RMV Electronics Inc. **Application Note:**		**Application #:** 00001
		Date: September 1994
Description:	Interfacing the ITC232-A to microcontroller/microprocessor peripheral ICs.	**Status:** Draft version

Abstract

By facilitating the connection of peripheral ICs that are oriented to microprocessor 'bus' systems directly to the ITC232-A, the unit's functionality can be greatly expanded.

OPERATIONAL DESCRIPTION

Please refer to the ITC232-A user's manual for specific details on the chip usage.

The interface between the peripheral devices and the ITC232-A is through all of Port C and at least 3 bits of Port B. Port C is used as a bi-directional data bus, while the Port B lines act as bus control lines. A sequence of ITC232-A port writes and reads effectively emulates the operation of a microprocessor data bus. The 3 Port B lines are an active low read (RD\), an active low write (WR\), and an active low chip select (CS\). For each additional peripheral IC added to the ITC232-A 'bus', there will be another chip select line used. While the data bus, read, and write lines are common to all devices, there will be a individual chip select line (Port B output) for each. Therefore the maximum number of peripheral devices that can be connected to the ITC232-A (disregarding Port B) is 6. The 8 connections to Port B being taken up by the read and write lines, and the 6 chip selects.

Many peripheral devices, however, also require local addressing or register selection, i.e. there may be several addressable registers in the same peripheral device. This configuration will require the use of some of the Port B connections. For example, the Intel 82C53 has 4 addressable registers within the device (Ctr0, Ctr1, Ctr2, and Control), requiring two additional address lines. Our system 'bus' must provide the address to this chip as well as the data and bus control lines. Two Port B connections are used as the address 'bus' lines. Similar in concept to the bus control and data lines, these same two lines can be used as address lines to ALL peripherals connected to our 'bus'. This configuration now allows 4 peripheral devices, each with a maximum of 4 addressable registers, to be connected to the ITC232-A. (And we still haven't touched Port A.)

In this example, the 'bus' system will consist of the following connections:

Port C (all 8 connections) - 8 bit bi-directional data bus
Port B, Bit 0 - bus address line 0
Port B, Bit 1 - bus address line 1
Port B, Bit 2 - active low read control
Port B, Bit 3 - active low write control
Port B, Bit 4 - peripheral device #1 chip select
Port B, Bit 5 - peripheral device #2 chip select
Port B, Bit 6 - peripheral device #3 chip select
Port B, Bit 7 - peripheral device #4 chip select

The normal sequence of events on a microprocessor bus is:
The address lines are set to select the appropriate register on the device we need to interface with.
The chip select to the desired device is brought low.
If the transfer is a write, the data bus direction is set to out and the required data value is put on the bus.

1. The bus control line write is brought low.
2. The bus control line write is returned high. (The data is transferred to the peripheral at this point.)

OR

3. If the transfer is a read, the data bus directions is set to in.
4. The bus control line read is brought low.
5. The data is read from the bus.
6. The bus control line read is returned high.
7. The chip select line is returned high.
8. The address and data lines can be returned to normal, then change as required by the next cycle.

There are various setup and hold timing constraints on a microprocessor bus, most of which are of no concern in the ITC232-A interface. This allows the order of the sequence used by the ITC232-A to be varied somewhat for ease of use.

The same sequence of events can be emulated with ITC232-A port configuration, read, and write command sequences. Note: the following sequence assumes the PC serial port has been configured and the functions ITC_send will transfer a string out the serial port, and ITC_data returns the value received from the ITC232-A port read command.

Port B has been set to outputs (PCB$FF).

Example #1: Write the data value $34 (decimal 52) to peripheral device #2, register address 1.

```
/* Set the data bus (Port C) to an output */
ITC_send("PCC$FF");
/* Put the value on the data bus */
ITC_send("PWC$34");
/* Set the appropriate device register address. Set the appropriate chip select lo and write control lo */
/* NOTE: we can get away with setting everything at one time since most peripheral devices have a
setup time of 0. This would have to be confirmed with each device used. If a problem occurs, split the
address and chip select setup command from setting the write line low into two steps */
ITC_send("PWB$D5");     /* binary 11010101 */
/* Return the write bit high. This is required as a single step because most peripheral chips have a finite
data hold time requirement */
ITC_send("PWB$DD");     /* binary 11011101 */
/* Then return the chip select back to high. Although the state of the address lines no longer matters,
they are set to zero for clarity */
ITC_send("PWB$FC");     /* binary 11111100 */
/* Then return the data bus back to inputs. Keep the bus as inputs at all times except a write operation */
ITC_send("PCC$00");
/* Done! The data byte has been sent to the peripheral device on the ITC232-A 'bus'. */
```

Example #2: Read the data value from peripheral device #3, register address 2.

```
/* The data bus (Port C) is already set as inputs, leave as is */
/* Set the appropriate device register address. Set the desired chip select lo and read control lo */
ITC_send("PWB$BA");     /* binary 10111010 */
/* The peripheral device will now put the data value onto the data bus (Port C) */
input_data = ITC_data("PRC$");  /* input_data contains the value from the peripheral IC */
/* Return the read control line hi */
ITC_send("PWB$BE");     /* binary 10111110 */
/* Then return the chip select back to high and return the address to 00 */
ITC_send("PWB$FC");     /* binary 11111100 */
/* Done! */
```

This same sequence of steps can be executed to read from or write to any bus oriented peripheral IC's from the ITC232-A. This includes I/O expansion, timer/counters, clocks, communications controllers, or even additional local memory.

If desired, the ITC232-A Port A can be added. This would allow for several options:

- Port A could be operated in parallel with Port C to form a 16 bit data bus for more advanced peripheral IC's.

- Port A (or a portion of) could be used to expand the number of chip selects and/or register (address) lines in any combination required. For example, if an attached peripheral device required the use of 4 address lines (to select one of 16 on board registers) the number of chip selects would be reduced to 2. Lines from Port A could be used to increase the number of chip selects as required.