John M. Blalock
3054 West Evans Dr.
Phoenix AZ 85023

A Printer for the KIM or SYM

The ubiquitous Selectric finds yet another home.

After I expanded my KIM-1 with a CRT terminal, 28K of additional RAM and Micro-Z's Microsoft BASIC ("Another KIM-1 Expansion," September 1979, p. 130), I felt the need to add hard-copy printout to the system. This article describes how I accomplished this goal and interfaced to my new SYM-1 as well.

The Selectric

Various models of IBM Selectric I/O typewriters are currently being removed from service and made available to the hobbyist for very reasonable prices. I was able to obtain a surplus Model 2970 Communications Terminal a short time ago for only $100! The reasonable price and high-quality printing available made it the printer of my choice. There was only one minor problem: I'd have to design the interface between the printer and the KIM.

The 2970 consisted of a power supply, logic assembly, cabinet and a Model 735 Selectric I/O typewriter. The Selectric contained all of the selector magnets (solenoids, for those who don't speak IBMese!) for tilt, rotate, index, carriage return and non-print (space).

I stripped the electronics out of the cabinet but left the power supply. Ac power to the supply and the Selectric is controlled by a circuit breaker on the cabinet, which is a convenient base for the Selectric to sit on. The remaining interface requirement was to ground the proper leads from the selector magnets with the correct coding and timing. The KIM, using its on-board timer and I/O capability, should easily handle these functions.

Interfacing Options

There are two ways to interface to a Selectric I/O—open loop or closed loop. With a closed loop system, signals generated by the Selectric mechanism are used to control the character timing. This is the preferred interface if heavy use is planned, but is somewhat more difficult to accomplish.

An open loop interface always must provide the worst-case timing. A character is sent out, and then the interface waits for the longest time it will take to print any character before sending another. The IBM manuals state that either method can be implemented successfully. However, open loop timing will slow down the throughput by a small amount.

The main disadvantage of the open loop system, in my opinion, is that the cycle clutch must stop the machine after each character. In a closed loop system, the next character can be transmitted prior to the clutch action, thus saving machine wear and tear. A closed loop system requires that the feedback contacts be carefully adjusted and that additional control lines be connected from the Selectric to the interface electronics.

The unexpanded KIM-1 only has 15 I/O lines available, and some of these on my KIM were already in use. I came up with the circuit shown in Fig. 1 to interface the KIM to the Selectric using only eight lines or one port address.

This is an open loop type of interface, but since I'm not planning on printing reams of paper daily, I don't expect the open loop disadvantages to be as important as the requirement for minimum I/O lines, minimum hardware and overall simplicity. It has been working satisfactorily now for several months. My biggest problem in getting it running was a misadjusted escapement mechanism inside the machine.

Implementing the Interface

I highly recommend that anyone attempting to interface or use a Selectric I/O typewriter obtain copies of IBM manuals 225-6595-1 and 225-1726-6. Along with the spare parts manual 123-1008-3, they contain a wealth of information on the machine's operation. Almost any

Listing 1. KIM interface program.

```
                        ;
                        ;  KIM SELECTRIC DRIVER
                        ;
                        ;  Routine to convert ASCII in accumulator
                        ;  to Selectric codes, and send to Selectric with
                        ;  proper timing, case, etc. if location 0400 hex
                        ;  is not equal to zero. Returns to KIM routine OUTCHR
                        ;  with all registers unchanged from entry.  Uses no
                        ;  page zero locations.
                        ;
0400:08         PAD:    equ $1700       ;  Output port to Selectric
0401:48         PADD:   equ $1701       ;  Data direction register
0402:A0 00 04   Timer:  equ $1707       ;  Divide by 1024
0405:A5         OUTCHR: equ $1EA0       ;  routine to send ASCII to CRT
0406:08         TypeOn: equ $0400       ;  Zero = don't type
0407:48         ASCII:  equ $0401       ;  Temporary
0408:A0 00 04   CRflag: equ $0402       ;  Storage
040B:A5         Temp:   equ $0403       ;    buffer
040C:68         SFTflag: equ $0405      ;    locations
040D:68                 org $0406       ;
040E:28         Start:  PHP             ;  Start of program
040F:A0 00 04           PHA             ;  Save registers
0412:A9 FF              LDA TypeOn      ;  Check if typing desired
0414:8D 01 17           BNE Type        ;  Yes, then Type
0417:A5                 PLA             ;  No, restore registers
0418:A0 00 04           PLP             ;
041B:28                 JMP OUTCHR      ;  and send to CRT
041C:4C A0 1E   Type:   LDA #$FF        ;  Set port to output
041F:A9 FF              STA PADD        ;
0421:8D 01 17
```

```
0417:68          29            PLA          ;  Get ASCII code
0418:8D 01 04    30            STA ASCII    ;  Save it
041B:48          31            PHA          ;  Save registers
041C:8A          32            TXA          ;
041D:48          33            PHA          ;
041E:98          34            TYA          ;
041F:48          35            PHA          ;
0420:D8          36            CLD          ;  Binary mode
0421:AD 01 04    37            LDA ASCII    ;  Get ASCII code
0424:C9 20       38            CMP #$20     ;  Is it a space?
0426:F0 17       39            BEQ Xspace   ;  Yes
0428:10 31       40            BPL GetCode  ;  Greater than space = valid code
042A:C9 0D       41            CMP #$0D     ;  Less than space, is it Return?
042C:D0 15       42            BNE CKLF     ;  no, go check if is a line feed
042E:A9 04       43            LDA #$04     ;  yes, carriage return
0430:8D 02 04    44            STA CRflag   ;  Set flag to non-zero
0433:A8          45            TAY          ;  Will send four characters
0434:A9 00       46            LDA #$00     ;  but next three characters will
0436:8D 03 04    47            STA Temp     ;  be nulls to allow for return
0439:A9 40       48            LDA #$40     ;  Carriage return code
043B:A2 7F       49            LDX #$7F     ;  long delay
043D:D0 57       50            BNE CharOut  ;  Send the carriage return
043F:A9 81       51 Xspace:    LDA #$81     ;  Space code
0441:D0 45       52            BNE Out1     ;  Send it
0443:C9 0A       53 CKLF:      CMP #$0A     ;  Is code a line feed?
0445:D0 72       54            BNE Return   ;  No, ignore it
0447:AD 02 04    55            LDA CRflag   ;  Check flag
044A:F0 07       56            BEQ Linefeed ;Send LF if last not  Return
044C:A9 00       57            LDA #$00     ;  Last character sent was Return
044E:8D 02 04    58            STA CRflag   ;  reset the flag
0451:F0 66       59            BEQ Return   ;  and ignore the line feed
0453:A0 01       60 Linefeed:  LDY #$01     ;  one character
0455:A2 7F       61            LDX #$7F     ;  long delay
0457:A9 20       62            LDA #$20     ;  line feed code
0459:D0 3B       63            BNE CharOut  ;  send the line feed
045B:29 7F       64 GetCode:   AND #$7F     ;  mask off high bit
045D:AA          65            TAX          ;  put in index X
045E:BD A1 04    66            LDA Codes-$21,X  ;  Get code in accumulator
0461:0A          67            ASLA         ;  shift left, put case in carry
0462:90 11       68            BCC LCchar   ;  Lower Case ?
0464:AC 05 04    69            LDY SFTflag  ;  no, check case flag
0467:D0 1F       70            BNE Out1     ;  Already in Upper Case, then send
0469:8D 03 04    71            STA Temp     ;  Need Case change, save code
046C:A0 02       72            LDY #$02     ;  will send two characters
046E:8C 05 04    73            STY SFTflag  ;  Set shift flag, Non zero = Upper
0471:A9 10       74            LDA #$10     ;  Shift code
0473:D0 15       75            BNE Out2     ;  Toggle shift, then send character
0475:AC 05 04    76 LCchar:    LDY SFTflag  ;  Lower case character, Check case,
0478:F0 0E       77            BEQ Out1     ;  Already in Lower case, then send
047A:8D 03 04    78            STA Temp     ;  Need Case change, save code
047D:A9 00       79            LDA #$00     ;  Reset shift flag,
047F:8D 05 04    80            STA SFTflag  ;  Zero = Lower Case
0482:A0 02       81            LDY #$02     ;  Will send two characters
0484:A9 10       82            LDA #$10     ;  Shift code
0486:D0 02       83            BNE Out2     ;  Toggle case, then send character
0488:A0 01       84 Out1:      LDY #$01     ;  Will send one character
048A:A2 00       85 Out2:      LDX #$00     ;  Don't forget to reset
048C:8E 02 04    86            STX CRflag   ;  the carriage return flag
048F:A2 40       87            LDX #$40     ;  normal delay time
0491:D0 03       88            BNE CharOut  ;  Send the character
0493:AD 03 04    89 Char2:     LDA Temp     ;  get additional character(s)
0496:8D 00 17    90 CharOut:   STA PAD      ;  Put in output port
0499:A9 0A       91            LDA #$0A     ;  0A = 10 milliseconds
049B:8D 07 17    92            STA Timer    ;  set timer
049E:8D 07 17    93            STA Timer    ;  make sure
04A1:AD 07 17    94 Delay1:    LDA Timer    ;  check timer
04A4:10 F8       95            BPL Delay1   ;  still counting, check again
04A6:A9 00       96            LDA #$00     ;  Timer timed out
04A8:8D 00 17    97            STA PAD      ;  turn off port
04AB:8E 07 17    98            STX Timer    ;  start delay time
04AE:8E 07 17    99            STX Timer    ;  make sure

04B1:AD 07 17   100 Delay2:    LDA Timer    ;  check timer
04B4:10 FB      101            BPL Delay2   ;  still counting, check again
04B6:88         102            DEY          ;  Delay over, More Characters?
04B7:00 DA      103            BNE Char2    ;  Yes, then send them
04B9:68         104 Return:    PLA          ;  No, restore all registers
04BA:A8         105            TAY          ;
04BB:68         106            PLA          ;
04BC:AA         107            TAX          ;
04BD:68         108            PLA          ;
04BE:28         109            PLP          ;
04BF:4C A0 1E   110            JMP OUTCHR   ;  and send character to CRT
                111  ;
                112  ;  Control Code Table for IBM 8all 952
                113  ;   Bit    Function
                114  ;    0       Tilt 1
                115  ;    1       Tilt 2
                116  ;    2       Rotate 1
                117  ;    3       Rotate 2
                118  ;    4       Rotate 2A
                119  ;    5       Rotate -5
                120  ;    6       Check

  Return     2   1209 04B9
  SFTflag    2   1029 0405
  Start      2   1030 0406
  Temp       2   1027 0403
  Timer      2   5895 1707
  Type       2   1042 0412
  TypeOn     2   1024 0400
  Xspace     2   1087 043F

                121  ;    7       Upper Case
                122  ;
                123  ;                      ASCII  Printed
04C2:61         124 Codes: byte $61 ;    21    !
04C3:C7         125        byte $C7 ;    22    "
04C4:43         126        byte $43 ;    23    #
04C5:41         127        byte $41 ;    24    $
04C6:DC         128        byte $DC ;    25    %
04C7:5D         129        byte $5D ;    26    &
04C8:E7         130        byte $E7 ;    27    '
04C9:FF         131        byte $FF ;    28    (
04CA:EF         132        byte $EF ;    29    )
04CB:EB         133        byte $EB ;    2A    *
04CC:D8         134        byte $D8 ;    2B    +
04CD:42         135        byte $42 ;    2C    ,
04CE:5E         136        byte $5E ;    2D    -
04CF:40         137        byte $40 ;    2E    .
04D0:7E         138        byte $7E ;    2F    /
04D1:5F         139        byte $5F ;    30    0
04D2:7F         140        byte $7F ;    31    1
04D3:5B         141        byte $5B ;    32    2
04D4:78         142        byte $78 ;    33    3
04D5:4F         143        byte $4F ;    34    4
04D6:6F         144        byte $6F ;    35    5
04D7:4B         145        byte $4B ;    36    6
04D8:68         146        byte $68 ;    37    7
04D9:67         147        byte $67 ;    38    8
04DA:47         148        byte $47 ;    39    9
04DB:C2         149        byte $C2 ;    3A    :
04DC:C8         150        byte $C8 ;    3B    ;
04DD:FB         151        byte $FB ;    3C    <
04DE:C5         152        byte $C5 ;    3D    =
04DF:DD         153        byte $DD ;    3E    >
04E0:DE         154        byte $DE ;    3F    ?
04E1:5C         155        byte $5C ;    40   "at"
04E2:FC         156        byte $FC ;    41    A
04E3:D8         157        byte $D8 ;    42    B
04E4:F8         158        byte $F8 ;    43    C
```

```
04E5:CC          159        byte $CC ;   44    D
04E6:EC          160        byte $EC ;   45    E
04E7:C8          161        byte $C8 ;   46    F
04E8:E8          162        byte $E8 ;   47    G
04E9:E4          163        byte $E4 ;   48    H
04EA:C4          164        byte $C4 ;   49    I
04EB:FD          165        byte $FD ;   4A    J
04EC:D9          166        byte $D9 ;   4B    K
04ED:F9          167        byte $F9 ;   4C    L
04EE:CD          168        byte $CD ;   4D    M
04EF:ED          169        byte $ED ;   4E    N
04F0:C9          170        byte $C9 ;   4F    O
04F1:E9          171        byte $E9 ;   50    P
04F2:E5          172        byte $E5 ;   51    Q
04F3:C5          173        byte $C5 ;   52    R
04F4:DA          174        byte $DA ;   53    S
04F5:FA          175        byte $FA ;   54    T
04F6:CE          176        byte $CE ;   55    U
04F7:EE          177        byte $EE ;   56    V
04F8:CA          178        byte $CA ;   57    W
04F9:EA          179        byte $EA ;   58    X
04FA:E6          180        byte $E6 ;   59    Y
04FB:C6          181        byte $C6 ;   5A    Z
04FC:FF          182        byte $FF ;   5B    (
04FD:DF          183        byte $DF ;   5C    |
04FE:EF          184        byte $EF ;   5D    )
04FF:CF          185        byte $CF ;   5E    ^
0500:FE          186        byte $FE ;   5F    _
0501:E7          187        byte $E7 ;   60    `
0502:7C          188        byte $7C ;   61    a
0503:58          189        byte $58 ;   62    b
0504:78          190        byte $78 ;   63    c
0505:4C          191        byte $4C ;   64    d
0506:6C          192        byte $6C ;   65    e
0507:48          193        byte $48 ;   66    f
0508:68          194        byte $68 ;   67    g
0509:64          195        byte $64 ;   68    h
050A:44          196        byte $44 ;   69    i
050B:7D          197        byte $7D ;   6A    j
050C:59          198        byte $59 ;   6B    k
050D:79          199        byte $79 ;   6C    l
050E:4D          200        byte $4D ;   6D    m
050F:6D          201        byte $6D ;   6E    n
0510:49          202        byte $49 ;   6F    o
0511:69          203        byte $69 ;   70    p
0512:65          204        byte $65 ;   71    q
0513:45          205        byte $45 ;   72    r
0514:5A          206        byte $5A ;   73    s
0515:7A          207        byte $7A ;   74    t
0516:4E          208        byte $4E ;   75    u
0517:6E          209        byte $6E ;   76    v
0518:4A          210        byte $4A ;   77    w
0519:6A          211        byte $6A ;   78    x
051A:66          212        byte $66 ;   79    y
051B:46          213        byte $46 ;   7A    z
051C:FF          214        byte $FF ;   7B    {
051D:C0          215        byte $C0 ;   7C    cent sign
051E:EF          216        byte $EF ;   7D    }
051F:80          217        byte $80 ;   7E    del
0520:80          218        byte $80 ;   7F    del

reference name table

                                  value
name      size      dec   hex
ASCII        2     1025  0401
CKLF         2     1091  0443
CRflag       2     1026  0402
Char2        2     1171  0493
CharOut      2     1174  0496
Codes        2     1218  04C2
Delay1       2     1185  04A1
Delay2       2     1201  04B1
GetCode      2     1115  045B
LCchar       2     1141  0475
Linefeed     2     1107  0453
OUTCHR       2     7840  1EA0
Out1         2     1160  0488
Out2         2     1162  048A
PAD          2     5888  1700
PADD         2     5889  1701
```

Listing 2 . SYM interface program.

```
                        ;      SYM SELECTRIC DRIVER
                        ;
                        ; Routine to convert ASCII in accumulator
                        ; to Selectric codes, and send to Selectric with
                        ; proper timing, case, etc. if location 1FFF hex
                        ; is not equal to zero. Returns to SYM routine TOUT
                        ; with all registers unchanged from entry.  Uses no
                        ; page zero locations.
                        ; Change OUTVEC to 1EE0 to use. ( SD 1EE0, A664 )
                        ; In response to "Memory Size ?" from BASIC, enter 7903
                        ; This routine uses the top 288 bytes of an 8K SYM memory.
                        ;
                PAD:    equ $A800   ;  Output port to Selectric
                PADD:   equ $A802   ;  Data direction register
                Timer:  equ $A417   ;  Divide by 1024
                TOUT:   equ $8AA0   ;  SYM routine, sends ASCII to CRT
                TypeOn: equ $1FFF   ;  Zero = Don't type
                ASCII:  equ $1FFE   ;  Temporary
                CRflag: equ $1FFD   ;    Storage
                Temp:   equ $1FFC   ;      buffer
                SFTflag: equ $1FFB  ;        locations
                        org $1EE0   ;
1EE0:08         Start:  PHP         ;  Start of program
1EE1:48                 PHA         ;  Save flags & accumulator
1EE2:AD FF 1F           LDA TypeOn  ;  Check if Typing desired
1EE5:D0 05              BNE Type    ;  Yes, then Type
1EE7:68                 PLA         ;  No, restore accumulator
1EE8:28                 PLP         ;  and flags
1EE9:4C A0 8A           JMP TOUT    ;  and send to CRT
1EEC:A9 FF      Type:   LDA #$FF    ;  Set port to output
1EEE:8D 02 A8           STA PADD    ;
1EF1:68                 PLA         ;  Get ASCII code
1EF2:8D FE 1F           STA ASCII   ;  Save it
1EF5:48                 PHA         ;  Save registers
1EF6:8A                 TXA         ;
1EF7:48                 PHA         ;
1EF8:98                 TYA         ;
1EF9:48                 PHA         ;
1EFA:D8                 CLD         ;  Binary mode
1EFB:AD FE 1F           LDA ASCII   ;  Get ASCII code
1EFE:C9 20              CMP #$20    ;  Is it a space?
1F00:F0 17              BEQ Xspace  ;  Yes, then send a space
1F02:10 31              BPL GetCode ;  > space is valid code
1F04:C9 0D              CMP #$0D    ;  < space, is it Return?
1F06:D0 15              BNE CKLF    ;  no, go check if it is a line feed
1F08:A9 04              LDA #$04    ;  yes, carriage return
1F0A:8D FD 1F           STA CRflag  ;  Set flag to non-zero
1F0D:A8                 TAY         ;  Will send four characters
1F0E:A9 00              LDA #$00    ;  but next three characters will
```

Selecric I/O could be interfaced using the information in this article and in those manuals.

The 2970 Selectric was originally used in a motel/hotel reservation system, according to its type ball, which had funny little symbols that would only be useful in printing a hotel bill, uppercase letters only, the numbers and only a few standard punctuation marks. I obtained a number 952 ball to replace it.

The 952 ball has both upper and lowercase letters, numbers and most of the ASCII symbols. I did not convert the interposers of the machine to match the new ball, so it was useless as a typewriter. But I'm only using it as a printer, so why waste the time and money on the interposers?

Another peculiarity (?) of the 2970 Selectric is that a space is considered a printable character and is obtained by energizing the non-print magnet along with one or more of the tilt, rotate or check magnets. I understand that a space is a control character on some of the other Selectric I/O machines. They would require slight modification of the hardware and software interfaces contained in

```
1F10:8D FC 1F    50              STA Temp    ;  be nulls to allow for return
1F13:A9 40       51              LDA #$40    ;  Carriage return code
1F15:A2 7F       52              LDX #$7F    ;  Long delay
1F17:D0 57       53              BNE CharOut ;Send the carriage return
1F19:A9 81       54 Xspace:      LDA #$81    ;  Space code
1F1B:D0 45       55              BNE Out1    ;  Send it
1F1D:C9 0A       56 CKLF:        CMP #$0A    ;  Is code a line feed?
1F1F:D0 72       57              BNE Return  ;  No, ignore it
1F21:AD FD 1F    58              LDA CRflag  ;  Check flag
1F24:F0 07       59              BEQ Linefeed ;Send LF if last not Return
1F26:A9 00       60              LDA #$00    ;  Last character sent was Return
1F28:8D FD 1F    61              STA CRflag  ;  reset the flag
1F2B:F0 66       62              BEQ Return  ;  and ignore the line feed
1F2D:A0 01       63 Linefeed:    LDY #$01    ;  one character
1F2F:A2 7F       64              LDX #$7F    ;  long delay
1F31:A9 20       65              LDA #$20    ;  line feed code
1F33:D0 3B       66              BNE CharOut ;  send the line feed
1F35:29 7F       67 GetCode:     AND #$7F    ;  mask off high bit
1F37:AA          68              TAX         ;  put in index X
1F38:BD 7B 1F    69              LDA Codes-$21,X ; Get code in Accumulator
1F3B:0A          70              ASLA        ;  shift left, put case in carry
1F3C:90 11       71              BCC LCchar  ;  Lower Case ?
1F3E:AC FB 1F    72              LDY SFTflag ;  no, check case flag
1F41:D0 1F       73              BNE Out1    ;  In Upper Case, send character
1F43:8D FC 1F    74              STA Temp    ;  Need Case change, save code
1F46:A0 02       75              LDY #$02    ;  will send two characters
1F48:8C FB 1F    76              STY SFTflag ;  Set shift flag. Non zero = Upper
1F4B:A9 10       77              LDA #$10    ;  Shift code
1F4D:D0 15       78              BNE Out2    ;  Toggle shift, then send character
1F4F:AC FB 1F    79 LCchar:      LDY SFTflag ;  Lower case character. Check case.
1F52:F0 0E       80              BEQ Out1    ;  In lower case, send character
1F54:8D FC 1F    81              STA Temp    ;  Need Case change, save code
1F57:A9 00       82              LDA #$00    ;  Reset shift flag.
1F59:8D FB 1F    83              STA SFTflag ;  Zero = Lower Case
1F5C:A0 02       84              LDY #$02    ;  Will send two characters
1F5E:A9 10       85              LDA #$10    ;  Shift code
1F60:D0 02       86              BNE Out2    ;  Toggle case, then send character
1F62:A0 01       87 Out1:        LDY #$01    ;  Will send one character
1F64:A2 00       88 Out2:        LDX #$00    ;  Don't forget to reset
1F66:8E FD 1F    89              STX CRflag  ;  the carriage return flag
1F69:A2 40       90              LDX #$40    ;  normal delay time
1F6B:D0 03       91              BNE CharOut ;  Send the character
1F6D:AD FC 1F    92 Char2:       LDA Temp    ;  get additional character
1F70:8D 00 A8    93 CharOut:     STA PAD     ;  Put in output port
1F73:A9 0A       94              LDA #$0A    ;  0A = 10 milliseconds
1F75:8D 17 A4    95              STA Timer   ;  set timer
1F78:8D 17 A4    96              STA Timer   ;  make sure
1F7B:AD 17 A4    97 Delay1:      LDA Timer   ;  check timer
1F7E:10 FB       98              BPL Delay1  ;  still counting, check again
1F80:A9 00       99              LDA #$00    ;  Timer timed out
1F82:8D 00 A8   100              STA PAD     ;  turn off port
1F85:8E 17 A4   101              STX Timer   ;  start delay time
1F88:8E 17 A4   102              STX Timer   ;  make sure
1F8B:AD 17 A4   103 Delay2:      LDA Timer   ;  check timer
1F8E:10 FB      104              BPL Delay2  ;  still counting, check again
1F90:88         105              DEY         ;  Delay over, Any more Characters?
1F91:D0 DA      106              BNE Char2   ;  Yes, then send them
1F93:68         107 Return:      PLA         ;  No, restore all registers
1F94:A8         108              TAY         ;
1F95:68         109              PLA         ;
1F96:AA         110              TAX         ;
1F97:68         111              PLA         ;
1F98:28         112              PLP         ;
1F99:4C A0 BA   113              JMP TOUT    ;  and send character to CRT
                114  ;
                115  ;  Control Code Table for IBM Ball 952
                116  ;   Bit    1=function
                117  ;    0       Tilt 1
                118  ;    1       Tilt 2
                119  ;    2       Rotate 1
                120  ;    3       Rotate 2
                121  ;    4       Rotate 2A
                122  ;    5       Rotate -5
                123  ;    6       Check
                124  ;    7       Upper Case
                125  ;
                126  ;                    ASCII  Printed
1F9C:C1         127 Codes:  byte $C1 ;    21     !
1F9D:C7         128         byte $C7 ;    22     "
1F9E:43         129         byte $43 ;    23     #
1F9F:41         130         byte $41 ;    24     $
1FA0:DC         131         byte $DC ;    25     %
1FA1:5D         132         byte $5D ;    26     &
1FA2:E7         133         byte $E7 ;    27     '
1FA3:FF         134         byte $FF ;    28     (
1FA4:EF         135         byte $EF ;    29     )
1FA5:EB         136         byte $EB ;    2A     *
1FA6:DB         137         byte $DB ;    2B     +
1FA7:42         138         byte $42 ;    2C     ,
1FA8:5E         139         byte $5E ;    2D     -
1FA9:40         140         byte $40 ;    2E     .
1FAA:7E         141         byte $7E ;    2F     /
1FAB:5F         142         byte $5F ;    30     0
1FAC:7F         143         byte $7F ;    31     1
1FAD:5B         144         byte $5B ;    32     2
1FAE:7B         145         byte $7B ;    33     3
1FAF:4F         146         byte $4F ;    34     4
1FB0:6F         147         byte $6F ;    35     5
1FB1:4B         148         byte $4B ;    36     6
1FB2:6B         149         byte $6B ;    37     7
```

this article, but the changes should not be difficult to accomplish.

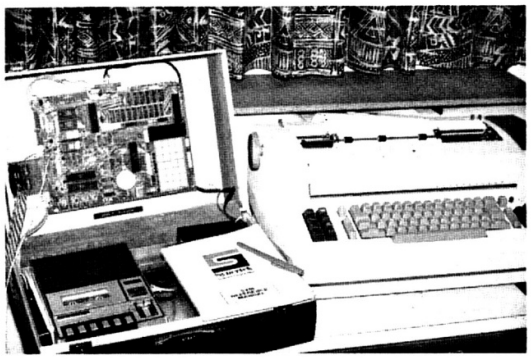
The control characters consist of Shift, Carriage Return and Index. Each time the Shift line is brought low, the machine changes case. It will stay in that case until the Shift line is pulsed again. This means that the KIM must keep track of what case the machine is in, only changing case when necessary. The SHIFT SYNC switch, a push-button type mounted on the Selectric, is used to initially synchronize it and the KIM as to which case is selected.

The Carriage Return function on the Selectric both returns the carriage to the left margin and advances the paper to the next print line. Therefore, the KIM must suppress any line-feed characters sent immediately after a Carriage Return, or double-spaced printing will result. Index is just IBMese for line feed and requires no further explanation.

The Check line is used in this interface scheme to differentiate between the printable and control characters. If CK in Fig. 1 is low, then $\overline{CK}$ will be high. Therefore, line D6, for example, activates U5B (ROT 5) and not U5C (Return). If CK were high, the reverse would be true. This multiplexing of the data lines allows the eight-bit port to control the eleven different signals needed by the Selectric.

Fig. 1 is the schematic of the complete hardware interface I am using. U1 and U2 are mounted on a small board in the KIM cabinet. A six-foot twisted-pair cable connects from this board to another small board in the 2970 cabinet. The numbers on the lines inside the dotted oval correspond to the pin numbers of the DB-25 connectors I used to terminate the cable.

The second board contains U3, U4, U5 and eleven identical relay driver circuits as shown. The collectors of the MJE-340s are wired to the selector magnets using the original 2970 cables. The other sides of the selector magnets are connected by the original wiring to the +48 volt output from the power supply in the 2970 cabinet.



The Selectric and the SYM-1.

Software

Most of the work of the interfacing has been left up to the KIM's software. It must provide for code conversion from ASCII to the codes that correspond to the appropriate tilt, rotate and check magnet combination for the type ball or select the required control character. It must keep track of the machine's case – upper or lower – suppress line feeds immediately preceded by carriage returns, etc. Also, it must provide the pulses with the proper timing.

All pulses should have a 10 millisecond duration with at least 67.5 milliseconds wait from the start of one pulse to the occurence of another, except for Index and Carriage Return, which require even longer delays. Listing 1 accomplishes all of these functions in the KIM.

This program listing should be reasonably easy to follow. If not, I wasted a lot of time on comments. It is located in my block of memory starting at 0400 hex. To prevent possible interference with other programs, it uses no page zero locations. If you relocate the program, the scratch locations at $0400-$0405 and the references to them must be changed. The rest of the code should be completely relocatable without change.

To output a character, most application software packages use a JSR OUTCHR (20 A0 1E). This is located at $2A51 in my version of BASIC, for example. Change it to JSR Start (20 06 04) to use the Selectric driver program. The program ends with a JMP to OUTCHR, which will then print the character on the terminal and return to the application program.

If location 0400 hex is a zero, the character will not be printed on the Selectric. Changing $0400 to any other value will cause the character to be printed. This can be done from BASIC using POKE 1024,9 to start printing and POKE 1024,0 to stop it.

The SYM-1

Shortly after I got the Selectric working with the KIM, I obtained one of Synertek's new SYM-1 boards. Although it uses essentially the same expansion bus as the KIM and has many other similarities, none of my KIM programs that use the onboard keypad and display would run on the SYM without major changes due to the way the SYM addresses its peripherals. Tom Pittman's Tiny BASIC, however, was easier to interface to the SYM than to the KIM! The SYM's expanded monitor (version 1.1), BASIC on ROM, higher-speed cassette interface and up to 4K of RAM on-board are excellent features.

Listing 2 is the revised Selectric driver software for the SYM. The small board plugged onto the SYM's auxiliary applications connector to the left side of it in the accompanying photo contains U1 and U2 of the hardware interface. The cable connected to it runs to the Selectric. It connects to the SYM +5 V, ground and port B of 6522 U28.

```
1FB3:67          150          byte $67 ;   38    8
1FB4:47          151          byte $47 ;   39    9
1FB5:C2          152          byte $C2 ;   3A    :
1FB6:CB          153          byte $CB ;   3B    ;
1FB7:FB          154          byte $FB ;   3C    <
1FB8:C3          155          byte $C3 ;   3D    =
1FB9:DD          156          byte $DD ;   3E    >
1FBA:DE          157          byte $DE ;   3F    ?
1FBB:5C          158          byte $5C ;   40    "at"
1FBC:FC          159          byte $FC ;   41    A
1FBD:D8          160          byte $D8 ;   42    B
1FBE:F8          161          byte $F8 ;   43    C
1FBF:CC          162          byte $CC ;   44    D
1FC0:EC          163          byte $EC ;   45    E
1FC1:C8          164          byte $C8 ;   46    F
1FC2:E8          165          byte $E8 ;   47    G
1FC3:E4          166          byte $E4 ;   48    H
1FC4:C4          167          byte $C4 ;   49    I
1FC5:FD          168          byte $FD ;   4A    J
1FC6:D9          169          byte $D9 ;   4B    K
1FC7:F9          170          byte $F9 ;   4C    L
1FC8:CD          171          byte $CD ;   4D    M
1FC9:ED          172          byte $ED ;   4E    N
1FCA:C9          173          byte $C9 ;   4F    O
1FCB:E9          174          byte $E9 ;   50    P
1FCC:E5          175          byte $E5 ;   51    Q
1FCD:C5          176          byte $C5 ;   52    R
1FCE:DA          177          byte $DA ;   53    S
1FCF:FA          178          byte $FA ;   54    T
1FD0:CE          179          byte $CE ;   55    U
1FD1:EF          180          byte $EF ;   56    V
1FD2:CA          181          byte $CA ;   57    W
1FD3:EA          182          byte $EA ;   58    X
1FD4:E6          183          byte $E6 ;   59    Y
1FD5:C6          184          byte $C6 ;   5A    Z
1FD6:FF          185          byte $FF ;   5B    (
1FD7:DF          186          byte $DF ;   5C    |
1FD8:EF          187          byte $EF ;   5D    )
1FD9:CF          188          byte $CF ;   5E    ^
1FDA:FE          189          byte $FE ;   5F    _
1FDB:E7          190          byte $E7 ;   60    `
1FDC:7C          191          byte $7C ;   61    a
1FDD:58          192          byte $58 ;   62    b
1FDE:78          193          byte $78 ;   63    c
1FDF:4C          194          byte $4C ;   64    d
1FE0:6C          195          byte $6C ;   65    e
1FE1:48          196          byte $48 ;   66    f
1FE2:68          197          byte $68 ;   67    g
1FE3:64          198          byte $64 ;   68    h
1FE4:44          199          byte $44 ;   69    i
1FE5:70          200          byte $70 ;   6A    j
1FE6:59          201          byte $59 ;   6B    k
1FE7:79          202          byte $79 ;   6C    l
1FE8:4D          203          byte $4D ;   6D    m
1FE9:6D          204          byte $6D ;   6E    n
1FEA:49          205          byte $49 ;   6F    o
1FEB:69          206          byte $69 ;   70    o
1FEC:65          207          byte $65 ;   71    q
1FED:45          208          byte $45 ;   72    r
1FEE:5A          209          byte $5A ;   73    s
1FEF:7A          210          byte $7A ;   74    t
1FF0:4E          211          byte $4E ;   75    u
1FF1:6E          212          byte $6E ;   76    v
1FF2:4A          213          byte $4A ;   77    w
1FF3:6A          214          byte $6A ;   78    x
1FF4:66          215          byte $66 ;   79    y
```

```
reference name table

                    value
name      size      dec   hex
ASCII        2     8190  1FFE
CKLF         2     7965  1F1D
CRflag       2     8189  1FFD
Char2        2     8045  1F6D
CharOut      2     8048  1F70
Codes        2     8092  1F9C
Delay1       2     8059  1F7B
Delay2       2     8075  1F8B
GetCode      2     7989  1F35
LCchar       2     8015  1F4F
Linefeed     2     7981  1F2D
Out1         2     8034  1F62
Out2         2     8036  1F64
PAD          2    43008  A800
PADD         2    43010  A802
Return       2     8083  1F93
SFTflag      2     8187  1FFB
Start        2     7904  1EE0
TOUT         2    35488  8AA0
Temp         2     8188  1FFC
Timer        2    42007  A417
Type         2     7916  1EEC
TypeOn       2     8191  1FFF
Xspace       2     7961  1F19
```

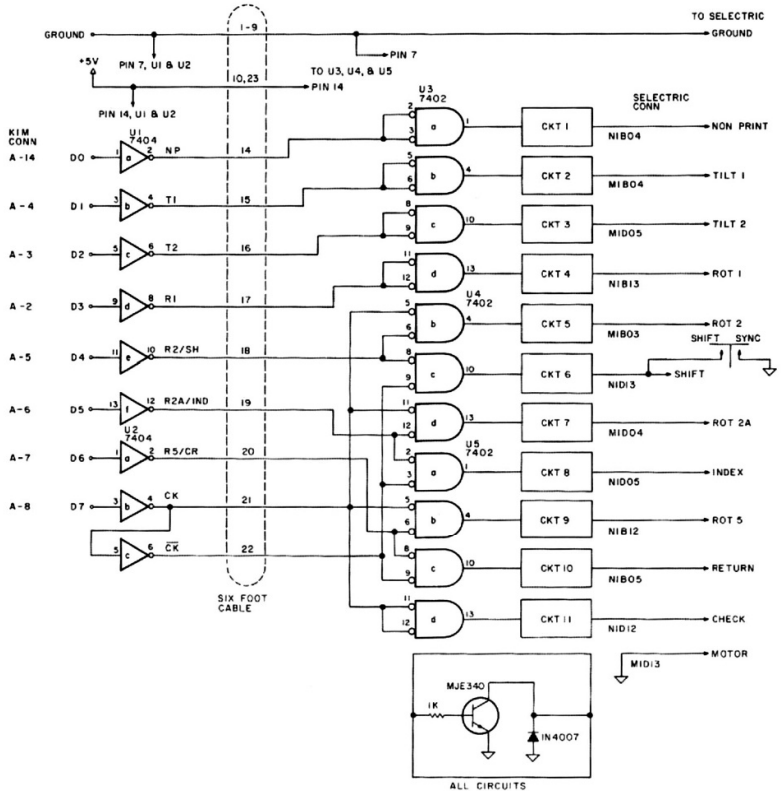


Fig. 1. The complete hardware interface schematic.

Those familiar with the SYM may have already noticed that the program is located from $1EE4 to $1FFF, which would only be available if the SYM had a full 8K of RAM. Or you may have noticed that the Synertek logo and name have been covered up with another added board that has DIP plugs connected to the sockets for the SYM's U12 and U19. If so, you are right!

This little board does expand the SYM to a full 8K of RAM. It is double-sided and requires plated through holes. Hidden jumpers run to pins of SYM's U1 to complete the required connections. The original SYM U12 and U19 plus eight more 2114s mount on this expansion board. The full 8K allows fairly extensive BASIC, Tiny BASIC or assembly-language programs to be used before the SYM runs out of memory space.

To expand memory beyond 8K would definitely require buffering such as used on the memory board in the abovementioned *Microcomputing* article. This little board does not have room for the buffer ICs dictated by good engineering practices. However, many of them are in use on SYMs other than mine without any reported problems.

Since the SYM vectors all of its inputs and outputs through a common memory location in its system RAM, interfacing BASIC or other application programs to the Selectric driver software is easy—just change the vector! Instructions for doing this are contained in the comments of Listing 2.

Conclusion

It is not difficult to interface an IBM Selectric I/O typewriter to a microcomputer, such as the KIM-1 and SYM-1, that has an eight-bit output port and an onboard timer. Excellent, although somewhat slow, print quality is obtainable with minimal hardware and software required. The KIM or SYM can now be used for those applications such as text editing that require hard copy. (The KIM was used to prepare this article.) It helps to have a hard-copy listing of those programs you're trying to debug, too! ■

Notes

Blank PC boards and instructions to expand the SYM to 8K of RAM are available from the author for $5 each plus a self-addressed stamped envelope. KIM-format cassette tapes of the Selectric interface software (specify version) will be provided for $2.50.