A LETTER FROM JIM

This news letter is to bring you up-to-date with events and ideas since issue 15.
Firstly and most importantly, there has been a critical omission concerning JMON from issue 15. If you wish to run JMON without the DAT BOARD fitted then a simple mod must be done. The mod is described below:
Fit a 4k7 resistor between pin 15 of the 4049 socket and D6 of the Z80 (pin 10).
This mod forms a "PRIORITY GATE" to gate the inverted DATA AVAILABLE signal to the DATA BUSS so the software can read it.
If this mod is not fitted, the keyboard will not work.
This mod is described in the "MAGIC SQUARE" article with a minor difference - it directs you to connect one end of the resistor to pin 17 of the Z80 rather that pin 15 of the 4049.
Use the method described here.
Thank-you for buying my package. I am sure you will find it useful in many ways.
Originally the aims for this package were: 1, To give the JMON user some idea how the software works, 2 To help offset the loses involved in writing over one hundred different versions (I kid you not) and 3, to provide an outlet for some of the "junk software" that I had accumulated over the years.
Unfortunately, only the first aim has been achieved as the amount of work I have put into this project has been far more than originally planned for. As a result of the extra effort, the package is now six pages longer and this means that I will have to collate each copy by hand as the photocopier spits the dummy after twenty. This is also why the other bits and pieces have been held over until another day.
On the bright side, the listing is complete and should help you to understand the workings of the MONitor and also provide some useful routines. Some good ones to use are the LED scan routine at 01BA, the convert A to display code routine at 01DA and the amazing tableless HEX to ASCII conversion at 0271.
Some of the routines will be difficult to understand at first and may take a fair amount of study. Remember that a lot of features have been crammed into a tiny 2k and as a result the routines are highly optimized and not designed to be understood easily.
Make sure you are familiar with what the software is doing on the outside of the TEC before you try to understand the software itself.
The actual MONitor can be studied first without having to worry about the more complicated MENU DRIVER, PERIMETER HANDLER and tape software. You will definitely need a good book on the Z80 instruction set so that you can understand what instructions have set or reset the flags.
The PERIMETER HANDLER and MENU DRIVER will be much easier to understand when the user operation instructions are published in issue 16 (or in a separate TEC publication if the rest of TE drags its heels again).

INQUIRIES

I cannot respond to all correspondence that I may receive, but if you have any questions it is still a good idea to write in as I may answer the question through TEC TALK or a possible newsletter (or whatever) for Jim's package owners (if the package is popular, no promises!). It is also possible that the package will be altered if several people ask the same question and the up- dated page sent out to current owners.
Inquiries may also be answered over the phone but please understand that I am not an employee of Talking Electronics and there is no guarantee that I will be here or the staff here will be able to answer your question. If you do ring, please keep in mind that we only have limited time to talk to you and the same phone is used by all the TE readers.

HOW LONG WILL JMON AND JIM'S PACKAGE REMAIN CURRENT

There are some improvements that can be made to JMON but at this stage I see no point in creating JMON-1A. Possibly in issue 16 a cheap tape with a new MONitor with more specialize features will be available. This will load into a 2K RAM That is then switched by hardware to the 0000 address. This MONitor will be used in a complimentary role to JMON, not as a replacement.
There are some utilities floating around for JMON and these may appear on a tape in issue 16 and work with the original JMON.
By the time issue 17 appears, a new 4k ROM with both JMON mk2 and the utilities might be the go. Currently, my TEC is set-up with a 4k ROM in the EPROM socket with both halves of the ROM available at the same time thanks to a simple TEC mod. This proto-type system is working well and the issue 17 JMON/utilities combination will use this idea.

ISSUE 16 RE-THINK

Throughout issue 15 there are hints of what is planned for issue 16. Since the production of issue 15 it appears that I will only have time to produce the keyboard and a scaled down version of the memory expansion board. Anything major will have to wait until issue 17. This is quite OK. as it allows more room for reader send-ins and questions. I am looking forward to both.

GLOSSARY

CURRENT EDITING LOCATION (082E)
THIS IS THE ADDRESS THAT IS USUALLY DISPLAYED IN THE ADDRESS SECTION ON THE TEC LED DISPLAY. IT IS THE ADDRESS THAT IS SUBJECT TO MODIFICATION BY JMON.

MONITOR CONTROL BYTE (MCB) (082B)
THIS BYTE CONTAINS THE INFORMATION OF THE CURRENT WORKING STATE OF JMON. THE INFORMATION HELD IN THIS BYTE IS:
1 - THE CURRENT MODE OF JMON.
E.G. DATA, ADDRESS OR FUNCTION (NOT SHIFT AS SHIFT IS TESTED AND HANDLED DURING THE DATA KEY HANDLER ROUTINE). BITS 4 AND 5 ENCODE THE CURRENT MODE IN THE FOLLOWING WAY. BOTH BITS ARE LOW FOR THE DATA MODE, BIT 4 IS HIGH FOR THE ADDRESS MODE, BITS 4 AND 5 ARE HIGH FOR THE FUNCTION MODE. BIT 4 IS CALLED THE ADDRESS/FUNCTION BIT AS THE SOFTWARE ONLY NEEDS TO TEST THIS BIT TO FIND IF EITHER THE ADDRESS OR FUNCTION MODE IS ACTIVE. BIT 5 IS THE FUNCTION MODE ENABLED BIT.
2 - THE NUMBER OF THE CURRENT FUNCTION I.E. 1,2 OR 3.
THIS IS ENCODED IN BITS 2 AND 3. IF NO FUNCTION OR FUNCTION-1 IS ENABLED THEN BOTH BITS ARE LOW. IF FUNCTION-2 IS SELECTED THEN BIT 2 IS HIGH AND BIT 3 IS LOW. IF FUNCTION-3 IS SELECTED THEN BIT 3 IS HIGH AND BIT 2 IS LOW.
3 - THE NUMBER OF NIBBLES ENTERED
THIS IS ENCODED IN BITS 0 AND 1. IF NO NIBBLES HAVE BEEN ENTERED IN THE CURRENT EDITING LOCATION THEN BOTH BIT ARE LOW. IF ONE NIBBLE HAS BEEN ENTERED THEN BIT 0 IS HIGH AND BIT 1 IS LOW IS TWO NIBBLES HAVE BEEN ENTERED THEN BIT 0 IS LOW AND BIT 1 IS HIGH. JMON USES THESE BITS WHEN DECIDING ON THE AUTO-INCREMENT FEATURE. BITS 6 AND 7 ARE NOT USED.

DISPLAY BUFFER ADDRESS - (082C/D)
THE CONTENTS OF 082C/D POINTS TO THE LOCATION IN MEMORY OF THE 6 BYTE DISPLAY BUFFER (0800 FOR JMON AND 0806 FOR THE STEPPER). THE DISPLAY BUFFER ADDRESS POINTS TO THE LOWEST ADDRESS OF THE DISPLAY BUFFER WHICH CONTAINS THE LOW ORDER DATA DISPLAY BYTE.

KEY PLANT
THE KEY PLANT IS A FAKE KEY STROKE THAT MAY BE GENERATED BY THE "DURING SCAN/KEY LOOP" USER PATCH. THE PLANT ALLOWS JMON'S MONITOR FUNCTIONS TO BE SOFTWARE CONTROLLED E.G. YOU MAY WISH TO VIEW THE CONTENTS OF MEMORY BYTE BY BYTE. WITH THE KEY PLANT YOU CAN SET JMON UP TO AUTOMATICALLY INCREMENT THE CURRENT EDIT LOCATION EVERY FEW SECONDS.
THE PLANT IS IDENTIFIED BY THE USER PATCH STORING THE REQUIRED KEY VALUE IN, AND SETTING BIT 7 OF THE INPUT KEY BUFFER (0820).

AUTO KEY STATUS BYTE (082A)
THIS BYTE HOLDS THE INFORMATION REQUIRED FOR THE AUTO KEY REPEAT SECTION. THE INFORMATION HELD IN THIS BYTE IS EITHER ONE OF THE FOLLOWING:
A "NEXT KEY DETECTION WILL BE A FIRST DETECTION" SO JMON WILL PROCESS THE KEY IMMEDIATELY (BIT 7 HIGH). A TIMER (BITS 0-6) THAT COUNTS A DELAY FOR THE AUTO REPEAT TIMING.

KEY PRESS FLAG (0825)
THIS FLAG IS USED TO REMEMBER IF THE ONE KEY PRESS HAS ALREADY BEEN DETECTED AND PROCESSED. THIS PREVENTS THE SAME KEY BEING PROCESSED EACH TIME THE SOFTWARE FINDS THAT IT IS PUSHED.
THIS IS THE WAY IT WORKS:
THE KEY PRESS FLAG IS ZEROED BY THE JMON DEFAULT VARIABLES AND THIS FLAGS A "NO KEY PRESSED" STATE. WHEN A KEY IS DETECTED THEN THIS FLAG IS TESTED AND IF ZERO THEN THE KEY IS ACCEPTED AS A FIRST KEY PRESS. IN THIS CASE THE KEY PRESS FLAG IS THEN SET TO FF TO REMEMBER THAT THE KEY PRESS HAS BEEN DETECTED. IF A KEY IS DETECTED AND THIS FLAG BYTE IS NOT ZERO, THEN THE KEY IS IGNORED. WHEN THE SOFTWARE FINDS THAT NO KEY IS BEING PRESSED, THEN THIS FLAG IS CLEARED TO ALLOW THE NEXT KEY PRESS DETECTED TO BE PROCESSED.
THIS FLAG IS USED BY THE RST 08, RST 10 RST 18 AND RST 20 KEYBOARD ROUTINES AS DESCRIBED IN ISSUE 15 TALKING ELECTRONICS AND ALSO THE STEPPER SOFTWARE.
THE AUTO KEY REPEAT ROUTINE DOES NOT USE THIS FLAG BYTE, DO NOT CONFUSE THIS FLAG WITH THE AUTO KEY STATUS BYTE WHICH IS USED BY THE AUTO KEY REPEAT SECTION.

TAPE FILE INFORMATION BLOCK
THIS IS A 12 BYTE BLOCK THAT CONTAINS THE FOLLOWING INFORMATION:
THE START ADDRESS OF THE BLOCK, THE NUMBER OF BYTES IN THE BLOCK, THE FILE NUMBER AND AN OPTIONAL GO ADDRESS OR FFFF IF OPTIONAL GO IS DISABLED. THE OTHER 4 BYTES ARE NOT USED AT THIS STAGE.
THIS BLOCK IS OUTPUTTED AND INPUTTED TO AND FROM THE TAPE ON EACH TAPE OPERATION.

"NEXT PC" BUFFER
THIS IS A TEMPORARY PLACE TO SAVE THE RETURN ADDRESS WHICH IS THEN USED AS THE ACTUAL PC VALUE FOR THE NEXT INSTRUCTION STEPPED.

FORCED HARD RESET
THIS IS ACHIEVED BY HOLDING DOWN A KEY WHEN RELEASING THE RESET. THE HARD RESET CAUSES JMON TO RE-BOOT ITS VARIABLES AND ALSO MASK OFF ALL THE USER PATCHES (EXCEPT THE RESET PATCH). THE MAIN PURPOSE OF A FORCED HARD RESET IS TO RECOVER THE TEC IF A USER PATCH ENTERS A CONTINUOUS LOOP.

AT THE START OF JMON, HL IS SAVED IN ITS SINGLE STEPPER BUFFER AND THE SOFT RESET DISPLAY VALUE IS PLACED IN THE CURRENT EDIT LOCATION BUFFER. THE ROUTINE THEN IS CONTINUED AT 006B.

```
0000 22 6E 08     LD (086E),HL  ;SAVE HL PART OF REGISTER SAVE
0003 2A 28 08     LD HL,(0828)  ;GET SOFT RESET INITIAL EDIT
0006 18 63        JR 006B       ;LOCATION AND CONTINUE AT 006B
```

RST 08 AND RST 10 (CF AND D7)
THESE TWO COMBINE TOGETHER TO SIMULATE A HALT INSTRUCTION. THIS IS DONE BY LOOPING UNTIL THE CURRENT (IF ANY) KEY PRESS IS RELEASED (RST 08), AND THEN LOOPING UNTIL A NEW KEY PRESS IS DETECTED (RST 10).

```
0008 E7           RST 20        ;TEST FOR KEY PRESS
0009 28 FD        JR Z,0008     ;LOOP IF KEY PRESSED
000B 00           NOP           ;ELSE
000C 00           NOP           ;MOVE
000D 00           NOP           ;TO
000E 00           NOP           ;NEXT
000F 00           NOP           ;RST
0010 E7           RST 20        ;TEST FOR KEY AGAIN
0011 20 FD        JR NZ,0010    ;LOOP IF KEY NOT PRESSED
0013 E6 1F        AND 1F        ;MASK OF JUNK BITS
0015 ED 47        LD I,A        ;STORE IN INTERRUPT REGISTER
0017 C9           RET           ;DONE
```

RST 18 (DF) AND RST (20)
RST 18 CALLS THE LED SCAN ROUTINE ONCE THEN MOVES ON INTO RST 20 THAT THEN CALLS A KEYBOARD READ ROUTINE.
THE KEYBOARD MUST BE READ CONTINUOUSLY OVER A PERIOD OF TIME, AS THE DATA AVAILABLE SIGNAL (BIT 6, PORT 3) (USUALLY) PULSES, WHEN A KEY IS PRESSED, IN TIME WITH THE KEY ENCODER CHIP'S SCANNING. IF THE KEY BOARD IS READ ONLY ONCE EVERY SECOND, THEN THE SOFTWARE MAY (AND PROBABLY) WILL TAKE SEVERAL SECONDS TO DETECT THE KEY.
THE NUMBER OF READ CYCLES FOR THE KEYBOARD IS LOADED INTO B.

```
0018 E5           PUSH HL       ;SAVE HL
0019 D5           PUSH DE       ;AND DE
001A CD 36 08     CALL 0836     ;CALL SCAN ROUTINE
001D D1           POP DE        ;RECOVER DE
001E E1           POP HL        ;AND HL
001F 00           NOP           ;NEXT RST
0020 C5           PUSH BC       ;SAVE BC
0021 06 20        LD B,20       ;B = NUMBER OF KEYBOARD SCAN LOOPS
0023 CD AD 06     CALL 06AD     ;CALL KEY READER/VALIDATER
0026 C1           POP BC        ;RECOVER BC
0027 C9           RET           ;DONE
```

RST 28 (EF)
START STEPPING FROM THE INSTRUCTION FOLLOWING THE RST 28

```
0028 E3           EX (SP),HL    ;GET RETURN ADDRESS FROM THE STACK
0029 22 58 08     LD (0858),HL  ;PUT IN "NEXT PC" BUFFER
002C E3           EX (SP),HL    ;FIX UP STACK
002D FB           EI            ;ENABLE INTERRUPTS
002E C9           RET           ;STEPPING WILL OCCUR AFTER RETURN
002F FF           RST 38        ;SPARE
```

RST 30 (F7)
TEST THE BUSY STATE OF THE LCD AND LOOP WHILE BUSY

```
0030 DB 04        IN A,04       ;READ STATUS BIT FROM LCD
0032 07           RLCA          ;PUT IN CARRY
0033 38 FB        JR C,0030     ;LOOP IF LCD BUSY
0035 C9           RET           ;DONE
0036 FF           RST 38        ;
0037 FF           RST 38        ;
```

RST 38 (FF)
INTERRUPT HANDLER FOR STEPPER AND BREAK-POINTS

```
0038 C3 12 03     JP 0312       ;JUMP TO STEPPER ROUTINE
003B FF           RST 38        ;UNUSED
003C FF           RST 38        ;"    "
003D FF           RST 38        ;"    "
003E FF           RST 38        ;"    "
003F FF           RST 38        ;"    "
```

```
0040 FF           RST 38        ;"   "

JUMP TABLE FOR EXTERNAL SOFTWARE TO USE JMON ROUTINES

0041 C3 DD 03     JP 03DD       ;MENU GATE
0044 C3 79 04     JP 0479       ;PERIMETER HANDLER ENTRY
0047 C3 ED 03     JP 03ED       ;SOFT MENU ENTRY
004A C3 9F 06     JP 069F       ;ERR-IN ENTRY
004D C3 B6 05     JP 05B6       ;PASS/FAIL/MENU
0050 C3 A3 04     JP 04A3       ;SOFT PERIMETER HANDLER ENTRY
0053 FF           RST 38        ;RESERVED
0054 FF           RST 38        ;"   "
0055 FF           RST 38        ;"   "
0056 FF           RST 38        ;"   "
0057 FF           RST 38        ;"   "
0058 FF           RST 38        ;"   "
0059 FF           RST 38        ;"   "
005A FF           RST 38        ;"   "
005B FF           RST 38        ;"   "
005C FF           RST 38        ;"   "
005D FF           RST 38        ;"   "

SHIFT-2 ROUTINE
THIS STORES THE CURRENT EDIT LOCATION IN THE "NEXT PC" BUFFER. THE INTERRUPTS ARE THEN
ENABLED AND THE PROGRAM JUMPS TO THE USER ROUTINE TO BE STEPPED. STEPPING OCCURS AT THE
CURRENT EDIT LOCATION (CEL).

005E 2A 2E 08     LD HL,(082E)  ;PUT CURRENT EDIT LOCATION IN
0061 22 58 08     LD (0858),HL  ;"NEXT PC" BUFFER
0064 FB           EI            ;ENABLE INTERRUPTS
0065 E9           JP (HL)       ;START STEPPING

NMI HANDLER (IMMEDIATE RETURN)

0066 ED 45        RETN          ;IGNORE NMI
0068 FF           RST 38        ;RESERVED
0069 FF           RST 38        ;FOR
006A FF           RST 38        ;A JUMP

CONTINUATION OF MONITOR

006B ED 56        IM 1          ;SET INTERRUPT MODE 1 FOR STEPPER
006D 22 2E 08     LD (082E),HL  ;STORE SOFT RESET INITIAL CEL
0070 21 76 00     LD HL,0076    ;LOAD HL WITH RE-ENTRY ADDRESS
0073 C3 18 03     JP 0318       ;JUMP TO SAVE REGISTERS


RE-ENTRY POINT AFTER SAVING REGISTERS

0076 31 20 08     LD SP,0820    ;SET STACK
0079 CD F7 02     CALL 02F7     ;CALL RESET PATCH HANDLER
007C E7           RST 20        ;LOOK FOR FORCED HARD RESET
007D 28 07        JR Z,0086     ;JUMP KEY PRESSED TO HARD RESET
007F 3A FF 08     LD A,(08FF)   ;CHECK HARD/RESET FLAG
0082 FE AA        CP AA         ;FOR AA
0084 28 1C        JR Z,00A2     ;JUMP TO SOFT RESET IF AA

HARD RESET
MONITOR DEFAULT VARIABLES ARE RE-BOOTED AND USER PATCHES MASKED OFF.

0086 21 0F 07     LD HL,070F    ;LOAD HL WITH START OF JMON DEFAULT
0089 11 20 08     LD DE,0820    ;VARIABLES ROM TABLE
008C 01 2B 00     LD BC,002B    ;DE IS THE RAM DE(stination)
008F ED B0        LDIR          ;AND BC THE COUNT: MOVE TABLE
0091 06 03        LD B,03       ;MASK OF THE THREE USER PATCHES
0093 3E C9        LD A,C9       ;BY PUTTING A RETurn AT THE FIRST
0095 12           LD (DE),A     ;LOCATION OF EACH
0096 13           INC DE        ;
0097 13           INC DE        ;
0098 13           INC DE        ;
0099 10 FA        DJNZ,0095     ;
009B CD D5 06     CALL 06D5     ;INITIALIZE/TEST FOR THE LCD
009E AF           XOR A         ;CLEAR HARD/SOFT
009F 32 FF 08     LD (08FF),A   ;RESET FLAG

THIS SECTION IS THE SOFT RESET SECTION. IT IS ALSO PART OF THE HARD RESET SECTION.
```

```
00A2 21 00 38      LD HL,3800     ;TEST FOR JMON UTILITIES ROM
00A5 7E            LD A,(HL)      ;
00A6 FE C3         CP C3          ;AND CALL ITS RESET ROUTINE
00A8 CC 00 38      CALL Z,3800    ;IF REQUIRED
00AB CD 3C 08      CALL 083C      ;CALL RESET TONE ROUTINE
00AE AF            XOR A          ;CLEAR MONITOR CONTROL BYTE
00AF 32 2B 08      LD (082B),A    ;0 = DATA MODE, NO NIBBLES ENTERED
```

EACH TIME A KEYBOARD INPUT OR USER PATCH "PLANT", IS PROCESSED, THE PROGRAM JUMPS BACK TO HERE SO THE DISPLAYS MAY BE UP-DATED.

```
00B2 2A 2E 08      LD HL,(082E)   ;GET CURRENT EDIT LOCATION (CEL)
00B5 ED 4B 2C 08   LD BC,(082C)   ;AND DISPLAY BUFFER ADDRESS
00B9 CD 30 08      CALL 0830      ;AND CONVERT CEL TO DISPLAY CODE
00BC 7E            LD A,(HL)      ;AND THEN CONVERT CONTENTS OF
00BD CD 33 08      CALL 0833      ;CEL TO DISPLAY CODE
00C0 CD 39 08      CALL 0839      ;CALL THE SET DOTS ROUTINE
00C3 CD 42 08      CALL 0842      ;CALL SCAN/KEY/LCD/PATCH ROUTINE
```

THE SECTION BELOW IS EXECUTED WHEN EITHER A KEY OR KEY "PLANT" IS DETECTED IN THE SCAN/KEY/LCD/PATCH ROUTINE ROUTINE

```
00C6 2A 2E 08      LD HL,(082E)   ;POINT HL TO CURRENT EDIT LOCATION
00C9 4F            LD C,A         ;PRESERVE INPUT KEY IN C
00CA 3A 2B 08      LD A,(082B)    ;GET MONITOR CONTROL BYTE (MCB)
00CD CB 67         BIT 4,A        ;TEST FOR ADDRESS OR FUNCTION MODE
00CF 47            LD B,A         ;STORE MCB IN B
00D0 79            LD A,C         ;GET INPUT KEY BACK IN A
00D1 20 2F         JR NZ,0102     ;JUMP IF ADDRESS OR FUNCTION MODE
00D3 FE 10         CP 10          ;TEST FOR "+"
00D5 20 0C         JR NZ,00E3     ;JUMP IF NOT TO TEST FOR "-"
```

"+" KEY HANDLER (WHEN IN DATA MODE ONLY)

```
00D7 23            INC HL         ;ADD 1 TO CURRENT EDIT LOCATION
```

COMMON CEL AND MCB UP-DATER
SEVERAL SECTIONS JUMP HERE TO STORE AN UP-DATED CEL AND CLEAR THE NIBBLE COUNTER.

```
00D8 22 2E 08      LD (082E),HL   ;STORE CEL
00DB 78            LD A,B         ;GET MCB
```

COMMON MCB UP-DATER
SOME KEY HANDLER SECTION THAT DON'T REQUIRE A NEW CEL (OR HAVE ALREADY STORED IT) JUMP HERE.

```
00DC E6 FC         AND FC         ;CLEAR NIBBLE COUNTER
00DE 32 2B 08      LD (082B),A    ;STORE MCB
00E1 18 CF         JR 00B2        ;JUMP BACK TO UPDATE DISPLAY
00E3 FE 11         CP 11          ;TEST FOR "-"
00E5 20 03         JR NZ,00EA     ;JUMP IF NOT TO TEST FOR "GO"
```

"-" KEY HANDLER (WHEN IN DATA MODE ONLY)

```
00E7 2B            DEC HL         ;DECREASE CEL ADDRESS BY ONE
00E8 18 EE         JR 00D8        ;JUMP TO COMMON CEL AND MCB UP-DATER
00EA FE 12         CP 12          ;TEST FOR GO
00EC 20 14         JR NZ,0102     ;JUMP IF NOT TO TEST FOR "AD"
```

"GO" HANDLER (WHEN IN DATA MODE ONLY)

```
00EE 3A 23 08      LD A,(0823)    ;TEST FOR ALTERNATE GO ADDRESS
00F1 FE AA         CP AA          ;IF (0823)=AA
00F3 28 05         JR Z,00FA      ;JUMP IF SET FOR ALTERNATE GO ADDR
00F5 2A 2E 08      LD HL,(082E)   ;ELSE GET CURRENT EDIT LOCATION
00F8 18 03         JR 00FD        ;SKIP ALTERNATE JUMP ADDRESS FETCH
00FA 2A 28 08      LD HL,(0828)   ;GET ALTERNATE GO ADDRESS
00FD 11 45 08      LD DE,0845     ;PUT RETURN ADDRESS ON STACK
0100 D5            PUSH DE        ;
0101 E9            JP (HL)        ;START USER EXECUTION
```

TEST HERE FOR ADDRESS KEY. IF THE KEY PRESSED IS NOT THE ADDRESS KEY, THEN A JUMP IS PERFORMED. OTHERWISE THE ADDRESS KEY IS PROCESSED.

```
0102 FE 13         CP 13          ;TEST FOR ADDRESS KEY
```

```
0104 20 0B        JR NZ,0111     ;JUMP IF NOT TO DATA KEY HANDLER
0106 78           LD A,B         ;GET MONITOR CONTROL BYTE (MCB)
0107 CB 68        BIT 5,B        ;TEST FOR FUNCTION MODE AND JUMP TO
0109 20 02        JR NZ,010D     ;CLEAR FUNCTION MODE BITS IF SO
010B EE 10        XOR 10         ;ELSE TOGGLE ADDRESS MODE BIT
010D E6 D3        AND D3         ;CLEAR ALL FUNCTION MODE BITS
010F 18 CB        JR 00DC        ;LOOP BACK TO COMMON MCB UP-DATER
```

A TEST FOR ADDRESS/FUNCTION MODE IS DONE. IF IN ADDRESS OR FUNCTION MODE A JUMP IS PERFORMED.

```
0111 78           LD A,B         ;GET MCB
0112 CB 67        BIT 4,A        ;TEST FOR ADDRESS OR FUNCTION MODE
0114 20 25        JR NZ,013B     ;JUMP IF EITHER MODE
```

A TEST FOR SHIFT IS DONE AND A JUMP IS PERFORMED IF IN THE SHIFT MODE TO THE FUNCTION/SHIFT HANDLER.

```
0116 DB 00        IN A,00        ;TEST FOR THE SHIFT KEY
0118 CB 6F        BIT 5,A        ;AND JUMP IF SHIFT IS PRESSED
011A 28 34        JR Z,0150      ;TO THE FUNCTION HANDLER
```

ANY TIME A DATA KEY IS PRESSED WHILE IN THE DATA MODE, IT IS PROCESSED STARTING HERE.

```
011C 78           LD A,B         ;GET MCB
011D E6 03        AND 03         ;MASK IT DOWN TO BYTE COUNTER
011F FE 02        CP 02          ;AND TEST FOR TWO NIBBLES ENTERED
0121 78           LD A,B         ;INPUT KEY VALUE BACK IN A
0122 20 0E        JR NZ,0132     ;JUMP IF NOT READY FOR AUTO INC
0124 F5           PUSH AF        ;SAVE MCB
0125 3A 27 08     LD A,(0827)    ;TEST AUTO INC MASK
0128 B7           OR A           ;IF NOT ZERO THEN JUMP AS USER
0129 20 04        JR NZ,012F     ;HAS SWITCHED OFF AUTO INC MODE
012B 23           INC HL         ;ELSE INCREMENT CEL BEFORE ENTERING
012C 22 2E 08     LD (082E),HL   ;NEW NIBBLE AND STORE NEW CEL
012F F1           POP AF         ;RECOVER MON CONTROL BYTE IN A
0130 E6 FC        AND FC         ;CLEAR BYTE COUNTER (BITS 0 AND 1)
0132 3C           INC A          ;ADD ONE TO NIBBLE COUNTER
0133 32 2B 08     LD (082B),A    ;STORE IT
0136 3A 20 08     LD A,(0820)    ;GET INPUT KEY FROM INPUT BUFFER
0139 18 11        JR 014C        ;JUMP TO ENTER IT
```

TEST HERE FOR A CONTROL KEY WHILE IN EITHER THE ADDRESS OR FUNCTION MODE AND JUMP TO ENCODE THE FUNCTION NUMBER BITS (2 AND 3 OF MCB). IF NOT A CONTROL KEY, THEN TEST FOR THE FUNCTION MODE AND JUMP TO FUNCTION JUMP CONTROL IF SO, ELSE SERVICE DATA KEY FOR ADDRESS MODE.

```
013B 3A 20 08     LD A,(0820)    ;GET INPUT KEY FROM INPUT BUFFER
013E CB 67        BIT 4,A        ;TEST FOR CONTROL KEY (+,- OR GO)
0140 20 2F        JR NZ,0171     ;JUMP IF CONTROL TO FUNCTION ENCODER
0142 CB 68        BIT 5,B        ;TEST FUNCTION MODE
0144 20 0A        JR NZ,0150     ;JUMP IF SO TO FUNCTION  JUMP CONTROL
```

DATA KEY PRESS WHILE IN THE ADDRESS MODE

```
0146 21 2E 08     LD HL,082E     ;POINT HL TO CEL BUFFER
0149 ED 6F        RLD            ;AND SHIFT IN THE NEW NIBBLE
014B 23           INC HL         ;AND MOVE THE OTHERS ACROSS
014C ED 6F        RLD            ;THIS RLD USED BY DATA MODE ALSO
014E 18 91        JR 00E1        ;JUMP (VIA A JUMP) TO UP-DATE DISPLAYS
```

FUNCTION AND SHIFT JUMP CONTROL
BITS 2 AND 3 OF THE MONITOR CONTROL BYTE (MCB) ARE THE FUNCTION IDENTIFIER BITS.
IF BOTH ARE ZERO THEN EITHER FUNCTION 1 IS SELECTED OR NO FUNCTION IS SELECTED. BECAUSE THIS IS THE ALSO THE NO FUNCTION MODE ENABLED STATE, THE SHIFT KEY, WHICH DOES NOT AFFECT THE MONITOR CONTROL BYTE, WILL ALSO WILL INVOKE FUNCTION 1. (THEREFORE THIS ROUTINE DOES NOT NEED TO TEST FOR THE SHIFT KEY).
IF BIT 2 IS HIGH THEN FUNCTION 2 IS SELECTED AND IF BIT 3 IS HIGH THEN FUNCTION 3 IS SELECTED.
DURING THIS ROUTINE, HL IS LOADED TO THE BASE OF THE REQUIRED JUMP TABLE MINUS TWO BYTES (ONE ENTRY). THIS IS BECAUSE THE OFFSET PROVIDED FROM THE KEYBOARD HAS BEEN INCREMENTED BY ONE. THIS SAVES TESTING FOR ZERO INPUT WHICH WOULD NOT ALLOW THE TABLE ACCESSING TO WORK CORRECTLY. THE REQUIRED BASE IS FOUND BY EXAMINING THE STATE OF THE BITS 2 AND 3 OF THE MONITOR CONTROL BYTE (MCB) AND LOADING HL ACCORDINGLY.
AS EACH ENTRY IS TWO BYTES LONG, THE TABLE POINTER (THE VALUE INSIDE HL), IS INCREMENTED TWICE FOR EACH DECREMENT OF THE INPUT VALUE (FROM THE KEYBOARD). WHEN THE REQUIRED TABLE

ENTRY IS FOUND, IT IS PUT INSIDE HL (VIA DE) AND THE ROUTINE JUMPS TO PART OF THE "GO" KEY ROUTINE TO CREATE A RETURN ADDRESS ON THE STACK AND EXECUTE THE SELECTED ROUTINE.

```
0150 78          LD A,B         ;PUT MONITOR CONTROL BYTE IN A
0151 E6 0C       AND 0C         ;MASK IT DOWN TO FUNCTION BITS
0153 21 DE 07    LD HL,07DE     ;JMON FUNCTION JUMP TABLE BASE -2
0156 28 0A       JR Z,0162      ;JUMP IF FUNCTION 1 OR SHIFT
0158 21 BE 08    LD HL,08BE     ;LOAD HL WITH USER TABLE -2
015B FE 04       CP 04          ;TEST FOR FUNCTION 2
015D 28 03       JR Z,0162      ;JUMP IF FUNCTION 2 (USER FUNCTION)
015F 21 1E 38    LD HL,381E     ;OTHERWISE MUST BE FUNCTION 3
0162 3A 20 08    LD A,(0820)    ;GET INPUT KEY FROM INPUT BUFFER
0165 3C          INC A          ;ADD ONE IN CASE IT WAS ZERO
0166 47          LD B,A         ;PUT IN B TO USE AS A LOOP COUNTER
0167 23          INC HL         ;LOOK THROUGH TABLE
0168 23          INC HL         ;FOR RIGHT JUMP VECTOR
0169 10 FC       DJNZ,0167      ;
016B 5E          LD E,(HL)      ;PUT IT IN HL
016C 23          INC HL         ;VIA DE
016D 56          LD D,(HL)      ;
016E EB          EX DE,HL       ;JUMP TO CREATE RETURN ADDRESS AND
016F 18 8C       JR 00FD        ;EXECUTE SELECTED ROUTINE
```

FUNCTION NUMBER ENCODER
THIS SECTION ENCODES THE FUNCTION IDENTIFIER BITS (BITS 2 AND 3) IN THE MONITOR CONTROL BYTE (BITS 2 AND 3) THEN SETS THE FUNCTION ENABLE BIT (BIT 5).
THE FUNCTION IDENTIFIER BITS ARE DERIVED FROM THE LEAST TWO SIGNIFICANT BITS OF THE INPUT CONTROL KEY (+, -, AND GO). THESE ARE SHIFTED LEFT TWICE TO ALIGN THEM TO THE FUNCTION SELECT BITS (BITS 2 AND 3) IN THE MCB. THE INPUT CONTROL KEY IS IN THE ACCUMULATOR ON ENTRY AND THE MONITOR CONTROL BYTE (MCB) IN B.

```
0171 E6 03       AND 03         ;MASK DOWN CONTROL KEY
0173 07          RLCA           ;SHIFT IT LEFT TWICE TO ALIGN BITS 0
0174 07          RLCA           ;AND 1 TO FUNCTION IDENTITY BITS IN MCB
0175 F6 20       OR 20          ;SET FUNCTION MODE ENABLED FLAG
0177 4F          LD C,A         ;SAVE IN C
0178 78          LD A,B         ;GET CURRENT MCB
0179 E6 D3       AND D3         ;CLEAR ANY PREVIOUS FUNCTION BITS
017B B1          OR C           ;MERGE TOGETHER
017C 32 2B 08    LD (082B),A    ;STORE MCB
017F 18 CD       JR 014E        ;JUMP VIA JUMPS TO UP-DATE DISPLAYS
```

THIS IS THE SCAN/KEY/LCD/PATCH ROUTINE. THIS ROUTINE LOOPS SCANNING THE LED DISPLAY AND SERVICING THE "DURING LOOP" USER PATCH UNTIL A KEY PRESS IS VALIDATED BY THE AUTO-KEY REPEAT SECTION. THE INPUT KEY IS RETURNED IN THE ACCUMULATOR AND IN THE INPUT BUFFER AT 0820 WITH THE ZERO FLAG SET AND CARRY CLEARED.
THREE PATCHES ARE SUPPORTED IN THIS ROUTINE. THEY ARE A PATCH BEFORE LOOP, A PATCH DURING THE LOOP AND A PATCH AFTER A VALID KEY PRESS.
THE "PLANT" IS A VALUE INSERTED INTO THE INPUT BUFFER (0820) BY THE DURING LOOP PATCH. THE "PLANT" VALUE IS IDENTIFIED BY BIT 7 OF THE INPUT BUFFER BEING SET. BIT 7 IS RESET BEFORE RETURNING TO SERVICE THE PLANT.
THIS ROUTINE USES A BYTE AT 082A, CALLED THE AUTO KEY STATUS BYTE AS A FLAG AND TIMER TO GENERATE THE AUTO REPEAT DELAY.

```
0181 CD 48 08    CALL 0848      ;CALL LCD ROUTINES
0184 CD 4B 08    CALL 084B      ;CALL PRE-SCAN USER PATCH
0187 CD 36 08    CALL 0836      ;CALL SCAN
018A CD 4E 08    CALL 084E      ;CALL USER "DURING LOOP" PATCH
018D 21 20 08    LD HL,0820     ;TEST KEY INPUT BUFFER BIT 7 FOR A
0190 CB 7E       BIT 7,(HL)     ;"PLANT" INSERTED BY USER DURING
0192 CB BE       RES 7,(HL)     ;PATCH: RESET BIT 7 RETURN TO
0194 C0          RET NZ         ;SERVICE "PLANT" IF BIT 7 NOT ZERO
0195 E7          RST 20         ;TEST FOR KEY PRESS VIA RST 20
0196 21 2A 08    LD HL,082A     ;SET HL TO POINT TO AUTO KEY STATUS
0199 38 04       JR C,019F      ;JUMP IF A KEY IS PRESSED
019B 36 80       LD (HL),80     ;ELSE SET AUTO KEY STATUS TO
019D 18 E8       JR 0187        ;NO KEY STATE AND CONTINUE LOOP
019F CD CA 06    CALL 06CA      ;CALL UNIVERSAL KEY INPUTTER
01A2 CB 7E       BIT 7,(HL)     ;TEST AUTO KEY STATUS FOR FIRST KEY
01A4 20 10       JR NZ,01B6     ;JUMP IF SO TO SET LONG KEY DELAY
01A6 35          DEC (HL)       ;ELSE COUNT DOWN KEY DELAY
01A7 20 DE       JR NZ,0187     ;LOOP IF NOT READY FOR KEY REPEAT
01A9 36 0C       LD (HL),0C     ;ELSE SET SHORT TIME DELAY BETWEEN
01AB CD 51 08    CALL 0851      ;KEYS: CALL USER "AFTER KEY" PATCH
01AE CD 3F 08    CALL 083F      ;CALL KEY TONE
01B1 AF          XOR A          ;SET ZERO FLAG AND CLEAR CARRY
```

```
01B2 3A 20 08      LD A,(0820)    ;PUT INPUT KEY IN A
01B5 C9            RET            ;AND RETURN FOR KEY SERVICE
01B6 36 70         LD (HL),70     ;SET KEY TIMER FOR LONG DELAY
01B8 18 F1         JR 01AB        ;JUMP TO SERVICE PATCH, TONE ETC.
```

THIS IS THE LED SCAN ROUTINE.

```
01BA 06 20         LD B,20        ;B IS THE SCAN BIT
01BC 2A 2C 08      LD HL,(082C)   ;GET ADDRESS OF DISPLAY BUFFER
01BF 7E            LD A,(HL)      ;GET FIRST BYTE
01C0 D3 02         OUT (02),A     ;AND OUTPUT IT TO SEGMENTS
01C2 78            LD A,B         ;GET SCAN BIT
01C3 D3 01         OUT (01),A     ;OUTPUT IT TO COMMONS
01C5 06 40         LD B,40        ;CREATE SHORT
01C7 10 FE         DJNZ,01C7      ;DELAY IN B
01C9 23            INC HL         ;INCREASE HL TO NEXT DISPLAY BYTE
01CA 47            LD B,A         ;GET SCAN BIT BACK IN B
01CB AF            XOR A          ;CLEAR THE LAST PORT OUTPUTTED TO
01CC D3 01         OUT (01),A     ;TO PREVENT "GHOSTING"
01CE CB 08         RRC B          ;SHIFT SCAN BIT ACROSS TO NEXT
01D0 30 ED         JR NC,01BF     ;COMMON: WHEN SCAN BIT FALLS INTO
01D2 D3 02         OUT (02),A     ;CARRY SCAN IS TERMINATED: CLEAR
01D4 C9            RET            ;PORT 2 AND RETURN
```

THIS ROUTINE CONVERTS HL TO DISPLAY CODE AND STORE THE DISPLAY CODE IN A BUFFER POINTED TO BY BC.

```
01D5 7C            LD A,H         ;PUT H IN A
01D6 CD 33 08      CALL 0833      ;CONVERT A TO DISPLAY CODE
01D9 7D            LD A,L         ;NOW DO FOR L
```

THIS SECTION CONVERTS THE BYTE IN A TO TWO DISPLAY BYTES.

```
01DA F5            PUSH AF        ;SAVE A
01DB 07            RLCA           ;SHIFT MSN TO LSN PLACE
01DC 07            RLCA           ;FOR NIBBLE AT A TIME CONVERSION
01DD 07            RLCA           ;
01DE 07            RLCA           ;
01DF CD E3 01      CALL 01E3      ;CONVERT FIRST NIBBLE
01E2 F1            POP AF         ;RECOVER A TO CONVERT SECOND NIBBLE
01E3 E6 0F         AND 0F         ;MASK OF HIGH NIBBLE
01E5 11 D0 07      LD DE,07D0     ;SET DE TO BASE OF CONVERSION
01E8 83            ADD A,E        ;TABLE: ADD A TO BASE
01E9 5F            LD E,A         ;UPDATE POINTER
01EA 1A            LD A,(DE)      ;GET DISPLAY CODE
01EB 02            LD (BC),A      ;STORE IN DISPLAY BUFFER
01EC 03            INC BC         ;INCREMENT DISPLAY BUFFER POINTER
01ED C9            RET            ;NIBBLE CONVERSION DONE
```

SET DOTS
THIS ROUTINE SETS THE DOTS IN THE DISPLAY BUFFER. IF IN ADDRESS MODE THEN 4 DOTS ARE SET IN THE ADDRESS DISPLAY BUFFER, IF IN A FUNCTION MODE, THEN ONE DOT IN THE ADDRESS DISPLAY - RIGHT MOST FOR FUNCTION 1 SECOND RIGHT FOR FUNCTION 2 AND THIRD RIGHT FOR FUNCTION 3. IF IN THE DATA MODE THEN 2 DOTS IN THE DATA DISPLAY BUFFER OR ONE DOT, ON THE RIGHTMOST DISPLAY, IF TWO NIBBLES HAVE BEEN ENTERED AND IN THE AUTO-INCREMENT MODE.

```
01EE 06 02         LD B,02        ;SET B FOR 2 DOTS
01F0 2A 2C 08      LD HL,(082C)   ;PUT DISPLAY BUFFER IN HL
01F3 3A 2B 08      LD A,(082B)    ;GET MONITOR CONTROL BYTE (MCB)
01F6 CB 67         BIT 4,A        ;TEST FOR ADDRESS OR FUNCTION MODE
01F8 28 1A         JR Z,0214      ;JUMP IF NOT TO DO DATA DOTS
01FA CB 6F         BIT 5,A        ;TEST ONLY FOR FUNCTION MODE
01FC 20 08         JR NZ,0206     ;JUMP IF FUNCTION MODE
01FE 06 04         LD B,04        ;ADDRESS MODE SO SET B FOR 4 DOTS
0200 CB E6         SET 4,(HL)     ;SET DOT IN DISPLAY BUFFER
0202 23            INC HL         ;NEXT LOCATION
0203 10 FB         DJNZ,0200      ;DO 4 TIMES
0205 C9            RET            ;DONE
0206 05            DEC B          ;FUNCTION MODE: SET B FOR ONE DOT
0207 CB 5F         BIT 3,A        ;TEST FOR FUNCTION 3
0209 20 06         JR NZ,0211     ;JUMP IF FUNCTION 3 TO ADD HL+1
020B CB 57         BIT 2,A        ;TEST FOR FUNCTION 2
020D 20 01         JR NZ,0210     ;JUMP IF FUNCTION 2 TO ADD HL+2
020F 23            INC HL         ;INCREMENT HL TO POINT TO THE
0210 23            INC HL         ;REQUIRED DISPLAY BYTE
0211 23            INC HL         ;
```

```
0212 18 EC        JR 0200        ;JUMP TO SET DOT
0214 23           INC HL         ;DATA MODE: HL NOW POINTS TO SECOND
0215 4F           LD C,A         ;LEFT MOST DISPLAY BUFFER: SAVE MCB
0216 3A 27 08     LD A,(0827)    ;IN C: TEST AUTO INCREMENT ENABLE
0219 B7           OR A           ;FLAG
021A 20 F3        JR NZ,020F     ;JUMP IF NO AUTO INCREMENT TO SET BOTH
021C CB 49        BIT 1,C        ;DATA DOTS: TEST BYTE COUNTER FOR 2
021E 28 EF        JR Z,020F      ;NIBBLES: JUMP IF NOT TO SET BOTH DATA
0220 23           INC HL         ;DOTS: ELSE SKIP DOT ON ONE DISPLAY
0221 05           DEC B          ;AND DECREASE DOT COUNT FROM 2 TO 1
0222 18 EB        JR 020F        ;JUMP TO ADJUST HL AND SET DOTS

MASKABLE RESET TONE ROUTINE
IF 0822 IS NOT ZERO THEN NO TONE

0224 CD 3F 08     CALL 083F      ;CALL TONE

MASKABLE TONE ROUTINE

0227 3A 22 08     LD A,(0822)    ;TEST SOUND MASK
022A B7           OR A           ;
022B C0           RET NZ         ;NO TONE IF NOT ZERO
022C 0E 40        LD C,40        ;LOAD C WITH PERIOD
022E 2E 31        LD L,31        ;LOAD L WITH NUMBER OF CYCLES
0230 AF           XOR A          ;CLEAR A
0231 D3 01        OUT (01),A     ;OUT TO SPEAKER
0233 41           LD B,C         ;
0234 10 FE        DJNZ,0234      ;DELAY FOR PERIOD
0236 EE 80        XOR 80         ;TOGGLE SPEAKER BIT
0238 2D           DEC L          ;DECREMENT CYCLE COUNT
0239 20 F6        JR NZ,0231     ;LOOP UNTIL ZERO
023B C9           RET            ;DONE

LCD ROUTINE
IF 0821 IS NOT ZERO, THEN LCD HAS BEEN MASKED OFF BY EITHER THE USER OR THE LCD
INTIALIZER/TESTER ROUTINE AND NO ACTION IS TAKEN ON THE LCD. THE RST 30 (F7) IS USED
EXTENSIVELY TO TEST AND WAIT FOR THE LCD BUSY FLAG. THROUGHOUT THESE NOTES, THE INVISIBLE
INTERNAL CURSOR ON THE LCD IS REFERRED TO AS THE CURSOR, WHILE THE ">" ON THE LCD IS
REFERRED TO AS THE PROMPT.

023C 3A 21 08     LD A,(0821)    ;TEST LCD MASK
023F B7           OR A           ;
0240 C0           RET NZ         ;NOT ZERO = LCD NOT REQUIRED OR FITTED
0241 3E 80        LD A,80        ;SET LCD CURSOR TO HOME
0243 D3 04        OUT (04),A     ;
0245 F7           RST 30         ;WAIT UNTIL LCD READY
0246 CD 53 02     CALL 0253      ;CALL SET-UP AND OUTPUT FIRST LINE
0249 3E C0        LD A,C0        ;SET CURSOR TO BOTTOM LINE
024B D3 04        OUT (04),A     ;
024D F7           RST 30         ;WAIT
024E CD 5A 02     CALL 025A      ;CALL ROUTINE TO OUTPUT BOTTOM LINE
0251 18 33        JR 0286        ;JUMP TO PROMPT ROUTINE

SET-UP
MODIFY CURRENT EDIT LOCATION ADDRESS IN HL SO THAT IT POINTS TO A BYTE AT AN ADDRESS
ENDING IN EITHER 0 OR 8.

0253 2A 2E 08     LD HL,(082E)   ;GET CEL AND PUT LOW BYTE IN A
0256 7D           LD A,L         ;THEN MASK OFF THE 3 LOWEST BITS
0257 E6 F8        AND F8         ;AS THE ADDR OF THE FIRST BYTE ON
0259 6F           LD L,A         ;THE LCD WILL END WITH 0 OR 8

OUTPUT A LINE

025A CD 6C 02     CALL 026C      ;CALL "HL TO ASCII OUTPUT"
025D 06 04        LD B,04        ;SET B FOR 4 BYTES ON A LINE
025F 3E 20        LD A,20        ;LOAD A WITH ASCII SPACE
0261 D3 84        OUT (84),A     ;CHARATER AND OUTPUT IT
0263 F7           RST 30         ;WAIT
0264 7E           LD A,(HL)      ;GET BYTE TO DISPLAY
0265 CD 71 02     CALL 0271      ;CONVERT AND OUTPUT IT
0268 23           INC HL         ;POINT TO NEXT BYTE
0269 10 F4        DJNZ,025F      ;DO FOR 4 BYTES
026B C9           RET            ;DONE

CONVERT HL TO ASCII (VIA CONVERT A) AND OUTPUT IT
```

```
026C 7C           LD A,H         ;CONVERT AND
026D CD 71 02     CALL 0271      ;OUTPUT H
0270 7D           LD A,L         ;THEN L

CONVERT A TO ASCII AND OUTPUT IT

0271 F5           PUSH AF        ;SAVE A FOR SECOND NIBBLE
0272 0F           RRCA           ;SHIFT HIGH NIBBLE ACROSS
0273 0F           RRCA           ;
0274 0F           RRCA           ;
0275 0F           RRCA           ;
0276 CD 7A 02     CALL 027A      ;CALL NIBBLE CONVERTER
0279 F1           POP AF         ;RECOVER LOW NIBBLE
027A E6 0F        AND 0F         ;MASK OFF HIGH NIBBLE
027C C6 90        ADD A,90       ;CONVERT TO
027E 27           DAA            ;ASCII
027F CE 40        ADC A,40       ;USING THIS
0281 27           DAA            ;AMAZING ROUTINE
0282 D3 84        OUT (84),A     ;OUTPUT IT
0284 F7           RST 30         ;WAIT
0285 C9           RET            ;DONE

LCD PROMPT AND MODE WORD OUTPUT
THE 3 LOWEST BITS OF THE CURRENT EDIT LOCATION (CEL) ARE USED AS A DISPLACEMENT WHICH IS
ADDED TO A TABLE BASE. THE TABLE ENTRIES ARE THE LCD ADDRESSES OF THE PROMPT LOCATIONS.
IF THE AUTO INCREMENT MODE IS ON AND 2 NIBBLES HAVE BEEN ENTERED, THE DISPLACEMENT IS
INCREMENTED SO THAT THE NEXT PROMPT ADDRESS TABLE ENTRY WILL BE ACCESSED TO MOVE THE
PROMPT TO ITS NEXT SCREEN LOCATION. THE TABLE IS 9 ENTRIES LONG. 8 ARE FOR THE SPACES
BETWEEN THE DATA BYTES AND THE NINTH IS TO PARK THE PROMPT AT THE TOP LEFT-HAND CORNER
WHEN A SCREEN CHANGE IS DUE

0286 3A 2E 08     LD A,(082E)    ;GET LOW BYTE OF CEL
0289 E6 07        AND 07         ;MASK IT DOWN TO THE 3 LOWEST BITS
028B 4F           LD C,A         ;SAVE IN C
028C 3A 27 08     LD A,(0827)    ;TEST FOR AUTO INCREMENT MODE
028F B7           OR A           ;0=ON
0290 3A 2B 08     LD A,(082B)    ;GET MCB
0293 57           LD D,A         ;PUT MCB IN D
0294 20 05        JR NZ,029B     ;JUMP IF AUTO INCREMENT MODE OFF
0296 CB 4F        BIT 1,A        ;TEST FOR 2 NIBBLES ENTERED: JUMP
0298 28 01        JR Z,029B      ;IF NOT: ELSE INCREMENT
029A 0C           INC C          ;DISPLACEMENT TO ADVANCE TO
029B 79           LD A,C         ;NEXT PROMPT LOCATION ADDRESS
029C 21 BD 07     LD HL,07BD     ;LOAD HL WITH BASE OF PROMPT
029F 85           ADD A,L        ;TABLE AND ADD DISPLACEMENT
02A0 6F           LD L,A         ;PUT LOW BYTE OF TABLE ADDRESS
02A1 7E           LD A,(HL)      ;IN L AND GET PROMPT ADDRESS IN A
02A2 D3 04        OUT (04),A     ;AND OUTPUT PROMPT ADDRESS TO LCD
02A4 F7           RST 30         ;WAIT
02A5 3E 3E        LD A,3E        ;LOAD A WITH ASCII FOR ">"
02A7 D3 84        OUT (84),A     ;OUTPUT PROMPT
02A9 F7           RST 30         ;WAIT

OUTPUT MODE WORD TO BOTTOM LEFT CORNER OF THE LCD.
IF THE MODE IS EITHER DATA OR ADDR, THEN THE FOUR ASCII BYTES ARE OUTPUTTED. IF IN THE
FUNCTION MODE, THEN ONLY THREE BYTES FROM THE TABLE ARE OUTPUTTED AND THEN THE FUNCTION
NUMBER IS CALCULATED AND OUTPUTTED.
NOTICE THAT FROM THE TABLE BASE THE FIRST ENTRY (DATA) HAS A ZERO DISPLACEMENT WHILE THE
SECOND (ADDR) HAS A DISPLACEMENT OF 4 AND THE THIRD (Fs-) HAS A DISPLACEMENT OF 12.
IF YOU LOOK AT THE TABLE AT 07AD, YOU WILL SEE THAT IT IS  STAGGERED WITH THE THIRD ENTRY
12 BYTES AWAY FROM THE BASE.

02AA 3E C0        LD A,C0        ;SET CURSOR TO BOTTOM LINE
02AC D3 04        OUT (04),A     ;OUTPUT
02AE F7           RST 30         ;AND WAIT
02AF 7A           LD A,D         ;PUT MONITOR CONTROL BYTE (MCB) IN A
02B0 0F           RRCA           ;SHIFT MODE BITS TO BITS 2 AND 3
02B1 0F           RRCA           ;TO USE AS TABLE DISPLACEMENT
02B2 57           LD D,A         ;SAVE IN D AND MASK OFF ALL BITS
02B3 E6 0C        AND 0C         ;EXCEPT THE 2 THAT FLAG BETWEEN DATA,
02B5 21 AD 07     LD HL,07AD     ;ADDR AND FUNCTION: A=0 IF DATA, 4 IF
02B8 85           ADD A,L        ;ADDR, 12 IF FUNCTION, NOTE THAT TABLE
02B9 6F           LD L,A         ;IS STAGGERED (SEE 07AD): ADD A TO BASE
02BA FE B9        CP B9          ;IF A=B9 THEN MODE IS FUNCTION MODE
02BC 01 84 04     LD BC,0484     ;LOAD C WITH PORT, B WITH BYTE COUNT
```

```
02BF 28 06        JR Z,02C7      ;JUMP IF FUNCTION MODE TO OUT 3 BYTES
02C1 ED A3        OUTI           ;OUT (HL) TO (C) B=B-1
02C3 F7           RST 30         ;HL=HL+1: WAIT FOR LCD BUSY FLAG
02C4 20 FB        JR NZ,02C1     ;LOOP UNTIL B=0
02C6 C9           RET            ;DONE
02C7 06 03        LD B,03        ;ONLY THREE BYTES FOR FUNCTION MODE
02C9 CD C1 02     CALL 02C1      ;CALL THE OUTPUT ROUTINE ABOVE
02CC 7A           LD A,D         ;PUT MCB (SHIFTED RIGHT TWICE) IN A
02CD E6 03        AND 03         ;MASK IT DOWN TO GET JUST THE FUNCTION
02CF C6 31        ADD A,31       ;NUMBER BITS: ADD ASCII "1"
02D1 18 AF        JR 0282        ;JUMP TO OUTPUT FUNCTION NUMBER
```

-END OF MONITOR ROUTINES- (EXCEPT KEYBOARD READER AT 06AD)

LCD PROMPT MOVING ROUTINES. (SHIFT AND FUNCTION 1)
THESE ROUTINES ALTER THE CURRENT EDIT LOCATION ADDRESS AND STORE IT IN ITS BUFFER. WHEN THE RETURN IS DONE, JMON IS RE-ENTERED AT 00B2 (VIA THE SOFT RE-ENTRY JUMP AT 0845, THE ADDRESS OF WHICH HAS BEEN PLACED ON THE STACK BY PART OF THE "GO" ROUTINE).

```
02D3 11 04 00     LD DE,0004     ;DE= +4
02D6 2A 2E 08     LD HL,(082E)   ;PUT CEL IN HL
02D9 19           ADD HL,DE      ;ADD TO GET NEW CEL
02DA 22 2E 08     LD (082E),HL   ;STORE IN CEL BUFFER
02DD C9           RET            ;DONE
02DE 11 FC FF     LD DE,FFFC     ;DE= -4
02E1 18 F3        JR 02D6        ;JUMP TO ADD
02E3 11 FF FF     LD DE,FFFF     ;DE= -1
02E6 18 EE        JR 02D6        ;JUMP TO ADD
02E8 11 01 00     LD DE,0001     ;DE= +1
02EB 18 E9        JR 02D6        ;JUMP TO ADD
02ED 11 08 00     LD DE,0008     ;DE+ +8
02F0 18 E4        JR 02D6        ;JUMP TO ADD
02F2 11 F8 FF     LD DE,FFF8     ;DE= -8
02F5 18 DF        JR 02D6        ;JUMP TO ADD
```

RESET PATCH CHECKER.
TESTS FOR PATCH REQUIREMENT AND UP TO THE FIRST 256 BYTES OF THE PATCH ROUTINE. THE CHECKSUM FEATURE ENSURES A WAY TO CHECK THAT THE PATCH OR PATCH VARIABLES HAVE NOT BEEN CORRUPTED BY A SYSTEM CRASH, OTHERWISE YOU MAY NEVER REGAIN CONTROL OF THE COMPUTER UNLESS YOU TURN IT OFF, (AND LOSE THE CONTENTS OF YOUR MEMORY - YOU CANNOT RECOVER IT BY A FORCED HARD RESET AS THE USER PATCH IS EXECUTED BEFORE THE FORCED HARD RESET TEST). (A FORCED HARD RESET IS WHEN A KEY IS HELD DOWN WHEN THE RESET KEY IS RELEASED).
IF YOU HAVE A NON VOLATILE MEMORY AT 0800 THE SITUATION WOULD BE ABSOLUTELY HOPELESS WITHOUT THIS CHECKER ROUTINE.
A VARIABLE CAN BE PASSED TO YOUR PATCH ROUTINE IN THE "C" REGISTER. TO DO THIS THE VARIABLE IS PLACED AT ADDRESS LOCATION 08B3.

```
02F7 3A B0 08     LD A,(08B0)    ;TEST FOR RESET PATCH REQUIRED
02FA FE AA        CP AA          ;
02FC C0           RET NZ         ;RETURN IF NOT
02FD ED 4B B3 08  LD BC,(08B3)   ;PUT NO OF BYTES IN B VARIABLE IN C
0301 2A B1 08     LD HL,(08B1)   ;START IN HL
0304 AF           XOR A          ;CLEAR A
0305 86           ADD A,(HL)     ;ADD CHECKSUM
0306 23           INC HL         ;
0307 10 FC        DJNZ,0305      ;UNTIL B=0
0309 21 B5 08     LD HL,08B5     ;POINT TO REQUIRED CHECKSUM
030C BE           CP (HL)        ;TEST FOR EQUAL
030D C0           RET NZ         ;ABORT IF NOT
030E 2A B6 08     LD HL,(08B6)   ;ELSE GET START ADDR
0311 E9           JP (HL)        ;AND DO RESET PATCH
```

STEPPER ROUTINE
THE STEPPER ROUTINE IS BROKEN UP INTO SEVERAL SECTIONS. THE FIRST IS THE REGISTER SAVE, WHERE ALL THE Z80 USER REGISTERS ARE STORED IN MEMORY.

```
0312 22 70 08     LD (0870),HL   ;STORE HL IN ITS REGISTER STACK SPOT
0315 21 44 03     LD HL,0344     ;LOAD HL WITH RETURN ADDRESS
```

MONITOR JUMPS TO HERE ON RESET TO PRESERVE USER REGISTERS.

```
0318 22 60 08     LD (0860),HL   ;STORE RE-ENTRY ADDRESS IN BUFFER
031B 2A 58 08     LD HL,(0858)   ;GET ADDRESS OF INSTRUCTION JUST
031E 22 68 08     LD (0868),HL   ;STEPPED AND PUT IT IN "NEXT PC"
0321 ED 73 7E 08  LD (087E),SP   ;BUFFER: SAVE STACK POINTER VALUE
0325 E1           POP HL         ;GET RETURN ADDR, THIS IS THE ADDRESS
```

```
0326 22 58 08      LD (0858),HL  ;OF NEXT BYTE TO STEP: STORE IN
0329 31 7E 08      LD SP,087E    ;"NEXT PC" BUFFER: LOAD REGISTER DUMP
032C 08            EX AF,AF'     ;STACK: PUSH ALTERNATE REGISTERS
032D D9            EXX           ;FIRST
032E E5            PUSH HL       ;SAVE ALL REGISTERS
032F D5            PUSH DE       ;
0330 C5            PUSH BC       ;
0331 F5            PUSH AF       ;
0332 FD E5         PUSH IY       ;
0334 DD E5         PUSH IX       ;
0336 08            EX AF,AF'     ;
0337 D9            EXX           ;
0338 3B            DEC SP        ;
0339 3B            DEC SP        ;
033A D5            PUSH DE       ;
033B C5            PUSH BC       ;
033C F5            PUSH AF       ;
033D 2A 60 08      LD HL,(0860)  ;RE-ENTER CALLING ROUTINE VIA
0340 E9            JP (HL)       ;THE ADDRESS IT SUPPLIED AT 0860
0341 31 6A 08      LD SP,086A    ;SHIFT 7 ROUTINE START (REG DISPLAY)
```

THE REGISTERS HAVE BEEN SAVED. NOW THE DISPLAY AND KEYBOARD HANDLER IS SET UP. THE STACK IS DECREMENTED BY TWO TO POINT TO THE "PC" BUFFER. THE ADDRESS IN THE "PC" BUFFER IS THE ADDRESS OF THE INSTRUCTION JUST STEPPED.
THE NUMBER OF THE FIRST REGISTER (1 FOR "PC") IS PUT INTO THE CURRENT REGISTER NUMBER BUFFER.

```
0344 21 06 08      LD HL,0806    ;CREATE NEW DISPLAY BUFFER
0347 22 2C 08      LD (082C),HL  ;
034A 3B            DEC SP        ;DECREASE SP BY 2 TO POINT TO THE
034B 3B            DEC SP        ;"PC" BUFFER
```

WHEN UP-DATING THE DISPLAY, THE ROUTINE MAY JUMP BACK TO HERE IF THE FIRST DISPLAY IS REQUIRED.

```
034C 3E 01         LD A,01       ;SET UP FOR THE FIRST REGISTER (PC)
034E 32 5A 08      LD (085A),A   ;DISPLAY
```

OR HERE IF IT HAS ALTERED THE CURRENT REGISTER NUMBER IN ITS STORAGE LOCATION (085A).

```
0351 3A 5A 08      LD A,(085A)   ;DISPLAY LOOP STARTS HERE
```

HL IS LOADED WITH THE STACK POINTER VALUE, (WHICH POINTS TO THE "PC" BUFFER), MINUS TWO. THE TWO IS SUBTRACTED BECAUSE AN EXTRA TWO WILL BE ADDED TO HL DURING THE REGISTER BUFFER CALCULATOR (IMMEDIATELY BELOW) AS THE NUMBER OF THE FIRST REGISTER IS 1 AND NOT ZERO.

```
0354 21 FE FF      LD HL,FFFE    ;HL=-2
0357 39            ADD HL,SP     ;HL=SP-2
0358 23            INC HL        ;INCREMENT HL TO POINT TO THE
0359 23            INC HL        ;CURRENT REGISTER BUFFER
035A 3D            DEC A         ;INDICATED BY THE NUMBER IN A
035B 20 FB         JR NZ,0358    ;
```

HL NOW POINTS TO THE CURRENT REGISTER BUFFER. THIS SECTION PUTS THE REGISTER(S) CONTENT(S) INTO HL AND CONVERTS IT TO DISPLAY CODE AND STORE THE DISPLAY CODE IN THE DISPLAY BUFFER.

```
035D 7E            LD A,(HL)     ;GET 16 BIT VALUE
035E 23            INC HL        ;AND PUT IT
035F 66            LD H,(HL)     ;BACK INTO
0360 6F            LD L,A        ;HL
0361 ED 4B 2C 08   LD BC,(082C)  ;PUT DISPLAY BUFFER ADDRESS IN BC
0365 CD 30 08      CALL 0830     ;CALL HL TO DISPLAY CODE ROUTINE
```

THIS SECTION CALCULATES THE ADDRESS OF THE REGISTER NAME FOR THE DATA DISPLAYS. THESE ARE STORED IN A TABLE. THE REQUIRED REGISTER NAME IS THEN TRANSFERRED TO THE DISPLAY BUFFER.

```
0368 3A 5A 08      LD A,(085A)   ;GET REGISTER NUMBER
036B C5            PUSH BC       ;PUT NEXT DISPLAY BUFFER
036C D1            POP DE        ;LOCATION INTO DE(stination)
036D 01 02 00      LD BC,0002    ;BC IS THE NUMBER OF DATA DISPLAYS
0370 21 92 07      LD HL,0792    ;HL=THE BASE OF THE NAME TABLE
0373 09            ADD HL,BC     ;ADD TO HL 2 FOR EACH
0374 3D            DEC A         ;REGISTER NUMBER TO ACCESS THE
0375 20 FC         JR NZ,0373    ;CURRENT REGISTER NAME
0377 ED B0         LDIR          ;MOVE REGISTER NAME INTO RAM
```

THE SCAN AND KEYBOARD ROUTINE ARE NOW CALLED (VIA THE RST 18). IF A VALID KEY IS PRESSED, THEN THE ZERO FLAG IS SET WHEN THE RST RETURNS.

```
0379 DF          RST 18        ; SCAN/KEY READ RST
037A 21 24 08    LD HL,0824    ;(HL)=AUTO STEP CONTROL/TIMER BYTE
037D 28 0B       JR Z,038A     ;JUMP IF VALID KEY PRESSED
```

NO KEY IS PRESSED SO THE ROUTINE CHECKS FOR THE AUTO REPEAT MODE ENABLED FLAG (BIT 7 AUTO STEP CONTROL/TIMER BYTE, ZERO IS AUTO STEP ENABLED) AND DECREMENTS THE COUNTER IF IT IS. IF THE COUNTER REACHES ZERO, THEN IT IS RELOADED AND THE ROUTINE JUMPS TO RECOVER THE REGISTERS AND STEP THE NEXT INSTRUCTION. IF NOT IN THE AUTO MODE OR THE COUNTER DOES NOT REACH ZERO, THEN THE ROUTINE LOOPS BACK TO SCAN THE DISPLAY AND WAIT FOR EITHER A KEY PRESS OR FOR THE COUNTER TO REACH ZERO.

```
037F CB 7E       BIT 7,(HL)    ;TEST FOR AUTO INCREMENT JUMP IF NOT
0381 20 F6       JR NZ,0379    ;ENABLED TO SCAN/KEY READ LOOP
0383 35          DEC (HL)      ;DECREMENT COUNTER: LOOP TO
0384 20 F3       JR NZ,0379    ;SCAN/KEY READ UNTIL COUNT=0
```

AT THIS POINT THE AUTO-STEP DELAY HAS REACHED ZERO AND IS RELOADED WITH THE DELAY VALUE. A JUMP IS THEN DONE TO RECOVER THE REGISTERS AND STEP THE NEXT INSTRUCTION.

```
0386 36 30       LD (HL),30    ;RESET AUTO STEP DELAY, JUMP TO RECOVER
0388 18 22       JR 03AC       ;REGISTERS AND STEP NEXT INSTRUCTION
```

KEY PROCESSING STARTS HERE
THE AUTO-STEP IS DISABLED AND THEN THE KEY IS IDENTIFIED AND HANDLED.
THE AUTO-STEP WILL BE RE-ENABLED IF THE KEY PRESSED IS A DATA KEY.

```
038A 47          LD B,A        ;SAVE KEY
038B 36 FF       LD (HL),FF    ;SET AUTO STEP CONTROL/TIMER BIT 7
038D 21 5A 08    LD HL,085A    ;THUS DISABLING THE AUTO REPEAT MODE
0390 78          LD A,B        ;POINT HL TO CURRENT REG No. BUFFER
0391 FE 10       CP 10         ;PUT INPUT IN A,TEST IT FOR "+"
0393 20 08       JR NZ,039D    ;JUMP IF NOT TO TEST FOR "-"
```

"+" KEY HANDLER
THE CURRENT REGISTER NUMBER IS INCREMENTED AND THEN CHECK TO SEE THAT IT HAS NOT EXCEEDED THE HIGHEST REGISTER NUMBER (0C). IF IT HAS, THE ROUTINE JUMPS TO RESET THE CURRENT REGISTER NUMBER WITH 1, OTHERWISE IT JUMPS TO THE DISPLAY LOOP.

```
0395 34          INC (HL)      ;INCREMENT REGISTER NUMBER
0396 7E          LD A,(HL)     ;AND CHECK TO SEE IF IT LARGER
0397 FE 0D       CP 0D         ;THAN HIGHEST REG No. (0C): IF LOWER
0399 38 B6       JR C,0351     ;THAN 0D JUMP TO DISPLAY LOOP ELSE
039B 18 AF       JR 034C       ;JUMP TO SET REGISTER NUMBER TO 1
039D FE 11       CP 11         ;TEST FOR "-"
039F 20 07       JR NZ,03A8    ;JUMP IF NOT
```

"-" HANDLER
ONE IS TAKEN FROM THE CURRENT REGISTER NUMBER AND THEN IT IS CHECKED FOR ZERO. IF IT BECOMES ZERO, THEN THE CURRENT REGISTER NUMBER IS SET TO THE HIGHEST REGISTER NUMBER (0C) TO WRAP-AROUND TO DISPLAY THE LAST REGISTER.

```
03A1 35          DEC (HL)      ;SUBTRACT 1 FROM REGISTER NUMBER
03A2 20 AD       JR NZ,0351    ;JUMP IF NOT 0 TO UP-DATE DISPLAY
03A4 36 0C       LD (HL),0C    ;ELSE SET TO LAST REGISTER
03A6 18 A9       JR 0351       ;AND UP-DATE
```

TEST FOR "GO"

```
03A8 FE 12       CP 12         ;TEST FOR "GO" AND JUMP IF NOT
03AA 20 1A       JR NZ,03C6    ;TO TEST FOR "AD" OR DATA KEY
```

"GO" KEY
THE GO KEY CAUSES STEPPING EXECUTION TO CONTINUE.
BEFORE STEPPING IS CONTINUED THOUGH, THE KEYBOARD IS READ AND THE PROGRAM LOOPS UNTIL ALL KEYS ARE RELEASED. THIS IS TO SEPARATE KEY PRESSES MEANT FOR THE STEPPER AND THOSE FOR THE ROUTINE BEING STEPPED. ONCE ALL KEYS ARE RELEASED, ALL THE REGISTERS ARE POPPED OF THE REGISTER DISPLAY STACK, THE STACK IS RESTORED TO ITS "REAL" POSITION AND THE INTERRUPTS RE-ENABLED. THE RETURN ADDRESS FOR THE ROUTINE BEING STEPPED, STILL THERE ON THE TOP OF THE REAL STACK, IS USED AS THE RETURN ADDRESS.

```
03AC E7          RST 20        ;WAIT UNTIL ALL KEYS ARE RELEASED
03AD 28 FD       JR Z,03AC     ;BEFORE RESTARTING
```

```
03AF E1           POP HL         ;RECOVER ALL
03B0 F1           POP AF         ;REGISTERS
03B1 C1           POP BC         ;IN
03B2 D1           POP DE         ;THE
03B3 E1           POP HL         ;REVERSE
03B4 DD E1        POP IX         ;ORDER
03B6 FD E1        POP IY         ;TO
03B8 08           EX AF,AF'      ;HOW
03B9 D9           EXX            ;THEY
03BA F1           POP AF         ;STORED
03BB C1           POP BC         ;
03BC D1           POP DE         ;
03BD E1           POP HL         ;
03BE 08           EX AF,AF'      ;
03BF D9           EXX            ;
03C0 ED 7B 7E 08  LD SP,(087E)   ;AND STACK POINTER
03C4 FB           EI             ;RE-ENABLE THE INTERRUPTS
03C5 C9           RET            ;RET TO STEP NEXT INSTRUCTION

TEST FOR "AD" KEY (RETURN TO JMON)

03C6 FE 13        CP 13          ;TEST FOR "ADDR" KEY
03C8 20 01        JR NZ,03CB     ;JUMP IF NOT TO ASSUME DATA KEY
03CA C7           RST 00         ;RETURN TO MONITOR

DATA KEY HANDLER (ENABLE AUTO STEP)

03CB 3E 20        LD A,20        ;SET AND ENABLE AUTO STEP IN THE
03CD 32 24 08     LD (0824),A    ;CONTROL/TIMER BYTE (BIT 7 LOW, 20
03D0 18 A7        JR 0379        ;CYCLES): JUMP TO DISPLAY LOOP


-END OF STEPPER-

START OF MENU
MENU IS SET-UP FOR TAPE ROUTINE HERE
THE VARIABLES ARE MOVED FROM ROM TO RAM AND THE DISPLAY BUFFER IS SET TO 0800.

03D2 21 7C 07     LD HL,077C     ;LOAD HL WITH START OF TAPE
03D5 11 80 08     LD DE,0880     ;VARIABLES: DE IS RAM DE(stination)
03D8 01 18 00     LD BC,0018     ;BC IS THE COUNT
03DB ED B0        LDIR           ;SHIFT VARIABLES
03DD 21 00 08     LD HL,0800     ;PUT DISPLAY BUFFER AT 0800
03E0 22 2C 08     LD (082C),HL   ;

MENU DISPLAY LOOP STARTS HERE
THE MENU ENTRY NUMBER (MEN), HOLDS THE NUMBER OF THE CURRENT MENU ENTRY ON THE DISPLAY.
ALL ACTIONS OF THE MENU DRIVER CENTRE AROUND THIS BYTE.
THE DISPLAY ON THE TEC LED DISPLAY IS GENERATED BY SHIFTING BOTH THE DATA AND ADDRESS
DISPLAY CODES INTO THE RAM DISPLAY BUFFER.
ALL THE POSSIBLE DATA AND ADDRESS DISPLAY CODES ARE STORED IN SEPARATE TABLES IN ROM,
THE BASE OF EACH IS ADDRESSED BY THE CONTENTS OF MEMORY LOCATIONS 0895 (DATA TABLE), AND
0893 (ADDRESS TABLE).
THE FIRST MENU ENTRY IS DENOTED BY A ZERO VALUE IN THE MENU ENTRY NUMBER (MEN). THIS
MEANS THAT THE POSSIBLE ZERO CONDITION MUST BY DETECTED AND THE TABLE ENTRY CALCULATOR
SECTION SKIPPED OVER. WHEN ACCESSING THE DISPLAY TABLES, THE MENU ENTRY NUMBER IS
DECREMENTED UNTIL ZERO AND EACH TIME AN OFFSET EQUAL TO THE LENGTH OF EACH TABLE ENTRY
(4 FOR ADDR AND 2 FOR DATA TABLES) IS ADDED TO THE POINTERS.
AFTER THE REQUIRED ENTRIES ARE FOUND, THEY ARE MOVED INTO THE RAM DISPLAY BUFFER.

03E3 3A 8F 08     LD A,(088F)    ;GET MENU ENTRY NUMBER (MEN)
03E6 ED 5B 95 08  LD DE,(0895)   ;DE POINTS TO DATA DISPLAY TABLE
03EA 2A 93 08     LD HL,(0893)   ;HL POINTS TO ADDR DISPLAY TABLE
03ED 01 04 00     LD BC,0004     ;BC IS BOTH AN INDEX OFFSET AND
03F0 B7           OR A           ;BYTE COUNTER (USED BELOW): TEST
03F1 28 06        JR Z,03F9      ;A AND SKIP CALCULATOR IF ZERO
03F3 09           ADD HL,BC      ;ADD 4 TO HL TO POINT TO NEXT ADDR
03F4 13           INC DE         ;DISPLAY AND 2 TO DE FOR NEXT DATA
03F5 13           INC DE         ;DISPLAY
03F6 3D           DEC A          ;DO UNTIL A=0
03F7 20 FA        JR NZ,03F3     ;
03F9 E5           PUSH HL        ;SAVE ADDR POINTER  (not required)
03FA D5           PUSH DE        ;AND DATA POINTER
03FB 11 00 08     LD DE,0800     ;SHIFT ACROSS ADDR DISPLAY
03FE ED B0        LDIR           ;TO 0800 (BC=0004 FROM ABOVE)
0400 E1           POP HL         ;POP DATA DISPLAY ADDR INTO HL
```

```
0401 0E 02        LD C,02        ;SET BC TO SHIFT DATA DISPLAY BYTES
0403 ED B0        LDIR           ;SHIFT THE BYTES TO DISPLAY RAM
0405 E1           POP HL         ;CLEAN UP STACK
```

THIS SECTION CALLS THE SCAN/KEY/LCD/PATCH ROUTINE.
WHEN A KEY IS DETECTED A KEY HANDLER ROUTINE IS CALLED. THIS KEY HANDLER IS COMMON TO BOTH THE MENU DRIVER AND THE PERIMETER HANDLER AND IS DOCUMENTED ON FURTHER.
IF THE "GO" KEY WAS PRESSED, THE ZERO FLAG WILL BE SET WHEN THE COMMON KEY HANDLER RETURNS AND THE ROUTINE JUMPS TO THE GO HANDLER. IF NOT, THEN A (UNUSED BY JMON) ROUTINE (AT 0897) IS CALLED AND FINDS AN IMMEDIATE RETURN.
THE RETURN INSTRUCTION WAS PLACED AT 0897 WHEN THE TAPE'S MENU VARIABLES WERE SHIFTED FROM ROM TO RAM (SEE 0793).
A JUMP THEN LOOPS BACK TO THE MAIN DISPLAY LOOP TO UP-DATE THE DISPLAYS IN CASE OF A NEW MENU ENTRY NUMBER (MEN) BEING PROVIDED BY THE KEY HANDLER.
THE GO HANDLER IS A SIMPLE TABLE ENTRY CALCULATOR THAT USES THE MENU ENTRY NUMBER TO INDEX THROUGH A TABLE OF THREE BYTE JUMPS. LIKE THE DISPLAY CALCULATOR, THE ZERO POSSIBILITY IS TESTED FOR AND THE CALCULATOR SECTION IS SKIPPED OVER IF ZERO. WHEN THE REQUIRED TABLE ENTRY IS POINTED TO BY HL, IT IS THEN JUMPED TO VIA JP (HL), AND THE TABLE ENTRY, ITSELF BEING A 3 BYTE JUMP THEN JUMPS TO THE SELECTED MENU ENTRY'S ROUTINE.

```
0406 CD 42 08     CALL 0842      ;CALL SCAN/KEY/LCD/PATCH ROUTINE
0409 21 8F 08     LD HL,088F     ;POINT HL TO MENU ENTRY NUMBER
040C CD B2 04     CALL 04B2      ;CALL COMMON KEY HANDLER
040F 28 05        JR Z,0416      ;JUMP IF KEY WAS "GO" ELSE CALL TO
0411 CD 97 08     CALL 0897      ;RETURN INSTRUCTION (UNUSED BY JMON)
0414 18 CD        JR 03E3        ;LOOP TO MAIN DISPLAY LOOP
```

MENU "GO" KEY HANDLER

```
0416 2A 91 08     LD HL,(0891)   ;POINT HL TO BASE OF JUMP TABLE
0419 3A 8F 08     LD A,(088F)    ;GET MENU ENTRY NUMBER
041C B7           OR A           ;TEST FOR ZERO
041D 28 06        JR Z,0425      ;SKIP CALCULATOR IF ZERO
041F 23           INC HL         ;FIND JUMP VECTOR FOR THE CURRENT
0420 23           INC HL         ;MENU HEADING
0421 23           INC HL         ;
0422 3D           DEC A          ;
0423 20 FA        JR NZ,041F     ;
0425 E9           JP (HL)        ;AND JUMP TO THE REQUIRED ROUTINE
```

PERIMETER HANDLER SET-UP ROUTINES FOR THE TAPE SOFTWARE
WHEN GO IS PRESSED IN THE MENU HANDLER, ONE OF THE IMMEDIATE FOLLOWING ROUTINES IS EXECUTED (WHEN THE MENU IS WORKING WITH THE TAPE SOFTWARE). THESE ROUTINES SET-UP THE VARIABLES FOR THE MAIN TAPE FUNCTIONS (SAVE, TEST CS, TEST BL AND LOAD). THE TWO TESTS AND THE LOAD ROUTINE IS BASICALLY THE ONE ROUTINE, EXCEPT THAT EACH HAS ITS OWN PRIVATE SIGN-ON BYTE. LATER YOU WILL SEE THE THE ROUTINE TO LOAD OR TEST IS BASICALLY THE SAME AND THIS "SIGN-ON BYTE" SEPARATES THE DIFFERENT FUNCTIONS AT THE CRITICAL STAGE.
THE COMMON SECTION FOR THE LOAD AND TESTS, SETS THE PERIMETER HANDLER TO HAVE TWO WINDOWS, ONE FOR THE FILE NUMBER AND ONE FOR THE OPTIONAL START ADDRESS. IT ALSO SETS THE OPTIONAL START WINDOW TO FFFF (NO OPTIONAL START ADDRESS BY DEFAULT) AND PUTS THE EXECUTING ADDRESS OF THE LOAD/TESTS ROUTINE IN THE PERIMETER "GO" JUMP ADDRESS BUFFER.
THE SAVE SET-UP SETS THE NUMBER OF WINDOWS TO 4 AND STORES THE EXECUTING ADDRESS OF THE SAVE PREAMBLE ROUTINE IN THE PERIMETER "GO" JUMP ADDRESS BUFFER (0888).
THE 4 TAPE SAVE WINDOWS ARE: THE FILE NUMBER, THE START, THE END AND THE OPTIONAL AUTO GO ADDRESS.
ALL THE ABOVE ROUTINES HAVE A COMMON SET-UP AREA. THIS COMMON AREA STORES THE ROUTINE'S JUMP ADDRESS, IN HL, AND THE NUMBER OF WINDOWS, IN A, BOTH PROVIDED FROM THEIR OWN DEDICATED SECTION. THE COMMON AREA ALSO CLEARS THE "ACTIVE WINDOW NUMBER" TO ZERO SO THAT THE PERIMETER HANDLER WILL BE ENTERED WITH THE FIRST WINDOW (FILE NUMBER) SHOWING.

"LOAD" SET-UP

```
0426 AF           XOR A          ;CLEAR A FOR LOAD SIGN-ON BYTE
```

COMMON AREA FOR LOAD AND TESTS

```
0427 32 8A 08     LD (088A),A    ;SAVE SIGN-ON BYTE IN BUFFER
042A 3E 01        LD A,01        ;LOAD A WITH NUMBER OF WANTED
042C 21 FF FF     LD HL,FFFF     ;WINDOWS -1 (2 WINDOWS): SET
042F 22 9A 08     LD (089A),HL   ;OPTIONAL START WINDOW TO FFFF
0432 21 31 05     LD HL,0531     ;LOAD HL WITH "GO" ADDR OF LOAD/TEST
0435 18 0D        JR 0444        ;ROUTINE: JUMP TO STORE HL AND A
```

"TEST BLOCK" SET-UP

```
0437 3E 02        LD A,02        ;2=TEST BLOCK SIGN-ON BYTE
```

```
0439 18 EC        JR 0427        ;JUMP TO TEST/LOAD COMMON AREA

"TEST CHECKSUM" SET-UP

043B 3E 03        LD A,03        ;3=TEST CHECKSUM SIGN-ON BYTE
043D 18 FA        JR 0439        ;JUMP TO TEST/LOAD COMMON AREA

SAVE SET-UP

043F 21 50 04     LD HL,0450     ;POINT HL TO START OF SAVE PRE-AMBLE
0442 3E 03        LD A,03        ;SET UP FOR 4 WINDOWS

COMMON AREA FOR ALL SET-UPS

0444 22 88 08     LD (0888),HL   ;STORE HL AND A
0447 32 87 08     LD (0887),A    ;
044A AF           XOR A          ;SET MEN TO FIRST WINDOW (FILE NUMBER)
044B 32 86 08     LD (0886),A    ;
044E 18 23        JR 0473        ;JUMP TO PERIMETER HANDLER
```

SAVE ROUTINE PRE-AMBLE
THE SAVE PREAMBLE FITS IN BETWEEN THE PERIMETER HANDLER AND THE ACTUAL SAVE ROUTINE. THE PURPOSE OF IT IS TO SHIFT ACROSS THE FILE NUMBER, THE START ADDRESS AND THE OPTIONAL GO ADDRESS. IT ALSO CALCULATES THE LENGTH OF THE BLOCK AND TRANSFERS IT ACROSS TO THE TAPE FILE INFORMATION BLOCK WHICH IS OUTPUTTED TO THE TAPE.
IF THE END IS LOWER THAN THE START THE ROUTINE WILL JUMP TO DISPLAY "Err -In".

```
0450 2A 9E 08     LD HL,(089E)   ;SHIFT OPTIONAL GO TO OUTPUT BUFFER
0453 22 AA 08     LD (08AA),HL   ;
0456 2A 9A 08     LD HL,(089A)   ;SHIFT START ADDRESS OF BLOCK
0459 22 A6 08     LD (08A6),HL   ;TO TAPE FILE OUTPUT BUFFER
045C EB           EX DE,HL       ;PUT START OF BLOCK IN DE
045D 2A 9C 08     LD HL,(089C)   ;GET END OF BLOCK IN HL
0460 B7           OR A           ;CLEAR CARRY
0461 ED 52        SBC HL,DE      ;CALCULATE NUMBER OF BYTES IN
0463 23           INC HL         ;BLOCK (DIFFERENCE +1)
0464 DA 4A 00     JP C 004A      ;JUMP IF CARRY TO "Err-In"
0467 22 A8 08     LD (08A8),HL   ;STORE COUNT IN FILE INFO OUTPUT
046A 2A 98 08     LD HL,(0898)   ;SHIFT FILE NUMBER TO
046D 22 A4 08     LD (08A4),HL   ;TAPE FILE INFO OUTPUT BUFFER
0470 C3 F0 04     JP 04F0        ;JUMP TO SAVE OUTPUT ROUTINE
```

FINAL TAPE SET-UP BEFORE THE PERIMETER HANDLER. THIS PLACES FFFF IN THE OPTIONAL GO WINDOW BEFORE ENTERING THE PERIMETER HANDLER.

```
0473 21 FF FF     LD HL,FFFF     ;PUT FFFF IN OPTIONAL GO WINDOW
0476 22 9E 08     LD (089E),HL   ;
```

PERIMETER HANDLER
THE PERIMETER HANDLER ROUTINE IS SIMILAR TO THE MENU DRIVER. THE MAYOR DIFFERENCES ARE LISTED BELOW:
THE PERIMETER HANDLER CREATES ITS OWN ADDRESS DISPLAY CODES BY CONVERTING THE CONTENTS OF THE ACTIVE WINDOW TO DISPLAY CODE AND THEREFORE DOES NOT REQUIRE A TABLE OF ADDRESS DISPLAY CODES.
ANOTHER DIFFERENCE IS THE ADDRESS OF THE ROUTINE TO BE EXECUTED ON A "GO" PRESS IS SUPPLIED BY THE CALLING ROUTINE. THEREFORE THE PERIMETER HANDLER DOESN'T REQUIRE A JUMP TABLE AND ASSOCIATED CALCULATER.
THE ONLY OTHER MAYOR DIFFERENCE IS THAT THE PERIMETER HANDLER HAS ITS OWN BUILT IN DATA KEY HANDLER WHILE THE MENU DOES NOT.
THE FRONT SECTION BELOW CALCULATES THE ADDRESS OF THE ACTIVE WINDOW AND THE ADDRESS OF THE DATA DISPLAY FROM THE DISPLAY TABLE.
THE MENU ENTRY NUMBER FROM THE MENU DRIVER HAS AN EQUIVALENT HERE. IT IS THE ACTIVE WINDOW NUMBER AND IS USED IN IDENTICAL FASHION.

```
0479 3A 86 08     LD A,(0886)    ;GET NUMBER OF ACTIVE WINDOW
047C 2A 84 08     LD HL,(0884)   ;GET ADDRESS OF FIRST (FILE) WINDOW+1
047F ED 5B 82 08  LD DE,(0882)   ;GET BASE OF DATA DISPLAY TABLE
0483 B7           OR A           ;TEST ACTIVE WINDOW NUMBER FOR ZERO
0484 28 07        JR Z,048D      ;SKIP CALCULATOR IF ZERO
0486 13           INC DE         ;FINE CURRENT DATA DISPLAY
0487 13           INC DE         ;AND WINDOW
0488 23           INC HL         ;
0489 23           INC HL         ;
048A 3D           DEC A          ;
048B 20 F9        JR NZ,0486     ;
```

AFTER THE ADDRESS+1 OF THE ACTIVE WINDOW IS CALCULATED, IT IS STORED IN A BUFFER (AT 088C). EACH TIME A DATA KEY IS PRESSED, HL IS LOADED FROM THIS BUFFER AND THEREFORE POINTS TO THE ACTIVE WINDOW. THE DATA CAN THEN BE SHIFTED INTO THE ACTIVE WINDOW IMMEDIATELY.

```
048D 22 8C 08     LD (088C),HL  ;STORE ACTIVE WINDOW ADDRESS+1
```

BELOW THE DATA DISPLAY BYTES ARE PUT INTO THE DATA SECTION OF THE DISPLAY BUFFER VIA HL.

```
0490 EB           EX DE,HL      ;PUT DATA DISPLAY ADDRESS IN HL
0491 7E           LD A,(HL)     ;GET RIGHT-HAND DISPLAY BYTE IN A
0492 23           INC HL        ;AND LEFT-HAND IN H
0493 66           LD H,(HL)     ;PUT RIGHT-HAND BYTE IN L
0494 6F           LD L,A        ;HL HOLDS THE DATA DISPLAY BYTES
0495 22 04 08     LD (0804),HL  ;STORE DATA DISPLAY IN BUFFER
```

BELOW THE 16 BIT CONTENTS OF THE ACTIVE WINDOW ARE CONVERTED TO DISPLAY CODE ARE PLACED IN THE ADDRESS SECTION OF THE DISPLAY BUFFER.

```
0498 EB           EX DE,HL      ;GET ACTIVE WINDOW ADDRESS FROM DE
0499 7E           LD A,(HL)     ;AND TRANSFER
049A 2B           DEC HL        ;THE 16 BIT CONTENTS OF THE ACTIVE
049B 6E           LD L,(HL)     ;WINDOW INTO HL
049C 67           LD H,A        ;READY TO COVERT TO DISPLAY CODE
049D 01 00 08     LD BC,0800    ;BC=DISPLAY BUFFER START
04A0 CD 30 08     CALL 0830     ;CALL CONVERSION HL TO DISPLAY CODE
```

THE DISPLAY BUFFER IS NOW SET-UP AND THE SCAN/KEY LOOP IS CALLED. WHEN A KEY IS PRESSED, A COMMON KEY HANDLER IS CALLED.
THE COMMON KEY HANDLER DOES ALL THE REQUIRED PROCESSING FOR THE "+", "- " AND "AD" KEYS. IF EITHER THE "GO" AR A DATA KEY IS PRESSED, THEN THE HANDLER RETURNS WITH THE FLAGS SET TO SIGNIFY THESE KEYS.
IF "GO" IS PRESSED THEN THE ZERO FLAG IS SET AND THE "GO" HANDLER BELOW IS EXECUTED. IF A DATA KEY IS PRESSED THEN THE ZERO FLAG IS CLEAR (NOT ZERO) AND CARRY FLAG IS CLEAR THE DATA KEY HANDLER IS EXECUTED IF THESE CONDITIONS ARE MET.

```
04A3 CD 42 08     CALL 0842     ;CALL SCAN/KEY/LCD/PATCH ROUTINE
04A6 21 86 08     LD HL,0886    ;POINT HL TO ACTIVE WINDOW NUMBER
04A9 CD B2 04     CALL 04B2     ;CALL COMMON KEY HANDLER
04AC 20 16        JR NZ,04C4    ;JUMP IF NOT GO KEY TO TEST FOR DATA
04AE 2A 88 08     LD HL,(0888)  ;OR CONTROL KEY: ELSE GET JUMP ADDRESS
04B1 E9           JP (HL)       ;STORED BY SET-UP AND GO
```

COMMON KEY HANDLER
BECAUSE THE PERIMETER HANDLER AND THE MENU DRIVER ARE VERY SIMILAR, THEY ARE ABLE TO SHARE A COMMON KEY HANDLER.
THE ACTION OF THE KEY HANDLER IS AS FOLLOWS:
IF THE "AD" KEY IS PRESSED, THEN THE RETURN ADDRESS IS POPPED OFF THE STACK AND A RETURN IS DONE TO THE CALLING ROUTINE (USUALLY JMON). IF THE "GO" KEY IS PRESSED, THEN THE ZERO FLAG WILL BE SET AND A RETURN DONE. IT IS THEN UP TO THE CALLING ROUTINE TO SERVICE THE "GO" KEY.
A DATA KEY WILL BE FLAGGED BY SETTING THE CARRY FLAG AND CLEARING THE ZERO FLAG. LIKE THE "GO" KEY, THE CALLING ROUTINE MUST DECIDE WHAT IT IS TO DO WITH THE DATA KEY (THERE IS A BUILT IN DATA KEY HANDLER FOR THE PERIMETER HANDLER).
IF EITHER THE "+" OR "-" KEYS ARE PRESSED THEN A SPECIAL ROUTINE IS CALLED. THIS ROUTINE WILL ALTER THE CURRENT NUMBER OF THE ACTIVE WINDOW OR MENU ENTRY. THE RESULT IS THAT WHEN THE DISPLAY IS UP-DATED, THE DISPLAYS WILL BE SHIFTED TO EITHER THE NEXT DISPLAY FOR "+" OR TO THE PREVIOUS ONE FOR "- " AND WRAP-AROUND IF REQUIRED.

```
04B2 FE 10        CP 10         ;IS THE KEY "+"
04B4 28 1B        JR Z,04D1     ;JUMP IF SO TO "+" HANDLER
04B6 FE 11        CP 11         ;IS IT "-"
04B8 28 17        JR Z,04D1     ;JUMP IF SO TO "-" HANDLER
04BA FE 13        CP 13         ;IS IT "AD"
04BC 20 02        JR NZ,04C0    ;JUMP IF NOT TO TEST FOR "GO"
04BE E1           POP HL        ;CLEAN UP STACK
04BF C9           RET           ;RETURN TO JMON (OR CALLING ROUTINE)
04C0 FE 12        CP 12         ;IS IT "GO"
04C2 3F           CCF           ;CLEAR CARRY IF NOT IF GO C=1 Z=1
04C3 C9           RET           ;IF DATA SET Z=0 C=0: RETURN
```

BELOW IS THE PERIMETER HANDLER DATA KEY HANDLER/DISCRIMINATOR
IF THE KEY WAS "+" OR "-" THEN IT HAS ALREADY BEEN HANDLED AND THIS CONDITION IS FLAGGED BY THE CARRY BEING SET. IN THIS CASE, A JUMP IS DONE BACK TO THE MAIN BODY TO UP-DATE THE DISPLAY OTHERWISE THE DATA KEY VALUE IS SHIFTED INTO THE ACTIVE WINDOW.

```
04C4 38 B3        JR C,0479     ;JUMP IF KEY WAS "+" OR "-"
04C6 2A 8C 08     LD HL,(088C)  ;POINT HL TO ACTIVE WINDOW+1
```

```
04C9 2B          DEC HL         ;POINT TO LOW ORDER BYTE
04CA ED 6F       RLD            ;SHIFT IN DATA KEY VALUE
04CC 23          INC HL         ;AND SHIFT OTHER NIBBLES
04CD ED 6F       RLD            ;ACROSS
04CF 18 A8       JR 0479        ;JUMP BACK TO UP-DATE DISPLAY
```

THIS ROUTINE IS CALLED FROM THE COMMON KEY HANDLER IF EITHER "+" OR "-" HAVE BEEN PUSHED.
THIS ROUTINE WILL EITHER INCREMENT OR DECREMENT THE MEMORY LOCATION ADDRESSED BY HL FOR THE "+" AND "-" KEY RESPECTIVELY. HL WAS LOADED BY THE CALLING ROUTINE TO POINT TO ITS MAIN CONTROLLING BYTE. THIS IS EITHER THE CURRENT MENU ENTRY NUMBER (MENU DRIVER), OR THE ACTIVE WINDOW NUMBER (PERIMETER HANDLER), BOTH OF WHICH HAVE BEEN DESCRIBED PREVIOUSLY. AFTER INCREMENTING OR DECREMENTING (HL), THIS ROUTINE THEN CHECKS THAT THE VALUE IN (HL) IS NOT GREATER THAT THE BYTE AT HL+1 (WHICH IS THE MAXIMUM NUMBER OF DISPLAYS LESS 1). KEEP IN MIND, IF IT UNDERFLOWED FROM ZERO IT WILL BECOME FF AND BE HIGHER THAN (HL). THIS SECOND BYTE (AT HL+1) IS THE NUMBER OF ALLOWABLE DISPLAYS-1 AND WAS PROVIDED BY THE ROM TABLE FOR THE (TAPE) MENU DRIVER, AND PROVIDED BY THE PERIMETER HANDLER SET-UP ROUTINES (REFER TO 042A AND 0442).
IF THE FIRST BYTE BECOMES HIGHER THAN THE SECOND, THEN THE ROUTINE CHECKS TO SEE WHICH KEY WAS PRESSED. IF THE "+" KEY WAS, THEN (HL) IS CLEARED. THIS WILL CAUSE MENU OR PERIMETER HANDLER TO SHOW ITS FIRST DISPLAY WHEN RE-ENTERED.
IF THE KEY WAS "-", THEN THE MAXIMUM NUMBER OF DISPLAYS-1 (WHICH IS THE SAME AS THE NUMBER OF THE FINAL DISPLAY) IS TRANSFERRED INTO (HL) (THE NUMBER OF THE CURRENT DISPLAY). THIS WILL CAUSE THE LAST DISPLAY TO BE SHOWN WHEN THE MENU DRIVER OR PERIMETER HANDLER IS RE-ENTERED.
IF THERE IS NO UNDERFLOW OR OVERFLOW THEN THE ROUTINE RETURNS JUST AFTER IT HAS EITHER INCREMENTED OR DECREMENTED THE CURRENT NUMBER OF THE MENU ENTRY NUMBER OR ACTIVE WINDOW NUMBER.
WHEN THE MENU DRIVER OR PERIMETER HANDLER ARE RE-ENTERED, THEY WILL SHOW THE NEXT DISPLAY FOR "+" OR THE PREVIOUS FOR "-" AND WRAP-AROUND AUTOMATICALLY IF REQUIRED.

```
04D1 4F          LD C,A         ;SAVE INPUT KEY VALUE IN C
04D2 23          INC HL         ;PUT MAX NUMBER OF DISPLAYS-1
04D3 46          LD B,(HL)      ;IN B
04D4 2B          DEC HL         ;RESET HL TO POINT TO CURRENT NUMBER
04D5 0F          RRCA           ;WAS KEY "+" OR "-"? BIT 0 WILL TELL
04D6 7E          LD A,(HL)      ;PUT CURRENT NUMBER IN A
04D7 38 02       JR C,04DB      ;JUMP IF KEY WAS "-"
04D9 3C          INC A          ;INCREASE A BY 2
04DA 3C          INC A          ;
04DB 3D          DEC A          ;DECREASE A BY ONE
04DC 04          INC B          ;ADD 1 TO MAX NUMBER-1: IS CURRENT
04DD B8          CP B           ;NUMBER EQUAL OR GREATER THAN MAX?
04DE 30 05       JR NC,04E5     ;JUMP IF SO TO UNDER/OVERFLOW HANDLER
04E0 77          LD (HL),A      ;ELSE STORE UPDATED CURRENT NUMBER
04E1 AF          XOR A          ;SET ZERO FLAG
04E2 3D          DEC A          ;CHANGE ZERO FLAG TO 0
04E3 37          SCF            ;AND SET CARRY
04E4 C9          RET            ;DONE
04E5 CB 41       BIT 0,C        ;TEST FOR "+" OR "-"
04E7 20 03       JR NZ,04EC     ;JUMP IF "-" TO SET CURRENT NUMBER
04E9 AF          XOR A          ;TO LAST DISPLAY: ELSE SET FIRST
04EA 18 F4       JR 04E0        ;DISPLAY: JUMP TO STORE NEW NUMBER
04EC 05          DEC B          ;CORRECT MAX NUMBER-1
04ED 78          LD A,B         ;SET A TO LAST DISPLAY NUMBER
04EE 18 F0       JR 04E0        ;JUMP TO STORE LAST DISPLAY NUMBER
```

THIS IS THE TAPE OUTPUT ROUTINE
THE ACTION IS AS FOLLOWS:
A LEADER OF LOW FREQUENCY TONE IS OUTPUTTED FOLLOWED BY THE FILE INFORMATION BLOCK.
AFTER THE FILE INFORMATION BLOCK IS OUTPUTTED, SEVERAL SECONDS OF HIGH FREQUENCY MIDDLE SYNC IS OUTPUTTED. THE TIME IT TAKES TO OUTPUT THE MIDDLE SYNC IS USED BY THE TAPE INPUT ROUTINE TO DISPLAY THE FILE NUMBER.
THE DATA TO BE SAVED ON TAPE IS BROKEN UP INTO BLOCKS OF 256 BYTES AND OUTPUTTED WITH A CHECKSUM AT THE END OF EACH BLOCK. A COUNTER IS SHOWN ON THE TEC LED DISPLAY THAT SHOWS HOW MANY COMPLETE BLOCKS LEFT (UP TO 16 BLOCKS).
IF THERE IS AN ODD SIZE BLOCK, IT IS OUTPUTTED AS THE LAST BLOCK.
AFTER ALL THE BLOCKS HAVE BEEN OUTPUTTED, AN END OF FILE HIGH FREQUENCY TONE IS OUTPUTTED.

```
04F0 21 00 30    LD HL,3000     ;HL HAS NUMBER OF LEADER CYCLES
04F3 CD 80 06    CALL 0680      ;CALL LOW TONE
04F6 21 A4 08    LD HL,08A4     ;HL IS START OF FILE INFORMATION BLOCK
04F9 06 0C       LD B,0C        ;LOAD B WITH NUMBER OF BYTES TO BE
04FB AF          XOR A          ;OUTPUTTED: ZERO A FOR CHECKSUM
04FC CD 4B 06    CALL 064B      ;CALL OUT BLOCK
04FF 21 00 50    LD HL,5000     ;LD HL WITH MID SYNC CYCLE COUNT
```

```
0502 CD 84 06     CALL 0684      ;CALL HIGH TONE
0505 2A A6 08     LD HL,(08A6)   ;LOAD HL WITH START OF OUTPUT BLOCK
```

OUTPUT LOOP STARTS HERE
THE DISCUSSION BELOW ON THE BYTE COUNTER AND BLOCK FORMATION APPLIES TO THE TAPE INPUT LOOP ALSO. THE TAPE INPUT LOOP DESCRIPTION WILL REFER YOU BACK TO THESE NOTES.
THE BYTE COUNT IS PUT INTO BC AND THEN A ROUTINE TO CONVERT B (THE TOTAL NUMBER OF FULL BLOCKS TO BE OUTPUTTED) TO DISPLAY FORMAT AND OUTPUT IT IS CALLED.
THE CONVERSION ROUTINE ALSO TESTS B FOR ZERO. IF B IS NOT ZERO, THE ROUTINE RETURNS WITH THE ZERO FLAG CLEAR (NOT ZERO) AND THE HIGH ORDER BYTE OF THE BYTE COUNT IN B IS DECREMENTED BY ONE AND STORED IN ITS BUFFER. THIS COUNTS DOWN THE BLOCKS. B IS THEN ZEROED SO THAT A FULL BLOCK (256 BYTES) WILL BE OUTPUTTED ON RETURNING.
IF THE HIGH ORDER BYTE OF THE BYTE COUNT (IN B) IS ZERO (NO FULL BLOCK OF 256 BYTES) THEN C (THE LOW ORDER BYTE OF THE COUNT) IS TRANSFERRED INTO B AND THE ZERO FLAG IS SET.
THE CONVERSION THEN RETURNS WITH THE NUMBER (IF ANY) OF REMAINING BYTES IN B.
AFTER THE CONVERSION ROUTINE HAS RETURNED, A JUMP IS DONE IF THE ZERO FLAG IS CLEAR (DENOTING A NOT ZERO STATE). THIS JUMP SKIPS AHEAD TO SAVE THE FLAGS AND OUTPUT ONE FULL BLOCK.
IF THE ZERO FLAG IS SET, THEN THE ROUTINE BELOW CHECKS TO SEE IF THE LOW ORDER BYTE (FROM C) THAT HAS BEEN PLACED IN B, IS ZERO. IF THE LOW ORDER BYTE IS ZERO, THEN ALL THE BYTES HAVE BEEN OUTPUTTED. THE ROUTINE THEN JUMPS TO DISPLAY "-END-S".
IF THE LOW ORDER BYTE OF THE COUNT IS NOT ZERO THEN THE ZERO FLAG IS SET AND SAVED ON THE STACK BEFORE WHAT ARE NOW KNOWN TO BE THE LAST IS OUTPUTTED.
BEFORE THE DATA IS SENT TO THE TAPE, A SHORT HIGH TONE SYNC IS OUTPUTTED TO COVER THE SOFTWARE OVERHEAD OF THE TAPE INPUT ROUTINE, AND A IS ZEROED TO BE USED AS THE CHECK-SUM.

```
0508 ED 4B A8 08  LD BC,(08A8)   ;LOAD BC WITH NUMBER OF BYTES
050C CD C9 05     CALL 05C9      ;CALL ROUTINE TO DISPLAY BLOCK COUNT
050F 20 05        JR NZ,0516     ;AND TEST LENGTH: JUMP IF FULL BLOCK
0511 78           LD A,B         ;TO OUTPUT: TEST LOW BYTE OF COUNT
0512 B7           OR A           ;IN B IS ZERO AND JUMP TO DISPLAY
0513 28 11        JR Z,0526      ;"-END-S" IF SO
```

THE XOR A INSTRUCTION BELOW SETS THE ZERO FLAG TO SIGNIFY THAT THE BLOCK ABOUT TO BE OUTPUTTED IS THE FINAL BLOCK. THE ROUTINE WILL THEN DISPLAY "-END-S" (AFTER A SHORT END SYNC TONE).

```
0515 AF           XOR A          ;SET ZERO FLAG
0516 F5           PUSH AF        ;AND SAVE ON STACK
```

AT THIS POINT IF THE ZERO FLAG ON THE STACK IS CLEAR (NOT ZERO STATE), THEN AFTER THE CURRENT BLOCK IS OUTPUTTED, THE ROUTINE WILL LOOP BACK TO START OF THE OUTPUT LOOP TO SEE IF THERE IS ANY MORE BYTES TO BE OUTPUTTED.

```
0517 D9           EXX            ;SWAP REGISTERS
0518 21 14 02     LD HL,0214     ;LOAD HL FOR SHORT BURST OF
051B CD 84 06     CALL 0684      ;HIGH TONE
051E D9           EXX            ;SWAP BACK REGISTERS
051F AF           XOR A          ;ZERO A FOR CHECKSUM
0520 CD 4B 06     CALL 064B      ;CALL OUTBLOCK
0523 F1           POP AF         ;RECOVER FLAGS AND JUMP IF
0524 20 E2        JR NZ,0508     ;THERE MIGHT BE MORE TO OUTPUT
```

ALL BLOCKS HAVE BEEN OUTPUTTED SO FINISH WITH A SHORT END TONE AND SET-UP END DISPLAY "-END-S".

```
0526 21 00 10     LD HL,1000     ;LOAD HL WITH SHORT END TONE
0529 CD 84 06     CALL 0684      ;CALL HIGH TONE
052C 3E 05        LD A,05        ;LD A TO INDEX "END-S DISPLAY
052E C3 E6 03     JP 03E6        ;JUMP BACK TO MENU
```

THIS IS THE START OF THE TAPE INPUT SECTION.
THE ACTION HERE IS TO DETECT A VALID LEADER BY COUNTING 1000H CYCLES OF LOW FREQUENCY TONE. AFTER THIS HAS BEEN DETECTED, THE ROUTINE WAITS UNTIL IT DETECTS THE START BIT OF THE FILE INFORMATION BLOCK. THE BLOCK IS THEN LOADED IN AND A CHECK-SUM COMPARE IS DONE. IF AN ERROR IS DETECTED, THE ROUTINE JUMPS TO DISPLAY "FAIL -XX", OTHERWISE THE FILE NUMBER IS CONVERTED TO DISPLAY FORMAT AND DISPLAYED FOR A FEW SECONDS.

```
0531 01 00 10     LD BC,1000     ;LOAD BC TO COUNT 1000 CYCLES
0534 CD 30 06     CALL 0630      ;CALL PERIOD
0537 38 F8        JR C,0531      ;LOOP UNTIL LOW TONE IS DETECTED
0539 0B           DEC BC         ;COUNT LONG
053A 78           LD A,B         ;PERIODS
053B B1           OR C           ;IF BC REACHES ZERO THEN IT IS
053C 20 F6        JR NZ,0534     ;ACCEPTED THAT A VALID FILE FOLLOWS
053E 06 0C        LD B,0C        ;LOAD B TO INPUT 12 BYTES AND
```

```
0540 21 A4 08     LD HL,08A4    ;POINT HL TO FILE INFO BLOCK INPUT
0543 CD 30 06     CALL 0630     ;BUFFER: CALL PERIOD
0546 30 FB        JR NC,0543    ;AND WAIT FOR LOW TONE TO END
0548 CD E7 05     CALL 05E7     ;CALL INBLOCK TO GET FILE INFO BLOCK
054B 20 54        JR NZ,05A1    ;JUMP NOT ZERO TO FAIL LOAD ROUTINE
054D 01 00 08     LD BC,0800    ;LOAD BC TO POINT TO DISPLAY BUFFER
0550 2A A4 08     LD HL,(08A4)  ;PUT FILE NUMBER INTO HL
0553 CD 30 08     CALL 0830     ;CONVERT HL TO DISPLAY CODE
0556 3E 47        LD A,47       ;PUT "F" IN DISPLAY BUFFER
0558 32 05 08     LD (0805),A   ;FOR "FILE"
055B 01 F2 01     LD BC,01F2    ;LD BC WITH THE DISPLAY ON TIME
055E C5           PUSH BC       ;SAVE ON STACK
055F CD 36 08     CALL 0836     ;CALL SCAN
0562 C1           POP BC        ;RECOVER BC
0563 0B           DEC BC        ;DECREMENT
0564 78           LD A,B        ;AND LOOP UNTIL
0565 B1           OR C          ;BC IS ZERO
0566 20 F6        JR NZ,055E    ;
```

AFTER A FILE INFORMATION BLOCK IS LOADED AND THE FILE NUMBER DISPLAYED, A TEST IS DONE ON THE REQUIRED FILE NUMBER WINDOW. FIRST IT IS TESTED FOR FFFF (LOAD/TEST NEXT FOUND FILE). IF FFFF, THE ROUTINE SKIPS AHEAD TO LOAD/TEST THE FILE. OTHERWISE THE REQUIRED FILE NUMBER IS SUBTRACTED FROM THE JUST LOADED FILE NUMBER, IF THE RESULT IS ZERO THEN THE FILE IS THE ONE SELECTED AND IS LOADED/TESTED.
THE OPTIONAL START WINDOW IS THEN TESTED FOR FFFF. IF IT IS, THE START ADDRESS FROM THE TAPE IS USED. IF THE OPTIONAL START BUFFER HAS SOMETHING OTHER THAT FFFF, THEN THE ADDRESS HERE IS USED AS THE START ADDRESS TO LOAD/TEST THE TAPE.

```
0568 2A 98 08     LD HL,(0898)  ;TEST FOR FFFF IN FILE NAME WINDOW
056B 23           INC HL        ;
056C 7C           LD A,H        ;
056D B5           OR L          ;
056E 2B           DEC HL        ;JUMP IF FILE WINDOW IS FFFF
056F 28 09        JR Z,057A     ;TO INPUT FILE REGARDLESS OF ITS NUMBER
0571 ED 5B A4 08  LD DE,(08A4)  ;ELSE TEST THAT INPUT FILE NAME
0575 B7           OR A          ;IS THE SAME AS THE ONE IN THE FILE
0576 ED 52        SBC HL,DE     ;NUMBER WINDOW AND JUMP IF NOT
0578 20 B7        JR NZ,0531    ;SELECTED FILE TO LOOK FOR NEXT FILE
057A 2A 9A 08     LD HL,(089A)  ;TEST THAT OPTIONAL START ADDRESS
057D 23           INC HL        ;IS FFFF
057E 7C           LD A,H        ;
057F B5           OR L          ;
0580 2B           DEC HL        ;
0581 20 03        JR NZ,0586    ;JUMP IF NOT, ELSE USE START ADDRESS
0583 2A A6 08     LD HL,(08A6)  ;PROVIDED FROM THE TAPE
```

THE MAIN LOAD/TEST ROUTINE STARTS HERE.
REFER TO THE DESCRIPTION OF THE BYTE COUNT AND BLOCK FORMATION AT THE OUTPUT SECTION ROUTINE (SEE 508).
WHEN ALL THE BLOCKS HAVE BEEN INPUTTED AND THE ROUTINE JUMPS TO DISPLAY PASS/FAIL -Ld ON THE LED DISPLAY.
HL IS POINTING TO THE PLACE IN MEMORY WHERE THE FILE WILL BE LOADED/TESTED.

```
0586 ED 4B A8 08  LD BC,(08A8)  ;PUT NUMBER OF BYTES INTO BC
058A CD C9 05     CALL 05C9     ;CALL B CONVERT AND TEST
058D 20 05        JR NZ,0594    ;JUMP IF NOT ZERO AS THERE IS AT
058F 78           LD A,B        ;LEAST ONE FULL BLOCK TO LOAD/TEST
0590 B7           OR A          ;CHECK THAT B (FORMALLY C)=0
0591 28 0A        JR Z,059D     ;JUMP IF SO AS ALL BYTES DONE
0593 AF           XOR A         ;ELSE SET ZERO FLAG TO REMEMBER
0594 F5           PUSH AF       ;SAVE FLAGS ON STACK
0595 CD E3 05     CALL 05E3     ;CALL INBLOCK
0598 20 06        JR NZ,05A0    ;JUMP IF LOAD/TEST FAILED
059A F1           POP AF        ;RECOVER FLAGS
059B 20 E9        JR NZ,0586    ;LOOP IF THERE MIGHT BE MORE
059D AF           XOR A         ;SET ZERO (SUCCESS) FLAG
059E 18 01        JR 05A1       ;JUMP TO END HANDLER
05A0 D1           POP DE        ;CLEAN UP STACK
05A1 20 10        JR NZ,05B3    ;JUMP IF FAILED LOAD/TEST
```

THE LOAD/TEST HAS PASSED. TEST HERE FOR OPTIONAL AUTO-GO AND FOR LOAD OPERATION (NO AUTO-GO FOR TEST OPERATIONS). START EXECUTION AT AUTO-GO ADDRESS IF REQUIRED.

```
05A3 2A AA 08     LD HL,(08AA)  ;PUT OPTIONAL GO ADDRESS IN HL
05A6 23           INC HL        ;TEST FOR FFFF
05A7 7C           LD A,H        ;AND JUMP
```

```
05A8 B5           OR L          ;IF FFFF
05A9 2B           DEC HL        ;AS THERE
05AA 28 07        JR Z,05B3     ;IS NO AUTO-GO
05AC 3A 8A 08     LD A,(088A)   ;TEST THAT A LOAD OPERATION WAS
05AF B7           OR A          ;DONE
05B0 20 01        JR NZ,05B3    ;SKIP JUMP IF IT WAS A TEST
05B2 E9           JP (HL)       ;ELSE AUTO START THE PROGRAM
```

THE POST LOAD/TEST MENU DISPLAYS ARE SET UP HERE. IF THE LOAD/TEST FAILED THE ZERO FLAG IS CLEAR THE ROUTINE WILL POINT TO THE "FAIL" DISPLAY. OTHERWISE IT IS SET TO POINT TO THE "PASS" DISPLAY. THE DATA DISPLAY IS CALCULATED BY ADDING THE MENU ENTRY NUMBER OF THE JUST PERFORMED OPERATION X2, TO THE TABLE BASE OF POST LOAD/TEST DATA DISPLAYS. (THE MENU ENTRY NUMBER IS STILL THE SAME AS IT WAS WHEN "GO" WAS PRESS FROM THE MENU).

```
05B3 11 68 07     LD DE,0768    ;LOAD DE TO BASE OF DATA DISPLAY
05B6 21 5C 07     LD HL,075C    ;TABLE AND HL "FAIL" DISPLAY
05B9 20 02        JR NZ,05BD    ;TABLE:
05BB 2E 58        LD L,58       ;ADJUST HL TO PASS IF ZERO
05BD EB           EX DE,HL      ;SWAP HL AND DE
05BE 3A 8F 08     LD A,(088F)   ;FIND WHAT OPERATION WAS PERFORMED
05C1 07           RLCA          ;AND DOUBLE VALUE AND ADD TO HL TO
05C2 85           ADD A,L       ;POINT HL AT POST TAPE OPERATION
05C3 6F           LD L,A        ;DATA DISPLAY ENTRY (SEE 0768-0771)
05C4 EB           EX DE,HL      ;SWAP BACK HL AND DE
05C5 AF           XOR A         ;ZERO A
05C6 C3 47 00     JP 0047       ;JUMP TO SOFT MENU ENTRY
```

THIS IS THE CONVERT/TEST B ROUTINE.
THE VALUE IN B IS CONVERTED AND OUTPUTTED TO PORT 2.
THEN B IS TESTED AND ONE OF THE FOLLOWING OPERATIONS IS PERFORMED. IF B=0 THEN C IS TRANSFERRED INTO B AND THE ZERO FLAG IS SET. IF B IS NOT 0 THEN B IS DECREMENTED, THE COUNT IS UP-DATED IN ITS BUFFER AND THE ZERO FLAG AND B IS CLEARED.

```
05C9 78           LD A,B        ;PUT HIGH BYTE OF COUNT IN A
05CA E6 0F        AND 0F        ;MASK TO ONE DIGIT
05CC 11 D0 07     LD DE,07D0    ;POINT DE TO DISPLAY CODE TABLE
05CF 83           ADD A,E       ;ADD A
05D0 5F           LD E,A        ;
05D1 1A           LD A,(DE)     ;GET DISPLAY VALUE
05D2 D3 02        OUT 02,A      ;OUTPUT IT TO DISPLAY
05D4 78           LD A,B        ;TEST HIGH BYTE
05D5 B7           OR A          ;FOR ZERO
05D6 28 09        JR Z,05E1     ;JUMP IF ZERO
05D8 05           DEC B         ;ELSE DECREASE COUNT BY ONE BLOCK
05D9 ED 43 A8 08  LD (08A8),BC  ;STORE COUNT
05DD 06 00        LD B,00       ;LOAD B FOR 256 BYTE OUTPUT BLOCK
05DF B7           OR A          ;CLEAR ZERO FLAG
05E0 C9           RET           ;DONE
05E1 41           LD B,C        ;PUT LAST BLOCK SIZE IN B
05E2 C9           RET           ;DONE
```

THIS BLOCK LOADS/TESTS THE BYTES IN FROM THE TAPE. THE NUMBER OF BYTES IS HELD IN B ON INPUT. AFTER THE SUB-ROUTINE THAT INPUTS A BYTE IS CALLED, A TEST AND JUMP IS DONE. THE TEST AND JUMP SELECT THE REQUIRED CODE TO PERFORM A LOAD OR TEST AS SELECTED FROM THE MENU BY THE USER. THE CHECK-SUM LOADED FROM THE TAPE HAS HAD ONE ADDED TO IT BY THE TAPE OUTPUT ROUTINE. THIS ADDED ONE IS REMOVED IN THIS ROUTINE BEFORE THE CHECK-SUM COMPARE IS DONE.

```
05E3 3A 8A 08     LD A,(088A)   ;GET CURRENT OPERATION
05E6 4F           LD C,A        ;SAVE IN C
05E7 AF           XOR A         ;CLEAR A FOR CHECKSUM
05E8 F5           PUSH AF       ;SAVE CHECKSUM
05E9 CD 0B 06     CALL 060B     ;CALL GET BYTE
05EC CB 49        BIT 1,C       ;TEST FOR CURRENT OPERATION
05EE 20 0E        JR NZ,05FE    ;JUMP IF A EITHER TEST
05F0 73           LD (HL),E     ;ELSE STORE INPUTTED BYTE IN MEMORY
05F1 23           INC HL        ;POINT TO NEXT LOCATION
05F2 F1           POP AF        ;GET CHECKSUM
05F3 83           ADD A,E       ;ADD TO NEW BYTE
05F4 10 F2        DJNZ,05E8     ;DO UNTIL BLOCK DONE
05F6 F5           PUSH AF       ;SAVE CHECKSUM
05F7 CD 0B 06     CALL 060B     ;GET TAPE CHECKSUM
05FA F1           POP AF        ;GET MEMORY CHECKSUM
05FB 1D           DEC E         ;CORRECT TAPE CHECKSUM
05FC BB           CP E          ;TEST CHECKSUMS TO SET FLAGS
05FD C9           RET           ;BLOCK DONE
```

```
05FE CB 41        BIT 0,C        ;TEST FOR WHICH TEST
0600 28 F0        JR Z,05F2      ;JUMP IF CHECKSUM ONLY TEST
0602 F1           POP AF         ;GET CHECKSUM
0603 57           LD D,A         ;SAVE IN D
0604 7B           LD A,E         ;GET INPUT BYTE
0605 BE           CP (HL)        ;TEST TO MEMORY
0606 23           INC HL         ;POINT TO NEXT LOCATION
0607 7A           LD A,D         ;PUT CHECKSUM BACK IN A
0608 28 E9        JR Z,05F3      ;JUMP TO MAIN LOOP IF ALL OK
060A C9           RET            ;RETURN IF ERROR
```

THIS ROUTINE INPUTS A SINGLE BYTE.

```
060B CD 18 06     CALL 0618      ;GET START BIT
060E 16 08        LD D,08        ;LOAD D FOR 8 BITS
0610 CD 18 06     CALL 0618      ;GET BIT
0613 CB 1B        RR E           ;PUT IT IN E
0615 15           DEC D          ;
0616 20 F8        JR NZ,0610     ;DO FOR EIGHT BITS,
```

THIS ROUTINE INPUTS A SINGLE BIT
THE STRUCTURE OF EACH BIT IS IMPORTANT TO UNDERSTAND AT THIS POINT. A LOGIC 0 IS REPRESENTED BY 4 SHORT PERIODS FOLLOWED BY 1 LONG PERIOD AND A LOGIC 1 BY 2 SHORT PERIODS AND 2 LONG PERIODS. THESE ARE HIGH SPEED FIGURES. FOR LOW SPEED THE ABOVE COUNTS ARE DOUBLED.
THE BITS ARE DECODED BY COUNTING THE RATIO OF SHORT PERIODS TO LONG PERIODS. A COMPLICATED METHOD OF COUNTING IS USED TO RESULT IN THE BIT VALUE BEING REFLECTED IN BIT 7 OF L. THE ROUTINE IS TERMINATED WHEN A SHORT PERIOD THAT FOLLOWED A LONG PERIOD IS DETECTED. THE LONG PERIOD IS FLAGGED WITH BIT 0 OF H. THE "SHORT AFTER LONG" PERIOD USED FOR TERMINATION IS ACTUALLY THE FIRST CELL OF THE NEXT BIT.
THE VALUE OF THE BIT INPUTTED IS THEN PUT INTO THE CARRY FLAG.

```
0618 D9           EXX            ;SWAP REGISTERS
0619 21 00 00     LD HL,0000     ;ZERO HL
061C CD 30 06     CALL 0630      ;CALL TO MEASURE PERIOD
061F 38 06        JR C,0627      ;JUMP IF SHORT PERIOD
0621 2D           DEC L          ;SET HIGH ORDER BIT OF L TO ONES
0622 2D           DEC L          ;
0623 CB C4        SET 0,H        ;REMEMBER THAT THE LONG PERIOD
0625 18 F5        JR 061C        ;HAS BEEN DETECTED: LOOP BACK
0627 2C           INC L          ;SHORT PERIOD SO ADD ONE TO L
0628 CB 44        BIT 0,H        ;TEST FOR SHORT AFTER LONG PERIOD
062A 28 F0        JR Z,061C      ;JUMP IF NOT
062C CB 15        RL L           ;END OF BIT: PUT BIT 7,L INTO
062E D9           EXX            ;CARRY: SWAP REGISTERS
062F C9           RET            ;INPUT BIT IN CARRY
```

THIS ROUTINE INPUTS AND MEASURES THE PERIOD OF EACH TAPE CELL AND COMPARES IT TO THE THRESHOLD BETWEEN A SHORT AND LONG PERIOD. THE CELL IS ALSO ECHOED ON THE TEC SPEAKER.

```
0630 11 00 00     LD DE,0000     ;ZERO DE FOR PERIOD MEASUREMENT
0633 DB 03        IN A,03        ;TEST TAPE LEVEL
0635 13           INC DE         ;TIME PERIOD
0636 17           RLA            ;PUT TAPE LEVEL INTO CARRY
0637 38 FA        JR C,0633      ;LOOP UNTIL IT GOES LOW
0639 AF           XOR A          ;ECHO IT ON
063A D3 01        OUT (01),A     ;THE TEC SPEAKER
063C DB 03        IN A,03        ;MEASURE SECOND HALF OF CYCLE
063E 13           INC DE         ;IN THE SAME FASHION AS ABOVE
063F 17           RLA            ;
0640 30 FA        JR NC,063C     ;THIS TIME LOOP UNTIL TAPE LEVEL
0642 3E 84        LD A,84        ;GOES HIGH: ECHO IT ON TEC SPEAKER
0644 D3 01        OUT (01),A     ;
0646 7B           LD A,E         ;GET PERIOD MEASUREMENT
0647 FE 1A        CP 1A          ;COMPARE IT TO THRESHOLD
0649 C9           RET            ;TO SET FLAGS: DONE
```

THIS ROUTINE OUTPUTS A BLOCK TO THE TAPE. THE NUMBER OF BYTES IS HELD IN B AND THE BLOCK IS ADDRESS BY HL. AFTER ALL THE BYTES HAVE BEEN OUTPUTTED, THE CHECKSUM +1, WHICH WAS ADDED UP AS EACH BYTE WAS OUTPUTTED, IS SENT TO THE TAPE.

```
064A 08           EX AF,AF'      ;GET CHECKSUM IN A
064B 5E           LD E,(HL)      ;PUT BYTE TO BE OUTPUTTED IN E
064C 83           ADD A,E        ;ADD FOR CHECKSUM
064D 08           EX AF,AF'      ;SAVE IN ALTERNATE AF
064E CD 57 06     CALL 0657      ;CALL OUT BYTE
0651 23           INC HL         ;POINT TO NEXT BYTE
```

```
0652 10 F6        DJNZ,064A     ;DO FOR ALL BYTES IN THE BLOCK
0654 08           EX AF,AF'     ;GET CHECKSUM
0655 3C           INC A         ;INCREASE IT BY ONE
0656 5F           LD E,A        ;PUT IT IN E

THIS ROUTINE OUTPUTS A SINGLE BYTE IN E TO THE TAPE. THE FORMAT IS 1 START BIT, EIGHT
DATA BITS AND 1 STOP BIT.

0657 16 08        LD D,08       ;SET D FOR 8 BITS
0659 B7           OR A          ;CLEAR CARRY AND CALL OUTBIT
065A CD 66 06     CALL 0666     ;TO OUTPUT BINARY ZERO FOR START BIT
065D CB 1B        RR E          ;PUT FIRST BIT IN CARRY
065F CD 66 06     CALL 0666     ;CALL OUT BIT
0662 15           DEC D         ;
0663 20 F8        JR NZ,065D    ;DO FOR 8 BITS
0665 37           SCF           ;SET CARRY TO OUTPUT STOP BIT (1)

THIS ROUTINE OUTPUTS A SINGLE BIT. IF THE CARRY IS SET, THEN A LOGIC 1 IS OUTPUTTED
OTHERWISE A LOGIC 0.
A 1 IS REPRESENTED BY 2 SHORT AND 2 LONG PERIODS.
A 0 IS REPRESENTED BY 4 SHORT PERIODS AND 1 LONG PERIOD.
L IS LOADED WITH DOUBLE THE LOW SPEED CYCLE COUNT AS IT IS USED TO COUNT THE HALF CYCLES
IN THE TONE ROUTINE. IF THE HIGH SPEED SAVE IS SELECTED, THEN THE CYCLE COUNT WILL BE
HALVED IN THE TONE ROUTINE.

0666 D9           EXX           ;SWAP REGISTERS
0667 26 00        LD H,00       ;ZERO H
0669 38 09        JR C,0674     ;JUMP IF BINARY 1 IS TO BE OUTPUTTED
066B 2E 10        LD L,10       ;LOAD L WITH HIGH TONE CYCLE COUNT
066D CD 84 06     CALL 0684     ;CALL HIGH TONE
0670 2E 04        LD L,04       ;LOAD L WITH LOW TONE CYCLE COUNT
0672 18 07        JR 067B       ;JUMP TO LOW TONE
0674 2E 08        LD L,08       ;LOAD L FOR HIGH TONE CYCLE COUNT
0676 CD 84 06     CALL 0684     ;FOR BINARY ONE: CALL HIGH TONE
0679 2E 08        LD L,08       ;LOAD L FOR LOW TONE CYCLE COUNT
067B CD 80 06     CALL 0680     ;CALL LOW TONE
067E D9           EXX           ;SWAP BACK REGISTERS
067F C9           RET           ;DONE

SET-UP FOR LOW TONE (LONG PERIOD)

0680 0E 29        LD C,29       ;LOAD C FOR LOW TONE
0682 18 02        JR 0686       ;JUMP TO TONE ROUTINE

SET-UP FOR HIGH TONE (SHORT PERIOD)

0684 0E 11        LD C,11       ;LOAD C FOR HIGH TONE

TONE ROUTINE
TESTS FOR LOW SPEED SAVE. IF SO THEN IT HALVES THE CYCLE COUNT IN L.

0686 3A 8F 08     LD A,(088F)   ;FIND WHICH SPEED
0689 B7           OR A          ;ZERO = HIGH SPEED
068A 20 02        JR NZ,068E    ;JUMP IF LOW SPEED
068C CB 3D        SRL L         ;ELSE HALVE CYCLE COUNT
068E 11 01 00     LD DE,0001    ;
0691 3E 84        LD A,84       ;TURN ON SPEAKER AND MIDDLE DISPLAY
0693 D3 01        OUT (01),A    ;
0695 41           LD B,C        ;
0696 10 FE        DJNZ,0696     ;PERIOD DELAY
0698 EE 80        XOR 80        ;TOGGLE SPEAKER BIT
069A ED 52        SBC HL,DE     ;DECREASE CYCLE COUNT
069C 20 F5        JR NZ,0693    ;JUMP IF NOT ALL CYCLES DONE
069E C9           RET           ;ELSE RETURN

THIS ROUTINE SETS UP THE "ERR-IN DISPLAY ON THE PERIMETER HANDLER.

069F 21 52 07     LD HL,0752    ;POINT HL TO "Err-In" DISPLAY
06A2 11 00 08     LD DE,0800    ;CODE AND DE TO RAM DEstination
06A5 01 06 00     LD BC,0006    ;BC(ount)
06A8 ED B0        LDIR          ;MOVE BLOCK
06AA C3 50 00     JP 0050       ;JUMP TO SOFT PERIMETER ENTRY

----END OF TAPE ROUTINES----
```

THIS ROUTINE IS THE KEYBOARD READER/VALIDATER
THE ACTION IS AS FOLLOWS:
A SHORT LOOP LOOKS FOR A KEY PRESS. IF NO KEY IS PRESSED, THEN THE KEY PRESS BUFFER (0825) IS CLEARED THE ZERO AND THE CARRY FLAG CLEARED AND THE ROUTINE RETURNS.
IF A KEY IS FOUND, THEN THE REMAINING LOOP COUNTS ARE WORKED OFF IN A DUMMY LOOP TO ENSURE EQUAL TIME IN EXECUTING THE ROUTINE.
IF IT IS THE FIRST TIME THAT THE KEY HAS BEEN DETECTED, THEN THE KEY PRESS FLAG WILL BE CLEAR.(IT WAS CLEARED BY THE MONITOR VARIABLES ON RESET). THE ROUTINE TESTS FOR THIS CONDITION AND IF TRUE THEN THE KEY IS ACCEPTED AS "VALID" AND FLAGGED BY A SET CARRY AND SET ZERO FLAG AND THE KEY PRESS FLAG IS SET TO INDICATE THE A KEY HAS BEEN DETECTED. THE INPUT IS THEN PLACED IN BOTH THE "I" REGISTER AND THE ACCUMULATOR. IF A KEY IS DETECTED BUT FOUND NOT TO BE VALID, I.E. IT HAS ALREADY BEEN DETECTED AND PROCESSED, THEN THE CARRY WILL BE SET BUT THE ZERO CLEARED. THIS ALLOWS THE AUTO KEY REPEAT SECTION TO KNOW THAT A KEY IS STILL BEING HELD DOWN. THE AUTO KEY REPEAT SECTION MAKE UP ITS OWN MIND WHETHER IT IS VALID OR NOT.

```
06AD DB 03        IN A,(03)      ;TEST FOR KEY PRESSED
06AF CB 77        BIT 6,A        ;
06B1 28 08        JR Z,06BB      ;JUMP IF KEY PRESSED
06B3 10 F8        DJNZ,06AD      ;LOOP LOOKING FOR KEY UNTIL B=0
06B5 AF           XOR A          ;CLEAR KEY PRESS FLAG
06B6 32 25 08     LD (0825),A    ;
06B9 3D           DEC A          ;SET A TO FF AND CLEAR ZERO FLAG
06BA C9           RET            ;DONE
06BB 3A 25 08     LD A,(0825)    ;GET KEY PRESS FLAG
06BE B7           OR A           ;TEST FOR ZERO
06BF 20 00        JR NZ,06C1     ;DUMMY JUMP TO EQUALIZE TIME
06C1 10 F8        DJNZ,06BB      ;FINISH LOOP
06C3 37           SCF            ;SET CARRY
06C4 20 F4        JR NZ,06BA     ;DUMMY JUMP TO RETURN
06C6 3D           DEC A          ;SET KEY PRESS FLAG TO FF
06C7 32 25 08     LD (0825),A    ;
06CA DB 00        IN A,(00)      ;GET INPUT KEY FROM ENCODER CHIP
06CC E6 1F        AND 1F         ;MASK OFF UNUSED BITS
06CE CB 7F        BIT 7,A        ;SET ZERO FLAG (THINK ABOUT IT!)
06D0 37           SCF            ;SET CARRY
06D1 32 20 08     LD (0820),A    ;STORE INPUT KEY
06D4 C9           RET            ;DONE
```

THIS ROUTINE IS CALLED ONCE ON EVERY HARD RESET. IT INITIALIZES THE LCD THEN TESTS THAT IT IS THERE (IT CANNOT DO IT THE OTHER WAY AROUND AS THE LCD NEEDS TO BE INITIALIZED BEFORE IT WILL RESPOND INTELLIGENTLY). IF THE LCD IS FITTED THEN THE ROUTINE WILL READ IN AN ASCII SPACE CHARACTER (20H) OR IF THE LCD IS NOT, JUNK FROM THE DATA BUSS.
20H IS SUBTRACTED FROM WHATEVER IS READ IN AND THE RESULT IS STORED IN THE LCD ENABLE BUFFER. IF THE RESULT IS ZERO THEN THE LCD IS ENABLED. IT IS VITAL TO KNOW IF THE LCD IS FITTED, OTHERWISE THE ROUTINE WHICH READS THE BUSY FLAG MAY LOOP FOREVER.

```
06D5 21 B5 07     LD HL,07B5     ;POINT HL TO LCD INITIALIZE TABLE
06D8 01 04 04     LD BC,0404     ;B=4 BYTES, C=PORT 4
06DB 11 00 05     LD DE,0500     ;DELAY BETWEEN
06DE 1B           DEC DE         ;EACH BYTE
06DF 7A           LD A,D         ;AS PER
06E0 B3           OR E           ;LCD MANUFACTER'S
06E1 20 FB        JR NZ,06DE     ;INSTRUCTIONS
06E3 ED A3        OUTI           ;OUTPUT (HL) TO (C). HL=HL=1,B=B-1
06E5 20 F4        JR NZ,06DB     ;JUMP IF B NOT 0
06E7 10 FE        DJNZ,06E7      ;SHORT DELAY
06E9 DB 84        IN A,(84)      ;INPUT FROM LCD TO SEE IF IT'S THERE
06EB D6 20        SUB 20         ;SUBTRACT ASCII SPACE, IF LCD FITTED
06ED 32 21 08     LD (0821),A    ;RESULT WILL BE ZERO: STORE THIS IN
06F0 C9           RET            ;LCD MASK: DONE
06F1 FF           RST 38         ;
06F2 FF           RST 38         ;
06F3 FF           RST 38         ;
06F4 FF           RST 38         ;
06F5 FF           RST 38         ;
06F6 FF           RST 38         ;
06F7 FF           RST 38         ;
06F8 FF           RST 38         ;
06F9 FF           RST 38         ;
06FA FF           RST 38         ;
06FB FF           RST 38         ;
06FC FF           RST 38         ;
06FD FF           RST 38         ;
06FE FF           RST 38         ;
06FF FF           RST 38         ;
```

JMON'S TABLES PAGE

AT 0700 IS THE TAPE'S MENU JUMP TABLE.

0700 C3 3F 04	HIGH SPEED SAVE	
0703 C3 3F 04	LOW SPEED SAVE	
0706 C3 3B 04	TEST BLOCK	
0709 C3 37 04	TEST CHECKSUM	
070C C3 26 04	LOAD TAPE	

BELOW ARE THE JMON DEFAULT RESET VARIABLES (A ZERO IS THE ACTIVE STATE UNLESS OTHERWISE STATED).

		RAM LOCATION
070F 00	KEY BUFFER	0820
0710 00	LCD ON/OFF FLAG	0821*
0711 00	SOUND ON/OFF	0822*
0712 FF	GO AT ALTERNATE GO ADDRESS IF AA	0823*
0713 FF	STEPPER KEY CONTROL/TIMER	0824
0714 00	KEY PRESSED FLAG	0825
0715 FF	UNUSED	0826
0716 00	AUTO INCREMENT ON/OFF	0827*
0717 00 0A	ALT GO ADDR/SOFT RESET EDIT LOCATION	0828*
0719 70	AUTO KEY REPEAT TIMER	082A
071A 00	MONITOR CONTROL BYTE	082B
071B 00 08	DISPLAY BUFFER ADDRESS	082C*
071D 00 09	INITIAL EDITING LOCATION	082E

BELOW ARE THE JMON INDIRECT JUMP ADDRESSES. THIS TABLE IS SHIFTED DOWN TO 0830 ON A HARD RESET.

071F C3 D5 01	CONVERT HL TO DISPLAY CODE	0830
0722 C3 DA 01	CONVERT A TO DISPLAY CODE	0833
0725 C3 BA 01	LED SCAN ROUTINE	0836
0728 C3 EE 01	SET LED DOTS	0839
072B C3 24 02	RESET TONES	083C
072E C3 27 02	TONE	083F
0731 C3 81 01	SCAN/KEY/LCD/PATCH LOOP	0842
0734 C3 B2 00	SOFT JMON ENTRY	0845
0737 C3 3C 02	LCD ROUTINE	0848

BELOW ARE THE DISPLAY TABLES FOR THE TAPE'S MENU ADDRESS DISPLAYS AND THE "ERR-IN" DISPLAY THAT IS SUPERIMPOSED OVER THE PERIMETER HANDLER.

073A A7 6F EA C7	"SAVE"
073E A7 6F EA C7	"SAVE"
0742 C6 C7 A7 C6	"TEST"
0746 C6 C7 A7 C6	"TEST"
074A C2 EB 6F EC	"LOAD"
074E 04 C7 64 EC	"-End"
0752 04 C7 44 44 28 64	"-Err In"
0758 4F 6F A7 A7	"PASS'
075C 47 6F 28 C2	"FAIL

BELOW ARE THE TAPE'S MENU DATA DISPLAYS.

0760 04 6E	"-H"
0762 04 C2	"-L"
0764 E6 C2	"bL"
0766 C3 A7	"CS"
0768 04 C6	"-t"
076A 04 A7	"-S"
076C C6 E6	"tb"
076E C3 A7	"CS"
0770 C2 EC	"Ld"

0772 - 077B (UNUSED)

* DENOTES CONTROL BYTES DESIGNED TO BE USER ALTERED (IN RAM).

BELOW IS THE PERIMETER HANDLER COMMAND STRING FOR THE TAPE SOFTWARE.

077C 00 FF C6 07 99 08 00 03 (FF FF; THE JUMP ADDRESS FOR THE TAPE ROUTINES IS SUPPLIED BY THE POST MENU SET-UP ROUTINES, SEE 0426-044E).

0786 - 0788 FF (RESERVED FOR COMMAND STRING EXPANSION).

BELOW IS THE TAPE'S MENU DRIVER COMMAND STRING.

0789 FF FF 00 04 07 3A 07 60 07

TAPE'S SOFTWARE MENU DATA KEY HANDLER ROUTINE JUMP VECTOR (A RETURN INSTRUCTION).

0793 C9

BELOW IS THE STEPPERS DATA DISPLAY CODES.

```
0794 4F C3              "PC"
0796 6F 47              "AF"
0798 E6 C3              "BC"
079A EC C7              "dE"
079C 6E C2              "HL"
079E 28 6E              "IX"
07A0 28 AE              "IY"
07A2 7F 57              "AF'"
07A4 F6 D3              "BC'"
07A6 FC D7              "dE'"
07A8 7E D2              "HL'"
07AA A7 4F              "SP"
```

07AC FF (UNUSED)

START OF STAGGERED TABLE OF JMON MODE WORDS FOR LCD

```
07AD 44 61 74 61        "Data"
07B1 41 64 64 72        "Addr"
```

LCD INITIALIZATION CODES

07B5 38 01 06 0C

THE REST OF THE JMON MODE WORD TABLE FOR LCD

```
07B9 46 73 2D            "Fs-"
```

07BC FF (UNUSED)

ADDRESS TABLE OF THE LCD PROMPT LOCATIONS.

07BD 84 87 8A 8D C4 C7 CA CD 80

TAPE'S PERIMETER HANDLER DATA DISPLAYS

```
07C6 04 47              "-F"
07C8 04 A7              "-S"
07CA 04 C7              "-E"
07CC 04 E3              "-G"
```

07CD - 07CF FF (UNUSED)

BELOW ARE THE DISPLAY CODE EQUIVALENTS OF THE HEX DIGITS 0 TO F LISTED IN ASCENDING ORDER.

07D0 EB 28 CD AD 2E A7 E7 29 EF 2F 6F E6 C3 EC C7 47

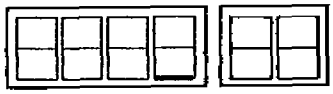
FINALLY AT 07E0 IS THE FUNCTION-1 AND SHIFT JUMP ADDRESSES.

07E0 D2 03 E3 02 5E 00 FF FF D3 02 AE 00 DE 02 41
07F0 ED 02 E8 02 F2 02 FF FF FF FF FF FF FF FF FF

SEGMENT TARGET GAME

By Mr. S Clarke, 2774

Segment Target is a simple game in which you must hit the moving segment in the bottom right of the address section. i.e.



Shoot when the highlighted segment is illuminated.

As each target is hit, the next one moves even FASTER! Any key can be used to shoot. Your score is stored at 08FF (in HEX)

****

SEGMENT TARGET, as presented below, has been written to run with the MON-1 series MONitors. By changing the LD A,I (ED 57) to RST 20/NOP (E7, 00) as described in the section on running old programs with JMON in issue 15, it will run equally as well with JMON.
Don't be content to just play SEGMENT TARGET GAME, see if you can improve on it!

-JIM

```
0900  11 00 38      LD DE,3800
0903  ED 53 A6 09   LD (09A6),DE
0907  3E 00         LD A,00
0909  32 FF 08      LD (08FF),A
090C  21 80 09      LD HL,0980
090F  7E            LD A,(HL)
0910  47            LD B,A
0911  23            INC HL
0912  7E            LD A,(HL)
0913  4F            LD C,A
0914  23            INC HL
0915  78            LD A,B
0916  FE FF         CP FF
0918  CA 6B 09      JP Z,096B
091B  D3 01         OUT (01),A
091D  79            LD A,C
091E  D3 02         OUT (02),A
0920  CD 2E 09      CALL 092E
0923  CD 3A 09      CALL 093A
0926  FE 12         CP 12
0928  CA 0C 09      JP Z,090C
092B  C3 0F 09      JP 090F
092E  ED 5B A6 09   LD DE,(09A6)
0932  1B            DEC DE
0933  7A            LD A,D
0934  FE 00         CP 00
0936  C8            RET Z
0937  C3 32 09      JP 0932
093A  ED 57/E7,00   LD A,I
093C  5F            LD E,A
093D  3E FF         LD A,FF
093F  ED 47         LD I,A
0941  7B            LD A,E
0942  FE FF         CP FF
0944  C8            RET Z
0945  78            LD A,B
0946  FE 04         CP 04
0948  C0            RET NZ
0949  79            LD A,C
094A  FE 80         CP 80
094C  C0            RET NZ
094D  3E 03         LD A,03
094F  D3 01         OUT (01),A
0951  3E FF         LD A,FF
0953  D3 02         OUT (02),A
0955  CD 2E 09      CALL 092E
0958  3A FF 08      LD A,(08FF)
095B  3C            INC A
095C  32 FF 08      LD (08FF),A
095F  ED 5B A6 09   LD DE,(09A6)
0963  15            DEC D
0964  ED 53 A6 09   LD (09A6),DE
0968  3E 12         LD A,12
096A  C9            RET
096B  11 00 BF      LD DE,BF00
096E  ED 53 A6 09   LD (09A6),DE
0972  3E FF         LD A,FF
0974  D3 01         OUT (01),A
0976  3E 85         LD A,85
0978  D3 02         OUT (02),A
097A  CD 2E 09      CALL 092E
097D  C7            RST 00

0980  20 01 10 01 08 01 04 01
0988  04 08 04 04 08 04 10 04
0990  20 04 20 40 20 80 10 80
0998  08 80 04 80 02 80 01 80
09A0  FF
```

WHIRL

by Jeff Kennett 3218

This clever routine for the 8x8 display continuously rotates the display around 90 degrees and produces quite an interesting effect. After a while the eyes are fooled and it begins to look like anything other than a rotating arrow head. One staff member thought it looked like a plus sign trying to rap dance!!
Experiment with the values in the table at 0A00 and the delay at 0927/8 to see what dazzling effects you can produce!

```
0900  CD 27 09      CALL 0927
0903  11 08 0A      LD DE,0A08
0906  06 08         LD B,08
0A08  C5            PUSH BC
0909  06 08         LD B,08
090B  21 00 0A      LD HL,0A00
090E  AF            XOR A
090F  CB 06         RLC (HL)
0911  1F            RRA
0912  23            INC HL
0913  10 FA         DJNZ 090F
0915  12            LD (DE),A
0916  13            INC DE
0917  C1            POP BC
0918  10 EE         DJNZ 0908
091A  01 08 00      LD BC,0008
091D  11 00 0A      LD DE,0A00
0920  21 08 0A      LD HL,0A08
0923  ED B0         LDIR
0925  18 D9         JR 0900
0927  06 50         LD B,06
0929  C5            PUSH BC
092A  06 80         LD B,80
092C  21 00 0A      LD HL,0A00
092F  7E            LD A,(HL)
0930  D3 05         OUT (05),A
0932  78            LD A,B
0933  D3 06         OUT (06),A
0935  06 40         LD B,40
0937  10 FE         DJNZ 0937
0939  47            LD B,A
093A  AF            XOR A
093B  D3 06         OUT (06),A
093D  23            INC HL
093E  CB 08         RRC B
0940  30 ED         JRNC 092F
0942  C1            POP BC
0943  10 E4         DJNZ 0929
0945  C9            RET

0A00: 18 30 60 FF FF 60 30 18
```

HEX TO BCD CONVERSION

By James Doran 3259

This SUB-ROUTINE will convert a hex number in A into its decimal equivalent and store the result in BC.
The hex number is held in A on entry.
The routine works by counting up in decimal while counting down the HEX number until zero.
This means that low numbers are converted quickly while larger numbers take longer.
The decimal counter is achieved by the use of the DECIMAL ADJUST ACCUMULATOR (DAA) instruction.

```
0900  06 00         LD B,00
0902  4F            LD C,A
0903  3E 00         LD A,00
0905  3C            INC A
0906  27            DAA
0907  30 02         JR NC,+2
0909  04            INC B
090A  3F            CCF
090B  0D            DEC C
090A  20 F7         JR NZ,-9
090C  4F            LD C,A
090D  C9            RET
```

Exit: BC = packed BCD equivalent of two hex digits in A.
The above routine is useful as a HEX to BCD conversion SUB-ROUTINE, but keep in mind the disadvantage of the length of time being very dependent on the magnitude of the HEX number to be converted.