Thank you for your interest in:

"INTRODUCTION TO CONTROLS, AUTOMATION & DATA ACQUISITION".

The intention of this project is to introduce you into the world of controls, automation, and data acquisition. I think you, like myself, will be amazed at how simple this actually is.

Included in this disk are the following:

1) Basic source code for 4 projects.
2) Simple binary numbering system.
3) Bit masking.
3) Full explanation of source code.
4) Simple interfacing schematics.
5) List of companies with low cost interfacing boards.

Before you rush out and buy an interface board, study all the information provided again and again. If you don't feel confident after this tutorial, find some other sources of information go more indepth than is provided here.

Any version of basic should work, however some adjustments might have to be made. You can also transpose the source code to the language of your choosing as long as it will address the base address of the interface board.

The list is endless of the things you can do with these boards.
Let your imagination go.
Here are a few:
Light controls
Motor controls
Switches on-off
Relays on-off
Turn devices on-off based on inputs from interface board (home alarms)
Turn devices on-off based on the computers clock
Temperature measurement.

All the demo software is in basic and is based on the "PIO 12" board from Metrabyte. The address is given in the file called "Manufact.lst".
Also, most of the information is based on the INTEL@ 8255 PIA chip. This is the route that I have taken in choosing my interfacing projects. If you choose another board without this chip, don't worry, it will be simpler than the ones based on the 8255, but not as versatile.
Note: It is hard to find a logical order to write all this information. Read all the information, then read it again. Chances are you will find the answers to your questions later in the tutorial. If not, write me. I'll be happy to respond to you ASAP.
Please remember this information is only to get you started. Find alternative sources to get an indepth understanding on each subject. There is, however enough information to get you started.
All the files on the disk are in ASCII format. You can load them into a wordprocessor or print them on your printer. There is also a batch file called [print.bat] that dumps all the files to the printer.

If you have any questions please write to me...

Jeff Beam
Beam Electronics
PO Box 44
Mount Nebo, W.V. 26679

Note: 8255 PIA chip is a trademark of INTEL CORP.

*************** BINARY WORDS *****************

Hopefully you have some knowledge on the binary numbering system. If not, try to find some books on this subject. It is not necessary to know everything on this subject as you will see how all this ties in.

Here is a crash course:

Let's start with the base first. The decimal numbering system has a base of 10, meaning each place, from right to left, is a multiple of 10. The value of the places from right to left - 1, 10, 100, 1000 etc.

An example is the best way to show you this. Look at the number 521. Start from the right or least significant digit and multiply it's weight with the number 1. Take the second digit and multiply it's weight (10) with the number 2. Then take the third digit and multiply it's weight (100) with the number 5. Add all this up...
1 + 20 + 500 = 521

The binary number system has a base of 2. We are only concerned with binary and decimal. The binary weights or values from right to left are: 1,2,4,8,16,32,64,128. Wherever there is a "1" in the binary word, add it's weight to get the total value. For example: Take the binary word (00000101) and count it from RIGHT to LEFT - 1,2,4,8,16 etc. The first digit is 1, so it gets counted and has a weight of 1. The second digit is 0 so it doesn't get counted. The third digit is 1, so it does get counted and it has a weight of 4. Adding these weights in the binary number gives the decimal number 5.
1 + 0 + 4 = 5

Study the chart below.

BINARY	DECIMAL	BINARY	DECIMAL
00000001	1	00001001	9
00000010	2	00001010	10
00000011	3	00001011	11
00000100	4	00001100	12
00000101	5	00001101	13
00000110	6	00001110	14
00000111	7	00001111	15
00001000	8	00010000	16

A BYTE is an 8-bit binary word .When writing a binary word to the interface board, it is sent as a BYTE.

CLEAR AS MUD - RIGHT ? Don't panic, keep studying. Try to find some reference material on binary numbering system if this isn't enough.

Ok, let's look at where we will be sending these binary words. Sending a binary word to a port is simple. Wait !!! What is a port???? As far as you are concerned, the port is the location on the interface board connector where the output controlling signals come from the computer. On most interface boards, an INTEL 8255 PIA chip is used. It has 3 8-bit ports which are accessible on the interface board connector as well. More information on the 8255 will follow.

When referring to 1's and 0's on the output ports, this means 5 volts and zero volts.

In BASIC :   OUT (port base address,byte)
BASIC sends the byte to the port in decimal form not binary.
Real example: OUT 768,2
This will put a 1 on line 2 and 0 on all other lines of the port.
Real example: OUT 768,9
This will put 1's on lines 1 and 4 and 0's on all other lines of the port.
Real example: OUT 768,255
This will put 1's on all the lines of the port.

Base address is the actual address of the interface board. The base address is usually selectable on the interface board. Once the base address is selected, then Port A is (base address + 0). Port B is (base address + 1). Port C is (base address + 2) and Control status word is (base address + 3). This will be explained further in the 8255 section and when you purchase your board.

This concludes our crash course on the binary numbering system and how they are used.

**************************** SWITCHES ****************************

One very important note here is to always watch your current rating on the device you wanting to turn on. Check the current capacity on your interface board !!! Buffering the output with a switching transistor is one solution to the problem. Small current relays is another. Just remember, don't overload the output.

If I wanted to turn a transistor on, I would send the proper binary word to the selected port which would switch the selected bit from 0 to 5 volts, thus turning on the transistor. To turn the transistor off, I would write a zero on the same line that I wrote the 1. (see schematic)

Example:

```
  10 OUT 768,1
  20 FOR T = 1 TO 1000:NEXT T
  30 OUT 768,0
```

Line 10 puts 5 volts on bit #1 on port 768 (remember 768 is the address)
Line 20 is a simple delay
Line 30 puts 0 volts on bit #1 thus turning the relay off.

This following example does the same except on bit #3.

```
  10 OUT 768,4      Remember, basic uses decimal numbers to talk to ports.
                    ie... 00000100 binary is the decimal number 4.
  20 FOR T = 1 TO 1000:NEXT T
  30 OUT 768,0
```

OUT 768,5 would put 5 volts on lines 1 and 3 of the output port.
The decimal number can go as high as 255, which would be binary word 11111111. This would put 5 volts on all 8 lines of the port.

****************** LOOKING AT INPUTS ********************

Inputs can be seen by looking at the port address, then reading the number.

Example:

```
   10 X=INP(768)  This statement assigns what is seen on port 768 to the
                  variable X. This will be a decimal number.
   20 IF X=0 THEN PRINT" All switches are off"
   30 IF X=1 THEN PRINT" Switch number 1 on port 768 is on, all else off"
   40 IF X=2 THEN PRINT" Switch number 2 on port 768 is on, all else off"
   50 IF X=5 THEN PRINT" Switch numbers 1 and 3 on port 768 are on"
   60 IF X=255 THEN PRINT" All switches are on"
```

Get it ???

Note: Always use pull-up when looking at inputs. This simply means put a 1k resistor from 5 volts to all eight bits of the input port. Using switches, you can pull the voltage on each bit to ground thus putting a 0 on that bit of the port. This gives a much more accurate reading than leaving the bit on the port floating.

************************ BIT MASKING ************************

Bit masking is the process that lets you change a bit on a port without disturbing any of the other bits. For example, if you wanted to turn on one line of the port regardless of the status of the other bits, you would use bit masking. This is useful because the computer remembers ,if you use variables as in line 20 below, the state of the output port used for the last write to the port. This is getting deep - I know - but hang in there. Here are some examples. Let's say port A or address 768 is our output port and port B or address 769 is our input port. Bit masking allows us to turn on or off the selected line on the output port without disturbing the others.

```
10 X=1
20 OUT 768,X
30 Y=INP(769)      Different address location
40 IF Y=1 THEN OUT 768,(X OR 2)
```

Now what we have done here is logically "OR" X, which is the last writing of port 768, with 2, which is line 2 of the port. So line 10 put X at 1. Line 20 turned line 1 of port 768 on. Line 30 and 40 are just an example of a decision that the computer can make. Line 30 looked at port 769 and assigned it to variable Y. Line 40 makes the decision then turns line 2 on without disturbing any of the other lines. X could have been any number.

Another example is to use bit masking to look at inputs individually. Example: let's say we wanted to look at bit 3 of port A. First read the port and assign it to X. Then logically (AND) X to another variable Y.

```
10 X=INP(768)
20 Y= X AND 4  ' 4 is the decimal number for binary number 00000100
30 IF Y = 4 THEN PRINT " Bit #3 is on "
40 IF Y = 0 THEN PRINT " Bit #3 is off"
```

Regardless of the value of X, Y will tell you if bit #3 is on or off. You may not have to use this unless your project gets really complicated.

************************** ANALOG INPUTS ******************************

If you purchase an analog board, the manufacture will probably provide a driver or library with the board which will make your life very simple. The driver will have the complicated software to do the conversions and return the results in a variable you can work with.

You can also interface an analog chip to digital lines on a digital I/O board (which is what we have covered so far). I will explain this in the advanced version of this tutorial. Having purchased this, your name will be put on a mailing list and I will be writing you when it is completed. If you just can't wait, find an analog to digital chip (there are many), then look at the timing chart. Write your program according to the timing chart using output lines on one of the ports. It's not easy and it's slow - but it works.

******************** Manufacture List ********************************

Here is a list of companies with some data acquisition boards.

1) Metrabyte
   440 Myles Standish Boulevard
   Tauton Ma.   02780
   (508)-880-3000

2) Omega
   One Omega Drive, Box 4047
   Stamford, Ct. 06907
   1-800-826-6342

3) Data Translation
   100 Locke Drive
   Marlboro, Ma 01752-1192
   (508)-481-3700

PO Box 906
State College, Pa. 16804
(814)-234-8087

5) B & C Microsystems Inc.
335 West Olive Ave.
Sunnyvale, Ca. 94086
(408)-730-5511

6) Sealevel
PO Box 1808
Easley, SC 29641
(803)-855-1581

7) All Aboard   (lowest price)
(508)-261-1123   Fax (508)-261-1094

8) Prairie Digital
28 Sixteenth Street
Prairie du Sac, WI 53578

9) United Electronic Industries, Inc.
10 Dexter Avenue
Watertown, MA 02172
(617)-924-1155   Fax (617)-924-1441   ********************** INTEL 8255 PIA CHIP **********************

This section is to get you familiar with the peripheral interface chip. The INTEL @ 8255 is a complex device, but we will only be using a small portion of it's complexity. I will be talking about one mode of it's operation which is simple. Some further study material can be obtained from INTEL if you want to dig deeper.

The 8255 is a general purpose, programmable I/O device. It has 24 I/O pins which may be individually programmed with software control. This eliminates the need for external logic.

We will be working with mode 0 which is the basic input/output mode. There are three ports on the 8255 labeled A, B, and C. Port A and B can be programmed to be either input or output but not both at the same time. Port C is a different in the fact that it can be split into two 4 bit ports. The same applies here that each 4 bit port can only be input or output but not both. In other words, port C upper can be input OR output but not both. The same with port C lower.

The programming of the chip is done by writing to the CONTROL STATUS REGISTER. This is a separate register with it's own address. Writing to this register is the same as writing to the other registers. In BASIC, OUT 771,(control word). The control word is a decimal number which represents the binary number of your configuration. I will give you some examples on the control word.

Here is a basic chart for determining the right Control Status Word for your application.

```
             CONTROL STATUS WORD
   D7  D6  D5  D4  D3  D2  D1  D0
   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦
   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦  Port C (lower)
   ¦   ¦   ¦   ¦   ¦   ¦   ¦   ¦-- 1=input
   ¦   ¦   ¦   ¦   ¦   ¦   ¦        0=output
   ¦   ¦   ¦   ¦   ¦   ¦   ¦
   ¦   ¦   ¦   ¦   ¦   ¦   ¦       Port B
   ¦   ¦   ¦   ¦   ¦   ¦   ¦------ 1=input
   ¦   ¦   ¦   ¦   ¦   ¦            0=output
   ¦   ¦   ¦   ¦   ¦   ¦
   ¦   ¦   ¦   ¦   ¦   ¦            Mode Selection
   ¦   ¦   ¦   ¦   ¦   ¦----------- 0=Mode 0
   ¦   ¦   ¦   ¦   ¦                 1=Mode 1
   ¦   ¦   ¦   ¦   ¦
   ¦   ¦   ¦   ¦   ¦                Port C (upper)
```

```
  !     !    !    !    !---------------- 1=input
  !     !    !    !                      0=output
  !     !    !    !
  !     !    !    !                     Port A
  !     !    !    !------------------- 1=input
  !     !    !                           0=output
  !     !    !
  !     !    !                          Mode Selection
  !     !    !-----------------------!  00=Mode 0
  !     !----------------------------!  01=Mode 1
  !                                      1X=Mode 2
  !
  !                                     Mode Set Flag
  !--------------------------------- 1 = active
```

EXAMPLE:

Port A input - Port B output - Port C output
Control Status Word is 10010000 or 144 ---- syntax OUT 771,144

Port A input - Port B input - Port C output
Control Status Word is 10010010 or 146 ---- syntax OUT 771,146

Port A output - Port B input - Port C output
Control Status Word is 10000010 or 130 ---- syntax OUT 771,130

Port A input - Port B output - Port C upper output - Port C lower input
Control Status Word is 10010001 or 145 ---- syntax OUT 771,145

The outputs on all three ports are all latched. This means if you turn a bit on it will stay on until you turn it off.

Don't kill yourself trying to understand this chip down to the silicon waffers. Get your interface board and try some of the examples until your get enough of an understanding to start some of your experiments.

```
10 'This is demo #1
20 'It demonstrates the process of outputing signals to controlling circuits.
30 CLS : KEY OFF
40 LOCATE 5, 20: PRINT " *********     DEMO #1      ***********"
50 LOCATE 7, 5: PRINT "This demonstrates the process of sending signals";
60 PRINT " to your circuits."
61 LOCATE 10, 5: PRINT "All bits on port A will come on, then off"
62 LOCATE 12, 5: PRINT "Then all bits on port B "
63 LOCATE 14, 5: PRINT "Then all bits on port C"
65 LOCATE 20, 20: PRINT "Press any key to continue"
70 WHILE INKEY$ = "": WEND
80 CLS
90 PRINT "This program configures the interface board to outputs on all ports"
100 OUT 771, 128 ' This configures all ports as outputs.
110 OUT 768, 255
120 LOCATE 5, 5: PRINT "All outputs on port A are on"
130 FOR t = 1 TO 1000: NEXT t
140 OUT 768, 0
150 LOCATE 6, 5: PRINT "All outputs on port A are off"
160 OUT 769, 255
170 LOCATE 8, 5: PRINT "All outputs on port B are on"
180 FOR t = 1 TO 1000: NEXT t
190 OUT 769, 0
200 LOCATE 9, 5: PRINT "All outputs on port B are off"
210 OUT 770, 255
220 LOCATE 11, 5: PRINT "All outputs on port C are on"
230 FOR t = 1 TO 1000: NEXT t
240 OUT 770, 0
250 LOCATE 12, 5: PRINT "All outputs on port C are off"
255 LOCATE 20, 5: PRINT " Do you want to continue ? "
260 k$ = INKEY$
270 IF k$ = "Y" OR IF k$ = "y" THEN CLS : GOTO 80
280 IF k$ = "N" OR IF k$ = "n" THEN END
290 GOTO 260
```

This is an explanation of the source code in demo1.bas.

Lines 10 through 90 is just screen print statements.
Line 100 is the command that writes to the Control Status Register. The number 128 sets the 8255 's ports as all outputs.
Line 110 puts 1's or 5 volts on all bits on port A.
Line 130 is a simple delay. Increasing the number 1000 will delay longer.
Line 140 puts 0's or 0 volts on all bits on port A.
Line 160 puts 1's on all bits on port B.
Line 180 is the delay.
Line 190 puts 0's on all bits on port B.
Line 210 puts 1's on all bits on port C.
Line 230 is the delay.
Line 240 puts 0's on all bits on port C.
Line 260 assigns the keyboard to k$.
Line 270 and 280 makes a decision on k$.

Remember, the outputs on the 8255 are latched. Meaning, they stay on until you turn them off by writing 0's to the bits you want to turn off.

```
10 'This is demo2
20 'This demonstrates how the interface board looks at inputs
25 CLS : KEY OFF
30 LOCATE 5, 10: PRINT " *****************     DEMO2    *****************"
40 LOCATE 7, 4: PRINT " This demo looks at inputs on port A";
50 PRINT " and displays them on the screen "
55 LOCATE 22, 2: PRINT " Press (CTRL-BREAK) to stop program"
60 OUT 771, 144' This put port A as input and B and C as output
70 X = INP(768)
80 LOCATE 15, 35: PRINT X
90 GOTO 70
```

This is the explanation on the source code demo2.bas.

Lines 10 through 50 are simple screen print commands.
Line 60 writes to the Status Control Register putting ports A as input and ports B and C as outputs.
Line 70 assigns port A to variable X.
Line 80 print the number on the port to the screen.

```
10 ' This is demo3
20 ' This demos time comparison with a known time with the system's clock.
30 CLS : KEY OFF
40 LOCATE 5, 10: PRINT " *****************    DEMO3    ******************"
50 LOCATE 7, 5: PRINT " This demo compares a known time with the system's"
60 LOCATE 8, 5: PRINT " clock and puts a pulse on Port A, bit 1 then ";
70 PRINT "turns it off."
80 LOCATE 22, 5: PRINT " Press (CTRL- BREAK) to stop program"
90 OUT 771, 128
100 OUT 768, 0
110 TIME$ = "00:00:00"
120 TIME1$ = "00:00:05"
130 IF TIME$ = TIME1$ THEN BEEP: OUT 768, 1 ELSE GOTO 130
140 FOR T = 1 TO 1000: NEXT T
150 OUT 768, 0
160 GOTO 40
```

This is the explanation for source code demo3.bas

Lines 10 through 80 are simple screen print commands.
Line 90 writes to the Status Control Register. This puts all ports as outpus.
Line 100 sets all bits on port A to 0's or 0 volts.
Line 110 sets the system clock to 00:00:00

Line 120 sets variable TIME1$ to 00:00:05.
Line 130 compares the system clock to the variable TIME$. If it matches then switch bit 1 on port A to 1 or 5 volts. After that, go to line 140 for a delay, then turn the bit off.
If there is not a match, then go to line 130 and check again.
Line 160 tells the program to go to line 40 which does it all again.

```
10 'This is demo4
20 CLS : KEY OFF
30 LOCATE 5, 15: PRINT "******************   Demo4   ******************"
40 LOCATE 7, 12: PRINT "This demos outputs on Port A based on inputs from";
45 PRINT " Port B"
50 LOCATE 20, 5: PRINT " Press (CTRL-BREAK) to end program"
60 OUT 771, 130'  Control Status
70 OUT 768, 0 '  Sets all bits on Port A off.
80 X = INP(769)
90 IF X < 255 THEN GOTO 200 ELSE GOTO 80
200 OUT 768, 1  'Turns bit 1 on Port A on.
210 LOCATE 15, 20: PRINT " A L A R M "
220 FOR T = 1 TO 1000: NEXT T
230 GOTO 70
```

This is the explanation for demo4.bas.

Lines 10 through 50 are simple screen print commands.
Line 60 writes to the Status Control Register. This puts port A and C as outputs and port B as inputs.
Line 70 sets all bits on port A to 0 or 0 volts.
Line 80 assigns port B to variable X.
Line 90 looks to see if any of the bits on port B go to 0. If not, go to line 80 and look again.
Line 200 puts a 1 or 5 volts on bit 1 of port A.
Line 210 prints "A L A R M" on the screen.
Line 220 is a simple delay to hold bit 1 - port A on.
Line 230 tells the program to go to line 70 which turns bit 1 on port A off.

DECIMAL	BINARY	DECIMAL	BINARY
1	0000 0001	129	1000 0001
2	0000 0010	130	1000 0010
3	0000 0011	131	1000 0011
4	0000 0100	132	1000 0100
5	0000 0101	133	1000 0101
6	0000 0110	134	1000 0110
7	0000 0111	135	1000 0111
8	0000 1000	136	1000 1000
9	0000 1001	137	1000 1001
10	0000 1010	138	1000 1010
11	0000 1011	139	1000 1011
12	0000 1100	140	1000 1100
13	0000 1101	141	1000 1101
14	0000 1110	142	1000 1110
15	0000 1111	143	1000 1111
16	0001 0000	144	1001 0000
17	0001 0001	145	1001 0001
18	0001 0010	146	1001 0010
19	0001 0011	147	1001 0011
20	0001 0100	148	1001 0100
21	0001 0101	149	1001 0101
22	0001 0110	150	1001 0110
23	0001 0111	151	1001 0111
24	0001 1000	152	1001 1000
25	0001 1001	153	1001 1001
26	0001 1010	154	1001 1010
27	0001 1011	155	1001 1011
28	0001 1100	156	1001 1100

29	0001 1101	157	1001 1101
30	0001 1110	158	1001 1110
31	0001 1111	159	1001 1111
32	0010 0000	160	1010 0000
33	0010 0001	161	1010 0001
34	0010 0010	162	1010 0010
35	0010 0011	163	1010 0011
36	0010 0100	164	1010 0100
37	0010 0101	165	1010 0101
38	0010 0110	166	1010 0110
39	0010 0111	167	1010 0111
40	0010 1000	168	1010 1000
41	0010 1001	169	1010 1001
42	0010 1010	170	1010 1010
43	0010 1011	171	1010 1011
44	0010 1100	172	1010 1100
45	0010 1101	173	1010 1101
46	0010 1110	174	1010 1110
47	0010 1111	175	1010 1111
48	0011 0000	176	1011 0000
49	0011 0001	177	1011 0001
50	0011 0010	178	1011 0010
51	0011 0011	179	1011 0011
52	0011 0100	180	1011 0100
53	0011 0101	181	1011 0101
54	0011 0110	182	1011 0110
55	0011 0111	183	1011 0111
56	0011 1000	184	1011 1000
57	0011 1001	185	1011 1001
58	0011 1010	186	1011 1010
59	0011 1011	187	1011 1011
60	0011 1100	188	1011 1100
61	0011 1101	189	1011 1101
62	0011 1110	190	1011 1110
63	0011 1111	191	1011 1111
64	0100 0000	192	1100 0000
65	0100 0001	193	1100 0001
66	0100 0010	194	1100 0010
67	0100 0011	195	1100 0011
68	0100 0100	196	1100 0100
69	0100 0101	197	1100 0101
70	0100 0110	198	1100 0110
71	0100 0111	199	1100 0111
72	0100 1000	200	1100 1000
73	0100 1001	201	1100 1001
74	0100 1010	202	1100 1010
75	0100 1011	203	1100 1011
76	0100 1100	204	1100 1100
77	0100 1101	205	1100 1101
78	0100 1110	206	1100 1110
79	0100 1111	207	1100 1111
80	0101 0000	208	1101 0000
81	0101 0001	209	1101 0001
82	0101 0010	210	1101 0010
83	0101 0011	211	1101 0011
84	0101 0100	212	1101 0100
85	0101 0101	213	1101 0101
86	0101 0110	214	1101 0110
87	0101 0111	215	1101 0111
88	0101 1000	216	1101 1000
89	0101 1001	217	1101 1001
90	0101 1010	218	1101 1010
91	0101 1011	219	1101 1011
92	0101 1100	220	1101 1100
93	0101 1101	221	1101 1101
94	0101 1110	222	1101 1110

95	0101 1111	223	1101 1111
96	0110 0000	224	1110 0000
97	0110 0001	225	1110 0001
98	0110 0010	226	1110 0010
99	0110 0011	227	1110 0011
100	0110 0100	228	1110 0100
101	0110 0101	229	1110 0101
102	0110 0110	230	1110 0110
103	0110 0111	231	1110 0111
104	0110 1000	232	1110 1000
105	0110 1001	233	1110 1001
106	0110 1010	234	1110 1010
107	0110 1011	235	1110 1011
108	0110 1100	236	1110 1100
109	0110 1101	237	1110 1101
110	0110 1110	238	1110 1110
111	0110 1111	239	1110 1111
112	0111 0000	240	1111 0000
113	0111 0001	241	1111 0001
114	0111 0010	242	1111 0010
115	0111 0011	243	1111 0011
116	0111 0100	244	1111 0100
117	0111 0101	245	1111 0101
118	0111 0110	246	1111 0110
119	0111 0111	247	1111 0111
120	0111 1000	248	1111 1000
121	0111 1001	249	1111 1001
122	0111 1010	250	1111 1010
123	0111 1011	251	1111 1011
124	0111 1100	252	1111 1100
125	0111 1101	253	1111 1101
126	0111 1110	254	1111 1110
127	0111 1111	255	1111 1111
128	1000 0000		