



## **Application Note: Interfacing to the PC32 Expansion Connector**

**DATE:** April 17, 1997

**TO:** Users of PC/SBC32

**FROM:** II Tech Support

**RE:** Designing cards for expanding the  
PC32/SBC32/PCI32/PC31/SBC31 using the  
expansion connector signals

**CC:**

### Scope

Many PC32/SBC32 users wish to design custom daughtercards using the expansion connector. This expansion connector gives access to all required signals from the PC32 to allow the user to connect memory mapped peripherals and connect to the DSP serial port. The daughtercard may implement analog or digital functions to augment the PC32/SBC32.

### Description

There are two methods of interfacing new cards to the expansion signals : memory mapping and synchronous serial port connection. Memory mapping provides a direct interface to the DSP data bus and provides the fastest, most flexible data transfer mechanism to the DSP, but could require as many as 90 signals for a full 32-bit interface to the DSP. The serial ports are much slower at only about 1 Mbyte/sec transfer rate maximum, but only require about 15 wires to interface to the DSP. Most chips will be designed to be memory-mapped such as UARTs, SCSI controllers etc., but some chips like the DSP102 A/D and DSP202 D/A chips provide direct connection to the serial port. An additional restriction on using the serial port is that the on-board analog of the SBC31/PC31 use the serial port for communications and virtually consume the ports in most applications.

## Memory Mapping

All of the products have two undedicated decoded regions of memory intended for add-on peripherals. These signals are “decodes0” and “decodes1”. These signals provide an easy method for attaching most devices to the bus provided that they are less than 64 locations in size on the PC31/SBC31 or 2K in size for the C32 products. For designs requiring larger memory decoding, the address lines must be directly decoded to define an unused region of memory for the memory-mapping.

There are several considerations which need to be addressed in the electrical design

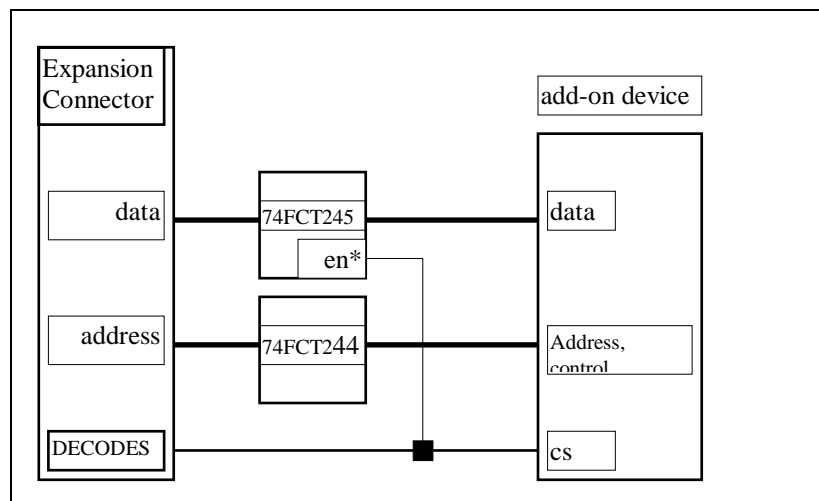
- decoding of a valid memory region
- bus loading and signal integrity
- wait states and timing

### Decoding

For simple peripherals, connect the decodes signal directly to the chip select of the device. The decodes signal is fully qualified with the memory STROBE signal from the DSP in addition to the addresses required for the memory decoding and has the timing necessary to gate the chip on-line as required. The C31 products have decodes mapped into the peripheral region starting at 0xFF FF00 in memory. The C32 products have decodes mapped to the IOSTROBE region of memory.

Most devices have R/W pin also which can be directly connected to R/W. Some devices use a READ and a WRITE signal as their critical timing signals, and the chip select is merely an enable to the device. In this case, use the RD and WR signals which are qualified version of R/W with the STROBE signal.

Typical Add-on Circuitry using Decodes Signal



Multiple devices can be mapped into each decodes region with further decoding since the decodes signals define a region of memory comprising many addresses. Use a fast decoding chip such as a 74FCT138 to further decode the addresses and provide a chip select to each device. In this manner, a single decodes can be used for multiple functions and implementing a full subsystem on a daughtercard.

Large memories and other devices can be mapped into memory by directly decoding the memory address signals qualified by STROBE. On the C31 products suggested memory page for use is 0x6 0000. For C32 products, the Strobe 0 region is available above 0x40000, but do not map anything onto strobe 1 as this is reserved for the fast on-board memory. Of course not all 24 address lines need be decoded, so pick a reasonable page size for decoding and this will reduce the amount of the signals required. The valid chip select signal must be anded with the address lines to produce the correct timing.

### *Signal Integrity and Loading*

The signals on the expansion connector are not buffered in any way to provide the fastest signals possible. Therefore the signals must be buffered if they are presented more than 1 load. Use only fast logic for buffering such as 74Fxx or 74FCTxx parts. The logic used for decoding should control the buffer timing for read cycles.

An important specification frequently overlooked is the time a chip takes to relinquish the bus after its chip select has been deasserted. This can be a problem particularly with slower chips that they continue to drive signals far into the next DSP cycle, thus trampling on the other devices on the bus. All devices without buffers to the bus should have a turn-off time no greater than 20 nS for a 50 MHz design. If the chip cannot relinquish the bus fast enough, use a buffer to isolate the chip.

All of the DSP signals are high speed and require good assembly techniques to maintain proper timing. Wire-wrapping is never possible. The preferred method is to use a printed circuit board. Point-to-point wiring can be used if the wires are kept short and grounding is good.

### *Wait States*

Wait states are usually required for peripherals to interact with the DSP. Devices usually require a fixed number of wait states to allow for access timing, but some devices are indeterminant and require control of the number of wait states on an access to access basis. The indeterminant devices require hardware support logic to control the wait state logic on the DSP.

For a fixed number of wait states, the hardware wait state pin can be controlled or the bus control register can be used to program software wait states. Software wait states are usually easier if the device does not exceed the maximum of 7 software wait states.

The number of software wait states is controlled by the bus control register. On the C31 products this is the PBCR register. For C32 products there are 3 such registers, one for each strobe region 0, 1 and IOSTROBE. The C31 products control the wait state pin directly to define the wait states for each memory region. Below 8M boundary, all devices are zero wait state. Above 8M the memory assumed to be one or more wait states, and the peripherals are all controlled to 4 wait states. Decodes are then 4 wait states, allowing devices up to 140 nS for access timing on a 50 MHz DSP card. The expansion connector has a signal "0WAIT" which is anded with the on-board logic to directly control the ready pin. The 0WAIT signal can be used to control the number of wait states if the PBCR is programmed to respect the RDY input pin on the DSP. This is the primary mechanism for accommodating devices with indeterminant or requiring many wait states. The 0 WAIT pin should be driven with a signal derived from the memory decode sufficiently long enough for that device, then released.

On the C32 products, the number of wait states is controlled by software wait states only. Devices are normally 0 wait states on strobe 1, one wait state on strobe 0, and 2 wait states on IOSTROBE. Decodes are normally 2 wait states, but can easily be reprogrammed as necessary in the bus control registers. If the hardware wait state pin must be used, reprogram the bus control register as necessary so that the software wait states can exist with the hardware control pin in the OR mode of software wait states and hardware wait states.

A cautionary note on the design of a wait state controller is that the STROBE signal from the C31 and C32 does not go high between each cycle. In fact STROBE remains low so long as read accesses are occurring to external memory. Wait state control should be timed by counting bus cycles in logic, do not rely on STROBE to reset the counter between cycles.

### **Synchronous Serial Port Connection**

Interfacing to the synchronous serial port requires very few wires and some chips have interface made specifically to talk with the serial port of the DSP. Some chips with this interface are the Burr-Brown DSP102 A/D and DSP202 D/A. Crystal Semiconductor also makes some A/D and D/A which require little or no logic support. The serial port is a simple clocked data stream which has a sync pulse at the beginning to signal frame start. Logic to talk to such a port is easily implemented in a programmable logic device which can then send data directly to the DSP serial port. Refer to the TI user's guide for a description of the signals.

In general, connect the serial output from your device to the DR0 pin of the DSP, your frame sync signal to FSR0 and program the DSP to generate a clock signal which you receive as a data clock. Use a decoder signal to trigger your device to send out a frame sync pulse followed by the data as prescribed in the TI users guide. If you need to receive data, the connection is similar, except the frame sync will be received by the add-on logic. A good software example for this mode are the A/D and D/A routines for the DSP102 and DSP202 provided with the PC31/SBC31.