TWONKY

May Hadley has designed an MPU music box that plays random tunes to the rules laid down by a compositional algorithm.

EVER SINCE THE computer was invented, whenever that was, there have been people who have sought to apply it in previously untouched fields. Doubtless the same will happen with the microprocessor to a much greater extent because of its vastly lower cost and wider circle of users. Certainly the amateur constructor can do far more than simply make miniature computers. Twonky is one such application in the field of computer music.

Macro Music

Music was first applied to computers in the late '50s. Machines of that vintage were often fitted with loudspeakers monitoring a register or address bit, to aid in software and hardware fault tracing. Cunning programmers soon realised how to make such computers play tunes when no-one was around to stop them, and so computer music was born. It grew rapidly.

One of its earliest exponents was Professor Lejaren Hiller of Illinois University who together with his colleague Prof. Leonard M. Isaacson conducted a series of studies which are described in their book 'Experimental Music' (McGraw Hill 1959). They began by using the computer to test the classical compositional rules of species counterpoint, developed in the seventeenth century by J. J. Fux and taught to music students ever since. A program was written which would generate random notes, test them against the rules and insert them where a suitable match was found. Though this sounds simple enough, it took several years to do, as the 'rules' were by no means complete: many things were assumed as being obvious by the musical theorists which had to be explicitly stated for the computer.

Suite Illiac

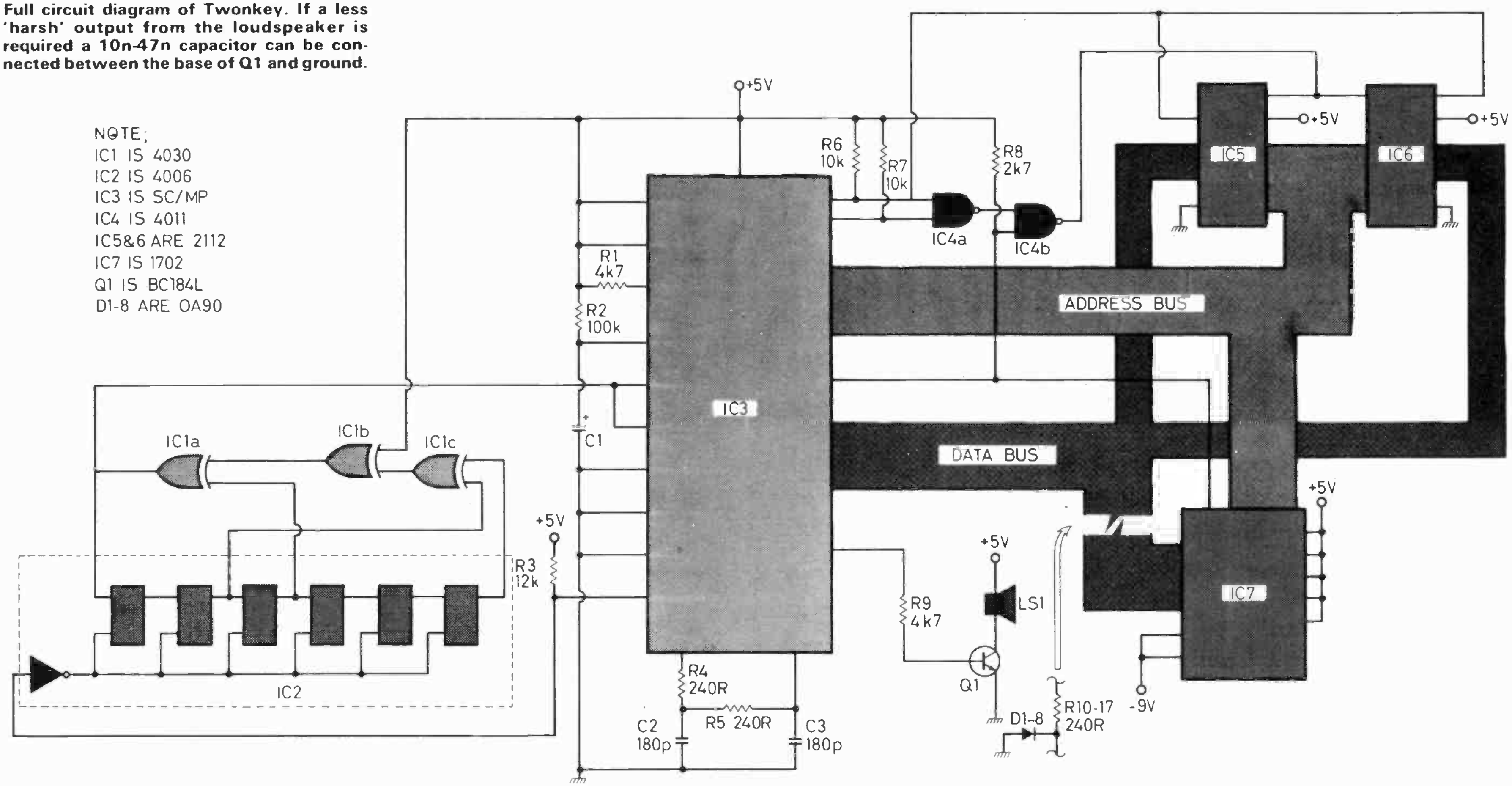
By this time, the original aim, which was to test the compositional rules in question, had become secondary to the fun of using the computer to generate new music.

Other styles and principles, ranging from the sixteenth to the twentieth centuries, were applied in something of a mixture, and the result served up as the 'Illiac Suite for String Quartet' (named after the famous ILLIAC IV computer on which they were composed.) This proved rather disappointing, sounding almost a parody of twentieth century chamber music.

Other workers, such as Professor J. K. Randall of Princeton University, developed slightly different lines of approach, including the one used by Twonky. Prof. Randall's work 'Prelude to Mudgett' may be heard on disc (Nonesuch 71245), and is a typical example of this style and approach.

While this effort was going into composition and stylistic analysis, ▶

Full circuit diagram of Twonkey. If a less 'harsh' output from the loudspeaker is required a 10n-47n capacitor can be connected between the base of Q1 and ground.



HOW IT WORKS - HARDWARE

The National Semiconductor SC/MP is a simple, cheap, 8-bit processor designed for use in minimal systems; to this end it has an on-chip clock generator and I/O facilities, and needs no bus buffers in small systems. The instruction set is not large, but contains such useful features as a wide range of addressing modes and the capability of double-indexed memory references.

Internally, the chip has seven main registers; an 8-bit accumulator, an 8-bit status register, four 16-bit pointer or index registers (one of which is dedicated as the program counter), an 8-bit extension register. All memory references (including jumps) are via an index register; the second byte of each memory reference instruction is a displacement which is added to the index register and NWDS, and NRDS high, to prevent spurious memory enables while the MPU outputs are in the high impedance mode between memory accesses.

Components R5, R8, C2, and C3 set the processor clock frequency at about 4MHz. R5 can be made variable to act as a tuning control, but must be between 100 ohms and 2kilohms. The MPU is reset on power-up by R3 and C1, and the first instruction is fetched from location OOIH.

IC6 and IC7 form a PRBS generator. An 18-stage shift register, clocked by the NADS strobe from the MPU has exclusive OR feedback arranged around it such that it will produce a stream of bits in a repeating sequence 2^{18}-1 (262,143) bits long. Within this overall sequence, the bit stream is random,

other people were engaged in turning the computer into a new musical instrument, a 'super synthesiser' (although this work was begun before Dr Moog invented the voltage controlled analogue synthesiser). Several programs have been developed; TEMPO by Glough and Sosman on an IBM 360/44, MUSIGOL at the University of Virginia, and the most widely used, MUSIC 4 (and its derivatives MUSIC's 4B, 4BF, and 5) at Bell labs and Princeton.

This is a program, mainly in FORTRAN IV but with some assembly language sections, which play tunes monophonically using squarewaves. The compositional algorithm (due to Prof. Randall) is based on two simply observations:-

1. Every tune has at least one highest note
2. Every tune can be split into two subtunes at least one not long, which may then themselves be regarded as tunes.

To compose a tune using these rules, we assume also that each tune only has one highest note, and that each subtune is half the length of the tune. We take a given note as the highest note in the whole tune and may be in the range −127 to 127. If the displacement has the value −128 the contents of the extension register are used as the displacement to be added (doubly indexed memory reference). There is, however, no explicit subroutine call instruction.

The status register contains carry, overflow, and interrupt enable flags, two sense input bits, and three user definable flags. These last five are taken to package pins to provide limited I/O capability. Twonky uses the sense-B input to read random bits from the pseudo-random binary sequence generator (PRBS generator) and the flags to drive the audio amplifier. Out of the total of 46 instructions, only 17 are used and these are shown in fig. 1 along with the status register bit allocation.

The program itself is 256 bytes long and lives in a 1702A EPROM at addresses OOOH to OFFH. 256 bytes of RAM in the shape of 2, 2112-A4 256 x 4 bit static chips are provided at addresses 1OOH to 1FFH. Address lines AO to A7 are common to IC2, 3, and 4, while A8 is taken to the CE input of IC2 to enable it at the correct range of addresses. Note that IC2 will be enabled by any memory access, read or write, in the correct address range. If a faulty program goes berserk and tries to write to ROM, two devices will be enabled onto the data bus at the same time. This might be fatal, were it not for R9-R16 which prevent a short circuit. Additionally, in conjunction with D1-D8, they prevent negative voltages from the PMOS ROM appearing at the inputs of the other, NMOS, devices on the bus.

The RAM is enabled by the signal from pin 11 of IC5, which will be low (RAM enabled) when A8 is high and either NWDS (not write data strobe) or NRDS (not read data strobe) is low. The resistors R7, 8, and 18 tie A8, i.e. the probability of the next bit at any point in the sequence being a one is constant at O.5. This random sequence is fed to sense-B on the MPU and is used by the software to provide the random element in each tune.

Also, since the sense -B input is not sampled by the MPU internal logic during the NADS strobe time, the random bit will always be read unambiguously.

The power to IC6 and IC7 is not switched; they are CMOS devices which when not being clocked draw only about a microamp. This is necessary, as should the shift register be in the all zero state on switch on, the generator will stick and produce a continuous stream of zeros. Logic could be incorporated to force ones into the register on switch-on, but unless it was very devious, would result in the dame sequence of pseudo-random bits (and hence tunes) occuring every time.

The audio output is taken from the MPU flag O output and amplified by Q1 to drive the speaker. A line level output may also be taken from flag 1 or 2 if desired.

There are two types of SC/MP processor available; this circuit uses the NMOS variety, which is cheaper, faster, uses less power and needs only +5V and ground. The older PMOS type can be used, but not all the control signals are the same, and so the circuitry around IC5 will need to be altered. The pitch will also be about an octave lower. Owners of SC/MP development systems, such as the introkit, MK14, or Scrumpi will be able to hook up a PRBS generator and loudspeaker to their systems with little trouble, and to relocate the code as appropriate. For further details on the SC/MP chip the data sheet, Nat. Semi pub. No. 426305290-001B (!) may be consulted.

0000	08	04	01	36	04	FE	32	C4	01	CA	00	C4	00	36	C4	00
0010	32	C6	01	C4	80	31	C4	80	01	C4	01	35	C1	00	C9	80
0020	01	F4	02	02	01	C9	80	06	D4	20	98	0A	C1	80	F4	01
0030	02	C9	80	01	90	10	01	F4	FE	02	01	C1	80	F4	01	02
0040	C9	80	01	F4	02	02	98	05	01	C5	02	90	CF	C4	00	32
0050	F4	F9	02	98	06	F4	07	02	32	90	B6	C4	EF	32	35	C4
0060	01	37	C4	FF	33	06	D4	20	9C	06	C4	97	31	3D	90	F5
0070	06	D4	20	9C	06	C4	9B	31	3D	90	08	C4	9F	31	3D	C4
0080	9F	31	3D	06	D4	20	9C	06	C4	9B	31	3D	90	D7	C4	9F
0090	31	3D	C4	9F	31	3D	90	CD	C4	01	90	06	C4	02	90	02
00A0	C4	03	35	C7	FF	01	C2	80	CB	00	01	F4	08	02	01	C2
00B0	80	01	C4	00	35	19	F4	FF	02	9C	FA	01	CB	01	33	98
00C0	04	33	C7	FF	3D	33	C3	01	31	C4	07	07	C4	0F	8F	00
00D0	C3	00	F4	FF	02	9C	FB	C4	00	07	C3	00	F4	FF	02	9C
00E0	FB	31	F4	FF	02	9C	E1	8F	3A	C7	02	33	9C	D7	35	3D
00F0	32	35	3C	44	48	51	5B	67	FE	F0	D6	BD	B3	A0	8F	7F

Hex dump of the PROM program for the Twonky composer

generates musical sounds as a series of digital samples which are fed to a D/A converter, usually via the intermediate medium of magtape. Sounds are described in terms of instruments, which are routine that use stored tables of sinewaves, exponentials, ramps and other waveforms to generate complex sound sources. These are coupled via filter, reverberations, stereo position and other modules into an 'orchestra,' which outputs the final sound onto tape. The music to be played is input in the form of note cards. These punched cards carry such details as pitch, rate of rise and fall of the envelopes, start time, and other, user-defined parameters.

One Hundred 'seconds

In the early days, it took as much as 50 to 100 seconds of computer time to generate a second of music, but with modern machines, synthesis can take place in real time or faster. The program is not, however, suitable for live performance use. The result of such programs can be most impressive, particularly in the hands of a skilled 'player.' Certainly, they are far more flexible and versatile than analogue synthesisers. They have the particular merit that if, for example, 96 oscillators are needed, the function OSCIL is merely called 96 times. This uses more processor time, but does not need any additional hardware.

MUSIC 4B, together with analogue sound synthesisers, is described in Hubert S. Howe's book 'Electronic Music Synthesis.' The field of digital sound synthesis is certainly an exciting one, but is somewhat beyond the reach of the amateur, although with powerful 16 bit machines such as the LSI 11 and TMS9900 becoming cheaper, it may not remain so for long.

A Little Micro Music

Twonky is a composing machine which also incorporates software to assign it randomly to one or other of the subtunes. The highest note in the other subtune must be lower than that in the first: we assume it is the next note down whatever scale we are using. However, each subtune may now itself be regarded as a new tune, provided it is at least two notes long. Hence in each first-level subtune, we take the highest note and assign it randomly to one or other of the second-level subtunes, adding the next lowest note in our scale as the highest note in the other. By repeating the process, we double the number of known notes in our tune (each of which is the highest note of some subtune) and increase the number of pitches by one for each level of splitting we indulge in. This process can hence be described as a random tree.

Seventh Level

In Twonky, seven levels of division are used to generate 128 subtunes each one note long, with a total range of 8 pitches (one octave of the scale of C major). The random decision at each level is produced by a hardware random number generator.

The rhythmic element in each tune is produced by selecting one of a small number of rhythm units or bars on a random basis and fitting the notes of the tune to that bar. The melodic algorithm weights the distribution of notes binorally, thus there are 2 F s (one of each octave), 7 G s 21 A s, 35 B s, 35 C s, 21 D s, and 7 E s. The tonic or key-note C occurs most frequently, lending a definite key to the melody. However, it is usual for the dominant G also to occur frequently, which it does not do. This gives all Twonky's compositions a unique and unusual style, somewhat like Mediaeval music (nothing to do with the use of a SC/MP MPU) this is enhanced by the ready tone of the square wave output.

PROGRAM LISTING

```
00  08              NOP              Dummy instruction — not executed
01  C4  01  START   LDI   1          Store 1 in location 510
03  36              XPAH  PTR 1
04  C4  FE          LDI   FEH
06  32              XPAL  PTR 2
07  C4  01          LDI   1
09  OA  00          STO   PTR 2+0
0B  C4  00          LDI   0          Clear cycle counter (No. of levels down
0D  36              XPAH  PTR 2      decision tree)
0E  C4  00          LDI   0          Each location in top half of memory is
10  32              XPAL  PTR 2      written to two locations starting at
                                     bottom of RAM with random incre-
                                     ment. Repeated 7 times
11  C6  01  OUTLOOP LD    PTR 2+1    Increment cycle counter
13  C4  80          LDI   80H        Set PTR 1=180H=384 (bottom of top
15  31              XPAL  PTR 1      half of RAM)
16  C4  80          LDI   —128       Extension register = —128
18  01              XAE              PTR 1 points to location being read, and
19  C4  01          LDI   1          is stepped upwards. EXT contains
1B  35              XPAH  PTR 1      displacemtn to location being written
1C  C1  00  INLOOP  LD    PTR 1+0
1E  C9  80          STO   PTR 1-128  Take contents of (PTR 1) and store 1 in
20  01              XAE              (PTR 1 + EXT) and in (PTR 1 + EXT + 2)
21  F4  02          ADI   2
23  02              CCL
24  01              XAE
25  C9  80          STO   PTR 1-128
27  06              CSA              Input random bit and either:
28  D4  20          ANI   00100000B=32
2A  98  0A          JZ    INCLOW
2C  C1  80          LD    PTR 1-128  Increment (PTR 1 + EXT + 2) or:
2E  F4  01          ADI   1
30  02              CCL
31  C9  80          STO   PTR 1-128
33  01              XAE
34  90  10          JMP   EXTEST
36  01      INCLOW  XAE              Increment (PTR 1 + EXT)
37  F4  FE          ADI   —2
39  02              CCL
3A  01              XAE
3B  C1  80          LD    PTR 1-128
3D  F4  01          ADI   1
3F  02              CCL
40  C9  80          STO   PTR 1-128
42  01              XAE
43  F4  02          ADI   2          Add 2 to EXT for next pass
45  02              CCL
46  98  05  EXTEST  JZ    PTEST      If EXT = 0 go to PTEST; 1 pass through
48  01              XAE              memory completed, else add 2 to PTR 1
49  C5  02          LD@   PTR 1+2    and loop back to INLOOP
4B  90  CF          JMP   INLOOP
4D  C4  00  PTEST   LDI   0          If cycle counter (PTR 2) = 7 then go to
4F  32              XPAL  PTR 2      note (PTR 2 = 0)
50  F4  F9          ADI   —7
52  02              CCL
53  98  06          JZ    NOTE
55  F4  07          ADI   7          else go to OUTLOOP
57  02              CCL
58  32              XPAL  PTR 2
59  90  B6          JMP   OUTLOOP
5B  C4  EF  NOTE    LDI   EFH        PTR 2 points to pitch table
5D  32              XPAL  PTR 2      (PTR was zero)
5E  35              XPAH  PTR 1      Clear high byte of PTR 1 (return address
```

```
83  06      SECPART CSA              Input random bit and either:
84  D4  20          ANI   00100000B
86  9C  06          JNZ   RHYTH3
88  C4  9B          LDI   9BH        Write middle sized note by subroutine
8A  31              XPAL  PTR 1      at 9BH, on return go to NXNOTE
8B  3D              XPPC  PTR 1
8C  90  D7          JMP   NXNOTE
8E  C4  9F  RHYTH3  LDI   9FH        or write 2 short notes by 2 calls to
90  31              XPAL  PTR 1      subroutine at 9FH
91  3D              XPPC  PTR 1
92  C4  9F          LDI   9FH
94  31              XPAL  PTR 1
95  3D              XPPC  PTR 1      Go to NXNOTE for next note from RAM
96  90  CD          JMP   NXNOTE
98  C4  01  WRNOTE  LDI   1          Subroutine WRNOTE. Entry point de-
9A  90  06          JMP   GO         termines note length divisor. This is
9C  C4  02          LDI   2          loaded to high byte of pointer con-
9E  90  02          JMP   GO         taining return address (as this is always
                                     zero)
A0  C4  03          LDI   3
A2  35      GO      XPAH  PTR 1      Load pitch code from RAM, use as index
A3  C7  FF          LD@   PTR 3-1    for table of pitches. Replace code with
A5  01              XAE              actual pitch from table. Pitch code stays
A6  C2  80          LD    PTR 2-128  in extension register
A8  CB  00          STO   PTR 3+0
AA  01              XAE              Add 8 to pitch code and use as index to
AB  F4  08          ADI   8          duration table (contains number of
AD  02              CCL              cycles for 0.362 secs at each pitch)
AE  01              XAE
AF  C2  80          LD    PTR 2-128
B1  01              XAE              Note length to EXT ready for division
B2  C4  00          LDI   0          divisor is fetched from pointer 1 high
B4  35              XPAH  PTR 1      byte which is zeroed at the same time
B5  19      S101    S10
B6  F4  FF          ADI   —1         Number of cycles divided by 2, 4, or 8
B8  02              CCL              depending on subroutine entry point to
B9  9C  FA          JNZ   S101       give the 3 different note lengths
BB  01              XAE
BC  CB  01          STO   PTR 3+1    Note length is stored in RAM location
BE  33              XPAL  PTR 3      immediately above corresponding pitch
BF  98  04          JZ    PLAY       if end of RAM (100H) has been reached,
C1  33              XPAL  PTR 3      go to play else decrement PTR and
C2  C7  FF          LD@   PTR 3-1    return
C4  3D              XPPC  PTR 1
C5  33      PLAY    XPAL  PTR 3      Load note length (PTR points to low
C6  C3  01          LD    PTR 3+1    byte of current note) and store in PTR 1
C8  31      POS     XPAL  PTR 1
C9  C4  07          LDI   00000111B  Load bit pattern to set flag bits in stat
CB  07              CAS              reg. Output high delay to make half
CC  C4  OF          LDI   15         cycles of audio waveform same length
CE  8F  00          DLY   0
D0  C3  00          LD    PTR 3+0    Load pitch from RAM (number of times
D2  F4  FF  ADI1    ADI   —1         round this delay loop for a half cycle at
D4  02              CCL              desired frequency)
D5  9C  FB          JNZ   ADI1
D7  C4  00          LDI   00000000B  Clear flag bits
D9  07              CAS              Takes audio output low
DA  C3  00          LD    PTR 3+0    Load pitch from RAM
DC  F4  FF  ADI2    ADI   —1         Delay loop for negative half cycle
DE  02              CCL
DF  9C  FB          JNZ   AD12       Decrement cycle counter. If not zero, go
E1  31              XPAL  PTR 1      to POS for another cycle
E2  F4  FF          ADI   —1
E4  02              CCL
```

```
5F  C4  01            LDI    1            will be stored here)
61  37                XPAH   PTR 3
62  C4  FF            LDI    FFH          Set PTR 3 = 511 = 1FFH · top of RAM)
64  33                XPAL   PTR 3
65  06       NXNOTE   CSA                 Start of note length writing loop
66  D4  20            ANI    00100000B    input random bit and either:
68  9C  06            JNZ    RHYTH1
6A  C4  97            LDI    97H          Write long note by subroutine at 97H,
6C  31                XPAL   PTR 1        on return, jump back to NXNOTE
6D  3D                XPPC   PTR 1
6E  90  F5            JMP    NXNOTE
70  06       RHYTH1   CSA                 or input random bit and either:
71  D4  20            ANI    00100000B
73  9C  06            JNZ    RHYTH2       Write middle sized note by subroutine
75  C4  9B            LDI    9BH          at 9BH, on return go to SECPART
77  31                XPAL   PTR 1
78  3D                XPPC   PTR 1
79  90  08            JMP    SECPART
7B  C4  9F   RHYTH2   LDI    9FH          or write 2 short notes by 2 calls to
7D  31                XPAL   PTR 1        subroutine at 9FH
7E  3D                XPPC   PTR 1
7F  C4  9F            LDI    9FH
81  31                XPAL   PTR 1
82  3D                XPPC   PTR 1
```

```
E5  9C  E1   JNZ    POS          Inter note gap
E7  8F  3A   DLY    58           Move note counter to next note
E9  C7  02   LD@    PTR 3+2      If PTR 3/200H go to play next note)
EB  33       XPAL   PTR 3
EC  9C  D7   JNZ    PLAY
EE  35       XPAH   PTR 1        Else go back to start for another tune!
EF  3D       XPPC   PTR 1
F0  32       DEFB   50           F   350.875 HZ    Pitches at 4MHZ
F1  35       DEFB   53           E   332.005 HZ
F2  3C       DEFB   60           D   294.985 HZ
F3  44       DEFB   68           C   261.645s HZ
F4  48       DEFB   72           B   247.645 HZ
F5  51       DEFB   81           A   221.045 HZ
F6  5B       DEFB   91           G   197.47  HZ
F7  67       DEFB   103          F   175.07  HZ

F8  FE       DEFB   254          0.362 SECS    Length of long note at
F9  F0       DEFB   240          0.361 SECS
FA  D6       DEFB   214          0.363 SECS
FB  BD       DEFB   189          0.361 SECS
FC  B3       DEFB   179          0.361 SECS
FD  A0       DEFB   160          0.362 SECS
FE  8F       DEFB   143          0.362 SECS
FF  71       DEFB   127          0.363 SECS
```

HOW IT WORKS - SOFTWARE

The programme itself falls naturally into four parts, which are shown in the four flowcharts. Of these, three (START, NOTE and WRNOTE) write the tune, and one (PLAY) plays it. Before describing the operation of each in more detail, a couple of notes are relevant.

— enclosing an expression in brackets turns it from a number into an address. Thus 510 is a number, but (510) means 'the contents of location 510'.

— all variables used in the flowcharts are actual machine registers except for the dummy variables A and B in WRNOTE. Of these, B is introduced only to improve readibility, while A is an argument passed to this subroutine from the main program. It is implemented in object code by calls to 3 different addresses for its 3 possible values.

START

This is the program section which implements the random decision tree to select the pitches used in the tune. In this section the notes are numbered from 1 to 8 (highest to lowest). The code for each note consists of two bytes, one for pitch and one for duration, which occupy consecutive locations. Pitches are always in even-numbered locations.

On reset, a 1 is written to the last note pitch location (510), and the loop counter PTR 2 is reset. The program then enters a loop in which each note in the top half of RAM, starting at the bottom and going up, is written to two successive note locations, starting at the bottom. of RAM and going up. The writing address catches up with the read address at location 510, which is written to 508 and back into 510. At each step one or other of the two locations is incremented, depending on the state of the random number generator.

Thus after one complete pass through this loop, our tune, which started out as one note — a one — in location 510, is now twice as long and has two notes, a one and a two, randomly arranged in locations 508 and 510. So far so good. We now repeat this loop a total of seven times, each time doubling the number of notes written, until the memory is full (128 notes). We will then have 8 different note numbers or pitches. In fact, what we have done is identical to the decision tree method in the text (try it yourself with pencil and paper).

This section occupies addresses 00H to 60H. PTR 2 is the loop counter which goes from one to seven. On reaching seven, the program branches to NOTE. Within the section, PTR 1 points to the location being read, and EXT contains the displacement from this address to that of the location being written into.

SC/MP fanatics may notice that a separate read-increment-write instruction sequence is used (at 2CH to 33H and at 3BH to 41H) instead of the increment and load single instruction. This is because the ILD instruction does not allow doubly-indexed addressing to be used. This is not made very clear in the databook, and had to be found out the hard way!

NOTE

NOTE is the program section concerned with writing the rhythm of the tune. It has three different note lengths to play with, of relative values 4, 2, and 1. Each bar or rhythm unit can be one of 4, 2 + 2, 2 + 1 + 1, 1 + 1 1 + 2, or 1 + 1 + 1 + 1, determined by random decisions. The flowchart for this section is more or less self explanatory. The notes of different lenghts are written by calls to the subroutine WRNOTE. This has three different entry points (98H, 9CG, AOH) which determine the length of note (long, medium or short). There is no explicit test for leaving the loop in this section as this is done in WRNOTE.

WR NOTE

On being called, this section reads the value of pitch code from RAM (starting at location 510 and going downwards) and uses it as an index to the table of pitches at locations FOH to F7H. The pitch obtained from this table is then stored in the same RAM location from which was read its code. Thus 3 will be replaced by 3CH, 8 by 67H etc. These pitches represent the length of a half cycle at the desired frequency in multiples of the time taken to go round the delay loops in PLAY.

By adding 8 to the pitch code the table of durations (F8H to FFH) is accessed in the same way. The duration is then divided by 2, 4 or 8 to give the required note length in terms of a number of cycles at its particular frequency. This number is then stored in the RAM location immediately above its corresponding pitch. WRNOTE then tests for the last note in the tune (PTR 3 = 255); if the last note has not been reached, control is returned tio NOTE, otherwise control passes to PLAY.

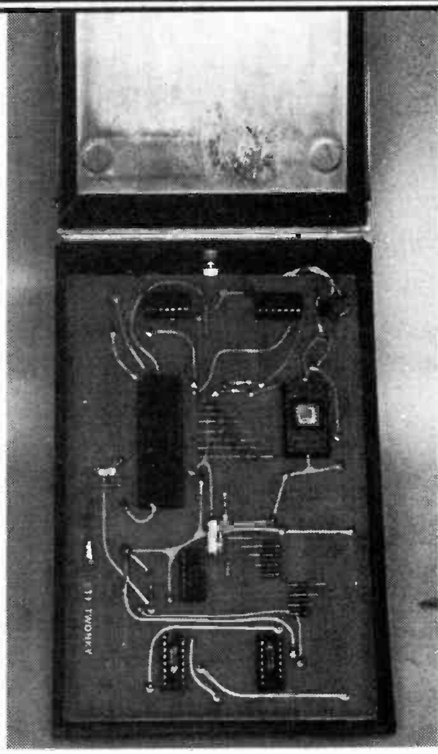
PLAY

This section is quite simple, consisting of two delay loops for pitch, and counters for duration and number of notes played. For each note in turn, the duration is first loaded to PTR 1. The pitch is loaded to the accumulator and the output taken high. The output remains high while the accumulator is decremented and tested for equality to zero. This gives a delay dependant on the initial pitch value. When zero is reached, the output is taken low and the pitch again loaded and decremented to zero. At the end of the second half cycle PTR 1, the duration counter, is decremented and tested for quality to zero. If not zero, another cycle of the same note is produced, otherwise the next note is played, after the end of tune test (PTR 3 = 512). When the end of the tune is reached, control returns to START to write and play a new tune.

Construction is quite straightforward. Sockets should be used for all IC's and normal MOS handling precautions taken. Begin by installing all through board links and testing them for continuity. Then add the resistors, capacitors, and discrete semiconductors. IC 5 may be fitted and the memory decoding checked. IC 6 & 7 should be added next, and the production of random bits at IC 7 pin 6 as pin 3 is clocked by shorting it to ground verified. Finally, add the LSI chips and switch on. Music should greet your ears within about 0.25 secs. Gaps of about this length occur every 128 notes as a new tune is written. The circuit meets all timing requirements with the 1702A only up to 3.5 MHz. Most 1702As will work happily at 4MHz, but the odd one may not. Reducing the clock frequency should effect a cure.

The PCB is single Eurocard size (100 × 160 mm) and will fit in one of the larger size veroboxes, which are designed for this standard. Batteries, either 4 × 1.5V + 1 × 9V dry cells or the equivalent nicads, will then fit under the circuit board, or the PCB may be left uncased. The only major problem which may arise is getting the EPROM programmed. Several firms offering such a service advertise on the pages of ETI and one of these should be able to help.

ETI



Photograph showing Twonky mounted in the larger sized Vero flip top case. The speaker and batteries are mounted under the PCB. The case is not very deep and a 'shallow' speaker must be used if Twonky is to be built in this case.

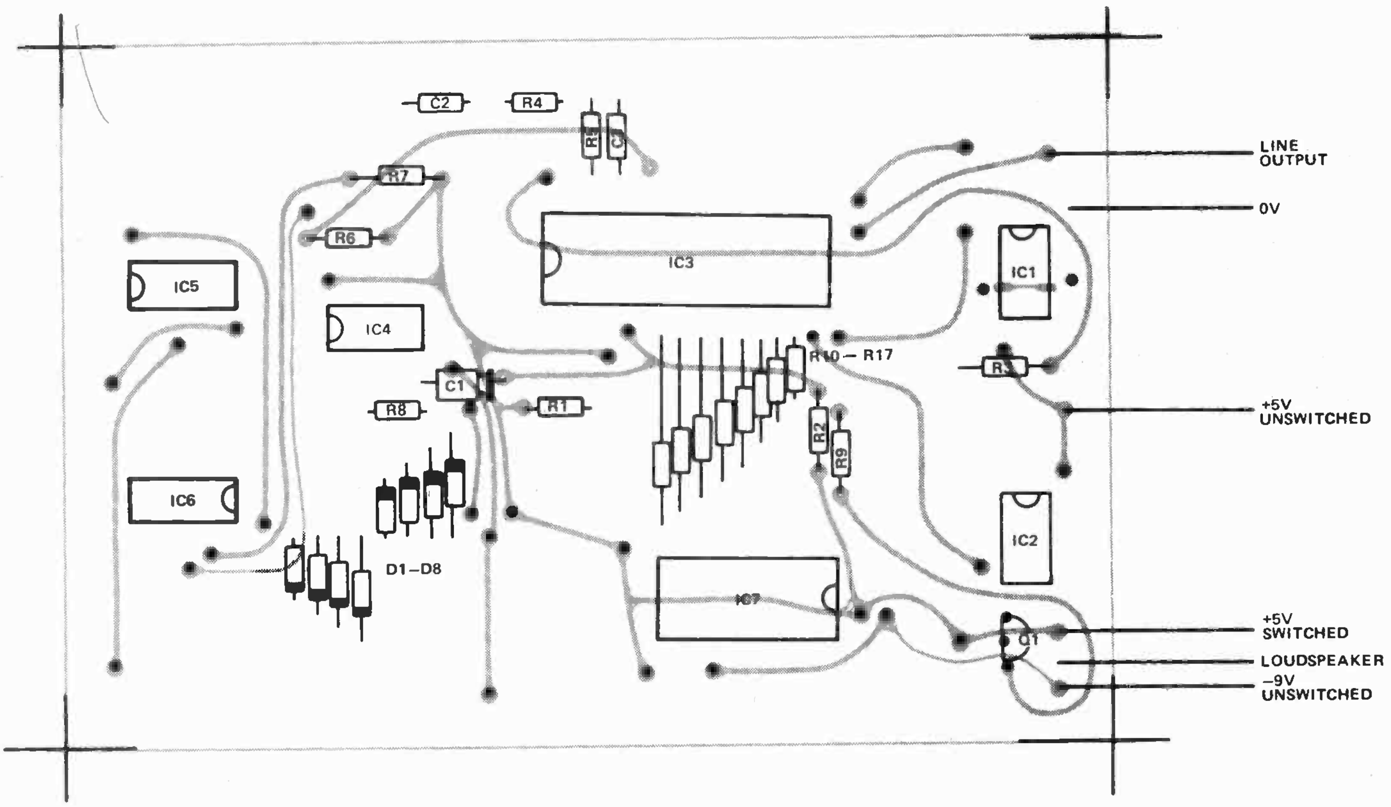
BUYLINES

Marshalls, see their advert in this issue for addresses, will be supplying an EPROM with the Twonky program burned in. They will also be able to supply all the other parts for this project except the PCB which will be available from Tamtronix, Ramar, Crofton etc.

PARTS LIST

RESISTORS	all ½W, 5%
R1, 9	4k7
R2	100k
R3	12k
R4, 5, 10-17	240R
R6, 7	10k
R8	2k7
CAPACITORS	
C1	4u7 16V electrolytic
C2,3	180p 16V ceramic
SEMICONDUCTORS	
IC1	4030
IC2	4006
IC3	SC/MP
IC4	4011
IC5, 6	2112
IC7	1702
Q1	BC184L
D1-8	OA90
MISCELLANEOUS	

PCB, loudspeaker, case batteries and clips.



Component overlay for the ETI Twonky. The wire link that is visible on the photo or the prototype's PCB has been replaced with a foil track.