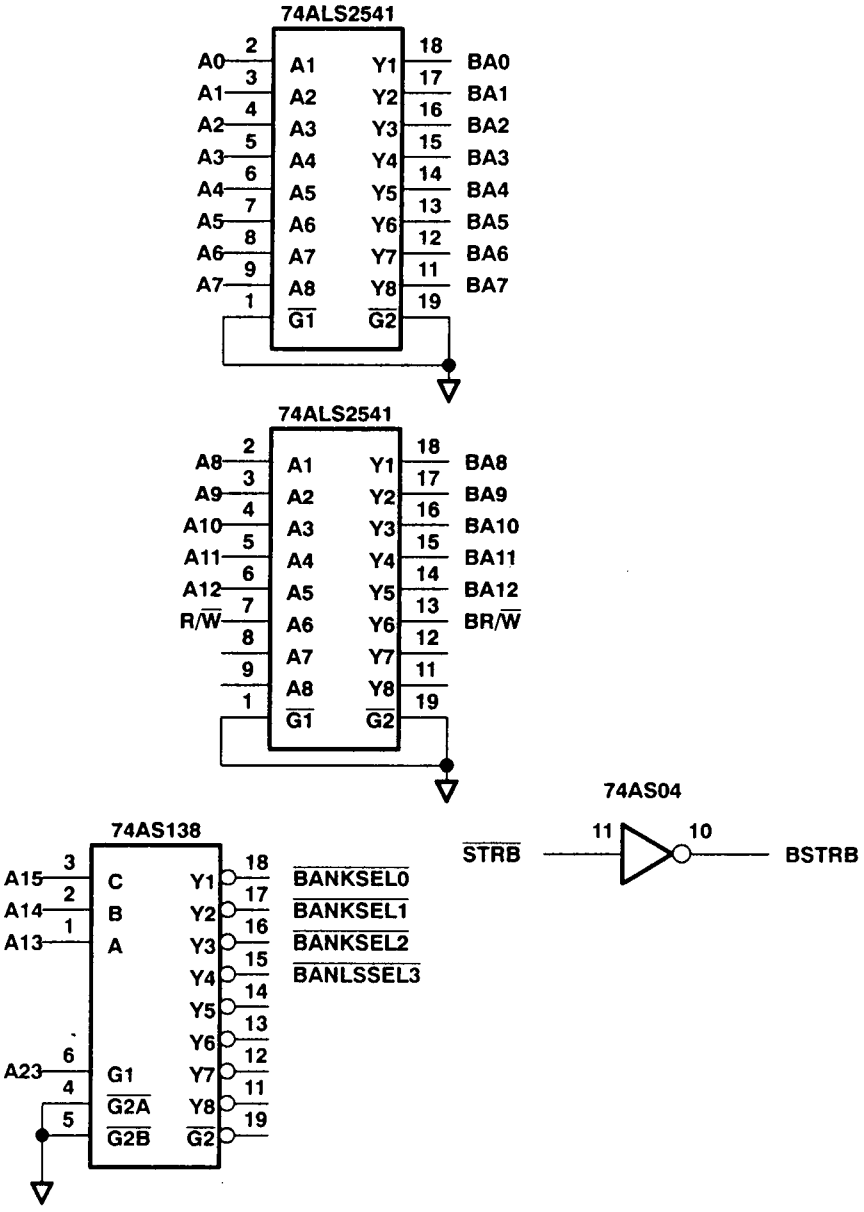


A wait state is required with this implementation of bank memory because of the added propagation delay presented by the address bus buffers used in the circuit. The wait state is not a function of the fact that the memory is organized as multiple banks or the use of bank switching. When bank switching is used, memory access speeds are the same as without bank switching once bank boundaries are crossed. Therefore, no speed penalty is paid when using bank switching except for the occasional extra cycle inserted when bank boundaries are crossed. It should be noted, however, that if the extra cycle inserted when crossing bank boundaries does impact software performance significantly, code can often be restructured to minimize bank boundary crossings, thereby reducing the effect of these boundary crossings on software performance.

The wait state for this bank memory is generated using the wait state generator circuit presented in the previous section. Because A23 is the signal which enables the entire bank memory system, the inverted version of this signal is ANDed with $\overline{\text{STRB}}$ to derive a one wait state device select. This signal is then connected in the circuit along with the other one wait state device selects. Thus, any time a bank memory access is made, one wait state is generated.

Each of the four banks in this circuit is selected using a decode of A15-A13 generated by the 74AS138 (see Figure 8). With the BNKCMPR register set to 0Bh, the banks will be selected on even 8K-word boundaries starting at location 080A000h in external memory space.

Figure 8. Bank Memory Control Logic



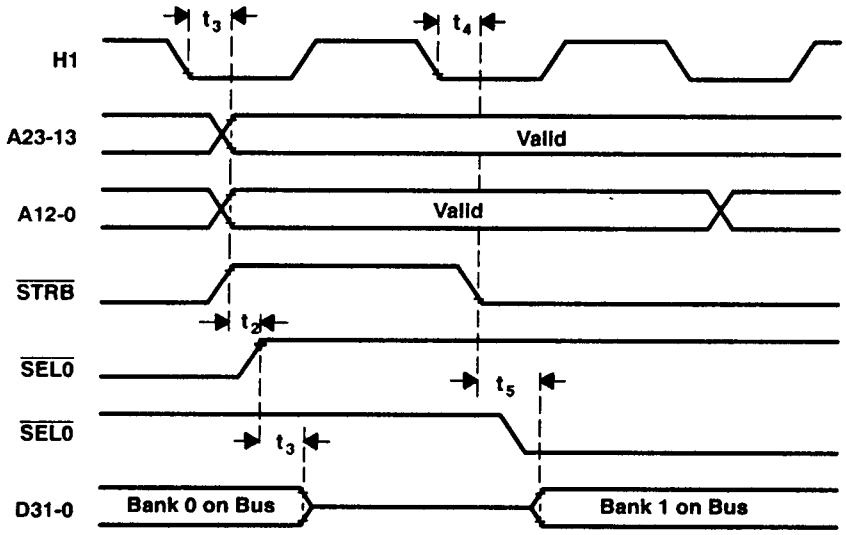
The 74ALS2541 buffers used on the address lines are necessary in this design since the total capacitive load presented to each address line is a maximum of 20 × 5 pF or 100 pF (bank memory plus zero wait–state static RAM), which exceeds the TMS320C30 rated capacitive loading of 80 pF. Using the manufacturers derating curves for these devices at a load of 80 pF (the load presented by the bank memory) predicts propagation delays at the output of the buffers of a maximum of 16 ns. The access time of a read cycle within a bank of the memory is therefore the sum of the memory access time and the maximum buffer propagation delay or 25 + 16 = 41 ns, which, since it falls between 30 and 90 ns, requires one wait state on the TMS320C30.

The 74ALS2541 buffers offer one additional system performance enhancement in that they include 25-ohm resistors in series with each individual buffer output. These resistors greatly improve the transient response characteristics of the buffers especially when driving CMOS loads such as the memories used here. The effect of these resistors is to reduce overshoot and ringing which is common when driving predominantly capacitive loads such as CMOS. The result of this is reduced noise and increased immunity to latchup in the circuit, which in turn results in a more reliable memory system. Having these resistors included in the buffers eliminates the need to put discrete resistors in the system which is often required in high speed memory systems.

This circuit could not have been implemented without bank switching, since data output's turn-on and turn-off delays would have caused bus conflicts. Here, the propagation delay of the 74AS138 is only involved during bank switches, where there is sufficient time between cycles to allow new chip selects to be decoded.

The timing of this circuit for read operations using bank switching is shown in Figure 9. With the BNKCMPR register set to 0Bh, when a bank switch occurs, the bank address on address lines A23-A13, is updated during the extra H1 cycle while $\overline{\text{STRB}}$ is high. Then, after chip select decodes have stabilized, and the previously selected bank has disabled its outputs, $\overline{\text{STRB}}$ goes low for the next read cycle. Further accesses occur at normal bus timings with one wait state as long as another bank switch is not necessary. Write cycles do not require bank switching due to the inherent address setup provided in their timings.

Figure 9. Timing For Read Operations Using Bank Switching



The timing for this interface is summarized in the Table 1.

Table 1. Bank Switching Interface Timing

Time Interval	Event	Time Period
t_1	H1 falling to address/STRB valid	14 ns
t_2	Add to select delay	10 ns
t_3	Memory disable from STRB	10 ns
t_4	H1 falling to STRB	10 ns
t_6	Memory output enable delay	3 ns

Expansion Bus Interface

The TMS320C30s expansion bus interface provides a second complete parallel bus which can be used to implement data transfers concurrently with and independent of operations on the primary bus. The expansion bus comprises two mutually exclusive interfaces controlled by the $\overline{\text{MSTRB}}$ and $\overline{\text{IOSTRB}}$ signals, respectively. This section discusses interface to the expansion bus using $\overline{\text{IOSTRB}}$ cycles; $\overline{\text{MSTRB}}$ cycles are essentially equivalent in timing to primary bus cycles, and are discussed in the previous section.

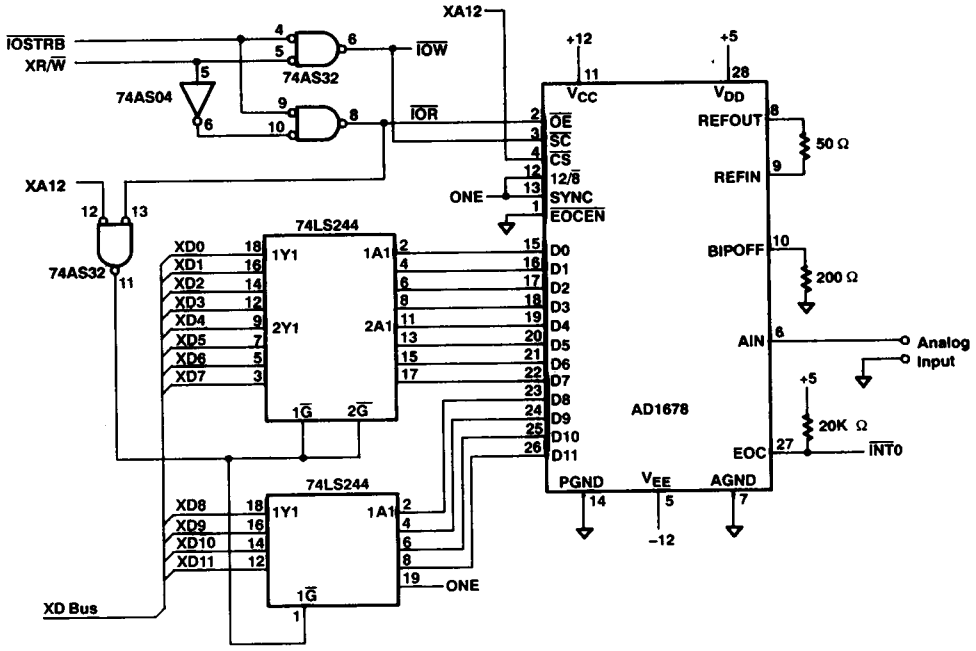
Unlike the primary bus, both read and write cycles on the I/O portion of the expansion bus are two H1 cycles in duration and exhibit the same timing. The XR/$\overline{\text{W}}$ signal is high for reads and low for writes. Since I/O accesses take two cycles, many peripherals that require wait states if interfaced either to the primary bus or using $\overline{\text{MSTRB}}$ may be used in a system without the need for wait states. Specifically, in cases where there is only one device on the expansion bus, devices with access times greater than the 30 ns required by the primary bus, but not more than 59 ns can be interfaced to the I/O bus without wait states.

A/D Converter Interface

A/D and D/A converters are components that are commonly required in DSP systems and interface efficiently to the I/O expansion bus. These devices are available in many speed ranges and with a variety of features, and while some may be used at full speed on the I/O bus, others may require one or more wait states.

Figure 10 shows an interface to an Analog Devices AD1678 analog to digital converter. The AD1678 is a 12-bit, 5 μs converter allowing sample rates up to 200 kHz and with an input voltage range of 10 volts bipolar or unipolar. The converter is connected according to manufacturers specifications to provide 0 to +10 volt operation. This interface illustrates a common approach to connecting devices such as this to the TMS320C30. Note that the interface requires only a minimum amount of control logic.

Figure 10. Interface to AD1678 A/D Converter



The AD1678 is a very flexible converter and is configurable in a number of different operating modes. These operating modes include byte or word data format, continuous or non-continuous conversions, enabled or disabled chip select function, and programmable end of conversion indication. This interface utilizes 12-bit word data format, rather than byte format to be compatible with the TMS320C30. Non-continuous conversions are selected, so that variable sample rates may be used, since continuous conversions occur only at a rate of 200 kHz. With non-continuous conversions, the host processor determines the conversion rate by initiating conversions through write operations to the converter.

The chip select function is enabled, so the chip select input is required to be active when accessing the device. Enabling the chip select function is necessary to allow a mechanism for the AD1678 to be isolated from other peripheral devices connected to the expansion bus. To establish the desired operating modes, the SYNC and 12/$\overline{8}$ inputs to the converter are pulled high and $\overline{EOCEN}$ is grounded, as specified in the AD1678 data sheet.

In this application, the converter's chip select is driven by XA12, which maps this device at 804000h in I/O address space. Conversions are initiated by writing any data value to the device, and the conversion results are obtained by reading from the device after the conversion is completed. To generate the devices Start Conversion ($\overline{SC}$) and Output Enable ($\overline{OE}$) inputs, $\overline{IOSTRB}$ is ANDed with XR/$\overline{W}$. Therefore, the converter is selected whenever XA12 is low, and $\overline{OE}$ is driven when reads are performed, while SC is driven when writes are performed.

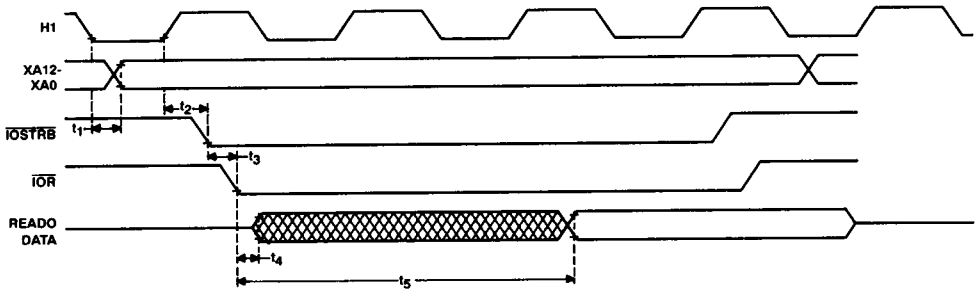
As with many A/D converters, at the end of a read cycle the AD1678 data output lines enter a high impedance state. This occurs after the Output Enable ($\overline{OE}$) or read control line goes inactive. Also common with these types of devices, is that the data output buffers often require a substantial amount of time to actually attain a full high-impedance state. When used with the TMS320C30, devices must have their outputs fully disabled no later than 65 ns following the rising edge of $\overline{IOSTRB}$, since the TMS320C30 will begin driving the data bus at this point if the next cycle is a write. If this timing is not met, bus conflicts between the TMS320C30 and the AD1678 may occur, potentially causing degraded system performance and even failure due to damaged data bus drivers. The actual disable time for the AD1678 can be as long as 80 ns, therefore buffers are required to isolate the converter outputs from the TMS320C30. The buffers used here are 74LS244s that are enabled when the AD1678 is read, and turned off 30.8 ns following $\overline{IOSTRB}$ going high.Therefore, the TMS320C30 requirement of 65 ns is met.

When data is read following a conversion, the AD1678 takes 100 ns after its $\overline{OE}$ control line is asserted to provide valid data at its outputs. Thus, including the propagation delay of the 74LS244 buffers, the total access time for reading the converter is 118 ns. This requires two wait states on the TMS320C30 expansion I/O bus.

The two wait states required in this case are implemented using software wait states, however, depending on the overall system configuration it may be necessary to implement a separate wait state generator for the expansion bus (refer to section on ready generation). This would be the case if there were multiple devices that required different numbers of wait states connected to the expansion bus.

Figure 11 shows the timing for read operations between the TMS320C30 and the AD1678. At the beginning of the cycle, the address and XR/$\overline{W}$ lines become valid t_1 = 10 ns following the falling edge of H_1. Then, after t_2 = 10 ns from the next rising edge of H_1, $\overline{IOSTRB}$ goes low, beginning the active portion of the read cycle. After t_3 = 5.8 ns, the control logic propagation delay, the $\overline{IOR}$ signal goes low, asserting the $\overline{OE}$ input to the AD1678. The '74LS244 buffers take t_4 = 30 ns to enable their outputs, and then, following the converters access delay and the buffer propagation delay (t_5 = 100 + 18 = 118 ns) data is provided to the TMS320C30. This provides approximately 46 ns of data setup before the rising edge of $\overline{IOSTRB}$. Therefore, this design easily satisfies the TMS320C30s requirement of 15 ns of data setup time for reads.

Figure 11. Read Operations Timing Between the TMS320C30 and AD1678



Unlike the primary bus, read and write cycles on the I/O expansion bus are timed the same with the exception that XR/$\overline{W}$ is high for reads and low for writes and that the data bus is driven by the TMS320C30 during writes. When writing to the AD1678, the '74LS244 buffers do not turn on and no data is transferred. The purpose of writing to the converter is only to generate a pulse on the converter's $\overline{SC}$ input, which initiates a conversion cycle. When a conversion cycle is completed, the AD1678's EOC output is used to generate an interrupt on the TMS320C30 to indicate that the converted data may be read.

It should be noted that for different applications, use of TLC1225 or TLC1550 A/D converters from Texas Instruments may be beneficial. The TLC1225 is a self-calibrating 12-bit-plus-sign bipolar or unipolar converter which features 10 μs conversion times. The TLC1550 is a 10-bit, 6 μs converter with a high speed DSP interface. Both converters are parallel-interface devices.

D/A Converter Interface

In many DSP systems, the requirement for generating an analog output signal is a natural consequence of sampling an analog waveform with an A/D converter and then processing the signal digitally internally. Interfacing D/A converters to the the TMS320C30 on the expansion I/O bus is also quite straightforward.

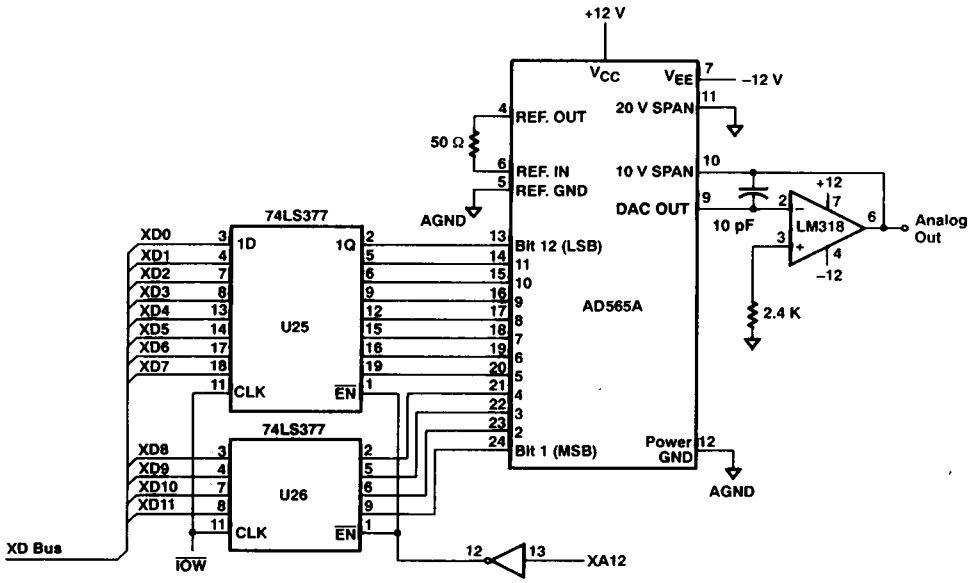
As with A/D converters, D/A converters are also available in a number of varieties. One of the major distinctions between various types of D/A converters is whether or not the converter includes latches to store the digital value to be converted to an analog quantity, and the interface to control those latches. With latches and control logic included with the converter, interface design is often simplified, however, internal latches are often included only in slower D/A converters.

Because slower converters limit signal bandwidths, the converter chosen for this design was selected to allow a reasonably wide range of signal frequencies to be processed, in addition to illustrating the technique of interfacing to a converter using external data latches.

Figure 12 shows an interface to an Analog Devices AD565A digital to analog converter. This device is a 12-bit, 250 ns current output DAC with an on-board 10 volt reference. Using an off-board current-to-voltage conversion circuit connected according to manufacturers specifications,

the converter exhibits output signal ranges 0 to +10 volts, which is compatible with the conversion range of the A/D converter discussed in the previous section.

Figure 12. Interface Between the TMS320C30 and the AD565A

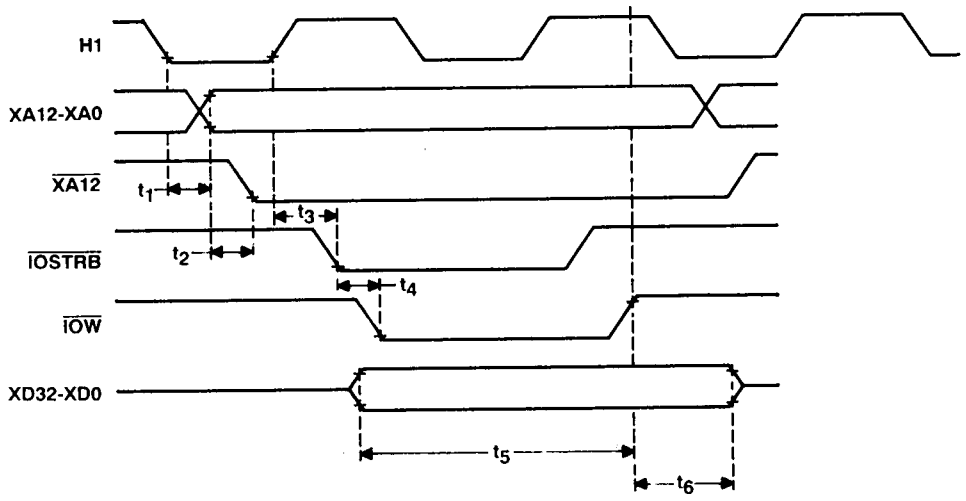


Because this DAC essentially performs continuous conversions based on the digital value provided at its inputs, periodic sampling is maintained by periodically updating the value stored in the external latches. Therefore, between sample updates, the digital value is stored and maintained at the latch outputs that provide the input to the DAC. This results in the analog output remaining stable until the next sample update is performed.

The external data latches used in this interface are '74LS377 devices that have both clock and enable inputs. These latches serve as a convenient interface with the TMS320C30; the enable inputs provide a device select function, and the clock inputs latch the data. Therefore, with the enable input driven by inverted XA12 and the clock input driven by $\overline{\text{IOW}}$, which is the AND of $\overline{\text{IOSTRB}}$ and XR/$\overline{\text{W}}$, data will be stored in the latches when a write is performed to I/O address 805000h. Reading this address has no effect on the circuit.

Figure 13 shows a timing diagram of a write operation to the D/A converter latches.

Figure 13. Write Operation to the D/A Converter Timing Diagram



Because the write is actually being performed to the latches, the key timings for this operation are the timing requirements for these devices. For proper operation, these latches require simply a minimal setup and hold time of data and control signals with respect to the rising edge of the clock input. Specifically, the latches require a data setup time of 20 ns, enable setup of 25 ns, disable setup of 10 ns and data and enable hold times of 5 ns. This design provides approximately 60 ns of enable setup, 30 ns of data setup, and 7.2 ns of data hold time. Therefore, the setup and hold times provided by this design are well in excess of those required by the latches. The key timing parameters for this interface are summarized in Table 2.

Table 2. Key Timing Parameter for D/A Converter Write Operation

Time Interval	Event	Time Period
t_1	H1 falling to address valid	10 ns
t_2	XA12 to XA12 delay	5 ns
t_3	H1 rising to IOSTRB falling	10 ns
t_4	IOSTRB to IOW delay	5.8 ns
t_5	Data setup to IOW	30 ns
t_6	Data hold from IOW	7.2 ns

System Control Functions

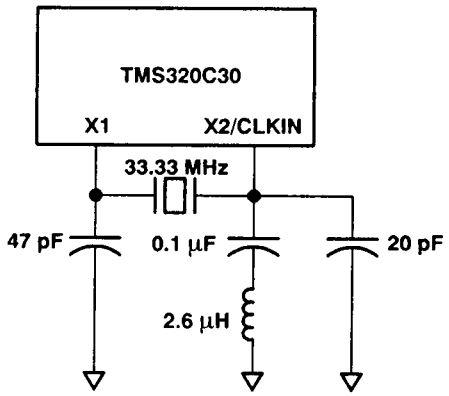
There are several aspects of TMS320C30 system hardware design that are critical to overall system operation. These include such functions as clock and reset signal generation and interrupt control.

Clock Oscillator Circuitry

An input clock may be provided to the TMS320C30 either from an external clock input or by using the on-board oscillator. Unless special clock requirements exist, using the on-board oscillator is generally a convenient method of clock generation. This method requires few external components and can provide stable, reliable clock generation for the device.

Figure 14 shows a clock generator circuit using the internal oscillator. This circuit is designed to operate at 33.33 MHz and since crystals with fundamental oscillation frequencies of 30 MHz and above are not readily available, a parallel-resonant third-overtone circuit is used.

Figure 14. Crystal Oscillator Circuit



In a third-overtone oscillator, the crystal fundamental frequency must be attenuated so that oscillation is at the third harmonic. This is achieved with an LC circuit that filters out the fundamental, thus allowing oscillation at the third harmonic. The impedance of the LC circuit must be inductive at the crystal fundamental and capacitive at the third harmonic. The impedance of the LC circuit is given by:

$$z\ (\omega) = \frac{L/C}{j\ [\omega_L - 1/\omega C\ ]} \tag{1}$$

Therefore, the LC circuit has a pole at:

$$\omega_p = \frac{1}{\sqrt{LC}} \tag{2}$$

At frequencies significantly lower than ω_p, the $1/(\omega C)$ term in (1) becomes the dominating term, while ω_L can be neglected. This gives:

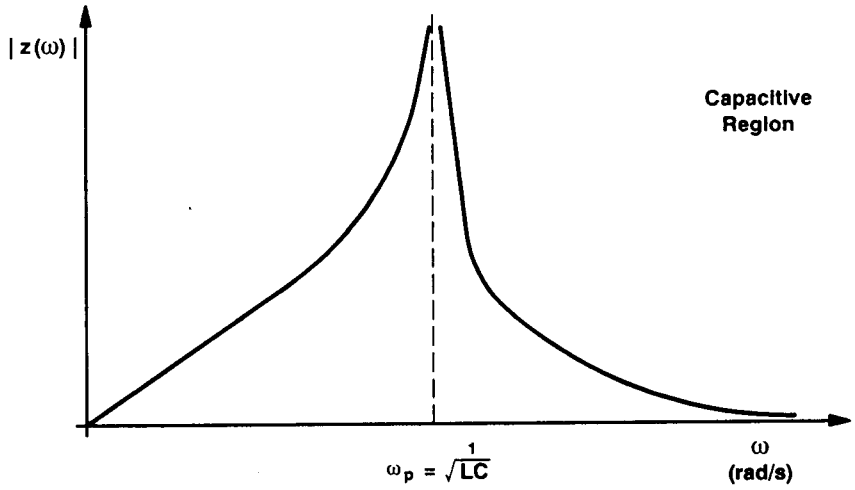
$$z\ (\omega) = j\omega L \text{ for } \omega < \omega_p \tag{3}$$

In (3), the LC circuit appears inductive at frequencies lower than ω_p. On the other hand, at frequencies much higher than ω_p, the ωL term is the dominant term in (1), and $1/(\omega C)$ can be neglected. This gives:

$$z(\omega) = \frac{1}{j\omega C} \quad \text{for } \omega > \omega_p \tag{4}$$

The LC circuit in (4) appears increasingly capacitive as frequency increases above ω_p. This is shown in Figure 15, which is a plot of the magnitude of the impedance of the LC circuit of Figure 14 versus frequency.

Figure 15. Magnitude of the Impedance of the Oscillator LC Network



Based on the discussion above, the design of the LC circuit proceeds as follows:

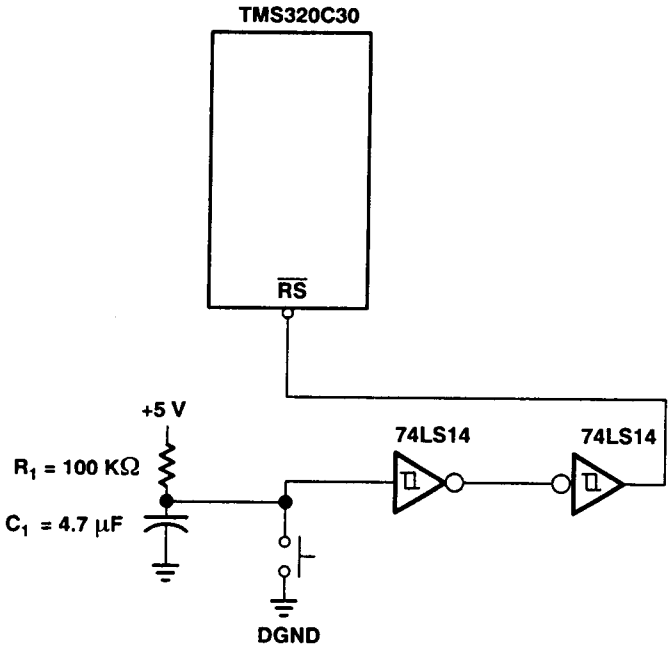
1) Choose the pole frequency ω_p approximately halfway between the crystal fundamental and the third harmonic.
2) The circuit now appears inductive at the fundamental frequency and capacitive at the third harmonic.

In the oscillator of Figure 13, choose ω_p = 22.2 MHz, which is approximately halfway between the fundamental and the third harmonic. Choose C = 20 pF. Then, using (2), L = 2.6 μH.

Reset Signal Generation

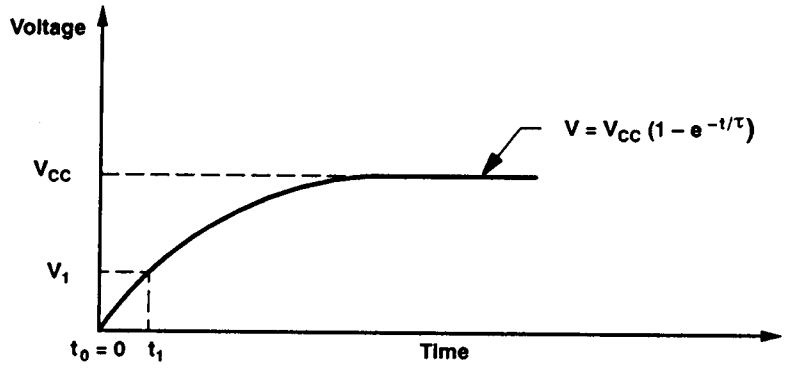
The reset input controls initialization of internal TMS320C30 logic and also causes execution of the system initialization software. For proper system initialization, the reset signal must be applied at least ten H1 cycles, i.e., 600 ns for a TMS320C30 operating at 33.33 MHz. Upon powerup, however, it can take 20 ms or more before the system oscillator reaches a stable operating state. Therefore, the powerup reset circuit should generate a low pulse on the reset line for 100 to 200 ms. Once a proper reset pulse has been applied, the processor fetches the reset vector from location zero which contains the address of the system initialization routine. Figure 16 shows a circuit that will generate an appropiate powerup reset circuit.

Figure 16. Reset Circuit



The voltage on the reset pin ($\overline{\text{RESET}}$) is controlled by the R_1C_1 network. After a reset, this voltage rises exponentially according to the time constant R_1C_1, as shown in Figure 17.

Figure 17. Voltage on the TMS320C30 Reset Pin.



The duration of the low pulse on the reset pin is approximately t_1, which is the time it takes for the capacitor C_1 to be charged to 1.5 V. This is approximately the voltage at which the reset input switches from a logic 0 to a logic 1. The capacitor voltage is given by:

$$V = V_{CC} \left[1 - e^{-\frac{t}{\tau}} \right] \quad (5)$$

where $\tau = R_1C_1$ is the reset circuit time constant. Solving (5) for t gives:

$$t = -R_1C_1 \ln \left[1 - \frac{V}{V_{CC}}\right] \quad (6)$$

Setting the following:

$R_1 = 100\ k\Omega$
$C_1 = 4.7\ \mu F$
$V_{CC} = 5\ V$
$V = V_1 = 1.5\ V$

gives t = 167 ms. Therefore, the reset circuit of Figure 16 provides a low pulse of long enough duration to ensure the stabilization of the system oscillator.

Note that if synchronization of multiple TMS320C30s is required, all processors should be provided with the same input clcock and the same reset signal. After powerup, when the clock has stabilized, all processors may then be synchronized by generating a falling edge on the common reset signal. Because it is in the falling edge of reset that establishes synchronization, reset must be high for a period of time (at least ten H1 cycles) initially. Following the falling edge, reset should remain low for at least ten H1 cycles and then be driven high. This sequencing of reset may be accomplished using additional circuitry, based on either RC time delays or counters.

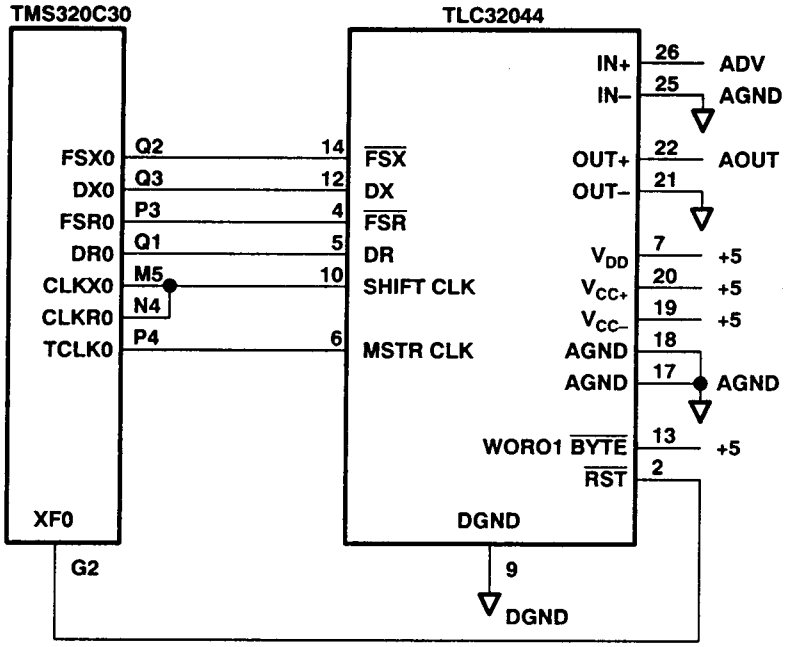
Serial Port Interface to AIC

For applications such as modems, speech, control, instrumentation, and analog interface for DSPs, a complete analog-to-digital (A/D) and digital-to-analog (D/A) input/output system on a single chip may be desired. The TLC32044 analog interface circuit (AIC) integrates on a single monlithic/CMOS chip a bandpass, switched-capacitor, antialiasing-input filter, 14-bit resolution A/D and D/A converters, and a lowpass, switched-capacitor, output-reconstruction filter. The TLC32044 offers numerous combinations of master clock input frequencies and conversion/sampling rates, which can be changed via digital processor control.

Four serial port modes on the TLC32044 allow direct interface to TMS320C30 processors. When the transmit and receive sections of the AIC are operating synchronously, it can interface to two SN54299 or SN74299 serial-to-parallel shift registers. These shift registers can then interface in parallel to the TMS320C30, other TMS320 digital processors, or to external FIFO circuitry. Output data pulses are emitted to inform the processor that data transmission is complete or to allow the DSP to differentiate between two transmitted bytes. A flexible control scheme is provided so that the functions of the AIC can be selected and adjusted coincidentally with signal processing via software control. Refer to the TLC32044 data sheet for detailed information.

When interfacing the AIC to the TMS320C30 via one of the serial ports, no additional logic is required. This interface is shown in Figure 18. The serial data, control and clock signals connect directly between the two devices and the AIC's master clock input is driven from TCLK0, one of the TMS320C30s internal timer outputs. The AIC's WORD/$\overline{\text{BYTE}}$ input is pulled high selecting 16-bit serial port transfers to optimize serial port data transfer rate. The TMS320C30s XF0, configured as an output, is connected to the AIC's reset ($\overline{\text{RST}}$) input to allow the AIC to be reset by the TMS320C30 under program control. This allows the TMS320C30 timer and serial port to be initialized before beginning conversions on the AIC.

Figure 18. AIC to TMS320C30 Interface



To provide the master clock input for the AIC, the TCLK0 timer is configured to generate a clock signal with a 50% duty cycle at a frequency of H1/4 or 4.167 MHz. To accomplish this, the timer 0 global control register is set to the value 3C1h, which establishes the desired operating modes. The timer 0 period register is set to 1 which sets the required division ratio for the H1 clock.

To properly communicate with the AIC the TMS320C30 serial port must be configured appropriately. To configure the serial port, several TMS320C30 registers and memory locations must be initialized. First the serial port should be reset by setting the serial port global control register to 2170300h. (The AIC should also be reset at this time. See description below of resetting the AIC using XF0). This resets the serial port logic and configures the serial port operating modes including data transfer lengths and enables the serial port interrupts. This also configures another important aspect of serial port operation: polarity of serial port signals. Because active polarity of all serial port signals is programmable, it is critical that the bits in the serial port global control register that control this be set appropriately. In this application all polarities are set to positive except FSX and FSR which are driven by the AIC and are true low.

The serial port transmit and receive control registers must also be initialized for proper serial port operation. In this application, both of these registers are set to 111h, which configures all of the serial port pins in the serial port mode, rather than the general purpose digital I/O mode.

With the operations described above completed, interrupts are enabled, and provided the serial port interrupt vector(s) are properly loaded, serial port transfers may begin after the serial port is taken out of reset. This is accomplished by loading E170300h into the global control register.

To begin conversion operations on the AIC and subsequent transfers of data on the serial port, the AIC is first reset by setting XF0 to zero at the beginning of the TMS320C30 initialization rou-

tine. Setting XF0 to zero is accomplished by setting the TMS320C30 IOF register to 2. This sets the AIC to a default configuration and halts serial port transfers and conversion operations until reset is set high. Once the TMS320C30 serial port and timer have been initialized as described above, XF0 is set high by setting the IOF register to 6. This allows the AIC to begin operating in its default configuration, which in this application is the desired mode. In this mode all internal filtering is enabled, sample rate is set at approximately 6.4 kHz, and the transmit and receive sections of the device are configured to operate synchronously. Conveniently, this mode of operation is appropriate for a variety of applications, and if a 5.184 MHz master clock input is used, the default configuration results in an 8 kHz sample rate which makes this device ideal for speech and telecommunications applications.

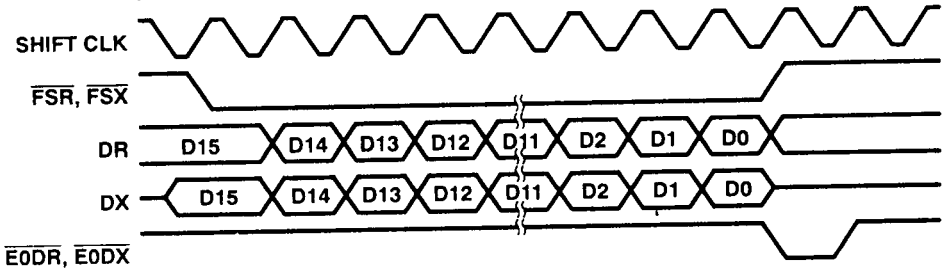
In addition to the benefit of a convenient default operating configuration, the AIC can also be programmed for a wide variety of other operating configurations. Sample rates and filter characteristics may be varied, in addition to which, numerous connections in the device may be configured to establish different internal architectures, by enabling or disabling various functional blocks.

To configure the AIC in a fashion different from the default state, the device must first be sent a serial data word with the two LSBs set to one. The two LSBs of a transmitted data word are not part of the transferred data information and are not set to one during normal operation. This condition indicates that the next serial transmission will contain secondary control information, not data. This information is then used to load various internal registers and specify internal configuration options. There are four different types of secondary control words distinguished by the state of the two LSBs of the control information transferred. Note that each secondary control word transferred must be preceded by a data word with the two LSBs set to one.

The TMS320C30 can communicate with the AIC either synchronously or asynchronously depending on the information in the control register. The operating sequence for synchronous communication with the TMS320C30 shown in Figure 19, is as follows:

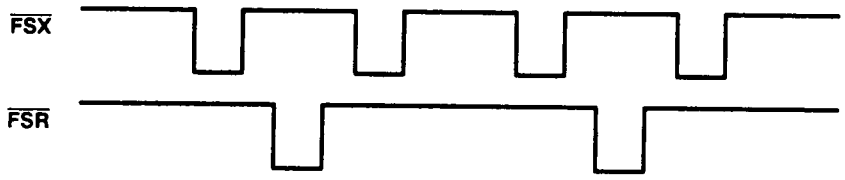
1) The $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ pin is brought low.
2) One 16-bit word is transmitted or one 16-bit word is received.
3) The $\overline{\text{FSX}}$ or $\overline{\text{FSR}}$ pin is brought high.
4) The $\overline{\text{EODX}}$ or $\overline{\text{OEDR}}$ pin emits a low-going pulse.

Figure 19. Synchronous Timing of TLC32044 to TMS320C30



For asynchronous communication, the operating sequence is similar, but $\overline{\text{FSX}}$ and $\overline{\text{FSR}}$ do not occur at the same time (see Figure 20). After each receive and transmit operation, the TMS320C30 asserts an internal receive (RINT) and transmit (XINT) interrupt, which may be used to control program execution.

Figure 20. Asynchronous Timing of TLC32044 to TMS320C30



XDS1000 Target Design Considerations

The TMS320C30 Emulator is an eXtended Development System (XDS1000) which has all the features necessary for full-speed emulation. The TMS320C30 uses a revolutionary technology to allow complete emulation via a serial scan path. If users provide a 12-pin header on their target system, realtime emulation can be performed using the TMS320C30 in their target system.

To use the emulation connector of the XDS1000, the signals shown in Figure 21. should be provided to a 12 pin header (two rows of six pins) with pin 8 cut out to provide keying. Table 3 describes the pins and signals present on the header.

Figure 21. 12 Pin Header Signals

Header Dimensions:
Pin-to-pin spacing: 0.100 inches (X,Y)
Pin width: 0.025 inches square post
Pin length:0.235 inches nominal

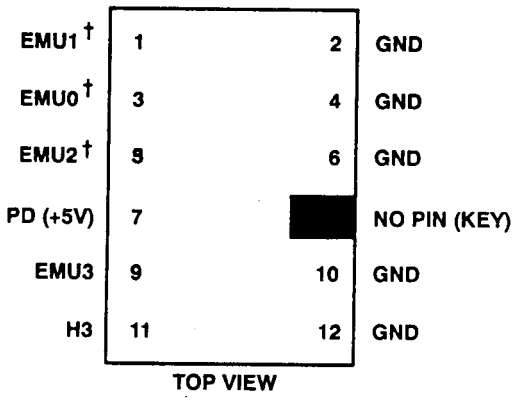


Table 3. Signal Description

Signal Name	Description
EMU0	Emulation pin 0.
EMU1	Emulation pin 1.
EMU2	Emulation pin 2.
EMU3	Emulation pin 3.
H3	TMS320C30 H3.
GND	Ground.
PD	Presence detect . It indicates that the cable is connected and target system is powered up. It should be tied to +5 volts in the target system.

In addition to the signals required at the emulation connector, the EMU4 through EMU6 signals on the TMS320C30 must also be appropiately connected to ensure proper emulation operation. The EMU4 signal must be tied to +5 volts and EMU5 and EMU6 must be left unconnected. Also, the RSV0 through RSV10 signals must be tied to +5 volts as described in the *Third-Generation TMS320 User's Guide* (literature number SPRU031).

Summary

The TMS320C30 is a high-performance 32-bit floating-point digital signal processor. Its dual parallel-interface busses and serial ports, along with a wide variety of additional support interfaces make the device an extremely flexible system-level DSP microprocessor. Using the techniques described in this report, the TMS320C30 can be used to implement sophisticated signal processing applications with the high precision and dynamic range provided by 32-bit floating-point arithmetic.

This application report has described the use of external interfaces on the TMS320C30 to connect it to memories, A/D and D/A converters, and numerous other peripheral devices, as well as the generation of wait states and other system functions.

The interfaces described in this report have all been built and tested to verify proper operation, and the techniques described can be extended to encompass design of more complex systems.