Photo 1: The Teleterminal Corporation Fly Reader for use with the KIM-1 microprocessor.

Come Fly With KIM

Rick Simpson
MOS Technology Inc
Valley Forge Corporate Center
950 Rittenhouse Rd
Norristown PA 19401

Many computer hobbyists start with nothing more than a processor, a small amount of programmable memory, a small onboard monitor such as MIKBUG or KIM and some front panel switches. Those with more foresight, or cash, will have a keypad or even a full keyboard for data entry and processor control. But even with a good monitor and a full keyboard and display, loading programs is a tedious chore at best, and there is an awful feeling when you turn off power, knowing that twenty minutes of typing just evaporated.

The next step in expanding the system is usually an audio cassette interface or a Teletype with paper tape reader and punch for the wealthy or fortunate. Now the tedious retyping is eliminated and a program, once written and recorded or punched, can be reloaded in a matter of minutes.

Many people stop at this point. When hand assembly of programs is required, a program of more than a few hundred bytes is rarely attempted. But as the software gap is slowly filled, more and more systems are being implemented with assemblers or BASIC interpreters. More memory is purchased to expand programmable memory from a few hundred bytes to 4 K, 8 K, or more. *[One firm now even markets a 64 K board! . . .* CH*]* Once again your memory has outrun your ability to fill it in a reasonable time.

For instance, using the Teletype paper tape reader or audio cassette interface on the KIM system, a 2 K Tiny BASIC interpreter takes almost ten minutes to load. A 12 K BASIC source program would take an hour. Even a 30 character per second interface only cuts this to twenty minutes. The alternatives seem to be a Tarbell or Suding type high speed cassette system, a 3M drive at 9600 bps or a floppy disk.

The floppy disk certainly solves the speed problem. We can now load 12 K in a few seconds, but at a cost of $1,000 to $2,000. The high speed cassette is reasonable in cost, about $200 including the high quality cassette unit required, but tricky to interface unless a manufacturer-supplied board or kit is available.

Although some magnetic tape units have start, stop, and search functions under program control, most users end up pushing the buttons. No hobbyist magnetic tape cassette unit can read a few (ie: one line or so) characters, stop and process the data, and then start and read some more, a real need when running an assembler with the source stored on the tape in a limited resource system.

After this lengthy preamble, you may have suspected that I have an alternative solution in mind, and I do: a high speed paper tape reader, manufactured by the Teleterminal Corporation, called the Fly Reader, shown in photo 1. Although a bit more expensive than the high speed cassette system (about $350), it is far faster; reading at 300 characters per second, it can load my Tiny BASIC in twenty seconds, or fill that 12 K of memory in two minutes. It is easy to interface, requires little software, and is extremely reliable. It needs only a single +5 V, 2 A power supply and is operated completely under program control. You can read as little as a single character at a time and can read in either direction; try that on your cassette!

Paper tape has always been the standard mass storage device for minicomputers, until floppy disks came along, and paper tape has been the most universal and inexpensive method of software distribution and interchange in the minicomputer field.

The basic problem is that it is only a reader; how do you punch the tape? There are several answers: Flexowriters and other similar low speed punches are becoming available, as are gobs of older 7 level machines. I've also seen higher speed punches, typically 60 characters per second, advertised for under $100. The fact that the punch is slow is not so important; typically you punch a tape once and read it many times. Even if you have no punch, the reader is a useful peripheral because much software is available already punched.

How it Works

The Fly Reader can read at such a high speed because it transfers 8 bits in parallel and contains only a single moving part: a stepping motor connected to a toothed wheel which engages the sprocket holes in the paper tape. Sensing of the holes in the tape is done by photodetectors rather than the mechanical fingers used in a low speed reader. This is a method similar to that used in the manual reader sold by Oliver Audio. Figure 1 shows a block diagram of the unit.

There are five control lines for the unit. All are compatible with standard TTL circuitry. The "load status" line is a logic 1, +5 V, if the reader is not ready because the feed gate is not closed. When tape is inserted and the gate is closed, this signal goes to logic zero.

In operation, the reader must be checked by software to see if the "reader ready" signal is at logic 1 to indicate that the reader is ready to read another tape character. The software must then issue a pulse from logic 1 to logic 0 whose width is between 500 ns and 500 μs. This READ pulse will start the reader and drive the reader ready line to logic 0. The software then watches the "data strobe" line. When data strobe goes to logic 1, the data can be read from the eight parallel output lines. If the program needs to read another character from tape, it must wait until reader ready goes back to logic 1, issue another read pulse, and wait for another data strobe. Figure 2 shows the flowchart for such software and figure 3 shows the interface timing diagrams.

Figure 2: Flowchart of the software for reading the paper tape with the Fly Reader.

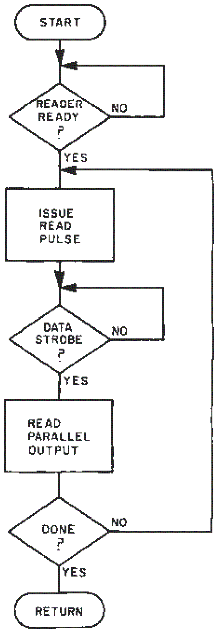
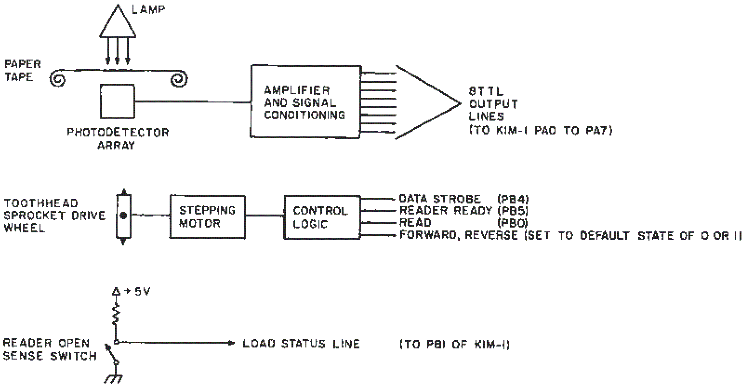


Figure 1: Block diagram of the Fly Reader. The input is achieved through an incandescent lamp and a photodetector array. The tape is advanced by a stepping motor allowing input of data either forwards or backwards. The reader open sense switch is closed when the paper tape is in the reader.


LAMP
PAPER TAPE
PHOTODETECTOR ARRAY
AMPLIFIER AND SIGNAL CONDITIONING
8 TTL OUTPUT LINES (TO KIM-1 PA0 TO PA7)
TOOTHHEAD SPROCKET DRIVE WHEEL
STEPPING MOTOR
CONTROL LOGIC
DATA STROBE (PB4)
READER READY (PB5)
READ (PB0)
FORWARD, REVERSE (SET TO DEFAULT STATE OF 0 OR 1)
+5V
READER OPEN SENSE SWITCH
LOAD STATUS LINE (TO PB1 OF KIM-1)

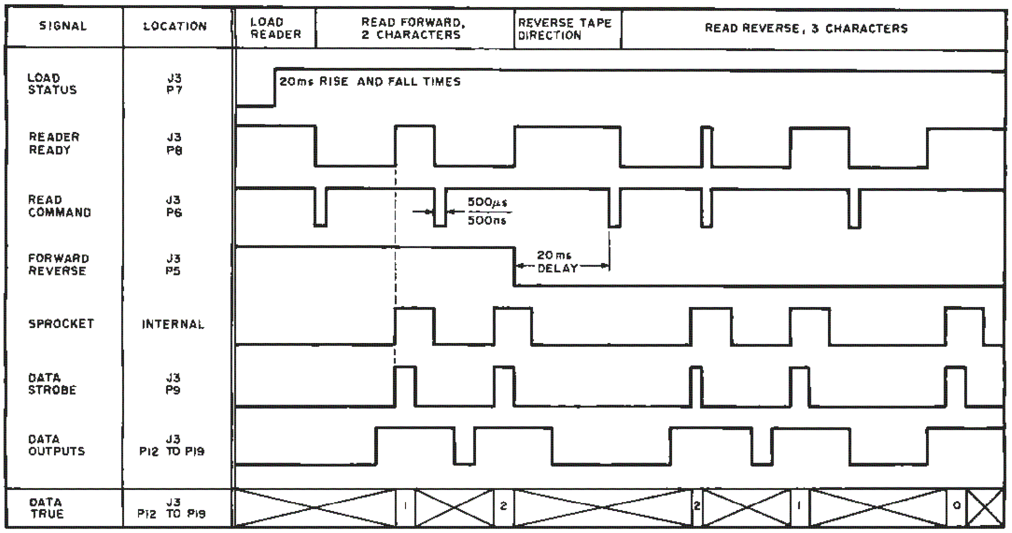


Figure 3: Timing diagram generated by the software of listing 1. The minimum width of the data strobe is 50 μs except when forced low by a new read command. The crossed out sections in the data true section indicate that the state of the output is unknown. A read command is issued only when the reader ready line is high.

Other Interface Possibilities (An Advertisement . . .)

The interface described here uses most of the available IO lines on the KIM-1. Systems supporting several IO devices may wish to interface the Fly Reader through a separate interface chip. The MCS6532 is one such interface chip for adding more IO to KIM-1 as well as an additional 128 bytes of programmable memory and another interval timer. Two of these chips (with ROMs) are already built into each KIM. Since the 6532 is a MOS rather than TTL device it does not load the KIM-1 address or data buses significantly. The MCS6532 is available for $16 postpaid from National Electro-Sales, 12063 W Jefferson, Culver City CA 90230.

Interfacing to Kim

As described above, the Fly Reader interface requires eight parallel input lines, three input control lines and one or two output control lines. Two output control lines are needed if the forward, reverse function is used; otherwise only one control line is needed. Since KIM-1 has 15 bidirectional IO lines the interface is very simple. The A data port lines PA0 to PA7 are programmed as input lines and connected to the parallel output lines from the Fly Reader. PB5 is connected to the reader ready line, PB1 is connected to the load status line, PB0 is connected to the read command line and programmed as an output line, and PB4 is connected to the data strobe line. A 5 V, 2 A power supply is connected to the reader and the interface is complete. When wiring the power connectors, you should make sure that separate power and ground wires are run back to the power supply for both the motor and logic connections. This will insure that current surges to the motor during stepping operations do not feed noise pulses into the control logic.

Interface Details

The fifteen KIM-1 IO lines are divided into two ports. Each port has a data direction register and a data register. Writing a 1 to a bit or bits in the data direction register configures the corresponding IO lines for output, writing a 0 sets them for input. For instance, writing a hexadecimal 02 to the A data direction register configures the PA0 line for input, PA1 for output, and PA2 through PA7 as input lines. Similarly, writing hexadecimal F0 to the B data direction register configures PB0 through PB3 as input lines and PB4, PB5 and PB7 as output lines. Note that there is no PB6, and PB7 has no output pullup; it is essentially an open collector output. Reading the A or B data register will show whether the signal at each input line is 1 or 0 and will show whether a 0 or 1 was previously written to any lines configured as outputs. Writing to a data register will set the appropriate output lines to 1 or 0 and does not affect lines programmed as inputs. Hexadecimal address location 1700 is the A port data direction register, hexadecimal 1702 is the B data direction register, hexadecimal 1701 is the A data register, and hexadecimal 1703 is the B data register.

The KIM Paper Tape Format

The software to drive the reader uses the same paper tape format as that used in the KIM-1 Q (paper tape dump) and L (paper tape load) commands. Thus any paper tape punched on a low speed punch by KIM-1 can be read by the Fly Reader. The KIM-1 paper tape format ignores any characters

```
        hexadecimal
address    code      label    op.   operand   commentary

1C4F                 START    EQU             $1C4F
4000    D8                    CLD             clear decimal mode;
4001    20  57  40   PTRLD    JSR   PTRINI    go to PTRINI;
4004    20  6F  40   LOAD     JSR   GETPTR    go to GETPTR;
4007    C9  3B                CMP   $3B     } if A not equal to 3B go to LOAD;
4009    D0  F9                BNE   $LOAD   }
400B    A9  00       LOADS    LDA   00        else A:=00;
400D    85  F7                STA   $F7     } store checksum;
400F    85  F6                STA   $F6     }
4011    20  8B  40            JSR   PTRBYT    go to PTRBYT; [get byte count]
4014    AA                    TAX             X:=A;
4015    20  91  1F            JSR   $1F91     compute checksum;
4018    20  8B  40            JSR   PTRBYT    get high address;
401B    85  FB                STA   FB        store high address pointer;
401D    20  91  1F            JSR   $1F91     compute checksum;
4020    20  8B  40            JSR   PTRBYT    get low address pointer;
4023    85  FA                STA   FA        store low address pointer;
4025    20  91  1F            JSR   $1F91     compute checksum;
4028    8A                    TXA             A:=X;
4029    F0  0F                BEQ   LOAD3     if A:=0 go to LOAD3;
402B    20  8B  40   LOAD2    JSR   PTRBYT    get data;
402E    91  FA                STA   FA,Y      store data;
4030    20  91  1F            JSR   $1F91     compute checksum;
4033    20  63  1F            JSR   $1F63     get next address;
4036    CA                    DEX             X:=X−1;
4037    D0  F2                BNE   LOAD2     go to LOAD2;
4039    E8                    INX             X:=X+1;
403A    20  8B  40   LOAD3    JSR   PTRBYT    get data;
403D    C5  F6                CMP   $F6       compare high order checksuin;
403F    D0  12                BNE   LOADER    if different go to LOADER;
4041    20  8B  40            JSR   PTRBYT    else get data;
4044    C5  F7                CMP   $F7       compare checksum;
4046    DO  0B                BNE   LOADER    if different go to LOADER;
4048    8A                    TXA             else A:=X;
4049    D0  B9                BNE   LOAD      if A not equal to 0 go to LOAD;
404B    A2  0C       LOAD7    LDX   0C        else X:= location of 'KIM';
404D    20  31  1E   LOAD8    JSR   $1E31     output message;
4050    4C  4F  1C            JMP   START     go to START;
4053    A2  11       LOADER   LDX   11        X:=location of 'ERR KIM';
4055    D0  F6                BNE   LOAD8     go to LOAD8;
4057    A9  01       PTRINI   LDA   $01       [initialization routine]
4059    8D  03  17            STA   $1703     A:=B port address;
405C    8D  02  17            STA   $1702     read flag:=1;
405F    AD  02  17            LDA   $1702     A:=B register;
4062    29  02                AND   $02       determine PB1;
4064    D0  08                BNE   OK        if reader ready go to OK;
4066    A9  58                LDA   'X'       else A:= 'X';
4068    20  A0  1E            JSR   $1EA0     output 'X';
406B    4C  57  40            JMP   PTRINI    go to PTRINI;
406E    60           OK       RTS             return;
406F    AD  02  17   GETPTR   LDA   1702      [subroutine to input one character]
4072    29  20                AND   $20       get bit from B data register;
4074    F0  F9                BEQ   GETPTR    if not ready go to GETPTR;
4076    A9  00                LDA   $00     } else output read pulse;
4078    8D  02  17            STA   $1702   }
407B    A9  01                LDA   $01     } turn off read pulse;
407D    8D  02  17            STA   $1702   }
4080    AD  02  17   CHECK    LDA   $1702   } get bit 5 from B data register;
4083    29  10                AND   $10     }
4085    F0  F9                BEQ   CHECK     if character not ready go to CHECK;
4087    AD  00  17            LDA   $1700     else get character;
408A    60                    RTS             return;
408B    20  6F  40   PTRBYT   JSR   GETPTR    get character;
408E    20  AC  1F            JSR   $1FAC     pack character;
4091    20  6F  40            JSR   GETPTR    get another character;
4094    20  AC  1F            JSR   $1FAC     pack character;
4097    A5  F8                LDA   $F8       A:=2 characters;
4099    60                    RTS             return;
409A    20  57  40   MAIN     JSR   PTRINI    go to PTRINI;
409D    20  6F  40   LOOP     JSR   GETPTR    go to GETPTR;
40A0    4C  9D  40            JMP   LOOP      go to LOOP;
                              END
```

Listing 1: The basic software needed to run the Fly Reader with the KIM-1 microprocessor. The software uses the KIM-1 monitor routines and substitutes the GETCH and GETBYT routines with routines GETPTR and PTRBYT. The new subroutine PTRINI properly configures the IO lines used with the reader. Subroutine PTRINI will output an endless number of 'X' characters until the tape is loaded into the reader. The listing was set prepared from a cross-assembly provided by the author. A symbol table showing the values of the symbols used and where they are referenced follows the assembly and will prove useful when it is necessary to relocate a program at a different starting address.

CROSS REFERENCE TABLE

Symbol	Value	Referenced
CHECK	4080	4085
GETPTR	406F	4004 4074 4088 4091 409D
LOAD	4004	4009 4049
LOADER	4053	403F 4046
LOADS	400B	****
LOAD2	402B	4037
LOAD3	403A	4029
LOAD7	404B	****
LOAD8	404D	4055
LOOP	409D	40A0
MAIN	409A	****
OK	406E	4064
PTRBYT	408B	4011 4018 4020 402B 403A 4041
PTRINI	4057	4001 406B 409A
PTRLD	4001	****
START	1C4F	4050

read until a semicolon is found; the next two characters give the hexadecimal number of bytes on the current line to be punched. This is followed by four characters, two bytes, giving the high order and low order bytes of the starting address for the data to follow. This is followed by the data which KIM-1 software always punches 24 bytes per line, a 2 character checksum for the line, and a carriage return. The carriage return is followed by six null characters, and a semicolon starts the next line. The last line punched contains 0 for the number of bytes, 0 for the address bytes and is followed by a four character checksum. When finished reading a paper tape, KIM-1 types 'ERR KIM' if the checksum does not compute (there has been an error in reading the tape), or just 'KIM' if the tape was read correctly.

The software consists of a copy of the KIM-1 monitor routine for reading paper tape modified by removing all calls to the GETCH and GETBYT routines and substituting two new routines, GETPTR and PTRBYT. A new subroutine, PTRINI is called at the beginning of the mainline program to properly configure the IO lines and check that the read head on the Fly Reader is closed. If it is not, KIM-1 will type out the character 'X' endlessly until the read head is closed. The software shown in listing 1 occupies 154 bytes starting at hexadecimal location 4000.

The Fly Reader is an excellent way to add a high speed paper tape reader to a microprocessor system. It is easy to interface and requires only a single +5 V supply. As a fast paper tape device it is considerably faster than an audio tape cassette system and offers increased flexibility of operation.■