Penny Pincher's Joystick Interface

Steven Wexler
1634 Buck Hill Dr
Huntingdon Valley PA 19006

One of the more entertaining input devices that can be operated by a human hand is the joystick. Physically, the device consists of a lever that moves in two dimensions. The lever operates two potentiometers, which translate the position of the lever into two analog resistance values. A joystick hardware interface, in conjunction with the appropriate software, can convert the resistance values into corresponding binary integer values. These integers can be used to move a cursor, alter music, or control a robot, along with a myriad of other applications.

There are several ways to interface a joystick to your computer. Each scheme has its advantages and disadvantages. The particular method I have chosen has the advantages of being inexpensive, easy to build, easy to understand, and of requiring a minimum of input/output (I/O) programming.

The disadvantages? This method is slower than some other interfaces I have seen, uses more software than do the expensive hardware-intensive schemes, and is less precise than some of the more elaborate circuit concoctions.

Operating Theory

The key to my "penny pincher's" joystick interface is the 556 dual timer configured as two monostable multivibrators or one-shots, as shown in figure 1. In English, this means that if you trigger the one-shot, its output will go high for a predetermined interval, after which the output will return to its normal low state.

By using a joystick potentiometer as a timing resistor, the duration of one output pulse will be proportional to the position, in one dimension, of the joystick lever. Software is used to convert the pulse duration into a binary value. Duplicating the circuit for the second timer, the other joystick potentiometer will yield a different output-pulse duration and binary value for the other dimension. Remember, joysticks operate in two or more dimensions.

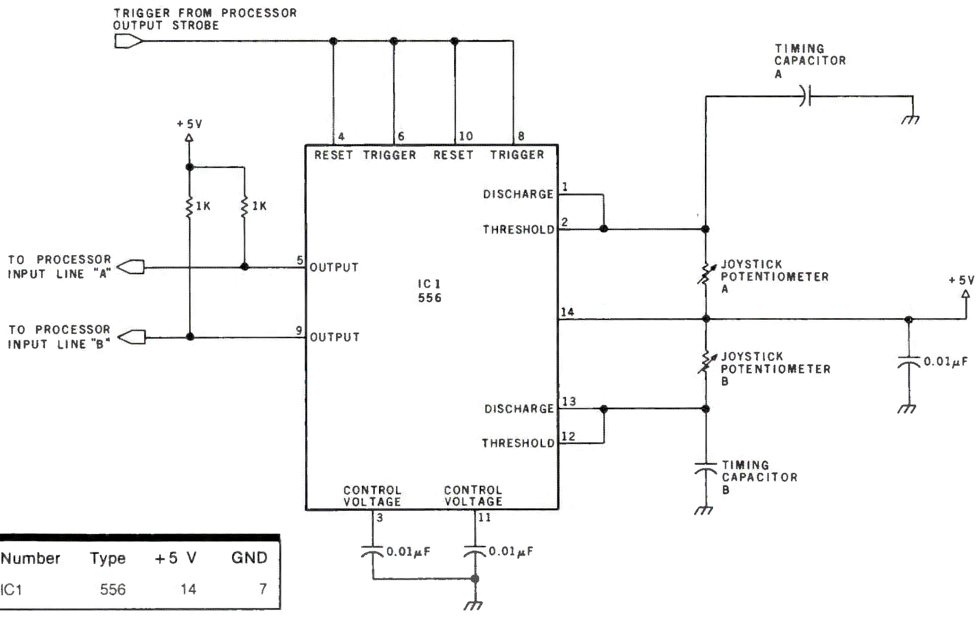
Joystick Interface Circuit

Careful study of figure 1 will reveal a most curious aspect of the interface. The *trigger* and *reset* lines for each circuit are all tied to a common processor output line. This certainly saves output lines, but how can you trigger and reset simultaneously? An explanation of the trigger requirements for the timer circuits should help to clear up this anomaly.

Normally, the timer will start to output a pulse on the high-to-low transition (ie: negative-going edge) of the input trigger signal. For the device to work properly, it is necessary to return the trigger input to its normal high state before the timed-output pulse returns low. In other words, before the device times out, the trigger input must go high.

If the timer receives a trigger signal in the middle of an output pulse, the signal is ignored. The obvious conclusion is that we must either trigger each of the 556 timers independently, or we must reset the second timer before it is triggered. Otherwise, how are we to avoid attempting to trigger the second timer before it has timed out from the initial signal? Tying the resets and triggers to a common computer-output line avoids the timing pitfall, while simplifying both hardware and software.

When the computer-output line goes low, the timing function is reset and the device returns to its initial state. As the processor-output line returns high (ie: positive-going edge), the circuit is reset before it is triggered; this allows the timing pulse to begin normally. The I/O line used to reset and trigger the 556 can also be used to reset and trigger additional joysticks. How's that for efficiency! I have not included the values of the timing capacitors and potentiometers



Number	Type	+5 V	GND
IC1	556	14	7

Figure 1: *The key to the penny pincher's joystick interface is the 556 dual timer, configured as two monostable multivibrators. The interval of each output pulse is determined by the joystick resistance, in conjunction with a user-selected timing capacitor.*

FINALLY! ONE ASSEMBLER — A·C·T™ BY SORCIM — LETS YOU ASSEMBLE CODE FOR 6502, 6800, 8080, 8085, AND Z80 CPUS . . . FOR ONLY $125!

In use for two years by **Sorcim** to facilitate program development, the **A.C.T.** (Assembly Code Translator) "universal" cross-assembler supports all 5 major processors and is now available to the public. Get out of writing hex and toggling switches; **A.C.T.** gives you efficient programming as well as the capability to develop code for one processor on a machine using a different processor. Running under CP/M* 1.4X or 2.X, **A.C.T.** features:

- Assembly speed greater than 1000 lines/min
- Complete cross reference
- Support for data type hex, octal, or binary (Intel or Motorola format)
- Supports macros
- Comprehensive set of pseudo-ops
- Absolute assemblies, system text file support, local proc definition (8080/5 & Z80 only), code file format (standard Intel hex), and many more special features.

A.C.T. fits in 24K of RAM (including CP/M), comes complete with manual and sample programs, and is shipped on a single-sided, single density, soft-sectored CP/M compatible 8" diskette. Want to know more about **A.C.T.**'s many talents? The manual is available separately for $15.

SORCIM

PO Box 32505
SAN JOSE, CA 95152

How to order: We accept UPS COD, Mastercard*, VISA®, personal checks, and certified checks. Californians add sales tax. Add $10 outside USA.

ALSO AVAILABLE FROM SORCIM FOR $175: PASCAL/M™, THE LOW-COST/-HIGH PERFORMANCE LANGUAGE FOR ALL Z80/8080/8085 COMPUTERS.

*CP/M is a trademark of Digital Research
A.C.T. and PASCAL/M are trademarks of Sorcim

in figure 1; these values depend on software, processor speed, and personal preference.

Software

The software needed for the penny pincher's interface is very straightforward. The 556 timers are triggered by setting the proper computer-output line first low, then high. After this, the processor should enter a tight, time-efficient counting loop until one circuit times out. The software should immediately store the count and then start the process over for the next timer. It is recommended that you disable interrupts during the counting process; otherwise an inaccurate count may occur.

Listing 1 presents the joystick-driving software for my KIM-1 computer (6502 processor). The program assumes that the reset/trigger line is tied to the KIM-1 I/O line B1. The timer's outputs are tied to B2 and B3; a second joystick may be tied to lines B4 and B5.

Utilizing consecutive I/O lines in this manner allows for efficient I/O line polling by merely shifting an I/O mask. Figure 2 is a flowchart of the

Circle 58 on inquiry card.

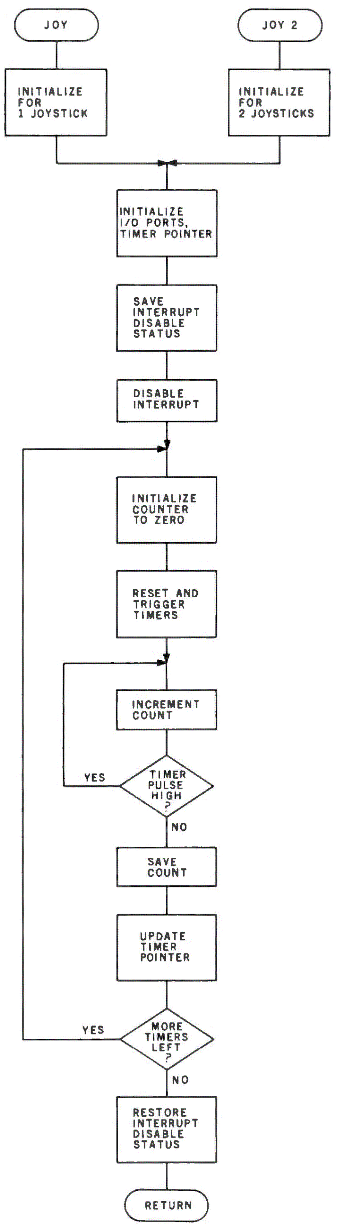

JOY
JOY 2
INITIALIZE FOR 1 JOYSTICK
INITIALIZE FOR 2 JOYSTICKS
INITIALIZE I/O PORTS, TIMER POINTER
SAVE INTERRUPT DISABLE STATUS
DISABLE INTERRUPT
INITIALIZE COUNTER TO ZERO
RESET AND TRIGGER TIMERS
INCREMENT COUNT
YES
TIMER PULSE HIGH ?
NO
SAVE COUNT
UPDATE TIMER POINTER
YES
MORE TIMERS LEFT ?
NO
RESTORE INTERRUPT DISABLE STATUS
RETURN

Figure 2: *The joystick-driving software consists mainly of a counting loop; this determines the stick position by timing the output pulse interval. High resolution can be attained by using a fast counting loop.*

Listing 1: *The software used on the author's KIM-1 system resets the interface timers with a low logic state on I/O line B1. When the same line goes high, the timers are retriggered. This technique, using only one output line, contributes to the simplicity of the hardware.*

```
                  POT    =$17E3     POT 1, Y AXIS
                  POT+1  =$17E4     POT 1, X AXIS
                  POT+2  =$17E5     POT 2, Y AXIS
                  POT+3  =$17E6     POT 2, X AXIS
                  PBD2   =$1702     PORT B DATA REGISTER
                  PBDD2  =$1703     PORT B DIRECTION REGISTER

8510  A2 01       JOY   LDX #1      ENTRY FOR ONE JOYSTICK.
8512  D0 01             BNE HOP     FORCED JUMP.
8514  A2 03       JOY2  LDX #3      ENTRY FOR TWO JOYSTICKS.
8516  A9 02       HOP   LDA #2      INITIALIZE TIMER POINTER.
8518  8D 03 17          STA PBDD2   SET LINE B1 FOR OUTPUT, REST INPUT.
851B  08                PHP         SAVE INTERRUPT STATUS.
851C  78                SEI         DISABLE INTERRUPT.
851D  0A          LP    ASL         UPDATE TIMER POINTER.
851E  A0 00             LDY #0      TRIGGER TIMER VIA
8520  8C 02 17          STY PBD2    LOW TO
8523  A0 02             LDY #2      HIGH TRANSITION
8525  8C 02 17          STY PBD2    OF LINE B1.
8528  A0 FF             LDY #FF     INITIALIZE COUNTER.
852A  C8          LP1   INY         UPDATE COUNT.
852B  2C 02 17          BIT PBD2    TEST TIMING PULSE.
852E  D0 FA             BNE LP1     IF HIGH, CONTINUE COUNT.
8530  48                PHA
8531  98                TYA
8532  9D E3 17          STA POT,X   SAVE COUNT.
8535  68                PLA
8536  CA                DEX
8537  10 E4             BPL LP      MORE TIMERS?
8539  28                PLP         NO, RESTORE INTERRUPT STATUS.
853A  60                RTS
```

program. Remember to keep the counting loop as efficient as possible.

Calibration

The count we obtain from the interface is equivalent to the duration of the timing pulse divided by the processing time required by the computer to execute one counting loop. My 6502 system, running at a clock frequency of 1 MHz, will execute the counting loop in listing 1 (hexadecimal 852A thru 852E) in 9 μs. It stands to reason that if you want a joystick to read from 0 to 100 on this machine, you would choose a potentiometer and capacitor that would set the maximum duration of the timing pulse to 909 μs (101 $\times$ 9 μs).

The following formula is used to derive the value of the timing capacitor:

$$C = \frac{\text{pulse duration}}{1.1 \times R}$$

where C is in farads, duration is in seconds, and R is in ohms. Assuming a joystick with 100 k-ohm potentiometers, a 0.0083 μF capacitor is needed to produce a 909 μs timing pulse. Since the actual value of most capacitors is not precisely known, it may be desirable to trim the maximum timer intervals. This can be done by placing extremely small-value capacitors in parallel with the main timing capacitor of the circuit that has the *smaller* maximum pulse of the two. Silver mica capacitors should work well here.

Construction

The circuit is quite simple and compact. With point-to-point wiring, several joystick interfaces can be constructed on a small circuit card. Placement of components is not critical. Each interface should draw less than 40 mA from a +5 V supply. Surplus joysticks can be purchased for about $4, while the 556 timer costs less than $1; so, for about $6 and one night's work, you can add this joystick interface to your system.■