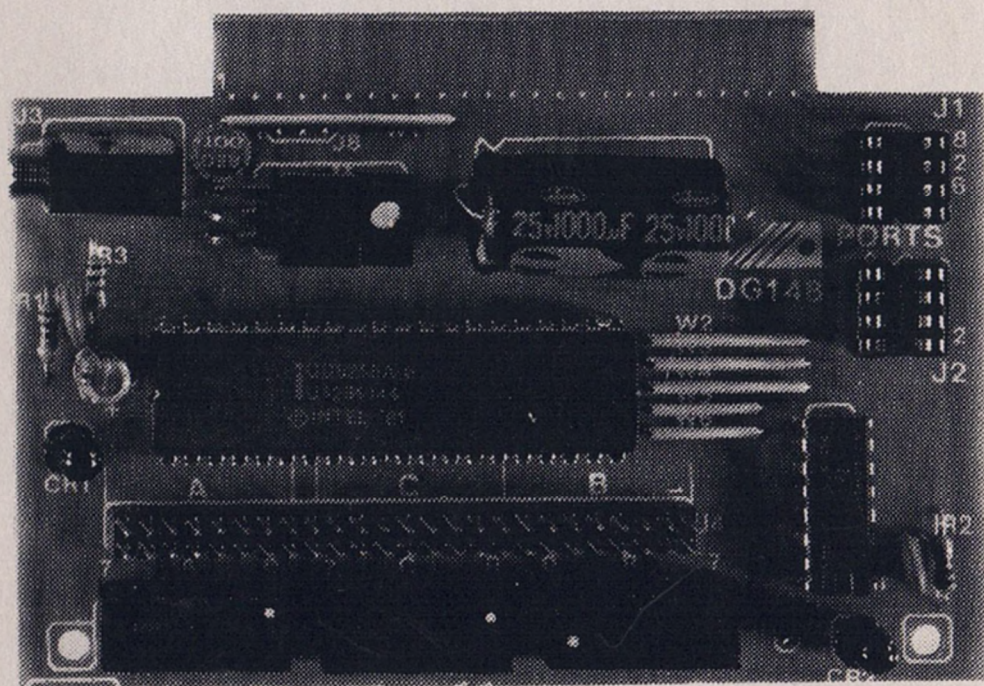


USER'S MANUAL

DG-148



24 LINE DIGITAL I/O CARD

Introduction

The following manual contains many technical details, however in many applications this card is easy to use. You only need to do the following:

- Select a port address (only necessary if you are using another A-BUS card or the Tandy Color Computer)
- Decide what kinds of ports you want: Input, Output, or a mix.
- Send a code to the card to tell it what you have decided.
- Connect your external devices: switches, LEDs, etc.
- Write your program, using the *OUT* command to access the output ports and the *INP* command to access the input ports. (Apple and Tandy Color Computer use *PEEK* and *POKE*).

Features

The DG-148 features twenty four lines which can be configured as TTL inputs or outputs, or a mix. These lines can also be used as latched inputs and outputs, or even bidirectional I/O.

Applications

The DG-148 can be configured many different ways. This makes it useful for a large range of applications. It can be used to connect up to 24 switches, or up to 24 outputs. If both inputs and outputs are used, a 144 key keyboard can be connected.

It is most useful for connecting existing equipment or instruments that have a parallel input or output, for example: digital meters, printers, encoded keyboards, etc. digital sensors.

The bidirectional port can be used to create an elaborate two way communication between the host computer and another computer.

Port Selection

Port selection of any A-BUS card is done in two steps.

- Predecoding (usually a block of 64 ports) is performed by the A-BUS Adapter Card. Refer to Adapter Card instructions for details.
- Each A-BUS board can be set to respond to one (or more, depending on the board) of the 64 ports. A combination of jumpers

is used to select the port. The scheme used is the same for most A-BUS cards, and is described in detail in the A-BUS Overview. It is slightly different on the DG-148, however, because the DG-148 uses 4 ports instead of one. If the following explanation is confusing, it may help to read the section on port addressing in the A-BUS Overview.

There are two jumpers on the board for selecting the port. These two jumpers let you select one set of four ports within the 64 ports selected by the Adapter Card. The numbers printed next to the jumpers add up to the first of the four ports used by the card. For example, if the jumper labelled "J1" is set to position "16", and the jumper labelled "J2" is set to "4" then the ports 20, 21, 22 and 23 will be used by the card. The card is factory set to use ports 0-3. The following pages assume the use of ports 0-3. You may select any four consecutive ports within the block defined by the adapter card. If ports other than 0-3 are used, be sure to substitute the correct port number in the examples. i.e.: if port 58 is used, a statement such as `A=INP(0)` becomes `A=INP(58)`.

Since the ports you use might change as you develop your system, it is a good idea to define the first port number as a variable at the beginning of your program, and use this variable with your *INP* or *OUT* statements. e.g. `A=INP(P)`

Tandy Color Computer users: you will need to change *INPs* to *PEEKs* and *OUTs* to *POKEs*. Also, if your A-BUS Adapter is left in the factory set condition, then you will have to move jumper J1 to 32 and use addresses 65376 to 65379 in your software.

Model III users: you will need to enable the I/O bus before using the card. Use `OUT 236,16` before each *INP* or *OUT*, or use `POKE 16912, PEEK(16912) OR 16`.

Controlling the Card

The DG-148 is based on the Intel 8255A chip. If you are familiar with this chip, you will know most of this information. If you haven't used this chip before, we hope that the following descriptions will help. The chip is quite versatile, and the variety of modes and bit configurations may be overwhelming. The following examples are intended to describe the three modes so you can best judge which mode will fit your project. You will find more examples in magazine

articles and books, as this chip is very popular for digital input/output applications.

Once you have selected the way you plan to use the card, you can look at the tables in the "Configuring" chapter to pick the control code to fit your application.

The Three Modes

The 24 I/O lines of the DG-148 are separated into three groups of 8 lines, Port A, Port B, and Port C. On the A-BUS, they correspond to ports 0, 1 and 2, respectively. The fourth port, (port 3) is used to control the modes of ports 1, 2 and 3.

Three types of I/O (input/output) can be implemented with the card. First let's describe the different modes, then we'll see how to select and use them.

Mode 0 - Simple input or output. Each of the three ports can be set to be simple 8 bit inputs or outputs, in any combination. If simple input is selected, then the data on that port is returned whenever that port is read by the computer. For example, let's say port B is chosen to be a simple input port. Whenever an $X=INP(1)$ is done, the data available at that time on port B of the card will be put into variable X in your computer memory.

If simple output is selected, then the data sent to that port is latched and held on that port. For example, let's say port C is chosen to be a simple output port. $OUT\ 2,255$ will set all the outputs of port C to a high state. $OUT\ 2,0$ will set them all low. The data is always available on the output lines and will remain until new data is sent. Ports A or B can be set to be only all input or all output. Port C is split into two halves which can be controlled independently.

Mode 1 - Strobed input or output. Only ports A and B can be set to use this mode. Some of the bits of port C are used in this mode for the control of the other ports. If strobed input is selected, then three control signals become available at port C. These three signals are called STB*, IBF, and INTR. STB* is an input. When STB* is brought low, the data on the port is read and held by the card. IBF (Input Buffer Full) is an output. IBF goes high to indicate that some data has been read. INTR is an interrupt pin, and will be discussed in the Interrupt chapter. The data strobed into the port is

held until new data is strobed in. When the data held in the buffer is read by the computer, IBF goes low to indicate that the buffer has been read by the computer and is now ready to receive more data. If the "strobed output" mode is selected, then the three signals on port C are called OBF* ACK* and INTR. When OBF* (Output buffer Full) is low, this indicates that there is data in the buffer ready to be read. Bringing ACK* (Acknowledge) low makes the data available on the port for reading, and causes OBF* to go high again. As before, INTR is the interrupt signal.

Mode 2 - Strobed bidirectional I/O. Only port A can be set to Mode 2. In this mode, strobed data input and output are available on the same port. Five bits of port C are used to control this port. They are called STB*, IBF, OBF* ACK* and INTR. These signals act the same as they did in Mode 1.

Mode Zero

Let's use some examples to show the power of this card. First, a simple example. Let's say you have 24 LEDs that you want to blink. You could set all the ports to output ports. Each LED (with a current limiting 1k resistor in series) is connected to the inputs and all the anodes are connected to +5V. Now let's examine the following lines:

```
10 OUT 3,128
```

This sends the control code to the control port. This configures all three ports (A, B and C) to simple latched output ports.

TRS-80 Model III: add the line:

```
5 OUT 236,16
```

Apple and Tandy Color Computer users:

Use *POKE* and *PEEK* (to the proper address) in place of *OUT* and *IN* in the following examples.

```
20 FOR Y=0 TO 2:OUT Y,0:NEXT Y
```

This line (20) turns all the LEDs on.

```
30 FOR Y=0 TO 2:OUT Y,255:NEXT Y
```

This line turns all the LEDs off.

```
40 FOR Y=0 TO 2
```

```
50   FOR X=0 TO 7
```

```
60     OUT Y,255-2upX
```

```
70   NEXT X:OUT Y,255
```

```
80 NEXT Y
```

Note: We use "up" here to mean exponentiation. Each computer has a different symbol for this, including arrows, carats, and brackets. Use the proper symbol in place of "up" here and in any other sample programs.

These five lines turn each of the LEDs on, one at a time. Line 40 loops through the three ports; line 50 loops through the eight LEDs on each port. Line 60 turns the LED on by setting the appropriate bit low, while keeping all the others high.

Let's say you want to control a few LEDs individually. For example, let's turn the fourth LED of Port B on. Counting from 0 (0,1,2,3) the fourth LED is bit 3. Now Port B corresponds to port 1. So

```
OUT 1,255-2up3
```

sets bit 3 of port 1 (the fourth LED on Port B) on. Let's say you want to turn the fifth LED of port A on.

```
OUT 0,255-2up4
```

As another example, turn the first LED on Port A on.

```
OUT 0,255-2up0
```

If you had LEDs rigged up all along, you would now see the fourth LED of Port B and the first LED of Port A on. But what happened to the fifth LED on Port A? You turned it off when you turned the first LED on. If you want to turn both on, use:

```
OUT 0,255-2up0-2up4
```

It's useful in cases like this to have in your software a variable to keep track of the LEDs that are on and off. We'll use L(0) to keep track of Port A, L(1) for Port B, and L(2) for Port C.

To turn on LED number X, where X is from 1 to 8, of Port Y and Y is from 0 to 2, without affecting any other LEDs, use:

```
L(Y)=L(Y) AND (255-2up(X-1));OUT Y,L(Y)
```

(Apple users: Apple BASIC doesn't have a true "AND" operator. See the Adapter manual.)

To turn LED number X, of port Y off use:

```
L(Y)=L(Y) OR 2up(X-1);OUT Y,L(Y)
```

Using these lines, you can manipulate one LED at a time without affecting the others.

Another common use of the DG-148 is to read up to 24 switches of one type or another. For this you would connect a pull up resistor (10K to +5V) to the input and a switch to ground.

Now you will need to use this statement to configure the card.

```
OUT 3,155
```

Now all 24 lines are rigged as inputs. This is the condition that exists when the system is first turned on (Reset).

If all the switches are open, the *INPs* from the ports will all read 255. If all switches are closed, the ports will read 0.

To examine Switch X, where X is a value from 1 to 8, of Port Y, where Y is 0, 1 or 2, you can use:

```
IF (INP(Y) AND 2up(X-1))=0 THEN PRINT"Switch closed"
```

```
IF (INP(Y) AND 2up(X-1))><0 THEN PRINT"Switch open"
```

Now let's assume you have a 32 key keyboard, arranged as 4 rows and 8 columns. To interface it to your computer, you could use 4 outputs and 8 inputs. To read the keyboard, you would keep 3 outputs high and 1 low at a time. Then look at the inputs. All inputs should be high (1) unless the key on that row and the selected column is pressed.

At the same time, you have a display which shows one character at a time. For this you need 8 outputs. The remaining four lines could be used for status LEDs. Altogether, you would like two 8 bit output ports and one 8 bit input port. The inputs will be set to "immediate", not strobed. You want to find the status at the time you read, not some point earlier. Also you want the outputs to be latched (held) until the next output is sent. This is Mode 0 output.

Arbitrarily, let's make Port A the display, Port B the keyboard input, and Port C the LEDs and the keyboard output port. The keyboard output is bits 0-3, since they're physically closer to B, and bits 4-7 are the LEDs. Thus A is Mode 0 Out, B is Mode 0 In, and C is Mode 0 Out. The control code for this configuration is 130. (The computation of this number is explained in detail in the chapter "Configuring". For now, let's just take the code and continue.)

This is a sample program which will look at the keyboard. You could type it in, but you wouldn't see anything happen, since you don't have a keyboard attached. It is presented only as an example. Some of the sections, such as the keyboard scan, are a little of tricky, so if you aren't interested in it, you may skip it.

```
10 OUT 3,130
```

```
20 GOSUB 100
```

```
30 IF K=0 THEN 20
```

```
40 OUT 0,K
```

```
50 C=C+1:IF C=16 THEN C=0
```

```
60 OUT 2,C*16
```

```
70 GOTO 20
```

```
100 K=0
```

```
110 FOR Y=0 to 3
```

```
120 S=(15-2upY)
```

```

130 OUT 2,S+C*16
140 I=255-INP(1)
150 IF I=0 THEN 200
160 FOR X=0 to 7
170 IF I AND 2upX = 0 THEN 190
180 K=X+Y*8+1
190 NEXT X
200 NEXT Y:RETURN

```

Let's examine the program line by line.

Line 10 sends the control code 130 to the control port. This establishes Ports A and C in the Output mode, and B in the Input mode.

Line 20 calls the keyboard scan subroutine. The key typed is returned in K. If K is 0, no key is pressed.

Line 30 checks to see if a key is pressed.

Line 40 sends the value of that key to our "display" on port A.

Line 50 counts up (from 0 to 15, then back to 0, etc.)

Line 60 sends the count out to the LEDs

Line 110 starts to loop through the 4 Output lines.

Line 120 establishes the value to be sent to the output port. Each time through the loop we want a different bit (for a different column) low. S will have bit 0 low and bits 1-3 high when Y is zero, then bit 1 low and bits 0, 2 and 3 high when Y is 1, etc.

Line 130 sends the output for the keyboard. Since this is the same port as the LEDs, we have to send the count out again (otherwise all the LED's would change, because the value on bits 4-7 would be 0.

Line 140 gets the input from Port B. The input is normally all high, except for the bits corresponding to a keystroke. In order to change the sense, we subtract from 255 (the highest possible value) and get 0 when no key is pressed, and 255 when all the keys in this column are pressed.

Line 150 checks to see if any key has been pressed in that column.

Line 160 starts to loop through each row.

Line 170 checks the row. If that row is not 0, then the key has been pressed in that column and row.

Line 180 defines the key to be returned as the Row + The Column times 8 + 1. Thus if the key at the first row (row 0), first column (column 0) is pressed, 1 is returned. If the key at the fourth column (column 3) and the last row (row 7) is pressed, then 32 is returned.

Mode One

To give an example of Mode 1, we'll change the previous example a bit. The "display" we used above was a simple latched output. Many displays, such as LCDs, have drivers attached that use a particular sequences. Let's assume that we have a type of driver that waits for a signal, (call it DR*, for Data Ready). When DR goes low, the display driver fetches a character (bringing a line, call it FETCH*, low). We'll connect this display is connected to Port A. We want a strobed output, so we need Mode 1. Port B will still be our keyboard, so we'll keep it as Mode 0 input. By using Port A in Mode 1, bits 3, 6 and 7 of Port C are used as control lines for port A. The control code for this arrangement is 162. This makes Port A a Mode 1 Output, Port B a Mode 0 Input, and the remaining bits in Port C (0-2,4-5) Mode 0 Outputs.

OUT 3,162

Our keyboard routine remains the same except that we can no longer use bit 3 as one of our outputs. We'll have to use bit 4 or 5 instead.

Simply connect bit 7 of Port C (STB*) to the FETCH* line of the display. Connect bit 6 of port C (ACK*) to DR* of the display. When an OUT to PORT A is done, for example:

OUT 0,65

The value (in this case 65) is stored on the DG148 card, and the signal OBF* goes low. The display sees the OBF* going low, and gets the data. The data is available on the data lines of Port A only when ACK* is low. After ACK* goes low, OBF* is brought high, which indicates that the card is ready for the next character.

The display does not have to grab the data right away. Once the OUT is done, the data is held, and your computer can do other things. The display then takes the data whenever it wants. The only thing to watch out for is to not send new data before the previous data has been read by the display. This is done by checking the status of bit 7 of port C. For example:

10 OUT 3,162

20 OUT 0,65

30 IF (INP(2) AND 128)=0 THEN 30

40 OUT 0,66

Line 10 sends the control code for this mode.

Line 20 sends a character to Port A (to be read by our special display.)

Line 30 checks to see if the data has been read. As long as bit 7 of port C (OBF*) is low, the buffer is full. When it is high, we can send the next character.

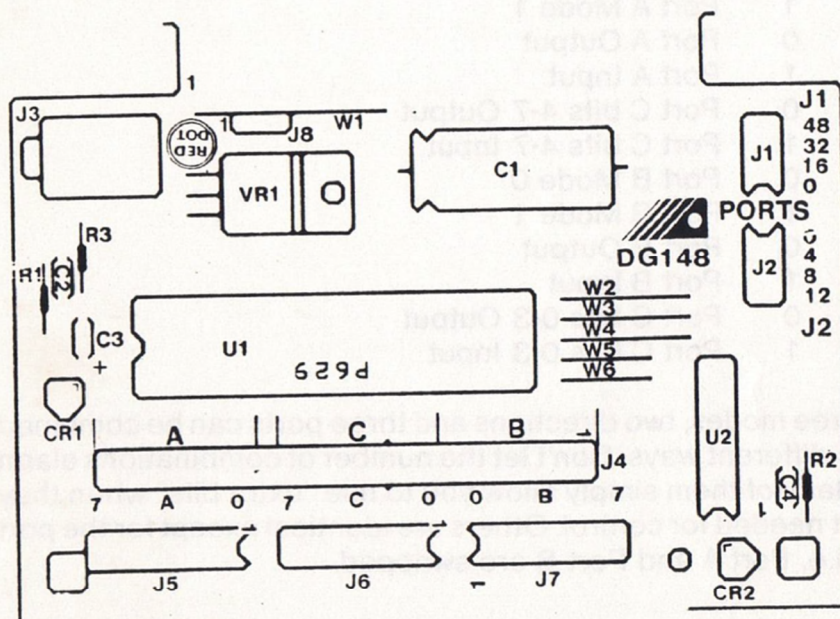
Line 40 sends the next character.

In practice, in a case like this, you would probably do more processing between the OUTs, and line 30 would probably be in a subroutine, where you could call it before each OUT is done.

Mode Two

Mode Two is complicated, and does not lend itself to a simple example. It is used primarily for interfacing two busses. Virtually all of the applications of the DG-148 use modes 0 and 1.

All the information needed to use this mode is included in Configuring. More information is available from the manufacturer's data book (The Intel Component Data Catalog).



Configuring

The three modes, explained previously, are:

Mode 0: Simple input or output

Mode 1: Strobed input or output

Mode 2: Bidirectional strobed input/output

The direction and modes of the three ports (A, B and C) is controlled by sending a code to the control port, which is the fourth port (port 3). This code is determined by combining the bits according to the following table. It is easier to use the table following this one, which lists each of the 54 possible combinations and the code used to obtain them. This first table is shown mainly to show where the numbers come from.

Bit	Status	Meaning
7	1	Bit set Mode
6	0	Port A Mode 0 or 1
	1	Port A Mode 2
5	0	Port A Mode 0
	1	Port A Mode 1
4	0	Port A Output
	1	Port A Input
3	0	Port C bits 4-7 Output
	1	Port C bits 4-7 Input
2	0	Port B Mode 0
	1	Port B Mode 1
1	0	Port B Output
	1	Port B Input
0	0	Port C bits 0-3 Output
	1	Port C bits 0-3 Input

The three modes, two directions and three ports can be combined in a 34 different ways. Don't let the number of combinations alarm you. Many of them simply allow you to use "extra bits" when they are not needed for control. Others are identical except for the port used, i.e. Port A and Port B are swapped.

Code	Port A Mode	Port B Mode	Port C Bits Bits	
128	0 Out	0 Out	0-7 Out	
129	0 Out	0 Out	0-3 In	4-7 Out
130	0 Out	0 In	0-7 Out	
131	0 Out	0 In	0-3 In	4-7 Out
132	0 Out	1 Out	3-7 Out	
134	0 Out	1 In	3-7 Out	
136	0 Out	0 Out	0-3 Out	4-7 In
137	0 Out	0 Out	0-7 In	
138	0 Out	0 In	0-3 Out	4-7 In
139	0 Out	0 In	0-7 In	
141	0 Out	1 Out	3-7 In	
143	0 Out	1 In	3-7 In	
144	0 In	0 Out	0-7 Out	
145	0 In	0 Out	0-3 In	4-7 Out
146	0 In	0 In	0-7 Out	
147	0 In	0 In	0-3 In	4-7 Out
148	0 In	1 Out	3-7 Out	
150	0 In	1 In	3-7 Out	
152	0 In	0 Out	0-3 Out	4-7 In
153	0 In	0 Out	0-7 In	
154	0 In	0 In	0-3 Out	4-7 In
155	0 In	0 In	0-7 In	
157	0 In	1 Out	3-7 In	
159	0 In	1 In	3-7 In	
160	1 Out	0 Out	0-2 Out	4-5 Out
161	1 Out	0 Out	0-2 In	4-5 Out
162	1 Out	0 In	0-2 Out	4-5 Out
163	1 Out	0 In	0-2 In	4-5 Out
164	1 Out	1 Out	4-5 Out	
166	1 Out	1 In	4-5 Out	
168	1 Out	0 Out	0-2 Out	4-5 In
169	1 Out	0 Out	0-2 In	4-5 In
170	1 Out	0 In	0-2 Out	4-5 In
171	1 Out	0 In	0-2 In	4-5 In
172	1 Out	1 Out	4-5 In	
174	1 Out	1 In	4-5 In	
176	1 In	0 Out	0-2 Out	6-7 Out
177	1 In	0 Out	0-2 In	6-7 Out
178	1 In	0 In	0-2 Out	6-7 Out
179	1 In	0 In	0-2 In	6-7 Out

180	1 In	1 Out	6-7 Out
182	1 In	1 In	6-7 Out
184	1 In	0 Out	0-2 Out 6-7 In
185	1 In	0 Out	0-2 In 6-7 In
186	1 In	0 In	0-2 Out 6-7 In
187	1 In	0 In	0-2 In 6-7 In
188	1 In	1 Out	6-7 In
190	1 In	1 In	6-7 In
192	2	0 Out	0-2 Out
193	2	0 Out	0-2 In
194	2	0 In	0-2 Out
195	2	0 In	0-2 In
196	2	1 Out	
198	2	1 In	

You will notice that in this table, all of Port C's bits were not always listed.

When Port A or B is not in mode 0, some of the bits of port C are defined as follows:

The remaining bits, if inputs, can be read as usual. Bits 4-7 can only only be used as outputs with Bit I/O (described later)

Port	Mode	Port C bit
A	Mode 1 Output	bit 7: OBFa* bit 6: ACKa* bit 3: INTRa
A	Mode 1 Input	bit 5: IBFa bit 4: STBa* bit 3: INTRa
A	Mode 2	bit 7: OBFa* bit 6: ACKa* bit 5: IBFa bit 4: STBa* bit 3: INTRa
B	Mode 1 Output	bit 2: ACKb* bit 1: OBFb* bit 0: INTRb
B	Mode 1 Input	bit 2: STBb* bit 1: IBFb* bit 0: INTRb

The Ports

The actual I/O lines are available at two connectors on the card. The signals are referred to by their port and bit number. For example, B0 is the first bit of port B.

The first connector is a fifty pin header. The two rows are identical and from pin 1 the signals are:

A7, A6, A5, A4, A3, A2, A1, A0, GND, C7, C6, C5, C4, C3, C2, C1, C0, B0, B1, B2, B3, B4, B5, B6, and B7.

The socket labelled "J5" and "A" is (from pin 1): A0, A1, A2, A3, A4, A5, A6, A7, +5, GND, GND, C7, C6, C5, C4, C3.

The socket labelled "J6" and "C" is (from pin 1):

C0, C1, C2, C3, C4, C5, C6, C7, GND, GND, GND, GND, GND, GND, GND, GND.

The socket labelled "J7" and "B" is (from pin 1):

C3, C2, C1, C0, GND, GND, GND, +5, B7, B6, B5, B4, B3, B2, B1, B0.

Test LED

For your convenience, an extra LED has been provided on the card so that, with a short wire, you can test the status of the lines. You could also connect this LED to one of the data lines and use it for a status output. This LED is the one farthest from the power jack. There is a hole next to it in which a test lead can be soldered. When the test lead is brought to +5, or an output in the "HI" state, the LED will go on.

Bit I/O

Each bit of Port C can be controlled independently through the control port. This is useful in some control applications, where you don't want to keep track of the current status all the time. It is also the only way to modify the control bits, when Modes 1 or 2 are used.

To reset any bit (make the bit zero) of Port C, simply send the bit number times two to the control port. The bit number is from zero to seven. For example, to reset bit 5, you would use:

OUT 3,10

to reset all the bits, you might use:

*FOR B=0 TO 7:OUT 3,B*2:NEXT B*

To set any bit of Port C, (to 1), all you need to do is send the bit number, times two, plus one. For example, to set bit 4, you could use:

*OUT 3,4*2+1*

or, in simplest form:

OUT 3,9

to reset all the bits you might use:

*FOR B=0 TO 7:OUT 3,B*2+1:NEXT B*

If you wish to know where these formulas come from, read the following. When the control codes were given above, you might notice that Bit 7 is always set ("1"). If Bit 7 of this Code sent to is 0, then the code does not set the mode, it does bit I/O. Therefore bit 7 must be zero for bit I/O, and then bits 1, 2 and 3 contain the bit number. Bit 0 is 0 if you want the bit reset, or 1 if you want the bit set.

Interrupts

Most applications can be accomplished without the use of the interrupt lines. Thus the following information is rarely needed and can be skipped.

When modes one or two are used, the DG-148 controls an interrupt line on its output. These interrupt lines are different from, and independent of, the interrupt line on the A-BUS and interrupt lines in your computer. There is also an A-BUS interrupt line which will be discussed later.

The interrupts generated by the DG-148 can be enabled or disabled from your computer. This enabling is done by setting an appropriate bit high (1). They are disabled when the bit is low (0). The enable bit for port B is port C bit 2. The enable bits for Port A are port C bit 4 for input and Port C bit 6.

For input, if IBF and STB* are high (meaning the data has been completely strobed into the DG-148), and the appropriate enable

bit is high (Port C bit 4 for A, Port C bit 2 for B) then the INTE line will go high, indicating an interrupt. This interrupt will be reset when the data is read by the host computer.

For output, if ACK* and OBF* are high (meaning the data sent by the host computer has been completely accepted by the external device), and the appropriate enable bit is high (Port C bit 6 for A, or Port C bit 2 for B), then the INTE line will go high. This interrupt will be reset when a byte is sent to the DG-148.

These lines are available on the output of the DG-148. For port A, the INTE line is Port C bit 3. For Port B, the INTE line is Port C bit 0.

If you need to use it, the A-BUS interrupt line back to the computer is accessible on the board. Three holes, labelled J8, near the power jack are the ground, the regulated 5V and the interrupt. These can be used to connect a signal to the A-BUS INT line. This line is usually not connected to the computer directly, but can be easily connected on the A-BUS adapter card. See the adapter card instructions for information on connecting this INT line to your computer. Note: This signal is usually active LOW, which is the reverse of the INTR lines of the DG-148. You will need to invert an INTR signal if you wish to connect it to your computer's interrupt line.

Specifications

Catalog Number:	DG-148
Description:	24 Line Digital I/O Card
Size:	4.1 x 3.0 in (104 x 76 mm)
Weight:	2.2 oz (61g)
Power requirements:	8 to 12V DC at 100mA
Number of ports used:	4
Input output threshold:	TTL compatible
Output drive port B and C:	1mA at 1.5V
Device address:	jumper selectable
Operating temperature:	0-55 degrees C
Relative humidity:	5 to 90%

DG148		SIZE FSCM NO.		DWG. NO.		REV	
DRAWN 12/12/78				P629		2	
ISSUED 2/1/83		SCALE		SHEET			

ALPHA Products

