# USER MANUAL

# PC58

## 8 Channel Tracking Synchro to Digital or

## Resolver to Digital

## Converter Card

**LOW COST BOARDS FOR IBM PC, XT, AT, PS/2 AND COMPATIBLE COMPUTER SYSTEMS**

First edition

August 1992

August 1992 printing

# TABLE OF CONTENTS:

# PREFACE:

This manual is written for users of the PC58 series I/O cards. It provides all the necessary information required to successfully operate and program the PC58 series.

This manual assumes:

a) That you have a detailed knowledge of synchro and resolver operation.

b) That you are familiar with the PC environment.

c) That you are capable of writing your own programs or modifying the demonstration software for your own use.

## FEATURES:

- LOW COST

- SYNCHRO OR RESOLVER OPTIONS AVAILABLE

- ACCURACY OF 8.5 ARC MINUTES ( 12 BITS )

- MODULES HAVE MAXIMUM 1us CONVERSION RATE

- HIGH QUALITY VELOCITY OUTPUTS

- DIRECTION BIT PROVIDED

- 1 TO 8 CHANNELS

- HIGH TRACKING RATE TO 100 RPS

- EXCITATION FREQUENCIES FROM 50 Hz TO 2600 Hz

- SOFTWARE SUPPLIED FOR DRIVING 1 AND 2-SPEED SYSTEMS

- SEPARATE REFERENCE INPUTS FOR EACH CHANNEL

# 1.0 INTRODUCTION.

The PC58 is a versatile, full size IBM PC card designed for 1 to 8 channels of tracking Resolver to Digital or Synchro to Digital conversion.

Options for input voltages of $90V_{L-L}$, $11.8V_{L-L}$, or $2.5V_{L-L}$ are available. Options of frequency range vary from 50Hz to 2600Hz.

A high quality velocity output is provided for each channel which has a full scale voltage of ±10V. A direction bit is also provided for each channel, which is readable by software.

The card is double buffered and the 12 bit angle information is provided in two 8-bit bytes to the computer. The I/O addressing space is switch selectable with a DIP switch.

## 1.1 Software.

Complete Pascal and C software drivers and demonstration programs are provided. In addition to the normal single-speed software, two-speed driver software is provided to allow two-speed systems to be implemented.

## 1.2 Applications.

The multi-channel PC58 is designed for use in high performance control and simulation systems. With its very high tracking rates, the PC58 is an excellent choice for applications including motor control, antenna positioning, CNC machine tooling, robot axis control and process control.

## 1.3 Ordering Information.

**Numerous models are provided:**

Two SYNCHRO models are provided:

| MODEL | REF. VOLTAGE | L-L VOLTAGE | FREQUENCY |
|---|---|---|---|
| A | 115V | 90V | 50 – 2600Hz |
| B | 26V | 11.8V | 50 – 2600Hz |

Three RESOLVER models are provided:

| MODEL | REF VOLTAGE | L-L VOLTAGE | FREQUENCY |
|---|---|---|---|
| C | 2.5 V | 2.5 V | 360 – 2600 Hz |
| D | 26 V | 11.8 V | 50 – 2600 Hz |
| E | 115 V | 90 V | 50 – 2600 Hz |

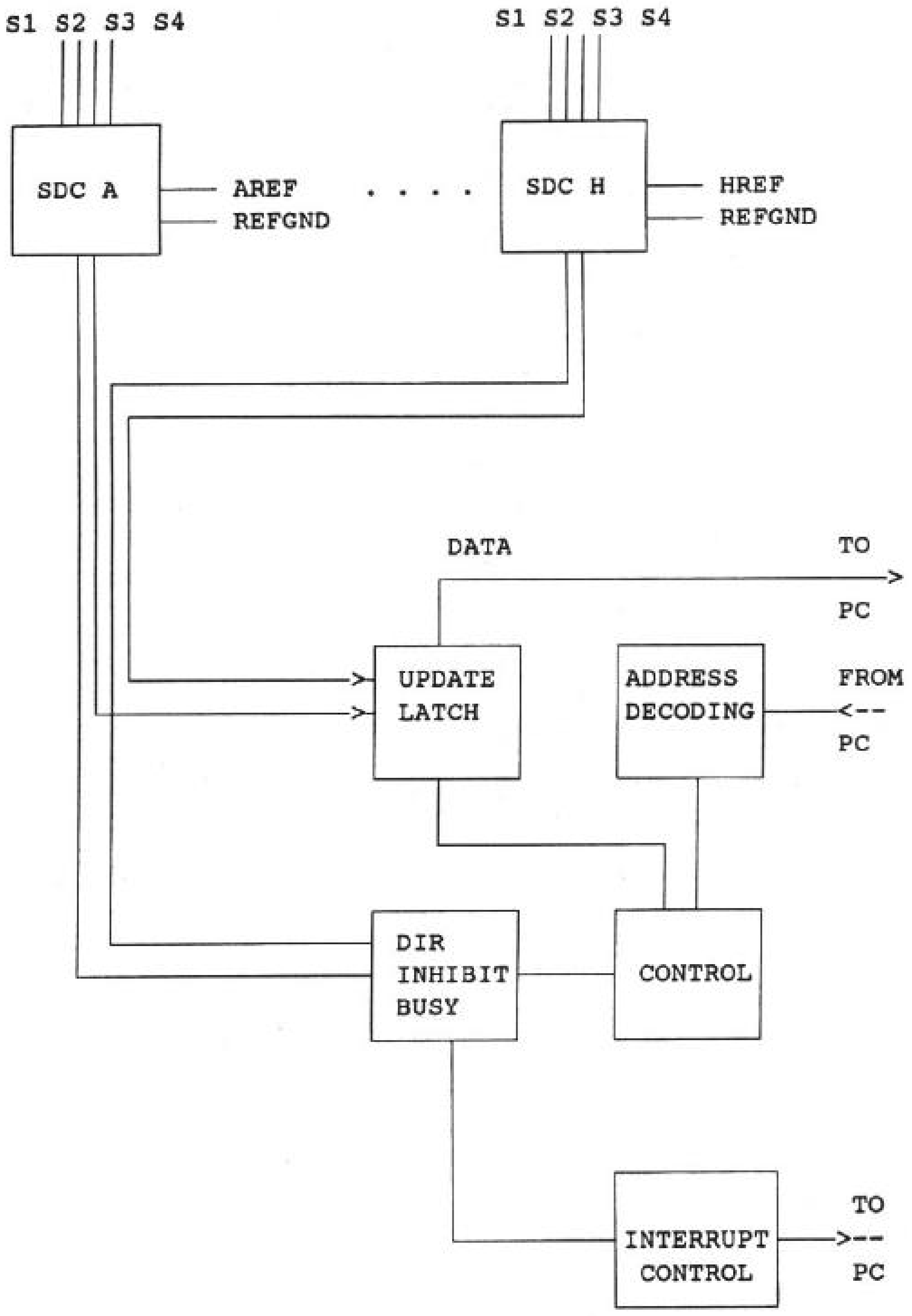
| TRACKING RATES | |
|---|---|
| 60 Hz | 20 RPS |
| 400 Hz | 50 RPS |
| 2 600 Hz | 100 RPS |

# 2.0 HARDWARE.

## 2.1 Architecture:

The PC58 is a full size IBM PC card which can accept up to eight channels of synchro or resolver to digital conversion channels. The maximum resolution is 12 bits, however, the resolution can be much improved if a two-speed system is set up with two single-speed systems and the appropriate software drivers.

The synchro or resolver signals are converted to digital angle information, which is updated at each equivalent least significant bit change of the shaft. The data is transferred via latches to the PC at a read instruction from the PC. It is arranged in two bytes per channel, therefore two reads from the PC will be required. A direction bit is provided so that the direction of each channel can be read into the PC. An inhibit latch is also provided to prevent a channel latch from being updated while it is being read. A BUSY status-latch can be polled  for busy status. The BUSY status is set when a channel has had a least significant bit change. The BUSY latch can be reset. An interrupt mode can also be set up to update the channels transparently.



BLOCK DIAGRAM OF PC58

| PC58 HARDWARE REQUIREMENTS | | |
|---|---|---|
| COMPUTER HARWARE REQUIREMENTS:<br>IBM PC/XT/AT or Compatible with 512k RAM | | |
| PARAMETER | UNITS | VALUE |
| POWER SUPPLY:<br>current consump. (5V) | A | 3.0 max. with 8 chan. |
| TEMPERATURE RANGE<br>operating temp.<br>storage | <br>°C<br>°C | <br>0 to 70<br>-55 to 125 |
| PHYSICAL ATTRIBUTES<br>full size IBM PC card | <br>in.<br>cm | <br>4.3 x 13.1 x 0.59<br>11.0 x 33.4 x 1.50 |

## 2.2 Addressing Requirements.

The PC58 card uses 20 I/O addresses and the base address is factory preset to be located at $320. This location may be adjusted by setting a DIP switch on the PC58 card. The address locations may be varied from $0000 to $0FE0 in steps of 32.

The address in decimal is set by the following formula:

```
A = 16 * ( SW1 * 128 + SW2 * 64 + SW3 * 32 + SW4
* 16 + SW5 *   8 + SW6 *  4 + SW7 * 2).
```

( ON = 1; OFF = 0)

| A = $320 | SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 |
|---|---|---|---|---|---|---|---|
| | OFF | OFF | ON | ON | OFF | OFF | ON |

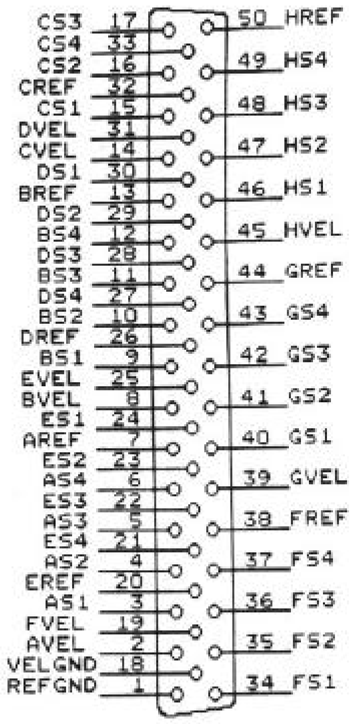
| eg. A = $700 | SW1 | SW2 | SW3 | SW4 | SW5 | SW6 | SW7 |
|---|---|---|---|---|---|---|---|
| | OFF | ON | ON | ON | OFF | OFF | OFF |

## 2.3 Interconnections.

### 2.3.1 Output Connector.

The output connector on the PC58 board is a **male** DB50 connector. The connections are shown below:

| | | | | | |
|---|---|---|---|---|---|
| 1 | REFGND | 18 | VELGND | 34 | FS1 |
| 2 | AVEL | 19 | FVEL | 35 | FS2 |
| 3 | AS1 | 20 | EREF | 36 | FS3 |
| 4 | AS2 | 21 | ES4 | 37 | FS4 |
| 5 | AS3 | 22 | ES3 | 38 | FREF |
| 6 | AS4 | 23 | ES2 | 39 | GVEL |
| 7 | AREF | 24 | ES1 | 40 | GS1 |
| 8 | BVEL | 25 | EVEL | 41 | GS2 |
| 9 | BS1 | 26 | DREF | 42 | GS3 |
| 10 | BS2 | 27 | DS4 | 43 | GS4 |
| 11 | BS3 | 28 | DS3 | 44 | GREF |
| 12 | BS4 | 29 | DS2 | 45 | HVEL |
| 13 | BREF | 30 | DS1 | 46 | HS1 |
| 14 | CVEL | 31 | DVEL | 47 | HS2 |
| 15 | CS1 | 32 | CREF | 48 | HS3 |
| 16 | CS2 | 33 | CS4 | 49 | HS4 |
| 17 | CS3 | | | 50 | HREF |



**Male DB50 connector as seen from rear of PC58.**

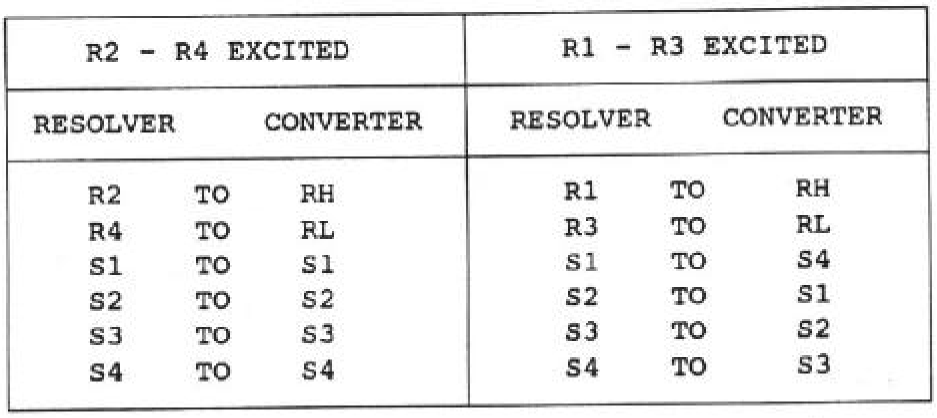
### 2.3.2 Pin Definitions.

Note that prefix A, B, C, D, E, F, G or H refers to the channel numbers 1, 2, 3, 4, 5, 6, 7 or 8 respectively.

a) **REFGND and xREF**: REFGND is the common reference point to which all REFLO points from each channel are connected. The xREF points are REFHI points and can be of different frequencies and voltages, depending on the modules. Therefore the reference sources must be connected to REFGND and AREF, BREF, etc. The reference voltage and frequency must be accurate to within ±10% of voltage and frequency to maintain the quoted accuracies of the converters. A solid state reference oscillator can be used. The ordering code is PC42.

b) **xS1**, **xS2**, **xS3**, and **xS4** are the inputs to the converters. These pins are connected to the synchro or resolver. For a synchro type, S1, S2, and S3 of the synchro go to S1, S2, and S3 of the converter. The rotor connections are R1 to xREF (REFHI) and R2 to REFGND (REFLO).

For a resolver, there are two conventions for connection to the modules:



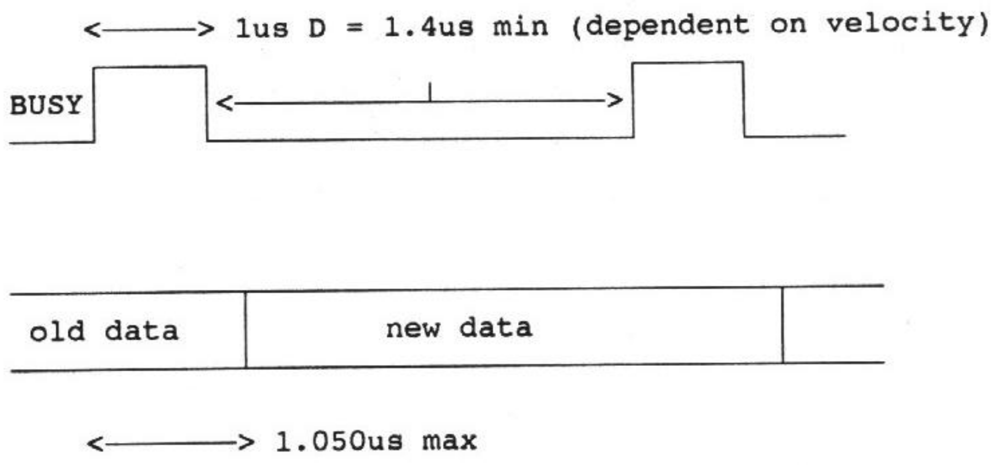
| R2 - R4 EXCITED | | | R1 - R3 EXCITED | | |
|---|---|---|---|---|---|
| RESOLVER | | CONVERTER | RESOLVER | | CONVERTER |
| R2 | TO | RH | R1 | TO | RH |
| R4 | TO | RL | R3 | TO | RL |
| S1 | TO | S1 | S1 | TO | S4 |
| S2 | TO | S2 | S2 | TO | S1 |
| S3 | TO | S3 | S3 | TO | S2 |
| S4 | TO | S4 | S4 | TO | S3 |

c) xVEL are the velocity outputs from the channels. The converter generates an analogue voltage proportional to the angular velocity of the shaft. The velocity output has a full scale voltage of ±10V. An external device can be connected to xVEL and VELGND.

## 2.4 Timing Requirements.

Up to a maximum of 1.050 us after the converter busy line has gone high, the data is available.

The converter ignores an inhibit signal if it is applied during an increment command.



```
                2. ♥
delay D = -----------    where RATE is the shaft velocity
          4096 * RATE
```

eg. The max. tracking rate for a 400Hz synchro on the PC58 is 50RPS.

```
        2.♥
D = ---------  = 30.7us
    4096 * 50
```

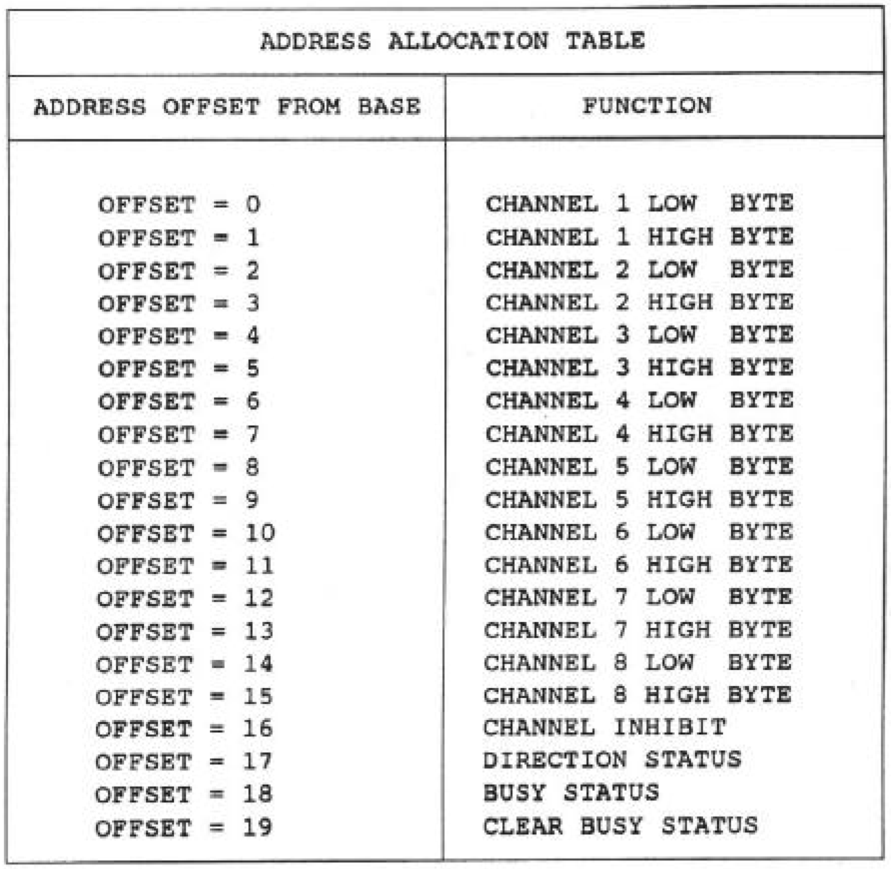
The PC58 uses high-speed components, thereby allowing it to be addressed with zero wait states.

# 3.0 SOFTWARE.

## 3.1 Programming the PC58.

### 3.1.1 Address Allocation:

The PC58 uses 20 consecutive address locations in the I/O space:



ADDRESS ALLOCATION TABLE

| ADDRESS OFFSET FROM BASE | FUNCTION |
|---|---|
| OFFSET = 0 | CHANNEL 1 LOW BYTE |
| OFFSET = 1 | CHANNEL 1 HIGH BYTE |
| OFFSET = 2 | CHANNEL 2 LOW BYTE |
| OFFSET = 3 | CHANNEL 2 HIGH BYTE |
| OFFSET = 4 | CHANNEL 3 LOW BYTE |
| OFFSET = 5 | CHANNEL 3 HIGH BYTE |
| OFFSET = 6 | CHANNEL 4 LOW BYTE |
| OFFSET = 7 | CHANNEL 4 HIGH BYTE |
| OFFSET = 8 | CHANNEL 5 LOW BYTE |
| OFFSET = 9 | CHANNEL 5 HIGH BYTE |
| OFFSET = 10 | CHANNEL 6 LOW BYTE |
| OFFSET = 11 | CHANNEL 6 HIGH BYTE |
| OFFSET = 12 | CHANNEL 7 LOW BYTE |
| OFFSET = 13 | CHANNEL 7 HIGH BYTE |
| OFFSET = 14 | CHANNEL 8 LOW BYTE |
| OFFSET = 15 | CHANNEL 8 HIGH BYTE |
| OFFSET = 16 | CHANNEL INHIBIT |
| OFFSET = 17 | DIRECTION STATUS |
| OFFSET = 18 | BUSY STATUS |
| OFFSET = 19 | CLEAR BUSY STATUS |

### 3.1.2 Register Functions.

```
OFFSET 0 TO 15 – ANGLE DATA – READ ONLY
```

Note that the data is **right justified** and hence the lower 8 bits are read from the LOW BYTE LATCH, whilst the upper 4 bits are read from the HIGH BYTE LATCH. See bit weights in the latter bit weight table.

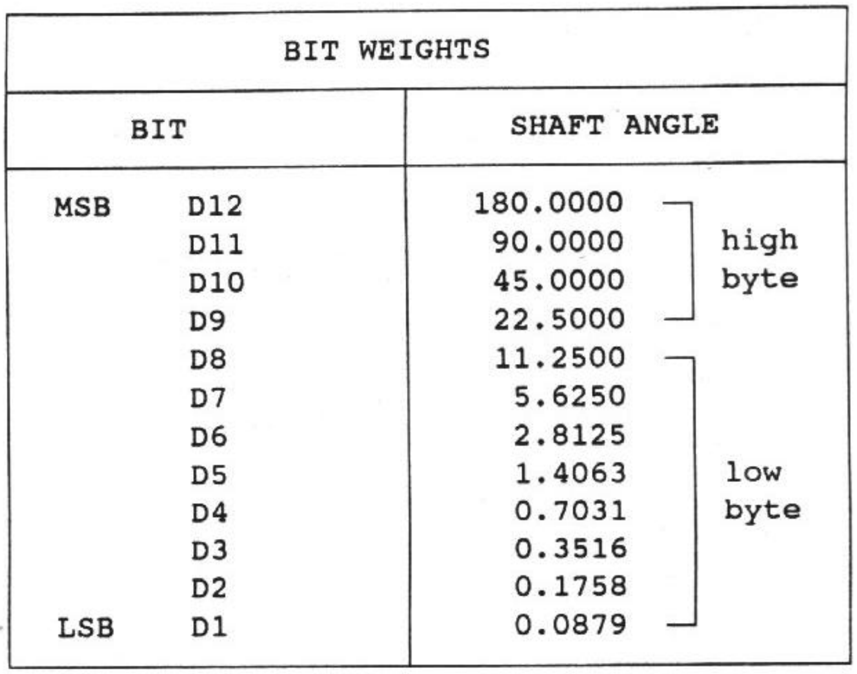
| OFFSET 16 - CHANNEL INHIBIT REGISTER - WRITE ONLY | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| BIT: | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CHAN. NUMBER: | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| BIT SET = INHIBIT<br>BIT CLEAR = ENABLE | | | | | | | | |

| OFFSET 17 - DIRECTION STATUS REGISTER - READ ONLY | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| BIT: | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CHAN. NUMBER: | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| BIT SET = COUNTING DOWN<br>BIT CLEAR = COUNTING UP | | | | | | | | |

| OFFSET 18 - CHANNEL BUSY STATUS - READ ONLY | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| BIT: | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CHAN. NUMBER: | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| BIT SET = BUSY | | | | | | | | |

| OFFSET 19 - CLEAR CHANNEL BUSY STATUS - WRITE ONLY |
|---|
| WRITE A '0' TO THE RELEVANT LOCATION |

### 3.1.3 Data Format.



| BIT WEIGHTS | | | |
|---|---|---|---|
| BIT | | SHAFT ANGLE | |
| MSB | D12 | 180.0000 | high byte |
| | D11 | 90.0000 | |
| | D10 | 45.0000 | |
| | D9 | 22.5000 | |
| | D8 | 11.2500 | low byte |
| | D7 | 5.6250 | |
| | D6 | 2.8125 | |
| | D5 | 1.4063 | |
| | D4 | 0.7031 | |
| | D3 | 0.3516 | |
| | D2 | 0.1758 | |
| LSB | D1 | 0.0879 | |

eg. An angle of 224 degrees would be 101000000000 binary or 2800 Hex.

## 3.2 Software Drivers and Demonstration Software.

### 3.2.1 Operation.

There are two modes of operation:

a) **Synchronous mode** where the data is read randomly.

b) **Asynchronous mode** where the data is read after each LSB shaft angle change.

a) For **synchronous mode**, the following algorithm is required:

INITIALIZE: SET INHIBIT = 0.

STEP 1: SET INHIBIT = 1.

STEP 2: WAIT MINIMUM OF 1us.

STEP 3: READ HIGH BYTE OF WHATEVER CHANNEL REQUIRED.

STEP 4: READ LOW BYTE OF WHATEVER CHANNEL REQUIRED.

STEP 5: SET INHIBIT = 0.

b) For **synchronous mode**, the following algorithm is required:

INITIALIZE: SET INHIBIT = 0.

WAIT FOR INTERRUPT OR POLL BUSY STATUS REGISTER FOR CONVERTER BUSY.

STEP 2: CHECK BUSY STATUS REGISTER TO DETERMINE WHICH CHANNEL IS READY.

STEP 3: SET INHIBIT = 1.

STEP 4: READ HIGH BYTE OF WHATEVER CHANNEL REQUIRED.

STEP 5: READ LOW BYTE OF WHATEVER CHANNEL REQUIRED.

STEP 6: SET INHIBIT = 0.

NOTE: THE ORDER OF BYTE RETRIEVAL IS IRRELEVANT IE. HIGH BYTE / LOW BYTE RETRIEVAL IS UNIMPORTANT.

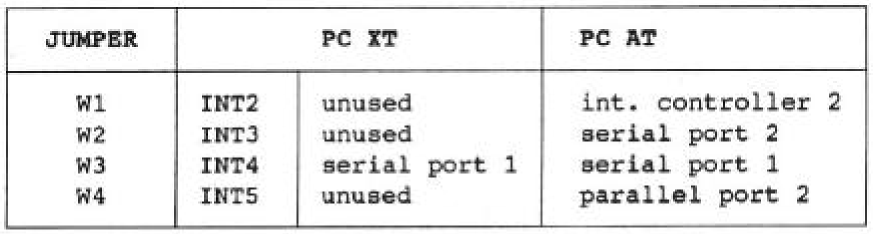
If all 8 channels are used, the above routines can be modified by merely looping the algorithms and reading the relevant channel on each iteration of the loop.

If fewer channels are used, the INHIBIT register can be set up to disable the unused channels.

**If interrupt mode is used, an interrupt jumper must be inserted in one of four positions to set up INT2, INT3, INT4 or INT5.**

**<u>Note that INT2 should only be used on PC XT systems</u>.**

The normal interrupt channel allocation is:



| JUMPER | PC XT | | PC AT |
|---|---|---|---|
| W1 | INT2 | unused | int. controller 2 |
| W2 | INT3 | unused | serial port 2 |
| W3 | INT4 | serial port 1 | serial port 1 |
| W4 | INT5 | unused | parallel port 2 |

The following formula can be used to convert the two bytes of each channel to a shaft angle:

$$\text{Shaft angle} = \frac{\text{HIGH BYTE} * 256 + \text{LOW BYTE}}{4096} * 360$$

[ shaft angle in degrees ]

Full software drivers for single and dual speed systems are provided. The drivers are written in Borland C++ and Turbo Pascal. Fortran and Basic drivers will be available shortly. Also included, is full demonstration software, which can be used in real applications.

### 3.2.2 Software Drivers.

A number of driver routines are provided in Turbo Pascal, Borland C++, and Microsoft QuickC on the accompanying disc. These can be converted to other versions of C and Pascal by modifying the interrupt functions. The driver files are broken up as follows:

|  |  |
|---|---|
| pc58.c | : the c drivers |
| pc58g.pas | : the general Pascal drivers |
| pc58i.pas | : the Pascal interrupt drivers |

If the interrupt drivers are not used, then the pc58i.tpu unit files can be omitted.

**The functions are:**

|  |  |
|---|---|
| set_base : | sets card base address. |
|  | set_num_chan : sets number of channels available on the card. |
|  | clears the channel array for initialization. |
|  | returns the status of a single channel. Returns True if the channel latch has been updated. |
| direction : | returns the direction of a single channel. '0' = counting up. |
| read_angle : | returns the angle of a single channel in degrees. |
| read_2speed_angle : | returns the angle of two channels in degrees for a two-speed system. The gear ratio must also be passed. |

| | |
|---|---|
| process_CB : | returns the status of the busy register. Returns True if the channel latch has been updated. Also automatically updates the channel array each time it is called. |
| combine_2_speed : | combines two channels for a two-speed system. The gearing ratio must also be passed. |
| I_read_angle : | returns the angle of a single channel in integer format. |
| I_to_R : | converts integer values to floating point values. |
| init_int : | initialises the interrupt routine. The interrupt routine automatically updates the channel array. |
| restate_old_int : | reinstates the old interrupt handler. |

**Channel array** is a global array and the data from the process_CB function and interrupt handler are stored in this array. Channels 0 to 7 are stored in sequence. The variable **num_chan** is also global and holds the number of channels to be processed.

**For C the function prototypes are:**

```
void set_base(unsigned int base_add);
    input parameters:base_add

void set_num_chan(int num);
    input parameters:num [1 to 8]
```

```
int  direction(int chan_num);
    input parameters:chan_num [0 to 7]

void clr_chan_array();
    input parameters:   none

int status(int chan_num);
    input parameters:chan_num [0 to 7]

float read_angle(int chan_num);
    input parameters:chan_num [0 to 7]

float read_2speed_angle(float ratio, int
chan_num1,
                   int chan_num2);
    input parameters:ratio
                    chan_num1 [0..7]
                    chan_num2 [0..7]

int process_CB();
    input parameters:none

float combine_2_speed(int ratio, int fine_chan,
                  int coarse_chan);
    input parameters:ratio
                    coarse_chan [0..7]
                    fine_chan [0..7]

int I_read_angle(int chan_num);
    input parameters:chan_num [0 to 7]

float I_to_R(int angle_in);
    input parameters:angle_in

void init_int(int int_num);
    input parameters:int_num [2, 3, 4, or 5]

void restate_old_int();
```

```
    input parameters: none
```

**Global variables:**

```
    int chan_array[8]
    int num_chan
```

When using the drivers, the include file **PC58.h** must be incorporated in the main source file. Please do not forget to create project files.

The correct function prototypes and calls for Microsoft QuickC and Borland C++ must be set up. This is done by setting the constant:

**#define lang QC** for Microsoft QuickC
or **#define lang BC** for Borland C++.

The project is then compiled. Most versions of Microsoft C and Borland C will also work with these settings.

**For Pascal the function prototypes are:**

```
procedure set_base(base_add : word);
    input parameters: base_add

procedure set_num_chan(num : integer);
    input parameters: num [1 to 8]

function direction(chan_num : integer) : boolean;
    input parameters: chan_num [0 to 7]

procedure clr_chan_array;
    input parameters:   none
```

```
function status(chan_num : integer) : boolean;
    input parameters:chan_num [0 to 7]

function read_angle(chan_num : integer) : real;
    input parameters:chan_num [0 to 7]

function read_2speed_angle(ratio : real,
chan_num1;                   chan_num2 :
integer) : real;
    input parameters:ratio
                     chan_num1 [0..7]
                     chan_num2 [0..7]

function process_CB : boolean;
    input parameters:none

function combine_2_speed(ratio : integer,
fine_chan;
                  coarse_chan : word) : real;
    input parameters:ratio
                     coarse_chan [0..7]
                     fine_chan [0..7]

function I_read_angle(chan_num : integer) :
integer;
    input parameters:chan_num [0 to 7]

function I_to_R(angle_in : integer) : real;
    input parameters:angle_in

procedure init_int(int_num : integer);
    input parameters:int_num [2, 3, 4, or 5]

procedure restate_old_int;
    input parameters:none
```

**Global variables:**

```
    chan_array[0..7] of integer;
    num_chan : integer;
```

When using the drivers, the unit **PC58g.tpu** must be incorporated in the main source file. If the interrupt drivers are used, then the include file **PC58i.tpu** must also be incorporated in the main source file.

**Note I:** The PC58i.pas unit is written in Turbo Pascal and will have to be modified and recompiled for Microsoft Pascal.

**Note II:** If an invalid channel number is read with read_angle or read_2speed_angle, -1 is returned.

### 3.2.3 Sample Programs.

Six sample programs are provided in Turbo Pascal and Borland C++. These can be converted to other versions of C and Pascal by modifying the clrscr functions, gotoxy functions, and delay functions to equivalents. The 'exe' files are also provided.

Programs for single speed systems:

PC58eg1.c and PC58eg1.pas : synchronous system
PC58eg2.c and PC58eg2.pas : asynchronous system
PC58eg3.c and PC58eg3.pas : interrupt driven

Programs for two speed systems:

PC58eg4.c and PC58eg4.pas : synchronous system
PC58eg5.c and PC58eg5.pas : asynchronous system
PC58eg6.c and PC58eg6.pas : interrupt driven

The 'exe' files for PC58eg1 to PC58eg6 are also provided.

**Program Description:**

**PC58eg1.X** reads all the channels synchronously and prints the angles on the screen. **The read_angle** function is used to read in the data.

**PC58eg2.X** reads all the channels asynchronously and prints the angles on the screen. The **process_CB** function is used to poll the busy status register for a latch update and automatically update the relevant locations in the channel array. The **I_to_R** function is used in the main program to convert the integer data from the channel array to floating point angle values.

**PC58eg3.X** reads all the channels asynchronously using interrupts and prints the angles on the screen. The interrupt handler is used to update the channel array transparent to the main program. The **I_to_R** function is used in the main program to convert the integer data from the channel array to floating point angle values.

**PC58eg4.X** reads all the channels synchronously and prints the angles on the screen. The **read_2speed_angle** function is used to read in the data.

**PC58eg5.X** reads all the channels asynchronously and prints the angles on the screen. The **process_CB** function is used to poll the busy status register for a latch update and automatically update the relevant locations in the channel array. The function **combine_2_speed** is then used to combine two channels for the two speed angle data.

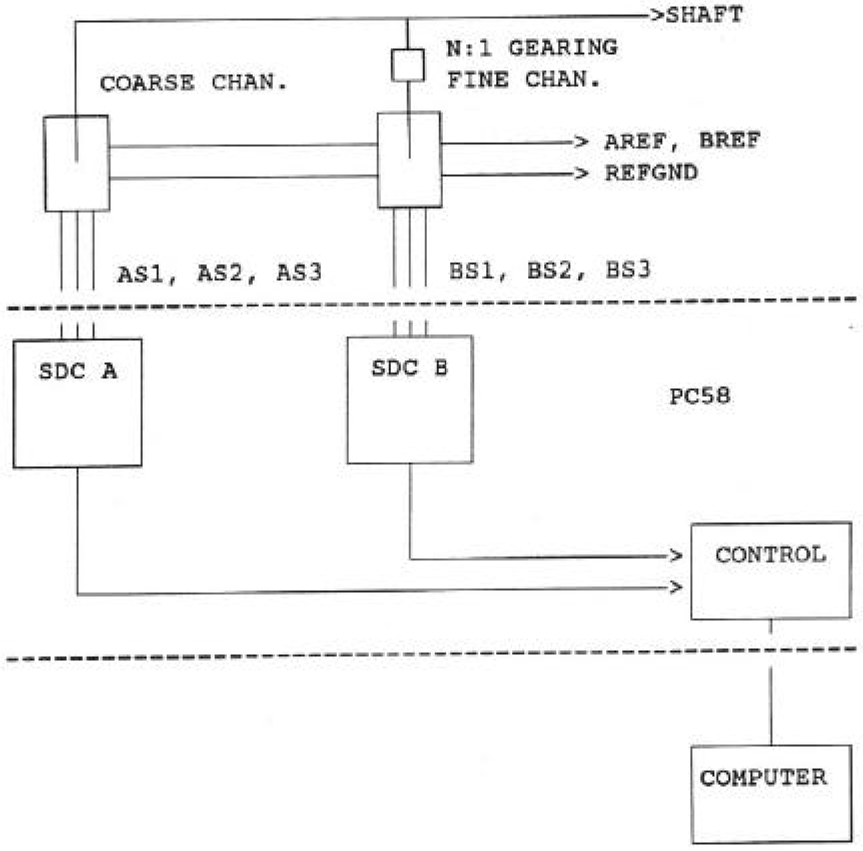
**PC58eg6.X** reads all the channels asynchronously using interrupts and prints the angles on the screen. The interrupt handler is used to update the channel array transparent to the main program. The function **combine_2_speed** is then used to combine two channels for the two speed angle data.

**Note I:** Do not forget to set up the project files in C.

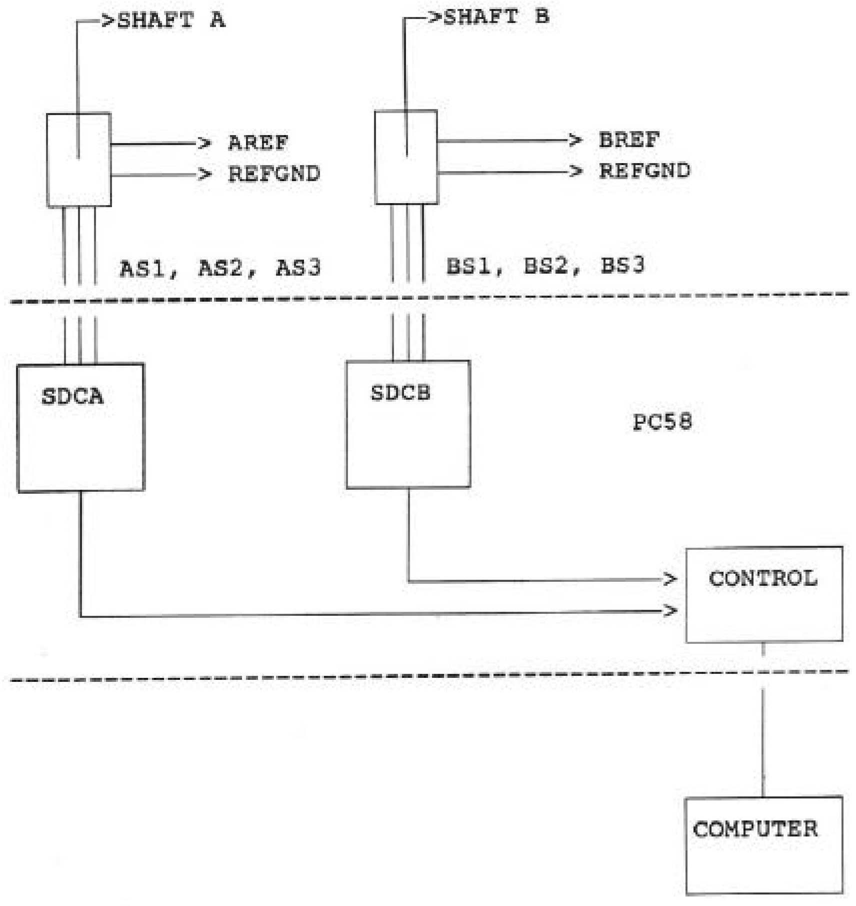
**Note II:** In all the two speed systems, the fine channel is the first channel and the coarse channel is the following channel.

## 4.0 Single and Dual Speed System Setup.

**Dual speed system:**



## 2-Channel Single Speed System



## BIBLIOGRAPHY:

1) **SYNCHRO CONVERSION : ILC DATA DEVICE CORPORATION, 1990.**

2) **SYNCHRO AND RESOLVER CONVERSION : ANALOG DEVICES, 1980.**

## APPENDIX A

## SPECIFICATIONS

# APPENDIX B

## APPLICATIONS

## (From Synchro & Resolver Conversion - Analog Devices)