A 6502 Personal System Design:

David Brader
POB 483
Electric City WA 99123

Documenting Kompuutar – A Guide to the Details

The design information included with this article, together with the excellent documentation provided by MOS Technology on the 6502 design, should be complete enough to enable the advanced experimenter to build a similar Kompuutar. The details provided here cover a basic processor, but do not include a detail design of a programmable memory board which is a necessary part of a usable system. David Brader is currently working on an 8 K dynamic memory board, with invisible refresh, to be used in Kompuutar. The details of the wiring and construction of Kompuutar are shown in the several figures, tables and photographs, as well as listing 1. As a short guide to these materials here is a detailed table of contents to the article.

Front Panel Assembly: *This is the circuit with various displays and switches, which is mounted on the front panel, and talks to the front panel interface module via a multiconductor cable from P2 to J2.*

Front Panel Interface Module: *This is the logical interface between the processor's backplane bus and the front panel. It is the home of address decoding and the read only memory with the front panel service programs.*

Central Processing Module: *This is the heart of the Kompuutar system, a board which contains the 6502 processor, and associated buffering and clocking circuitry which defines the backplane bus structure of the system.*

TIM Interface Module: *This card is provided so that the MOS Technology "Terminal Interface Monitor," or TIM program, can be used with Kompuutar.*

Miscellaneous Items:

Kompuutar

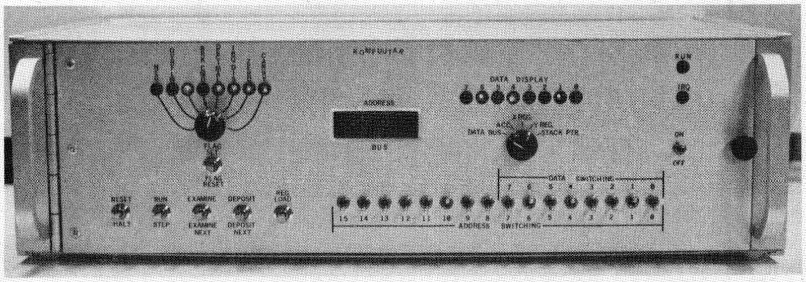


Photo 1: The completed Kompuutar, viewed towards its front panel. The controls of the front panel are chosen with the Data General NOVA's front panel as a mental model. In addition to the binary data display, there is a 4 digit hexadecimal address display (black rectangle) and an 8 bit binary flag display. The control panel is serviced by a read only memory routine.

Kaveat Kompuutar

It is with some trepidation that we present the details of the Kompuutar design. The design is complete and comprehensive, but Murphy is addicted to complete and comprehensive designs. Thus we'd like readers to be aware that there is a nonzero probability that errors exist in this magazine representation of author David Brader's design. We suggest that serious homebrewers of Kompuutar treat these pages as a detailed design guide, to be used with the standard design documentation of the chips involved. But as with any road map, do not be afraid to question and verify what you see with your own knowledge and experience.

David Brader reports that a local friend of his has built a second Kompuutar from the same set of blueprints which were the source of the circuit in this article. The experiences of the second builder were reflected in his corrections and changes to the drawings which are part of the normal "author proof" cycle applied to articles. Based on our own experiences with microprocessors, this report from David, and a tremendous amount of "desk debugging" of the article, we believe the information presented here is complete and buildable. However we highly recommend that readers who attempt to duplicate the design have sufficient experience with digital hardware and logic so that detailed understanding of its operation is possible. This is not a novice's project.

It all started at WESCON 1975, in San Francisco. It was there that I discovered what a "hospitality suite" is. In a hotel not far from the convention site, MOS Technology Inc had set up their WESCON hospitality suite. A hospitality suite is a bit like Las Vegas: some refreshments, a couple of elegantly decorative ladies, flashing lights and shiny gizmos, and the age old desire to persuade you and your money to part company.

I decided to stop and at least get a free drink. A man by the bar said, "Help yourself," so, being afraid of a one drink limit, I poured a double. As I left the bar area, I spotted a friend. We struck up a conversation about common friends and assignments, which lasted through half my drink and all of my clearheadedness. As our conversation ended, I noted some blinking LEDs and shiny new printed circuit boards. These boards were surrounded by several professional looking guests, giving the hardware an illusion of significance. So I went over to investigate.

I listened, wide eyed, to the saga of the MCS6502 as I slowly finished my drink. After the story ended, everyone seemed to be forming a line in a different part of the suite. Feeling part of the group now, I moved to the line. A little bit later, I remember being at the head of the line and the last thing I recall was handing two 20 dollar bills to a very pretty lady.

That evening, after sobering up, I discovered what I had done. There on my bed, stark naked, was a bright new MOS Technology Inc MCS6502 microprocessor chip and its manuals. Well, now the only

Backplane (P1) Pin Designation	Logic Diagram Mnemonics	Description	Backplane Pin Designation	Logic Diagram Mnemonics	Description
A	+ 5 V	voltage supply	1	GND	ground
B	IRQ 1	interrupt 1	2	IRQ 5	interrupt 5
C	A 0	address bus lines (even)	3	A 1	address bus lines (odd)
D	A 2		4	A 3	
E	A 4		5	A 5	
F	A 6		6	A 7	
H	A 8		7	A 9	
J	A 10		8	A 11	
K	A 12		9	A 13	
L	A 14		10	A 15	
M	IRQ 2	interrupt 2	11	IRQ 6	interrupt 6
N	IRQ 3	interrupt 3	12	IRQ X	any interrupt pending
P	D 0	data bus lines (even)	13	D 1	data bus lines (odd)
R	D 2		14	D 3	
S	D 4		15	D 5	
T	D 6		16	D 7	
U	SO	set overflow flag	17	SYNC	synchronize
V	PHICLK	Φ1 clock	18	PH2CLK	Φ2 clock
W	MASRST	master reset	19	RDY	ready
X	R/W	read and write	20	PANRST	panel reset
Y	IRQ 4	interrupt 4	21	NMI	nonmaskable interrupt
Z	GND	ground	22	+ 5 V	voltage supply

Table 1: Kompuutar bus list. This table gives the backplane socket pin identifications, mnemonics used in the logic diagrams, and a short description of the line's use. The pin designations are the standard ones printed on the Vector prototyping cards and embossed in the typical 44 pin sockets.

thing to do was to build a computer with the chip. After several days reading, I realized that building a computer was not going to be all that easy. I also realized that the initial $36.75 investment was but a drop in the proverbial bucket of costs.

Designing the Kompuutar System

Since my $36.75 investment was going to need considerable financial and design support, it was clear that making a project out of the computer would require planning. The first thing I had to accomplish was a specification of the features I wanted in my machine. I had had a good deal of experience with the Data General NOVA 1200 minicomputer, which led me to favor its functional front panel switch setup. With this input, I decided that the new machine would have the front panel functions of master reset, halt, program run, single instruction step, memory examine, examine the next memory location, deposit, desposit to the next memory location, load processor register, and enter data or address information from switches. I also knew that I wanted to be able to display the information on the data lines and address lines. I decided to use hexadecimal LED displays for the address bus information, but not for the data bus. My reasoning was that the address bus is always considered to be a numerical value, whereas the data bus is sometimes considered to be numeric data, but is sometimes viewed as a combination of individual bits. (If the data bus was showing hexadecimal E6 and you wanted to know if bit 5 was on or off, you would probably have to think for a while to make sure.) Another argument in favor of discrete LED indicators for each bit is the fact that hexadecimal displays are a bit more expensive.

After reading more about the MCS6502, a trait common to the other single chip processors revealed itself. The status register, accumulator, index register X, index register Y and stack pointer register do not come out of the chip on their own sets of pins. All that information was going to be hidden from the operator (me) sitting in front of the machine. I knew I would have to design digital logic to get that information out of the chip and displayed upon some sort of front panel. I even decided to go one step further and build in the capability to set or reset the status flags from the front panel. Being able to throw a switch and set the carry flag, for example, is a very handy capability when debugging a conditional branch in some program.

I decided to use toggle switches for the 16 data inputs because the state of individual switches could then be tested in software and used to control options in a program. This complicates the entry of an address (which is displayed in hexadecimal) but gains an ability to write applications programs which can be modified by the state of these input switches.

With these considerations in mind, the front panel design was firmed up as a starting point for the processor. I then started to work on the detailed logic design of what came to be called Kompuutar in my lexicon. After a month's work, I realized that the front panel logic was going to contain nearly 220 TTL integrated circuits if I implemented it with a conventional logic design. After that false start, I thought about a simplification made possible by a read only memory program, or "firmware" as it is sometimes called. I could replace most of the front panel logic with a program burned into a single read only memory integrated circuit. With a PROM program and 16 bytes of volatile programmable memory, the 6502 processor itself would operate the front panel of the system. The integrated circuit count for the front panel including the programmable read only memory and 16 bytes of volatile solid state memory was now reduced to just over 50 packages.

System Design Philosophy

During the process of designing the front panel, I worked out a total system design philosophy which goes like this:

- There would be a central processing unit and peripherals. The peripherals would be interfaced to the processor with a minimum of hardware by using memory address interfaces wherever possible.
- The system would be modular. A common backplane would be defined. Each module would be connected to the other modules through this backplane. Each socket on the backplane would be wired pin by pin to every other socket on the backplane.
- An address allocation map for the system would be defined. This would define addresses for hardware (peripherals), firmware (read only memory programs) and main programmable memory use.

The front panel design I had already created follows the first point of this philosophy quite well. It has several separate peripherals. Some are input devices, some are output devices, and some are a combination of both functions. Each is interfaced as a memory address and operated by firmware with a minimum of supporting hardware. Details of front panel operation will be discussed a little later in this article.

The physical arrangement of the design implements the details of the second point in the philosophy. The front panel assembly is connected to the top of a Vector prototyping card which contains the programmable read only memory with the front panel servicing routines. This card in turn plugs into the backplane bus which is implemented with a Vector card cage and edge connectors. By pulling the front panel card out of the backplane, the Kompuutar system can be isolated from the front panel completely. Similarly, the rest of the Kompuutar system is fabricated on Vector 3662 cards. Each card module contains one complete section of the system. These modules include the central processing unit card with the 6502 and bus interfacing chips, and a terminal interface card. Eventually 8 K byte programmable (volatile) memory cards will be part of the system. The cage I used has room for eight memory cards for a total of 64 K bytes. Since the backplane is wired from pin to corresponding pin of each socket, the cards can be placed in any available socket in the card cage. Table 1 shows the definitions of all the bus pins. In developing the system, I used an extender card plugged into the backplane so that I could have access to the various modules with an oscilloscope probe.

The third part of the design philosophy

Address Range	Type of Hardware	Usage of Region
0000 to 3FFF	Volatile programmable memory	This is the general programmable memory region for user applications programs. Locations 0000 to 01FF are dedicated to scratch pad and stack use by the architecture of the 6502 processor.
4000 to 6FFF	Unimplemented	This region is reserved for 12 K of general user memory expansion.
7000 to 73FF	TIM read only memory	When the TIM monitor interface card is in the system, this area is reserved.
7400 to 7FFF	Unimplemented	
8000	Scratch pad memory with external visibility	Current accumulator value maintained by front panel service program
8001	Scratch pad memory with external visibility	Current X register value
8002	Scratch pad memory with external visibility	Current Y register value
8003	Scratch pad memory with external visibility	Current processor flag values
8004	Scratch pad memory	
8005	Scratch pad memory with external visibility	Current stack pointer value
8006	Scratch pad program begins here	
8007 to 8008	Scratch pad	Current address register value, displayed through locations 8014 and 8015
8009 to 800C	Scratch pad program area	
800D	Scratch pad memory with external visibility	Current data at memory location in address register locations 8007 to 8008
800E-800F	Scratch pad	
8010	Read only data input	Front panel request register (see table 3)
8011	Read only data input	Low order address and data switch register
8012	Read only data input	High order address switch register
8013	Write only display	Flag data latch and binary display
8014	Write only display	Low order address display latch
8015	Write only display	High order address display latch
801F	Idle command address	References cause processor to idle
8020 to 80FF	Peripherals	Unimplemented hardware device addresses
8100 to EFFF	Unimplemented	
F000 to FDFF	Programmable read only memory allocations for systems programs	This area is expected to be used by interrupt service routines, utility subroutines and the like, programmed into read only memory parts.
FE00 to FFFF	Read only memory	This region is allocated to the firmware which controls the front panel. The 6502's interrupt vectors are programmed into the last portion (see listing 1).

Table 2: A memory allocation map for Kompuutar. When interfacing both peripherals and programming to a single memory address space, it helps to make a memory map to keep track of allocations.

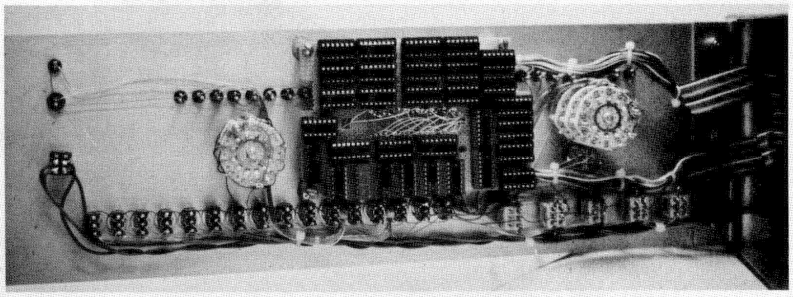


Photo 2: The reverse side of the front panel assembly for Kompuutar. The various switches, indicators and the front panel electronics board are seen in this picture. The P2-J2 cables run to the front panel interface module at right.

was implemented by picking address allocations. (See table 2 for a detailed list of the allocations.) I decided early in the project that 16 K bytes of memory would be a good start for general programming uses. I had to allocate this volatile user oriented programmable memory, as well as all the addresses for peripheral hardware and "firmware" read only memory programs. The address range of the 6502 is from 0 to 65,535 (0000 to FFFF in hexadecimal). Since the architecture of the chip itself uses addresses 0000 to 01FF for dedicated functions which must be in programmable memory, I assigned the 16 K byte block of main memory to the lowest part of the addressing range, from hexadecimal 0000 to 3FFF. I was interested in the possibility of occasionally using the MOS Technology TIM monitor, so I reserved locations 7000 to 73FF for use by that program's read only memory. I allocated the control panel scratch memory and peripheral ports starting at address 8000 hexadecimal, with the addresses starting at 8020 reserved for general peripheral use as I expand the system. At the end of the address range, I reserved the 4096 bytes from addresses F000 to FFFF for read only memory containing various systems routines. The high end of this range is reserved for the control panel support program and the interrupt vectors of the MOS Technology 6502 design.

Backplane

The backplane of the card cage (see table 1) carries the address bus, the bidirectional data bus, six vectored interrupt lines, and other functional signals as detailed in table 1. All signals that pass through the backplane are interpreted to be logical 1 or "true" in a low voltage (TTL 0) state. A high voltage (TTL 1) state is interpreted as a logical 0 or "false" state. Each module which connects to the backplane uses TTL inverting buffer circuits for signals sent or received. The +5 V (VCC) and ground (GND) connections are arranged on the card edge connectors such that by plugging a module into the backplane upside down, polarity to the card will not be reversed. This simple arrangement eliminates the need for keying the cards; while it prevents physical destruction of the card due to inadvertent reversal of orientation, ·the system should not, of course, be expected to work with one or more cards reversed relative to the balance of the cards in the system.

The vectored interrupt lines of the backplane are defined by some logic implemented on the central processing unit card (see figures 3). This card contains logic necessary to cause hardware vectoring of interrupt levels to one of the six possible interrupt service routines. The vectoring

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Flag Select Bits			Flag Control	Control Request Functions			

Flag Select Bits:

0 = CARRY flag
1 = ZERO flag
2 = IRQ DIS flag
3 = DECIMAL flag
4 = BRK CMD flag
5 = Not used
6 = OVRFLW flag
7 = NEG flag

Flag Control:

"1" = set flag
"0" = reset flag

Control Request Functions:

0 = Null command (pass NMI)
1 = FLAG MODIFICATION
2 = DEPOSIT NEXT
3 = DEPOSIT
4 = EXAMINE NEXT
5 = EXAMINE
6 = STEP
7 = HALT
8 = REG LOAD
9 to 15 = Unused

Table 3: Control request word layout. The control request word, located at address 8010 in memory address space, is an input to the processor with this format. It is used by the front panel service program of listing 1 to govern the operation of the panel based on settings of various switches.

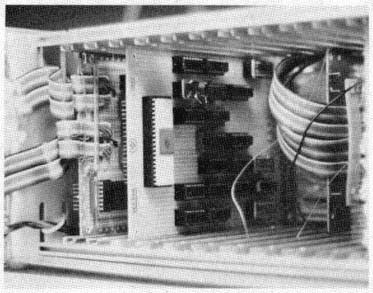


Photo 3: The front side of the backplane framework with card guides. This picture shows the front panel interface module at the left, the central processor module, a gap of several card slots, then the TIM interface module's edge with cables dangling.

is accomplished by using the selected service routine address for the interrupt vector requested by the 6502 processor during its interrupt sequence. The service routines are assumed to reside in programmable read only memory chips located in a separate module elsewhere on the backplane bus. The processor hardware also incorporates a priority arrangement of these six interrupts. Thus when two interrupts occur simultaneously the system has no problem: the higher priority one is serviced first. This becomes important when several interrupt driven peripherals are used with the system. An automatic reset function initializes the system when power is turned on, a separate interrupt for the 6502 which is supported on the processor board.

As an alternative to the front panel logic, the memory map of table 2 shows allocations for the MOS Technology TIM monitor integrated circuit, MCS6530-004. This "Terminal Interface Monitor" allows the user to use an ASCII serial device such as a Teletype or other terminal. In making a board to support TIM, I also included an 8 bit parallel interface to allow the possibility of using a high speed paper tape reader with Kompuutar. Details of the TIM module are shown in figures 4.

Front Panel Logic

Getting into more of the details of the system, I'll concentrate mainly on the place where I started my design, the front panel. The front panel logic is composed of input devices, output devices, 16 bytes of scratch pad memory, logic of the ready and nonmaskable interrupt timing, address decoders, switch debouncers, a data bus multiplexer, command encoder, line buffers and the control program in a programmable read only memory. The overall design of the front panel is found in figure 1, with details spread out in figures 1.1 thru 1.9. Photos 1 and 2 give further details.

There are four input sources of data in the front panel design. Each source is selected by the address decoding logic, which in turn allows the proper source to be input through the data bus multiplexer. The first source of input is the control request register. This source carries data from the command encoder, the flag selection switch, the flag modification switch and the register load switch. Table 3 shows the bit assignments of this source, which is located at hexadecimal address 8010 in memory address space.

The second source of input is the low

Text continued on page 112

Wire	Logic Diagram Mnemonic	Description	Wire	Logic Diagram Mnemonic	Description
1 to 4	+5 V	voltage source	21	X800X	address selection lines
5	D0	data bus lines	22	X800F	
6	D1		23	X8013	
7	D2		24	X8014	
8	D3		25	SEL 1	multiplexer select lines
9	Φ1 CLK	phase 1 clock	26	SEL 2	
10	Φ2 CLK	phase 2 clock	27	BSOUT	multiplexer disable
11	IRQX	any interrupt pending	28	—	—
12	A3	address bus line	29	X8015	address selection lines
13	D4	data bus lines	30	XFEEA	
14	D5		31	X801F	
15	D6		32	—	—
16	D7		33	—	—
17	A2	address bus lines	34	REST	front panel reset
18	A1		35	RDY	ready
19	A0		36	NMI	nonmaskable interrupt
20	R/W	read/write	37 to 40	GND	ground

Table 4: Wiring list for the J2-P2 cable. This cable runs from the front panel interface board in the card cage to the front panel assembly, as seen in photos 2 and 3. [In the author's version of Kompuutar, the wiring was direct without use of a plug and jack; in order to simplify nomenclature in presenting the article, we've used a numerical indentification of signal paths as if a 40 wire cable and connectors had been used . . .CH]

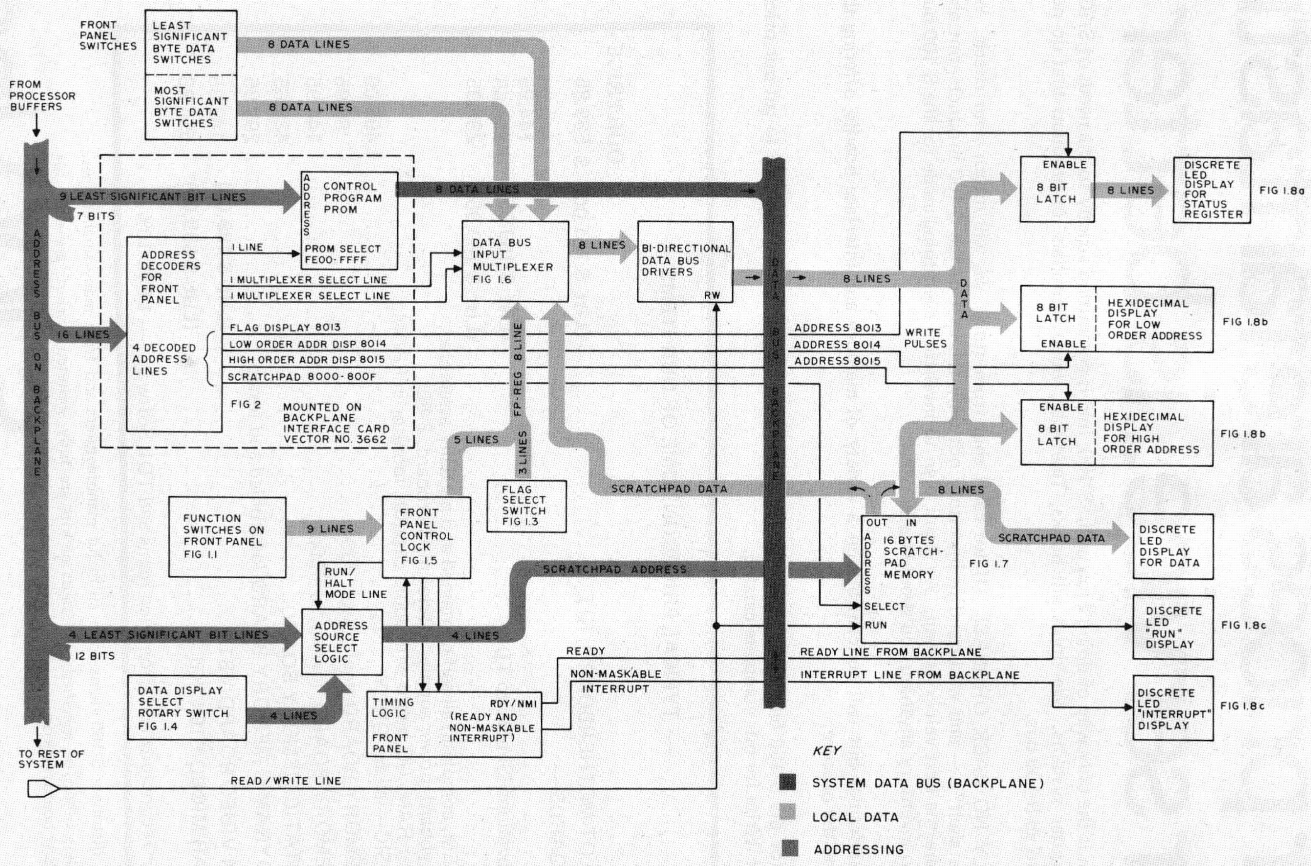
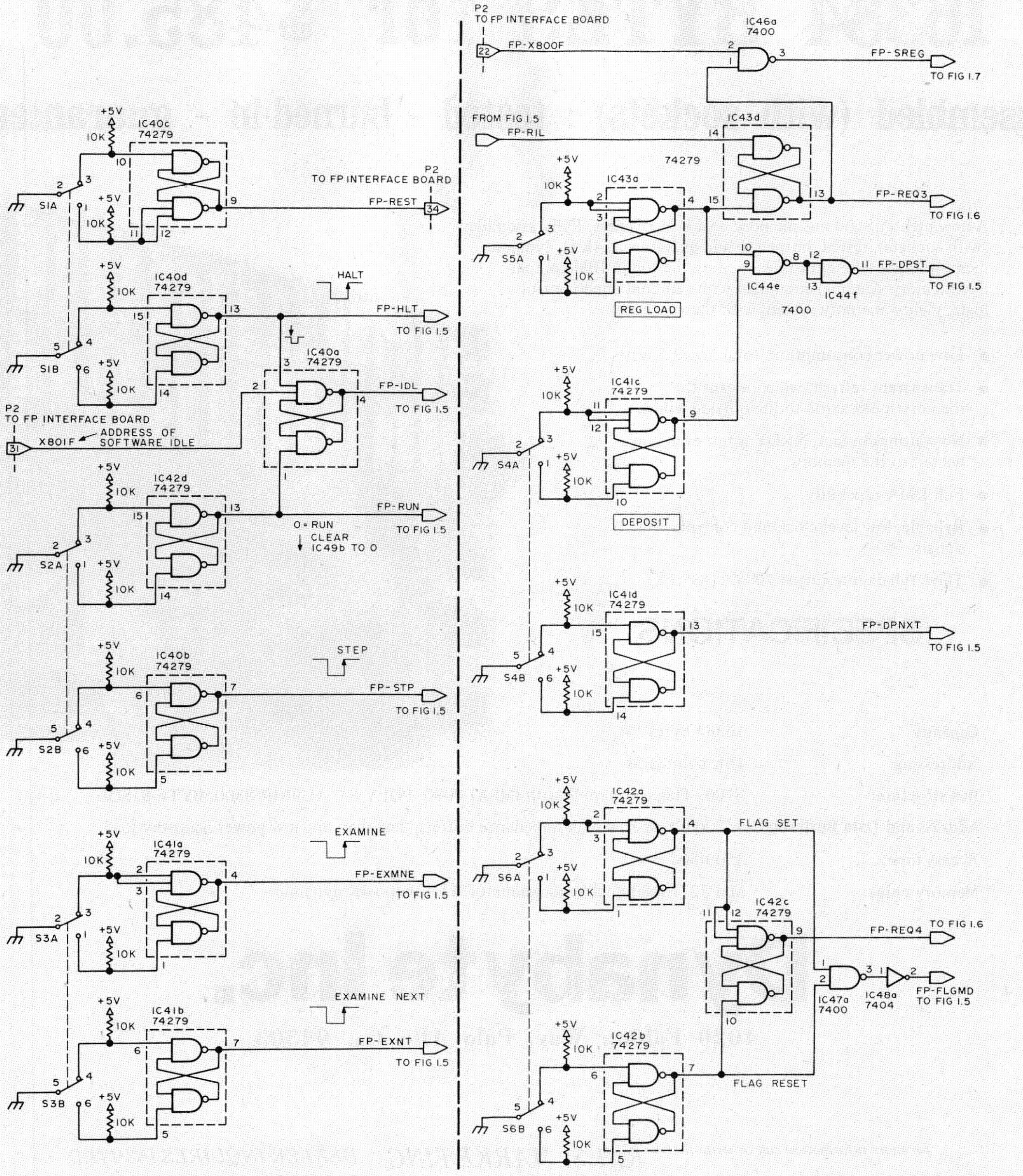

FRONT PANEL SWITCHES
LEAST SIGNIFICANT BYTE DATA SWITCHES
MOST SIGNIFICANT BYTE DATA SWITCHES
8 DATA LINES
8 DATA LINES
FROM PROCESSOR BUFFERS
ADDRESS BUS ON BACKPLANE
9 LEAST SIGNIFICANT BIT LINES
7 BITS
ADDRESS
CONTROL PROGRAM PROM
PROM SELECT FE00-FFFF
8 DATA LINES
16 LINES
ADDRESS DECODERS FOR FRONT PANEL
1 LINE
1 MULTIPLEXER SELECT LINE
1 MULTIPLEXER SELECT LINE
FLAG DISPLAY 8013
LOW ORDER ADDR DISP 8014
HIGH ORDER ADDR DISP 8015
SCRATCHPAD 8000-800F
4 DECODED ADDRESS LINES
FIG 2
MOUNTED ON BACKPLANE INTERFACE CARD VECTOR NO. 3662
DATA BUS INPUT MULTIPLEXER FIG 1.6
8 LINES
BI-DIRECTIONAL DATA BUS DRIVERS
RW
FP-REG 8 LINE
5 LINES
3 LINES
FLAG SELECT SWITCH FIG 1.3
SCRATCHPAD DATA
FUNCTION SWITCHES ON FRONT PANEL FIG 1.1
9 LINES
FRONT PANEL CONTROL LOCK FIG 1.5
RUN/HALT MODE LINE
SCRATCHPAD ADDRESS
4 LEAST SIGNIFICANT BIT LINES
12 BITS
ADDRESS SOURCE SELECT LOGIC
4 LINES
DATA DISPLAY SELECT ROTARY SWITCH FIG 1.4
4 LINES
TIMING LOGIC
FRONT PANEL
RDY/NMI (READY AND NON-MASKABLE INTERRUPT)
READY
NON-MASKABLE INTERRUPT
TO REST OF SYSTEM
READ/WRITE LINE
DATA BUS BACKPLANE
8 LINES
DATA
ADDRESS 8013
ADDRESS 8014
ADDRESS 8015
WRITE PULSES
ENABLE
8 BIT LATCH
8 LINES
DISCRETE LED DISPLAY FOR STATUS REGISTER
FIG 1.8a
8 BIT LATCH
ENABLE
HEXIDECIMAL DISPLAY FOR LOW ORDER ADDRESS
FIG 1.8b
ENABLE
8 BIT LATCH
HEXIDECIMAL DISPLAY FOR HIGH ORDER ADDRESS
FIG 1.8b
8 LINES
OUT
IN
ADDRESS
16 BYTES SCRATCH-PAD MEMORY
SELECT
RUN
FIG 1.7
SCRATCHPAD DATA
DISCRETE LED DISPLAY FOR DATA
DISCRETE LED "RUN" DISPLAY
FIG 1.8c
READY LINE FROM BACKPLANE
INTERRUPT LINE FROM BACKPLANE
DISCRETE LED "INTERRUPT" DISPLAY
FIG 1.8c
KEY
SYSTEM DATA BUS (BACKPLANE)
LOCAL DATA
ADDRESSING

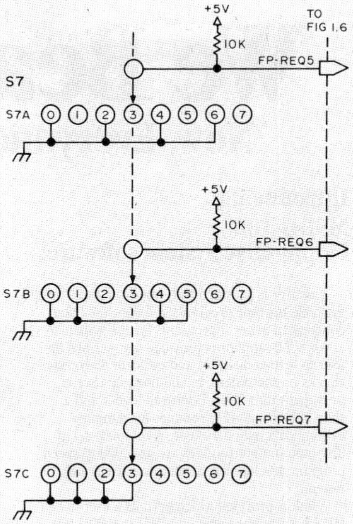
Figure 1: Block diagram of Kompuutar's front panel logic. The Kompuutar design uses a read only memory program to manipulate the contents of memory interactively using function switch inputs and solid state display outputs. This diagram serves as a functional road map to the various components of the display and its interface board.



NOTE: ALL PULLUP RESISTORS ARE PHYSICALLY MOUNTED ON THEIR ASSOCIATED SWITCHES.

Figure 1.1: Switch debouncing logic. This is a detail logic diagram suitable for construction of Kompuutar. As in all the logic of this design, all resistors are 1/4 W unless otherwise noted, and standard TTL integrated circuits are used for miscellaneous functions. Debouncing is done with set-reset flip flops contained in the 74279 part, which we have noted in the discrete logic form internal to dotted lines. The flip flops can be wired out of gates (7400, 7410) if desired, should the 74279 be unavailable in the builder's parts bin. Integrated circuit power wiring for the entire design is summarized by IC number in table 5.

Figure 1.2: Flag selection switch. The control panel service program of Kompuutar uses the binary encoded 3 bit value on the output of this switch to determine which processor flag is to be set or reset using an appropriate function selection. This switch is a rotary switch which has three poles and eight positions.



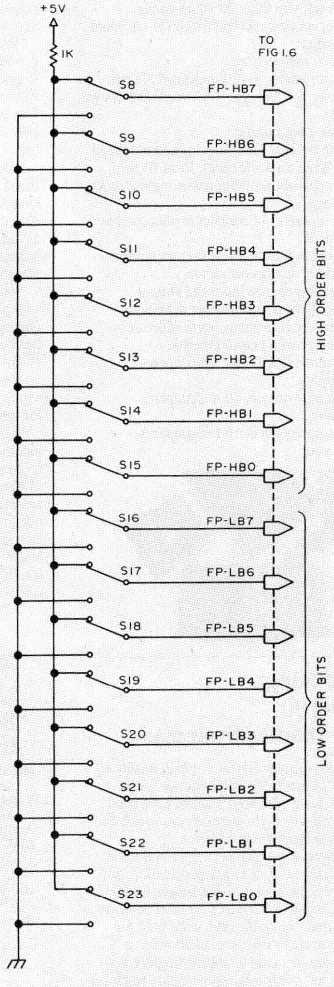
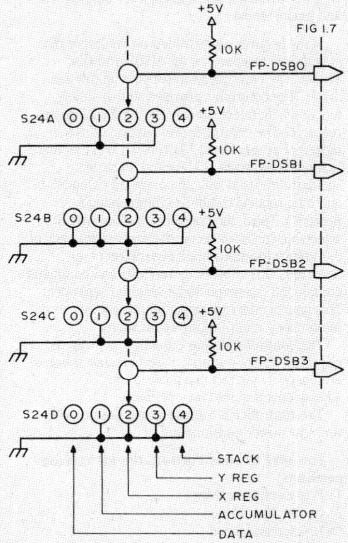


Figure 1.3: Front panel data entry switches. The entry of static data is accomplished by 16 toggle switches. These switches are connected to either logical 1 (+5 V through a 1 K resistor) or logical 0 (ground).

Figure 1.4: Display data selection switch. The control panel service program uses the binary encoded 4 bit value on the output of five possible words for default display from the control panel scratch pad located at addresses 8000 to 800F. The five addresses selected are for the accumulator (0), X index (1), Y index (2), stack register (5) or data register (D).



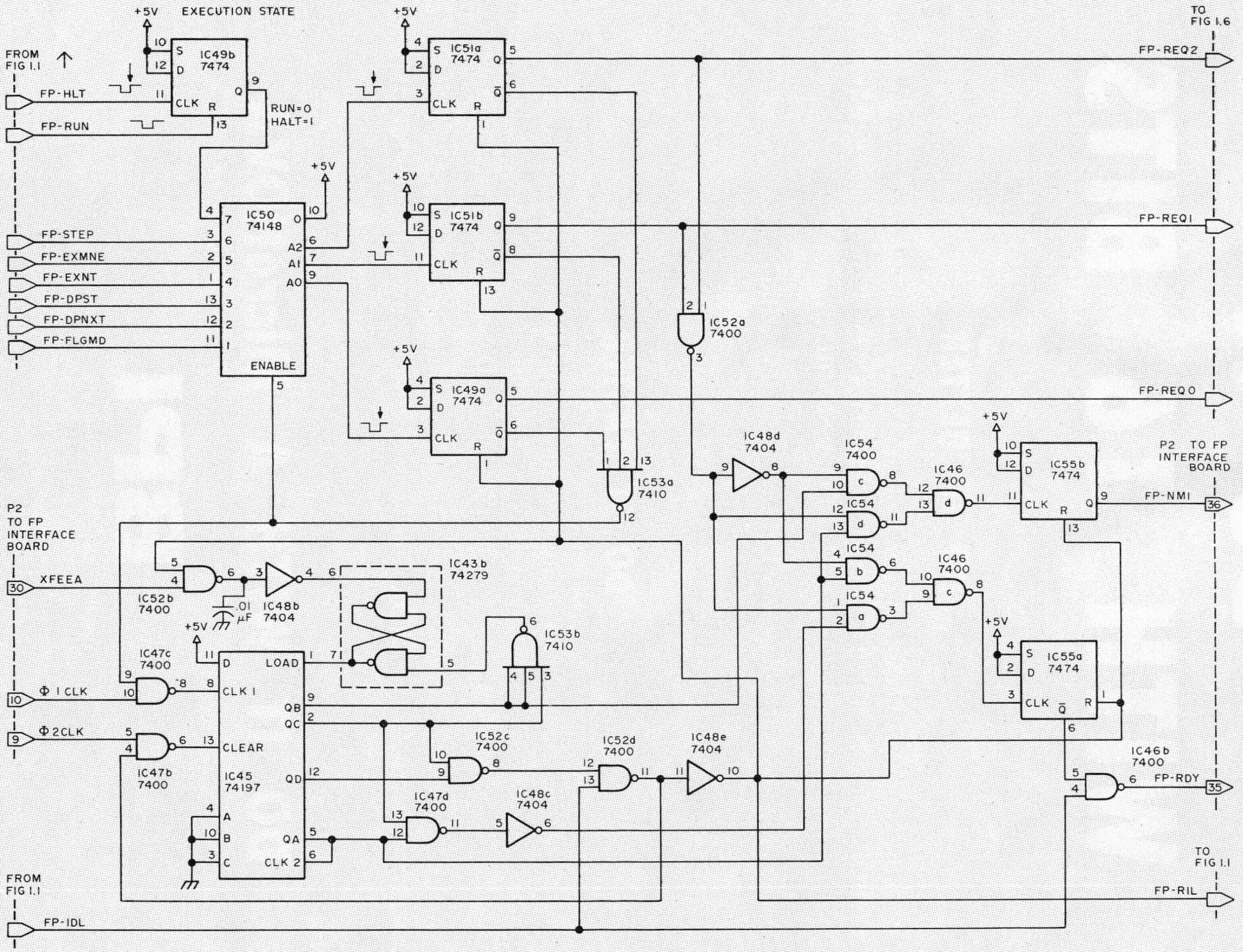

EXECUTION STATE
+5V
FROM
FIG 1.1
FP-HLT
FP-RUN
IC49b
7474
RUN=0
HALT=1
IC50
74148
ENABLE
FP-STEP
FP-EXMNE
FP-EXNT
FP-DPST
FP-DPNXT
FP-FLGMD
IC51a
7474
IC51b
7474
IC49a
7474
S
D
CLK
R
Q
Q̄
A2
A1
A0
IC52a
7400
IC53a
7410
TO
FIG 1.6
FP-REQ2
FP-REQ1
FP-REQ0
IC48d
7404
IC54
7400
IC46
7400
IC55b
7474
IC55a
7474
P2 TO FP
INTERFACE
BOARD
FP-NMI
36
IC46b
7400
FP-RDY
35
TO
FIG 1.1
FP-RIL
P2
TO FP
INTERFACE
BOARD
XFEEA
30
Φ 1CLK
10
Φ 2CLK
9
IC52b
7400
IC48b
7404
.01
μF
IC43b
74279
IC53b
7410
IC47c
7400
IC47b
7400
IC45
74197
LOAD
CLK 1
CLEAR
QB
QC
QD
QA
CLK 2
A
B
C
IC52c
7400
IC52d
7400
IC48e
7404
IC47d
7400
IC48c
7404
FROM
FIG 1.1
FP-IDL

Figure 1.5: Control logic for front panel functions. This logic generates the function request code (read from address 8010 bits 0 to 2), and controls the NMI line of the 6502 to implement single step execution of the processor.

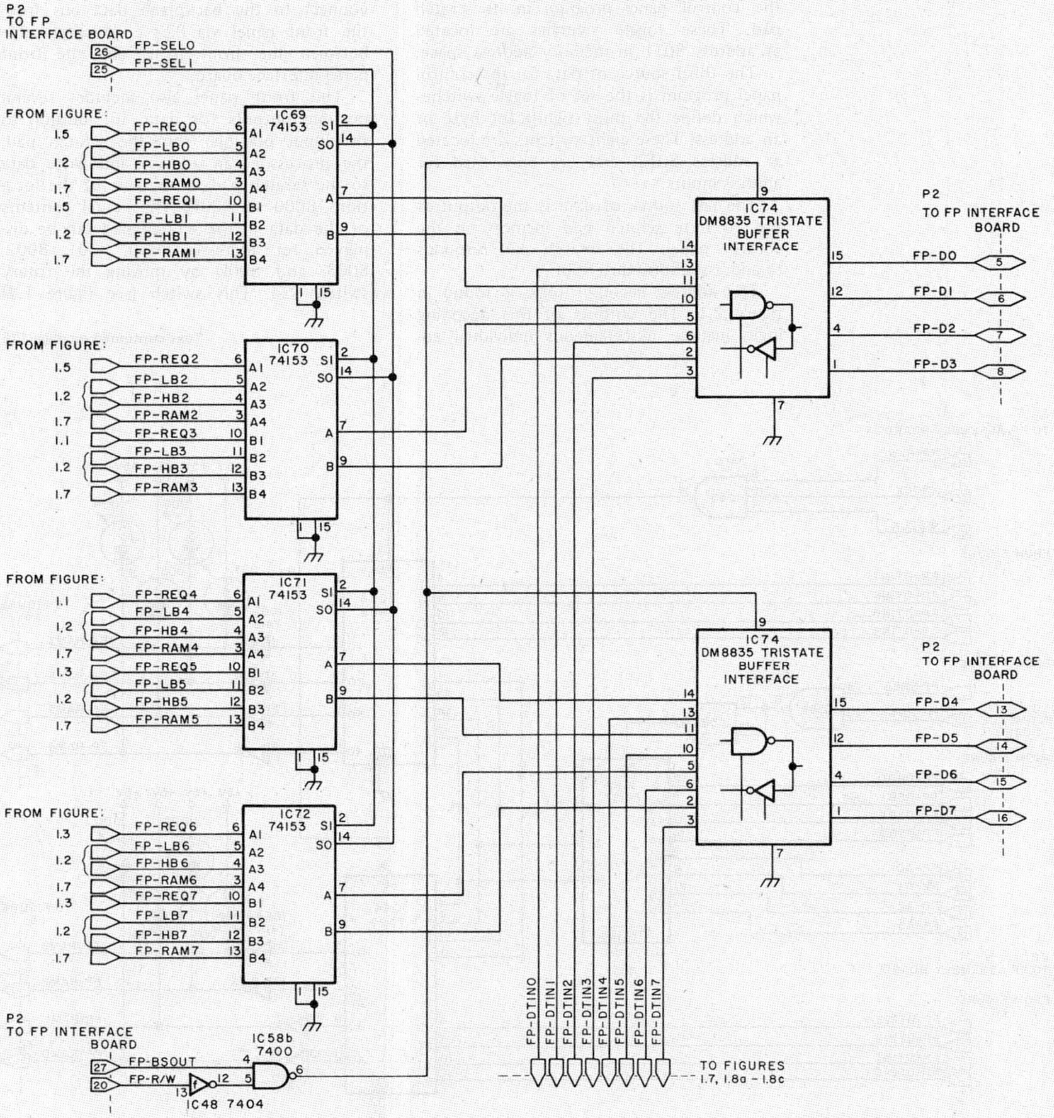


Figure 1.6: Data source multiplexer and bus interface. The sources of data read from the front panel logic are four: the two 8 bit data entry switch registers of figure 1.3, the 8 bit control request word from figure 1.5, and the output of the scratch pad programmable memory (lines labeled "RAM") from figure 1.7. These are selected by a 2 bit addressing code generated on the front panel interface board of figures 2.

Text continued from page 102

order byte of the data entry switches. The eight toggle switches of this switch register are used to enter a byte into the data bus or into the least significant byte of the address register which is maintained by the control panel program in its scratch pad. These toggle switches are located at address 8011 in memory address space.

The third source of data for the control panel program is the set of toggle switches which define the most significant byte of an address. These eight switches are located at address 8012, and are only used for address inputs.

The last source of data is the output of the 16 byte scratch pad memory in the control panel. The scratch pad responds to addresses 8000 thru 800F.

The address decoding logic is found in figure 2.1. The outputs of this decoding logic include miscellaneous individual address selections, plus the selection signals which are used to control the data input multiplexer found in figure 1.6. The selection signals are generated by the priority encoder IC35, and are used to pick one of the four sources for routing to the bus interface gates IC73 and IC74. These gates connect to the backplane data bus from the front panel via P2's connecting cable between the front panel and the front panel interface board.

The front panel also includes several possible outputs for data. In addition to the input possible from the scratch pad, the processor can address and write data to the scratch pad in any one of the locations 8000 to 800F. The actual contents of the data in the scratch pad can be displayed for addresses 8000, 8001, 8002, 8003, and 800D by moving the rotary switch S24. This switch (see figure 1.4)

Text continued on page 116

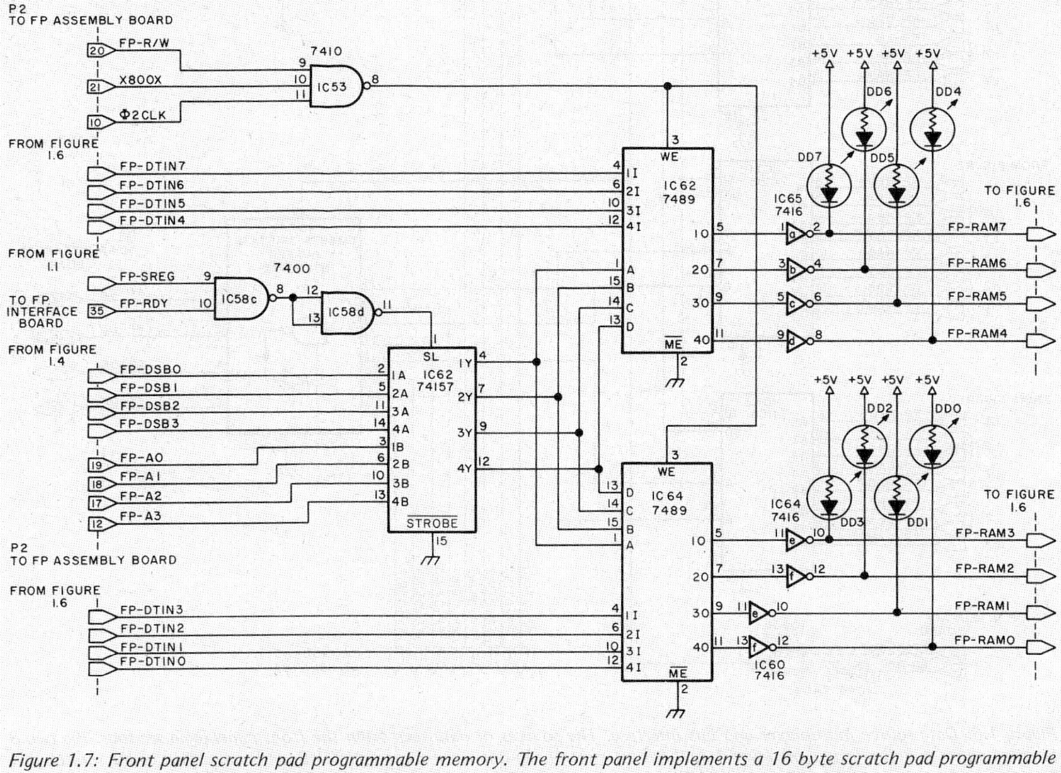


Figure 1.7: Front panel scratch pad programmable memory. The front panel implements a 16 byte scratch pad programmable memory at addresses 8000 to 800F. This memory is used for data storage and for storage of a scratch pad program segment which is modified during execution of the front panel service routines.

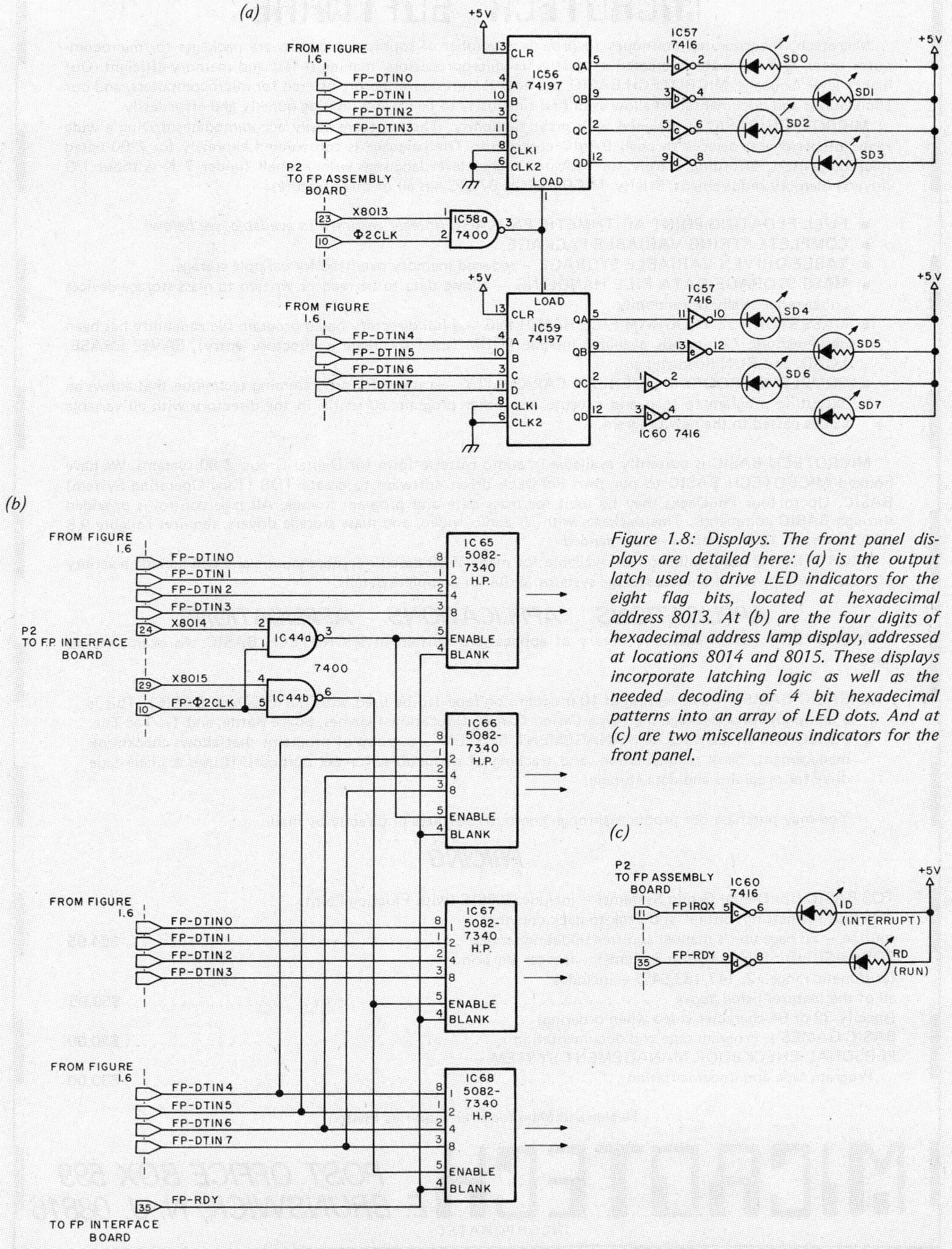
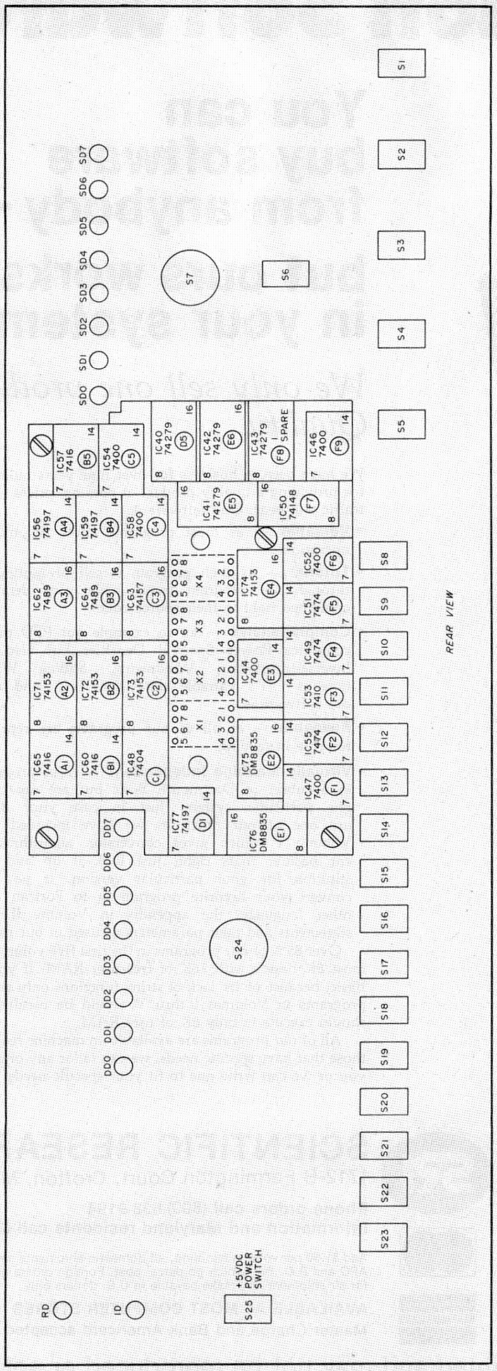


Figure 1.8: Displays. The front panel displays are detailed here: (a) is the output latch used to drive LED indicators for the eight flag bits, located at hexadecimal address 8013. At (b) are the four digits of hexadecimal address lamp display, addressed at locations 8014 and 8015. These displays incorporate latching logic as well as the needed decoding of 4 bit hexadecimal patterns into an array of LED dots. And at (c) are two miscellaneous indicators for the front panel.



Text continued from page 112

defines an address value which is presented to the 7489 scratch pad memories IC61 and IC63 by 74157 multipelexer IC62 when the scratch pads are not being referenced by the processor. Since the control panel program references the scratch pad only occasionally, the normal state is an address selected by S24 determining which of the five scratch pad locations is seen in the display lamps for scratch pad data. Thus the scratch pad memory has several potentially visible bytes of memory address space and acts as an output device.

A second output device is an 8 bit data latch whose outputs activate eight discrete LED devices. This device, located at address 8013 in memory address space, is used to display the processor's status register bits. The front panel control program is responsible for maintaining current information in this display (as is the case with all the display outputs).

Address display information is latched into four digits of hexadecimal display provided by Hewlett-Packard HP5082-7340 parts, IC65, IC66, IC67 and IC68 in figure 1.8b. Each of these displays has a built-in 4 bit data latch which retains information defined by writing to the memory locations 8014 (low order) and 8015 (high order).

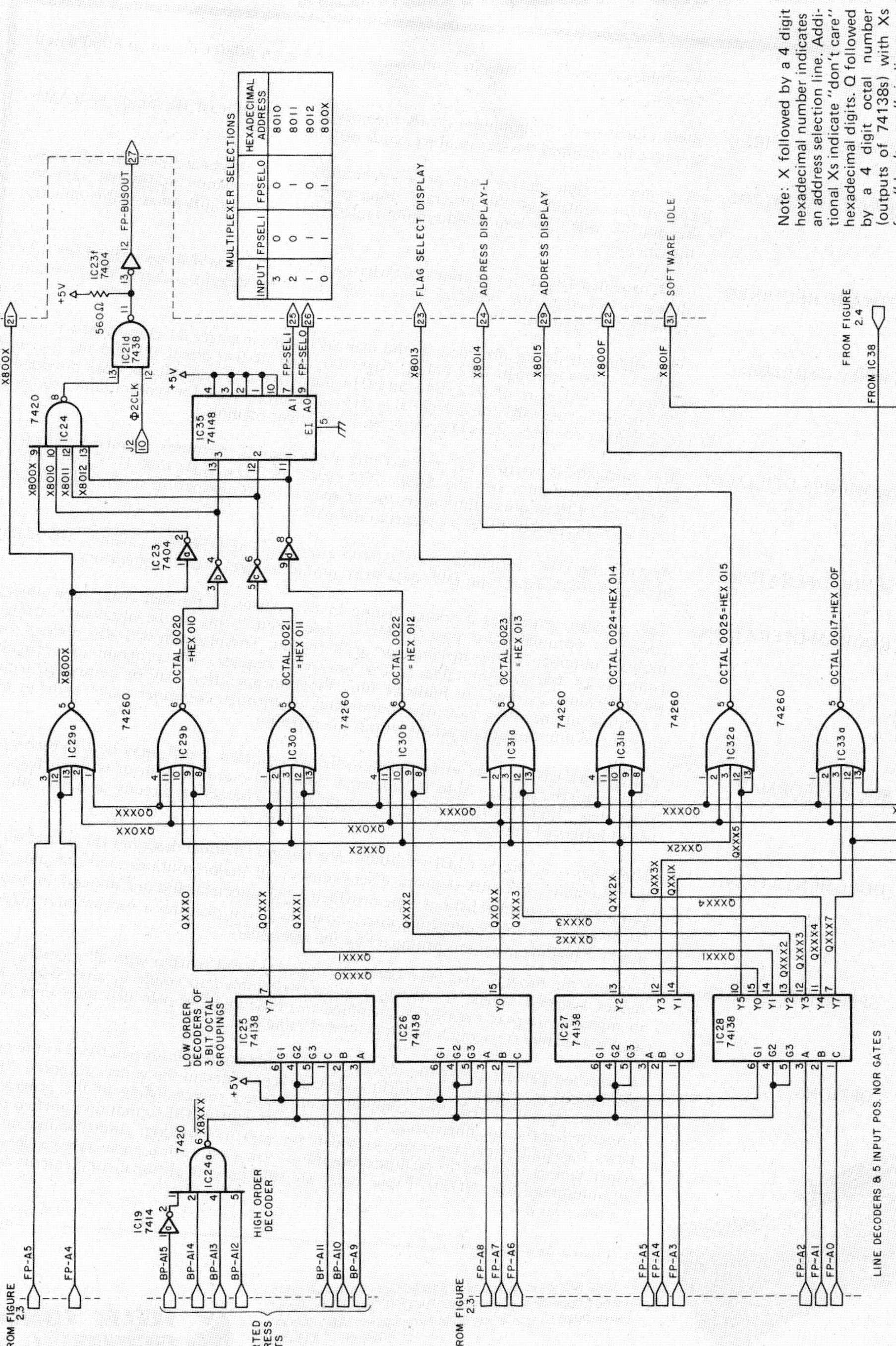
Two miscellaneous indicators are also included in the design. These single bit LED displays are connected to the processor's ready (RDY) and main interrupt (IRQ) lines. The RUN indicator lights

Text continued on page 119

Figure 1.9: Front panel mechanical layout. This is a detail drawing to scale of the physical layout of the front panel as seen in the photographs, along with the front panel electronics board which is mounted on standoffs behind the panel.

NOTES:

1. All ICs except locations E1 and E2 are Texas Instruments SN74XXX series TTL logic. E1 and E2 are three-state bidirectional bus drivers made by National.
2. Switches S1 thru S6 are Alco model MTF-206SA.
3. Switches S8 thru S23 are Alco model MTF-106D.
4. Switch S7 is a Centralab model PA-2009 rotary.
5. Switch S24 is a Centralab model PA-2011 rotary.
6. RD, ID, DD0 to DD7 and SD0 to SD7 are discrete LED displays. HP model 5082-4860.
7. X1-X4 are dot matrix hexadecimal LED displays. HP model 5082-7340.
8. All pull up resistors shown on schematics in conjunction with front panel switches are mounted on those switches.



MULTIPLEXER SELECTIONS

INPUT	FPSEL1	FPSEL0	HEXADECIMAL ADDRESS
3	0	0	8010
2	0	1	8011
1	1	0	8012
0	1	1	800X

Note: X followed by a 4 digit hexadecimal number indicates an address selection line. Additional Xs indicate "don't care" hexadecimal digits. Q followed by a 4 digit octal number (outputs of 74138s) with Xs for "don't care" indicates a low order 12 bit address decode.

Text continued from page 116

up when the machine is in the run mode, and the interrupt indicator remains lit while an interrupt is pending. Interrupt control logic is contained in the devices requesting an interrupt, with the processor's priority encoder defining the backplane signal IRQX (backplane pin 12) which indicates that some interrupt is pending (and also signals the processor through its IRQ input, pin 4.) Thus when interrupt driven IO is used, the interrupt indicator lamp will flicker if appreciable interrupt processing wait states occur as various devices request attention.

Other Front Panel Functions

The front panel logic includes logic of the ready (RDY) and nonmaskable interrupt (NMI) timing, shown in detail in figure 1.5. These lines are used to generate signals which affect the processor in a manner very similar to interrupts. The HALT and single STEP switch are used to generate signals for these lines. This timing logic causes the 6502 processor and its control program to implement fairly conventional single stepping and program halt or restart functions. HALT or STEP switches are used to cause the processor to complete the present instruction, then execute one more instruction. Any other switch activated on the front panel causes this logic to allow the processor to complete only its present instruction. The hardware protocol of this logic locks out all front panel functions when the processor is running, except for the HALT switch.

The front panel's interfaces to human fingers are through various function switches. These switches are debounced using set-reset flip flops which come four to a package in the 74279 part. The debouncing logic guarantees that only one pulse is received for each activation of a switch.

Getting Kompuutar Into Operation

The operation of the front panel's control logic with respect to the actual processor

Figure 2.2: Buffering of processor control signals at the front panel interface module.

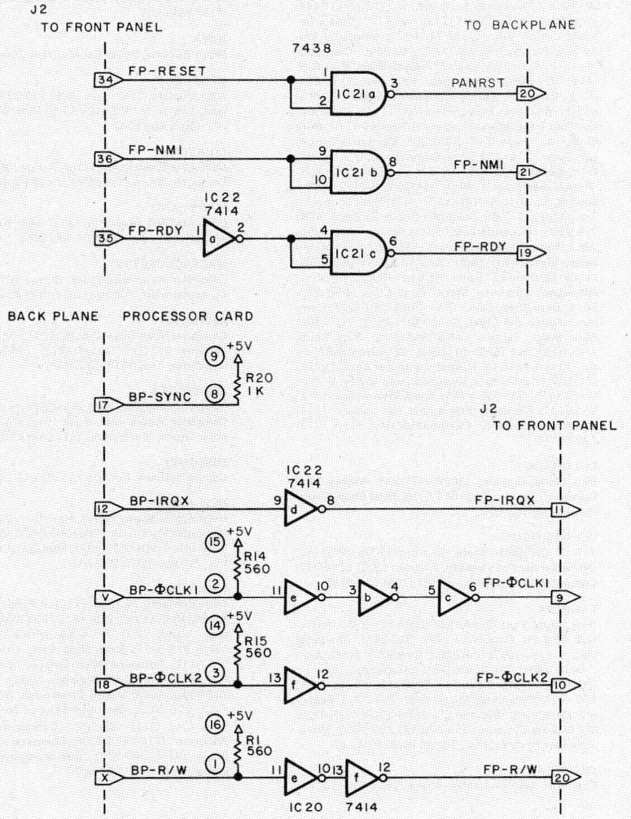


Figure 2.1: Front panel interface module address decode logic. This logic decodes the several addresses in the 8000 to 801F range which are used by the front panel design of Kompuutar. Since 3 bit decoders are used, octal intermediate terms are used to symbolize the outputs of the 74138s prior to logical sums performed by the 74260 OR gates. Outputs of the circuit are discrete select lines for several addresses, plus two source selection lines for the data bus input multiplexer of figure 1.6.

can be illustrated by walking verbally through a typical sequence of operations. First, let's assume that the machine is in RUN mode, which is indicated by a low level on the output of the execution state flip flop, IC49b pin 9. This is the normal situation for a fully executing 6502 program contained in the system's main programmable memory region. Next, press the front panel's HALT switch, S1. Upon release of the HALT switch the debounce logic completes one HALT pulse which is processed by the command encoding logic of figure 1.5. When the HALT line makes

Text continued on page 128

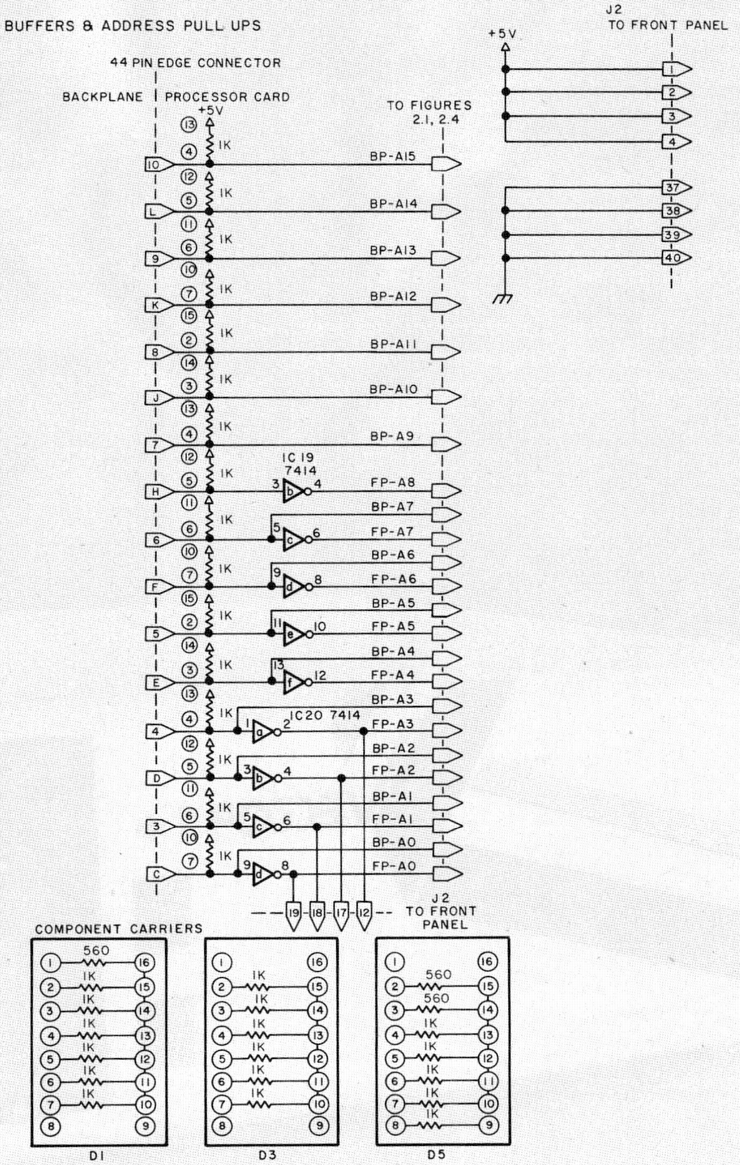


Figure 2.3: Pull up resistors for backplane address lines, and inverting receivers for local use in the front panel interface.

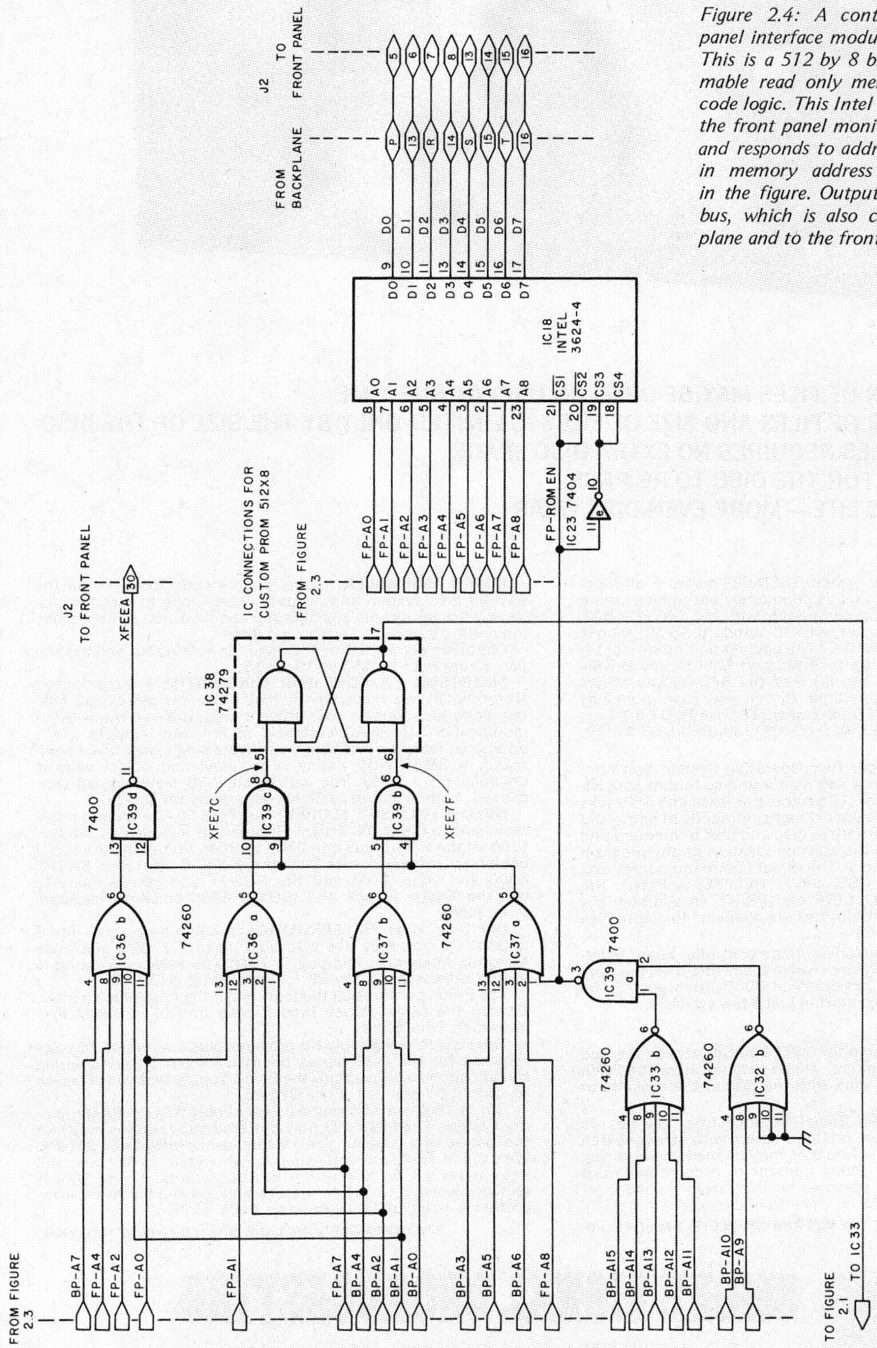


Figure 2.4: A continuation of the front panel interface module address decode logic. This is a 512 by 8 bit fusible link programmable read only memory module with decode logic. This Intel 3624-4 PROM contains the front panel monitor program of listing 1 and responds to addresses FE00 thru FFFF in memory address space using decoding in the figure. Output is directly to the data bus, which is also connected to the backplane and to the front panel assembly.

Figure 2.5: Mechanical layout of the front panel interface module. This board is built on a standard Vector Electronic Co prototyping card, and plugs into the 44 pin backplane connector (two sides with 22 pins each).

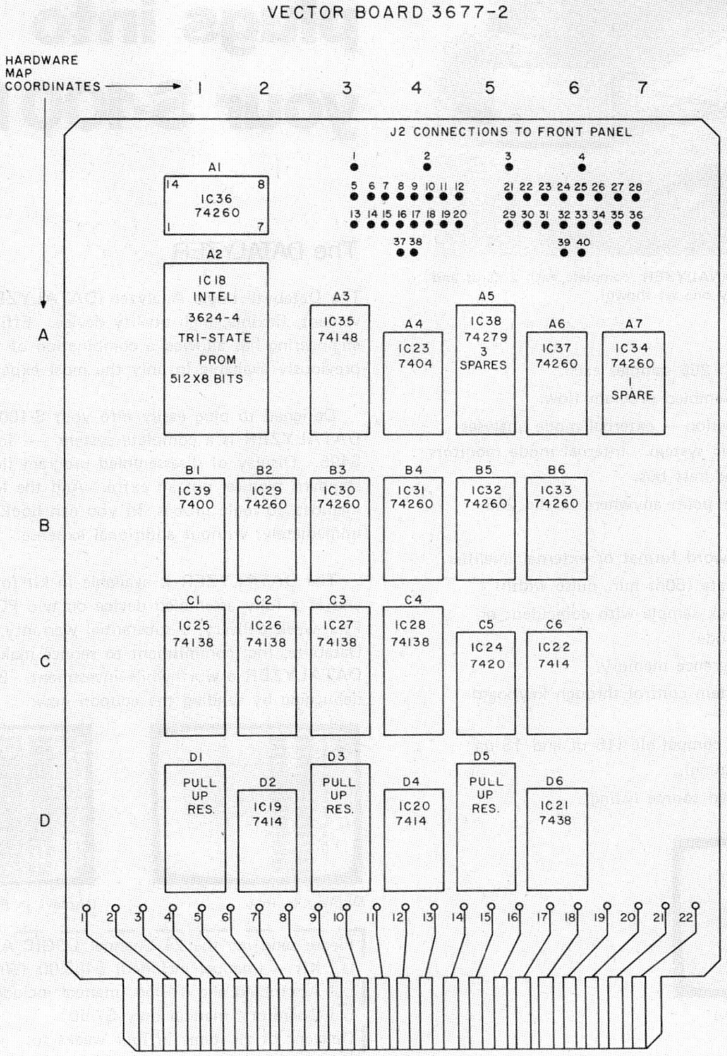
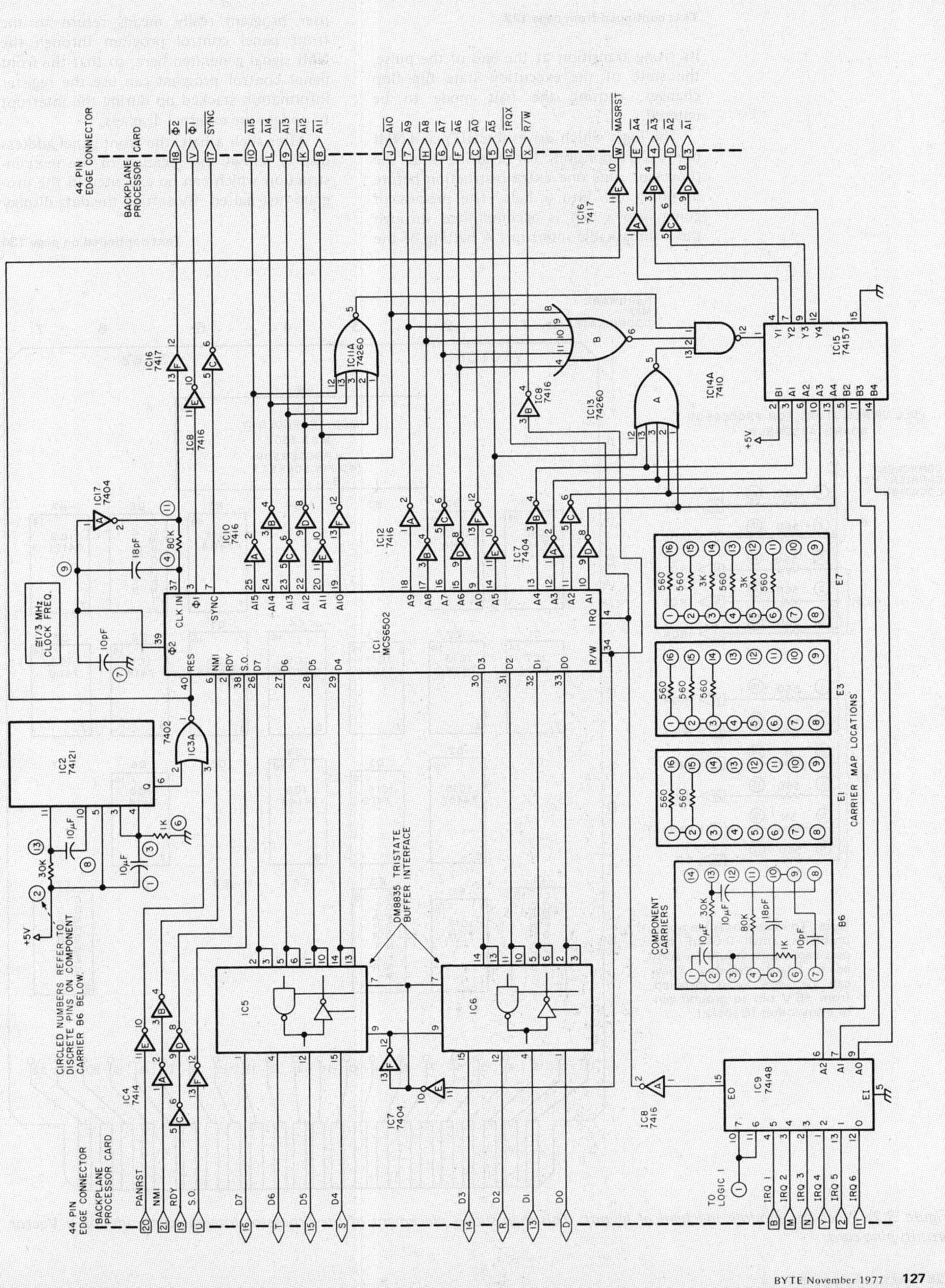


Figure 3.1: Processor module diagram. The 6502 processor module contains the 6502, its data bus and address bus buffering logic, and logic of the priority interrupt structure which substitutes one of six interrupt vector addresses on the low order address lines based on the priority of an interrupt. This overrides the processor generated address of FFFE or FFFF for an interrupt via the IRQ input. IRQ interrupt vectors are located at addresses FFE4 to FFEE in memory address space, with six priority levels. Note that the BRK instruction maps to vector FFE0 (see listing 1).


44 PIN EDGE CONNECTOR
BACKPLANE PROCESSOR CARD
IC1 MCS6502
CLK IN
Φ1
Φ2
SYNC
RES
NMI
RDY
S.O.
IRQ
R/W
≅1/3 MHz CLOCK FREQ.
IC2 74121
IC3A 7402
IC4 7414
IC5
IC6
DM8835 TRISTATE BUFFER INTERFACE
IC7 7404
IC8 7416
IC9 74148
IC10 7416
IC11A 74260
IC12 7416
IC13 74260
IC14A 7410
IC15 74157
IC16 7417
IC17 7404
CIRCLED NUMBERS REFER TO DISCRETE PINS ON COMPONENT CARRIER B6 BELOW.
+5V
30K
10µF
1K
80K
18pF
10pF
560
3K
PANRST
NMI
RDY
S.O.
D7
D6
D5
D4
D3
D2
D1
D0
TO LOGIC 1
IRQ 1
IRQ 2
IRQ 3
IRQ 4
IRQ 5
IRQ 6
Φ2
Φ1
SYNC
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A0
A5
IRQX
R/W
MASRST
A4
A3
A2
A1
COMPONENT CARRIERS
B6
E1
E3
E7
CARRIER MAP LOCATIONS

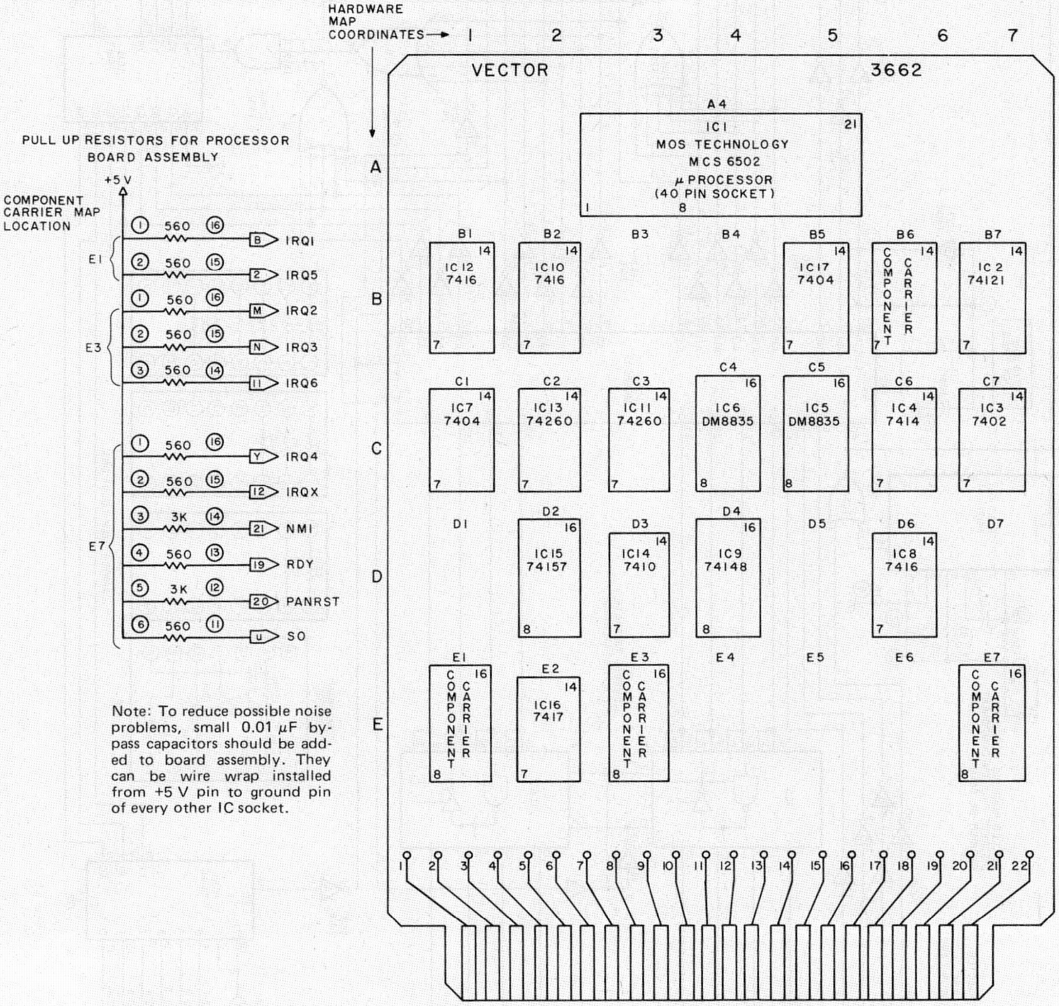
Text continued from page 122

its rising transition at the end of the pulse, the state of the execution state flip flop changes, causing the halt mode to be entered.

The logic which drives the RDY and NMI lines is responsible for assuring that the processor runs one extra instruction before dropping off into a halt. The process of going into a halt is accomplished through the nonmaskable interrupt. A halting of the user program really means return to the front panel control program through the NMI signal generated here, so that the front panel control program can use the register information stacked up during the interrupt to update the external displays.

After such a halt, the front panel address display shows the location of the next instruction which can be executed in the program just halted. By setting the data display

Text continued on page 134



Note: To reduce possible noise problems, small 0.01 μF by-pass capacitors should be added to board assembly. They can be wire wrap installed from +5 V pin to ground pin of every other IC socket.

Figure 3.2: Processor module mechanical layout. This map shows placement of the processor integrated circuits on a Vector prototyping card.

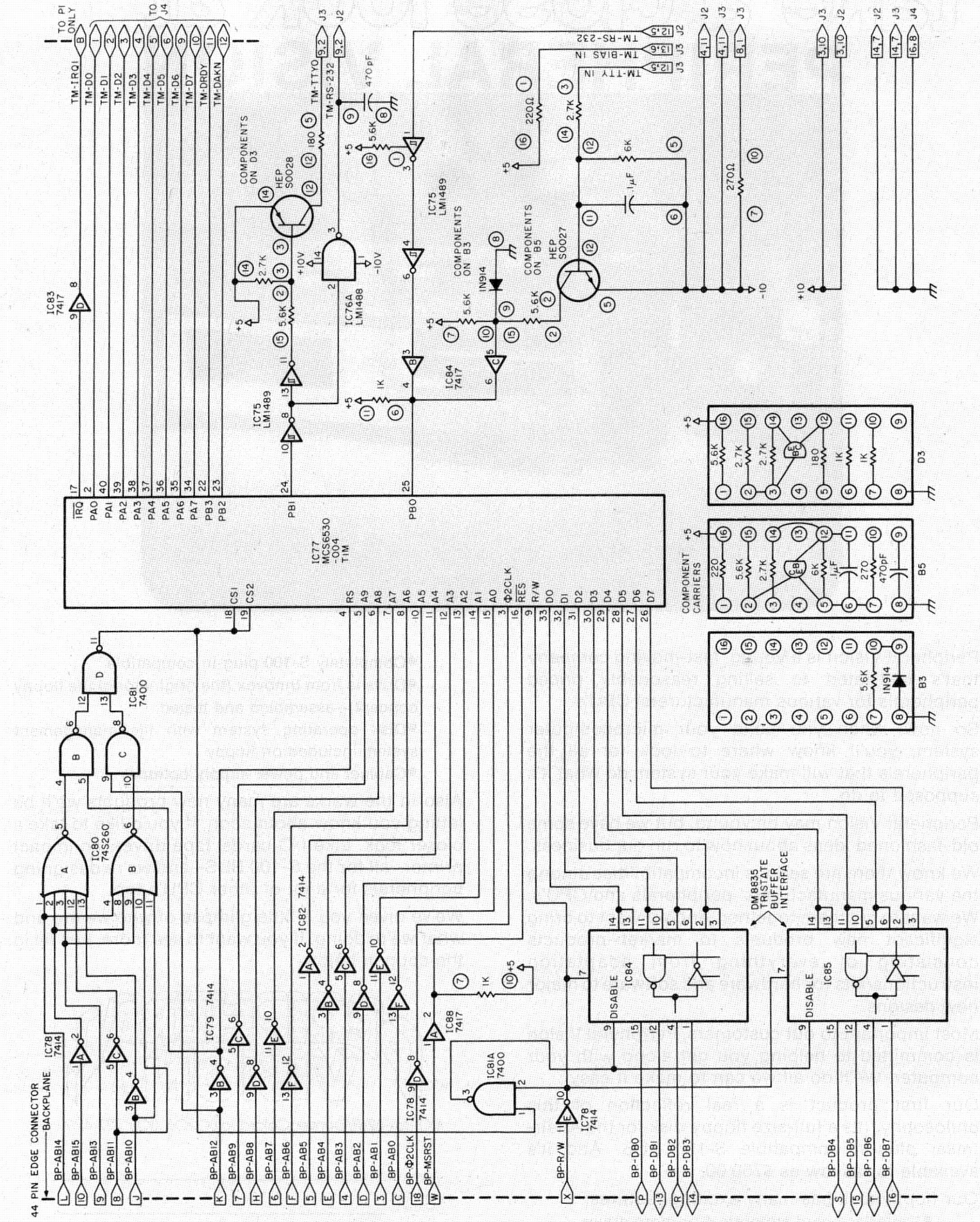
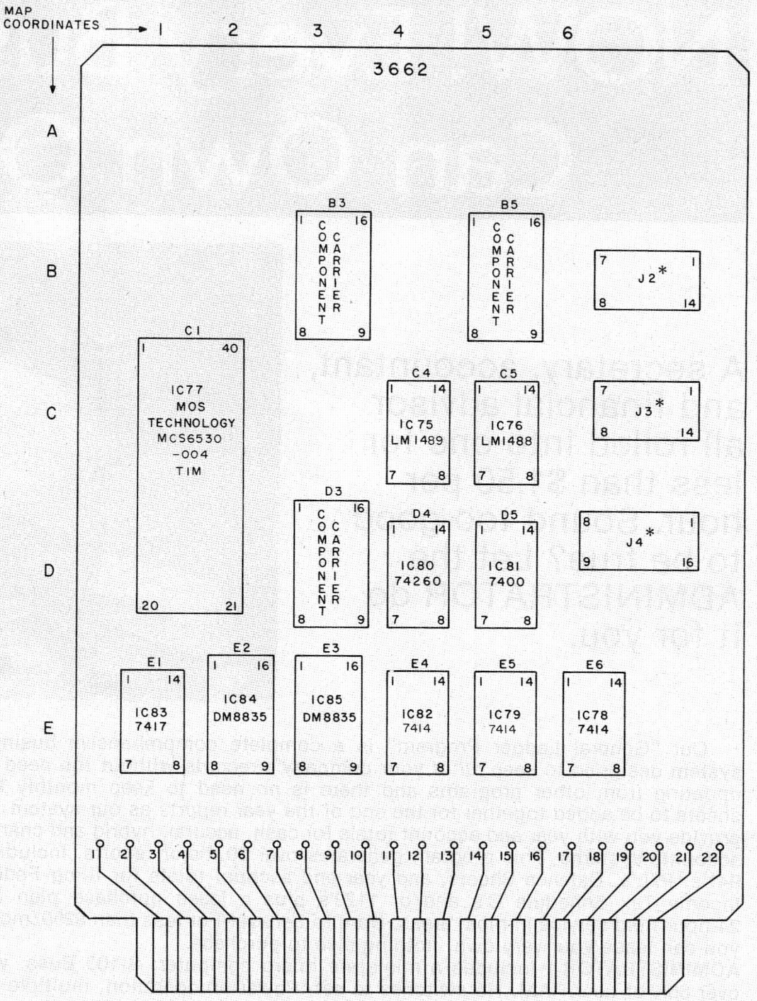


Figure 4.1: TIM interface module details. The MOS Technology TIM monitor program resides in a single MCS6530 ROM and peripheral interface circuit. The TIM interface module allows Kompuutar to be used with any serial terminal.



J4: Parallel Interface

Pin	Mnemonic	Description
1	D0	data bus lines
2	D1	
3	D2	
4	D3	
5	D4	
6	D5	
7	—	—
8	GND	ground
9	D6	data bus lines
10	D7	
11	DRDY	data ready
12	DAKN	data acknowledge
13	—	—
14	—	—
15	—	—
16	GND	ground

*Jacks 2, 3 and 4 are standard IC wire wrap sockets. They are used as cable connectors by mating with special "Augat" plugs.

J2: RS-232 Interface

Pin	Mnemonic	Description
1	—	—
2	RS-232 OUT	seal data standard (out)
3	+10 V	voltage source to peripherals
4	−10 V	voltage source to peripherals
5	RS-232 IN	seal data standard (in)
6	—	—
7	GND	ground
8	—	—
9	RS-232 OUT	seal data standard (out)
10	+10 V	voltage source to peripherals
11	−10 V	voltage source to peripherals
12	RS-232 IN	seal data standard (in)
13	—	—
14	GND	ground

J3: Teletype Interface

Pin	Mnemonic	Description
1	−10 V	voltage source to peripherals
2	TTYOT	Teletype (out)
3	+10 V	voltage source to peripherals
4	+10 V	voltage source to peripherals
5	TTYIN	Teletype (in)
6	BIASIN	pull up voltage source
7	GND	ground
8	−10 V	voltage source to peripherals
9	TTYOT	Teletype (out)
10	+10 V	voltage source to peripherals
11	−10 V	voltage source to peripherals
12	TTYIN	Teletype (in)
13	BIASIN	pull up voltage source
14	GND	ground

Figure 4.2: TIM interface module mechanical layout. This shows the physical arrangement of the wire sockets used to implement the TIM terminal interface for Kompuutar.

Text continued from page 128

selection switch to point to the DATA position, the contents of memory at this location are displayed, having been transferred to the front panel scratch pad location 800D by the service program during the halt operation. At other positions of the switch, it is possible to view copies of the X register, Y register, accumulator A or the stack pointer register. The current contents of the processor status register are shown in the eight discrete LED outputs at location 8013 in memory address space.

If it is desired to modify a status flag, the flag select switch can be rotated to the desired bit, and the flag set or flag reset function be used to alter the flag state. If desired, memory can be examined or altered using the deposit, deposit next, examine or examine next routines activated by appropriate panel switches. Pressing RUN con-

Text continued on page 137

Central Processor Module Integrated Circuits

IC	Type	+5 V	GND	Map Location in Figure 3.2
1	MSC6502	8	1, 21	A3, 4, 5
2	74121	14	7	B7
3	7402	14	7	C7
4	7414	14	7	C6
5	DM8835	16	8	C5
6	DM8835	16	8	C4
7	7404	14	7	C1
8	7416	14	7	D6
9	74148	16	8	D4
10	7416	14	7	B2
11	74260	14	7	C3
12	7416	14	7	B1
13	74260	14	7	C2
14	7410	14	7	D3
15	74157	16	8	D2
16	7417	14	7	E2
17	7404	14	7	B5

Front Panel Interface Module Circuits

IC	Type	+5 V	GND	Map Location in Figure 2.3
18	Intel3624-4	22, 24	12	A2
19	7414	14	7	D2
20	7414	14	7	D4
21	7438	14	7	D6
22	7414	14	7	C6
23	7404	14	7	A4
24	7420	14	7	C5
25	74138	16	8	C1
26	74138	16	8	C2
27	74138	16	8	C3
28	74138	16	8	C4
29	74260	14	7	B2
30	74260	14	7	B3
31	74260	14	7	B4
32	74260	14	7	B5
33	74260	14	7	B6
34	74260	14	7	A7
35	74148	16	8	A3
36	74260	14	7	A1
37	74260	14	7	A6
38	74279	16	8	A5
39	7400	14	7	B1

Front Panel Assembly Circuits

IC	Type	+5 V	GND	Map Location in Figure 1.9
40	74279	16	8	D5
41	74279	16	8	E5
42	74279	16	8	E6
43	74279	16	8	F8
44	7400	14	7	E3
45	74197	14	7	D1
46	7400	14	7	F9
47	7400	14	7	F1
48	7404	14	7	C1
49	7474	14	7	F4
50	74148	16	8	F7
51	7474	14	7	F5
52	7400	14	7	F6
53	7410	14	7	F3
54	7400	14	7	C5
55	7474	14	7	F2
56	74197	14	7	A4
57	7416	14	7	B5
58	7400	14	7	C4
59	74197	14	7	B4
60	7416	14	7	B1
61	7489	16	8	A3
62	74157	16	8	C3
63	7489	16	8	B3
64	7416	14	7	A1
65	HP-5082-7340	7	6	X1
66	HP-5082-7340	7	6	X2
67	HP-5082-7340	7	6	X3
68	HP-5082-7340	7	6	X4
69	74153	16	8	A2
70	74153	16	8	B2
71	74153	16	8	C2
72	74153	16	8	E4
73	DM8835	16	8	E2
74	DM8835	16	8	E1

TIM Interface Module Circuits

IC	Type	+5 V	+10 V	−10 V	GND	Map Location in Figure 4.2
75	LM1489	14	—	—	7	C4
76	LM1488	—	14	1	7	C5
77	MCS6530	20	—	—	1	C1
78	7414	14	—	—	7	E6
79	7404	14	—	—	7	E5
80	74S260	14	—	—	7	D4
81	7400	14	—	—	7	D5
82	7404	14	—	—	7	E4
83	7417	14	—	—	7	E1
84	DM8835	16	—	—	8	E2
85	DM8835	16	—	—	8	E3

Table 5: Integrated circuit summary for Kompuutar. This table summarizes the 85 integrated circuits used in Kompuutar, arranged in groupings by the circuit modules of the system. The column labelled "Map Location" identifies the physical position of the circuits on the various boards of the system, as shown in the physical layout diagrams in the figures. Note that the physical layouts represent a good workable arrangement of sockets. There is no logical requirement that the particular map positions used in these diagrams be followed to the letter in another implementation of the system.

Listing 1: The front panel service program. This program is resident in a programmable read only memory part wired for addresses FE00 to FFFF, as shown in figure 2.4. The listing is a symbolic assembly language version of the program combined with the hand assembled object code provided by author Brader. The information at the end of the PROM area includes the interrupt vectors which are implemented in the Kompuutar system.

```
Hexadecimal  Hexadecimal
  Address       Code      Label     Op   Operand    Commentary
                          * PANEL DRIVER NONMASKABLE INTERRUPT
                            ENTRY
FE00         8D  00  80   PNMI      STA  SVACC      [Save state
FE03         8E  01  80             STX  SVX         of processor registers
FE06         8C  02  80             STY  SVY         at interrupt] ;
FE09                      * RECOVER PROCESSOR STATUS & RETURN
                            ADDRESS AT INTERRUPT
FE09         68           UNSTK     PLA
FE0A         AA                     TAX
FE0B         68                     PLA
FE0C         A8                     TAY
FE0D         68                     PLA
FE0E                      * SAVE STATUS, RETURN ADDRESS & STACK IN
                            SCRATCH PAD
FE0E         8E  03  80   SAVERET   STX  SVPFLG
FE11         8C  07  80             STY  SVADDR
FE14         8D  08  80             STA  SVADDR+1
FE17         BA                     TSX
FE18         8E  05  80             STX  SVSTK
FE1B                      * INTERROGATE CONTROL FUNCTION REQUEST
                            REGISTER
FE1B         AD  10  80             LDA  REQST
FE1E         0A                     ASL
FE1F         0A                     ASL
FE20         0A                     ASL
FE21         0A                     ASL
                          * ENTER PROGRAM TREE TO DECODE & EXECUTE
                            REQUEST
FE22         8D  04  80             STA  SVAAA
FE25         A9  20                 LDA  #$20
FE27         2C  04  80             BIT  SVAAA
FE2A         10  03                 BPL  *+5        If request greater than 7
FE2C         4C  7F  FF             JMP  REGLD       then go do register load
                                                     routine;
FE2F         70  1C                 BVS  XHSEE      Else if 4 < request < 7
                                                     then halt, step, examine
                                                     or exam next;
FE31         D0  0D                 BNE  XDDN       Else if 2 < request < 3
                                                     then deposit or deposit
                                                     next;
                          * CONCLUDE REQUEST IS 0 OR 1
FE33         A9  10                 LDA  #$10       [Set up mask for bit
                                                     test] ;
FE35         2C  04  80             BIT  SVAAA
FE38         D0  03                 BNE  *+5        Is request 0?
FE3A         4C  99  FE             JMP  EXITCP     If request = 0 then leave
                                                     control program via
                                                     normal NMI;
FE3D         4C  36  FF             JMP  FLAG       If request = 1 then go do
                                                     FLAG function service;
                          * CONCLUDE REQUEST IS 2 OR 3
FE40         A9  10       XDDN      LDA  #$10
FE42         2C  04  80             BIT  SVAAA
FE45         D0  03                 BNE  *+5        Is request 2?
FE47         4C  29  FF             JMP  DPSNXT     If request = 2 then go do
                                                     DEPOSIT NEXT
                                                     service;
FE4A         4C  1E  FF             JMP  DEPOSIT    If request = 3 then go do
                                                     DEPOSIT service;
FE4D                      * CONCLUDE REQUEST IS 4 TO 7
FE4D         D0  0D       XHSEE     BNE  XHLSS      If request = 6 or request =
                                                     7 then go do HALT or
                                                     STEP;
                          * CONCLUDE REQUEST IS 4 OR 5
FE4F         A9  10                 LDA  #$10
FE51         2C  04  80             BIT  SVAAA
FE54         D0  03                 BNE  *+5        Is request 4?
FE56         4C  FD  FE             JMP  EXMNXT     If request = 4 then go do
                                                     EXAMINE NEXT;
FE59         4C  EE  FE             JMP  EXAMINE    If request =   then go do
                                                     EXAMINE;
                          * CONCLUDE REQUEST IS 6 OR 7
FE5C         A9  10       XHLSS     LDA  #$10       [Set up mask for bit
                                                     test] ;
FE5E         2C  04  80             BIT  SVAAA      [Test for odd or even
                                                     number] ;
FE61         F0  14                 BNE  SETTRAP    If request = 7 then skip
                                                     setup logic;
                          * CONCLUDE REQUEST IS 6
FE63                      * SET UP SCRATCH PAD INSTRUCTION SEQUENCE
FE63         A9  8D                 LDA  #$8D
FE65         8D  09  80             STA  DUMMY2
FE68         A9  0D                 LDA  #$0D
FE6A         8D  0A  80             STA  DUMMY3
FE6D         A9  80                 LDA  #$80
FE6F         8D  0B  80             STA  DUMMY4
FE72         A9  60                 LDA  #$60
FE74         8D  0C  80             STA  DUMMY5
FE77                      * SET UP SCRATCH PAD INSTRUCTION SEQUENCE
FE77         A9  AD       SETTRAP   LDA  #$AD       [Dummy in an LDA
FE79         8D  06  80             STA  DATATRAP    instruction using cur-
                                                     rent address field as its
                                                     address] ;
FE7C                      * ROUTINE TO EXECUTE SCRATCH PAD PROGRAM
                            TO SET UP LATEST SVDATA VALUE
FE7C         20  06  80   SETUP     JSR  DATATRAP   [Call scratch pad resident
                                                     subroutine] ;
FE7F         AD  03  80             LDA  SVPLFG
FE82         8D  13  80             STA  STATUS
FE85         A9  0D                 LDA  #$0D       [Patch the scratch pad
FE87         8D  0A  80             STA  DUMMY3      program] ;
FE8A                      * DISPLAY THE CURRENT ADDRESS
FE8A         AD  07  80             LDA  SVADDR
FE8D         8D  14  80             STA  LADDR
FE90         AD  08  80             LDA  SVADDR+1
FE93         8D  15  80             STA  HADDR
FE96         4C  9D  FE             JMP  CONTINUE
FE99                      * CONTROL PANEL MONITOR EXIT POINT
FE99         38           EXITCP    SEC             [Set carry to indicate end
                                                     of monitor];
FE9A         4C  9E  FE             JMP  SETSTAK
FE9D         18           CONTINUE  CLC             [Clear carry to continue
                                                     monitor] ;
FE9E                      * BEGIN ROUTINE TO SET UP STACK AND
                            TEMPORARILY LEAVE MONITOR
FE9E         AD  08  80   SETSTAK   LDA  SVADDR+1
FEA1         48                     PHA
FEA2         AD  07  80             LDA  SVADDR
FEA5         48                     PHA
FEA6         AD  03  80             LDA  SVPFLG
FEA9         48                     PHA             [Push processor status
                                                     onto stack] ;
FEAA         AE  05  80             LDX  SVSTK      [Subtract 3
FEAD         CA                     DEX              from
FEAE         CA                     DEX              old
FEAF         CA                     DEX              stack pointer
FEB0         9A                     TXS              to compensate pushes] ;
                          * RESTORE REGISTERS
FEB1         AE  01  80             LDX  SVX
FEB4         AC  02  80             LDX  SVY
FEB7         AD  00  80             LDA  SVACC
FEBA         8D  0E  80             STA  LSTACC
FEBD         B0  3C                 BCS  LEAVE      If carry set, then leave
                                                     monitor now;
FEBF                      * MANIPULATIONS FOR SINGLE STEP CASE
                            FOLLOW
FEBF         A9  FF                 LDA  #$FF
FEC1         4D  00  80             EOR  SVACC      [Invert old accumulator
                                                     value] ;
FEC4         8D  00  80             STA  SVACC
FEC7         A9  FF                 LDA  #$FF
FEC9         4D  01  80             EOR  SVX        [Invert old X value] ;
FECC         8D  01  80             STA  SVX
FECF         A9  FF                 LDA  #$FF
FED1         4D  02  80             EOR  SVY        [Invert old Y value] ;
FED4         8D  02  80             STA  SVY
FED7         A9  FF                 LDA  #$FF
FED9         4D  05  80             EOR  SVSTK      [Invert old stack register
                                                     value] ;
FEDC         8D  05  80             STA  SVSTK
FEDF         A9  FF                 LDA  #$FF
FEE1         4D  0D  80             EOR  SVDATA     [Invert old data register
                                                     value] ;
FEE4         8D  0D  80             STA  SVDATA
FEE7         AD  0E  80             LDA  LSTACC     [Restore old accumulator
                                                     value] ;
FEEA         40                     RTI             [Return from interrupt,
                                                     exiting panel service
                                                     program] ;
FEEB                      * LEAVE PANEL SERVICE FOR USER NMI SERVICE
                            CASE
FEEB         4C  00  F6   LEAVE     JMP  USRNMI
FEEE                      * NOTE:  USRNMI IS A READ ONLY MEMORY
                            ROUTINE AT F600, NOT DEFINED HERE
FEEE                      *
FEEE                      *         EXAMINE SERVICE ROUTINE
FEEE         AD  11  80   EXAMINE   LDA  DSWLOW
FEF1         8D  07  80             STA  SVADDR
FEF4         AD  12  80             LDA  DSWHIGH
FEF7         8D  08  80             STA  SVADDR+1
FEFA         4C  77  FE             JMP  SETTRAP    [Go back to display
                                                     routine and exit] ;
FEFD                      *         EXAMINE NEXT SERVICE ROUTINE
FEFD         A0  00       EXMNXT    LDY  #0
                          * INCREMENT TO NEXT ADDRESS
FEFF         AE  07  80   ADVANCE   LDX  SVADDR
FF02         E8                     INX
FF03         8E  07  80             STX  SVADDR
FF06                      * CHECK FOR OVERFLOW
FF06         E0  00                 CPX  #0
FF08         F0  0A                 BEQ  RAISHI     If low order = 0 then
                                                     increment high order;
FF0A                      * CHECK TO SEE IF INITIALIZATION NEEDED
FF0A         C0  00       QNEEDINI  CPY  #0
FF0C         F0  03                 BEQ  *+5
FF0E         4C  7C  FE             JMP  SETUP      If Y = 0 then simply
                                                     execute scratch pad
                                                     program;
FF11         4C  77  FE             JMP  SETTRAP    If Y = 0 then set up prior
                                                     to executing again;
FF14                      * INCREMENT HIGH ORDER BYTE OF CURRENT
                            ADDRESS
FF14         AE  08  80   RAISHI    LDX  SVADDR+1
FF17         E8                     INX
FF18         8E  08  80             STX  SVADDR+1
FF1B         4C  0A  FF             JMP  QNEEDINI
FF1E                      *
FF1E                      * DEPOSIT SERVICE ROUTINE
FF1E         A9  8D       DEPOSIT   LDA  #$8D       [Change scratch pad pro-
FF20         8D  06  80             STA  DATATRAP    gram to store accumu-
                                                     lator (STA) as its first
                                                     operation] ;
FF23         AD  11  80             LDA  DSWLOW
FF26         4C  7C  FE             JMP  SETUP
FF29                       *
FF29                       * DEPOSIT NEXT SERVICE ROUTINE
FF29         A9  8D       DPSNXT    LDA  #$8D       [Change scratch pad pro-
FF2B         8D  06  80             STA  DATATRAP    gram to store accumu-
                                                     lator (STA) as its first
                                                     operation] ;
FF2E         AD  11  80             LDA  DSWLOW
FF31         A0  AA                 LDY  #$AA       [Load Y with any nonzero
                                                     value, AA in particular] ;
```

Text continued from page 134

tinues execution of the program at the currently displayed address.

After building the design, I found a couple of subtle points in the operation of the control panel. The first point to note is that when using the RESET switch, the processor must be in the RUN mode of the RUN versus STEP switch. Second, bit 5 of the status display is useless and unimplemented in the 6502 hardware. The register load switch (REG LOAD) is best used for modifying the contents of the accumulator, index registers, but not the stack pointer. Using the register load switch to modify the stack pointer can result in problems when resuming execution of a program. Once whatever memory loading or register alteration chores required by a program have been accomplished, execution can be resumed using the RUN switch to cause the control panel program to return from interrupt using the register contents stored in the control panel scratch area.

A Versatile Configuration

The design of Kompuutar is quite readily adaptable to the 6800 processor as a substitute for the 6502 if personal programming preferences or availability of chips dictates such a switch. The similarities between the two processors are quite extensive, and in fact were the bone of contention of a lawsuit (since settled) shortly after the 6502 came out. At the system level, here are the major differences to be aware of:

- *The pinouts of the 40 pin package used for each processor are different, but the signal definitions of NMI, data bus lines, IRQ, reset, address bus, etc, are equivalent.*
- *The 6800 uses four read only memory interrupt vectors at addresses FFF8 to FFFF in memory address space, whereas the 6502 uses only three interrupt vectors; the definitions of the interrupt vectors for reset, nonmaskable interrupt and maskable interrupts are similar.*
- *The instruction sets differ, so the front panel service programs shown with this article would need to be recoded if a 6800 is used.*
- *The definition of the clock used by the processor differs in the details of its drive circuit.*

The major features of either a 6502 or 6800 system at the level of the backplane bus defined here would be nearly identical. ■

Listing 1, continued:

```
Hexadecimal  Hexadecimal
  Address       Code       Label     Op    Operand    Commentary

FF33         4C  FF  FE              JMP   ADVANCE
FF36                      *
FF36                      * FLAG REGISTER CHANGE SERVICE ROUTINE
FF36         AD  10  80  FLAG        LDA   REQST
FF39                      * ALIGN THE FLAG SELECTION BITS WITH FOUR
                            RIGHT SHIFTS
FF39         4A                      LSR
FF3A         4A                      LSR
FF3B         4A                      LSR
FF3C         4A                      LSR
FF3D         8D  04  80              STA   SVAAA
FF40         A9  01                  LDA   #$1
FF42         2C  04  80              BIT   SVAAA      Is flag data bit on?
FF45         F0  05                  BEQ   SETYZER    If flag data = 0 then
                                                        Y:= 0;
FF47         A0  FF                  LDY   #$FF       Else Y:= $FF;
FF49         4C  4E  FF              JMP   PROCFLAG
FF4C         A0  00      SETYZER     LDY   #$00
FF4E         4E  04  80  PROCFLAG    LSR   SVAAA      [Final right shift aligns
                                                        the 3 bit flag code] ;
FF51         A2  07                  LDX   #$07       [Load loop count] ;
                          * LOOP SETS X
FF53         EC  04  80  FLGLOOP     CPX   SVAAA      If SVAAA = X then
FF56         F0  04                  BEQ   FCHANGE      go change this flag;
FF58         CA          CYCLE       DEX
FF59         4C  53  FF              JMP   FLGLOOP
FF5C         A9  01       FCHANGE    LDA   #$01
                          * LOOP ALIGNS MASK ON PROPER FLAG BIT
FF5E         E0  00       FLOOK      CPX   #$00
FF60         F0  05                  BEQ   *+7
FF62         0A                      ASL
FF63         CA                      DEX
FF64         4C  5E  FF              JMP   FLOOK
FF67                      * AT END OF LOOP WITH MASK IN PLACE, CHANGE
                            FLAG BIT
FF67         C0  00                  CPY   #$0        Is data 0?
FF69         D0  0B                  BNE   SETFLG     If not then go set flag;
FF6B         49  FF                  EOR   #$FF
FF6D         2D  03  80              AND   SVPFLG     [Turn off the flag bit
                                                        with inverted mask] ;
FF70         8D  03  80              STA   SVPFLG
FF73         4C  77  FE              JMP   SETTRAP
FF76         0D  03  80   SETFLG     ORA   SVPFLG     [Turn on the selected bit
                                                        from mask] ;
FF79         8D  03  80              STA   SVPFLG
FF7C         4C  77  FE              JMP   SETTRAP
FF7F                      * REGISTER LOAD SERVICE ROUTINE
FF7F                      * REGISTER LOAD SERVICE ROUTINE
FF7F         A9  0F       REGLD      LDA   #$0F
FF81         8D  0A  80              STA   DUMMY3
FF84         A9  AE                  LDA   #$AE       [Define LDX as first
                                                        operation of scratch pad
                                                        program] ;
FF86         8D  06  80              STA   DATATRAP
FF89         A9  FF                  LDA   #$FF
FF8B         4D  0D  80              EOR   SVDATA     [Invert data] ;
FF8E         8D  0D  80              STA   SVDATA
FF91         AD  11  80              LDA   DSWLOW
FF94         4C  7C  FE              JMP   SETUP
— — — — — — — — — — — unused space — — — — — — — — — — —
FFA0         A2  FF       RESET      LDX   #$FF       [Initialize stack
FFA2         9A                      TXS               pointer] ;
FFA3         4C  06  70              JMP   TIM        [Jump to TIM monitor
                                                        on system RESET] ;
— — — — — — — — — — — unused space — — — — — — — — — — —
FFE0         00  F0       BRK Instruction vector
FFE2         XX  XX       Not used
FFE4         00  FC       IRQ6 Lowest priority
FFE6         00  FB       IRQ5
FFE8         00  FA       IRQ4
FFEA         00  F9       IRQ3 Interrupt vectors
FFEC         00  F8       IRQ2
FFEE         00  F7       IRQ1 Highest priority

— — — — — — unused space — — — — — — — — — — — — — —
FFFA         00  FE       NMI interrupt vector location
FFFC         A0  FF       RESET interrupt vector location
FFFE         xx  xx
```

Kompuutar Peripheral and Scratch Pad Data Symbols:

Address	Symbol	Description
8000	SVACC	Saved accumulator value, scratch pad
8001	SVX	Saved X index register value, scratch pad
8002	SVY	Saved Y index register value, scratch pad
8003	SVPFLG	Saved processor flag register value, scratch pad
8004	SVAAA	Scratch value
8005	SVSTK	Saved stack register value
8006	DATATRAP	First operation of scratch pad program (absolute addressing through SVADDR)
8007	SVADDR	Low order saved address value, scratch pad program absolute address for DATATRAP
8008	SVADDR+1	High order saved address value
8009	DUMMY2	Balance of scratch pad program
800A	DUMMY3	
800B	DUMMY4	
800C	DUMMY5	
800D	SVDATA	Data value at current SVADDR
800E	LSTACC	Accumulator storage temporary
800F	SVAAA	Scratch value
8010	REQST	Front panel request input word
8011	DSWLOW	Low order data switch register input (data or address information)
8012	DSWHIGH	High order data switch register input (address information only)
8013	STATUS	Status lamps output
8014	LADDR	Low order address display output
8015	HADDR	High order address display output