```
1E2A:8E F0 19     97 PLOT:  STX   FLAG        ;STORE ACTION FLAG
1E2D:48           98        PHA               ;SAVE ROW TEMPORARILY
1E2E:98           99        TYA               ;MOVE COLUMN TO A-REG
1E2F:A2 00       100        LDX   #$00        ;PRESET PIXEL MASK INDEX
1E31:4A          101        LSRA              ;DIVIDE COLUMN BY 2
1E32:8D EF 19    102        STA   COL         ;SAVE TRUE COLUMN FOR LATER
1E35:90 01       103        BCC   CEVEN       ;BRANCH IF COLUMN WAS EVEN
1E37:E8          104        INX               ;OTHERWISE BUMP MASK INDEX
1E38:4A          105 CEVEN: LSRA              ;DIVIDE COLUMN BY 2 AGAIN
1E39:8D F2 19    106        STA   CINDX       ;SAVE  MAP COLUMN INDEX
1E3C:B0 04       107        BCS   RNIBL       ;BRANCH IF CHAR IN RIGHT NYBBLE
1E3E:E8          108        INX               ;ELSE ADD
1E3F:E8          109        INX               ; 4 TO
1E40:E8          110        INX               ;  COLUMN
1E41:E8          111        INX               ;    MASK INDEX
1E42:68          112 RNIBL: PLA               ;RETRIEVE ROW
1E43:4A          113        LSRA              ;DIVIDE ROW BY 2
1E44:8D EE 19    114        STA   ROW         ;SAVE AS TRUE ROW
1E47:90 02       115        BCC   REVEN       ;BRANCH IF ROW WAS EVEN
1E49:E8          116        INX               ;ELSE BUMP MASK
1E4A:E8          117        INX               ; INDEX BY 2
1E4B:0A          118 REVEN: ASLA              ;MULTIPLY TRUE ROW BY 2
1E4C:A8          119        TAY               ; TO USE AS ROW TABLE INDEX
1E4D:B9 FB 19    120        LDA   RTAB,Y      ;GET ROW POINTER
1E50:85 EE       121        STA   RPTR        ; FROM ROW TABLE
1E52:B9 FC 19    122        LDA   RTAB+1,Y    ;  AND STORE IN
1E55:85 EF       123        STA   RPTR+1      ;    PAGE ZERO
1E57:AC F2 19    124        LDY   CINDX       ;RETRIEVE MAP COLUMN INDEX
1E5A:B1 EE       125        LDA   (RPTR),Y    ;GET SCREEN MAP BYTE
1E5C:2C F0 19    126        BIT   FLAG        ;TEST ACTION FLAG
1E5F:10 0D       127        BPL   SETPX       ;BRANCH IF NOT 'TEST'
1E61:A0 00       128        LDY   #$00        ;CLEAR Y-REG FOR RETURN
1E63:3D F3 19    129        AND   MASK,X      ;TEST PIXEL WITH MASK
1E66:F0 01       130        BEQ   PXOFF       ;BRANCH IF PIXEL 'OFF'
1E68:C8          131        INY               ;BUMP Y IF PIXEL 'ON'
1E69:A9 00       132 PXOFF: LDA   #$00        ;SET RETURNED A TO ZERO
1E6B:4C 4C D1    133        JMP   BSRET       ;RETURN VALUE TO BASIC
1E6E:1D F3 19    134 SETPX: ORA MASK,X        ;FORCE PIXEL 'ON'
1E71:70 03       135        BVS   RSTRB       ;BRANCH IF WE DID IT RIGHT
1E73:5D F3 19    136        EOR   MASK,X      ;OTHERWISE TURN PIXEL 'OFF'
1E76:91 EE       137 RSTRB: STA   (RPTR),Y    ;RESTORE SCREEN MAP BYTE
1E78:E0 04       138        CPX   #$04        ;CHECK NYBBLE CHAR IS IN
1E7A:B0 04       139        BCS   LNIBL       ;BRANCH IF IN LEFT NYBBLE
1E7C:29 0F       140        AND   #$0F        ;IN RIGHT NYBBLE- JUST MASK
1E7E:90 04       141        BCC   GETCH       ;GO GET PIXEL CHARACTER
1E80:4A          142 LNIBL: LSRA              ;SHIFT MAP
1E81:4A          143        LSRA              ; BYTE TO GET
1E82:4A          144        LSRA              ;  PIXEL INDEX
1E83:4A          145        LSRA              ;    IN RIGHT HALF
1E84:AA          146 GETCH: TAX               ;TRANSFER PIXEL INDEX TO X
1E85:BD EB 1D    147        LDA   CHAR,X      ;GET PIXEL CHAR + MODE BIT
1E88:4A          148        LSRA              ;TRANSFER MODE TO CARRY
1E89:48          149        PHA               ;SAVE CHARACTER
1E8A:AD F1 19    150        LDA   MODE        ;GET KTM-2 MODE FLAG
1E8D:30 06       151        BMI   MREV        ;BRANCH IF WE ARE IN REVERSE
1E8F:90 16       152        BCC   MDOK        ;BRANCH IF BOTH MODES NORMAL
1E91:A2 52       153        LDX   #'R'        ;SETUP TO CHANGE TO REVERSE
1E93:B0 04       154        BCS   MDCHG       ;GO CHANGE MODE
1E95:B0 10       155 MREV:  BCS   MDOK        ;BRANCH IF BOTH MODES REVERSE
1E97:A2 72       156        LDX   #'r'        ;SETUP TO CHANGE TO NORMAL
```



```
1E99:49 FF       157 MDCHG: EOR   #$FF       ;FLIP MODE FLAG
1E9B:8D F1 19    158        STA   MODE        ;STORE AS NEW KTM-2 MODE
1E9E:A9 1B       159        LDA   #$1B        ;ESCAPE CHARACTER
1EA0:20 47 8A    160        JSR   SEND        ;SEND TO KTM-2
1EA3:8A          161        TXA               ;TRANSFER MODE CONTROL TO A
1EA4:20 47 8A    162        JSR   SEND        ;SEND TO KTM-2
1EA7:A9 1B       163 MDOK:  LDA   #$1B        ;ESCAPE CHARACTER
1EA9:20 47 8A    164        JSR   SEND        ;SEND TO KTM-2
1EAC:A9 3D       165        LDA   #'='        ;ABSOLUTE CURSOR ADDRESSING
1EAE:20 47 8A    166        JSR   SEND        ;SEND TO KTM-2
1EB1:AD EE 19    167        LDA   ROW         ;GET TRUE ROW ADDRESS
1EB4:18          168        CLC               ;ADD BIAS REQUIRED
1EB5:69 20       169        ADC   #$20        ; BY KTM-2
1EB7:20 47 8A    170        JSR   SEND        ;SEND TO KTM-2
1EBA:AD EF 19    171        LDA   COL         ;GET TRUE COLUMN
1EBD:18          172        CLC               ;ADD BIAS REQUIRED
1EBE:69 20       173        ADC   #$20        ; BY KTM-2
1EC0:20 47 8A    174        JSR   SEND        ;SEND TO KTM-2
1EC3:68          175        PLA               ;RETRIEVE PIXEL CHARACTER
1EC4:4C 47 8A    176        JMP   SEND        ;SEND TO KTM-2 AND RETURN
```

reference name table

name	size	value dec	value hex
BSRET	2	53580	D14C
CEVEN	2	7736	1E38
CHAR	2	7659	1DEB
CINDX	2	6642	19F2
CLEAR	2	7675	1DFB
CLRM	2	34595	8723
COL	2	6639	19EF
FLAG	2	6640	19F0
GETCH	2	7812	1E84
LNIBL	2	7808	1E80
MAP	2	6699	1A2B
MAPE	2	7658	1DEA
MASK	2	6643	19F3
MDCHG	2	7833	1E99
MDOK	2	7847	1EA7
MODE	2	6641	19F1
MREV	2	7829	1E95
PLOT	2	7722	1E2A
PXOFF	2	7785	1E69
RESET	2	7712	1E20
REVEN	2	7755	1E4B
RNIBL	2	7746	1E42
ROW	2	6638	19EE
RPTR	1	238	00EE
RSTRB	2	7798	1E76
RTAB	2	6651	19FB
SEND	2	35399	8A47
SET	2	7720	1E28
SETPX	2	7790	1E6E
TEST	2	7716	1E24
WPOFF	2	35718	8B86
WPON	2	35740	8B9C
ZWORK	1	254	00FE

```
100 REM: DEMONSTRATION PROGRAM FOR BILL GOWANS'
110 REM: HI-DENSITY PLOT ROUTINE FOR THE KTM-2/80
120 REM: (Edited slightly by Lux)
130 REM:
140 C=&"1DFB":S=&"1E28"
150 ESC$=CHR$(27)
160 X=2:Y=3:X1=1:Y1=1
170 Q=USR(C,0)
180 PRINT ESC$+"G"
190 FOR I=1TO2066
200 Q=USR(S,256*Y+X)
210 IFX>158ORX<1THEN X1=-X1
220 IFY>46ORY<1THENY1=-Y1
230 X=X+X1:Y=Y+Y1
240 NEXT
250 PRINT ESC$+"g"+ESC$+"r"
260 PRINT ESC$+"="+CHR$(32+21)+CHR$(32+0);
270 END
OK
```

```
19F0 00 00 00 01 02 04 08 10,1F
19F8 20 40 80 2B 1A 53 1A 7B,2C
1A00 1A A3 1A CB 1A F3 1A 1B,10
1A08 1B 43 1B 6B 1B 93 1B BB,78
1A10 1B E3 1B 0B 1C 33 1C 5B,62
1A18 1C 83 1C AB 1C D3 1C FB,CE
1A20 1C 23 1D 4B 1D 73 1D 9B,BD
1A28 1D C3 1D 00 00 00 00 00,BA
 0FBA
.
```

Continued on Page 18

NOTE: For the KTM-2/40, change 158 to 78 in line 210.

A BUG IN THE RAE-1 RELOCATING LOADER?

We received the following letter from J. J. Sullivan, 19 Sylvester Drive, Kallangur, Qld., 4503, Australia, during the bi-monthly "crisis" period when we get SYM-PHYSIS ready for the printers, and thought that the question posed was worth an immediate answer:

Dear Dr. Luxenberg,

I have discovered an interesting problem with the RAE-1. I solved it, so it is no worry but I have enough curiosity for six cats.

Originally I had intended to leave the relocating loader alone add depend on your relocate programme but changed my mind for two reasons. One was the discovery that Relocate doesn't catch everything. For example, it misses several adjustments in the Ultra-Renumber programme,two that you are warned about and one that you are not warned about---except possibly by indirection and hindsight. The other reason was that I read your RAE Notes and when I cross-referenced them to the manual, particularly section 4.6, paragraph five, I started going round in circles.

The only solution was to punch up the relocater source code and start experimenting. Eventually I got it and understood what everyone was talking about. If only someone had said "Use OU instead of PA" it would have saved me a lot of trouble.

Anyway, I had the loader in memory add I had it as a relocatable tape so I set out to load it. I followed instructions religiously---and absolutely nothing happened. I tried everything, even to disassembling the programme and laboriously checking it, byte by byte, against the code in the manual. It seemed ridiculous to suspect the programme, since it worked for Synertek and it worked for you, but there was nothing else left.

Eventually I zeroed in on line 3810. Why the three byte offset? I spent a long time with the monitor programme but I still couldn't see the reason. In fact, as I saw it, that offset was a guarantee that the tape wouldn't move. finally, I changed the code to 20 78 8c and everything worked like a charm. I loaded the tape, relocated it and used it to load itself again. I figured that was a pretty fair check.

I immediately duplicated and amended the source programme and stored it for future reference.

As you can see, I have no immediate problem except this bump of curiosity. Consequently, I will be intently watching future issues of RAE Notes and the newsletter to see if there is any reference to this matter, because I don't imagine I will be the only one with this problem.

What has me baffled is the fact that the programme worked for Synertek and worked for you. I don't see how it could.

Yours faithfully,

(J. J. Sullivan)

Dear Mr Sullivan:

Our early version of the relocating loader appears to be identical to the one published in the RAE-1 Reference Manual, at least in the area in question, and works with MON 1.1; it will not work with MON 1.0. Your fix will make the loader work with both MON 1.0 and MON 1.1.

Here is the explanation for both the "why" and the "why not". If you go directly to LOADT at line 3810 Recorder 0 (write) will start. Of course if you have turned it off this is no problem. Since you are not in RAE when you use the loader, you will have to turn on Recorder 1 (read) by hand. This is no problem either, since you enter with .G 0200, start the tape manually, and stop it when the "." appears again. We have never bothered to add on the relay for the read recorder, since the "S" prompt on the SYM tells us when to start the read recorder. Besides, one day soon we will be all disk!

The entry at LOADT+$3 skips the turnon of Recorder 0 in MON 1.1, but could get you lost in MON 1.0 (have never tried it, and have not checked out the code since MON 1.0 is obsolete). While the starting addresses for LOADT are the same in both MONs, the subroutines differ nearly everywhere else; they even use different timers (6532 vs. 6522). "Historically speaking", the changes were made to eliminate a KIM format read bug in MON, a JMP WARMSTART bug in BAS-1, and the need to hit RST to abort an unwanted LOADT. Many other changes were included at the same time to very much enhance the versatility of the VIM (Versatile Interface Monitor).

If you replace LOADT+$3 with LOADT, as you have done, note that much of the coding between lines 3720 and 3810 can be dropped because the instructions are repeated in JSR START, which is called by LOADT.

Hope this satisfies your curiosity. I enjoy using the relocating loader, and .CT; one day soon I hope to have disk system equivalents for both of these. And yes, it is unfortunate, but true, that the manual does not make it explicitly clear that, to produce a relocatable object code dump on tape, when you are assembling from tape, you must use >OU, instead of >PA for the second pass!

My major regret these days is that 95% of my time on the SYM is spent processing words, rather than doing all of the work with graphics, music, voice synthesis, pattern recognition, etc., for which I feel both my SYM and I were destined!

I always enjoy your letters.

Regards,

Lux

Continued from Page 16

```
1DE8 00 00 00 C1 99 97 E8 95,6E
1DF0 C9 BD 92 93 BC C8 94 E9,1A
1DF8 96 98 F9 A9 0C 20 47 8A,E7
1E00 20 86 8B A9 2B 85 FE A9,18
1E08 1A 85 FF A9 EA 8D 4A A6,C6
1E10 A9 1D 8D 4B A6 A9 00 8D,40
1E18 F1 19 20 23 87 4C 9C 8B,87
1E20 A2 00 F0 06 A2 80 30 02,73
1E28 A2 40 8E F0 19 48 98 A2,6E
1E30 00 4A 8D EF 19 90 01 E8,C6
1E38 4A 8D F2 19 B0 04 E8 E8,2C
1E40 E8 E8 68 4A 8D EE 19 90,D2
1E48 02 E8 E8 0A A8 B9 FB 19,23
1E50 85 EE B9 FC 19 85 EF AC,84
1E58 F2 19 B1 EE 2C F0 19 10,73
1E60 0D A0 00 3D F3 19 F0 01,5A
1E68 C8 A9 00 4C 4C D1 1D F3,44
1E70 19 70 03 5D F3 19 91 EE,B8
1E78 E0 04 B0 04 29 0F 90 04,1C
1E80 4A 4A 4A 4A AA BD EB 1D,B3
1E88 4A 48 AD F1 19 30 06 90,C2
1E90 16 A2 52 B0 04 B0 10 A2,E2
1E98 72 49 FF 8D F1 19 A9 1B,F7
1EA0 20 47 8A 8A 20 47 8A A9,0C
1EA8 1B 20 47 8A A9 3D 20 47,65
1EB0 8A AD EE 19 18 69 20 20,64
1EB8 47 8A AD EF 19 18 69 20,8B
1EC0 20 47 8A 68 4C 47 8A,01
 6601
```

SCOPE GRAPHICS AND COMPUTER 'GENERATED' MUSIC

Here, combined, are a couple of novelty demo programs, that have resided in our high RAM, along with our utility programs, for years. They have been written as subroutines callable from MON, BAS, and RAE, and return to the caller when the Terminal BREAK key is held down. The music program is based on T. C. O'Haver's 'More Music for the 6502', BYTE, June 1978. The scope graphics program is based on one given by Roy Flacco in 'Graphics Interface', which he calls 'Starburst Graphics', in 6502 User Notes Issue 9/10. Mr. Flacco's program is, in turn, based on D. John Anderson's 'Serendipitous Circles', BYTE, August, 1977. Incidentally, the 'Swirl' program supplied with MTU's Visible Memory is closely related.

The original articles fully describe how to change parameters to change the appearance of the display, or the sound of the music. Our version of the programs initializes the starting values to provide an interesting mixture of the 'expected' and the 'unexpected'. Sorry there's no source code, but the programs are short, and the algorithms are simple! The programs have been moved to low RAM for smaller SYM's, and will require two simple six-bit DAC's, as shown in the sketch. The design is a modification of the one given in Chamberlin's music article; the resistor values were changed to fit values carried in stock by Radio Shack. A second sketch shows an 'add-on' to provide an eight-bit DAC. A simple, one transistor, or single chip, amplifier of nearly any type will provide the audio. The two DAC's are connected to PA0 through PA5 and PB0 through PB5 on the Application Connector The sketches are rough (please forgive the quality); maybe one day SYM can be trained to do my drawings on paper, as well as my typing.

'STARBURST' SCOPE GRAPHICS PROGRAM

```
0200-  A9 F2     LDA   #F2
0202-  85 F5     STA   F5
0204-  A9 8E     LDA   #8E
0206-  85 F6     STA   F6
0208-  A9 3F     LDA   #3F
020A-  8D 03 A0  STA   A003
020D-  8D 02 A0  STA   A002
0210-  A5 F6     LDA   F6
0212-  4A        LSR   A
0213-  49 FE     EOR   #FE
0215-  EA        NOP
0216-  EA        NOP
0217-  38        SEC
0218-  65 F5     ADC   F5
021A-  EA        NOP
021B-  EA        NOP
021C-  85 F5     STA   F5
021E-  4A        LSR   A
021F-  18        CLC
0220-  65 F6     ADC   F6
0222-  EA        NOP
0223-  EA        NOP
0224-  85 F6     STA   F6
0226-  4A        LSR   A
0227-  85 F8     STA   F8
0229-  49 FF     EOR   #FF
022B-  38        SEC
022C-  69 00     ADC   #00
022E-  EA        NOP
022F-  EA        NOP
0230-  85 EA     STA   EA
0232-  A5 F5     LDA   F5
0234-  4A        LSR   A
0235-  85 F7     STA   F7
0237-  49 FF     EOR   #FF
0239-  EA        NOP
023A-  EA        NOP
023B-  38        SEC
023C-  69 00     ADC   #00
023E-  85 E9     STA   E9
0240-  A0 04     LDY   #04
0242-  A6 F7     LDX   F7
0244-  A5 F8     LDA   F8
0246-  20 64 02  JSR   0264
0249-  A6 E9     LDX   E9
024B-  A5 F8     LDA   F8
024D-  20 64 02  JSR   0264
0250-  A6 E9     LDX   E9
0252-  A5 EA     LDA   EA
0254-  20 64 02  JSR   0264
0257-  A6 F7     LDX   F7
0259-  A5 EA     LDA   EA
025B-  20 64 02  JSR   0264
025E-  88        DEY
025F-  10 E1     BPL   0242
0261-  4C 7C 02  JMP   027C
0264-  18        CLC
0265-  69 20     ADC   #20
0267-  8D 00 A0  STA   A000
026A-  8A        TXA
026B-  18        CLC
026C-  69 20     ADC   #20
026E-  8D 01 A0  STA   A001
```



```
0271-  A9 20     LDA   #20
0273-  8D 1D A4  STA   A41D
0276-  2C 04 A4  BIT   A404
0279-  10 FB     BPL   0276
027B-  60        RTS
027C-  20 86 83  JSR   8386
027F-  B0 02     BCS   0283
0281-  90 8D     BCC   0210
0283-  60        RTS

0200 A9 F2 85 F5 A9 8E 85 F6,C7
0208 A9 3F 8D 03 A0 8D 02 A0,0E
0210 A5 F6 4A 49 FE EA EA 38,46
0218 65 F5 EA EA 85 F5 4A 18,50
0220 65 F6 EA EA 85 F6 4A 85,C9
0228 F8 49 FF 38 69 00 EA EA,7E
0230 85 EA A5 F5 4A 85 F7 49,96
0238 FF EA EA 38 69 00 85 E9,78
0240 A0 04 A6 F7 A5 F8 20 64,DA
0248 02 A6 E9 A5 F8 20 64 02,8E
0250 A6 E9 A5 EA 20 64 02 A6,D8
0258 F7 A5 EA 20 64 02 88 10,7C
0260 E1 4C 7C 02 18 69 20 8D,55
0268 00 A0 8A 18 69 20 8D 01,AE
0270 A0 A9 20 8D 1D A4 2C 04,95
0278 A4 10 FB 60 20 86 83 B0,7D
0280 02 90 8D 60,FC
 44FC
```

'MUSIC GENERATOR' PROGRAM

```
0284-  A9 08     LDA   #08
0286-  85 EE     STA   EE
0288-  A9 0F     LDA   #0F
028A-  85 EF     STA   EF
028C-  A9 0D     LDA   #0D
028E-  85 F2     STA   F2
0290-  A9 3F     LDA   #3F
0292-  8D 03 A0  STA   A003
0295-  A0 00     LDY   #00
0297-  98        TYA
0298-  29 F0     AND   #F0
029A-  4A        LSR   A
029B-  4A        LSR   A
029C-  4A        LSR   A
029D-  4A        LSR   A
029E-  85 F0     STA   F0
02A0-  98        TYA
02A1-  29 0F     AND   #0F
02A3-  25 F0     AND   F0
02A5-  65 F0     ADC   F0
02A7-  25 EF     AND   EF
02A9-  85 F0     STA   F0
02AB-  A2 00     LDX   #00
02AD-  A5 F2     LDA   F2
02AF-  85 F4     STA   F4
02B1-  BD 00 03  LDA   0300,X
02B4-  8D 01 A0  STA   A001
02B7-  8A        TXA
02B8-  18        CLC
02B9-  65 F0     ADC   F0
02BB-  AA        TAX
02BC-  C6 F1     DEC   F1
02BE-  D0 06     BNE   02C6
02C0-  C6 F4     DEC   F4
02C2-  D0 ED     BNE   02B1
02C4-  F0 04     BEQ   02CA
02C6-  EA        NOP
02C7-  18        CLC
02C8-  90 E7     BCC   02B1
02CA-  C8        INY
02CB-  C6 F3     DEC   F3
02CD-  D0 C8     BNE   0297
02CF-  A5 EE     LDA   EE
02D1-  85 F3     STA   F3
02D3-  A9 02     LDA   #02
02D5-  85 F0     STA   F0
02D7-  20 86 83  JSR   8386
02DA-  90 CF     BCC   02AB
02DC-  60        RTS

0284 A9 08 85 EE A9 0F 85 EF,50
028C A9 0D 85 F2 A9 3F 8D 03,F5
0294 A0 A0 00 98 29 F0 4A 4A,7A
029C 4A 4A 85 F0 98 29 0F 25,78
02A4 F0 65 F0 25 EF 85 F0 A2,E8
02AC 00 A5 F2 85 F4 BD 00 03,B8
02B4 8D 01 A0 8A 18 65 F0 AA,87
02BC C6 F1 D0 06 C6 F4 D0 ED,8B
02C4 F0 04 EA 18 90 E7 C8 C6,86
02CC F3 D0 C8 A5 EE 85 F3 A9,C5
02D4 02 85 F0 20 86 83 90 CF,C4
02DC 60,24
 3124
```

HEX DUMP OF 'VOICE' TABLE 'MUSIC GENERATOR'

```
0300 32 34 35 36 36 37 38 39,AF
0308 39 3A 3A 3B 3B 3B 3C 3D,86
0310 3C 3C 3C 3C 3C 3C 3C 3C,66
0318 3C 3C 3C 3B 3B 3B 3B 3B,41
0320 3A 3A 3A 3A 3A 3A 39 39,0F
0328 39 39 39 39 39 39 39 39,D7
0330 3A 3A 3A 3A 3A 3B 3B 3B,AA
0338 3B 3C 3C 3C 3D 3D 3D 3D,8D
0340 3E 3E 3E 3E 3F 3F 3F 3F,81
0348 3F 3F 3F 3F 3F 3F 3F 3F,79
0350 3E 3E 3E 3D 3D 3C 3E 3B,62
0358 3B 3A 39 38 38 37 36 35,22
0360 36 33 32 31 32 2F 2E 2D,AA
0368 2E 2B 2A 29 2A 27 26 25,F2
0370 24 23 22 21 21 20 1F 1F,FB
0378 1E 1E 1D 1D 1F 1D 1E 1C,E7
0380 1C 1C 1D 1D 1D 1D 1D 1E,CE
0388 1E 1F 1F 20 20 21 21 22,CE
0390 23 23 24 24 25 26 26 27,F4
0398 28 28 29 29 29 2A 2A 2B,3E
03A0 2B 2B 2B 2B 2B 2B 2B 2A,95
03A8 2A 2A 29 29 28 27 27 26,D7
03B0 25 24 23 22 21 20 1F 1D,E2
03B8 1C 1B 19 18 17 15 14 13,9D
03C0 11 10 0F 0D 0D 0B 09 08,03
03C8 07 06 05 04 03 03 03 01,23
03D0 01 00 00 00 00 00 01 00,25
03D8 03 00 01 01 01 02 03 04,34
03E0 07 06 07 08 09 0B 0C 0D,7D
03E8 0F 10 12 13 15 16 18 1A,1E
03F0 1B 1D 1F 20 23 23 25 27,27
03F8 28 2A 2B 2C 2E 2F 30 31,8E
 278E
```

COMPUTER MUSIC

One of the most helpful articles available on computer played (not computer composed) music is Hal Chamberlin's "A Sampling of Techniques for Computer Performance of Music", BYTE, September 1977. This "classical" article has been reprinted in The BYTE Book of Computer Music, available at many computer stores, and will prove to be your best starting point. Next, read Hal's updating article on "Advanced Real-Time Music Synthesis Techniques", BYTE, April 1980. We heard a demonstration of Chamberlin's advanced techniques, at the West Coast Computer Faire in March, and were much impressed.

In the original article, Mr. Chamberlin gives 6502 subroutines for tone generation, and shows a simple one-transistor amplifier you can hang onto any output port bit (on the SYM you can adapt any one of the unused on-board buffers for this purpose). You can use either timed delay loops or the pair of timers in one of the VIA's to generate any desired tone for any desired duration. We recommend that you try both methods. With either of these approaches the sound timbre is limited to what you can get by changing the duty-cycle of the square wave.

For a richer range of timbre, Hal (and we) recommend the DAC (digital-analog converter) approach. The article gives all circuit details necessary to build-your-own, so we will not repeat the details here. You can also use any commercially available D/A chip. We recommend that that you consider the complete DAC board manufactured by Mr. Chamberlin's company, Micro Technology Unlimited (MTU). It includes its own audio amplifier, and also includes a sharp cut-off low-pass filter, necessary to eliminate the "aliasing" distortion introduced by sampling a wave-form table at too high a rate. This distortion is particularly annoying on the higher frequency notes. A copy of the original article is supplied with the board, as is a KIM demonstration tape. Since the KIM tape is incompatible with SYM (pages zero and one are included), we have made arrangements with MTU to provide SYM tapes. MTU also has an Advanced Music Software package written for the KIM. We will provide an Appendix to their package and a cassette for the SYM. See back page for ordering information.

We have been using the Advanced Music Software package for nearly two years. It contains a Fourier synthesis subroutine for generating wave shapes, the NOTRAN (NOte TRANslator) Compiler, the NOTRAN Interpreter, and a demonstration NOTRAN "Score". The SYM-1 version has been reorganized to eliminate problems with pages zero and one read-in, and is started with an .E instead a .G, to initialize the page zero data. Whenever visitors ask about our SYM, "But what is it good for?", they are most impressed with SYM'S rendition of "The Star Spangled Banner", "Exodus", the NOTRAN score and, at Christmas time, "Deck The Halls". Only the NOTRAN Compiler portion of the Advanced Music Software package requires a terminal, but because the input/output portion of the program is written as a "patch", you may write your own, to make use of the hex pad and segment displays.

MUSIC FOR THE SYMPLE SYM

You can play some interesting music on the completely "unimproved" SYM-1. The only added "hardware" you will need, and you can "borrow" that, is a "cheapie" AM radio tuned to a clear spot on the dial, and parked near the SYM. I have a radio sitting near my floppy disk system, and the rhythm effects during a long disk-to-disk copy helps to pass the time away. Later, you may wish to add a small speaker or a transistor radio type earphone through a one transistor buffer. Use one of the four available transistor buffers on the SYM itself. These may be re-wired as desired, and to or from any I/O pin. If your cassette recorder permits monitoring during recording, you may use it as your audio output device. And, now, about software.....

MORE ON JACK BROWN'S THREE BASIC ENHANCEMENTS

Jack Brown is now using RAE-1 instead of the very good Microware Assembler he adapted from his KIM-1 system. He has also replaced his older terminal with a KTM-2/80, and he will be getting a copy of the SYM WORD PROCESSOR (SWP-1).

We are declaring his original articles "out-of-print" (we Xeroxed copies of the originals, as the orders came in, and could still make additional copies, if required), and replacing them with a second edition. The second edition includes a 16 page manual, and a cassette dump of the source code in RAE format, which is heavily commented. The full source code will require .CT. We think that we will also include an abbreviated source code, with the original line numbers, but stripped of comments and remarks, so that it can be assembled in a single pass on a 16 K SYM, if possible. The new package will be available 1 June 1980.

We keep careful records on what each individual subscriber buys from the User's Group, so we can send them errata sheets and updates. To keep faith with those who purchased any of the original three Brown articles, we will consider the second edition to be in the nature of an update, and allow full credit for previous purchases to be applied against the cost of the second edition.

HIGH RESOLUTION GRAPHICS

As you have seen in Bill Gowans' article, any terminal with cursor control can be used as a "plotter", with resolution up to the number of cursor positions available. If, in addition, the terminal, like either the KTM-2 or the KTM-2/80, provides a set of graphics symbols, the resolution may be doubled.

"Self-contained" systems, e.g., Pet and Apple, do not communicate with their built-in CRT screens over a serial data line. Rather, a portion of memory is "mapped" onto the screen. The memory is treated by the 6502 as ordinary memory; the 6502 need not concern itself (no software is required) with getting the points on the screen.

If you wish high resolution graphics, like the Apples' 280x192, you will need an 8 K memory board with video capability. There are a number of such boards available; the one to get depends mainly on the expansion bus structure, and system package approach you select. We like MTU's package approach (it took us over two years to make up our minds!), so we now have their 8 K Visible Memory. We have had it less than a week, and took off a few hours from preparing this issue to get "Random Checkerboard", "Swirl", and "Life" going. Note that with its resolution of 320x200 it will permit a text display of 22 lines of 53 characters. This is better than the KTM-2, but I will still want the -2/80 for word processing.

The Visible Memory and a simple QWERTY keyboard can be used together in place of a serial terminal. Software (for KIM-1) is provided. Nelson Edwards, who played a large role in designing SUPERMON, has sent me a portion of his SYM version of the MTU software, to help us in our conversion. The SYM version will be shorter than the KIM version because of all of the utility subroutines in SUPERMON!

Note that Bill Gowans called his graphics with a single parameter USR function, combining Y and X into one parameter. The Visible Memory will need a two parameter USR function, since 320 > 255. A far better approach, however, is to patch a full set of Graphics Commands to BAS. We will be working on this ourselves, and will serve as a "clearing house" for information on Visible Memory Software.

BASIC AND THE 2K SYMBOLIC ASSEMBLER

Many SYMmers use BASIC as their 'first language', and do their text processing in BASIC, rather than with RAE. For the occasional short machine language utilities they write to support BASIC, the 2KSA is a natural. Here is a portion of a letter from Bruce Thompson, Applied Physics, Cornell University, Ithaca, NY 14853, and a copy of the program he mentions, written in 2KSA format. Speaking of 2KSA, we will shortly be mailing out an update sheet.

Enclosed is a short program called by BASIC's USR to dump or load specific memory locations, e.g.if you POKE'd a data file into some unused memory, you can dump it under program control; or you can bring in successive data files to be used by BASIC. On load the error code is returned.

Basic USR Module

LSDATA - To load or save specific memory locations under program control.

Called by X=USR(address,Flag/File,start,end)

where address is that of the module
Flag/file has a zero in the first byte for load
anything else in the first byte for save
has the file no. in the second byte
start is the start address of the data
end is the address of the last byte of the data
X will be zero for no error
47.=$2F no EOF
255.=$FF framing error
204.=$CC checksum error

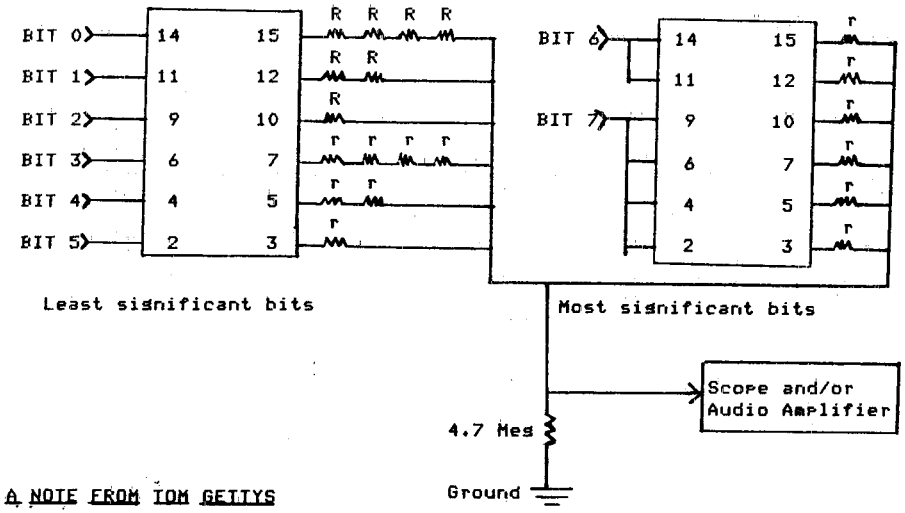
For example: X=USR(&"0E00",&"00FF",&"0C00",&"0DFF")
will load the next file on the tape into locations $0C00 to $0DFF inclusive and indicate a read error by the value of X, provided the module is located at $0E00.

```
20868B  LSDATA  JSR ACCESS
AA              TAX             set END from [A,Y] and add 1
C8              INY
8C4AA6          STY     PARM
D001            BNE     NOBUMP
E8              INX
8E4BA6  NOBUMP  STX     PARM+1
68              PLA
8D4CA6          STA     PARM+2  set START L
68              PLA
8D4DA6          STA     PARM+3  set START H
68              PLA
8D4EA6          STA     PARM+4  set FILE#
A080            LDY#    80      high speed flag
68              PLA             00=load
F005            BEQ     LOAD
20878E          JSR     DUMPT
9009            BCC     CSET
20788C  LOAD    JSR     LOADT
A8              TAY             error code for load in Y
A900            LDA#    00
B001            BCS     CSET
A8              TAY
209C8B  CSET    JSR     NACCESS
4C4CD1          JUMP    RETURN
```


INEXPENSIVE D/A CONVERTER

SEE CORRECTION IN ISSUE #4

Chip is a 4050. Pin 1 is +5 V. Pin 8 is Ground. Pins 13 and 16, N.C. R is 220 K; r is 27 K.
You will need one for music, two for scope graphics. The most significant bits section is optional, but you may want it for music applications.



A NOTE FROM TOM GETTYS

A common control structure is the implementation of a computed GOTO or GOSUB. It is not unusual for the flow of control within a program to depend on data entered by a user, as in an editor or interactive game program, or on periodic sampled inputs such as those in real-time control systems.

Here are two methods of implementing an indexed indirect JMP or JSR on a 6502-based machine. The first method, called 'vectoring', is used extensively by the SYM monitor and is one reason the SYM is so versatile a computer. Three bytes are reserved, with the first containing a hex 4C (JMP). After the target address has been computed or looked up it is placed in the next 2 bytes of the vector. A JMP or JSR to the vector causes control to pass to the selected module.

The second method, however, is the more effective and concinnate. Let's suppose we wish to call routine X, and that the address table is structured as 2 rows: TBL.LO containing the low-order bytes and TBL.HI the high-order bytes. Consider the following routine:

```
CALL.X  LDA TBL.HI,X  ;GET ADDRESS X, HIGH BYTE
        PHA           ;AND PUSH IT TO THE STACK
        LDA TBL.LO,X  ;GET ADDRESS X, LOW BYTE
        PHA           ;AND PUSH IT TO THE STACK
        RTS           ;GO TO ROUTINE X
```

By doing a JMP or JSR to CALL.X an indexed indirect JMP or JSR will be effected to the Xth routine. One point to be observed here is that the execution of a RTS instruction pops the stack into the program counter, and then increments it. Thus the addresses in the table must be one less than their actual value.

A TAPE OPERATING SYSTEM FOR SYM

Frank Winters, School of Marketing, University of New South Wales, P. O. Box 1, Kensington, Sydney, Australia 2033, sent us a brief note, and an "unreadable" tape a few weeks ago. The tape sounded rather high pitched, I thought. That very same day I received a letter from Manfred Burow, Kapuzinerstr. 2, D-8000 Muenchen 2, West Germany, who explained how he only got perfect cassette performance by lowering C16 to to 0.022 uFd, but could now read cassettes written at 5600 Baud. I tried Frank's tapes again at 2800 Baud, and they read beautifully!

The program Frank sent was a "teaser". I wrote for more info and he sent a new cassette with source code, and some handwritten notes, describing his Tape OPerating System. He calls it TOPS, I call it TOPSY, because like Topsy in "Uncle Tom's Cabin", it seems to have just grown. He has added solenoids to his recorders, for start, stop, fast forward and rewind, under computer control. He formats the tapes, they contain their own index data, etc., just like a disk system. The source calls out some external addresses by hex values, so I can't relocate it too easily. Will tell you more about it next issue.

Frank sent along a long voice recording telling me about his work and other interests; I still owe him a personal answer. Frank would like to hear from other hams on 20 meters. His call is VK2BLF.

FIX FOR THE BUG IN MOSER'S PADDLE GAME

Kin-Ming Kwok, 22 Tung Choi St., 10th Floor, Flat A, Mongkok, Hong Kong, offers the following fix for the bug mentioned in the listing of the game:

```
1000 LN = 23
1260 PRINT CHR$(64)
1270 NEXT: PRINT CHR$(27)+CHR$(103);
1335 AA=USR(4096+132,0)
1770 AA=USR(AA*256): PRINT CHR$(8);
```

SOFTWARE RECOMMENDATION

Jeff Holtzman has sent us preview copies of several very useful utility packages for SYM-1, both on cassette and in EPROM. We have tested the cassette versions (no extra PROM sockets yet!) and have found them very well designed, indeed. He is offering a package of SUPERMON Extensions, which includes an interactive trace/debug feature, SYM-BUG, and the following new commands:

CMD	PAR.NR	DESCRIPTION
A	0-2	Memory dumped as ASCII
B	0-1	Sets/deletes BRK instruction
F	0	Prints user flags as binary
F	2	Finds user string (hex and/or ASCII)
H	3	Performs 16 bit Boolean algebra - AND, OR, EOR
K	0-1	Dumps stack with checksum
P	0	Sets/resets line printer driver (see note 1)
R	3	Program relocater - adjusts abs. and rel addresses
T	0	Enter interactive trace mode
X	0-2	Disassembler (see note 2)
Y	0	User link - does indirect JMP to sys. ram loc. JUMP6
Z	2	Calculates 16 bit check sum of memory (prints sum only)

SYM-BUG, and the Command Extensions, are available in object code on cassette for $16, and in 2716 EPROM for $50, including a User's Manual. The User's Manual is available separately for $6. The fully commented source code listing is available for $10. Cassette versions are assembled at $0200 or $3800. EPROM version is assembled at $F000. Custom assembly at other locations is an additional $2. Overseas add $2 for Air Mail Postage. Please order direct from Jeff Holtzman, 6820 Delmar #203, St. Louis, MO 63130.

MISCELLANEOUS NOTES

Our HDE, Inc., disk system is working quite well, thank you. Only one very minor bug that we have found; in the warm start of FODS after reset, it "stutters" once, then continues properly. Lanny Maude, of Advanced Computer Products, 1310 Edinger, Santa Ana, CA 92705, has a copy of our SYM/FODS System Disk, and will shortly have his own SYM-1 operating with the HDE Disk System. Incidentally, Advanced Computer Products is the first computer store to sell SYM-PHYSIS over the counter. They issue a very informative catalog; if you write for one, tell 'em "SYM-PHYSIS sent me".

Plans for interfacing the MC 6847/AMI 68047 VDG alphagraphics chip to SYM, at a cost of less than $60, are now available from Marc Aagenas. Plans include a schematic, wiring check list, parts list, and driver software source code listings. Price in U.S. Funds is $10.00 in the U.S., $11.00 in Canada, and $15.00 elsewhere. Send orders to Marc Aagenas, 1674 East M-36, Pinckney, Mich.48169, U.S.A. Please include a mailing label with your name and address.

Our SYM now speaks to us, through Dave Kemp's SP-1 Speech Synthesizer Interface to the Texas Instruments' "Speak & Spell" (tm) (see page 1-21). It's fun to use it with .V, to help verify a long object code entry. SYM now speaks only "Hex", but the SP-1 Manual explains how to extend its vocabulary. If SYM can speak and play music, surely I should be able to teach it to sing! Would any other users of the SP-1 like to swap software?

One of my associates, "Skip" Frisbee, lent me his home-built, General Instrunents AY-3-8910 chip based, computer controlled, sound generation system. The parts cost under $50; and it has real potential for music and sound effects creativity. "Skip" promises us some -8910 driver software as soon as I return his system to him!

Here are some tips for beginners only, others may skip: After you have added the indispensable power supply, and the convenient cassette recorder, start reading Lance Leventhal's "6502 Assembly Language Programming" (see page 2-27). Next you will want on-board memory expansion; see page 3-27 for prices on "sets" of 2114 memory chips. If, after adding a terminal, your finances are temporarily strained, and you need some low-cost software to exercise your terminal, consider either Tiny BASIC, or the 2KSA, depending on your specific interests or applications. By this time, you are no longer a beginner, and will then want either BAS-1 or RAE-1, or both, and an additional 4K of on-board memory using the Blalock board. You might want to add the MTU DAC, described in this issue, even before the terminal, to give you some interfacing experience. In the next issue we will describe memory expansion approaches from which you can select, when you are ready to go "all the way".

Sorry that the mail comes in so fast that we have an ever increasing queue. Have tried to answer all "crisis" mail; other letters must wait. If you have real problems with SYM, feel free to call. We'll get your problem solved, somehow. Had better stop now; so tired I tried to insert two floppies into the same drive at the same time!

A TERMINAL TIP

To put your terminal on "LOCAL", if you want to "doodle" with the KTM-2 while in MON, or if you want to print date, time, title, remarks, etc., on your TTY, or other printing terminal, use Control O. After doodling, or printing, return your terminal to "LINE" with another Control O. This feature is not-too-well explained in section 9.7 of the SYM Reference Manual.

SHOPPING LIST OF ITEMS AVAILABLE FROM SYM-1 USERS' GROUP
All prices given below are now obsolete. Please use prices on the most recent issued 'Shopping List'.

CARL MOSER'S SYM WORD PROCESSOR (SWP-1):
FULLY COMMENTED SOURCE CODE ON CASSETTE. THE MANUAL IS ALSO ON CASSETTE, WITH EXAMPLES OF THE USE OF SWP-1.
APERIODIC UPDATES AND FULL SUPPORT WILL BE PROVIDED.
PRICE $35.00, FIRST CLASS/AIR MAIL WORLD WIDE.

JACK GIERYIC'S 'JACK-BUILT PROGRAMS':
ON CASSETTE, WITH INSTRUCTION SHEET.
1. DEPTH CHARGE
2. OTHELLO
3. CONCENTRATION
4. GRAPHICS DEMONSTRATION PACKAGE
5. PLOT
6. BAR GRAPH
PRICE $6.00 FOR ANY ONE, $5.50 EACH FOR ANY ADDITIONAL PROGRAM.
ALL SIX FOR $30.00, FIRST CLASS/AIR MAIL WORLD WIDE.

JACK BROWN'S BASIC ENHANCEMENTS:
SECOND EDITION, SOURCE CODE ON CASSETTE IN RAE FORMAT, WITH SIXTEEN PAGE MANUAL. THE ORIGINAL EDITION, AS DESCRIBED IN SYM-PHYSIS ISSUE #2, IS NOW OUT-OF-PRINT. PURCHASERS OF THE ORIGINAL EDITION WILL RECEIVE FULL CREDIT TOWARDS THE PURCHASE OF THE SECOND EDITION.
APERIODIC UPDATES AND FULL SUPPORT WILL BE PROVIDED.
PRICE $35.00, FIRST CLASS/AIR MAIL WORLD WIDE.

MICRO TECHNOLOGY UNLIMITED PRODUCTS (SYM VERSIONS ONLY):
DAC MUSIC BOARD WITH HARDWARE MANUAL AND BYTE ARTICLE REPRINT. CASSETTE WITH OBJECT CODE AND THREE SONGS IS SUPPLIED.
PRICES, FIRST CLASS/AIR MAIL $51.00 US/CANADA, $52.00 EUROPE $53.00 ASIA/PACIFIC.
ADVANCED MUSIC SOFTWARE PACKAGE, WITH FULLY COMMENTED SOURCE CODE, AND OBJECT CODE ON CASSETTE.
PRICES, FIRST CLASS/AIR MAIL $21.50 US/CANADA, $22.00 EUROPE, $23.00 ASIA/PACIFIC.
VISIBLE MEMORY SOFTWARE ON CASSETTE WITH SUPPLEMENT TO MTU MANUAL AVAILABLE 1 JUNE. PLEASE WRITE FOR PRICES.

2114 MEMORY CHIPS:
6 CHIPS (3 K) FOR $33.00 FOR ON BOARD SOCKETS
8 CHIPS (4 K) FOR $42.00 FOR BLALOCK MEMORY BOARD
14 CHIPS (7 K) FOR $72.00 FOR BOTH
OVERSEAS ADD $1.00 FOR POSTAGE

SEE ISSUE #2 FOR PRICES ON THE FOLLOWING:
EXTENDED TINY BASIC FOR SYM-1, PITTMAN
6502 ASSEMBLY LANGUAGE PROGRAMMING, LEVENTHAL
RAE NOTES UPDATING SERVICE

SEE ISSUE #1 FOR PRICES ON THE FOLLOWING:
2K SYMBOLIC ASSEMBLER, DENISON
SYNERTEK TECHNICAL NOTES
SUPERMON VERSION 2
RAE-1/2
SYM-1 SCHEMATIC

WRITE OR CALL FOR PRICES ON OTHER SYM PRODUCTS, SOFTWARE OR HARDWARE.

BLALOCK ADDRESS CHANGE

John Blalock's correct address for the 4 K Memory Expansion Board, and the 'Double ROM' Adapter, is P. O. Box 39356, Phoenix, Arizona 85069.

BULK RATE
U.S. Postage
PAID
Chico, CA 95927
Permit # 430

TIME VALUE PRINTED MATTER

SYM-PHYSIS
SYM-1 Users' Group
P.O. Box 315
Chico, CA 95927

Address Correction Requested