Instructions for 16 line programmable Input/Output Port - TE 10

Note:- Always disconnect power supply to the ZX81/80 before plugging in the I/O Port.

INTRODUCTION

The Sinclair ZX81/80 computer has brought the cost of computing down to a price within reach of the hobbyest,home enthusiast & educational establishments. The computer & its Sinclair add on's provide an exceptional value for money system, however, the system is inherently restricted to handling information generated from within, or at best from externa cassette. With the addition of an Input/Output Port the user is able to input & output signals TO & FROM the outside world, giving a whole new dimension of uses for the ZX81/80 computer.

The Port arrangement.

The port utilises an encased double sided printed circuit board. Protruding from the front of the unit is a 23-23 way edge connector. This connector plugs directly into the rear of the ZX81/80. On the rear of the port is a 23-23 way plug, allowing the port to be used in conjunction with the 16K RAM pack & printer. Note :- It is not essential to use either the RAM pack or the printer to operate the port. The Input/Output to the port is via a 21-21 way plug, which protrudes from the left hand side of the unit, carrying the 16 Input/ Output lines.

Addressing The Port

As an introduction to addressing the port, it is suggested that the user re-reads chapters 24/25 of the Sinclair manual, it would also be beneficial (for the advanced user) to read chapter 26 but is not absolutly necessary to understand it.

The port has 16 fully programmable Input/Output lines, which are divided into two 8 line ports, A & B. In order to transfer data to & from the unit, each port, A & B is required to have an address. In practice because of the versitility of the port, four addresses are required. The TE 10 port is what is known as a 'Fully Decoded Port', what this means is that unlike most other ports it may be used with virtually any other add on, Sinclair or otherwise, ie, 16K, 32K, 64K RAM pack, full keyboards, printers, colour graphics boards etc. Because the port is fully decoded it is necessary to access it by a very short & simple machine code program, which is held in a BASIC REM statement, as we shall see later, (Note :- You do not have to understand machine code to use the port.)

The port is based on the powerful Z80A PIO integrated circuit, which has many features & modes of operation, however, we need only consider its use in a straight forward Input/ Output manner. Since the device is fully programmable, it is necessary first of all to inforn the two 8 line ports, A & B that they are to be used in Input or Output mode. This is done by writing a code to the STATUS registers within the chip, the address of these registers is as follows :-

PORT	A	111
PORT	B	127

The codes that have to be sent to the STATUS registers to determine whether Input or Output mode is required are :-

INPUT mode	79
OUTPUT mode	15

eg : Let us assume port A is required to be in output mode, then 15 is sent the port A status register whose address is 111.

This chip has two other registers for both ports A & B. When a port is in input mode its data register will contain the data that has been read in from the input lines. When a port is in output mode its data register will contain the data that is to be sent to the output lines. The address of the data registers are :-

PORT A	79
PORT B	95

SUMMARY :- We have seen that the Z80A PIO has four registers built into it, two for each port (A&B), one status register & one data register. We have also seen that codes need to be sent to the status registers, so that the computer knows whether Input or Output mode has been selected for each port. Fig 1 shows one of four possible modes that the port can adopt :- A in / B out, other possible modes of operation are :- A in / B in, A out / B out, A out / B in.

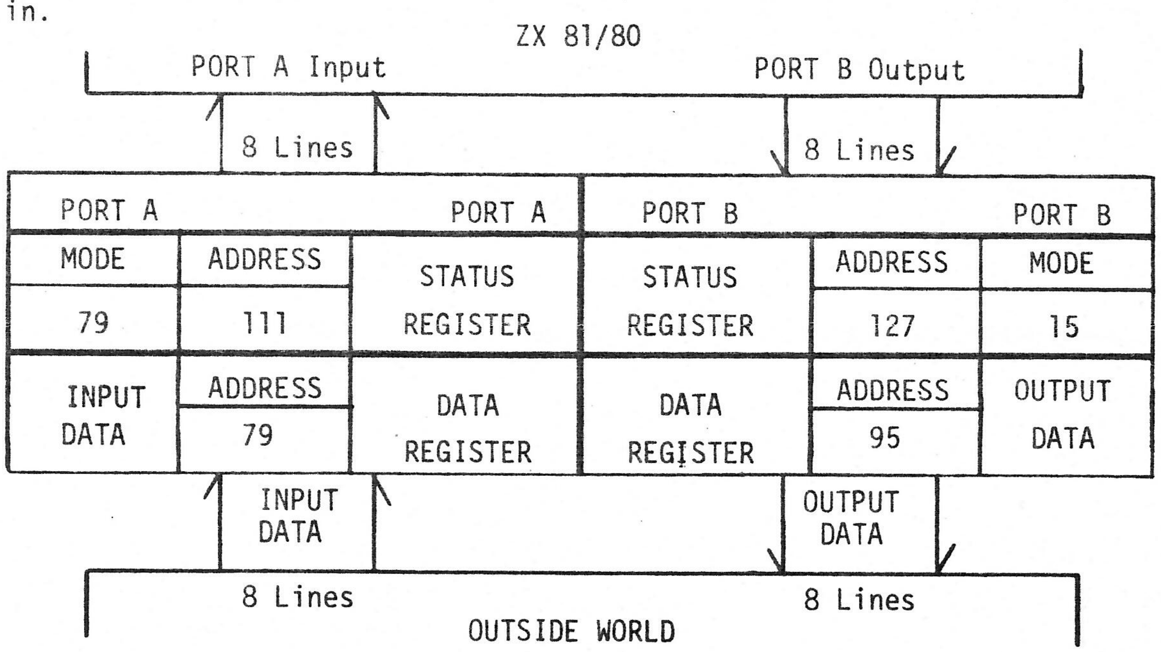


Fig 1

How do we communicate with the port.

As was mentioned earlier a short simple machine code program is necessary to access the Port. This program is contained in one REM & one poke statement, which must be the first two lines of any basic program. ie :-

```
1  REM  ▆space  Y P PEEK  P Y P : P * P TAN
        1   2    3 4  5    6 7 8 9 10 12  13       Character number in line 1
2  POKE 16524,237                      11
```

This looks basically meaningless, however it has put into memory the shell of a machine code program which will control the Port. The "P's" that occur in the REM statement represent 'POKE's' for the four registers' addresses & codes which need to be POKEd in to determine the port's mode of operation, ie as shown in Fig 1, A in / B out.

cont-

This can be explained with reference to the following table :-

Explanation of REM characters.

Character No's	Characters	Characters to be POKEd in REM in place of "P's".	Memory Address
1	■	------	16514
2	SPACE	------	16515
3	Y	------	16516
4	P	79 or 15 Input or Output mode	16517
5	PEEK	------	16518
6	P	111 or 127 A or B status Reg address	16519
7	Y	------	16520
8	P	Output data (if in output mode)	16521
9	:	------	16522
10	P	79 or 95 A or B Data Reg address	16523
11	*	------	16524
12	P	72 or 121 (Input or Output)	16525
13	TAN	------	16526

Fig 2

We have omitted to state the reason for POKing 72 or 121 (Input or Output) in character line No 12, it is sufficient to say that this modifies the machine code program depending whether we are operating in Input or Output mode, ie for character No 12:-

```
INPUT (M/C)    72
OUTPUT(M/C)    121
```

It is important to note that the REM & POKE lines 1 & 2, have to be the first two lines in any BASIC program that is accessing the port. This is essential as they must not move around in memory.

How do we put the Port into operation ?

Let us type in the following example :-

EXAMPLE 1 To Output the decimal number 5 to port A.

Line Number

```
 1   REM ■SPACE Y P PEEK  P Y P : P * P TAN
 2   POKE 16524,237                           These two lines never change.

10   POKE 16517,15                            Output mode programmed.
20   POKE 16519,111                           PORT A status register address.
30   POKE 16521,5                             Output data (Decimal 5).
40   POKE 16523,79                            Port A data register address.
50   POKE 16525,121                           OUTPUT.

60   LET X=USR 16514                          Executes the program.
```

Note :- When typing in the REM statement there are no spaces between the characters, except after REM & PEEK characters & the ZX81/80 puts these in. SPACE is a character.

If you have difficulty finding the characters for the REM statement, try a combination of the SHIFT / GRAPHICS & FUNCTION keys.(See pages 31 & 37 of the Sinclair manual.)

Having RUN this program we find "5" will appear on port A outputs (This can be checked using LED's (lamps) or via the purpose built 8 WAY indicator unit TE 15. The output will of course be in binary ie, 00000101.

The variable 'X' in the execution line 40 is used as a dummy variable, when a port is used in Output mode, however when in input mode 'X' contains the input data, read in through the the port.

Note :- After executing the above program, the REM statement in line 1 will change to 1 REM ■SPACE Y ? PEEK ? Y ■ : ? GOSUB ? TAN This is due to the POKE statements in lines 10 to 30 inc.

Example 2   To Input through Port B.

We first need to set up data to be read in, on the Input lines, this of course will be in the form of logical 0's & 1's ie,0v & +5 volts, our purpose built 8 way switch unit TE17 is ideal for this purpose.

Let us assume the binary data 00010101 (decimal 21) is set up on the port B lines. The program is as follows:-

```
1    REM ■SPACE Y P  PEEK  P Y P : P * P TAN} Remember NO spaces between characters
2    POKE 16524,237                          } These two lines never change.
10   POKE 16517,79                             Input mode programmed.
20   POKE 16519,127                            Port B status register address.
30   REM This line not required in Input mode.
40   POKE 16523,95                             Port B data register address.
50   POKE 16525,72                             Input.

60   LET X=USR 16514                           Executes program.
70   PRINT X                                   Displays data read in, on the screen.
```

The number 21 will now appear on the screen. Note: line 20 is not required as we are in Input mode.

Accessing the Port form within a main program.

In order to access the port more than once from within a main program it is recommended that a subroutine be written. This allows the main program to call the subroutine as and when access is required to the Port.

Example 3   PORT A in output mode using a subroutine.
                   'Outputting 129 to Port A'

```
1    REM ■SPACE Y P  PEEK  P Y P : P * P TAN}
2    POKE 16524,237                          } These two lines never change.

100  REM Main Program.
110  LET DATA 1=129                            Binary 10000001
120  GOSUB 1000

130  STOP

1000 REM Subroutine to output data to port A.
1010 POKE 16517,15                             Output mode programmed.
1020 POKE 16519,111                            Port A status address.
1030 POKE 16521,DATA1                          Output data.
1040 POKE 16523,79                             Port A data register address.
1050 POKE 16525,121                            Output.

1060 LET X=USR 16514                           Execute program.
1070 RETURN
```

Writing four such routines, one for each combination of:- Port A/B IN/OUT would allow a main program to move data in and out of the ZX81/80 with ease i.e.:-

Port A in Output mode.
Port A in Input mode.
Port B in Output mode.
Port B in Input mode.

Such subroutines would be as follows:-

PORT A in Input mode.

```
2000  REM Subroutine to Input data through Port A.
2010  POKE 16517,79                            Input mode programmed.
2020  POKE 16519,111                           Port A status register address.
2030  REM This line not required in Input mode.
2040  POKE 16523,79                            Port A Data register address.
2050  POKE 16525,72                            Input.
2060  LET DATA2=USR 16514                      Executes program.
2070  RETURN.
```

PORT B in Input mode.

```
3000  REM Subroutine to input data through Port B.
3010  POKE 16517,79                              Input mode programmed.
3020  POKE 16519,127                             Port B status register address.
3030  REM  This line not required in Input mode.
3040  POKE 16523,95                              Port B data register address.
3050  POKE 16525,72                              Input.
3060  LET DATA3=USR 16514                        Executes program.
3070  RETURN
```

The data read in by the above subroutines will be held in DATA2 & DATA3, simply putting a PRINT statement in the main program will display the data read in ie.

PRINT DATA2

PORT B in output mode.

```
4000  REM Subroutine to Output data to Port B
4010  POKE 16517,15                              Output mode programmed.
4020  POKE 16519,127                             Port B status register address.
4030  POKE 16521, DATA4                          Output data.
4040  POKE 16523,95                              Port B data register address.
4050  POKE 16525,121                             Output.
4060  LET X=USR 16514                            Executes program.
4070  RETURN
```

Note:-

Of course if it is required to use the Port in one of its most simplest forms ie:-

16 Lines Input (A & B) or 16 Lines Output (A & B out)
or 8 Lines Input + 8 Lines OUtput (A in & B out, or vica versa)

Then a simple combination of the above subroutines will accommodate your needs.

EXAMPLE 4 You need the 16K RAM pack to RUN this program.

This program reads in a number from PORT A displays it on the screen & then outputs it to PORT B. It is first necessary to set up a binary number on the Port A input lines, switch unit TE 17 is ideal for this purpose. A method of displaying the Port B output lines is also required, indicator box TE 18 is ideal for this purpose.

```
1      REM  ■SPACE Y P  PEEK  P Y P : P * P TAN
2      POKE 16524,237
5      REM MAIN PROGRAM
10     GOSUB 2000
20     PRINT "We have now read in the following data from Port A"
30     PRINT
40     PRINT DATA2
50     LET DATA4=DATA2
60     GOSUB 4000
70     PRINT
80     PRINT "We have now output the data read in from Port A to Port B, which was"
90     PRINT
100    PRINT DATA4
110    STOP

2000   REM Subroutine to input data through Port A.
2010   POKE 16517,79
2020   POKE 16519,111
2030   REM Line not required in input mode.
2040   POKE 16523,79
2050   POKE 16525,72
2060   LET DATA2=USR 16514
2070   RETURN
```

```
4010    POKE 16517,15
4020    POKE 16519,127
4030    POKE 16521, DATA4
4040    POKE 16523,95
4050    POKE 16525,121
4060    LET X=USR 16514
4070    RETURN.
```

Note:- It is not necessary to number the lines as we have done, you may put the program and subroutines anywhere in memory you like, except lines 1 & 2 which must always be the first two lines in the program.

Are you interested in machine code ?

Once you have mastered the Port you may wish to read the following section, it should be noted however, that it is not necessary to understand any of this to use the Port, it is given as information only.

The REM statement in line 1 and the POKE statement in line 2 form the shell of the machine code program. These characters put into memory codes which the processor will execute directly, namely machine code. Looking at the table below in conjunction with pages 181 to 187 of the Sinclair manual, we see how the machine code program is derived.

Address	Character	Decimal code	Hexadecimal code	Machine code meaning
16514	▚	6	06	LD B,00
16515	SPACE	0	00	
16516	Y	62	3E	LD A,??
16517	P	53	Mode to be POKEd in.	
16518	PEEK	211	D3	OUT A,??
16519	P	53	Status reg address to be POKEd in.	
16520	Y	62	3E	LD A,??
16521	P	53	Output data to be POKEd in.	
16522	:	14	0E	LD C,??
16523	P	53	Data reg address to be POKEd in.	
16524	*	237	ED - See line 2	OUT (C),A
16525	P	53	Input/Output	IN (C),C
16526	TAN	201	C9	RET

FIG 3

Line 2 POKE 16524,237. This line simply puts in ED H into 16524 as this cannot be done from within the REM statement.

The machine code program, when say Port A is in Output mode is:-

```
        LD B,00        Clears register B
        LD A,0F        Output mode loaded into reg A
        OUT A,6F       Reg A OUT to port A status reg.
        LD A, DATA     Reg A loaded with data (out mode)
        LD C,4F        Reg C loaded with A data reg address.
FIG 4   OUT (C),A      Reg A out to address in reg C.
        RET            Return to BASIC program.
```

See pages 167 of the Sinclair manual. When a Port is in Input mode, the incoming data is loaded into register C, register B has been set to 0 (zero) on returning to a basic program the contents of the BC register pair are loaded into the variable in the USR command. ie:

LET X=USR 16514

On returning to BASIC, X will contain the contents of register BC. Note:- BC is a 16 BIT register pair, B is the high order 8 BITS and C the low order 8 BITS and since the 8 lines of a port have been read into register C and register B is zero, the variable 'X' will contain the data read in from the port.

If you wish to check that the machine code program is in the memory, after the REM statement and POKE statements in lines 1 & 2 have been executed, then run this program.

```
5010  FOR N = 16514 TO 16526
5020  PRINT N,PEEK N
5030  NEXT N
```

This program will list the decimal code of the machine code program, see FIG 3.

For the advanced user

The Z80A PIO chip is an extremely versatile Input/Output controller. It has many features not covered in these notes, including interupt driven data transfer for high speed, a Control mode, which allows all 16 lines to be defined individually as I/O lines. Handshake lines are also available on the output plug of the port. These lines RDY & $\overline{\text{STB}}$ allow interupt driven communication between the ZX81/80 and the peripheral device. The RDY line goes high in Output mode when valid data has been loaded by the ZX81/80 into the port. It also goes high in Input mode, when the input data loaded into the port has been read by the ZX81/80. The $\overline{\text{STB}}$ lines are taken low, when in output mode by the peripheral device to acknowledging the receipt of data. It is also taken low in Input mode, by the peripheral device to indicate to the ZX81/80 that the data on the input lines is valid. Both the RDY & $\overline{\text{STB}}$ lines have other uses in connection with bedirectional and control modes.

A 9 page data sheet is available TE 1N, see catalogue and price list.

FAULT DIAGNOSIS

1) I/O port connected & cursor does not appear.

   A) If the port was bought, assembled and tested, check contacts are clean in the Sinclair & the port. Use contact cleaner TE 1S, see catalogue.

   B) If the port was bought in kit form, check for solder splashes shorting tracks. Check the Z80A PIO is in the correct way round, pin 1 is the bottom left hand pin & its position is also marked on the board. Check the diodes are the correct way round, the line on them is the cathode and this is marked by a 'C' on the board. Check contacts are clean as in A above.

2) The cursor appears when the port is in on its own, but not when the RAM pack is on.

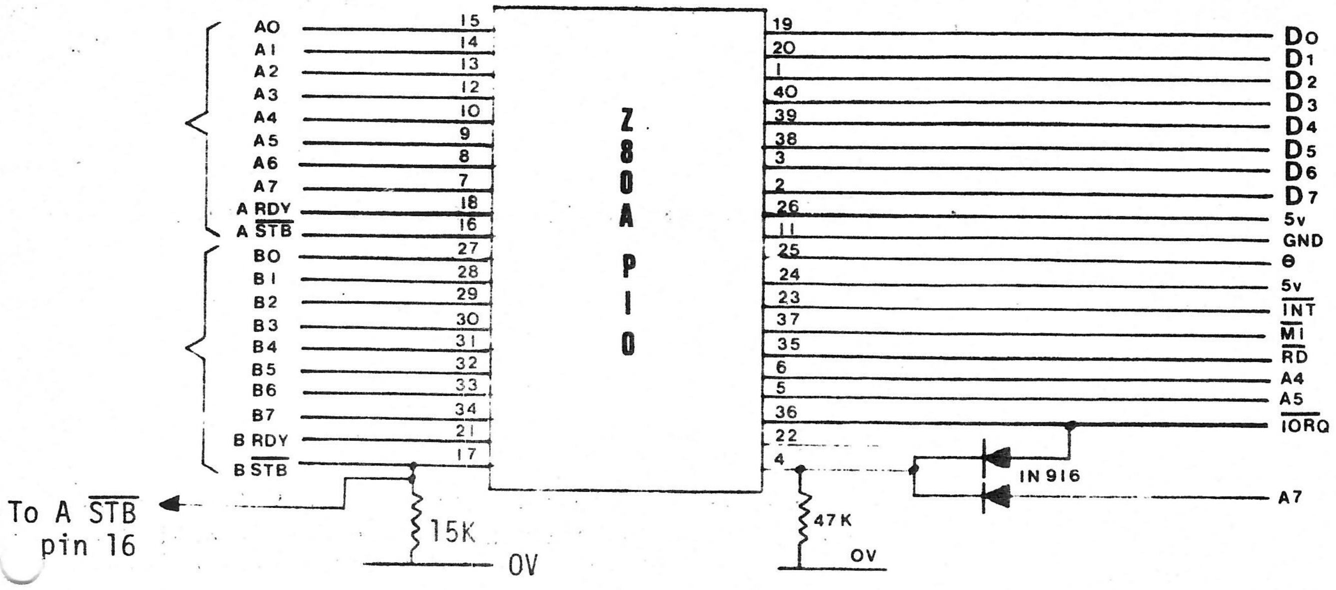
   A) This is almost certainly due to dirty contacts on the edge connectors, clean the connections at both ends of the port, using contact cleaner.

3) One or more lines of the port are dead, due to electrical damage.

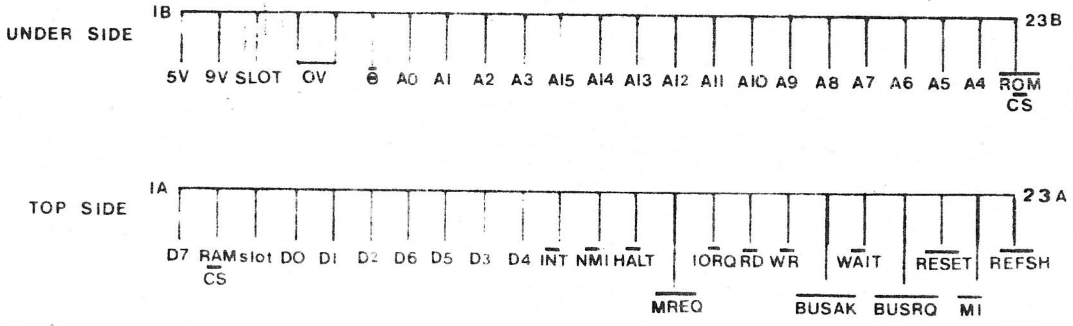
   A) The only remedy is to replace the Z80A PIO chip see list of spares in the catalogue.

If a problem still persists with your port, check for fundamental errors and ensure that the set up procedure is correct.

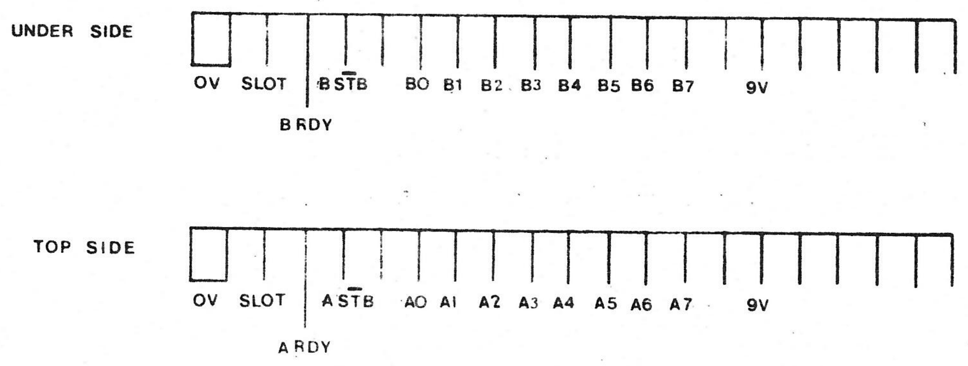
16 Line I/O Port circuit diagram



Connections rear of ZX81/80



Output Connections, 16 line I/O Port.



UNDER SIDE
0V
SLOT
B RDY
B $\overline{\text{STB}}$
B0
B1
B2
B3
B4
B5
B6
B7
9V
TOP SIDE
0V
SLOT
A RDY
A $\overline{\text{STB}}$
A0
A1
A2
A3
A4
A5
A6
A7
9V