Expand Your KIM

Part 5: A/D interfacing (for joysticks!)

John Blankenship
datamart, inc.
3001 No. Fulton Dr. N.E.
Atlanta GA 30305

No matter what kind of computer you have, this article can help you add four channels of analog input for a fraction of the cost of other methods I've seen. If you've been building the KIM System, this analog board will complete the project.

I designed the KIM System with several requirements in mind for the analog ports: I required four channels (so that two joysticks could be inter-

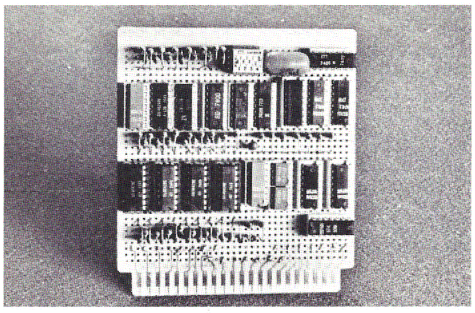

Photo 1. Four channels of A/D, two channels of D/A and an input port for sense switches.

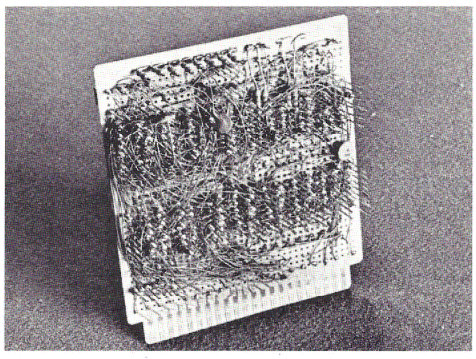

Photo 2. Circuits are wire-wrapped on a 44-pin board.

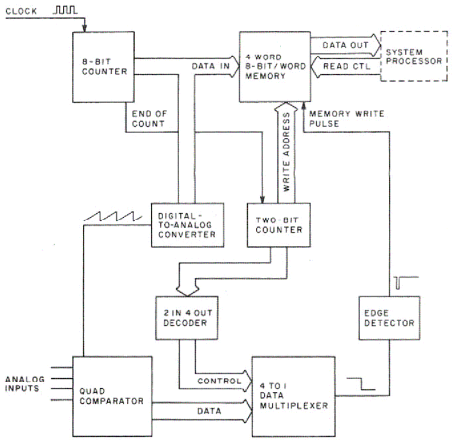

Fig. 1. Block diagram of the A/D converter.

faced), with each sampled often enough to provide reasonable accuracy for use as a video game input device. To make the use of these ports easy, I wanted each to be read as a normal memory. Finally, each of the A/D (analog-to-digital) channels had to be easily switchable to other devices besides the joysticks.

Besides the A/D ports, I also wanted at least two D/A (digital-to-analog) ports to experiment with music, speech synthesis, motor control, etc. I also wanted a port for sense switches to give me a full complement of methods for interfacing with my machine. I combined all these circuits on one board and labeled it *External Interfacing* in my previous articles.

Photos 1 and 2 show the board itself. Although I was able to cram the circuit onto a 4½-inch-square board, I would recommend epoxying a vector board on the top to give more room for the components.

Fig. 1 shows the basic block diagram for the A/D circuits. The four-word memory is one of the major secrets of making this circuitry both inexpensive and easy to use. This memory is made up of two 74LS170 chips composed of four 4-bit words each. I chose these chips because they have separate *read* and *write* controls, thus enabling read and write operations to occur simultaneously.

The A/D circuitry will update each of these memory locations with a number that is proportional to the analog input. The output of the memory chips is connected to the data bus so that they appear as standard memory to the processor.

The eight-bit counter continually generates sequential numbers from 0 to 255. A D/A converter converts these numbers to an analog voltage which, for all practical purposes, is an increasing ramp. This ramp is fed to four comparator circuits that compare the ramp voltage to the analog inputs.

The comparators output a level 1 when the ramp voltage equals the analog input. Since the ramp voltage also equals the number in the eight-bit counter, it is implied that the instant a comparator fires, the eight-bit counter contains the digital equivalent of the analog voltage being applied to that comparator.

The remainder of the circuit has one major function . . . it must decide which comparator fired, and form an address for the four-word memory so the eight-bit counter data can be gated into the appropriate location.

I chose to control the write address with a two-bit counter. Since this counter increments every time the eight-bit counter completes a full cycle, the addresses 0, 1, 2 and 3 are being applied sequentially to the *write* address, and each is held there for the full cycle of the eight-bit counter.

Additionally, this two-bit counter is decoded and used to enable only one of the four comparators (the one corresponding to the write address) at a time. The level change indication from the multiplexer is converted to a narrow pulse and used to activate the write line on the memory chips.

As explained above, the four memory locations are continually, and automatically, refreshed with the digital equivalent of four analog inputs. The processor needs only to read these locations for the latest updates.

Fig. 2 shows the actual schematic of the A/D circuit. The 7493 simply reduces the frequency to a trackable rate. The 1408L8, D/A converter, outputs a current ramp that is converted to a voltage ramp by the 741

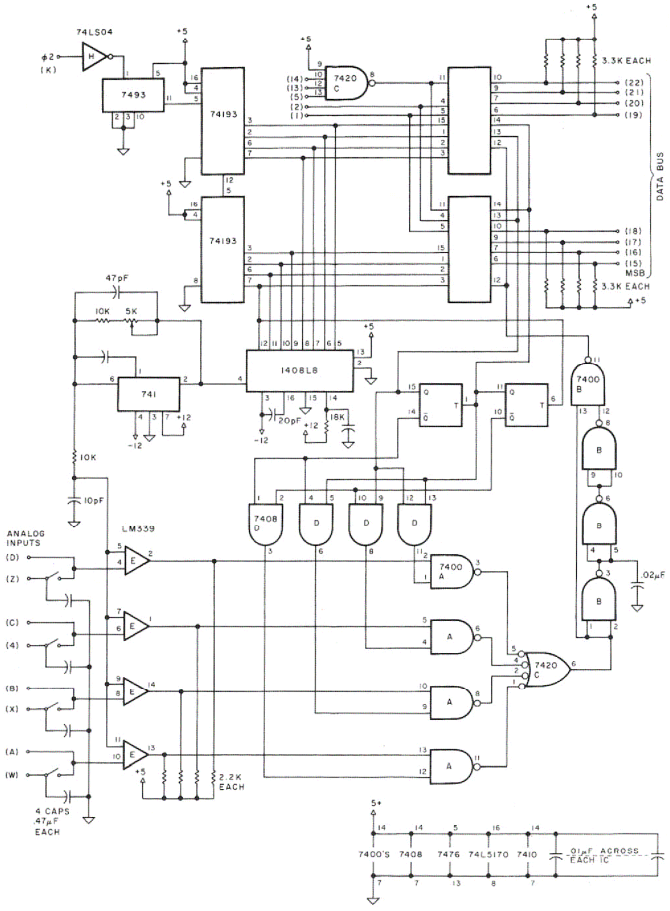

Fig. 2. Schematic of A/D converters.

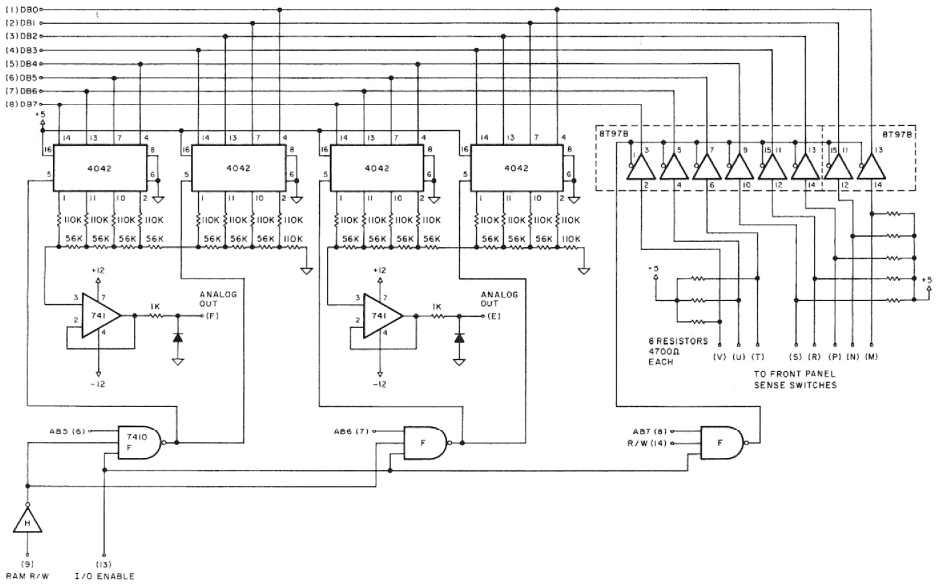

Fig. 3. Schematic of D/A converters and input port for sense switches.

op amp. The 7400 labeled B acts as a one-shot to perform as the edge detector.

Half of the 7420 is used to decode the address bus for processor reads. Address decoding will be discussed in more detail later in this article.

Since the 74LS170s are open collector, rather than Tri-state outputs, pull-up resistors are required for interfacing with the bus. The DIP switch disconnects the joystick inputs. Once they're disconnected, you can input other signals to the converter by way of the backplane jacks (see my earlier articles).

The other two functions, D/A and sense switches, are detailed in Fig. 3. Since I felt that the accuracy of the D/A conversion was not critical, I chose not to use the Motorola D/A converter chip used in the A/D circuit. If I had used the Motorola chip, I would have had to use two eight-bit registers to hold the data, the two D/A chips themselves and a current-to-voltage converter.

I chose to use MOS registers for my output ports. Since MOS gates output *exactly* Vcc and zero volts for their corresponding high and low levels, I used them to drive a resistive ladder directly. Additionally, since MOS chips represent a very small load, they can be hung on the bus without buffering. (Note: MOS chips do represent a relatively large capacitive load, and hanging them directly on the bus is not good practice in expandable systems. In this case, however, I knew exactly what loads I would be dealing with and was able to determine that enough drive capability was present.)

The 741s in Fig. 3 are used as unit gain amplifiers for buffering purposes. The 7410 is used to decode out the address lines to determine which port is being used. The sense switches are connected to the inputs of Tri-state buffers. The outputs of these buffers gate the switch data onto the bus when enabled.

Port Function	Page	Loc
Dazzler Mode control	80	0F
Dazzler ON/OFF, Address	80	0E
Right vertical joystick	80	10
Right horizontal joystick	80	11
Left vertical joystick	80	12
Left horizontal joystick	80	13
Sense switches	80	80
D/A port A	80	20
D/A port B	80	40

Fig. 4. Summary of special addresses used by the KIM-1 System.

A/D D	A	1	AB0
A/D C	B	2	AB1
A/D B	C	3	AB2
A/D A	D	4	AB3
D/A B	E	5	AB4
D/A A	F	6	AB5
−12	H	7	AB6
JS REF. Volt.	J	8	AB7
Ø2	K	9	RAM R/W
Ground	L	10	+12
SS 0	M	11	
SS 1	N	12	+5
SS 2	P	13	I/O ENABLE
SS 3	R	14	W/R
SS 4	S	15	DB7
SS 5	T	16	DB6
SS 6	U	17	DB5
SS 7	V	18	DB4
JS LH	W	19	DB3
JS LV	X	20	DB2
JS RH	Y	21	DB1
JS RV	Z	22	DB0

Fig. 5. Pin-out designations for the external interface board.

In order to better understand the I/O functions, you might reread my article ("Expand Your KIM!" Part 3, *Kilobaud*, February 1978, p. 68) in which I explain how I decoded part of the address lines to indicate an I/O operation, rather than a memory transfer.

All my I/O ports (including the four-word memory used for A/D) are partially enabled by this I/O enable. Since I know how many total ports I designed for, I only partially decoded the low-order address lines. This drastically limited the number of ports available on the KIM System, but the ease of implementation, as well as the reduction in cost, made it well worthwhile.

Fig. 4 summarizes the I/O addresses used uniquely by my system. If you convert these hex addresses to binary, you can see how the appropriate address lines are used to enable each port decoder.

There are only two major differences between input and output decoding. The first is that the R/W or the $\overline{R/W}$ line is used to indicate the direction of the transfer. Second, the *write* pulse for an output port must be coincident with the trailing edge of the Ø2 clock. Again, I refer you to Part 3 of this series for more details.

Fig. 5 shows the pin-out designations for the external interfacing board. These match the mainframe wiring done in Part 2 of this series.

In order to insure that builders of the KIM System fully understand how to utilize the joystick interface, I have included a short program in Fig. 5 that will enable you to draw with the joystick on the TV screen. The sense switches control the colors of the two-color dot that is moved by the joystick.

This program serves a useful function as an educational endeavor, and that's about all. However, I do feel that builders of the KIM System will find it useful as a reference. I have tried to functionally describe each section with comments.

This completes the hardware series on my KIM-1 system, which now contains 17K of RAM and supports both BASIC and FOCAL. I'm also in the process of implementing a new language with an ease of use and a speed of operation somewhere between assembly language and BASIC.

Because my system is to be multilingual, I have chosen to avoid ROM in favor of RAM for all functions except the KIM monitor. I'm also planning several surprises that I hope to share in the future. ■

Address	Contents	Label	Mnemonic
00 00	LOC PAGE	DATA	STORE POINTER
	:Set mode and starting address for the dazzler		
02	A9 10	INIT	LDA #$10
04	8D 0F 80		STA MODE
07	A9 90		LDA #$90
09	8D 0E 80		STA BEGADDR
	:Get horizontal joystick position		
0C	AD 11 80	START	LDA JOYHOR
	:Place 4 MSB into 4 LSB and save		
0F	4A		LSR
10	4A		LSR
11	4A		LSR
12	4A		LSR
13	85 00		STA LOC
	:Get vertical joystick position		
15	AD 10 80		LDA JOYVER
	:Check for and set up proper page of screen display		
18	30 07	TOP	BMI BOTTOM
1A	A0 20		LDY #$20
1C	84 01		STY PAGE
1E	4C 25 00		JMP CONT
21	A0 21	BOTTOM	LDY #$21
23	84 01		STY PAGE
	:Remove MSB and keep only the next four		
25	0A	CONT	ASL
26	29 F0		AND #$F0
	:Combine LSB and MSB into one word and save		
28	05 00		ORA LOC
2A	85 00		STA LOC
	:Put color (sense switches) into accumulator		
2C	AD 80 80		LDA SENSE
	:Prepare for an indirect store using 00 and 01 as pointer		
2F	A2 00		LDX #$00
	:Store color		
31	81 00		STA LOC PAGE
	:Begin Again		
33	4C 0C 00		JMP START

Fig. 6. Sample program for drawing on TV using joystick.