WISCONSIN COMPUTER SOCIETY
P.O. Box 159
Sheboygan Falls, WI 53085

FIRST CLASS
LETTER MAIL

WISCONSIN COMPUTER SOCIETY

NEWSLETTER

****

Volume 3, Issue #4 April 1978 Don Stevens, Editor

****

MEETING NOTICE

The monthly meeting of the Wisconsin Computer Society will be held this coming Saturday, APRIL 1, 1978. It will be held at our usual meeting place - WAUKESHA TECHNICAL INSTITUTE (Room 202 - Administration Building).

PROGRAM AGENDA

Club members should be informed that CP/M is quickly becoming the standard for 8080 and Z-80 Disc based systems. The CP/M users group now has a library of over 30 floppy discs of public domain software. DON STEVENS will demonstrate his 8080 based system running under CP/M. Discussion of club interests will follow.

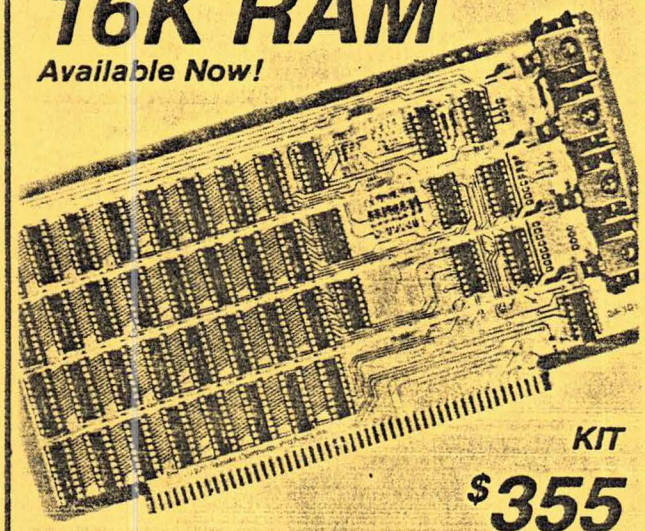
NEWSLETTER

The Newsletter has a new format this month. The Editor is still looking for articles of interest to publish.

Fully Static — 250 nsec. TMS 4044-25

16K RAM

Available Now!



KIT

$355

Assembled, tested unit — $375

Z-80A 4Mhz. Fast

Our memory board was designed to operate without wait states in a 4 Mhz. Z-80A system and allows a generous 100 nsec. for the CPU board buffers. Our board "loafs along" in an 8080 or 8085 system. Even if you are using a slower CPU today, don't get caught buying a memory board which may become obsolete if you decide to switch to a faster, more sophisticated CPU tomorrow.

Fully Static is Best

Our board uses the state-of-the-art Texas Instrument TMS 4044-25. It needs no clocks and no refresh. It uses a single 8V power supply and won't be obsoleted when you buy the next generation system using a single power supply.

Fully S-100 Bus Compatible

Each 4K addressable to any 4K slot, on-board DIP switch memory protect, RAM disable, DMA capability.

Commercial Quality Components

First quality factory parts, fully socketed, buffered, masked both sides, silk-screened, gold contacts, bus bars for lower noise.

Guaranteed

ASSEMBLED UNITS: if unsatisfied for any reason — return undamaged unit within 10 days for full refund. Parts and labor guaranteed for one year.

KITS: MOS parts factory tested good — no free replacements. All other parts guaranteed one year.

Shipping

If we cannot ship within four weeks we will phone for instructions, returning money if you desire.

This 16K RAM boards is available to members at about $325.00 depending on size of group purchase. Contact Don Stevens.

FOR SALE:

FLOPPY DISC SYSTEMS with or without S-100 Interface Board. Contact Don Stevens

GROUP PURCHASE

FLOPPY DISCS & Mini-FLOPPY DISCS available. Top Quality product. If interested, contact Don Stevens

FOR SALE

Processor Technology 3P+S Board. Still in original carton. Contact Don Stevens

FOR SALE

IBM 632 Electronic Typing Calculator. Best offer - Contact Will Piette at 246-6634 after 6 p.m.

WHAT WOULD YOU LIKE TO

BUY or SELL

Free listing to club members,etc.

This article "SOFTWARELAND" is reprinted courtesy of the San Diego Computer Society January 1978 issue of PERSONAL SYSTEMS

SOFTWARELAND

by Dick Lindberg

One of the strengths of a board like the Processor Tech VDM-1 is the ability to randomly access any spot on the screen. Often, however, it is run merely as a terminal substitute, outputting onto the bottom line and moving the lines up the screen. While it certainly works very well this way, it is a waste of the board's potential. Think how nice it would be if important messages and data didn't disappear off the top of the screen while the screen filled up with trivia. A very sophisticated looking display can be made qhich simplifies the operation of the program and the display of the results.

In order to use the VDM-1 in this manner, it is necessary to have two functions: 1) display a message on the screen and 2) accept a message from the keyboard displaying it on the screen. This second one is the hardest, so that is the one we will do this month.

The following subroutine has the following functions:

1. It clears the work area in the calling program to spaces.
2. It clears the screen area where the keyboard message will appear to spaces.
3. It puts an inverted space as a cursor in the first screen position.
4. It accepts characters from the keyboard until a specific number of characters have been entered or a carriage return entered.
5. If a "rubout" is entered, the cursor is moved back and the last character entered is replaced with a space.
6. When the message is complete, the cursor is removed from the screen and the contents of the screen transfered to the program work area.

Register settings are:

H-L = program work area address
D = screen line number
E = screen character position
B-C = maximum characters in the message

The pseudocoding for the subroutine shows how much the subroutine actually does.

```
CLEAR MESSAGE AREA
CALCULATE ACTUAL SCREEN STARTING ADDRESS
CLEAR SCREEN AREA
DO GET A CHARACTER
   (UNTIL MAXIMUM CHARACTERS ENTERED
   OR CARRIAGE RETURN ENTERED)
      IF B-C = 0
         SET MAX CHARACTERS INDICATOR
      ELSE
         PUT CURSOR ON SCREEN
         GET A CHARACTER FROM KEYBOARD
         IF CARRIAGE RETURN
            SET MAX CHARACTERS INDICATOR
         ELSE
            IF RUBOUT
               BACKSPACE
            ELSE
               PUT CHARACTER ON SCREEN
            ENDIF
         ENDIF
         CLEAR CURSOR
         MOVE TO NEXT POSITION
      ENDIF
ENDDO
MOVE SCREEN AREA TO MESSAGE AREA
```

Here is the coding of the subroutine.

```
ØØ4Ø                     ORG    4ØH
ØØ4Ø   2152ØØ            LXI    H,MSG
ØØ43   11Ø5Ø5            LXI    D,Ø5Ø5H
ØØ46   Ø1ØAØØ            LXI    B,1Ø
ØØ49   CD5CØØ            CALL   AFC
ØØ4C   C34CØØ   JHLT:    JMP    JHLT
ØØ4F   C3ØØB8            JMP    ØB8ØØH
ØØ52            MSG:     DS     1Ø
```

This is the test program. It shows the setting of the registers. The input will be 10 characters long and displayed on line 5, position 5, and put into 'MSG'.

```
ØØ5C   E5        AFC:    PUSH   H
ØØ5D   C5                PUSH   B
ØØ5E   D5                PUSH   D
ØØ5F   3EØØ              MVI    A,Ø
ØØ61   D38C              OUT    8CH
ØØ63   362Ø              MVI    M,2ØH
ØØ65   ØB                DCX    B
ØØ66   E5                PUSH   H
ØØ67   Dl                POP    D
ØØ68   13                INX    D
ØØ69   CD95ØØ            CALL   MVC
```

This is the beginning of the subroutine. First the register contents are saved for later use. Next, the screen is reset. I set up my board before the standard output port was changed to C8. The last part moves spaces to the message area. Note the 'PUSH H, POP D' to move H-L to D-E.

```
ØØ6C   Dl                POP    D
ØØ6D   CDA2ØØ            CALL   CLNA
ØØ7Ø   Cl                POP    B
ØØ71   C5                PUSH   B
ØØ72   E5                PUSH   H
ØØ73   ØB                DCX    B
ØØ74   E5                PUSH   H
ØØ75   Dl                POP    D
ØØ76   13                INX    D
ØØ77   362Ø              MVI    M,2ØH
ØØ79   CD95ØØ            CALL   MVC
```

This section moves spaces to the screen area to be used. The routine 'CLNA' calculates the real memory address from the line number and character position in D-E.

```
ØØ7C   El                POP    H
ØØ7D   Cl                POP    B
ØØ7E   C5                PUSH   B
ØØ7F   E5                PUSH   H
```

Register values are reset and saved on the stack. As written, this routine uses the stack for almost all its work area.

```
ØØ8Ø   3AB2ØØ    AFCl:   LDA    EOM
ØØ83   FEØØ              CPI    Ø
ØØ85   CA8EØØ            JZ     AFC2
ØØ88   CDB3ØØ            CALL   CHAR
ØØ8B   C38ØØØ            JMP    AFCl
```

The subroutine will remain in this loop until the flag 'EOM' goes to zero.

```
ØØ8E   El        AFC2:   POP    H
ØØ8F   Cl                POP    B
ØØ9Ø   Dl                POP    D
ØØ91   CD95ØØ            CALL   MVC
ØØ94   C9                RET
```

Here the screen is moved to the message area. An additional instruction should be added at this point to reset 'EOM' so that the subroutine will work the second time it is executed.

```
ØØ95   78        MVC:    MOV    A,B
ØØ96   Bl                ORA    C
ØØ97   C8                RZ
ØØ98   ØB                DCX    B
ØØ99   7E                MOV    A,M
ØØ9A   EB                XCHG
ØØ9B   77                MOV    M,A
ØØ9C   EB                XCHG
ØØ9D   23                INX    H
ØØ9E   13                INX    D
ØØ9F   C395ØØ            JMP    MVC
```

This moves a block of memory from one location to another. This coding is a little different from the way I have done it before, but the principle is the same.

```
ØØA2  7A        CLNA:   MOV     A,D
ØØA3  ØF                RRC
ØØA4  ØF                RRC
ØØA5  E6CØ              ANI     ØCØH
ØØA7  83                ADD     E
ØØA8  6F                MOV     L,A
ØØA9  7A                MOV     A,D
ØØAA  ØF                RRC
ØØAB  ØF                RRC
ØØAC  E6Ø3              ANI     Ø3H
ØØAE  F688              ORI     88H
ØØBØ  67                MOV     H,A
ØØBl  C9                RET
```

This calculates the real address starting from a line number in register D and a character position in register E. The 2 low order bits from register D are put into the high order bits of register L. The value in register E is put in the 6 low order bits of register L. The value in register D is moved right 2 places and added to 88H. If your board's memory is set to CC00H, use CCH.

```
ØØB2  Øl        EOM:    DB      1
ØØB3  78        CHAR:   MOV     A,B
ØØB4  B1                ORA     C
ØØB5  C2BFØØ            JNZ     CHRl
ØØB8  97                SUB     A
ØØB9  32B2ØØ            STA     EOM
ØØBC  C3E2ØØ            JMP     CHEX
```

If B-C is zero then 'EOM' is set to zero. The rest of the routine is bypassed.

```
ØØBF  36AØ      CHRl:   MVI     M,ØAØH
ØØCl  CDØØBC            CALL    ØBCØØH
```

A0H is an inverted space. BC00H is my keyboard input routine.

```
ØØC4  FEØD              CPl     ØDH
ØØC6  C2DØØØ            JNZ     CHR2
ØØC9  97                SUB     A
ØØCA  32B2ØØ            STA     EOM
ØØCD  C3DCØØ            JMP     CHR4
```

If the returned character is a carriage return, 'EOM' is cleared.

```
ØØDØ  FE7F      CHR2:   CPI     7FH
ØØD2  C2DBØØ            JNZ     CHR3
ØØD5  CDE3ØØ            CALL    BKSP
ØØD8  C3DCØØ            JMP     CHR4
```

If the returned character is a rubout, backspace the screen.

```
ØØDB  77        CHR3:   MOV     M,A
                         ;
ØØDC  7E        CHR4:   MOV     A,M
ØØDD  E67F              ANI     7FH
ØØDF  77                MOV     M,A
ØØEØ  23                INX     H
ØØEl  ØB                DCX     B
```

The returned character is put on the screen. The inversion bit is removed and the registers are updated.

```
ØØE2  C9        CHEX:   RET
                         ;
ØØE3  C9        BKSP:   RET
ØØØØ                    END
```

The backspace routine is not included. We will go over it in the next column. We will also add some refinements to make the display even more professional.

Proposed Input/Output Standard

by

Proposed Input/Output Standard
by Ray James and Jim Matthews

A standard in programming input/output is needed. This standard should be one that a maximum number of micro-computer users can utilize. These users have their R.O.M.s at different locations and they are of different lengths. A standard that does not dedicate a fixed area of R.A.M. is necessary.

A standard that would probably work for most of these users is to have an I/O jump table in their program. This jump table would have jumps to the addresses of the I/O routines. In a program, you would call the jump that goes to the I/O routine you want to use. See figure 1 for an example of this.

When a program is received from someone, you go into his jump table, which is plainly marked in the program listing, and change his jump addresses to the addresses of your I/O routines. You can now load his program and run.

The data, that you want to output, is put in the 'A' register then you call the jump. For an input you call the jump and when the program returns the data input is in the 'A' register. For X-Y coordinates you put them in the 'D' & 'E' registers (X in 'D', Y in 'E').

If enough people use this standard, you will be able to trade programs with each other. Then you will not have to reinvent the wheel each time you want a program.

If you have any suggestions or questions, contact me at 449-1813 or jim matthews, P.O. Box 699, Santee, Ca. 92071.