

USER'S REFERENCE MANUAL

IOCP-74

PC/104 Input/Output Coprocessor Module

Model No. 100-7585

Doc. No. M7585 Rev: 1.4 06/21/06

DISCLAIMER: This document contains proprietary information regarding SCIDYNE and its products. The information is subject to change without notice. SCIDYNE makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SCIDYNE shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material. No part of this document may be duplicated in any form without prior written consent of SCIDYNE.

WARRANTY: SCIDYNE warrants this product against defects in materials and workmanship and, that it shall conform to specifications current at the time of shipment, for a period of one year from date of shipment. Duration and conditions of warranty may be superseded when the product is integrated into other SCIDYNE products. During the warranty period, SCIDYNE will, at its option and without charge to Buyer, either repair or replace products which prove defective. Repair or replacement of a defective product or part thereof does not extend the original warranty period.

WARRANTY SERVICE: For warranty service or repair, this product must be returned to a service facility designated by SCIDYNE. The Buyer must obtain prior approval and a Return Material Authorization (RMA) number before returning any products. The RMA number must be clearly visible on the shipping container. The Buyer shall prepay shipping and insurance charges to the service facility and SCIDYNE shall pay shipping and insurance charges to Buyer's facility for products repaired or replaced. SCIDYNE may, at its discretion, bill the Buyer for return shipping and insurance charges for products received for repair but determined to be non-defective. Additionally, the Buyer shall pay all shipping charges, duties and taxes for products returned to SCIDYNE from another country.

LIMITATION OF WARRANTY

The forgoing warranty shall not apply to defects resulting from improper or negligent maintenance by the Buyer, Buyer-supplied products or interfacing, unauthorized modifications or misuse, operation outside the published specifications of the product or improper installation site preparation or maintenance, or the result of an accident. The design and implementation of any circuit using this product is the sole responsibility of the Buyer. SCIDYNE does not warrant the Buyer's circuitry or malfunctions of SCIDYNE products that result from the Buyer's circuitry. In addition, SCIDYNE does not warrant any damage that occurs as a result of the Buyer's circuit or any defects that result from Buyer-supplied products. This Warranty does not cover normal preventative maintenance items such as fuse replacement, lamp replacement, resetting of circuit breakers, cleaning of the Product or problems caused by lack of preventative maintenance, improper cleaning, improper programming or improper operating procedures. No other warranty is expressed or implied. SCIDYNE specifically disclaims the implied warranties of merchantability and fitness for a particular purpose. Some states do not permit limitation or exclusion of implied warranties; therefore, the aforesaid limitation(s) or exclusion(s) may not apply to the Buyer. This warranty gives you specific legal rights and you may have other rights which vary from state to state.

CERTIFICATION

Testing and other quality control techniques are utilized to the extent SCIDYNE deems necessary to support this warranty. Specific testing of all parameters is not necessarily performed, except those mandated by government requirements.

30 DAY MONEY-BACK GUARANTEE

SCIDYNE Products are sold with a 30-Day Money-Back Guarantee. If you are not completely satisfied with your purchase, simply return it within 30 days of receipt for a full refund. The Buyer must obtain a Return Material Authorization (RMA) number before returning any products. The entire package, including hardware, software, documentation, discount coupons and


any other accessories supplied must be returned intact and in new and working condition. This guarantee will not be honored for packages that are not returned complete and intact. The Buyer shall prepay shipping and insurance charges to SCIDYNE. To expedite the return process, the RMA number must be clearly visible on the shipping container.


LIFE SUPPORT POLICY

Certain applications may involve the risks of death, personal injury or severe property or environmental damage ("Critical Applications").

SCIDYNE products are not designed, intended, authorized or warranted to be suitable for use in life-support applications, devices or systems or other critical applications without the express written approval of the president of SCIDYNE.

SAFETY AND USAGE CONVENTIONS

 **NOTE:** *Contains important information and useful tips that will assist in the understanding and operation of the product.*

 **CAUTION:** *Calls attention to a procedure, practice or condition that could possibly cause personal injury or damage to equipment.*


 **WARNING:** *Calls attention to a procedure, practice or condition that could possibly cause severe bodily injury, death or extensive equipment damage.*

Table of contents

<u>SECTION 1- HARDWARE</u>	1
Introduction	1
Component Identification	2
Setting the Module Base Address	4
Serial communications	5
Analog Input / Output	7
12-Bit Analog Outputs	8
12-Bit Analog Inputs	8
8-Bit Analog Inputs	8
Expanding the number of 12-Bit analog inputs	8
Digital Input / Output	9
Prototyping Area	10
Parallel Bus Interface	10
SPI Bus	11
Host Interrupts	13
J8 Host Interrupt Selection	13
<u>SECTION 2 - IOCPBIOS FIRMWARE</u>	14
Introduction	14
Firmware Architecture	15
IOCP-74 Input/Output Map	16
Data Array Register	17
Command Register	18
Command #0 - Set Data Array Index Pointer	19
Command #3 - Clear Host Interrupt Requests	20
Interrupt Status Register	21
IOCPbios Functions	22
adc_update	23
add, div, mul	24
dac_update	25
gen_host_irq	26
push_w_reg, pop_w_reg	27
rd_eemem	28
rd_pic_adc	29
serial_rx	30
serial_tx	31

set_dig_dir	32
setbaud	33
spi_select	34
spi_xfer	35
update_dig_io	36
wr_eemem	37
Appendix - A Host Interface Software	38
Appendix - B Specifications	41
Appendix - C Schematic diagram	42
Appendix - D GAL Schematic Diagram	43

Terminology and conventions used throughout this publication

Host This is the computer or similar device into which the IOCP-74 is plugged.

Logic conditions Unless otherwise noted, logic signals are designated as TRUE and FALSE. Names with an asterisk (*) postscript are inverted or active low. Unless otherwise noted TRUE is considered logic "1" (+5vdc) and FALSE is logic "0" (0vdc).

Microcontroller This refers to the microcontroller used on the IOCP-74. The factory default device is a PIC16C74 but other devices such as a PIC16C77 are also supported.

Numbering Systems Computerized equipment often requires that its numeric data be represented in different forms depending on the audience and information being conveyed. Decimal numbers are typically used for end-user data entry and display while internally these values are converted and manipulated in native binary. Hexadecimal numbers are often used by programmers as an intermediate level between binary and decimal notations.

Base	Name	Format
2	Binary	10111001
10	Decimal	185
16	Hexadecimal	0xB9 or B9 ₁₆

Multi-Byte Word Formats

Unless otherwise specified numbers or registers spanning multiple bytes are stored in "little endian" format. The first address (ADDR+0) will contain the Least Significant Byte (LSB) while the Most Significant Byte (MSB) will reside at the highest address.

ADDR+0	ADDR	ADDR+n
LSB	LS <----> MS	MSB

Introduction

The IOCP-74 is an 8-bit PC/104 compliant module designed to perform sophisticated measurement and control operations with minimal host intervention. Applications range from simply relieving a host from high overhead tasks such as data acquisition and parsing communication protocols to using the IOCP-74 as an intelligent “virtual-peripheral” which can execute complex front-end computations, process control loops and logical sequences. It incorporates a 20Mhz Microchip® PIC16C74 RISC Microcontroller to manage all onboard peripherals and independently performs user defined functions. Standard features include: two 12-bit and three 8-bit analog inputs, two 12-bit analog outputs, eight I/O rack compatible digital channels, advanced timer functions (PWM/Capture/Compare), 2k serial EEPROM, shared interrupts and a PIC supervised RS232/RS485 serial communications port. A prototyping area allows extra hardware to be added easily by means of clearly labeled access to buffered PC/104 data, address lines, decoded control signals,

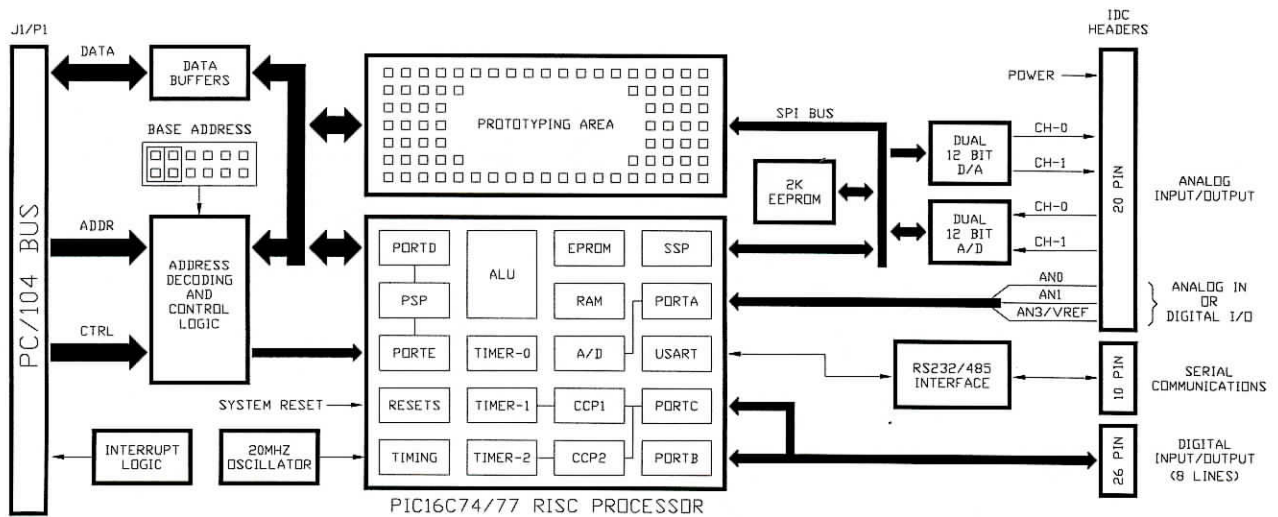


Figure 1 - Simplified Block Diagram

SPI circuitry and support for both through-hole and surface-mount devices. All module data variables and parameters are stored in a RAM array which the host accesses by means of a sequential FIFO interface. No special drivers are required since the module accepts standard input/output commands from DOS and Windows programs. The IOCP-74 is fully programmable using all the familiar and readily available PIC development tools. In addition, the assembly language source code for the pre-programmed factory default configuration is provided royalty-free. It offers a solid foundation which users can rapidly build on when customizing the module for specific tasks. Development efforts are further accelerated by taking advantage of the vast amount of supplemental software routines and application information associated with this very popular microcontroller family. A standard J1/P1 stack-through connector allows the IOCP-74 to reside anywhere within an 8-bit PC/104 stack. Adding an optional J2/P2 connector provides 16-bit stack-through compatibility and access to all upper interrupt request lines.

Component Identification

To properly apply the IOCP-74 it is necessary to become familiar with its various components. The following figure and accompanying table briefly describe their functions and locations. Subsequent sections of this manual explain the components in greater detail.

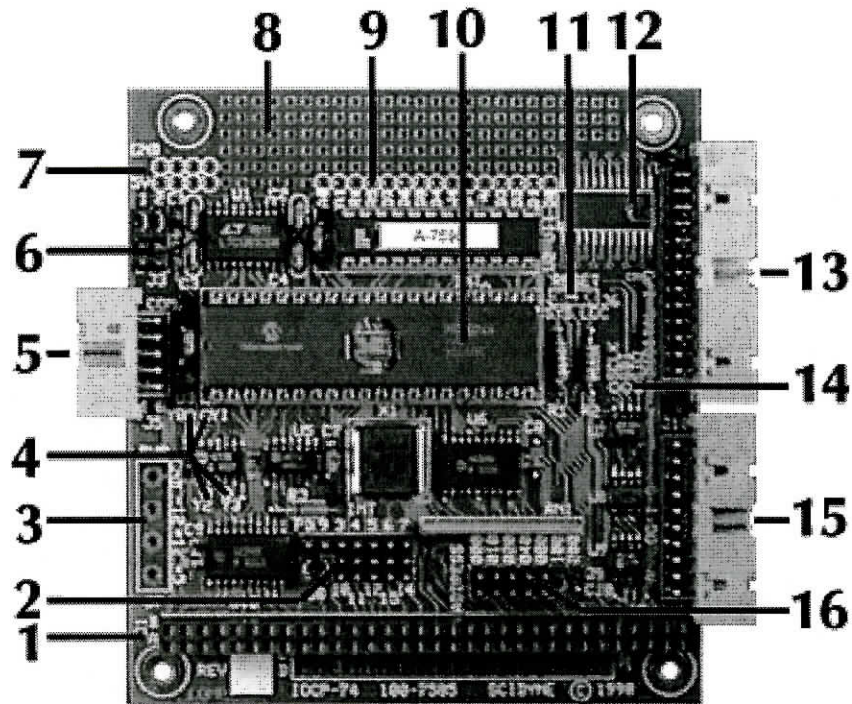



Figure 2 - IOCP-74 Component Identification

IOCP-74 Component Identification	
Item	Description
1	PC/104 J1/P1 Connector This connector is the 8 bit PC/104 bus. An optional 20 pin connector (J2/P2) can be installed to upgrade the IOCP-74 for 16 bit stack-through compatibility and to gain access to upper interrupt request lines.
2	Interrupt configuration jumpers (J8) This jumper block sets which interrupt request line will be used by the IOCP-74 to interrupt the host. The optional J2/P2 connector must be installed if upper interrupt request lines (IRQ10, 11, 12, 14 or 15) will be used.
3	External power connector This feature is not currently supported.
4	Serial Peripheral Interface chip selects These four signals are used in conjunction with the SPI circuitry to individually enable user supplied SPI devices. Y0, Y1, Y2 and Y3 are active low.
5	Communication interface connector (J5) This 10 pin IDC header is used to connect external serial interface devices to the IOCP-74.

6	<u>Communication configuration jumpers (J3)</u> These jumpers configure the serial interface for either RS-232 or RS-485 operation.
7	<u>Logic power connections</u> +5vdc and system common from the host computer are routed to these points and is available for powering users circuitry.
8	<u>Prototyping area</u> These pads support through-hole and wire-wrap construction techniques to facilitate the building of custom circuitry.
9	<u>Parallel bus signals</u> These connections provide convenient access to selected PC/104 bus signals including 8-bit data bus, A0, A1 and input/output control line.
10	<u>PIC16C74 RISC Microcontroller</u> This socketed device controls all IOCP-74 operations. The factory default device is a Microchip® PIC16C74/JW but other similar devices are also supported.
11	<u>Reset Selection (J6)</u> This jumper selects whether the IOCP-74 processor reset will be driven by the host (SYS) or be generated LOcally by the module (LOC). The factory default setting is hardwired to SYS..
12	<u>SOIC pads</u> These pads are used to mount a 24 pin surface mount SOIC device. Each pad is routed to a through-hole which makes working with the surface mount chip much easier.
13	<u>Digital I/O connector (J4)</u> This 26 pin IDC header is used to connect to external digital devices and is pinout compatible with eight position solid-state relay racks.
14	<u>Serial Peripheral Interface bus</u> These signals form the SPI bus. The SPI bus is used by the microcontroller but may also be used to add devices which support that signaling method.
15	<u>Analog I/O connector (J10)</u> This 20 pin IDC header is used to connect the IOCP-74 to external analog devices.
16	<u>Base Address Jumpers (J9)</u> This jumper block sets the base address where the IOCP-74 will reside in the hosts I/O map.

Setting the Module Base Address

The IOCP-74 occupies 8 consecutive I/O bytes and can be placed on any 8 byte boundary within the hosts I/O map. The factory default I/O address is 0x300 (768₁₀) but is easily changed to accommodate any special requirements. The seven position jumper block, J9, determines the base address. Each jumper position corresponds to a “weighted” I/O address as shown in the following table. The actual starting I/O address where the module resides is calculated by simply adding together the “weight” for each jumper that is installed.

 **NOTE:** Addresses between 0x000 through 0x0ff are generally used by the host and should be avoided. Make sure the I/O address selected will not conflict with any existing I/O hardware.

Example: The factory default address is set by placing jumpers in positions 0x100 and 0x200.

Installed jumper	Address value “weight”
J9-6	0x100
J9-7	+ 0x200

	0x300 ₁₆ = 768 ₁₀ = BASE ADDRESS

J9 Base Address Jumper Settings							
J9	-1	-2	-3	-4	-5	-6	-7
Weight (Dec)	0x008 (8)	0x010 (16)	0x020 (32)	0x040 (64)	0x080 (128)	0x100 (256)	0x200 (512)

Shaded area represents J9 factory default installed jumper positions. All other positions are left open. The values printed on the circuit board are in hexadecimal notation.

Serial communications

The IOCP-74 provides one non-isolated serial asynchronous communications port which is controlled by the microcontroller. The port has no specifically defined function and may be freely applied by the user to perform many practical functions. A typical application might involve using the IOCP-74 to parse a communication protocol. This would relieve the host from having to respond to each character as a message is constructed and avoid the time consuming task of node address detection and checksum computations.

The IOCP-74 uses the microcontrollers USART to implement the serial communications interface. Parameters such as baud rate are set via software. The reader should refer to the manufactures data sheet for complete information on all the functions associated with the USART.

External devices connect to the IOCP-74 serial port using one of two signaling methods, RS-232 or RS-485. Jumper block J3 sets which method will be used.

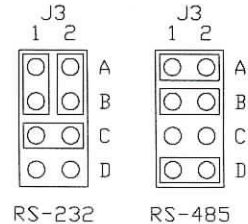


Figure 3 - Serial Port configuration

RS-232

RS-232 is commonly used to connect the IOCP-74 to a single serial device. It offers an almost universal point-to-point interface since most personal computers have a RS-232 port which makes connecting to these devices straight-forward. RS-232 is specified for distances less than 25 wire feet. Because of its single-ended nature, RS-232 offers limited noise immunity and has a potential to be affected by ground loops.

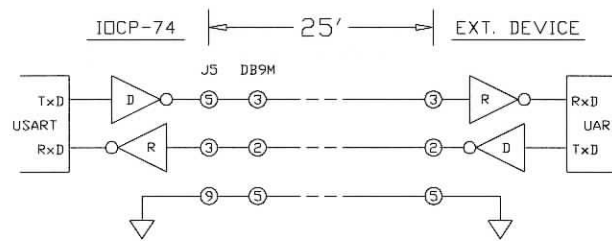


Figure 4- Simple RS-232 Connections

RS-485

RS-485 uses a pair of wires for both the transmit and receive signals. Each pair forms a differential signal which provides excellent noise immunity and is practical over distances up to 4000 wire feet. The IOCP-74 uses RS-485 transmit drivers which are tri-stated during idle conditions thus allowing the IOCP-74 to be used in multi-drop applications. For best performance each end of the network should be terminated with resistors. Resistors which are 120 ohms, 0.25W and carbon composition are commonly used.

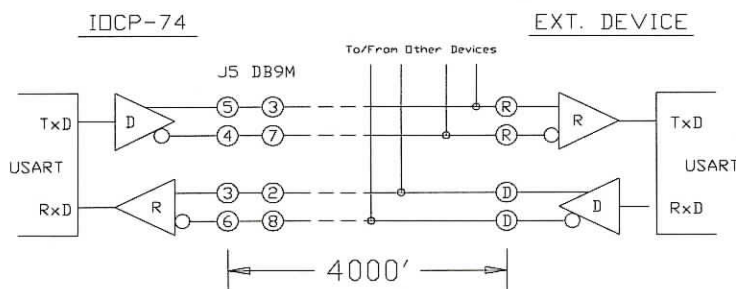


Figure 5 - Full Duplex RS-485 Connections

NOTE: External converters are often used to change a computer's standard RS-232 port to RS-485. Some brands of these devices invert the polarity of the RS-485 signals, expecting a similar device and inversion at the other end. To reinvert the signals for proper use with the IOCP-74 it may be necessary to wire to the converter so that the IOCP-74 inverted signals connect to the converter's non-inverted signals and visa versa.

The serial communication signals are routed to a 10 pin IDC header designated as J5. Its wiring pinout and signal description are shown in the following table. The signals form a standard DTE interface when connected to a DB9-M connector as depicted and is the same arrangement used in most personal computers.

J5 Communication connector			
DB9 Male	J5 Pin	I/O	Description
1	1	-	Not used
2	3	I	RS-232 This pin is the receive input RS-485 This is the negative (inverting) differential receive input
3	5	O	RS-232 This is the transmit output RS-485 This is the negative (inverting) differential transmit output. The signal is high impedance when idle.
4	7	O	+5V This pin provides +5 volts and is supplied by the host computer logic supply. This signal may be used in conjunction with external resistors to create pull-up/pull-down termination required when implementing some RS-485 multidrop communications networks. Unless required for termination, no connections should be made to this pin.
5	9	O	GND This pin is ground of the host computer. In RS-232 communications, this signal must be connected to the ground of the other device. This signal may be used in conjunction with external resistors to create pull-up/pull-down termination required when implementing some RS-485 multidrop communication networks.
6	2	-	Not used
7	4	O	RS-232 Request-To-Send (RTS) This signal is the Request-To-Send output. Although not required to implement RS-232 communications, it can be used as a gating signal to indicate when a transmission is about to, and is taking place. This signal will change from space (-10vdc) to mark (+10vdc) just prior to data transmission. It will remain active for the entire transmission and return low after all bits of the last character have been transmitted. RS-485 This is the positive (non-inverting) differential transmit output. The signal is high impedance when idle.
8	6	I	RS-232 Not used RS-485 This is the positive (non-inverting) differential receive input
9	8	-	Not used
-	10	-	Not used

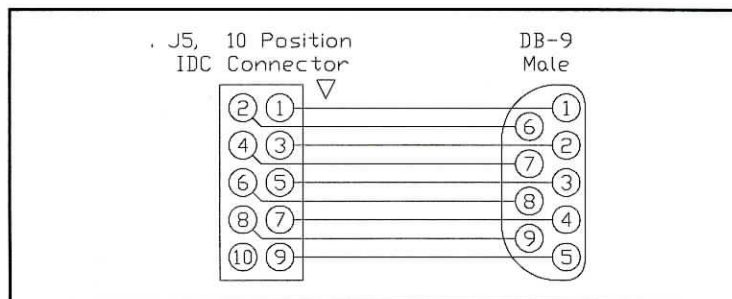


Figure 6 - Communication Cable Wiring Schematic

Analog Input / Output

The IOCP-7 provides five analog input channels and two analog output channels. All analog signals are non-isolated and routed to a 20 pin header designated J10.

J10, Analog Input/Output connector pinout	
Pin	Description
1	<p>12-Bit Analog Inputs These pins are associated with the LTC1298 dual 12-bit analog-to-digital converter. This device is configured by the software to operate in either of two basic modes, single ended or differential. Each pin's function depends on which mode is being used. In both cases, the input range is 0-5vdc.</p> <p>Single ended: Analog input AI-0 Differential: This is the differential input and can be software configured for positive or negative polarity.</p>
3	
5	<p>12-Bit Analog Outputs These pins are associated with the LTC1446 dual 12-bit digital-to-analog converter. The output range is 0-4.095vdc (1mv/step).</p> <p>Analog output AO-0</p>
7	
9	<p>8-Bit Analog inputs These pins are associated with the 8-bit analog-to-digital converter located within the microcontroller. Although described here as analog inputs, they could alternately be configured in software to function as general purpose digital I/O.</p> <p>RA0, Analog input AI-2</p>
11	
13	
15	<p>Power Supplies These pins are used to supply power to external circuitry connected to the IOCP-74 via connector J10. The power is supplied by the host and will only be available if already present on the host.</p> <p>-12Vdc, Unfused</p>
17	
19	
2, 4, 6, 8	<p>Analog Ground These pins form a pseudo analog ground with the intent of minimizing the influence of electrical noise and measurement errors. Their routing on the IOCP-74 circuit board is such that they do not share the same pathway as potentially 'dirty' digital circuits. Unfortunately, the PC/104 bus does not have provisions for true separate grounds thus requiring that these signals join digital ground at the P1/J1 connector.</p>
10, 12, 14	<p>Digital Ground These pins are directly connected to digital ground. External circuits employing digital circuits should use these pins as ground.</p>
16, 18, 20	<p>Unassigned These pins are not used and may be freely assigned by the user when customizing the IOCP-74.</p>

⚠ WARNING: *The design of user's external circuitry which the analog signals connect to can vary widely depending on the particular application being implemented. For this reason the signals are not buffered or protected in any way which could degrade the intended systems performance. Precautions must be taken not to exceed the limits of the modules analog I/O devices or permanent damage may result.*

12-Bit Analog Outputs

The two 12-bit analog outputs use a LTC1446 digital-to-analog converter. The output range of this device is 0-4.095vdc or expressed another way, 1mv/step. Each output is capable of sourcing or sinking 5ma and can drive 1000pf loads without going into oscillation. The reader should refer to the manufacture's data sheet for additional information about the LTC1446.

12-Bit Analog Inputs

The two 12-bit analog inputs use a LTC1298 analog-to-digital converter. This device is software programmable to appear as two single ended or one differential analog input. When configured for differential operation the two analog inputs (AI-0, AI-1) form the differential inputs. The assignment of input polarity is also software programmable. The modules Vcc (5vdc supplied by host) is used as the conversion reference. This results in an input sensitivity of about 1.22mv/step (5v / 4096). Variations in the hosts power supply can affect this value and the resulting digitized number. For the most stable reading digital filtering should be used. The maximum input voltage must not exceed 5.3Vdc or permanent damage to the module may result. The reader should refer to the LTC1286/LTC1298 data sheet for additional information about this device.

8-Bit Analog Inputs

The PIC microcontroller includes a multi-channel 8-Bit analog to digital converter. Three of the channels have been reserved for use by the user. Alternately these signals can be used as general purpose digital I/O. When used as analog inputs, the reference voltage for the converter may be configured to be either the modules Vcc (5vdc) or an external reference applied to AI-4. The external reference must operate within the bounds of the analog-to-digital converter as outlined in the PIC microcontroller documentation.

Expanding the number of 12-Bit analog inputs

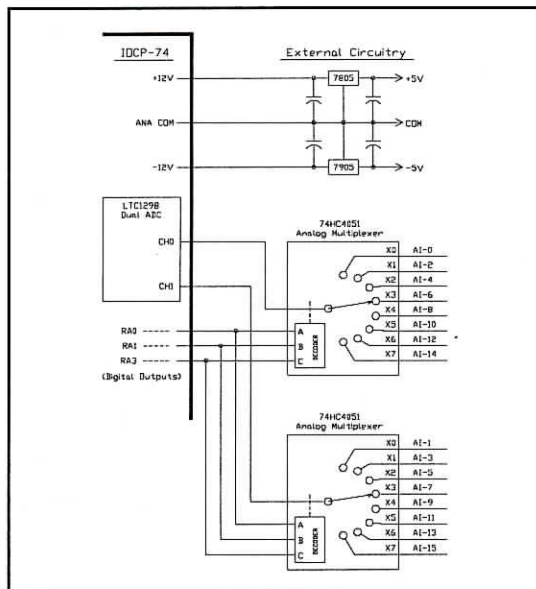


Figure 7 - Analog Input Expansion

If the three 8-bit analog input channels are configured for digital outputs an external circuit like the one shown can be used to expand the number of 12-bit analog inputs from just two to sixteen. Additional circuitry could also be included to change the input range, accept current signals instead of voltage or provide optical isolation.

NOTE: *To easily accommodate custom external circuitry SCIDYNE offers a universal terminal board. This product provides a convenient solder-pad/wire-wrap prototyping area, mounting holes for various sizes of IDC, a 16 position removable screw terminals and is designed to fit in a SnapTrack® or mount on standoffs. See product bulletin PB7593 for further details.*

Digital Input / Output

The IOCP-74 provides eight general purpose digital input/Output channels. The majority of these signals are derived directly from PORTB of the microcontroller but two of the signals come from PORTC so that special timer functions associated with that port are available to the user. The signals are routed to a 26 pin IDC connector, J10. This connector is pin-out compatible with industry standard eight position I/O racks.

Digital Inputs

The six PORTB inputs can be programmed to provide weak pull-ups which makes interfacing to components such as keypads, switches and contact closures much easier. The PORTC inputs do not have this capability and must be used with external pull-up resistors or be used in a manner that doesn't require pull-ups. The input levels on any input must not exceed the published specifications. The upper four digital bits (RB<7-4>) can be used for change-of-state detection interrupts.

Digital Outputs

The outputs provide a relatively high drive capability, $\pm 25\text{ma}$ @ 5Vdc. This allows solid-state relays, LED displays and many other devices to be driven directly without the need for additional circuitry. Some devices, such as speakers, should be AC coupled through a capacitor to prevent excessive current draw when the output is in a static state.

Devices such as servo motors or electro-mechanical relays operate at higher currents and/or voltages than can be supplied by a standard digital output. If additional drive capability is required, circuits like those shown in the figure to the right can be used.

J4, Digital Input/Output Connector		
Pin	Source	Description
1	Host	+5Vdc, Unfused
3, 5, 7	---	Not used
9	RB7	DIO.7
11	RB6	DIO.6
13	RB5	DIO.5
15	RB4	DIO.4
17	RB3	DIO.3
19	RC2/CCP1	DIO.2
21	RC1/T1OSCI/CCP2	DIO.1
23	RB0/INT	DIO.0
25	Host	+5Vdc, Unfused
2-26 (even)	Host	Digital common

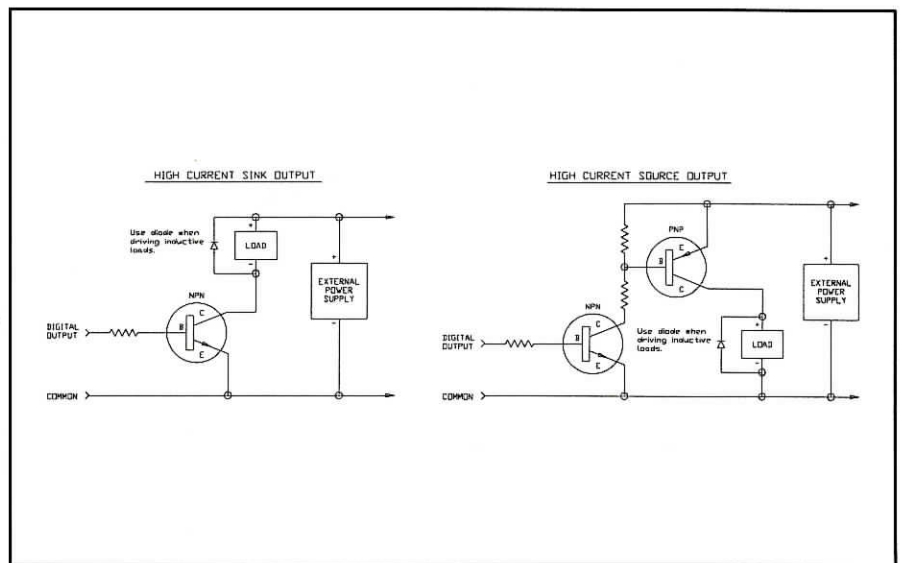


Figure 8 - High current drive circuits

Prototyping Area

The IOCP-74 allows a user to include additional circuitry to meet special applications requirements. The prototyping area supports both solder-pad and wire-wrap construction methods. In addition, a 24 pin SOIC outline is provided for mounting a single surface mount device. The versatility of the IOCP-74 permits a user to combine both parallel bus or SPI (Serial Peripheral Interface) devices in their designs.

Parallel Bus Interface

The parallel bus is used to connect devices which have a byte oriented interface. The operation of this interface is controlled entirely by the host. It runs independently from the IOCP-74 and shares only the address decoding logic and 8-bit data buffers with the onboard microcontroller. A minimal decoding scheme is used which readily supports devices that have separate inputs for read, write and chip enable. Typical components in this category would include the familiar 8255 peripheral interface adapter and 8254 timer. Adding minor logic, to qualify the PC/104 bus IOR* and IOW* signals with the decoded address strobe, would allow simpler devices such as latches and buffers to also be used. Two chip selects, spanning four host I/O address bytes (2 words), are available for user circuitry. The available signals are described below and example circuits are shown in figure 8.

Parallel Bus Signal Descriptions		
Name	I/O	Description
A<0, 1>	O	System Address These lines are used to address locations within the I/O space assigned to the IOCP-74. A0 is the least significant bit (LSB) These lines are generated by either the processor or DMA controller. They are active high.
R*, W*	O	R* Input/Output Read This signal commands an I/O device to drive its data onto the data bus. This signal is active low and is generated by the host. W* Input/Output Write This command line instructs an I/O device to read the data on the data bus. This signal is active low and is generated by the host.
RS	O	System Reset This signal is generated by the host during a system reset. It is normally low, goes high during reset and then returns low. Local resets do not generate this signal
D<0-7>	I/O	Buffered Data These eight signals are the buffered PC/104 data bus. They are normally tristate but become active when the host computer is accessing the IOCP-74 through I/O operations.
4	O	Decoded address Strobe These signals are generated by the I/O decoding logic. They are normally high but go low during I/O operations within their respective address range and in conjunction with the PC/104 Bus IOR* and IOW* signals. Address strobe 4 Active: BASE ADDRESS + 4, 5 Use with peripheral devices which provide both input and output controls or with additional logic to create sub-decoded input and output strobes.
6	O	Address strobe 6 Active: BASE ADDRESS + 6, 7 Use with peripheral devices which provide both input and output controls or with additional logic to create sub-decoded input and output strobes.

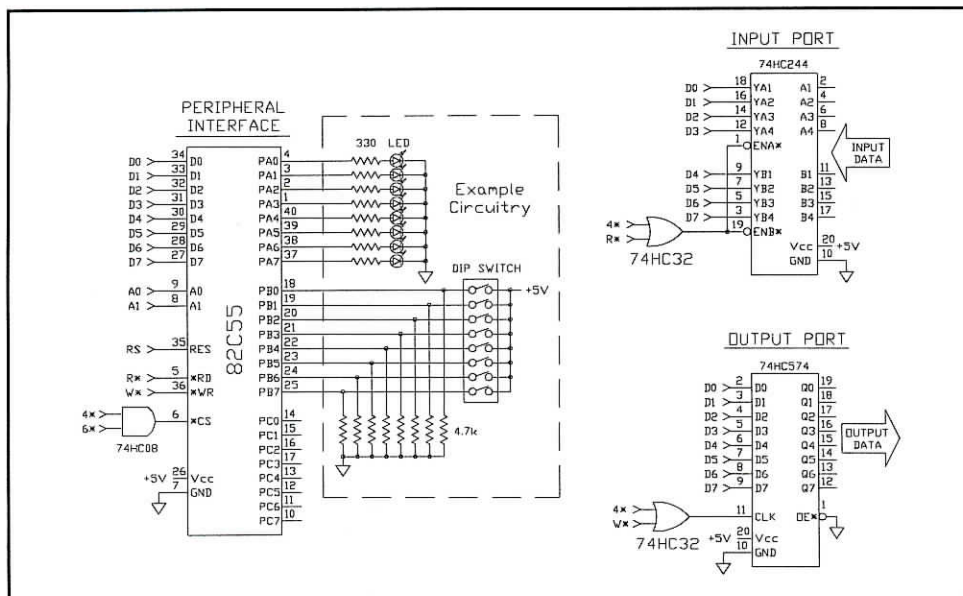


Figure 9 - Parallel Bus Prototype Circuit Examples

SPI Bus

To reduce pin counts and achieve higher levels of integration many component manufactures are supplying their products with a Serial Peripheral Interface (SPI). SPI is basically a synchronous serial communication standard in which the equivalent of multiple parallel bus bits are shifted one at a time across two or three wires. The IOCP-74 uses SPI to communicate to many of its own internal peripherals and provides the hardware and firmware for a user to connect four additional SPI devices. The signals available for SPI are listed below.

Serial Peripheral Interface Signal Descriptions		
Name	I/O	Description
SCLK	O	Serial Clock This signal is the shift-clock generated by the microcontroller SSP circuitry.
SDI	I	Serial Data Input Serial data is shifted into the microcontroller on this line. Connect this signal to the serial data output of all the SPI devices.
SDO	O	Serial Data Output Serial data is shifted out of the microcontroller on this line. Connect this signal to serial data inputs of all the SPI devices.
Y0*, Y1*, Y2*, Y3*	O	Decoded chip selects These signals are generated by the microcontroller in conjunction with the SPI decoding logic and application software. They are normally high but go low during a SPI data transfer to the selected device. Each select can be used to enable a single SPI device

The SPI signals are not located next to the other prototyping signals. The diagram to the right shows their position on the IOCP-74 PCB.

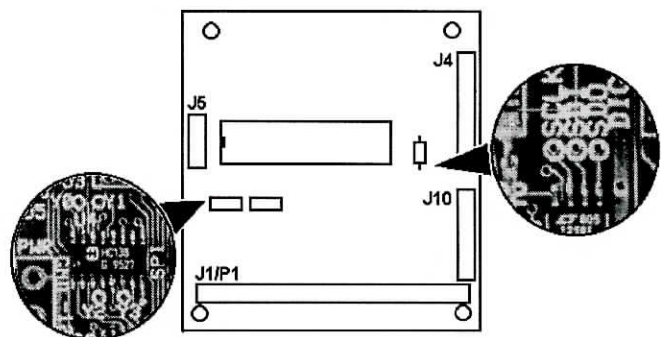


Figure 10 - SPI Bus signal locations

A typical SPI wiring arrangement is shown below. The IOCPbios includes routines which perform the necessary lower level functions and makes adding SPI device to the IOCP-74 straightforward

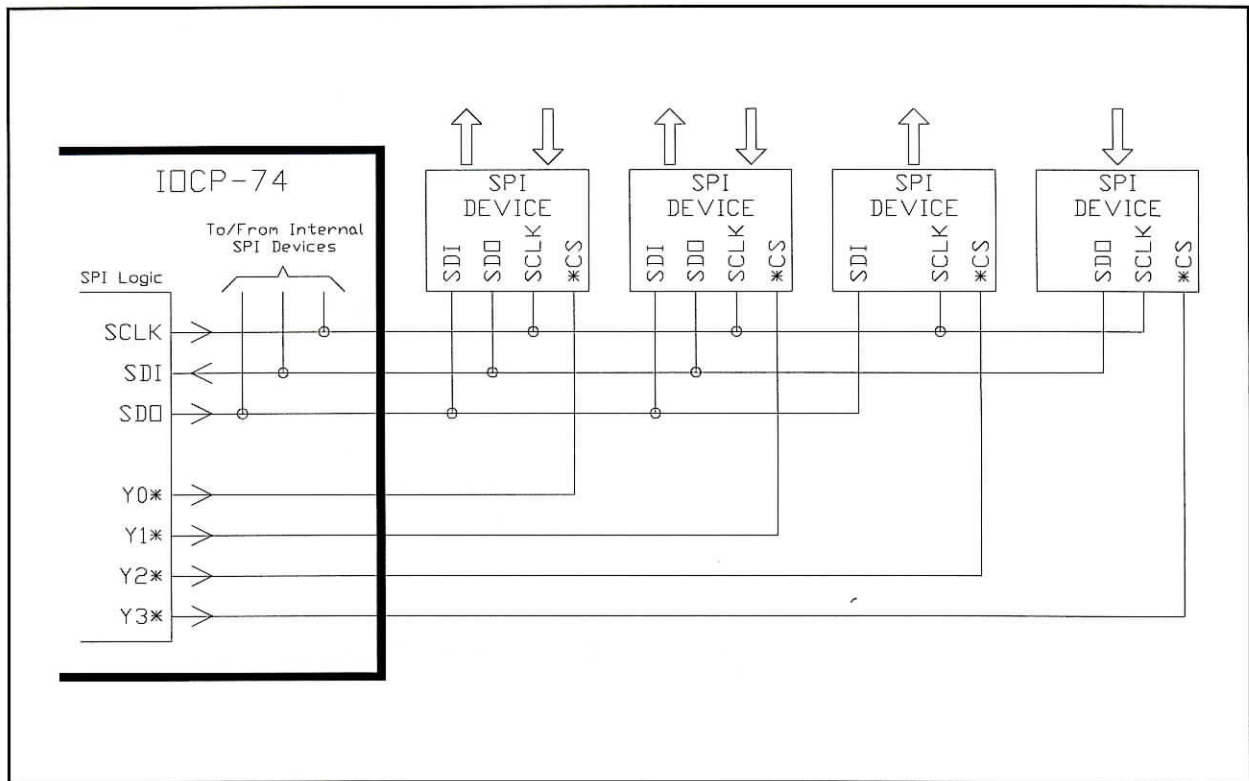



Figure 11 - Typical SPI connections

Host Interrupts

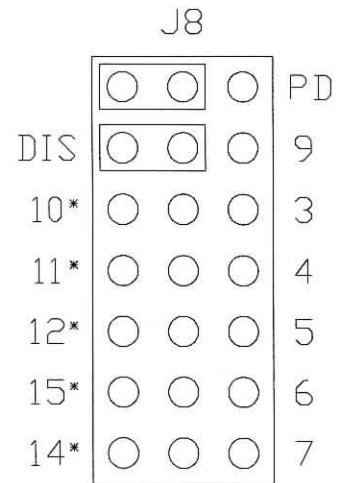
The IOCP-74 provides the capability to generate host interrupt requests using the onboard microcontroller. This feature is very useful when the IOCP-74 will be performing functions independently. Host resources will only be required when the IOCP-74 has determined a user pre-programmed event has occurred.

J8 Host Interrupt Selection


Jumper block J8 configures which host interrupt will be associated with the IOCP-74. Any interrupt may be used but the optional J2/P2 connector will be required to access the upper interrupt request channels (IRQ10, 11, 12, 14 or 15). An interrupt is selected by placing a shorting jumper between the center row of J8 and the corresponding interrupt pin. Interrupt capability is disabled by placing a shorting jumper at the DIS position of J8. The interrupt driver on the IOCP-74 conforms to the method for interrupt sharing as outlined in the PC/104 specification. This method recommends that one of the PC/104 modules sharing an IRQ provide a passive pull-down resistor to ground. The IOCP-74 can supply the pull-down resistor when a shorting jumper is installed at the PD position of J8. The pull-down resistor has no effect when interrupts are disabled.

 **NOTE:** Try selecting an interrupt which is not currently being used by other system resources. Certain interrupts have a defacto standard usage and should be avoided. If interrupts must be shared make sure all the software applications involved supports interrupt sharing. To prevent excessive current draw and the possibility of erroneous operation, use only one pull-down per IRQ.

IRQ9 is re-directed to IRQ2 on most AT style computers.



* Requires optional J2/P2 Connector

 **NOTE:** The host Interrupt Service routine MUST issue a CMD3 to the IOCP-74 hardware to clear the interrupt request. Please refer to the IOCPbios section of this manual for further details..

Introduction

The IOCP-74 is supplied with an integrated set of assembly language routines which directly control many of the modules basic core functions. Collectively these routines, or firmware, will be referred to as **IOCPbios**. These routines, by themselves, do not satisfy any particular application but instead offer a reasonable starting point on which users can build and customize the IOCP-74 to meet specific requirements. The software is provided in ASCII format and is easily modified and enhanced by the user. The source code may be combined with other languages such as BASIC, C or directly compiled using the MPASM PIC assembler. This assembler can be downloaded for free at the microchip web site (www.microchip.com). When modifying the source code it is suggested that any original routine, variable and macro names as well as their purpose be preserved. This will assure compatibility with future releases of the IOCP-74 software.

⚠ WARNING: *For easier explanations, the requirement to be applicable with a diverse range of applications and to promote quicker understanding by all users, simplicity was given precedence over robustness during the development of the IOCPbios software. For these reasons, the determination of usefulness, suitability and sufficiency of the supplied routines when implementing custom applications is the sole responsibility of the user.*

This section of the manual will give mid-level explanations of the core routines contained within IOCPbios as well as its basic overall structure. The user should refer frequently to the actual assembly language source code for a more detailed understanding of its operation. Where possible, sample code fragments and illustrations are included to help exemplify the concepts.

Firmware Architecture

The IOCPbios firmware and application utilizing it can be visualized by the layout in figure 12. It basically is separated into two parts, an upper level and a lower level. The lower level is further distinguished by the general purpose software routines and interrupt level functions.

The main loop operates by examining the **CYC_SYNC_BIT**. This flag is set by the interrupt service routine supporting TMR-0. The default period is 13.1ms but is easily changed in software to accommodate any particular user requirements. When the main loop sees the flag is set, it immediately resets it for next period and then executes the software comprising the application. After the application software has completed, the main loop repeats its examination of the **CYC_SYNC_BIT** waiting for it to be set again.

The advantage of this type of architecture is that each segment of the main loop “knows” exactly how often it runs. Provided that the worst case execution time for all the application software doesn’t exceed the main loop period, each program segment is assured of running once every 13.1ms. This “knowledge” can be very useful when performing control applications which must be related to actual engineering time units (such as feet/minute). It also provides a structure which makes writing, debugging and maintaining the software much easier. Any time sensitive routines can be delegated to interrupts (or sub cycles) and their results reported to the upper level software.

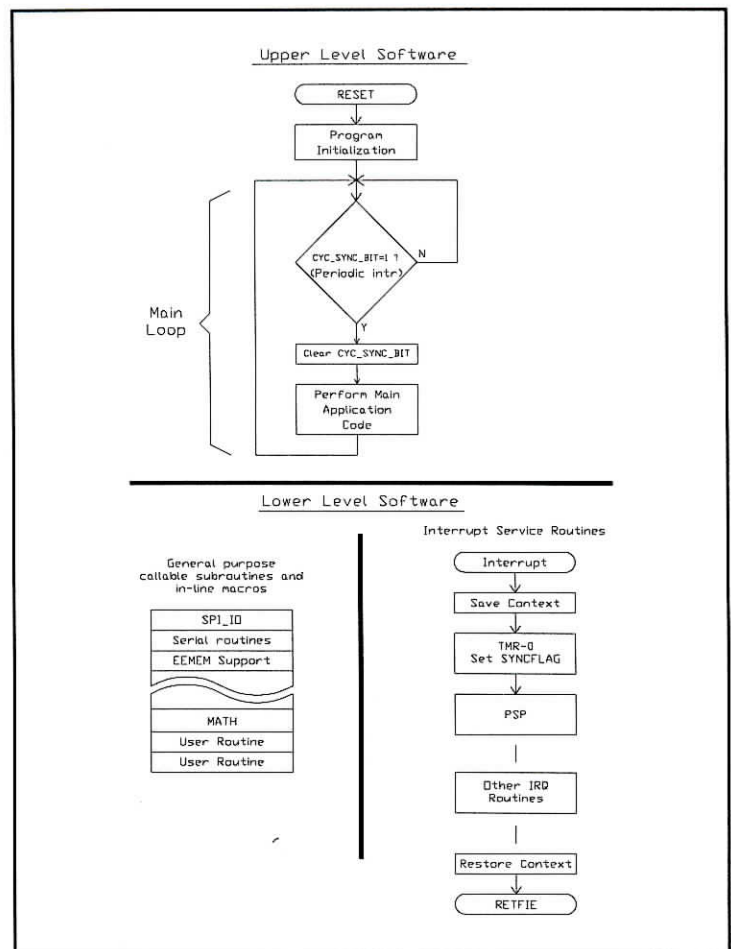


Figure 13 - IOCPBIOS software structure

IOCP-74 Input/Output Map

The hosts views the IOCP-74 module as eight consecutive 8-bits bytes within its Input/Output map. The table below shows the organization of the bytes as seen by the host. The registers are discussed in greater detail in subsequent sections of this manual.

IOCP-74 I/O MAP SUMMARY										
I/O Address (Byte)	Register Name	R/W	Host Data Bus							
			D7	D6	D5	D4	D3	D2	D1	D0
0	Data Array Register	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	Command Register	W	0, 0 = Set data array index pointer		AUTO-INC	A5	A3	A2	A1	A0
			0, 1 = Available							
			1, 0 = Available							
			1, 1 = Clear interrupts		X	X	X	X	X	X
2	Interrupt Status	R	X	X	X	X	X	X	X	INTR-STAT
	Processor Reset	W	X	X	X	X	X	X	X	X
3	Reserved	R/W	X	X	X	X	X	X	X	X
4, 5	Prototype Parallel Select "4"	R/W	D7	D6	D5	D4	D3	D2	D1	D0
6, 7	Prototype Parallel Select "6"	R/W	D7	D7	D5	D4	D3	D2	D1	D0

Notes:


- 1) X = Not used or not implemented. Bit read will be undefined
- 2) Addresses are relative to the modules base address

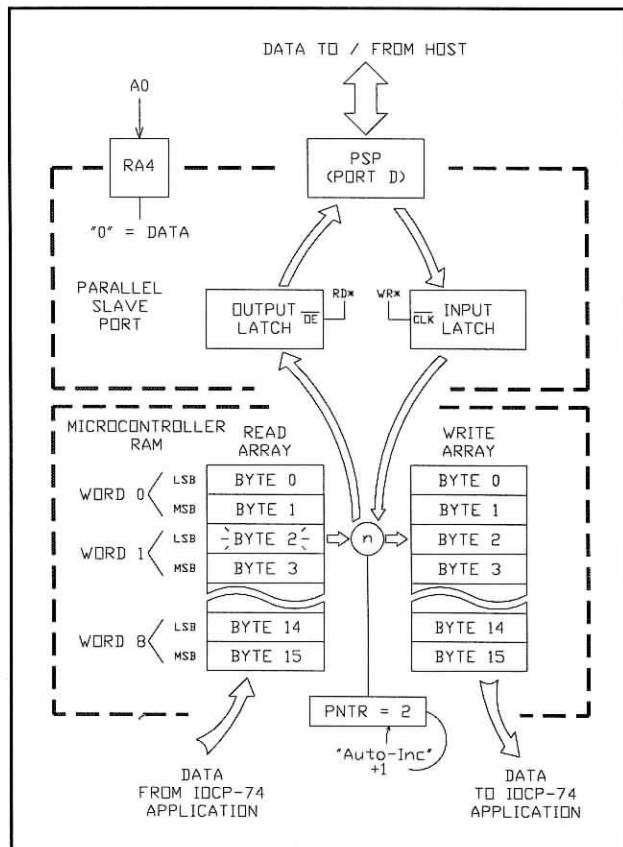
Data Array Register

The host exchanges application information with the IOCP-74 microcontroller by means of sequentially accessed data arrays. These arrays reside in the microcontroller RAM and operate in conjunction with special IOCPbios software and the Parallel-Slave-Port feature of the PIC16C74. Information written by the host is stored in **WRITE** array. Information produced by the microcontroller and intended for the host is stored in the **READ** array.

The write array holds data which, when acted upon by the microcontroller, allows the host to directly influence the operation of the IOCP-74 module. Items such as the analog output values, out-going communications streams or any users run-time configurable application parameters might reside in the write array.


The read array contains information ultimately destined for the host and might include items for the value of the analog inputs, the status of some internal operation or perhaps a counter representing the number of pulses appearing at one of the digital inputs.

 **NOTE:** The array names were given from the hosts perspective. The microcontroller actually "writes" into the read array and "reads" from the write array.



Exactly what data the read and write arrays hold, an arrays overall size and how the data is organized is determined solely by the users application software. Although each array location is a single byte, no limitations are imposed against using multiple bytes to hold larger numbers or data strings. The only true restriction is that the host software and IOCP-74 software understand and maintains the same organization.

The array data appears to the host through a single 8-bit I/O port. For this reason, a single pointer is used to individually address each location within each of the multi-element arrays. The pointer, which is write only, is shared between the read and write arrays and operates the same way for both. The value contained in the pointer will cause the corresponding array location to be accessible by the host. Which array will be accessed is controlled by the hosts software through Input (Read Array) or Output (Write Array) instructions directed at the module. The pointer can optionally be configured to automatically increment after each data array access. This feature is particularly useful for a host to rapidly exchange data with the IOCP-74 without having to update the pointer in between data accesses. When the end of the array is reached the pointer wraps around to the first location. If the pointer auto-increment is not used, the host will always read or write the same array location until either the pointer is changed or auto-increment feature is enabled.

 **NOTE:** Although the arrays are part of the microcontrollers RAM, the pointer value is relative to the base of the array regardless of the arrays absolute address. EX: Setting the pointer to 0 will allows accesses to the first array location NOT address zero of the microcontrollers RAM.

Command Register


The command register is a write only register which the host uses to instruct the microcontroller on specific functions to perform. The two upper data bits determine how the remaining six data bits are interpreted by the module firmware. The two functions currently defined are setting the data array index pointer and clearing module generated interrupts.

CMDREG:							0x01
Bit 7	6	5	4	3	2	1	Bit 0
W	W	W	W	W	W	W	W
CMD1	CMD0	The function of these bits vary depending on the state of CMD bits					
Reset:	0	0	0	0	0	0	0

Command Register Functions			
CMD1 Bit	CMD0 Bit	CMD#	Description
0	0	0	Set data array index pointer
0	1	1	Available for user
1	0	2	Available for user
1	1	3	Clear interrupt request

Command #0 - Set Data Array Index Pointer

The user accessible application data is contained within the microcontroller in the form of data arrays. The host sequentially accesses the array elements through a single I/O port (See Data Array Register). Each element to be accessed is indirectly referenced by the DATA ARRAY INDEX POINTER. The same pointer is shared for both read and write data array operations. The host can manually set this pointer to continuously reference the same location within the array. Optionally, an auto-increment bit can be set which causes the pointer to reference the next logical array location after each array access. In this mode, the pointer will wrap around to the start of the array after the host accesses the last array location. During reset the pointer is set to access the first array location and auto-increment is enabled.

 **NOTE:** The index value written to the pointer corresponds to a single byte in the read or write array and is in relation to the base of the array not its absolute address within the microcontrollers RAM memory.

CMDREG: CMD0 - Set Data Array Index Pointer

0x01

Bit 7	6	5	4	3	2	1	Bit 0
W	W	W	W	W	W	W	W
CMD1=0	CMD0=0	AUTOINC	A5	A3	A2	A1	A0
Reset: 0	0	1	0	0	0	0	0

CMD[1:0] Command control bits

Both of these bits being zero instruct the IOCP-74 firmware to interpret the remaining bits as parameters for the setting of the data array index pointer.

AUTOINC Automatic Increment

This bit controls whether the data array index pointer will automatically increment after each DATA REGISTER access. A logic "1" sets automatic incrementation. A logic "0" disables this function causing the same array location to be repeatedly read or written to for each DATA REGISTER access.

Bit 4 Not used. Write as zero for future compatibility

A[4:0] Data array index pointer address

These bits set the data array index pointer.


Command #3 - Clear Host Interrupt Requests

After servicing an IOCP-74 generated interrupt request, the hosts interrupt service routine MUST clear the request. This is accomplished by writing to the command register with both CMD bits set. Since this command does not interfere with other command register instructions or data array operations it can be issued at any time.

CMDREG: CMD3 - Clear Interrupt Requests							0x01
Bit 7	6	5	4	3	2	1	Bit 0
W	W	W	W	W	W	W	W
CMD1=1	CMD0=1	0	0	0	0	0	1
Reset:	Na	Na	Na	Na	Na	Na	Na

CMD[1:0] Command control bits
 Both of these bits being logic one instruct the IOCP-74 firmware to clear the interrupt status bit in hardware and software. .

Bits 5-1 Not used. Write as zero for future compatibility

 **NOTE:** This command only clears the hardware interrupt request to the host. The user must supply the supporting software within the IOCP-74 application to manage the interrupt request internally..


Interrupt Status Register

The Interrupt Status Register allows the host to determine if the IOCP-7 has caused an interrupt request. It is primarily used when an interrupt shared amongst multiple modules and the service routine must determine which device has caused the request. It can alternately be used without interrupts to periodically "Poll" the IOCP-74 to see if the module is signaling an anticipated event has occurred and requires special host services. Only the least significant bit (bit 0) contains meaningful information. The INTRSTAT bit is cleared by writing CMD3 to CMDREG.

INTRSTAT: Interrupt Status Register							0x02
Bit 7	6	5	4	3	2	1	Bit 0
R	R	R	R	R	R	R	R
X	X	X	X	X	X	X	INTRSTAT
Reset:	Na	Na	Na	Na	Na	Na	0

Bit 7-1 Not used. The state of these bits is indeterminate. The host software should mask them off or use an instruction that only tests the INTRSTAT bit.

INTRSTAT Interrupt Status
This bit is set when the module firmware generates a host interrupt request. Interrupts must be properly configured for the interrupt to be passed up to the host. This bit is reset by writing a CMD3 to CMDREG.

 **NOTE:** *The IOCP-74 firmware has been designed with a special feature which facilitates production testing of the board. Writing any value to the Interrupt Status Register from the host can generate a reset signal to the PIC processor. Users may also want to use this feature but careful consideration should be observed. The duration of the host INTRSTAT write operation controls the duration of the reset signal. On some particularly fast host processor boards the write operation may not be sufficiently long enough to reliably reset the PIC. It is suggested to fully test this feature under various conditions before permanently including it in your application. A hard reset, such as when first powering the system on, is handled differently in the firmware and does not affect the reset.*

IOCPbios Functions

The following pages list the IOCPbios function calls and give simple program examples.

Function: adc_update

Revision: 1.00

Description: This function digitizes the voltage applied at the two 12-bit analog input channels, AI-0 and AI-1. The results are stored in RAM locations ai_ch0 and ai_ch1 respectively. Both of these locations are 16-bits (1 word, 2 bytes) in size with only the lower 12 bits (B<0-11>) actually containing data. The unused bits will read as zero. The IOCP-74 uses a LTC1298 as the analog-to-digital converter. This device provides for two single ended or one differential measurement. By default the single ended mode is used. The convertors reference voltage is derived from the digital logic +5vdc supply. This results in the 12-bit resolution (1 part in 4096) equaling approximately 1.221mv/step. Fluctuation in the supply voltage will directly affect this number. More details can be found in the LTC1298 manufactures data sheet.

Conditions:

Before: None:

After: ai_ch0 = value of digitized voltage applied to AI-0
ai_ch1 = value of digitized voltage applied to AI-1

Example Software: Update the 12-bit analog input values and present to host.

```
main:
loop:  BANK0                ;Switch to bank-0

      btfss opsys_var0,CYC_SYNC_BIT ;Test if time for action
      goto loop              ;No, try again
      bcf opsys_var0, CYC_SYNC_BIT ;Yes, clear bit for next time
      clrwdt                ;Clear the watch-dog

      call adc_update        ;Perform analog to digital conversion

      movf ai_ch0+0,w        ;Get AI-0 low byte
      movwf psp_read_ary+0  ;Store it for host access
      movf ai_ch0+1,w        ;Get AI-0 high byte
      movwf psp_read_ary+1  ;Store it for host access

      movf ai_ch1+0,w        ;Get AI-1 low byte
      movwf psp_read_ary+2  ;Store it for host access
      movf ai_ch1+1,w        ;Get AI-1 high byte
      movwf psp_read_ary+3  ;Store it for host access

      goto loop             ;Repeat the loop
```

Function: add, div, mul

Revision: 1.00

Description: Mathematical functions. RAM locations are used as temporary storage for parameters.

add - performs unsigned addition on two 16-bit numbers

$$\mathbf{math_word0 = math_word0 + math_word1}$$

(Sum) (Addin #1) (Addin #2)

div - Divides a 16-bit dividend by a 16-bit divisor resulting in a 16-bit quotient and 16-bit remainder

$$\mathbf{math_word0 . math_word2 = math_word0 / math_word1}$$

(Quotient) (Remainder) (Dividend) (Divisor)

mul - Performs a 16-bit x 16-bit unsigned multiplication producing a 32-bit result

$$\mathbf{math_word2 , math_word0 = math_word0 x math_word1}$$

(Result HW) (Result LW) (Multiplier) (Multiplicand)

Function: dac_update

Revision: 1.00

Description: This function uses the values stored in RAM locations ao_ch0 and ao_ch1 to update the voltages produced by the two 12-bit analog output channels, AI-0 and AI-1 respectively. Both of these locations are 16-bits (1 word, 2 bytes) in size with only the lower 12 bits (B<0-11>) actually containing valid data. The unused bits should be written as zero for future compatibility. The IOCP-74 uses a LTC1446 for the digital-to-analog conversion. The convertor has an internal reference of 4.095 volts resulting in the 12-bit resolution (1 part in 4096) equaling approximately 1mv/step.

Conditions:

Before: Set ao_ch0 with a value (0 - 4095) resulting in an AO-0 output of 0 - 4.095vdc
Set ao_ch1 with a value (0 - 4095) resulting in an AO-1 output of 0 - 4.095vdc

After: None

Example Software: Read values entered from host and update analog outputs

```
main:
loop:  BANK0                ;Switch to bank-0

      btfss opsys_var0,CYC_SYNC_BIT    ;Test if time for action
      goto loop                       ;No, try again
      bcf opsys_var0, CYC_SYNC_BIT     ;Yes, clear bit for next time
      clrwdt                           ;Clear the watch-dog

      movf psp_write_ary+0,w           ;Get AO-0 low byte from host
      movwf ao_ch0+0                  ;Store it for conversion
      movf psp_write_ary+1,w           ;Get AO-0 high byte from host
      movwf ao_ch0+1                  ;Store it for conversion

      movf psp_write_ary+2,w           ;Get AO-1 low byte from host
      movwf ao_ch1+0                  ;Store it for conversion
      movf psp_write_ary+3,w           ;Get AO-1 high byte from host
      movwf ao_ch1+1                  ;Store it for conversion

      call dac_update                 ;Update the LTC1446 dual dac

      goto loop                       ;Repeat the loop
```

Function: gen_host_irq

Revision: 1.00

Description: Generates a hardware interrupt request to the host. The IOCP-74 interrupt logic must be properly configured for this function to work correctly. The host must issue a CMD#3 to clear the interrupt request before exiting its interrupt service routine. See the hardware and command register sections of this manual for more details.

Conditions:

Before: None

After: Sets the hardware latch whose output is routed through the IOCP-74 interrupt logic to the host.

Example Software: Periodically interrupt the host. at a 1hz rate.

```
main:
    movlw 76                ;Preload the counter modulus
    movwf loop_cntr        ;76 loops x 13.1ms/loop = 1hz

loop: BANK0                ;Switch to bank-0
    btfss opsys_var0,CYC_SYNC_BIT ;Test if time for action
    goto loop              ;No, try again

    bcf opsys_var0, CYC_SYNC_BIT ;Yes, clear bit for next time
    clrwdt                 ;Clear the watch-dog

;--- Test if time for host periodic intr
    decfsz loop_cntr      ;Dec the loop counter
    goto no_intr         ;Jump around if not zero

    movlw 76              ;Reload the counter modulus
    movwf loop_cntr
    call gen_host_irq     ;Generate the IRQ

no_intr:
    goto loop
```

Function: push_w_reg, pop_w_reg

Revision: 1.00

Description: Pushes (stores) and pops (retrieves) the contents of the w register using a software implemented virtual stack. These functions emulate the popular register push and pop instructions found in other processors but absent in the PIC16C74 microcontroller. It should be noted that these functions are simple in nature and have certain limitations such as not being able to be used in interrupt routines. They do however make temp storage and parameter passing easy and conserve scarce RAM locations. The size of the stack is set by an equate in the IOCPbios software. To make their use even easier two macros have been created, PUSHW and POPW.

Conditions:

Before: None

After: None

Example Software: Demonstrate push_w_reg and pop_w_reg using PUSHW and POPW macros.

```
main:
loop: BANK0                ;Switch to bank-0
    btfss opsys_var0,CYC_SYNC_BIT ;Test if time for action
    goto loop              ;No, try again

    bcf opsys_var0,CYC_SYNC_BIT ;Yes, clear bit for next time
    clrwdt                 ;Clear the watch-dog
    movlw 0xaa             ;Load w with a value

    PUSHW                  ;Save the w value on stack
    clrw                   ;Destroy the w contents
    POPW                   ;Retrieve the original value

    goto loop
```


Description: This function reads data from the IOCP-74 serial EEPROM. This type of memory is non-volatile so its contents are retained indefinitely (200+yrs) without requiring power. The device used (25C160) provides 16k bits of storage in a 2k x 8bit organization.

Conditions:

Before: Preload the first two bytes of spi_outbuf with the 16-bit starting address to be read. Spi_outbuf(0) will contain the low byte while spi_outbuf(1) contains the high byte. Since the memory is only 2k, the upper four bits (A12-A15) of the high byte are not used. However, they should be set to zero for future compatibility. Preload W with the number of bytes to read.

After: The EEMEM data is place in spi_inbuf in the order it was read. Spi_inbuf(0) contains the first byte.

Example Software: Read a word from the EEMEM. Place the data in the read array so the host can access and display it.

```
main:
loop:  BANK0                ;Switch to bank-0

      btfss opsys_var0,CYC_SYNC_BIT    ;Test if time for action
      goto loop                       ;No, try again
      bcf opsys_var0, CYC_SYNC_BIT     ;Yes, clear bit for next time
      clrwdt                           ;Clear the watch-dog

      ;--- Read a word from EEMEM
      movlw 0x00                       ;Preset starting EEMEM address Low byte
      movwf spi_outbuf+0
      movlw 0x00                       ;Preset starting EEMEM address High byte
      movwf spi_outbuf+1
      movlw 2                          ;Read 1 word (2 bytes) from EEMEM

      call rd_eemem                   ;Perform the EEMEM read operation

      ;--- Put the word in the read array for host access
      movf spi_inbuf+0,w               ;Get word low byte
      movwf psp_read_ary+0            ; and store it

      movf spi_inbuf+1,w               ;Get word high byte
      movwf psp_read_ary+1            ; and store it

      goto loop                       ;Repeat the loop
```

Function: rd_pic_adc

Revision: 1.00

Description: This function digitizes the voltage applied to the designated PIC analog input channel. The 8-bit result is returned in the W register. By default the analog to digital convertor uses the +5Vcc as the conversion reference voltage (approx. 19.61mv/step). Optionally the converter can be configured so that an external reference may be applied to the AI-4 (RA3/AN3/Vref) input. These analog inputs can alternately be used as digital I/O. The reader should refer to the PIC16C74 data sheet for details.

Conditions:

Before: Preload W with the channel to be digitized as follows:

Value	IOCP-74 Analog Input Channel	PIC name
0	AI-2	RA0/AN0
1	AI-3	RA1/AN1
2	AI-4	RA3/AN3/VREF
3	Disable PIC analog to digital converter (Saves power)	

After: W returns with 8-bit value

Example Software: Digitize the three 8-bit analog inputs and present the results to the host.

```
main:
loop:  BANK0                                ;Switch to bank-0

      btfss opsys_var0,CYC_SYNC_BIT        ;Test if time for action
      goto loop                             ;No, try again
      bcf opsys_var0, CYC_SYNC_BIT         ;Yes, clear bit for next time
      clrwdt                                ;Clear the watch-dog

      ;--- Read IOCP-74 AIN-2 (PIC AN0)
      movlw 0                               ;Set channel to digitize
      call rd_pic_adc                       ;Do call, w returns with value
      movwf psp_read_ary+0                 ;Store it for host access

      ;--- Read IOCP-74 AIN-3 (PIC AN1)
      movlw 1                               ;Set channel to digitize
      call rd_pic_adc                       ;Do call, w returns with value
      movwf psp_read_ary+1                 ;Store it for host access

      ;--- Read IOCP-74 AIN-4 (PIC AN3)
      movlw 2                               ;Set channel to digitize
      call rd_pic_adc                       ;Do call, w returns with value
      movwf psp_read_ary+2                 ;Store it for host access

      goto loop                             ;Repeat the loop
```

Description: This function sets up the IOCP-74 to receive characters via its asynchronous communication port. Once initialized, receptions occurs in the background using the PIC16C74 USART and interrupt driven software. As each character is received it is placed contiguously in the rx_comm_buf. The first character is placed in rx_comm_buf(0) the next in rx_comm_buf(1) and so on. Reception is terminated once a carriage return (ASCII 0x0d) is received, the buffer is full or an error condition occurs. Status bit are provided in comm_ctrl to determine the quality of the reception. Characters received after a normal reception are discarded.

Conditions:

Before: Call serial_rx to initialize hardware and software for serial receptions. Serial port parameters such as baud rate must have been previously configured.

After: The reception is complete when RX_ENABLE_BIT of comm_ctrl_reg is clear. If the reception was successful the characters will appear in rx_comm_buf in the order they were received. If an error occurred the offending condition can be determined by examination of bits within comm_ctrl_reg.

Example Software: Receive data and place in first five locations of read array for host access

```

main:
loop:  BANK0           ;Switch to bank-0

      btfss opsys_var0,CYC_SYNC_BIT ;Test if time for action
      goto loop        ;No, try again
      bcf opsys_var0,CYC_SYNC_BIT ;Yes, clear bit for next time
      clrwdt          ;Clear the watch-dog

      BANK1
      btfsc comm_ctrl_reg,RX_ENABLE_BIT ;Alreay receiving?
      goto loop       ;Yes, skip update and setup

      BANK1           ;Move the received data
      movf rx_comm_buf+0,w ; from the rx_comm_buf to the
      BANK0           ; read table for examination
      movwf psp_read_ary+0 ; by the host
      BANK1           ;For simplicity move only 5 characters
      movf rx_comm_buf+1,w ;Although more may be deeper in buffer
      BANK0
      movwf psp_read_ary+1
      BANK1
      movf rx_comm_buf+2,w
      BANK0
      movwf psp_read_ary+2
      BANK1
      movf rx_comm_buf+3,w
      BANK0
      movwf psp_read_ary+3
      BANK1
      movf rx_comm_buf+4,w
      BANK0
      movwf psp_read_ary+4
      BANK1
      movf rx_comm_buf+5,w
      BANK0
      movwf psp_read_ary+5
      call serial_rx ;Setup to do a reception
      goto loop     ;Repeat the loop

```

Description: This function causes an asynchronous communication from the IOCP-74 serial communication port.

Conditions:

Before: Ideally the TX_BUSY_BIT of comm_reg should be checked to see if a transmission is already in progress. Disturbing the transmit buffer, counter or hardware can effect any transmission already under way. Preload tx_comm_buf with the outgoing characters. Preload W with the number of outgoing bytes.

After: Transmission will begin if a previous transmission is not already in progress. W returns FALSE (0) if the transmission could not be started by the IOCP-74. If the transmission was started then W returns TRUE (1).

Example Software: This example sends the ASCII string ">Hello" to a terminal connected to the IOCP. The serial port is automatically configured using the IOCP default parameters so no additional setup is required.

```

main:
;-- Preload TX buffer with outgoing message
; Note: If same message is always sent, As in this example, the
; buffer really only needs to be loaded once. If different messages
; will be sent then the TX_BUSY_BIT should be first checked to verify
; that tx_comm_buf is free and can be safely overwritten.

BANK1          ;Switch to bank-1 where TX buffer is

movlw ">"      ;Load Each buffer location with one
movwf tx_comm_buf+0 ; character of the outgoing message
movlw "H"
movwf tx_comm_buf+1
movlw "e"
movwf tx_comm_buf+2
movlw "l"
movwf tx_comm_buf+3
movlw "l"
movwf tx_comm_buf+4
movlw "o"
movwf tx_comm_buf+5
movlw 0xd      ;Force a carriage return
movwf tx_comm_buf+6
movlw 0xa      ; line feed
movwf tx_comm_buf+7

loop: BANK0    ;Switch to bank-0

btfss opsys_var0,CYC_SYNC_BIT ;Test if time for action
goto loop     ;No, try again
bcf opsys_var0,CYC_SYNC_BIT ;Yes, clear bit for next time

clrwdt       ;Clear the watch-dog

movlw 0x08   ;Set # of outgoing bytes = 8, tx_comm_buf<0-7)
call serial_tx ;Start/attempt transmission


goto loop    ;Repeat the loop

```

Function: set_dig_dir

Revision: 1.00

Description: This function sets the IOCP-74 digital channels as input and output operations. Although the data directions can be configured by other means, this function consolidates and simplifies the process given that the IOCP-74 digital I/O uses bits from PORTB and PORTC of the PIC16C74. Before calling the **dig_io_dir** register must be loaded with the desired I/O configuration. Bits which are set will configure the corresponding digital I/O channel as an input. Setting a bit to zero will configure the corresponding digital I/O channel as an output.

 **NOTE:** Some digital channels are shared with special PIC functions. Care should be taken to assure that using this routine will not adversely affect how those functions operate..

Conditions:

Before: Preload **dig_io_dir** register with desired I/O configuration, 0 = Output, 1 = Input

dig_io_dir register bit assignments								
dig_io_dir	D7	D6	D5	D4	D3	D2	D1	D0
IOCP-74	DIO. 7	DIO. 6	DIO. 5	DIO. 4	DIO. 3	DIO.2	DIO.1	DIO.0
PIC Reference	RB7	RB6	RB5	RB4	RB3	RC2/CCP1	RC1/T1OSI/CCP2	RB0/INT

After: None

Example Software:

```
movlw 0xfc           ;Set DIO<0,1> as outs, DIO<2-7> as ins
movwf dig_io_dir     ;Preload dig_io_dir
call set_dig_dir     ;Perform setup
```

Function: setbaud

Revision: 1.00

Description: This function configures the USART circuitry for popular serial communication baud rates. It is most useful if the baud rate must be changed at run-time via software. Optionally, the user can load the SPBRG directly with any valid modulus to generate other standard and non-standard baud rates. The baud rates are based on the system 20mhz clock and affect both transmit and receive operations. For serial communications to work properly the IOCP-74, and any devices connected to it, must use the same communication parameter.

IOCP-74 default communication parameters are:

Baud : 9600 baud
Data bits : 8
Stop bits : 1
Parity : none

Conditions:

Before: Preload W with a value of 0 - 5 corresponding to the following standard baud rates:

Value	Baud Rate (Bits/sec)	Baud Rate Error
0	1200	Actual = 1221, +1.73%
1	2400	Actual = 2404, +0.16%
2	4800	Actual = 4808, +0.16%
3	9600	Actual = 9469, -1.36%
4	19200	Actual = 19531, +1.73%
5	38400	Actual = 39063, +1.73%

After: USART SPBRG register is loaded with the modulus for the desired baud rate.

Example Software: Load W with the correct modulus for baud rate#1 (2400 baud).

```
movlw 1 ;Request modulus for 2400 baud  
call setbaud ;Set the baud rate (load SPBRG)
```

Function: spi_select

Revision: 1.00

Description: All the devices sharing the SPI bus are individually enabled so that only the one device selected will respond to the transfer. The spi_select function makes the actual hardware selection process invisible to the programmer. After a transfer session is complete the device must be deselected. This is most easily done with the SPI_DESELECT macro.

Conditions:

Before: Load w with the value corresponding to the device to be selected.

IOCPbios Equate Symbol	Value	Device
SPI0_USER0	0	SPI select-0, Available for user circuitry
SPI1_USER1	1	SPI select-1, Available for user circuitry
SPI2_USER2	2	SPI select-2, Available for user circuitry
SPI3_USER3	3	SPI select-3, Available for user circuitry
SPI4_ADC	4	SPI select-4, Selects LTC1298 A/D
SPI5_DAC	5	SPI select-5, Selects LTC1446 D/A
SPI6_EEMEM	6	SPI select-6, Selects 25C16 EEPROM
SPI7_INTREQ	7	SPI select-7, Used to generate host IRQ

After: Users software must deselect the SPI device

Example Software: Select a SPI device

```
movlw SPI2_USER2      ;This device on spi bus will be activated
call spi_select        ;Activate the SPI device
```

Description: The IOCP-74 hardware supports SPI (Serial Peripheral Interface) communications which is controlled by this function. The communication is simultaneously bi-directional (full-duplex), as data is shifted out, data is also shifted in. Any data to be transmitted must first be placed in spi_outbuf with location zero being the first outgoing byte. The number of bytes to transfer must be preset by loading spi_count. A value of 1 will transfer 1 byte and so on. The device for which the transfer is intended must be selected (activated) before a transfer will be recognized. This is done by loading w before calling the function spi_select. There are four device chips selects available for user circuitry identified by values 0-3. Once these prerequisites have been satisfied the function spi_xfer is called to perform the actual transfer. After the transfer is complete the device must be deselected. This is most easily done using the macro SPI_DESELECT. Any data shifted in during the transfer will appear in spi_inbuf. Many SPI devices are input or output only making the data outputted or inputted (respectively) indeterminate.

Conditions:

Before: If outgoing data transfer required:

- load spi_outbuf with data, spi_outbuf+0 contains first outgoing byte
- Set spi_count to number of byte to transfer
- Load w and call spi_select to activate the SPI device

After: Users software must deselect the SPI device

If incoming data transfer required data:

- Read spi_inbuf for data, spi_inbuf+0 contains first byte shifted in.

Example Software: Send two bytes out using SPI

```

main:
loop: BANK0                ;Switch to bank-0
    btfss opsys_var0,CYC_SYNC_BIT    ;Test if time for action
    goto loop                        ;No, try again
    bcf opsys_var0, CYC_SYNC_BIT     ;Yes, clear bit for next time
    clrwdt                           ;Clear the watch-dog

    movlw b'00010000'                ;1st byte of outgoing data
    movwf spi_outbuf+0               ;Store in spi output buffer
    movlw b'00011100'                ;2nd byte of outgoing data
    movwf spi_outbuf+1               ;Store in spi output buffer


    movlw 2                          ;Set number of bytes to xfer
    movwf spi_count                  ;Preset counter
    movlw 0                          ;Device #0 of spi bus will be activated
    call spi_select                   ;Activate the SPI device
    call spi_xfer                     ;Do the xfer, 2 bytes out and 2 bytes in
    SPI_DESELECT                     ;Deselect all SPI devices

    ;--- Optional code to read bytes shifted in
    ; movf spi_inbuf+0                ;Read 1st byte shifted in
    ; *** user code to do something
    ; movf spi_inbuf+1                ;Read 2nd byte shifted in
    ; *** user code to do something

    goto loop                        ;Repeat the loop

```


Description: This function updates the IOCP-74 digital channels. Although the digital data can be written and read by other means, this function consolidates and simplifies the process given that the IOCP-74 digital I/O uses bits from both PORTB and PORTC of the PIC16C74. Before calling, the **dig_io_data** register must be loaded with the desired output data. All bits that are set will cause the corresponding channels configured as outputs to be set high. Similarly, bits which are zero will cause those outputs to be cleared. Bits corresponding to inputs are 'Don't care'. Once the **update_dig_io** routine writes the data it then reads the digital I/O channels and places the result in **dig_io_data**. The bits corresponding to outputs are 'refreshed' with the data just written while those corresponding to inputs reflect the state of the digital channel.

 **NOTE:** Because the IOCP-74 digital channels are spread across multiple PIC ports, not all the output channels will update simultaneously. Some applications, such as driving stepping motors, it is desirable that the outputs change at exactly the same moment. In those applications, it is suggested that the required outputs be grouped together from the same originating source, i.e. PIC PORTB.

Conditions:

Before: Preload **dig_io_data** register with desired output states. If only inputs are used, then this prerequisite can be skipped.

dig_io_data register bit assignments								
dig_io_data	D7	D6	D5	D4	D3	D2	D1	D0
IOCP-74	DIO.7	DIO.6	DIO.5	DIO.4	DIO.3	DIO.2	DIO.1	DIO.0
PIC Reference	RB7	RB6	RB5	RB4	RB3	RC2/CCP1	RC1/T1OSI/CCP2	RB0/INT

After: **dig_io_data** is updated with the state of the digital channels.

Example Software:

```


;--- Port directions initialized earlier in program using this code
movlw 0xfc          ;Set DIO<0,1> as outs, DIO<2-7> as ins
movwf dig_io_dir    ;Preload dig_io_dir
call set_dig_dir    ;Perform setup
|
|
|
|
|
|
|
bsf   dig_io_data,1 ;Set bit 1 (DIO.1)
call  update_dig_io ;Do writes and reads of digital channels

```

Function: wr_eemem

Revision: 1.00

Description: This function writes data to the IOCP-74 serial EEPROM. This type of memory is non-volatile so its contents are retained indefinitely (200+ yrs) without requiring power. The device used provides 16k bits of storage in a 2k x 8bit organization.

 **NOTE:** Care should be taken not to exceed the write cycle endurance capabilities of the memory. Although data can be rewritten ten million times, it wouldn't take long to reach this limit if the wr_eemem function was inadvertently placed in a program loop. It is also important to know that each write operation takes about 5ms. During this time the wr_eemem function suspends other IOCP-74 operations. In addition, only 16 bytes may be written at one time. The EEMEM also provides for data write protection but is not implemented in this version of the IOCPbios. More details can be found in the 25C160 manufactures data sheet.

Conditions:

Before: Preload the first two bytes of spi_outbuf with the 16-bit starting address to be read. Spi_outbuf(0) will contain the low byte while spi_outbuf(1) contains the high byte. Since the memory is only 2k, the upper four bits (A12-A15) of the high byte are not used. However, they should be set to zero for future compatibility. The data to be written is stored in spi_outbuf starting at offset 2. Preload W with the number of bytes to write.

After: None

Example Software: Write a word to EEMEM

```
main:
    BANK0
    ;--- Write a word to EEMEM
    movlw 0x00                ;Preset starting EEMEM address Low byte
    movwf spi_outbuf+0
    movlw 0x00                ;Preset starting EEMEM address High byte
    movwf spi_outbuf+1

    movlw 0xA7                ;Word value Low byte
    movwf spi_outbuf+2
    movlw 0x03                ;Word value High byte
    movwf spi_outbuf+3

    movlw 2                    ;Write 1 word (2 bytes) to EEMEM

    call wr_eemem              ;Perform the EEMEM write operation

loop: BANK0                    ;Switch to bank-0

    btfss opsys_var0,CYC_SYNC_BIT ;Test if time for action
    goto loop                  ;No, try again
    bcf opsys_var0, CYC_SYNC_BIT ;Yes, clear bit for next time
    clrwdt                      ;Clear the watch-dog
    ;--- No code to execute
    goto loop                  ;Repeat the loop
```

Appendix - A Host Interface Software

The IOCP-74 can be used with any host software which directly support hardware input and output operations. Languages such as Quick Basic and C inherently have instructions for performing these operations. Some software packages, such as Microsofts Visual Basic, do not and require supplemental routines (DLLs) to gain this functionality. Regardless of which host software is used the same basic approach is used: a block of data is moved from the IOCP-74 into the host (read) or a block of data is moved from the host into the IOCP-74 (write). The a data blocks organization and their content must be “understood and adhered to” on both sides. The easiest way to accomplish this is to have the host contain two arrays in its memory corresponding to the read and write arrays of the IOCP-74. In C, and some other languages, the arrays could actually be structures where each element of the array can be defined by a specific data type such as int, char or float.

The following listing is a QuickBasic program which illustrates these concepts. This program (CP_EX1.BAS) and its counterpart for the IOCP-74 module (CP_EX1.ASM) are included on the IOCP-74 distribution floppy disk. Other resources can be found a the SCIDYNE web site www.scidyne.com. In operation the program prompts the user for analog output values and reports back the analog input of the IOCP-74. The IOCP-74 module takes the words written to it and converts them to voltage outputs. The analog inputs are digitized and the results placed into the read table for host access and display.

```
'-----
' Copyright © 1998 SCIDYNE
'
' File: CP_EX1.BAS
' By: Mark W. Durgin
' Date: 11-03-98
' Revision: 1.00
' IOCP-74 Demonstration program
'-----
' This program demonstrates interfacing the IOCP-74 to a host
' running Microsoft QuickBasic. This program assume that the IOCP-74 has
' been programmed and is running the CP_EX1.ASM demonstration software.
'
' Note: This program was created and tested on a 25mhz 386SX embedded
' host computer. Some timing changes may be required to run on
' higher performance computers.
'-----
DEFINT A-Z                                'Unless otherwise declared, use integer variables
'-- Define program constants
CONST BASE.ADDR = &H300                    'Define where in IO map IOCP-74 is located
CONST data.reg = BASE.ADDR                'Data Array Register location
CONST cmd.reg = BASE.ADDR + 1             'Command Register location
CONST brd.reset = BASE.ADDR + 2          'Board Reset
CONST auto.inc.mask = 32                  'Automatic pointer increment bit
CONST ARRAY.SIZE = 16                    'Size of data arrays
'-- Declare arrays
DIM read.ary(ARRAY.SIZE)                 'Incoming data read from IOCP-74
DIM write.ary(ARRAY.SIZE)                'Outgoing data written to IOCP-74
```

```

'-- Initz the program
CLS                                'Clear the display
PRINT "-----" ' and show a banner
PRINT "    SCIDYNE"
PRINT "IOCP-74 Evaluation program"
PRINT "  CP_EX1.BAS  V1.00"
PRINT "-----"

'-- Reset the IOCP-74
' NOTE: This isn't actually required because a host power-on reset
' also generates an IOCP-74 reset. However, It is a good idea to
' include it during software development so that the IOCP-74 starts
' in a known state without having to reset the host.

OUT brd.reset, &HFF                'Generate a IOCP-74 hardware reset
PRINT "Hardware being reset ";      'Show user status and
FOR x = 0 TO 10                    ' wait for reset to take
  PRINT " ";
  FOR dly = 0 TO 25000: NEXT      'Software delay loop
NEXT
PRINT " Done"
PRINT

'-- This is the main loop
top:
PRINT "-----"

'-- Prompt user for analog output values
PRINT "---- Analog Outputs ----"

INPUT "AO-CH0: "; ao.ch0.val      'Input AO-CH0 value, 0-4095, 1mv/count
INPUT "AO-CH1: "; ao.ch1.val      'Input AO-CH1 value, 0-4095, 1mv/count

' Separate each analog output WORD into its low byte and high byte components.
' This is required because the IOCP-74 uses only an eight bit interface.
' Once separated, each byte is stored at its assigned positions
' within the write array of the host.

write.ary(0) = INT(ao.ch0.val MOD 256) 'AO-CH0 Low byte
write.ary(1) = INT(ao.ch0.val / 256)   'AO-CH0 High byte

write.ary(2) = INT(ao.ch1.val MOD 256) 'AO-CH1 Low byte
write.ary(3) = INT(ao.ch1.val / 256)   'AO-CH1 High byte

```

```

'-- Output the entire write array to the IOCP-74
OUT cmd.reg, 0 + auto.inc.mask                                'Set the IOCP-74 data array pointer to
                                                             ' location 0 and enable auto-inc
FOR index = 0 TO ARRAY.SIZE - 1                             'Loop for the number of array elements
  OUT data.reg, write.ary(index)                             'Send each BYTE
NEXT                                                         'Loop til done

'-- Input and copy the entire IOCP-74 read array into the host
OUT cmd.reg, 0 + auto.inc.mask                                'Set the IOCP-74 data array pointer to
                                                             ' location 0 and enable auto-inc
FOR index = 0 TO ARRAY.SIZE - 1                             'Loop for the number of array elements
  read.ary(index) = INP(data.reg)                             'Read and store each BYTE
NEXT                                                         'Loop til done

'-- Display to user each analog input value
PRINT : PRINT "--- Analog inputs ---"

'-- Display AI-0 (12-bit)
ai.word = read.ary(0) + (read.ary(1) * 256)                 'Make two bytes one word
PRINT "AI-0: "; ai.word

'-- Display AI-1 (12-bit)
ai.word = read.ary(2) + (read.ary(3) * 256)                 'Make two bytes one word
PRINT "AI-1: "; ai.word

'-- Display AI-2 (8-bit)
PRINT "AI-2: "; read.ary(4)

'-- Display AI-3 (8-bit)
PRINT "AI-3: "; read.ary(5)

'-- Display AI-4 (8-bit)
PRINT "AI-4: "; read.ary(6)

GOTO top
END

```

Appendix - B Specifications

I/O Peripherals:

Analog: Inputs: LTC1298, Two single-ended or one differential 12-bit (1 in 4096), ± 1 LSB, 0-5vdc, 11kHz max. PIC A/D, Three single-ended 8-bit (1 in 256), ± 1 LSB, Int./Ext. V_{REF} , 5vdc max. These three inputs can alternately be used as general purpose digital I/O.

Outputs: LTC1446, Two 12-bit (1 in 4096) ± 0.2 LSB, 0-4.095vdc, ± 5 mA

Digital: Eight digital I/O channels, CMOS levels, ± 25 mA. Pinout and drive compatible with I/O racks and other peripheral devices. Six "PORTB" bits provide software programmable pull-ups and change of state detection. Two "PORTC" bits provide alternate timer functions including Pulse-Width-Modulation, timer clock source and input/output capture/compare modes.

Memory: High speed non-volatile SPI EEPROM, 2k x 8-bit

Communications: PIC supervised serial asynchronous/synchronous communications. Software configured parameters. Supports 9-bit protocols and half/full duplex. Jumper selectable RS-232 or multi-drop RS-485

Addressing: Occupies 8 consecutive bytes in hosts I/O map. Jumper selectable between 0 through $0x3f8_{16}$

Interrupt: Software generated, supports sharing. Jumper selectable IRQ 3, 4, 5, 6, 7, 9, (10, 11, 12, 14 or 15)*

Processor: Socketed PIC16C74 (supports derivatives such as PIC16C77), 40 Pin DIP, 4K EPROM, 198 bytes RAM On-board 20mhz clock oscillator. Refer to Microchip® PIC16C74 data sheet for complete device information.

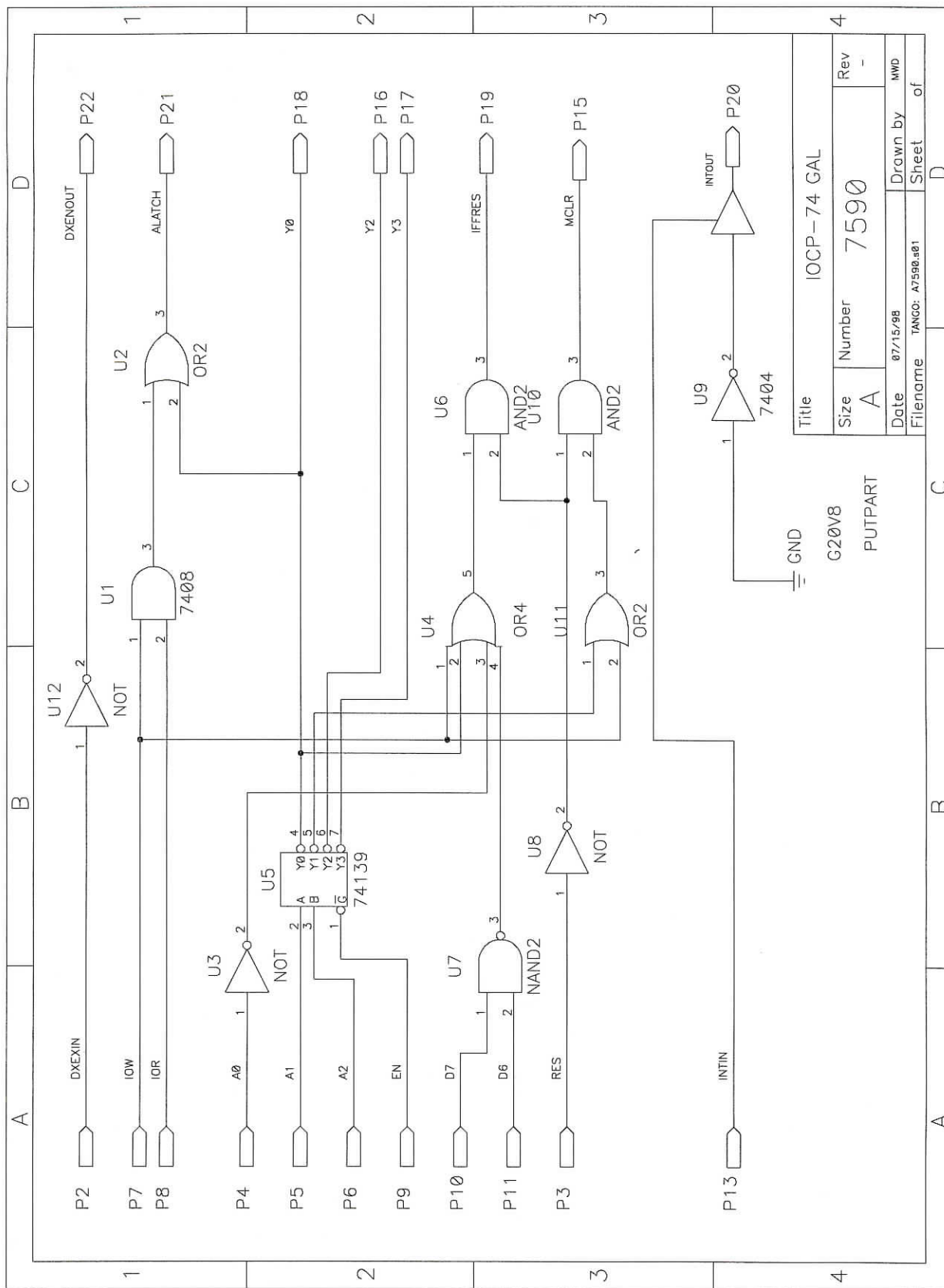
Prototyping area: 120+ solder-pad/wire-wrap holes, 0.036" dia., 0.1" spacing. One 24 SOIC pattern with wire access to pads. Access to: Logic power; PC/104 buffered D<0-7>, A<0-1>, IOR*, IOW*, Resets and 2 decoded I/O selects; Serial Peripheral Interface bus (SDI, SDO, SCLK) and 4 decoded SPI selects.

Power requirement: +5vdc $\pm 5\%$ @ 125mA typical. Unloaded outputs, user circuitry exempt.

Dimensions: PC/104 compliant, 3.55"W x 3.75"L x 0.6"H. 8-bit stack through. Holes for optional J2/P2 connector.

Environmental: Operating temperature: 0°C to 65°C Non-condensing relative humidity: 5% to 95%

Appendix - D GAL Schematic Diagram



Title		IOCP-74 GAL	
Size	Number	Rev	
A	7590	-	
Date	07/15/98	Drawn by	MWD
Filename	TANCO: A7590.a01	Sheet	of

G20V8
PUTPART