U-BCD
APPLE INTERFACE BOARD
USER MANUAL

'make it easy on yourself'

U-MICROCOMPUTERS

01.047

U-BCD
APPLE INTERFACE BOARD
USER MANUAL

U-MICROCOMPUTERS LTD.,
WINSTANLEY INDUSTRIAL ESTATE,
LONG LANE,
WARRINGTON,
CHESHIRE,
WA2 8PR,
ENGLAND.

TEL: 0925-54117 TELEX: 668920

01.047 NOV. 1981

THE U-BCD

A. INTRODUCTION

The U-BCD represents probably the most cost effective method of collecting and processing physical measurement data with the APPLE microcomputer. Due to its ability to collect up to eight digits of BCD data it is capable of transferring measurements with a very high order of resolution.

The ability to interface with the physical world opens many new doors for the Apple user. Rather than a pure number cruncher the Apple can, for example, become a powerful and intelligent process monitoring tool and together with a relay switch board actually control a process environment, making decisions based on data transferred by the U-BCD.

The board was designed to be compatable with a wide range of BCD outputs as there are often slight differences between makes of instrument. This means that some customising of the software which drives the board may have to be done in order to interface perfectly with your instrument (panel meter). This is not difficult, however, and a familiarity with BASIC and Binary are the only skills required.

This manual should be read in conjunction with the one supplied with your instrument and together they should give you a complete picture of what it is necessary to achieve in terms of the interfacing.

B. THE INSTRUMENT

Most instruments with a BCD output will be compatable with the U-BCD but the output must be of the isolated type. This is usually stated in the specification for the instrument.

Another thing to watch for when interfacing is compatability of naming. For example, the line referred to as Data Ready in this manual is sometimes called Print in those from other manufacturers.

If you have not yet purchased an instrument for your application it would be worthwhile making sure that the type you propose holds valid data on the BCD output as well as the display when its HOLD line is activated. This type is much easier to use and could save considerable time and effort when programming.

C. DESCRIPTION OF OPERATION

In simple terms the U-BCD is merely a collection of chips which allow the Apple to monitor the logic states of 35 separate input lines, and control the state of one output line.

The BCD method of data transfer uses four wires to represent the numbers 0-9. Each wire can be at logic states 1 (5 volts) or logic state 0 (0 volts).
Examples of the relationship between logic levels and the decimal numbers they represent are shown in figure 1.

FIG. 1

DECIMAL	BCD
0	0000
1	0001
2	0010
4	0100
6	0110
9	1001

BCD counting is in fact identical to binary except only 10 of the possible 16 logic states are used. Therefore for each BCD digit used, four lines are required. The U-BCD uses 32 of its 35 input lines to read a possible eight BCD digits. The remaining 3 input lines are used for monitoring extra output lines which may be provided on your instrument such as data ready, over range and polarity.

The single output line is intended for use with the "HOLD" input normally provided on instruments and use of this line greatly reduces programming effort.

Connection to all the above lines is made via the 37 way D-type socket supplied with the board. Pin assignments are shown in figure 2.

Control and monitoring of the various lines is achieved by examining or changing (Peek or Poke) the data stored in those memory locations in the Apple memory map reserved for use by the slot in which the U-BCD is placed. The board can be used in any slot except 0 and only the software driving the board requires modification. Details of which memory locations are used and their function are given in the next section.

FIG. 2 37 WAY CONNECTOR PIN ASSIGNMENTS

PIN	FUNCTION	PIN	FUNCTION
1	1	20	80000
2	2	21	100000
3	4	22	200000
4	8	23	400000
5	10	24	800000
6	20	25	1000000
7	40	26	2000000
8	80	27	4000000
9	100	28	8000000
10	200	29	10000000
11	400	30	20000000
12	800	31	40000000
13	1000	32	80000000
14	2000	33	STATUS INPUT
15	4000	34	STATUS INPUT
16	8000	35	STATUS INPUT
17	10000	36	HOLD (OUTPUT)
18	20000	37	GROUND
19	40000		

D. USER PROGRAM

As stated previously, the U-BCD is driven by user software, either BASIC or machine code, to give maximum flexibility when interfacing with various makes of BCD output.

An example program in BASIC is provided and discussed. This program will in fact cater for the greater majority of BCD outputs and may be used with little modification as a subroutine in the users application program.

However, before discussing actual program examples, it is useful to understand what the software has to achieve in terms of link-up with the hardware on the U-BCD.

On most instruments the BCD output is continually being updated with new information. The frequency with which this update occurs varies widely from every second or so to hundreds of times per second. The programmer wishing to capture this data has two choices, either he reads the data off the BCD output while it remains stable, a period of time dictated by the update rate and accompanied by the presence of the data ready signal, or alternatively, the Hold line if available can be used to freeze the data until the computer has finished reading it in. The latter method is much easier to use as the timing of the software relative to the instrument updates can be ignored.

Using the Hold method would produce the following chain of events:

1) The computer activates hold line on instrument. This freezes the last read valid on the BCD output.
2) The values of status inputs such as polarity are tested.

3) The computer selects the first BCD digit it wishes to read by addressing a particular location in memory.
4) The selected BCD digit is read by addressing the appropriate memory location. The data having been placed at this location when it was selected.
5) The program assigns the value of the digit to a variable.
6) The program loops back to select the next digit to be read, repeating steps 4 and 5.
7) When all required digits have been read the program assembles them in to a suitable format e.g. a single decimal value.
8) The Hold line is deactivated to allow the instrument to continue taking measurements until another reading is required.

The above chain of events is complicated somewhat if no hold line is available as instead of event 1, the computer must poll the data ready signal to check when data is valid. Once confirmed, the program can continue with the other steps. However, all required BCD digits must be read in before the BCD output updates again, otherwise erroneous data could be read. This means that the programmer must check that the time taken to read in the digits is less than the time between updates on the instrument.

The memory locations used by the programmer to select the BCD digits, read them and control the hold line are slot dependant and are detailed in FIG 3. The format of the data word and its relationship with the pins on the 37 D-type connector must be understood before a program using the board can be written. The layout of this data is detailed in FIG 4.

E. PROGRAMMING IN BASIC

We assume for this program a 4 digit instrument with a Hold line which is activated when high. A polarity signal is connected to pin 33 on the D-type connector and is high (logic 1) for positive data. A data ready signal is connected to pin 34 and is high when data is valid.

The example program supplied will now be examined.

LINE 10 TO 30
Should be self explanitory. The variable SN is assigned with the slot number being used by the U-BCD.

LINE 40 Here the variables MM and DD are assigned. MM is the address used to select the least significant BCD digit (The one connected to pins 1,2,3 and 4 on the D type socket). Using this address which will be read (Peek) to obtain the data and status signals. Both are slot dependent and so are calculated relative to the variable SN.

LINE 50 MN is assigned. MN is also an address used to select the least significant hold line (pin 36 on the D type) is held high.

LINE 60 MM is addressed and the value 0 placed at that location. In fact the value sent is of no consequence. Simply by addressing MM we have selected the first digit and brought the hold line low. The actual purpose of this line is just to bring the hold line low

and allows the BCD output to update.

LINE 70 We now address MM + 8 therefore bringing the hold line high. This freezes the BCD output with valid data and allows us to continue to read it. With some instruments a delay will have to be inserted between lines 60 and 70 to allow the BCD output enough time to update.

LINE 80 Before reading BCD we must first establish the state of the polarity signal connected to bit 4 of the data address. To do this we read the data and assign it to a variable which is tested in the next line. The least significant digit is contained in the first four bits of this data, however, this is not relevant as it is one of the higher bits in which our interest lies. These higher bits remain the same whichever BCD digit is selected. The value returned as Q will be in the range 0 to 127 as bit 7 is held low by the board.

LINE 90 This line tests for polarity and dictates what number is used to mask the BCD data when it is read later in the program. The value of Q which was assigned in line 80 will be in the range 0 to 127. The least significant four bits represent a BCD digit, however, bits 4,5 and 6 are the status inputs and could also be high or low. In this example bit 5 is connected to the data ready signal and when high data is valid. Using the hold technique this should always be the case and so no specific testing of this bit is required. We simply subtract 32 from the value of Q to mask this bit. Bit 4 is connected to the polarity signal and this could be either 1 (positive) or 0 (negative). Its state must therefore be assessed before the data can be correctly masked. If bit 4 is 1 then this will add 16 to the value Q. Bit 5 will add 32 anyway therefore if we subtract 48 from Q and if the value returned is equal to 0 or greater, then bit 4 must have been 1 and so the polarity positive. In this case the rest of line 90 is not acted upon and the program continues with line 100. If, however, the value returned from the subtraction is less than 0, bit 4 must have been 0 and so the polarity negative. In this case the correct masking value, 32, is assigned to Q1. The program then jumps past line 100 and continues to line 110.

LINE 100 In the event that the polarity signal was 1, so adding 16 to Q, the correct number for masking bits 4 and 5 of the input data will be 48. This is assigned to Q1.

LINE 110 This is the start of the loop which reads in the 4 digits from the BCD output. When reading 8 digits the loop would simply execute 8 times rather than 4, i.e. 0 to 7.

LINE 120 Here the memory address used to select each digit is calculated, and assigned to variable A. MN, the address used to select the least significant digit, with the hold line high, is added to the displacement dictated by the loop (the value of I). The addresses used for subsequent digits are consecutive.

LINE 130 The required BCD digit is selected by placing a random value into the location addressed by variable A.

LINE 140 The value of the selected BCD digit is assigned as an element in the array N (I). Before being stored as N (I) it is masked by subtracting Q1. This strips the value of the affect of bits 4 and 5, leaving correct value of the BCD digit in the range 0 to 9.

LINE 150 The loop continues until each of the BCD digits has been selected and read.

LINE 160 Here the 4 BCD digits are compiled into one numeric value and the result stored as T.

LINE 170 The program now decides whether or not to make T positive or negative depending on the earlier examination of the polarity signal.

LINE 180 TO 210
These lines are concerned only with printing the value of T on the screen. If T is changed from the last read value it is printed if not the screen remains unchanged. After printing T the program returns to take another reading.

Some instruments may use different logic levels to those assumed in this program. For example, if the polarity signal was 0 when positive and 1 when negative, adjustment would be required. This could be done by simply exchanging the values of 48 and 32 in the program.

Data ready was included although it was not strictly necessary to the operation of this example. It was included merely to demonstrate its possible position in the data word and the way in which it would be necessary to mask it if it was used as part of the hand shaking in the program. In this example it could have been left unconnected and its pin (34) tied to ground. No masking of bit 5 would then be necessary.

FIG. 3 IMPORTANT MEMORY LOCATIONS SUMMARY

Address	Function
49280 + (16*SN)	Selects least significant BCD digit. Pins 1 2 3 4 of 37 way D type. Keeps hold line low.
49288 + (16*SN)	As above but hold line high.
49281 + (16*SN)	Selects second L.S. BCD digit. Pins 5 6 7 8 of 37 way D type. Keeps hold line low.
49289 + (16*SN)	As above but hold line high.

This pattern continued up to address 49287 + (16*SN) with select Pins 29 30 31 and 32 with hold line low.

FIG. 4 LAYOUT OF 8 BIT WORD USED TO READ DATA

Address used to read all selected data = 49152 + (16*SN)

Bit	
0	Least significant bit of selected BCD word.
1	Next significant bit of selected BCD word.
2	Next significant bit of selected BCD word.
3	Most significant bit of selected BCD word.
4	Status input, Pin 33 on D type connector.
5	Status input, Pin 34 on D type connector.
6	Status input, Pin 35 on D type connector.
7	Held low by U-BCD.

If any of the status inputs are unused, they should be held low by connecting the appropriate pin on the 37 way D type connector to Gnd (pin 37 on the D type).

SN=Slot Number - N.B. Slot Zero Cannot Be Used.

FIG. 5 DEMONSTRATION PROGRAM

```
 10  REM BCD BOARD TEST
 20  HOME
 30  PRINT "ENTER SLOT NUMBER...";: INPUT SN
 40  MM = 49280 + (16*SN):DD = 49152 + (256*SN)
 50  MN = MM + 8
 60  POKE MM,0
 70  POKE MM,0
 80  Q = PEEK (DD)
 90  IF Q - 48 < 0 THEN Q1 = 32: GOTO 110
100  Q1 = 48
110  FOR I = 0 TO 3
120  A = (MM + 8) + I
130  POKE A,0
140  N(I) = PEEK (DD) - Q1
150  NEXT
160  T = N(0) + (N(1) * 10) + (N(2) * 100) + (N(3) * 1000)
170  IF Q1 = 32 THEN T = 0 - T
180  IF T = T1 GOTO 200
190  HOME : PRINT T
200  T1 = T
210  GOTO 60
```

01.047 NOV. 1981

U-MICROCOMPUTERS

U-Microcomputers Limited,
Winstanley Industrial Estate,
Long Lane, Warrington, Cheshire WA2 8PR
Telephone 0925-54117/8 Telex 668920