PHILIPS

Electronic components and materials

Technical note 087

# Audio cassette recorder interfaces for the 2650 microprocessor

The compact cassette is an ideal medium for data storage and transport for microcomputer systems: it is small, robust and simple to use. However, the cost of a digital cassette recorder can be disproportionately high compared with the rest of a microcomputer system.

An economical solution is to use an audio cassette recorder. This is a relatively cheap device, which many people already possess. The use of an audio cassette recorder obviously involves compromises in performance, but, with a carefully designed interface, the result can be a cheap, reliable data-storage system. Two approaches are described here: firstly, a simple hardware interface; and secondly, a more sophisticated hardware interface. Error-correcting software is described with the first interface although, in practice, this has been found unnecessary.

## 1 Simple hardware with error-correcting software

This approach uses a hardware interface designed from a minimum of easily obtained components. No changes to the cassette recorder are required, nor is special software necessary. The interface is capable of operating at up to 1200 baud.

To complement this simple hardware design, an error-correcting program is described to enable reliable operation.

### Hardware

Data is recorded by switching a 5 kHz oscillator on and off and recording the resulting signal on the cassette. Thus the presence of a 5 kHz tone represents a one, and the absence of a 5 kHz tone represents a zero. An unrecorded tape will thus contain all zeros. Figure 1 shows the basic operation of the hardware. The 5 kHz tone has been chosen so that the circuit can be used with even the

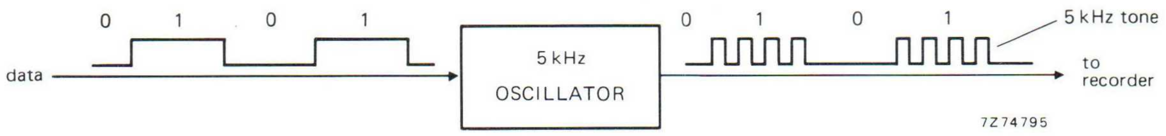

Fig. 1 Basic recording technique.

signetics

cheapest and simplest cassette recorders. This approach has been found extremely reliable in the laboratory, using standard cassettes in conjunction with the error-detecting Absolute Object format as described in Application Memo SS51.

## Recording circuit

Figure 2 shows two circuits to produce the encoded signal for the cassette recorder. Alternatively, the 5 kHz signal can be generated by software. $R_6$ in Fig. 2(a) is used to ensure fast starting of the oscillator. However, the value must be such that the oscillator can be turned on and off correctly.

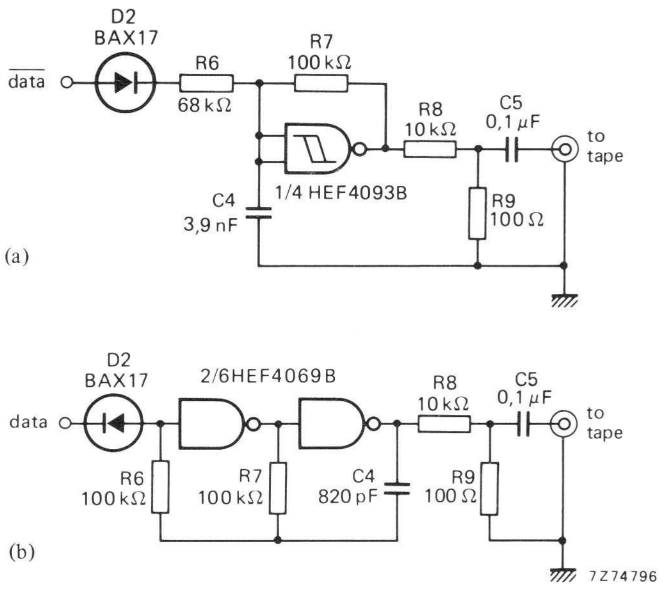

Fig. 2 Two examples of switched oscillator circuits.

## Playback circuit

The playback circuit is shown in Fig. 3. The first half of the LM393 is used to square the signal from the cassette recorder, while the second half acts as a monostable to provide pulses of equal length. When a 5 kHz signal is received, the resulting pulse train prevents the voltage across the capacitor, on the input to the Schmitt trigger, from rising above the threshold voltage, causing a one to be presented on the data output. When the input from the recorder is a zero-signal (no 5 kHz tone), the capacitor is not discharged and the output is a zero. Figure 4 shows the waveforms through the playback circuit.

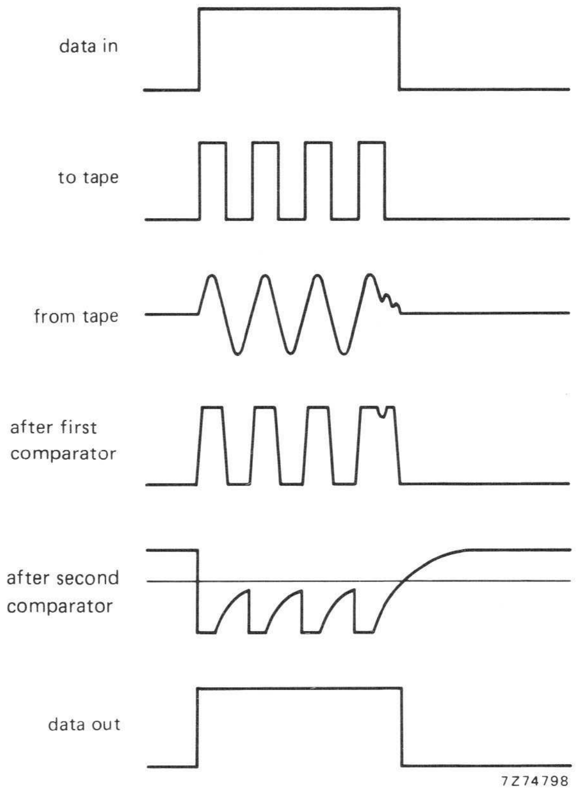

Fig. 4 Waveforms associated with the circuit of Figs 2 and 3.

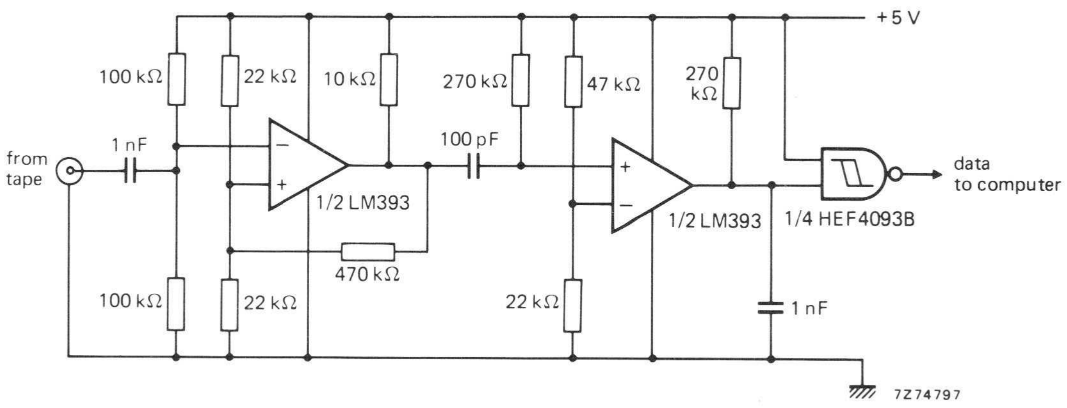

Fig. 3 Playback circuit.

## Software

The software determines the data format, the baud rate and the error-correcting characteristics of the data storage system. The audio cassette recorder does not have facilities such as start, stop, reverse etc. that are common to digital cassette recorders. This severely limits any actions that can be taken when errors are detected during reading. A program to provide error-correction for drop-out errors up to 511 bits long every 2049 bits is described here. Alternatively, when using a TWIN system, a suitable output format is available from the WHEX command. Although this alternative does not provide error-correction, it has been found extremely reliable in the laboratory.

### Data format

The data is recorded in blocks of 32 data bytes plus 6 control bytes. The format of a block is shown in Fig. 5. The number-of-bytes entry is fixed at 32 except for the last block, where the value 66 is used to indicate end of data transmission. The two block control characters (BCCs) are obtained by performing an exclusive OR operation with each data byte on the BCC, and then rotating the BCC one place left. The initial BCC is an 'FF' byte.

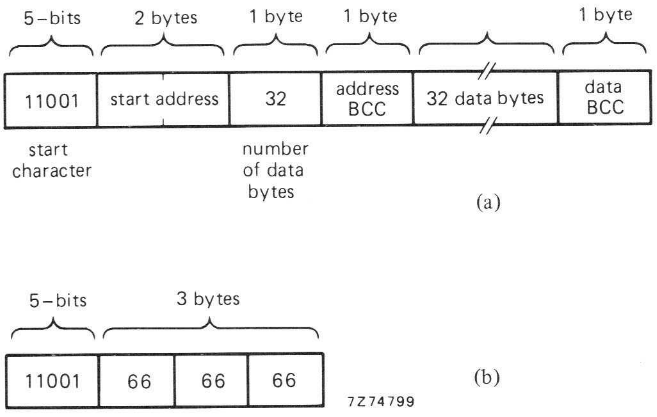

Fig. 5 Data format.
(a) data block;
(b) last block to indicate end-of-data.

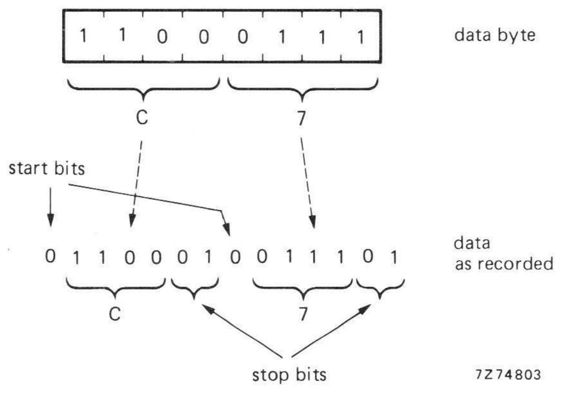

Fig. 6 Splitting a data byte and adding start and stop bits.

Because clock information is not recorded, data should be treated as asynchronous, and each byte thus preceded and followed by start and stop bits respectively. In Fig. 6, a data byte is split into two hexadecimal characters and each of these preceded by a start bit (0) and followed by two stop bits (01).

The entire block (data plus control bytes, with start and stop bits) is recorded three times to form a triad. The address counter is then incremented by 32 and the next block recorded three times. The last block contains no data: it is used to indicate end-of-data and consists of a start character followed by three bytes containing the value 66. The program then detects a block length of 66 and recognizes end-of-data.

Automatic control of the recording level is a common feature among the cheaper cassette recorders. This can cause a high noise level when recording zeros and a distorted signal when recording the first characters of a sequence. These effects can be overcome by preceding the data by a leader of ones of about 600 ms, to allow the recording level to stabilize. A short sequence of ones is recorded before each data block to provide synchronization following possible drop-outs.

### Bit and byte rate

These rates determine the storage capacity of the tape and the sensitivity of the recording to drop-outs. As the bit rate is increased, more bits are affected by a bad section of tape and the number of cycles of the 5 kHz tone recorded per bit is reduced. The byte rate can be increased by decreasing the number of repetitions of each block.

### Error correction

The error-correction system is based on the assumption that errors will occur with a length of 511 bits or less, only once in any sequence of 2049 bits. This means that at least one block in every triad should be error-free.

The read section of the program looks for a start character and then stores the start address, block length and address BCC. If the address BCC is correct, the program stores the following 32 data bytes. If the data BCC is correct, the next start address (old address + 32) is written into an address control location (ADCO). The following data blocks are only stored if the address in ADCO corresponds to start address of the block and the address BCC is correct. This ensures that the repetitions of a block read correctly the first time do not overwrite that block.

If either of the BCCs is incorrect, or if the address in ADCO does not correspond to the block start address, the program branches to ERRA and adds one to the error count (ERCN). The data could still be read correctly if ERCN is 4 in the case where errors occur in the last two blocks of one triad and the first two blocks on the next triad. Consequently, the program aborts when ERCN reaches 5. If all three blocks of a triad contain errors, the address in ADCO will be permanently out of step with the following block start addresses, and the error count will increment on each block until the program aborts.

## Performance

The performance of the system is summarized in Table 1.

*TABLE 1 Performance of simple hardware and error-correcting software recording system*

| | |
|---|---|
| recorded bit rate (bits/s) | 1200 |
| data byte rate (bytes/s) | 23,4 |
| storage capacity (bytes) on C60 cassette | 126 k |
| maximum drop-out for guaranteed correction | 511 bits in 2049 bits |
| transport speed tolerance | ± 7,5% |
| program size | 416 bytes + 9 RAM |

## Cassette recorder operation

### Writing

First, the start and end addresses of the data to be written (2 bytes each) must be written into the locations at label ADRS (line 16). The tape recorder should then be switched to record and an oral identification of the tape recorded. The processor is then directed to the write start address. STRT (line 28). The program will then output a leader of ones for 600 ms, followed by the data bytes between the addresses specified and halt at address 4F0 (line 105).

### Reading

To read back the data from cassette, the processor is directed to the READ start address STAR (line 107) and the recorder switched to replay. The program will halt at FINI (line 158) if the data has been read correctly or at ERRS (line 165) if the data was not recovered correctly.

# 2 Frequency-shift keying system

In the system previously described, loss of a cycle of the 5 kHz tone would result in corruption of the data signal. This could be overcome by increasing the time constant of the detector and decreasing the baud rate to give a greater degree of redundancy. This is done in the FSK system, which uses a baud rate of 880 baud compared to 1200 baud for the simple system. The FSK system also possesses a higher noise immunity as it must discriminate between two separate frequencies instead of using a simple amplitude detector.

The FSK method requires considerably more hardware than the simple method previously described. The software storage requirement is about the same, although error-correction is not included. However, the software offers more flexibility in the use of read/write operations. The data is stored at a rate of 880 baud, so that each side of a standard C60 cassette can easily store 64 kbytes. The data is written in blocks of variable size, with the relevant source or destination memory addresses specified in the write or read commands.

## FSK recording

In the FSK system, a one is represented by a high tone and a zero by a low tone. The choice of these frequencies is determined by the bandwidth of the recorder, the redundancy and transfer rate of the recording and the frequency ratio required by the demodulator.

The maximum frequency, $f_1$, is about 7 kHz and the transfer rate has been selected as 880 baud. The whole multiple of the transfer rate that is nearest 7 kHz is 8 x 880 = 7,04 kHz. This means that a one will be recorded as a series of eight sine-waves. Because the demodulator will still work correctly with one or two cycles missing, the requirement for redundancy in the recording is satisfied.

The frequency for a zero, $f_0$, is determined by the frequency ratio required by the demodulator. A ratio of $f_0/f_1 = 0,8$ is suitable and allows the zero to be recorded as six cycles of 5,28 kHz.

## Data format

The eight-bit bytes are recorded serially, with an odd-parity bit for each byte. The use of the odd-parity bit ensures that each recorded byte contains a one, which is required to maintain synchronization.

The data format is shown in Fig. 7. The data is preceded by a leader of ones for about 20 seconds. This allows the recording level to stabilize and provides synchronization of the clock on playback whilst also easing manual operation of the recorder. The leader is followed by a number of zeros to act as start bits before the data commences.

In order to provide a greater degree of synchronization, a Return-to-Zero FSK method is used. This means that a zero is represented by $f_0$ for a full bit-time, while a one is represented by $f_1$ for only 60% of a bit-time, the remaining 40% being $f_0$. This is illustrated in Fig. 8.

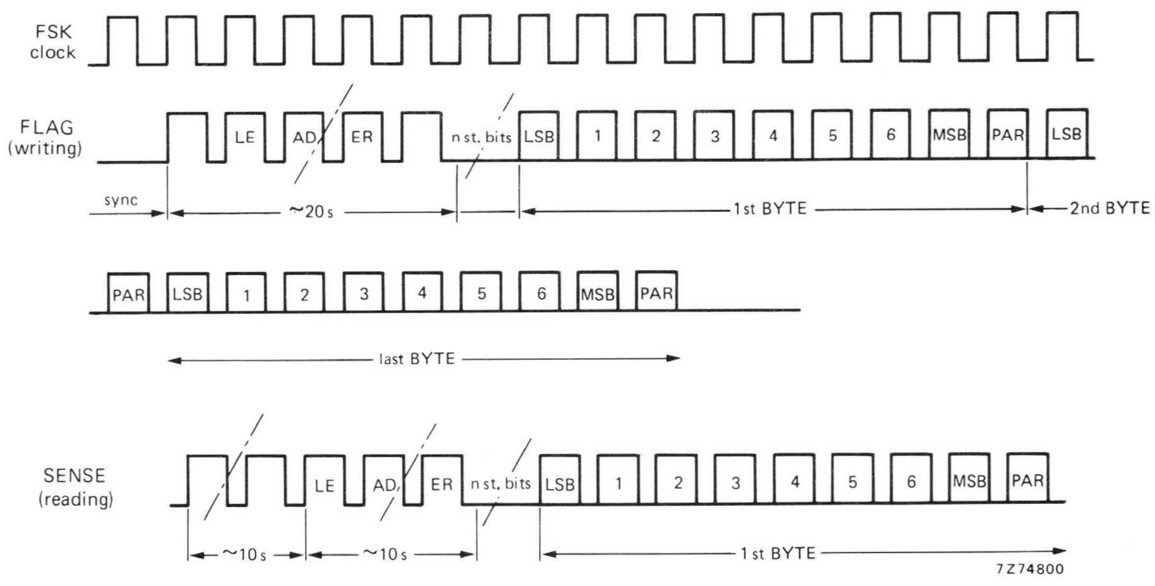

Fig. 7 Data format of FSK system.

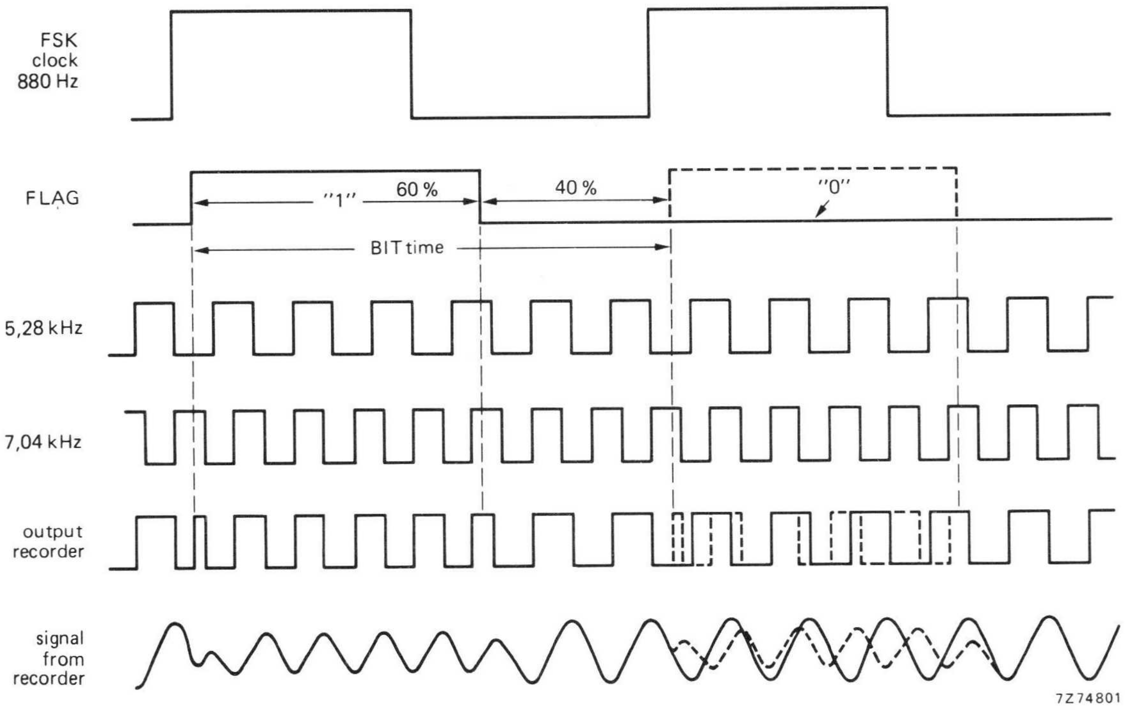

Fig. 8 Return-to-Zero FSK recording method.

## Hardware

The cassette recorder used for this system was a Philips Automatic Recorder type N2215. Brief specifications of this recorder are given in Table 2. The recorder has an automatic recording level control and a three-digit tape position counter. Any recorder of similar or better specification could be used in this application.

The interface between the recorder and the microprocessor system is shown in block diagram form in Fig. 9. Only six connections are required, three to the microcomputer and three (2 signals and a common) to the recorder. Figure 10 shows the circuit diagram of the interface hardware.

*TABLE 2 Recorder specification in brief*

| | |
|---|---|
| bandwidth | 80 – 10 000 Hz (±8 dB) |
| output | 1 V rms into 12 kΩ |
| required input | 0,2 mV into 2 kΩ |
| wow and flutter | < 0,4% |
| tape speed | 4,76 cm/s ±2% |

### Modulator

Phase-locked loops are used to generate the three frequencies required: 880 Hz for baud rate, 5,28 kHz for $f_0$ and 7,04 kHz for $f_1$. The baud rate is determined by $U_3$, and can be adjusted by $R_1$. $U_4$ and $U_5$ generate $f_1$ and $f_0$ respectively, being synchronized to the 880 Hz baud rate clock. The NAND gates of $U_2$ are used so that the FLAG output of the 2650 microprocessor controls the frequency to be recorded ($f_0$ or $f_1$), see Fig. 8.

Although the signals generated by the phase-locked loops are square waves, the limited bandwidth of the recorder suppresses the harmonics, reducing the signals to the fundamental sine-waves.

Elements $U_7$ and $U_8$, four-bit counters, are used to improve the synchronization of the frequencies $f_0$ and $f_1$.

### Demodulator

The demodulator is a phase-locked loop, $U_6$, set to a frequency between $f_0$ and $f_1$. When driven by the output of the recorder, the phase-locked loop develops an error voltage across pins 6 and 7, which is then detected as zero or one by the comparators, $U_2$. One output of the comparators is used to synchronize the 880 Hz baud rate clock and the other provides the demodulated data to the SENSE input of the microprocessor. When there is no input from tape, the output of the demodulator must be inhibited. This is accomplished by the comparator $U_2$, pins 8, 9 and 14, which inhibits the demodulator output when there is no signal from the tape.

The demodulator can be easily adjusted by means of $R_2$. While the leader (all ones) is read, $R_2$ is adjusted to make the signal across pins 4 and 5 of $U_2$ symmetrical.

## Software

The software is a program that reacts to commands typed on the keyboard/teletype by the user. The actions that can be specified in the commands are: write a number of bytes, or read a number of bytes. The number of bytes is determined by the start and end addresses that must be specified, thus the read/write action can access any part of the memory.

The program relies upon the use of the 2650 PIPBUG* monitor/loader: this is assumed to be resident in the microcomputer system. Figure 11 shows a simplified flow chart for the program. A listing is given in the Appendix B.

The program does not provide error correction, although this could be incorporated in the same manner as in system 1 without difficulty.

* See 2650 Application Memo SS50.

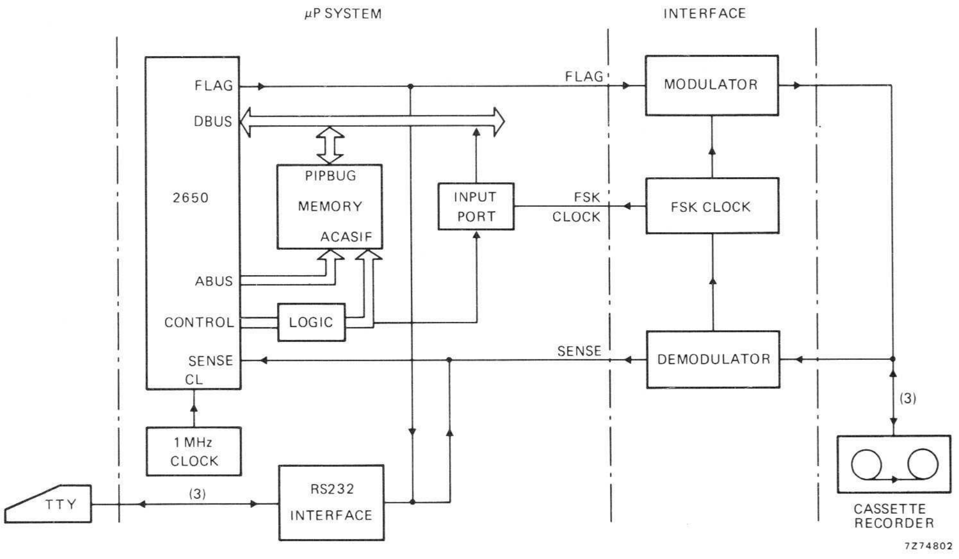

Fig. 9 Block diagram of the FSK recording system.

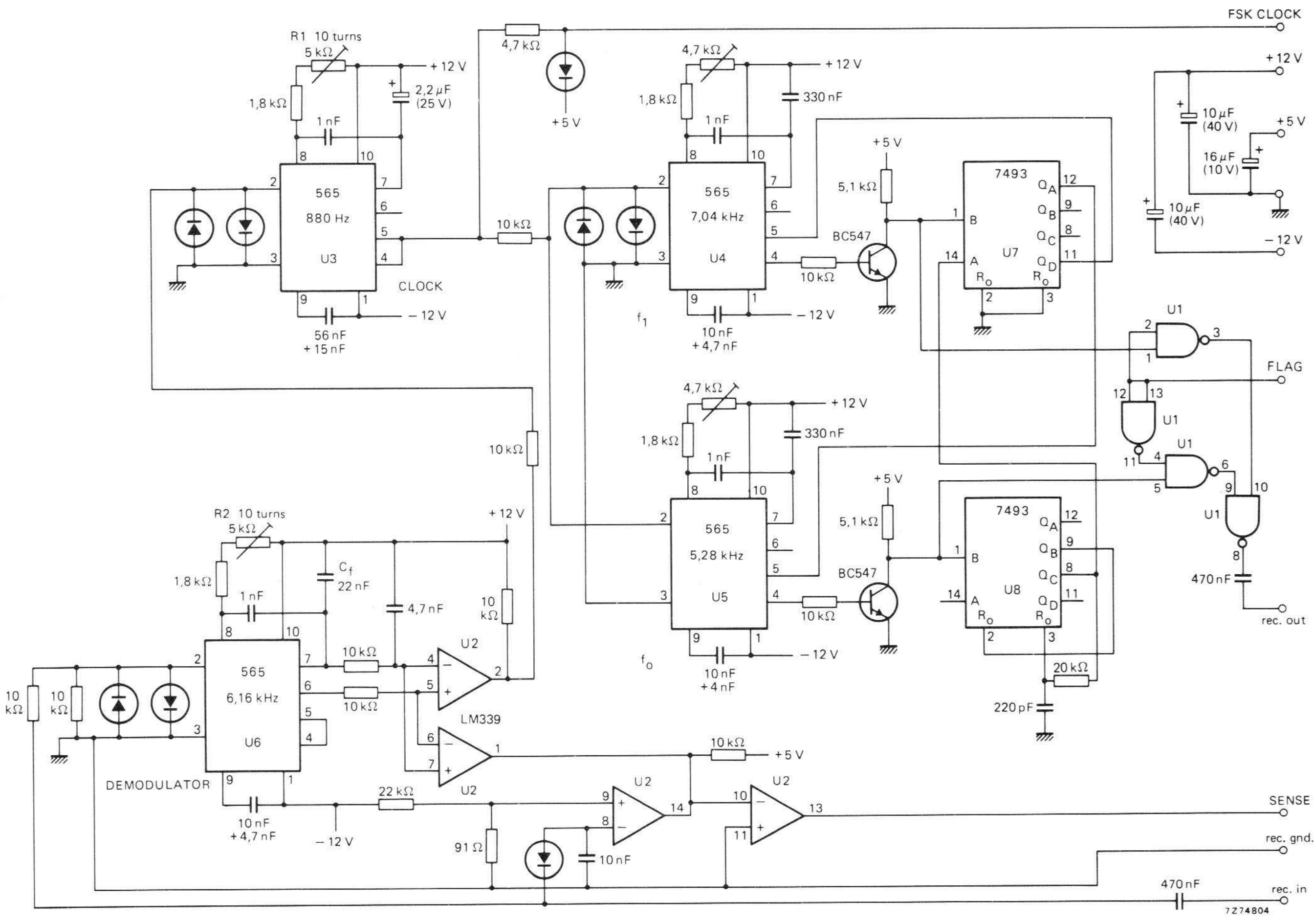

Fig. 10 Circuit diagram of the FSK interface.

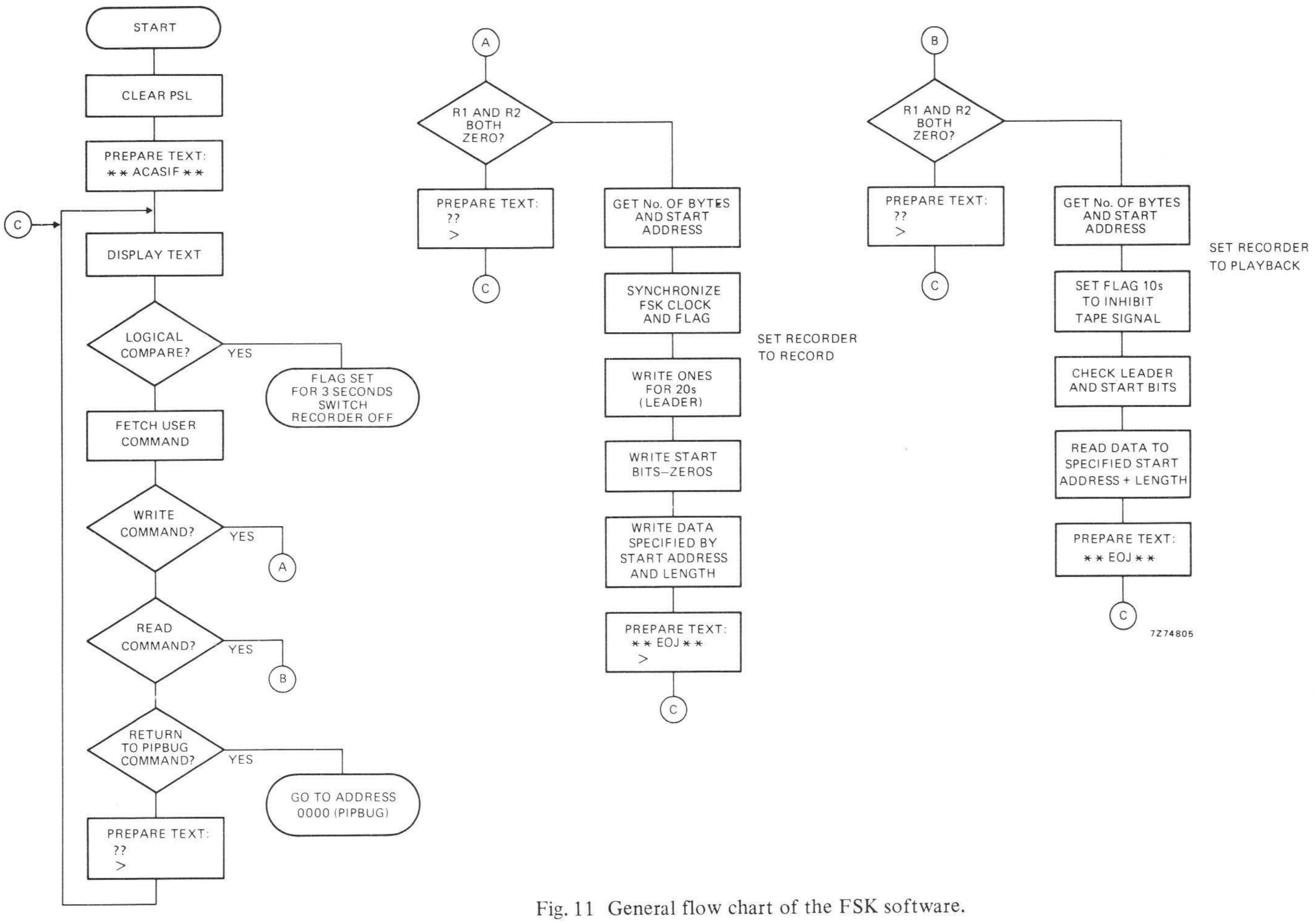
START
CLEAR PSL
PREPARE TEXT: ** ACASIF **
DISPLAY TEXT
LOGICAL COMPARE?
YES
FLAG SET FOR 3 SECONDS SWITCH RECORDER OFF
FETCH USER COMMAND
WRITE COMMAND?
YES
READ COMMAND?
YES
RETURN TO PIPBUG COMMAND?
YES
GO TO ADDRESS 0000 (PIPBUG)
PREPARE TEXT: ?? >
R1 AND R2 BOTH ZERO?
PREPARE TEXT: ?? >
GET No. OF BYTES AND START ADDRESS
SYNCHRONIZE FSK CLOCK AND FLAG
SET RECORDER TO RECORD
WRITE ONES FOR 20s (LEADER)
WRITE START BITS–ZEROS
WRITE DATA SPECIFIED BY START ADDRESS AND LENGTH
PREPARE TEXT: ** EOJ ** >
SET RECORDER TO PLAYBACK
SET FLAG 10s TO INHIBIT TAPE SIGNAL
CHECK LEADER AND START BITS
READ DATA TO SPECIFIED START ADDRESS + LENGTH
PREPARE TEXT: ** EOJ **
A
B
C
7Z74805

Fig. 11 General flow chart of the FSK software.

## Cassette recorder operation

If careful use is made of the tape position counter, several independent blocks of data can be stored on the same cassette. Before recording, the tape should be rewound so that the beginning of the oxide has just passed the recording head. The tape counter is then set to zero. The first block of data can then be recorded, the tape wound forwards until the position counter reads, for example, 50, and then the next block of data written. Provided that the user keeps accurate notes of the position at which the data is recorded, it will be retrievable. A standard C60 cassette can easily contain up to eight blocks of data, each of 8 kbytes plus leader, on each side.

When the tape has been positioned correctly, the cassette program should be entered. The text, ** ACASIF **, will be displayed and the read/write command entered. The command format is:

```
R nnnn aaaa
W nnnn aaaa
P
```

where nnnn is the number of bytes to be transferred, four hexadecimal digits, aaaa is the start address of the data, four hexadecimal digits and R, W, P represent the commands read, write or return-to-PIPBUG. If the command is not rejected (?? displayed) the recorder should be switched to playback/record as appropriate. When the data transfer is complete, the text, ** EOJ **, is displayed and the program waits for the next command.

# Appendix 1

Assembler listing for the error-correcting program for system 1.

```
      PIP ASSEMBLER VERSION 2 LEVEL 1                                           PAGE    1

LINE  ADDR  LABL  B1 B2 B3 B4 ERROR SOURCE

  1         0000                          R0   EQU    0
  2         0001                          R1   EQU    1
  3         0002                          R2   EQU    2
  4         0003                          R3   EQU    3
  5         0000                          Z    EQU    0
  6         0001                          P    EQU    1
  7         0002                          N    EQU    2
  8         0003                          UN   EQU    3
  9         0000                          EQ   EQU    0
 10         0001                          GT   EQU    1
 11         0002                          LT   EQU    2
 12         0010                          RS   EQU    H'10'
 13         0008                          WC   EQU    H'08'
 14         0002                          COM  EQU    H'02'
 15                                            ORG    H'440'
 16         0440                          ADRS RES    4         STRT/END ADRS
 17         0444                          HOLD RES    1
 18         0445                          ADCO RES    1         ADRS CONTROL
 19         0446                          STOR RES    1
 20         0447                          ERCN RES    1         ERROR COUNT
 21         0448                          ADRF RES    1         ADRS FLAG
 22                                       * PROGRAM STORES 2650 MEMORY BYTES ON AUDIO CASSETTE-
 23                                       * WITH ERROR CORRECTION FOR 511 BITS DROP OUT IN 2049-
 24                                       * BITS AND SPEED VARIATION OF + OR - 7.8 PER CENT.-
 25                                       * DATA RATE IS 23.4 BYTES/SEC (46.8 HEX CHARS/SEC).
 26                                       * WRITE CASSETTE STRT
 27   0449  0449  75 FF                   STRT CPSL    H'FF'
 28   044B        76 40                        PPSU    H'40'
 29   044D        0C 04 42                     LODA,R0 ADRS+2    SUB STRT & END ADRS
 30   0450        0D 04 43                     LODA,R1 ADRS+3
 31   0453        AD 04 41                     SUBA,R1 ADRS+1
 32   0456        77 08                        PPSL    WC
 33   0458        AC 04 40                     SUBA,R0 ADRS
 34   045B        75 08                        CPSL    WC
 35   045D        CC 04 42                     STRA,R0 ADRS+2    ADRS+2 HOLDS NO. OF BYTES
 36   0460        CD 04 43                     STRA,R1 ADRS+3
 37   0463        07 00                        LODI,R3 0
 38   0465        04 FF                        LODI,R0 H'FF'
 39   0467        B0                           WRTC,R0
 40   0468  0468  20                      TIM1 EORZ    R0        LEADER OF N*1
 41   0469        F8 7E                        BDRR,R0 $
 42   046B        F8 7B                        BDRR,R3 TIM1
 43   046D  046D  06 03                   BGN  LODI,R2 3
 44   046F  046F  07 00                   BGN0 LODI,R3 0
 45   0471        05 FF                        LODI,R1 H'FF'
 46   0473        77 10                        PPSL    RS
 47   0475        07 06                        LODI,R3 6
 48   0477        04 FF                        LODI,R0 H'FF'
 49   0479        B0                           WRTC,R0
 50   047A  047A  20                      TIM2 EORZ    R0        RE-SYNC FOR READ PROG.
 51   047B        F8 7E                        BDRR,R0 $
 52   047D        F8 7B                        BDRR,R3 TIM2
```

```
LINE  ADDR  LABL  B1 B2 B3 B4 ERROR SOURCE

  53  047F        04 CF                       LODI,R0 H'CF'      O/P STRT CHAR CF
  54  0481        3F 05 D0                    BSTA,UN OPR0
  55  0484        75 10                       CPSL    RS
  56  0486  0486  0F 64 40               BGN1 LODA,R0 ADRS,R3    BEGIN ADDRESS O/P
  57  0489        3F 05 B3                    BSTA,UN OBYT
  58  048C        21                          EORZ    R1
  59  048D        D0                          RRL,R0
  60  048E        C1                          STRZ    R1         BCC STEP
  61  048F        87 01                       ADDI,R3 1
  62  0491        E7 02                       COMI,R3 2
  63  0493        98 71                       BCFR,EQ BGN1       BR TWICE
  64  0495        04 20                       LODI,R0 32
  65  0497        3F 05 B3                    BSTA,UN OBYT       O/P BLK LENGTH
  66  049A        21                          EORZ    R1
  67  049B        D0                          RRL,R0
  68  049C        3F 05 B3                    BSTA,UN OBYT       O/P ADRS BCC
  69  049F        05 FF                       LODI,R1 H'FF'      ADRS & BCC O/P COMPLETE
  70  04A1        07 00                       LODI,R3 0
  71  04A3  04A3  0F E4 40               BGN2 LODA,R0 *ADRS,R3   BEGIN DATA O/P
  72  04A6        3F 05 B3                    BSTA,UN OBYT
  73  04A9        21                          EORZ    R1
  74  04AA        D0                          RRL,R0
  75  04AB        C1                          STRZ    R1
  76  04AC        87 01                       ADDI,R3 1
  77  04AE        E7 20                       COMI,R3 32
  78  04B0        98 71                       BCFR,EQ BGN2       BR 32 TIMES
  79  04B2        3F 05 B3                    BSTA,UN OBYT       O/P DATA BCC
  80  04B5        FE 04 6F                    BDRA,R2 BGN0
  81  04B8        20                          EORZ    R0         ADD 32 TO STRT ADRS
  82  04B9        05 20                       LODI,R1 32
  83  04BB        8D 04 41                    ADDA,R1 ADRS+1
  84  04BE        77 08                       PPSL    WC
  85  04C0        8C 04 40                    ADDA,R0 ADRS
  86  04C3        75 08                       CPSL    WC
  87  04C5        CD 04 41                    STRA,R1 ADRS+1
  88  04C8        CC 04 40                    STRA,R0 ADRS
  89  04CB        0D 04 43                    LODA,R1 ADRS+3     SUB 32 FROM NO. OF BYTES
  90  04CE        0C 04 42                    LODA,R0 ADRS+2
  91  04D1        A5 20                       SUBI,R1 32
  92  04D3        77 08                       PPSL    WC
  93  04D5        A4 00                       SUBI,R0 0
  94  04D7        75 08                       CPSL    WC
  95  04D9        CD 04 43                    STRA,R1 ADRS+3
  96  04DC        CC 04 42                    STRA,R0 ADRS+2
  97  04DF        9E 04 6D                    BCFA,N  BGN
  98  04E2        04 CF                       LODI,R0 H'CF'      STRT CHAR
  99  04E4        3F 05 D0                    BSTA,UN OPR0
 100  04E7        07 03                       LODI,R3 3
 101  04E9  04E9  04 66                  EOFI LODI,R0 H'66'      END CHAR
 102  04EB        3F 05 B3                    BSTA,UN OBYT
 103  04EE        FB 79                       BDRR,R3 EOFI
 104  04F0        40                          HALT               WRITE TO CASSETTE COMPLETE
```

```
LINE  ADDR  LABL  B1 B2 B3 B4 ERROR SOURCE

 105                                   * READ CASSETTE STRT
 106  04F1  04F1  75 FF                STRR CPSL    H'FF'
 107  04F3        76 40                     PPSU    H'40'
 108  04F5        04 01                     LODI,R0 1
 109  04F7        CC 04 48                  STRA,R0 ADRF
 110  04FA        20                        EORZ    R0
 111  04FB        CC 04 47                  STRA,R0 ERCN
 112  04FE  04FE  3F 05 71             FIND BSTA,UN IBYT        FIND START CHAR.
 113  0501        1B 7B                     BCTR,UN FIND
 114  0503  0503  75 10                NDST CPSL    RS
 115  0505        07 00                     LODI,R3 0
 116  0507        05 FF                     LODI,R1 H'FF'
 117  0509  0509  3F 05 71             L002 BSTA,UN IBYT
 118  050C        CF 64 40                  STRA,R0 ADRS,R3     STR STRT ADRS
 119  050F        21                        EORZ    R1
 120  0510        D0                        RRL,R0
 121  0511        C1                        STRZ    R1          BCC
 122  0512        87 01                     ADDI,R3 1
 123  0514        E7 03                     COMI,R3 3
 124  0516        98 71                     BCFR,EQ L002
 125  0518        04 66                     LODI,R0 H'66'       LOOK FOR END 66
 126  051A        EC 04 42                  COMA,R0 ADRS+2
 127  051D        1C 05 60                  BCTA,EQ FINI
 128  0520        3F 05 71                  BSTA,UN IBYT        3 ADRS IN
 129  0523        E1                        COMZ    R1          CHK ADRS BCC
 130  0524        9C 05 61                  BCFA,EQ ERRA
 131  0527        04 01                     LODI,R0 1
 132  0529        EC 04 48                  COMA,R0 ADRF
 133  052C        18 08                     BCTR,EQ HERE
 134  052E        0C 04 41                  LODA,R0 ADRS+1      CHK FOR NEXT BLOCK
 135  0531        EC 04 45                  COMA,R0 ADCO
 136  0534        98 2B                     BCFR,EQ ERRA
 137  0536  0536  07 00                HERE LODI,R3 0           ADRS CORRECT
 138  0538        05 FF                     LODI,R1 H'FF'
 139  053A  053A  3B 35                L003 BSTR,UN IBYT
 140  053C        CF E4 40                  STRA,R0 *ADRS,R3
 141  053F        21                        EORZ    R1
 142  0540        D0                        RRL,R0
 143  0541        C1                        STRZ    R1          BCC
 144  0542        87 01                     ADDI,R3 1
 145  0544        E7 20                     COMI,R3 32
 146  0546        98 72                     BCFR,EQ L003
 147  0548        3F 05 71                  BSTA,UN IBYT        32 BYTES IN
 148  054B        E1                        COMZ    R1          CHK DATA BCC
 149  054C        98 13                     BCFR,EQ ERRA
 150  054E        0C 04 41                  LODA,R0 ADRS+1      DATA CORRECT
 151  0551        84 20                     ADDI,R0 32
 152  0553        CC 04 45                  STRA,R0 ADCO        INC. ADCO BY 32
 153  0556        20                        EORZ    R0
 154  0557        CC 04 47                  STRA,R0 ERCN        RESET ERCN
 155  055A        CC 04 48                  STRA,R0 ADRF        RESET ADRF
 156  055D        1F 04 FE                  BCTA,UN FIND
```

```
     PIP ASSEMBLER VERSION 2 LEVEL 1                                                     PAGE    4

LINE  ADDR  LABL  B1 B2 B3 B4 ERROR SOURCE

 157  0560  0560  40                      FINI HALT               PROGRAMME READ CORRECTLY
 158  0561  0561  04 01                   ERRA LODI,R0 1
 159  0563        8C 04 47                     ADDA,R0 ERCN       INC. ERCN
 160  0566        CC 04 47                     STRA,R0 ERCN
 161  0569        E4 05                        COMI,R0 5
 162  056B        18 03                        BCTR,EQ ERRS       TOO MANY ERRORS
 163  056D        1F 04 FE                     BCTA,UN FIND
 164  0570  0570  40                      ERRS HALT               PROGRAMME NOT READ CORRECTLY
 165                                      * SUBROUTINES SUBROUTINES SUBROUTINES SUBROUTINES
 166                                      * I/P PGM BYTE IN TWO BYTES
 167  0571  0571  77 10                   IBYT PPSL     RS
 168  0573        06 00                        LODI,R2 0          FIRST/SECOND FLAG
 169  0575  0575  05 00                   ON0  LODI,R1 0
 170  0577        07 05                        LODI,R3 5
 171  0579  0579  30                      ON1  REDC,R0
 172  057A        1A 7D                        BCTR,N  ON1        BR IF BIT 7 = 1
 173  057C        3B 30                        BSTR,UN HABI       HALF BIT DELAY
 174  057E  057E  3B 2A                   ON2  BSTR,UN FBIT       FULL BIT DELAY
 175  0580        30                           REDC,R0
 176  0581        44 80                        ANDI,R0 H'80'      TAKE BIT 7
 177  0583        61                           IORZ    R1
 178  0584        C1                           STRZ    R1
 179  0585        D1                           RRL,R1
 180  0586        FB 76                        BDRR,R3 ON2        BR FOR 5 BITS
 181  0588        3B 20                        BSTR,UN FBIT       STOP BIT
 182  058A        D1                           RRL,R1
 183  058B        D1                           RRL,R1
 184  058C        D1                           RRL,R1
 185  058D        E5 C8                        COMI,R1 H'C8'
 186  058F        1C 05 03                     BCTA,EQ NDST
 187  0592        45 F0                        ANDI,R1 H'F0'      TAKE FIRST HALF OF BYTE
 188  0594        E6 01                        COMI,R2 1
 189  0596        18 07                        BCTR,EQ OUT
 190  0598        06 01                        LODI,R2 1          FLAG = 1
 191  059A        CD 04 44                     STRA,R1 HOLD       STR MS HALF BYTE
 192  059D        1B 56                        BCTR,UN ON0
 193  059F  059F  D1                      OUT  RRL,R1
 194  05A0        D1                           RRL,R1
 195  05A1        D1                           RRL,R1
 196  05A2        D1                           RRL,R1             XXXX3210
 197  05A3        01                           LODZ    R1
 198  05A4        8C 04 44                     ADDA,R0 HOLD       ADD TWO HALF BYTES
 199  05A7        75 10                        CPSL    RS
 200  05A9        17                           RETC,UN
 201  05AA  05AA  04 50                   FBIT LODI,R0 80         FULL BIT DELAY
 202  05AC        1B 02                        BCTR,UN HABD
 203  05AE  05AE  04 2B                   HABI LODI,R0 43         HALF BIT DELAY
 204  05B0  05B0  F8 7E                   HABD BDRR,R0 £
 205  05B2        17                           RETC,UN
 206                                      * O/P R0 IN TWO BYTES AND HOLD R0
 207  05B3  05B3  77 10                   OBYT PPSL     RS
 208  05B5        C2                           STRZ    R2           PUT BYTE IN R2
```

```
      PIP ASSEMBLER VERSION 2 LEVEL 1                                    PAGE    5

 LINE  ADDR  LABL  B1 B2 B3 B4 ERROR SOURCE

  209  05B6        CC 04 44                 STRA,R0 HOLD
  210  05B9        46 0F                    ANDI,R2 H'0F'
  211  05BB        86 50                    ADDI,R2 H'50'     R2 = (0101)3210
  212  05BD        44 F0                    ANDI,R0 H'F0'
  213  05BF        84 05                    ADDI,R0 H'05'     R0 = 7654(0101)
  214  05C1        D2                       RRL,R2
  215  05C2        D2                       RRL,R2
  216  05C3        D2                       RRL,R2
  217  05C4        D2                       RRL,R2            R2 = 3210(0101)
  218  05C5        3B 09                    BSTR,UN OPR0      O/P R0 MS FIRST
  219  05C7        02                       LODZ    R2
  220  05C8        3B 06                    BSTR,UN OPR0      LS SECOND
  221  05CA        0C 04 44                 LODA,R0 HOLD      REPLACE R0
  222  05CD        75 10                    CPSL    RS
  223  05CF        17                       RETC,UN
  224  05D0  05D0  CC 04 46            OPR0 STRA,R0 STOR      PUT BYTE IN STOR
  225  05D3        05 05                    LODI,R1 5         5 COUNTER
  226  05D5        20                       EORZ    R0        START BIT
  227  05D6        3B 0E                    BSTR,UN OPBI      O/P BIT 7 OF R0
  228  05D8        0C 04 46                 LODA,R0 STOR      LOAD BYTE INTO R0
  229  05DB  05DB  3B 09               OP8  BSTR,UN OPBI      O/P 76543
  230  05DD        D0                       RRL,R0
  231  05DE        C0                       NOP
  232  05DF        F9 7A                    BDRR,R1 OP8
  233  05E1        04 80                    LODI,R0 H'80'     STOP BIT
  234  05E3        3B 01                    BSTR,UN OPBI
  235  05E5        17                       RETC,UN
  236  05E6  05E6  07 53               OPBI LODI,R3 83        O/P OPC7
  237  05E8        B0                       WRTC,R0
  238  05E9        FB 7E                    BDRR,R3 £
  239  05EB        17                       RETC,UN
  240                                       END


TOTAL ASSEMBLER ERRORS =     0
```

# Appendix 2

Assembler listing for the program for system 2 (FSK).

```
TWIN ASSEMBLER VER 2.0      *** AUDIO CASSETTE INTERFACE ***    PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0002                 *  J. PENNINGS 770504-1330
0003                 **********************************
0004                 * 2650 AUDIO CASSETTE INTERFACE*
0005                 **********************************
0006                 *
0007                 * DEFINITIONS OF SYMBOLS:
0008                 *
0009 0000            R0   EQU    0          PROCESSOR REGISTERS
0010 0001            R1   EQU    1
0011 0002            R2   EQU    2
0012 0003            R3   EQU    3
0013 0080            S    EQU    H'80'      PSU: SENSE
0014 0040            F    EQU    H'40'           FLAG
0015 0020            II   EQU    H'20'           INTERRUPT INHIBIT
0016 0007            SP   EQU    H'07'           STACKPOINTER
0017 00C0            CC   EQU    H'C0'      PSL: CONDITION CODE
0018 0020            IDC  EQU    H'20'           INTER DIGIT CARRY
0019 0010            RS   EQU    H'10'           REGISTER BANK SELECT
0020 0008            WC   EQU    H'08'           1=WITH, 0=NO CARRY
0021 0004            OVF  EQU    H'04'           OVERFLOW
0022 0002            COM  EQU    H'02'           1=LOG., 0=ARITH. COMP.
0023 0001            C    EQU    H'01'           CARRY/NO BORROW
0024 0000            Z    EQU    0          BRANCH COND: ZERO
0025 0001            P    EQU    1                       POSITIVE
0026 0002            N    EQU    2                       NEGATIVE
0027 0000            EQ   EQU    0                       EQUAL
0028 0001            GT   EQU    1                       GREATER THAN
0029 0002            LT   EQU    2                       LESS THAN
0030 0003            UN   EQU    3                       UNCONDITIONAL
0031 0000            A1   EQU    0                       ALL BITS ARE 1
0032 0002            N1   EQU    2                   NOT ALL BITS ARE 1
0033                 *
0034 0080            OP   EQU    H'80'      ODD PARITY.
0035 0080            BP8  EQU    H'80'      BYTE HAS 8 BITS.
0036 0008            DB8  EQU    H'08'
0037 0004            NSB  EQU    4          NUMBER OF START BITS
0038                 *
0039 02B4            COUT EQU   H'2B4'     PIPBUG CHAR. OUT SUBROUTINE.
0040 005B            LINE EQU   H'05B'     PIPBUG LINE SUBROUTINE.
0041 0427            BPTR EQU   H'427'     PIPBUG BUFFERPOINTER.
0042 0413            BUFF EQU   H'413'     PIPBUG BUFFER ADDRESS.
0043 02DB            GNUM EQU   H'2DB'     PIPBUG GET NUMBER SUBROUTINE.
0044 008A            CRLF EQU   H'08A'     PIPBUG LINE FEED SUBROUTINE.
0045                 *
0046                 *RAM DEFINITIONS.
0047 0000                ORG    H'440'
0048 0440            PSA  RES    2         POINTER START ADDRESS
0049 0442            PSI  RES    2         POINTER SIZE.
0050 0444            TXTA RES    2         STARTADDRESS OF TEXT.
```

```
LINE ADDR  OBJECT  E SOURCE

0052                 ******************************************************
0053                 *
0054                 *        COMMAND HANDLER FOR ACASIF
0055                 *
0056                 ******************************************************
0057                 *
0058                 *
0059                 *    W PPPP QQQQ    :WRITE H'PPPP' BYTES ON TAPE
0060                 *                   FROM MEMORY, START AT ADR. H'QQQQ'
0061                 *
0062                 *    R PPPP QQQQ    :READ H'PPPP' BYTES FROM TAPE
0063                 *                   TO MEMORY,STARTING AT ADR H'QQQQ'
0064                 *
0065                 *    P              :RETURN TO PIPBUG.
0066                 *
0067                 ******************************************************
0068 0446                 ORG     H'450'
0069 0450 75FF      NIIT CPSL     H'FF'
0070 0452 0405           LODI,R0 <TXT1     GET ADDRESS HIGH.
0071 0454 0535           LODI,R1 >TXT1     GET ADDRESS LOW.
0072 0456 3F0519    STC  BSTA,UN TEXT,     BRANCH TO SUBROUTINE TEXT
0073 0459 20             EORZ    R0        CLEAR R0.
0074 045A B502           TPSL    COM       TEST COMPARE BIT
0075 045C 3C0560         BSTA,Z  DLA3
0076 045F 3F005B         BSTA,UN LINE      FETCH COMMAND LINE.
0077                 *
0078 0462 20             EORZ    R0        CLEAR R0.
0079 0463 CC0427         STRA,R0 BPTR      CLEAR BUFFER POINTER.
0080                 *
0081 0466 0C0413         LODA,R0 BUFF      FETCH BUFFER DATA.
0082 0469 7702           PPSL    COM       LOGICAL COMPARE
0083 046B E457           COMI,R0 A'W'
0084 046D 1810           BCTR,EQ WRTA      BRANCH IF WRITE COMMAND.
0085 046F E452           COMI,R0 A'R'
0086 0471 1C056F         BCTA,EQ READ      BRANCH IF READ COMMAND.
0087 0474 E450           COMI,R0 A'P'
0088 0476 1C0000         BCTA,EQ 0         BRANCH TO PIPBUG IF P COMMAND
0089 0479 0405      SIZO LODI,R0 <TXT3     GET ADDRESS HIGH.
0090 047B 054F           LODI,R1 >TXT3     GET ADDRESS LOW.
0091 047D 1B57           BCTR,UN STC
0092                 *
0093 047F 3F02DB    WRTA BSTA,UN GNUM      GET NUMBER FROM BUFF IN R1,R2.
0094 0482 02             LODZ    R2
0095 0483 9805           BCFR,EQ SNZW      BRANCH IF R2 NONZERO.
0096 0485 01             LODZ    R1
0097 0486 9802           BCFR,EQ SNZW      BRANCH IF R1 NONZERO
0098 0488 1B6F           BCTR,UN SIZO      SIZE IS ZERO
0099 048A CD0442    SNZW STRA,R1 PSI       STORE SIZE HIGH INTO PSI
0100 048D CE0443         STRA,R2 PSI+1     STORE SIZE LOW INTO PSI+1
0101 0490 3F02DB         BSTA,UN GNUM      GET NUMBER FROM BUFF IN R1,R2.
0102 0493 CD0440         STRA,R1 PSA       STORE ST ADR HIGH INTO PSA
0103 0496 CE0441         STRA,R2 PSA+1     STORE ST ADR LOW INTO PSA+1
```

```
TWIN ASSEMBLER VER 2.0     *** AUDIO CASSETTE INTERFACE ***   PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0105                 *SYNCHRONIZE FLAG WITH FSK CLOCK
0106                 *
0107 0499 33         LFSH REDC,R3           READ FSK CLOCK
0108 049A 987D            BCFR,Z  LFSH      BRANCH IF FSK IS HIGH
0109 049C 7440            CPSU    F
0110                 *
0111                 *DUMP "ONES" FOR 20 SEC.
0112                 *
0113 049E 33         LFSL REDC,R3           READ FSK CLOCK.
0114 049F 187D            BCTR,Z  LFSL      BRANCH IF FSK CLOCK = 0
0115 04A1 0400            LODI,R0 H'00'
0116 04A3 053C            LODI,R1 H'3C'     SET DELAY=20 SEC.
0117 04A5 7640       NEON PPSU    F
0118 04A7 064B            LODI,R2 H'4B'
0119 04A9 FA7E            BDRR,R2 $         DELAY = 700 USEC.
0120 04AB 7440            CPSU    F
0121 04AD 33         LFSM REDC,R3
0122 04AE 187D            BCTR,Z  LFSM      BRANCH IF FSK CLOCK = 0
0123 04B0 F873            BDRR,R0 NEON
0124 04B2 F971            BDRR,R1 NEON
0125 04B4 0504            LODI,R1 NSB       NUMBER OF START BITS
0126                 *
0127                 *DUMP STARTBITS
0128                 *
0129 04B6 33         LFSN REDC,R3
0130 04B7 187D            BCTR,Z  LFSN      BRANCH IF FSK CLOCK = 0
0131 04B9 0670            LODI,R2 H'70'
0132 04BB FA7E            BDRR,R2 $         DELAY = 1 MSEC
0133 04BD F977            BDRR,R1 LFSN
0134                 *
0135                 *DUMP PROGRAM/DATA
0136                 *
0137 04BF 0D0442          LODA,R1 PSI       LOAD SIZE HIGH INTO R1
0138 04C2 0E0443          LODA,R2 PSI+1     LOAD SIZE LOW INTO R2
0139 04C5 1802            BCTR,Z  PROG      BRANCH IF SIZE = XX00
0140 04C7 8501            ADDI,R1 1
0141 04C9 0C8440     PROG LODA,R0 *PSA      LOAD (FIRST) BYTE INTO R0
0142 04CC 3B20            BSTR,UN RSUB      DUMP (FIRST) BYTE.
0143 04CE 3B0E            BSTR,UN IAC
0144 04D0 FA77            BDRR,R2 PROG
0145 04D2 F975            BDRR,R1 PROG
0146 04D4 3F0560          BSTA,UN DLA3
0147 04D7 0405            LODI,R0 <TXT2     GET ADDRESS HIGH.
0148 04D9 0544            LODI,R1 >TXT2     GET ADDRESS LOW.
0149 04DB 1F0456          BCTA,UN STC
```

```
LINE ADDR  OBJECT  E SOURCE

0151                 *INCREMENT OF ADDRESSCOUNTER.
0152                 *
0153 04DE 0702       IAC  LODI,R3 2
0154 04E0 0F4440     LOPC LODA,R0 PSA,R3,-  FETCH L/U HALF ADR CNTR
0155 04E3 D805            BIRR,R0 NEXT
0156 04E5 CF6440          STRA,R0 PSA,R3    RESTORE L/U HALF ADR CNTR
0157 04E8 5B76            BRNR,R3 LOPC
0158 04EA CF6440     NEXT STRA,R0 PSA,R3    RESTORE L/U HALF ADR CNTR
0159 04ED 17              RETC,UN
0160                 *
0161                 *SUBROUTINE TO OUTPUT ON FLAG FOR TAPEDUMP
0162                 *
0163 04EE 7711       RSUB PPSL     RS+C      SET RS AND C.
0164 04F0 0508            LODI,R1 DB8       BYTE HAS 8 BITS.
0165 04F2 33         LFSK REDC,R3           READ FSK CLOCK INTO R3
0166 04F3 187D            BCTR,Z  LFSK      BRANCH IF FSK CLOCK =0
0167 04F5 50              RRR,R0            PUT (LSB) INTO BIT #7,ETC
0168 04F6 F480            TMI,R0  H'80'     TEST BIT #7
0169 04F8 1A0C            BCTR,N  NONE      IF 1 SET FLAG
0170 04FA 7640            PPSU    F
0171 04FC B501            TPSL    C         TEST CARRY BIT
0172 04FE 1804            BCTR,Z  NON1      BRANCH IF C=1
0173 0500 7701            PPSL    C         SET CARRY BIT
0174 0502 9802            BCFR,Z  NONE      BRANCH IF C NONZERO
0175 0504 7501       NON1 CPSL    C         CLEAR CARRY BIT
0176 0506 0646       NONE LODI,R2 H'46'     SET DELAY
0177 0508 FA7E            BDRR,R2 $         DELAY INTO 2ND HALVE OF FSK CLOCK
0178 050A 7440            CPSU    F
0179 050C B508            TPSL    WC        TEST WITH CARRY BIT
0180 050E 1806            BCTR,Z  BYCO      BRANCH IF BYTE COMPLETE
0181 0510 F960            BDRR,R1 LFSK      IF BYTE NOT COMPL. ROTATE R1 RIGHT
0182 0512 7708            PPSL    WC        SET WC BIT
0183 0514 1B5C            BCTR,UN LFSK
0184 0516 7519       BYCO CPSL    RS+C+WC
0185 0518 17              RETC,UN
```

```
LINE ADDR  OBJECT  E SOURCE

0187                  *SUBROUTINE FOR OUTPUT OF TEXT
0188                  *
0189 0519 CC0444      TEXT STRA,R0 TXTA      STORE ADDRESS HIGH
0190 051C CD0445           STRA,R1 TXTA+1    STORE ADDRESS LOW
0191 051F 0700             LODI,R3 0
0192 0521 0FE444      ATEX LODA,R0 *TXTA,R3  GET CHARACTER
0193 0524 E426             COMI,R0 A'&'
0194 0526 14               RETC,EQ           RETURN IF & CHARACTER
0195 0527 E424             COMI,R0 A'$'
0196 0529 9805             BCFR,EQ BTEX
0197 052B 3F008A           BSTA,UN CRLF      LINE FEED IF $ CHARACTER
0198 052E DB71             BIRR,R3 ATEX
0199 0530 3F02B4      BTEX BSTA,UN COUT      OUTPUT CHARACTER
0200 0533 DB6C             BIRR,R3 ATEX
0201                  *
0202                  *
0203 0535 242A2A41    TXT1 DATA    A'$**ACASIF**$$>&'
     0539 43415349
     053D 462A2A24
     0541 243E26
0204 0544 2A2A454F    TXT2 DATA    A'**EOJ**$$>&'
     0548 4A2A2A24
     054C 243E26
0205 054F 3F3F2424    TXT3 DATA    A'??$$>&'
     0553 3E26
0206 0555 2A2A4552    TXT4 DATA    A'**ERR**$$>&'
     0559 522A2A24
     055D 243E26
0207                  *
0208                  *SET FLAG FOR 3 SEC
0209                  *
0210 0560 7640        DLA3 PPSU    F        SET FLAG
0211 0562 0400             LODI,R0 H'00'
0212 0564 0500             LODI,R1 H'00'
0213 0566 0603             LODI,R2 H'03'    SET DELAY=3 SEC
0214 0568 F87E        LOPA BDRR,R0 LOPA
0215 056A F97C             BDRR,R1 LOPA
0216 056C FA7A             BDRR,R2 LOPA
0217 056E 17               RETC,UN
0218                  *
0219 056F 3F02DB      READ BSTA,UN GNUM      GET NUMBER FROM BUFF IN R1,R2
0220 0572 02               LODZ    R2
0221 0573 9806             BCFR,EQ SNZR      BRANCH IF NONZERO
0222 0575 01               LODZ    R1
0223 0576 9803             BCFR,EQ SNZR      IDEM
0224 0578 1F0479           BCTA,UN SIZO      GET ADDRESS OF TXT3
0225 057B CD0442      SNZR STRA,R1 PSI       STORE SIZE HIGH INTO PSI
0226 057E CE0443           STRA,R2 PSI+1     STORE SIZE LOW INTO PSI+1
0227 0581 3F02DB           BSTA,UN GNUM      GET NUMBER FROM BUFF IN R1,R2
0228 0584 CD0440           STRA,R1 PSA       STORE ST ADR HIGH INTO PSA
0229 0587 CE0441           STRA,R2 PSA+1     STORE ST ADR LOW INTO PSA+1
```

```
LINE ADDR  OBJECT  E SOURCE

                     *SET FLAG FOR 10 SEC
                     *
     058A 3F0560     UNEQ BSTA,UN DLA3
     058D 0403            LODI,R0 H'03'     SET DELAY = 10 SEC
     058F F879            BDRR,R0 UNEQ
     0591 3F05CC     ACAL BSTA,UN FSKC      START ACASIF LOAD
     0594 12              SPSU              LOOK FOR LEADER
     0595 1A04            BCTR,N  ONEL      BRANCH IF S=1
     0597 0600            LODI,R2 H'00'     CLEAR R2
     0599 1B76            BCTR,UN ACAL
     059B FA74       ONEL BDRR,R2 ACAL      BRANCH IF LEADER < 256
     059D 0504            LODI,R1 NSB       NUMBER OF START BITS
     059F 3F05CC     SBNF BSTA,UN FSKC
     05A2 12              SPSU              LOOK FOR START BITS
     05A3 1A7A            BCTR,N  SBNF      BRANCH IF START BITS NOT FOUND
     05A5 F978            BDRR,R1 SBNF
     05A7 7440            CPSU    F         TRIGGERPULSE
     05A9 7640            PPSU    F
                     *
                     *LOAD PROGRAM/DATA
                     *
     05AB 0D0442          LODA,R1 PSI       LOAD SIZE HIGH INTO R1
     05AE 0E0443          LODA,R2 PSI+1     LOAD SIZE LOW INTO R2
     05B1 1802            BCTR,Z  PROD      BRANCH IF SIZE = XX00
     05B3 8501            ADDI,R1 1
     05B5 3B27       PROD BSTR,UN BITM
     05B7 01              LODZ    R1
     05B8 7510            CPSL    RS        CLEAR RS BIT
     05BA CC8440          STRA,R0 *PSA      STORE (FIRST) BYTE INTO (ST) ADR
     05BD 3F04DE          BSTA,UN IAC
     05C0 FA73            BDRR,R2 PROD
     05C2 F971            BDRR,R1 PROD
     05C4 0405            LODI,R0 <TXT2     GET ADDRESS HIGH
     05C6 0544            LODI,R1 >TXT2     GET ADDRESS LOW
     05C8 1F0456          BCTA,UN STC
```

```
LINE ADDR  OBJECT  E SOURCE

0267                 *SUBROUTINE TO START DELAY IF FSK GOES HIGH.
0268                 *
0269 05CB 51         FHSD RRR,R1             ROTATE R1 RIGHT
0270 05CC 30         FSKC REDC,R0            READ FSK CLOCK
0271 05CD 9804            BCFR,Z  TOVF       BRANCH IF SENSE IS NONZERO.
0272 05CF 7704            PPSL    OVF        SET OVF BIT
0273 05D1 1B79            BCTR,UN FSKC
0274 05D3 B504       TOVF TPSL    OVF        TEST OVF BIT
0275 05D5 9875            BCFR,Z  FSKC       BRANCH IF OVF IS NONZERO
0276 05D7 7504            CPSL    OVF        CLEAR OVF BIT
0277 05D9 0439            LODI,R0 H'39'      SET DELAY=570 USEC
0278 05DB F87E            BDRR,R0 $
0279 05DD 17              RETC,UN
0280                 *
0281                 *SUBROUTINE TO RECOVER ONE BYTE FROM TAPE
0282                 *
0283 05DE 7710       BITM PPSL    RS         SET REGISTER SELECT
0284 05E0 0500            LODI,R1 H'00'      CLEAR R1
0285 05E2 0608            LODI,R2 D'8        BYTE HAS 8 BITS
0286 05E4 0780            LODI,R3 OF         ODD PARITY
0287 05E6 3B63            BSTR,UN FHSD
0288 05E8 12         LODB SPSU               SENSE BIT IS LOADED INTO BIT#7 OF R0
0289 05E9 1A02            BCTR,N  ONEC       BRANCH IF SENSE IS 1
0290 05EB 1B02            BCTR,UN FORG
0291 05ED 6580       ONEC IORI,R1 BP8        INSERT DATABIT INTO R1
0292 05EF A601       FORG SUBI,R2 H'01'      KEEP TRACK OF NUNBER OF BITS
0293 05F1 23              EORZ    R3
0294 05F2 C3              STRZ    R3         DO PARITY CHECK.
0295 05F3 7A56            BSNR,R2 FHSD
0296 05F5 5A71            BRNR,R2 LODB
0297 05F7 3B53            BSTR,UN FSKC
0298 05F9 12              SPSU               PARITY BIT INTO BIT #7 OF R0
0299 05FA 23              EORZ    R3
0300 05FB C3              STRZ    R3         DO PARITY CHECK.
0301 05FC 9A09            BCFR,N  SGN1       BRANCH IF PARITY NOT WRONG
0302 05FE 7510            CPSL    RS         CLEAR REGISTER SELECT
0303 0600 0405            LODI,R0 <TXT4      GET ADDRESS HIGH
0304 0602 0555            LODI,R1 >TXT4      GET ADDRESS LOW
0305 0604 1F0456          BCTA,UN STC
0306 0607 17         SGN1 RETC,UN
0307 0450                 END     NIIT

 TOTAL ASSEMBLY ERRORS = 0000
```

Printed in The Netherlands    Date of release: 16 June 1978    T.D.    9399 500 08701