# Interfacing COMPUKIT

## Part 7 D.E.Graham

THIS MONTH we conclude the series with discussion of an Interrupt Driven Clock, and with applications of the Analogue to Digital Converter section of the *P.E.* Analogue Board. These include voltage, temperature, light and sound measurement with v.d.u. bar chart displays, plus a data logger and sampling oscilloscope. But first we continue the discussion of the use of interrupts, begun last month.

### 6522 INTERRUPTS

The interrupt lines of the 6821 on the Decoding Module and the 6522 on the Analogue Board have both been connected to the UK101's $\overline{IRQ}$ line so that interrupts from both these devices may be readily handled. Unfortunately space has not allowed a discussion of the use of 6821 interrupts during this series. The principles involved however are similar to those for 6522 interrupts, and for further information on the 6821 the reader is referred to the data sheets.

6522 interrupts are handled through two sets of registers, the Interrupt Flag Register or IFR (register 13 at 61357) and the Interrupt Enable Register or IER (register 14 at 61358). These are represented in Table 6.3 of part 6 of the series. From this it will be seen that one bit of the IFR and one bit of the IER are associated with each of the control lines CA1, CA2, CB1 and CB2, and one with each timer and the shift register.

The various bits of the IFR are automatically set by the 6522 when events such as the timing out of the timers, or particular transitions on the peripheral control lines occur, and these flags may be inspected periodically by the CPU. If the interrupt is used however, such continual polling is not necessary, and the CPU need only inspect the IFR once an interrupt has been sensed and accepted. The IFR will then tell the CPU which one of the 7 possible functions caused the interrupt.

To condition the 6522 to cause an interrupt request (ie by taking $\overline{IRQ}$ low) when any of its flags are set is achieved by previously setting the corresponding bit in the Interrupt Enable Register. This register uses bit 7 to indicate whether specified bits are to be set or unset. If bit 7 is a one, then writing to any other bit of the register will set that bit. Thus, placing 130 (=128+2) into the IER codes bit 7 with a one, so that the remaining data (2) will cause bit 1 to be set. No other bit positions will be affected. Unsetting bits in the IER is accomplished with bit 7 at zero, so that writing 65 to the IER will unset bits 6 and 0, again leaving the remainder unchanged. To clear the register completely the IER should be loaded with 127. Pressing the Reset button on the Decoding Module will also zero this register.

### INTERRUPT DRIVEN CLOCK

As an example of the use of interrupts with the 6522 we will look at the implementation of an interrupt driven clock on the UK101.

This will employ timer T1 on the 6522 to produce an interrupt every 50ms, and the UK101 will be conditioned to respond to this by updating a series of memory locations in which it holds the current time in hours, minutes, seconds, and twentieths of a second.

Table 7.1 gives the assembler listing of a program which accomplishes this. It resides at 0230 hex in an unused area of RAM below the UK101's BASIC file space. The program falls into two parts: the interrupt service routine itself, from 0230 to 0278 hex; and the initialisation routine from 027D to 02A6 hex. This latter is executed only once at the outset, and we will look at this first.

The first five lines set a jump to 0230 instruction at the UK101's IRQ vector. Line 780 initialises the memory location TWENT that is used to count twentieths of a second. The next sequence is used to configure the 6522. Lines 790 and 800 set timer T1 in the continuous mode by placing 64 into the ACR. Lines 810 and 820 enable the T1 interrupt by loading the IER with 192 (=128+64), thus setting bit 6. The instruction CLI at line 830 clears the interrupt mask, while lines 840 to 870 load the low and high order bytes of the T1 latches with the values 78 and 195, so initiating the count. The count total is thus 195 x 256 + 78 (=49998). The 2 micro-secs delay inherent in the counter make this up to 50000, so giving a 50ms interrupt repetition rate. The final command executes a return from subroutine.

The interrupt service routine starting at 0230 first places the accumulator on the stack, and executes a read from the low byte of T1's count registers. This latter has the effect of automatically clearing the T1 interrupt flag, so taking $\overline{IRQ}$ high again in preparation for the reception of further interrupt requests. The program then decrements the location TWENT (at 022A hex), and checks to see whether it has reached zero. If not it restores the accumulator from the stack, and exits the interrupt routine. If zero has been reached on the other hand, it increments the location SECS (at 022B hex), checking to see whether 60 seconds have been counted, in which case it increments MINS, and so on.

To start the clock once the program has been entered, it is only necessary to execute the subroutine at 027D. This may be accomplished from BASIC using a USR(X) call, or from a machine code routine with a JSR command. We will access it from BASIC.

### RUNNING THE INTERRUPT CLOCK FROM BASIC

The BASIC program listed in Table 7.2 consists of two parts. The first loads the clock program of Table 7.1. The second allows the current time to be inserted in the three locations SECS, MINS and HRS, and uses a USR(X) call to enter the clock initialisation routine, thus starting the clock. Lines 3000 onwards then cause the time to be POKEd to the centre of the screen. Line 2100 determines the screen

position of the display, and this should be altered for Superboard use.

The clock display program may be exited by pressing the space bar at any time, and as long as the UK101 Reset is not pressed, the interrupt clock will continue to operate. Other programs may then be loaded, run and saved as required without disrupting the clock, and these may if desired access the continually updated time by PEEKing locations 555, 556 and 557 for seconds, minutes and hours respectively. There seems to be just one proviso to this: programs which use more than the simplest configuration of GOSUB commands must be avoided, since due to an error in the UK101's firmware the IRQ vector has been located within the stack. As a result, subroutine calls can corrupt the interrupt vector at 01CO, and are in turn corrupted by it. It is for this reason that the program of Table 7.2 avoids subroutine calls, and makes use of flags F and F1 to get around the problems which this creates.

Once the clock has been started, the whole program may be erased with a NEW command, if desired, so as to save memory space, again without affecting the running of the clock.

As with other applications of the 6522, the accuracy of the interrupt clock is tied to that of the UK101's 8 MHz crystal. In my machine this caused the clock to run slow by about 2 secs every hour. This could of course be corrected by using a closer tolerance crystal. But a much simpler way is to alter the value loaded into the low byte of T1's latch register—this is the last data item (78) on line 210 of the program of Table 7.2. With this method it should be possible to achieve accuracies of the order of 1 sec in 16 hours, which should be sufficient for most foreseeable applications.

## MACHINE CODE TO BASIC TRANSLATOR

Lines 100 to 340 of the program on Table 7.2, which enter the interrupt clock program in 6502 code using BASIC, were themselves produced by a translator program written for the purpose. Since this program can translate the contents of any section of memory into BASIC POKE statements, it can be used for saving data or programs stored anywhere in the machine. The program is listed in Table 7.3. It first requests the start and end address of the memory block to be recorded. These may be in hex or decimal, since a hex to decimal conversion routine is included in the program. Next it requests the line number that the BASIC program which it will write is to start from, and the title of the program. It then instructs the cassette recorder to be set to record, and writes the program directly on to it. The machine may then be cleared, and the cassette loaded and run as if it were a normal program.

## ANALOGUE TO DIGITAL CONVERTER

The fourth section of the Analogue Board consists of a multiplexed 8 channel analogue to digital converter. The heart of this unit is a ZN427 monolithic A/D converter i.c. See Fig. 7.1. This uses the so-called successive approximation technique to achieve conversion times of some 20μs or less. This is a far more efficient means of conversion than the other commonly used method which simply causes a D/A converter to sequence its output from zero to 255, using a comparator to stop it when the output matches the analogue signal to be measured. In this case conversion may take as long as 255 clock cycles.

Fig. 7.2 gives the circuit of the A/D section of the Analogue Board. The 8 analogue inputs are taken through SK6, whose full pinout is given in Fig. 7.3. From here they are applied to IC8, a 4051 8 way analogue switch. This selects one of the 8 channels according to the logic states of its pins 6, 9, 10 and 11. These in turn are determined by the

```
 80 0000          ;INTERRUPT CLOCK AT 022B, C & D
 90 0000          START=$0230
100 0230          *=START
110 0230          TWENT=START-6
120 0230          SECS=START-5
130 0230          MINS=START-4
140 0230          HRS=START-3
150 0230          ACR=61355
160 0230          IER=61358
170 0230          TL=61348
180 0230          TH=61349
200 0230 48       BEGIN PHA ;INTERRUPT SERVICE ROUTINE
220 0231 ADA4EF LDA TL
230 0234 CE2A02 DEC TWENT
240 0237 D03B   BNE OUT
250 0239 A914   LDA #20
260 023B 8D2A02 STA TWENT
270 023E A901   LDA #1
280 0240 18     CLC
290 0241 6D2B02 ADC SECS
300 0244 8D2B02 STA SECS
310 0247 C93C   CMP #60
320 0249 D029   BNE OUT
330 024B A900   LDA #0
340 024D 8D2B02 STA SECS
350 0250 A901   LDA #1
360 0252 18     CLC
370 0253 6D2C02 ADC MINS
380 0256 8D2C02 STA MINS
390 0259 C93C   CMP #60
400 025B D017   BNE OUT
410 025D A900   LDA #0
420 025F 8D2C02 STA MINS
430 0262 A901   LDA #1
440 0264 18     CLC
450 0265 6D2D02 ADC HRS
460 0268 8D2D02 STA HRS
470 026B C918   CMP #24
480 026D D005   BNE OUT
490 026F A900   LDA #0
500 0271 8D2D02 STA HRS
510 0274 EA     OUT NOP
520 0275 EA     NOP
530 0276 EA     NOP
540 0277 68     PLA
550 0278 40     RTI
560 0279 00     BRK
570 027A 00     BRK
580 027B 00     BRK
590 027C 00     BRK
700 027D          ;INITIALISE CLOCK
710 027D A94C   LDA #$4C
720 027F 8DC001 STA $01C0
730 0282 A930   LDA #<30
740 0284 8DC101 STA $01C1
750 0287 A902   LDA #>02
760 0289 8DC201 STA $01C2
770 028C A914   LDA #20
780 028E 8D2A02 STA TWENT
790 0291 A940   LDA #64
800 0293 8DAEEF STA ACR
810 0296 A9C0   LDA #192
820 0298 8DAEEF STA IER
830 029B 58     CLI
840 029C A94E   LDA #<4E
850 029E 8DA4EF STA TL
860 02A1 A9C3   LDA #>C3
870 02A3 8DA5EF STA TH
880 02A6 60     RTS
```

**Table 7.1. Assembler Listing of Interrupt Clock**

```
OK
LIST

 90 REM INTERFACING UK101 PROGRAM 19
100 REM 6522 INTERRUPT CLOCK AT 0230 HEX
105 REM LOCATIONS  560  TO  678
110 DATA 72 , 173 , 164 , 239 , 206 , 42 , 2 , 208 , 59 , 169
120 DATA 20 , 141 , 42 , 2 , 169 , 1 , 24 , 109 , 43 , 2
130 DATA 141 , 43 , 2 , 201 , 60 , 208 , 41 , 169 , 0 , 141
140 DATA 43 , 2 , 169 , 1 , 24 , 109 , 44 , 2 , 141 , 44
150 DATA 2 , 201 , 60 , 208 , 23 , 169 , 0 , 141 , 44 , 2
160 DATA 169 , 1 , 24 , 109 , 45 , 2 , 141 , 45 , 2 , 201
170 DATA 24 , 208 , 5 , 169 , 0 , 141 , 45 , 2 , 234 , 234
180 DATA 234 , 104 , 64 , 0 , 0 , 0 , 0 , 169 , 76 , 141
190 DATA 192 , 1 , 169 , 48 , 141 , 193 , 1 , 169 , 2 , 141
200 DATA 194 , 1 , 169 , 20 , 141 , 42 , 2 , 169 , 64 , 141
210 DATA 171 , 239 , 169 , 192 , 141 , 174 , 239 , 88 , 169 , 78
220 DATA 141 , 164 , 239 , 169 , 195 , 141 , 165 , 239 , 96
310 FORC=0 TO  118
320 READ I
330 POKE  560 +C, I
340 NEXT
2000 PRINT:PRINT:PRINT:PRINT:PRINT
2005 PRINT:PRINT,"INTERRUPT CLOCK"
2007 PRINT:PRINT:PRINT"  ENTER PRESENT TIME"
2010 INPUT"    HOURS";H
2020 INPUT"    MINUTES";M
2030 INPUT"    SECS";S
2035 POKE557,H
2040 POKE556,M
2045 POKE555,S
2050 POKE11,125
2060 POKE12,2
2070 X=USR(X)
2100 Z=53660
2110 FORQ2=1TO16:PRINT:NEXT
2200 REM MAIN CLOCK LOOP
2230 IFF1=0THEN3000
2240 F1=0
2250 IFF=2THEN6000
2260 GOTO4000
3000 REM CLOCK DISPLAY
3020 H=PEEK(557)
3030 M=PEEK(556)
3040 S=PEEK(555)
3100 HS=STR$(H)
3120 MS=STR$(M)
3140 SS=STR$(S)
3160 T$="  "+HS+":"+MS+":"+SS+"    "
3200 REM POKE ROUTINE
3210 FORI=1TOLEN(T$)
3220 Q=ASC(MID$(T$,I,1))
3230 POKEZ+I,Q
3240 NEXT
3500 REM SPACE BAR SCAN
3510 POKE530,1
3520 POKE57088,253
3530 IFPEEK(57088)=239THENF1=1
3540 POKE530,0
3545 IFF=2THEN5550
3550 GOTO2200
4000 PRINT:PRINT:PRINT"  DISPLAY ROUTINE EXITED"
4010 PRINT"  CLOCK CONTINUES"
4020 PRINT"  TO RESUME DISPLAY, RUN 2100
4030 POKE530,0
OK
```

**Table 7.2. Interrupt Clock in BASIC**

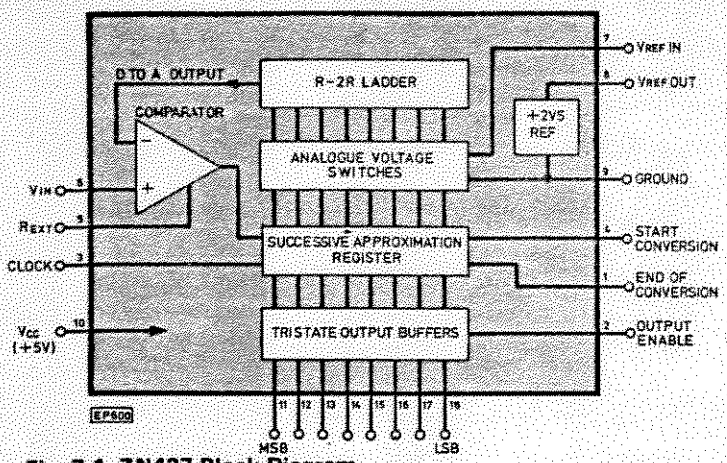



**Fig. 7.1. ZN427 Block Diagram**

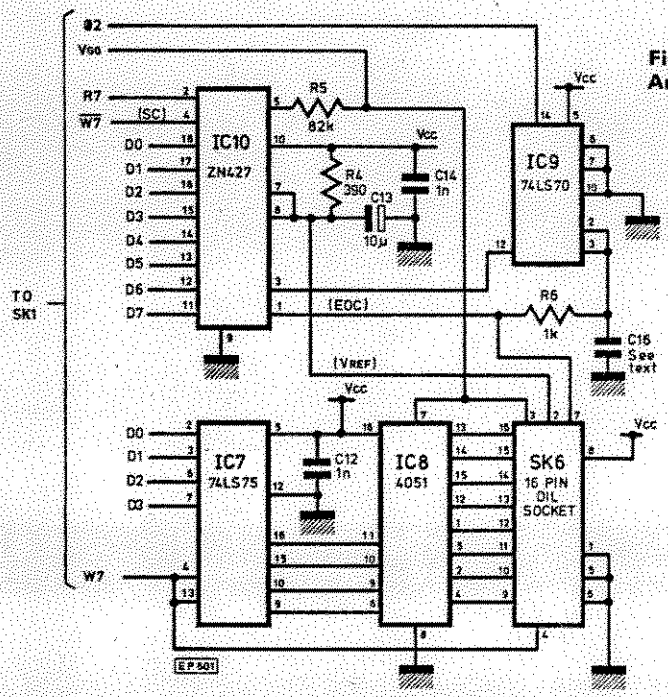



**Fig. 7.2. A/D section of Analogue Board**

quad latch IC7. This is enabled by line W7 from the Decoding Module. The 8 channels are selected by POKEing the channel number (0-7) to address 61319. Thus the command POKE 61319, 6 will connect channel 6 (at pin 10 of SK6) to the input of the converter.

The converter itself requires a low-enable Start Conversion pulse, and this is supplied by $\overline{W7}$ from the Decoding Module. This means that each time the multiplexer at 61319 is addressed, a new conversion sequence is initiated.

The clock input of the converter at pin 3 requires a frequency less than 600 kHz. In order to satisfy this condition, IC9 is used to divide the UK101 Ø2 clock by two, giving a clocking frequency of 500 kHz. Other timing conditions of the ZN427 are satisfied by delaying the operation of the divide by two counter until the ZN427 End of Conversion signal (EOC) goes negative. This is achieved by taking pin 1 of IC10 to pins 2 and 3 of IC9. In the prototype, this was further delayed by the insertion of C16 (50nF); and this capacitor was included in the component overlay diagram published in part 4 of the series. This extra delay time is not necessary for any of the applications described below, and it is recommended that C16 be removed from the board, so greatly enhancing conversion times.

The EOC signal produced by the ZN427 to indicate the completion of conversion has been taken to pin 7 of SK6, and may be monitored by the CPU if required, although conversion is so fast that there is barely time to monitor it in machine code (and certainly not in BASIC) before conversion is completed. Once conversion has taken place, the converter may be read at 61319. Thus the following two commands will print the value of the analogue input on channel 4.

**POKE 61319, 4**
**PRINT PEEK(61319)**

The POKE and PEEK statements may of course be put within a FOR loop to sequence through each of the 8 channels:

**FOR A = 0 TO 7**
**POKE 61319, A**
**PRINT PEEK(61319)**
**NEXT**

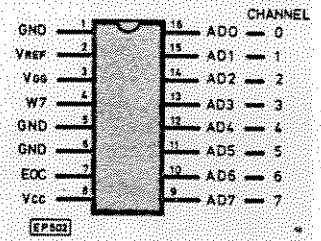



**Fig. 7.3. Connections to SK6 on Analogue Board**

```
50 REM INTERFACING UK101 PROGRAM 20
60 REM MACHINE CODE TO BASIC TRANSLATOR
70 PRINT:PRINT:PRINT:PRINT
72 PRINTTAB(6)"MACHINE CODE TO BASIC TRANSLATOR"
74 PRINT:PRINT
75 PRINT"     THIS PROGRAM PRODUCES A TAPED PROGRAM"
76 PRINT"     IN BASIC FOR REPRODUCING ANY BLOCK OF"
77 PRINT"     MEMORY LOCATIONS"
90 PRINT:PRINT:PRINT
92 PRINT"     ARE START AND END ADDRESSES OF MEMORY"
93 PRINT"     BLOCK IN HEX"
95 INPUTY$
96 IFLEFT$(Y$,1)="Y"THEN600
98 PRINT
100 INPUT"    START ADDRESS IN MEMORY";AS
105 PRINT
110 INPUT"    END ADDRESS IN MEMORY";AE
120 INPUT"    FIRST LINE NO OF BASIC PROG";NB
130 PRINT"    TITLE OF BASIC PROGRAM"
135 INPUTA$
140 SAVE
150 PRINT"    START RECORDER"
160 INPUT"    ENTER S TO START";S$
200 PRINTNB; "REM ";A$
205 PRINTNB+5; "REM LOCATIONS ";AS;" TO ";AE
210 FORA=0TO(AE-AS)STEP10
220 PRINTNB+10+A;" DATA";
230 FORB=0TO9
240 PRINTPEEK(AS+A+B);
250 IFAS+A+B=AETHENB=10
254 IFB<9THENPRINT",";
260 NEXT
270 PRINT
280 NEXT
400 Z=NB+100+10*INT((AE-AS)/10)
410 PRINTZ;" FORC=0 TO ";AE-AS
420 PRINTZ+10;" READ I"
430 PRINTZ+20;" POKE ";AS;"+C, I"
440 PRINTZ+30;" NEXT"
450 FOREX=1TO5000:NEXT
500 POKE517,0
510 FORA=1TO16:PRINT:NEXT
520 PRINT,"RECORDING COMPLETE"
550 END
600 PRINT"    START ADDRESS (HEX) IN MEMORY"
620 GOSUB900
630 AS=N
635 PRINT
640 PRINT"    END ADDRESS (HEX) IN MEMORY"
650 GOSUB900
660 AE=N
670 GOTO120
900 REM HEX CONVERSION
910 INPUTQ$
920 IFLEN(Q$)<>4THENPRINT:PRINT"  4 DIGIT FORMAT ONLY":GOTO900
930 N=0
940 X$="0123456789ABCDEF"
950 FORJ=1TO4
960 FORI=1TO16
970 IFMID$(Q$,J,1)=MID$(X$,I,1)THEN1010
980 NEXTI
990 PRINT:PRINT"  CHARACTER NOT IDENTIFIED - REDO"
1000 GOTO900
1010 N=N+(I-1)*16^(4-J)
1020 NEXTJ
1030 PRINT"    DECIMAL EQUIVALENT = ";N
1050 RETURN
OK
```

**Table 7.3. Machine Code to Basic Translator Program**

The conversion scale of the ZN427 is determined by the voltage applied to its pin 7. On the Analogue Board this is connected to the ZN427's precision 2·55 volt reference source, which results in a very convenient conversion scale of 10 mV per digit, so that an input of 2·17 volts for example will produce a digital value of 217, and so on.

## EIGHT CHANNEL DISPLAY PROGRAM

The program in Table 7·4 may be used to monitor the input levels on all 8 channels of the converter. Analogue to digital conversion is carried out in lines 220 and 230, and the updated conversion result POKEd to the screen, after suitable processing. As may be seen from the photograph, a horizontal bar graph representation is also produced for each channel. This latter is only altered in subsequent passes if a change in input voltage is detected. The program may be exited by depressing the space bar.

## VOLTAGE MEASURING

As it stands, the converter will measure voltages from 0 to 2·55 volts presented to the input pins of its 8 channels (ie pins 16 to 9 of SK6, corresponding to channels 0 to 7 respectively). For accurate conversion, each input should be loaded with an impedance of 10k or less. This is necessary to ensure an impedance match with the input circuitry of the ZN427, and also in order to avoid charge carry-over from one channel to the next as the multiplexer sequences through the channels. If for example a voltage is applied to channel 0, and the input of channel 1 is left floating, channel 1 will read a non-zero voltage as measured in the display program of Table 7.4. This is avoided by keeping input loads below 10k and by earthing unused inputs.

If it is required to increase the voltage range of any channel of the converter, this may be accomplished by using a resistor network. That shown in Fig. 7.4 gives a scale factor of two so that 5 volts will give a reading of 250. The resistors used should be 1% types or better, although if these are not available, lower tolerance resistors could be trimmed (by constructing series/parallel combinations) so as to give a reading of 127 or 128 when point X is connected to pin 2 of SK6. This is the ZN427's own 2·55 volt reference source. The function of C in the dropper network is to keep the input impedance low enough to match the ZN427. Similar techniques may be used to extend the range further.

## ANALOGUE JOYSTICK

Each of the 8 channels of the converter may be used with a virtually infinite variety of sensors. We will give circuits for use with some of the more obvious. Fig. 7.5 gives a circuit for a two-direction joystick. Each joystick potentiometer is supplied by the 2·55 volt reference source, and 100nF capacitors are again used to keep the input impedance low. As the circuit stands the two joysticks are connected to channels 0 and 1, and very straightforward software could be used to move a cursor or other character to any point of the screen in response to the position of the joystick. With the analogue use of the joystick, as opposed to the digital applications discussed in parts 3 and 5 of the series, the cursor could immediately move to the selected spot without having to traverse all intermediate screen positions first.

## LIGHT SENSOR

Fig. 7.6 gives the circuit for an l.d.r. light sensor. With R1 at 1k this produces a good range of output, from very dim conditions (a v.d.u. screen at 1 metre) to brightly lit interiors. Higher precision in dark conditions may be achieved by increasing the value of R1, while improving the range in very bright conditions will be limited by the current drawn.

**Table 7.4. Eight Channel A/D Converter Display**

```
100 REM INTERFACING UK101 PROGRAM 21
110 REM 8 CHANNEL A/D CONVERTER DISPLAY
120 FORA=1TO8:PRINT:NEXT
130 PRINTTAB(10)"8 CHANNEL A/D CONVERTER"
140 PRINT:PRINTTAB(13)"(SPACE BAR EXITS)"
150 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
160 FORA=1TO3000:NEXT
170 FORA=1TO16:PRINT:NEXT
180 GOSUB 470
190 Z=53335
200 S=61319
210 FORA=0TO7
220 POKEB,A
230 M=PEEK(8)
240 S=Z+A*128
250 GOSUB 300
260 GOSUB 560
270 NEXT
280 GOSUB 410
290 GOTO210
300 REM NUMERICAL DISPLAY ENTRY POINT
310 REM SCREEN POKE ROUTINE
320 N$=STR$(N)
330 L=LEN(N$):REM STRING DISPLAY ENTRY POINT
340 FORM=1TOL
350 O=ASC(MID$(N$,M,1))
360 POKES+M,O
370 NEXT
380 IFPEEK(S+M)=32THEN 400
390 POKES+M,32:M=M+1:GOTO 380
400 RETURN
410 REM SPACE BAR MONITOR
420 POKE530,1
430 POKE57088,253
440 IFPEEK(57088)=239THENPOKE530,0:END
450 POKE530,0
460 RETURN
470 REM TEXT HANDLING
480 B$="CHANNEL"
490 Z=53323
500 FORA=0TO7
510 N$=B$+STR$(A)
520 S=Z+A*128
530 GOSUB 330
540 NEXT
550 RETURN
560 REM BAR DISPLAY CALCULATION
570 N1=INT(N/10+.5)
580 N2=N1-N(A)
590 IFN2=0THENN(A)=N1:RETURN
600 IFN2<0THEN 650
610 FORQ1=N(A)TON1-1
620 POKES+64+Q1,161
630 NEXT
640 GOTO 680
650 FORQ1=N(A)TON1STEP-1
660 POKES+64+Q1,32
670 NEXT
680 N(A)=N1
690 RETURN
OK
```

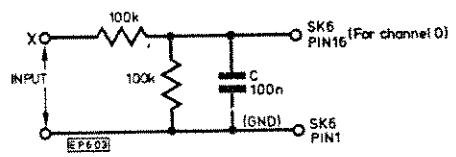



**Fig. 7.4. Divide by two network**

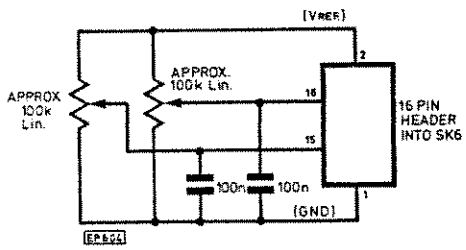



**Fig. 7.5. Connection of 2 axis joystick to A/D converter**

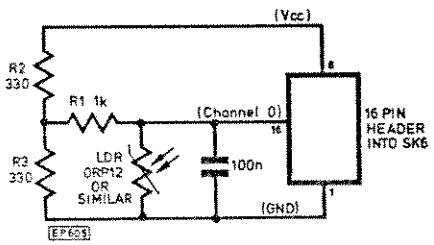



**Fig. 7.6. Connection of light sensor to A/D converter**

The left hand end of R1 could have been connected to the 2·55 volt reference source at pin 2 of SK6, but this is easily overloaded (maximum loading about 1k), and it is more prudent to use the pair of dropper resistors shown; or alternatively a 2·5 volt zener could be used. With the circuit of Fig. 7.6, quite small changes in light level may be discerned; and by using an l.d.r. on each channel, pattern recognition experiments may be undertaken. For such an application nine (or better still, sixteen) would be a more appropriate number of light sensors to use, in which case a second multiplexer could be added, either driven by one port of the PIA or VIA, or using a second 74LS75 latch, for which purpose the decoded line W7 has been taken out to pin 4 of SK6 on the Analogue Board. In such an application it should be noted that POKEing a one to bit 3 at 61319 renders the analogue switch IC8 open circuit.

## TEMPERATURE SENSOR

Thermistors may be used for temperature measurement using a simple dropper resistor in the same manner as described for the l.d.r. application, although only fairly high resistance types are suitable, in order to avoid self-heating effects.

Far more precise measurement may be achieved for little extra outlay by using a temperature sensing diode such as the LM335Z. This is essentially a diode whose forward voltage is dependent on its temperature in a very linear manner, at the rate of 10mV per degree Kelvin. This means that its voltage increases by 10mV for every degree Centigrade above absolute zero (−273°C). Thus at 20°C its forward voltage will be 2·93. This voltage is unfortunately beyond the direct range of the A/D converter. But it is easily reduced with a resistor network such as that shown in Fig. 7.7. This will produce a reading of about 146 at 20°C.

To improve the reading precision, a d.c. amplifier with offset facility can be used between the sensor and the converter. Fig. 7.8 gives a circuit for such an arrangement. With this the voltage gain of the amplifier can be altered from unity to about 50 by adjusting VR1, and an offset of ±3 volts at maximum gain can be achieved by altering VR2. It should be noted however, that even at quite modest gain settings, the earth loop caused by using the UK101's mains transformer to drive the Decoding Module produces hum problems, and a small fluctuation appears on the reading. To avoid this, a separate 9-0-9 volt transformer should be used to power the Decoding Module.

To obtain the highest precision with the LM335Z (1°C error at room temperature) it should be used in conjunction with a 10k potentiometer as in Fig. 7.9, and the potentiometer adjusted to give an output as close to 2·982 volts as possible at 25°C.

The d.c. amplifier of Fig. 7.8 may also be used for increasing the sensitivity of the converter for use in other applications, such as d.c. voltage measurement etc.

## AUDIO DETECTION

The circuit of Fig. 7.10 allows sound levels to be measured with the A/D converter. Like the temperature sensor amplifier of Fig. 7.8, it uses a 741 op. amp., but no earth loop hum problems arise in this case because no offset is required, and pin 2 can be taken down to earth. The values of R1 and C1 in the integrator following the diode detector circuit should be selected to give a time constant appropriate to the use to which the unit is to be put. Generally speaking this should be somewhat greater than the sample repetition time of the program driving the converter. In the case of the 8 channel display program of Table 7.4, a time constant of about one tenth of a second is appropriate, and may be

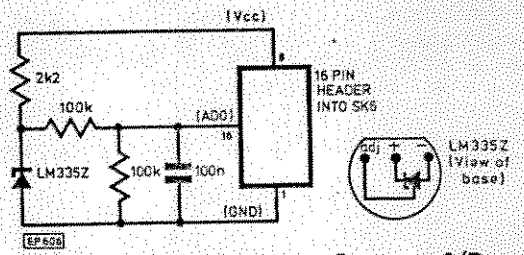



**Fig. 7.7. Connection of Temperature Sensor to A/D converter**

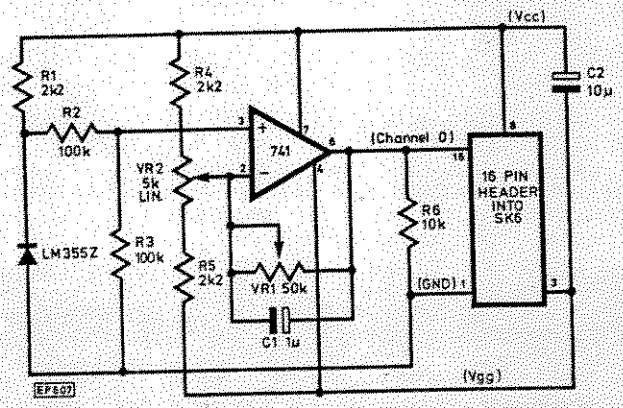



**Fig. 7.8. Temperature Sensor with voltage amplifier**

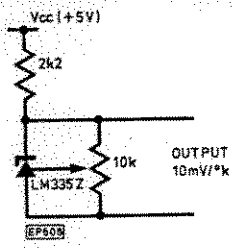



**Fig. 7.9. Calibration of LM335Z temperature sensor**

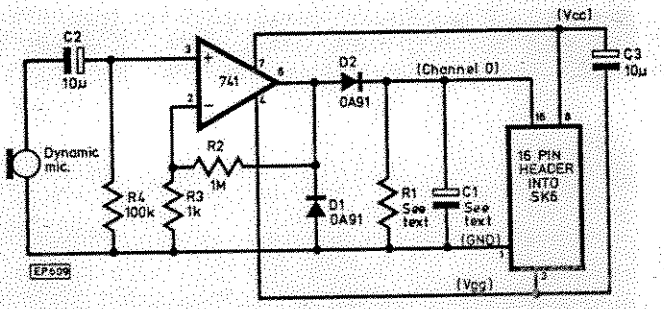



**Fig. 7.10. Audio Detection using the A/D converter**

achieved using 10k for R1 and 10µ for C1. For much higher sample repetition rates, such as those achievable with the sample 'scope program below, C1 may be removed completely for certain applications.

## DATA LOGGER

The interrupt clock described earlier may be used in conjunction with the A/D converter to create a simple data logger. A program that accomplishes this is given in Table 7.5. To use it, load the interrupt clock program in BASIC of Table 7.2, then load the data logger supplement of Table 7.5. The latter makes use of routines in the former, including the interrupt clock itself, to implement the data logging function. When RUN is executed, the interrupt clock will operate as before, but now pressing the space bar will cause a transfer to the data logger. This requests a start time for logging in hours, minutes and seconds; then a repetition

```
4090 REM INTERFACING UK101 PROGRAM 22
4100 REM DATA LOGGER SUPPLEMENT
4110 REM TO BE LOADED ON TOP OF
4120 REM PROGRAM 19 (THE INTERRUPT
4130 REM CLOCK)
5000 REM DATA LOGGER
5020 F=2:X=53276:NA=0
5100 FORQ2=1TO16:PRINT:NEXT
5110 PRINT,"DATA LOGGER"
5130 PRINT:PRINT:PRINT"   START TIME OF RECORDING"
5140 INPUT"   HOURS";SH
5142 INPUT"   MINUTES";SM
5144 INPUT"   SECONDS";SS
5150 PRINT:PRINT"   REPETITION PERIOD OF READINGS"
5160 INPUT"   HOURS";RH
5170 INPUT"   MINUTES";RM
5180 INPUT"   SECONDS";RS
5190 PRINT:PRINT"   NUMBER OF READINGS"
5200 INPUTNR
5210 DIMA(NR,8):DIMA$(NR)
5220 FORQ4=1TO16:PRINT:NEXT
5230 PRINT" NO";TAB(11)"TIME";TAB(33)"DATA"
5500 GOTO2200
5550 REM ENTRY POINT
5560 IFS=SSANDM=SMANDM=SMTHEN5600
5570 GOTO2200
5600 REM DATA AQUISITION
5605 PRINT" ";NA+1;"..";
5610 PRINTT$;
5620 FORQ2=0TO7
5625 IFQ2=4THENQ3=-24:PRINT
5630 POKE61319,Q2
5640 A(NA,Q2)=PEEK(61319)
5650 PRINTTAB(Z3+6*Q2+Q3);
5660 PRINTA(NA,Q2);
5670 NEXT
5680 Q3=0
5690 PRINT:PRINT
5695 NA=NA+1:IFNA=NRTHEN6100
5700 CS=0:CM=0
5710 SS=SS+RS
5720 IFSS>59THENSS=SS-60:CS=1
5730 SM=SM+RM+CS
5740 IFSM>59THENSM=SM-60:CM=1
5750 SM=SM+RH+CM
5760 IFSM>=24THENSM=SM-24
5800 GOTO2200
6000 PRINT,"EXIT FROM DATA LOGGER"
6010 GOTO6110
6100 PRINT,"DATA LOGGING COMPLETE"
6110 PRINT,NA;"READINGS TAKEN"
6120 PRINTTAB(15)"TIME";T$
6125 PRINT:PRINT
6130 PRINT"TO DISPLAY CLOCK, RUN 2100"
6140 PRINT"TO RESTART LOGGER, RUN 5000"
```

**Table 7.5. Data Logger Supplement**

```
100 0000          ;SAMPLE SCOPE
110 0000          START=$0230
120 0230          *=START
130 0230          ;INPUT ROUTINE
140 0230 AE2E02   LDX START-2
150 0233 AD2F02   BB LDA START-1
160 0236 8D87EF   STA 61319
170 0239 AC2D02   LDY START-3
180 023C 88       AA DEY
190 023D EA       NOP
200 023E EA       NOP
210 023F EA       NOP
220 0240 D0FA     BNE AA
230 0242 AD87EF   LDA 61319
240 0245 9D9002   STA $0290,X
250 0248 CA       DEX
260 0249 D0E8     BNE BB
270 024B 60       RTS
280 024C 00       BRK
290 024D 00       BRK
300 024E 00       BRK
310 024F 00       BRK
400 0250          ;OUTPUT ROUTINE
410 0250 AE2E02   CC LDX START-2
420 0253 BD9002   DD LDA $0290,X
430 0256 8D80EF   STA 61320
440 0259 AC2C02   LDY START-4
450 025C 88       EE DEY
460 025D EA       NOP
470 025E EA       NOP
480 025F EA       NOP
490 0260 D0FA     BNE EE
500 0262 CA       DEX
510 0263 D0EE     BNE DD
520 0265 4C5002   JMP CC
```

**Table 7.6. Assembler Listing of Sample 'Scope**

period in hours, minutes and seconds; and lastly the number of loggings required. This latter number sets the dimension of two arrays A$(I), which records the time of each reading, and A(I,J), which contains the 8 data values recorded at each reading.

The program then initiates the continued POKEing of the time to the centre of the top line of the screen, at the same time continually checking to see whether the start time has been reached. When it has, the A/D converter sequences through its 8 channels, and the resultant data is stored in the array for later use. The number of the reading, the time and the 8 data values are also printed on to the screen. See accompanying photograph. The program then waits for the duration of the repetition period, and again stores and prints the data measured. If hard copy or cassette recording of the data is required, an appropriate routine may be added at 6200 to be executed after data logging has been completed. And as with the interrupt clock itself, the space bar may be used to exit the data logger.

The logger program is highly flexible, and can take readings at intervals as short as 1 second, or as long as 24 hours, and the number of readings allowed is limited only by the memory size of the machine. A 4k machine will allow some 15 or 16 readings, while with 8k this rises to somewhere above 110. If memory size is a problem, a considerable saving could be made by storing the measured data individually in RAM by means of POKE statements to a protected area. Alternatively the program could be modified so as to print out, but not to store, the data, in which case there is no limit to the number of readings taken.

The uses to which the data logger may be put are legion. It could for example be used to monitor the temperature, noise, light and humidity levels in a given locality; and checks could be made on the effectiveness of thermal insulation by recording inside and outside temperature over a 24 hour period. It could also be used to gather data on the temperature dependence of a resistance, or to record the voltage decay curve of a CR combination of sufficiently long time constant. Additionally, the logger could be used in conjunction with the event recorder using T2 of the VIA described last month, to log the number of events of a given kind occurring within any period. Those events could variously be counts from a Geiger-Muller tube, passages across a photocell, the occurrence of sound levels above a given threshold etc. Once the data has been collected from whatever source, the UK101 can then be used to analyse it in any appropriate manner, and to present the results graphically if so desired.

## WAVEFORM SAMPLING

The very fast conversion rate attainable with the ZN427 A/D converter—around 50,000 per second—allows the sampling and digital storage of waveforms at frequencies up to a few kHz. Using the converter a series of samples may be taken in very rapid succession, and stored in RAM, from where they may be analysed at leisure. For example, Fourier analysis might be performed on the data, or the sampled waveform could be replayed continuously through the D/A converter for steady display on an oscilloscope screen. Alternatively the UK101 VDU screen could be used to display the sampled waveform.

For such applications it is obviously essential to use machine code, since using BASIC would only allow some 100 or so samples to be taken every second. Table 7.6 gives an assembler listing of a two-part program which has a sampling and replay facility. The first part, which is located at 0230 hex, causes a series of samples to be taken and stored in RAM above 0290 hex. The sampling rate, the number of samples, and the channel number from which the samples are taken are determined by the contents of locations 022D, 022E and 022F respectively, and these may be set up before the program is run. The second part of the program, which begins at 0250 hex, allows the replay of the stored data in a continuous loop to the D/A converter located at 61320 dec. The number of samples to be replayed, and the replay rate, are determined by the contents of locations 022E and 022C respectively, and again these parameters may be set up before the routine is run.

A program in BASIC is given in Table 7.7 which loads the above program, and then controls the sampling and replay parameters with a series of INPUT statements. The program sets the channel number to 0, and the number of samples to 100 in lines 280 and 290. These may be altered if required, though 100 is the maximum number of samples allowed. It then requests a sample rate from 1 (fast) to 255 (slow), and after making the sample, requests a replay rate. It then replays through the D/A converter in a continuous mode, which can only be exited through a reset and warm-start.

The accompanying photographs show a 'scope trace of a 100Hz waveform that was sampled and replayed using this program. The original waveform is also shown. As may be seen from this, fidelity of reproduction is very good, even though no sample-and-hold circuitry has been used in the A/D conversion.

Sampling with this arrangement is fast enough for the lower end of the audio spectrum, and the circuit of Fig. 7.10 may be used in conjunction with the sampler to allow sampling of parts of speech signals and other audio waveforms.

As suggested above, the 100 data samples produced by the sampler may be analysed at leisure, or modified before replaying them through the D/A converter. If it is required to work on more than one set of samples, for comparison purposes for example, the 100 bytes of data may be transferred to another location using PEEK and POKE commands. The 100 bytes from 0290 hex would then be used simply as a temporary input buffer for the incoming data.

## VDU SAMPLE SCOPE

If no suitable oscilloscope is available, or a larger display is required, the add-on program of Table 7.8 will allow for any 40 byte wide part of the sampled waveform to be displayed on the UK101 screen. When this program is loaded on top of the sample scope program of Table 7.7, it requests an offset factor to be entered (0-60) after the sampling routine has been executed, and POKEs up a display on the screen, using the characters 128-135 to achieve a vertical resolution of 128. The offset factor determines whereabouts along the 100-wide sample the display is taken from, and altering this moves the 40-byte wide sample window to left or right through the 100 bytes of data. If the space bar is depressed during the display, a new offset factor may be entered to change the position of the window. If a number greater than 60 is entered here, the program re-enters the sampling routine so that a new sample may be taken. The accompanying photograph shows the screen representation of the 100 Hz signal shown in the oscilloscope trace photographs referred to earlier. As may be seen, reproduction quality is quite good given the constraints imposed by the UK101's low resolution graphics.

## CUSTOMISED WAVEFORM GENERATOR

The D/A section of the machine code routine referred to above may also be used on its own for the creation of customised waveforms. The 100 bytes of memory space above 0290 hex can be filled from BASIC using SIN, COS, LOG, RND, power or other functions of the user's choice. The replay routine in machine code may then be executed to generate the waveform repeatedly, and at speed, through the D/A converter at 61320.

The number of bytes of digitised waveform sent to the D/A converter before the sequence is repeated is determined by the data held in location 558 dec. This may be POKEd with any value from 1 to 100 so as to achieve a smooth follow on from the end of one displayed cycle of the waveform to the next.

**Table 7.7. Sample Oscilloscope in BASIC**

```
100 REM INTERFACING UK101 PROGRAM 23
110 REM SAMPLE SCOPE PROGRAM
120 GOSUB 360
130 REM
140 FORA=1TO16:PRINT:NEXT
150 PRINT,"SAMPLE SCOPE"
160 PRINT,"ON CHANNEL 0"
170 PRINT:PRINTQQ$
180 GOSUB270
190 POKE11,48:POKE12,2:X=USR(X)
200 PRINT:PRINT:PRINT"     SAMPLE MADE"
210 PRINT:PRINT
220 INPUT"    REPLAY RATE (1-255)";RR
230 PRINT:PRINT"    REPLAY IN PROGRESS"
240 PRINT"    USE RESET KEYS TO EXIT"
250 POKE556,RR
260 POKE11,80:POKE12,2:X=USR(X)
270 REM INITIALISATION
280 C=0:REM SET CHANNEL
290 N=100:REM SET NO OF SAMPLES (100 = MAX)
300 INPUT"    SAMPLE RATE 1 (HIGH) TO 255 (LOW)";R
310 IFR<10RR>255THEN300
320 POKE559,C
330 POKE558,N
340 POKE557,R
350 RETURN
360 REM MACHINE CODE PROGRAM
370 REM LOCATIONS  560  TO  615
380 DATA 174 , 46 , 2 , 173 , 47 , 2 , 141 , 135 , 239 , 172
390 DATA 45 , 2 , 136 , 234 , 234 , 234 , 208 , 250 , 173 , 135
400 DATA 239 , 157 , 144 , 2 , 202 , 208 , 232 , 96 , 0 , 0
410 DATA 0 , 0 , 174 , 46 , 2 , 189 , 144 , 2 , 141 , 136
420 DATA 239 , 172 , 44 , 2 , 136 , 234 , 234 , 234 , 208 , 250
430 DATA 202 , 208 , 238 , 76 , 80 , 2
440 FORC=0 TO  55
450 READ I
460 POKE  560 +C, I
470 NEXT
480 RESTORE
490 RETURN
```

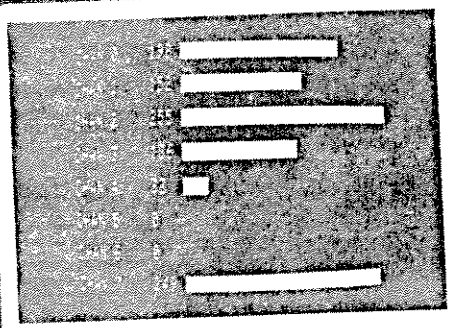
**Table 7.8. Supplement to Sample Oscilloscope for Screen Display of Waveform**

```
205 REM INTERFACING UK101 PROGRAM 24
206 REM GIVES SCREEN DISPLAY OF
207 REM SAMPLED WAVEFORM
208 REM LOAD ON TOP OF PROGRAM 23
210 GOTO7000
7000 REM SCREEN DISPLAY ROUTINE
7050 INPUTZZ
7060 IFZZ<0 OR ZZ>60THEN110
7070 ZZ=60-ZZ
7100 X=54222
7110 FORQ=1TO16:PRINT:NEXT
7120 S=656 +ZZ+40
7130 FORN=0TO40
7140 H=INT((PEEK(S-N))/2)
7150 H1=INT(H/8)
7160 H2=INT((H-H1*8))
7170 POKEN+X-64*H1,H2+128
7200 NEXT
7250 POKE530,1
7260 POKE57088,253
7270 IFPEEK(57088)<>239THEN7260
7280 POKE530,0
7290 GOTO7000
```

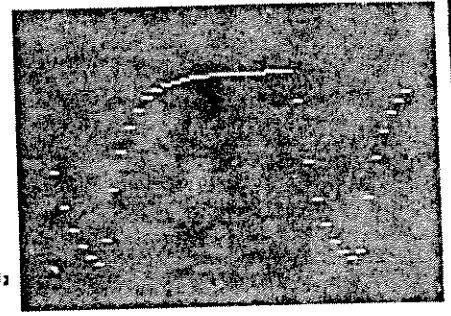
### A NOTE ON THE USE OF NEW MONITORS

All programs in this series are fully compatible with the UK101's original monitor, and with Technomatic's CE1 monitor. But difficulties may arise with other new monitors when using the 3 or 4 programs that contain machine code routines located at 0230 hex, such as the Interrupt Clock and the Sample Scope program. The reason for this is that monitors such as Cegmon, Wemon and Compshop's new monitor use some of the space above 022A hex for scratchpad purposes. With Cegmon this space can be kept free by disabling the screen editor, and using the original screen handling routines. But to get around the problem more generally, an alternative set of programs have been prepared for the Interrupt Clock and Data Logger, and the Sample Scope, which relocate the machine code routines at the top of memory. There is no space to list these programs here, but they are included in the cassette containing the 24 numbered programs from this series, available from *Technomatic Ltd.*

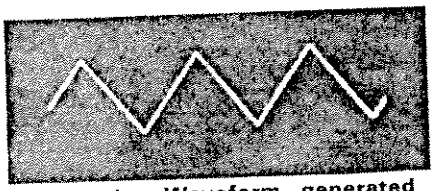
On Wemon and Compshop's new monitor, but not on the CE1 or Cegmon, the IRQ vector has also been relocated, and this must be accounted for in programs such as the Interrupt Clock and Data Logger, which make use of IRQ, by modifying lines 720, 740 and 760 of the assembler listing in Table 7.1. Further details on this modification are given in the data supplied with the cassette, since the shifted version of these programs must in any case be used with these two monitors.



Screen Photograph of Eight Channel Display Program

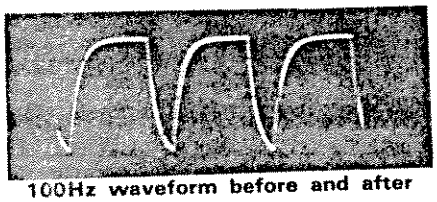


Screen display of sampled 100Hz waveform

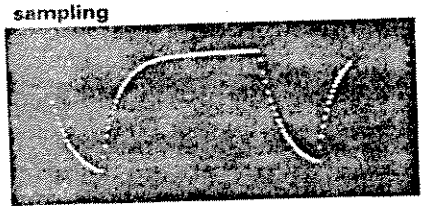


Triangular Waveform generated by the Customised Waveform Generator, and output through the D/A Converter

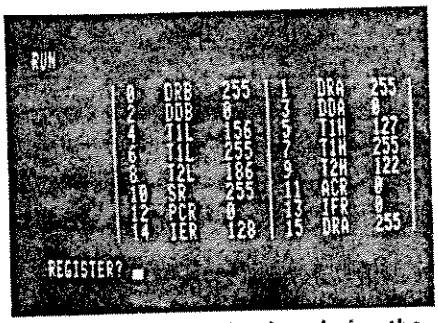
Screen Photograph of Data Logger



100Hz waveform before and after sampling



# INTERFACING COMPUKIT




Screen photograph taken during the running of the 6522 Handling Program (Program 12) described last month

**NEXT MONTH** . . . we shall be publishing an **EPROM PROGRAMMER** which may be driven from this interface system.

Additionally, we will be taking a look at **Chromosonic's Colour Board** in use with the UK101

**Constructor's Note**

In order to improve stability in the offset adjustment on the D/A converter op amp IC11, R10 on the Analogue Board should be increased to 47k.

Filling the data space using the program below will produce an even triangular wave made up of exactly 100 samples:

```
2000 FORA=0TO99
2010 B=4×A
2020 IFA>49THENB=4×(99−A)
2030 POKE656+A,B
2040 NEXT
2050 GOTO300
```

The accompanying photograph shows the waveform produced using this program. As may be seen, its shape is good, and since it is constructed of a relatively large number of samples, its true step nature is not apparent in the photograph. It should be noted when creating other waveforms using the machine code replay sequence, that although the storage space for the waveform begins at 0290 hex, this location is the *last* rather than the *first* to be accessed, so that the *last* byte of waveforms placed into these locations for subsequent replay should be at 0290 hex, the first being at 0290+N-1, where N is the number of samples to be replayed.

## CONCLUSION

Although this is the final article of the present series, it by no means exhausts the possible applications of the UK101 Decoding Module and Analogue Board. Further applications are planned for future issues of P.E., and it is hoped that ideas will also be contributed by UK101 and Superboard users through the columns of P.E.'s MicroPrompt. ★