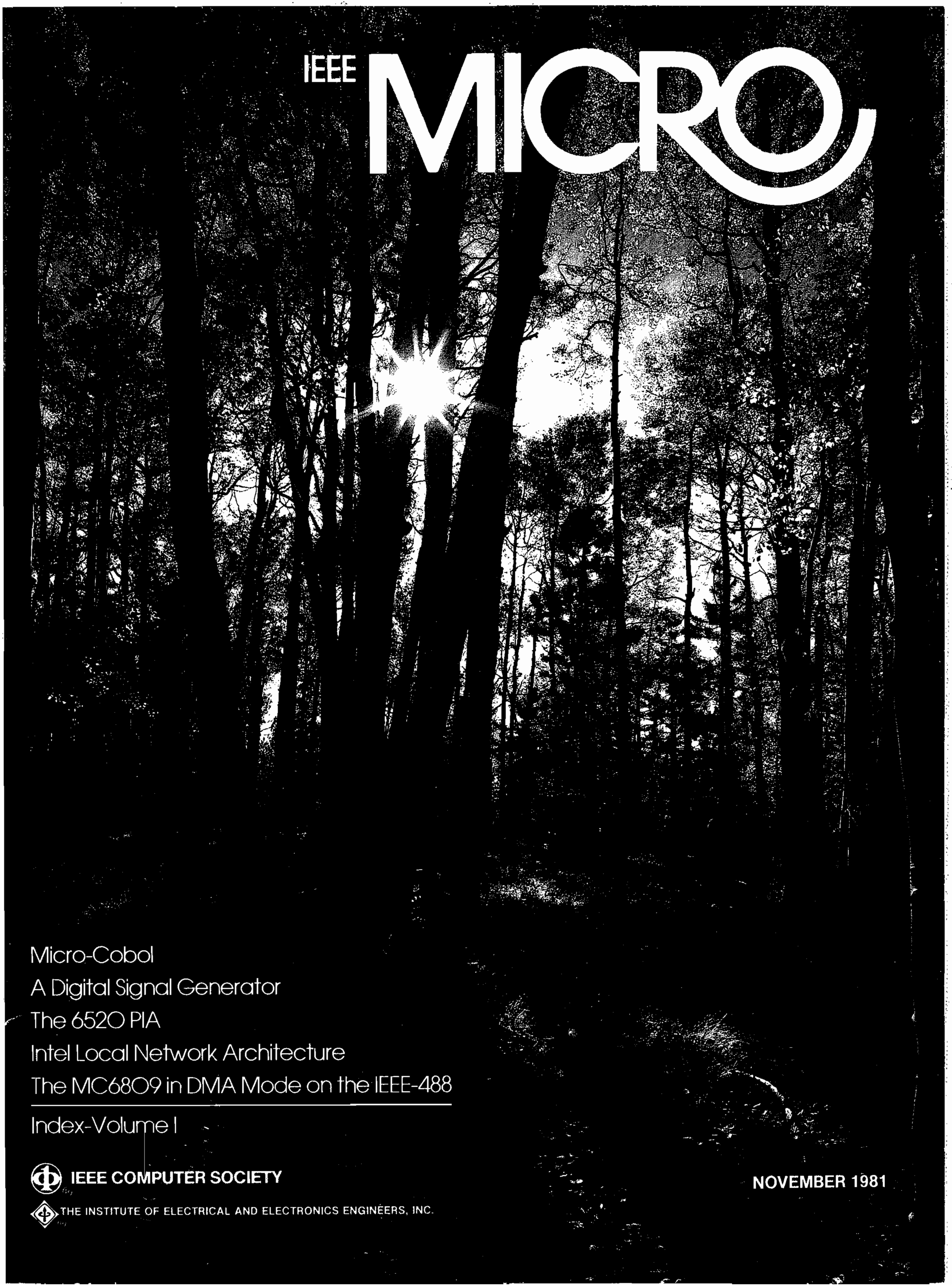
IEEE MICRO



Micro-Cobol

A Digital Signal Generator

The 6520 PIA

Intel Local Network Architecture

The MC6809 in DMA Mode on the IEEE-488

Index-Volume I

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

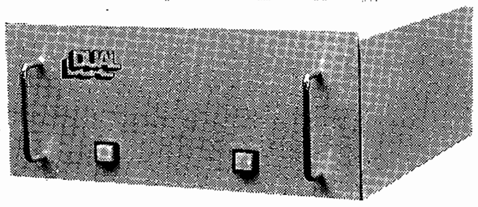
NOVEMBER 1981

POWER & RELIABILITY

68000 μP on the S-100 Bus?

YES, AVAILABLE NOW FROM DUAL SYSTEMS!

- ☐ **8 mHz 68000 microprocessor.**
- ☐ **16-megabyte direct addressing.**
- ☐ **32-bit internal arithmetic.**
- ☐ **Minicomputer type instructions** including MULTIPLY.
- ☐ **FULL IEEE-696 S-100 compliance.** Runs with all 4 mHz S-100 boards and automatically runs faster when accessing Dual Systems memory boards, for FULL SPEED OPERATION OF THE 68000.
- ☐ **Powerful vectored interrupts.** 7 Vectored interrupts, including NMI, as well as alternate mode having up to 256 interrupts.
- ☐ **On board monitor ROM** for immediate use.
- ☐ **Connector for future addition of memory management** unit for multi-user operating systems.
- ☐ **Built to the highest industrial standards** with 200 hour burn-in.



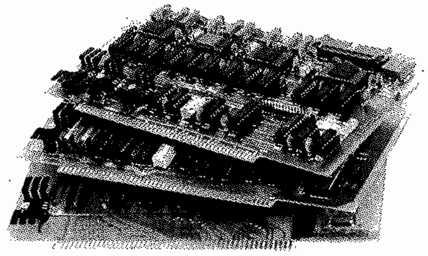
CPU/68000 CPU board..... $1195

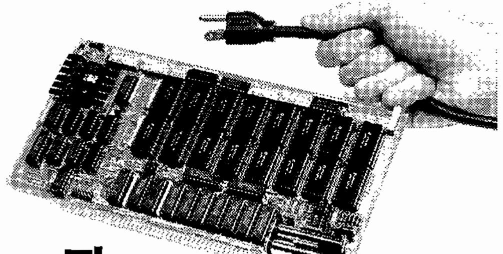
32K-byte 8/16-bit NONVOLATILE RAM board, for secure storage of programs you are developing. Allows FULL SPEED CPU operation. CMEM-32K, per 32K-bytes .. $895

32K-byte 8/16-bit EPROM board, EPROM-32K $245

Serial I/O board, SIO-2 ...... $285

All of the above with cabinet, power supply and backplane..... $3685





The Ultimate IEEE S-100 Memory Would...

- ☐ **BE NONVOLATILE** holding data for up to eight years with the power off.
- ☐ **RUN AT 6 MHZ** without wait states.
- ☐ **RUN IN 8 OR 16-BIT** systems with 8 or 16-bit wide data paths.
- ☐ **HAVE EXTENDED 24-BIT ADDRESSING** and bank select.
- ☐ **HAVE DYNAMICALLY MOVABLE WRITE PROTECT AREAS** to prevent accidental erasure of programs and critical data.
- ☐ **GENERATE POWER-FAIL** interrupts for orderly system shutdown & power failure recovery.

...Available Now from Dual Systems

The Dual Systems CMEM memory boards combine high-speed CMOS memories with new 5-10 year lithium batteries to give you the nonvolatility of an EPROM board while retaining the instant writability of a high-speed read/write RAM. These industrial grade boards are ruggedly built and are burned-in for 200 hours.

- ☐ CMEM-32K, 32K-bytes ... $895
- ☐ CMEM-16K, 16K-bytes.... $795
- ☐ CMEM-8K, 8K-bytes...... $695

Sales representatives in most metropolitan areas.

OEM and Dealer pricing is available.

Toughest Boards in Town... IEEE 696/S-100

SUPER RELIABLE NON-STOP CLOCK

Keeps time with power off. Our industrial clock utilizes a new lithium battery for 3-9 years use. Easiest clock to program you'll ever see. Runs in all S-100 systems.

- ☐ Year, date, hrs, mins, secs, msecs.
- ☐ Uses new LSI CMOS chip.
- ☐ Vectored interrupts.
- ☐ CLK-24 .... $250

A/D CONVERTER

IEEE696/S-100 AIM-12 industrial standard module designed for industrial analog-to-digital use.

☐ Runs in all S-100 systems.
☐ 32-channel, 16-differential ☐ 12-bit resolution/accuracy. ☐ 25-microsecond conversions.
☐ Instrumentation amplifier.
☐ BASIC program provided. ☐ AIM-12, $695 or $785 w/1-1000 gain transducer amplifier.

D/A CONVERTER

AOM-12 IEEE696/S-100 industrial level digital-to-analog (D/A) converter.

- ☐ 12-bit ± 1/2 L.S.B. accuracy over full 0-70°C temperature range.
- ☐ Outputs 0-10, ± 5, or ± 10 volts.
- ☐ Short circuit protection, all outputs.
- ☐ Switch-programmable for multiple boards.
- ☐ AOM-12, $575

VIC 4-20

Standard output for industrial control 4-20 mA D/A converter. Used in conjunction with the D/A board.

VIC4-20, $445.

DUAL 77 Data Acquisition and Control System-

Built to industrial standards; designed for severe environments. BASIC language makes programming easy. Access to hundreds of sensors. Expandability to meet your increased needs. Nonvolatile memory. Power interruption recovery with automatic restart. DUAL 77 is economical; $5985 & up.

DUAL SYSTEMS CONTROL CORPORATION DUAL **system reliability/system integrity**

720 Channing Way • Berkeley • CA 94710 • (415) 549-3854 or (415) 549-3890 • TWX 910366 2035

Reader Service Number 1

IEEE MICRO

November 1981 Volume 1 Number 4 (ISSN 0272-1732)

ARTICLES

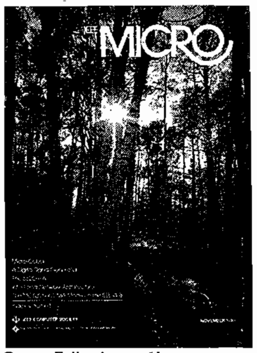

Cover: Fall colors at Hart Prairie, San Francisco Peaks, Arizona.

Photo: Jerry Jacka
Design: Jay Simpson

DEPARTMENTS

A student's or designer's ability to predict the outcome of a process is only as good as the model for that process.

An Improved Model for a Microcomputer Component—The 6520 PIA

Donald F. Hanson
University of Mississippi

Digital integrated circuits have dramatically increased in capability over the last ten years. More and more uses are found for IC chips every year. Every new part is usually accompanied by a data sheet and a manual. The detailed operation of the part, however, is usually not included in either, so few people other than the original designers know how the part actually works or is implemented in silicon. This often keeps users from finding novel applications for the part. New approaches to the delivery of product information—ones which let users learn the details of operation and implementation that the designers know—are required. Here, using the 6520 peripheral interface adaptor, we present an example of such an approach. We hope this model will help users understand how the 6520 is made. (A similar model for the 6820 PIA has already been presented by Jordan and Smith.[1])

Models

In engineering, a leader (or professor) often communicates new material to a student by describing models for the student to use. Models give the student a mental image of the process being carried out. If the student has an adequate model at hand, then he can analyze it to determine a new state of the process. If the student does not have an adequate model, then he will have trouble determining the correct final state of the process. The student's ability to predict the outcome of a process is only as good as the model for that process.

The situation is even more critical for the digital engineer—he will come in contact with many different types of digital equipment and will probably not have time to read the manuals for every system he has to use or repair. Some standard, easily understood method is needed for the description of the operation of individual digital microcomputer components (a microcomputer component is either a microprocessor or a related part like a PIA).

Modeling levels. Microcomputer component models should express the structure of hardware algorithms. Bell and Newell[2] identify five levels at which any digital system may be modeled:

- the PMS—processor memory switch—level, which describes the system in terms of processing units, memory components, peripherals, and switching networks;
- the instructional (programming) level, where instructions and their interpretation rules are specified;
- the RT—register-transfer—level, where registers are system elements and where the data transfer among them is specified according to some rule;
- the switching circuit level, where the system structure consists of an interconnection of gates and flip-flops and where the behavior of the system is described by a set of boolean equations; and
- the circuit level, where gates and flip-flops are replaced by circuit elements such as transistors, diodes, and resistors.

A sixth level[3] is the device level, which deals with p, n, and metal areas.

The RT level has been divided[4] into three sublevels:

- the RT structural level, where timing information is intact and the system is described in terms of actual components and their interconnections;
- the RT functional level, where timing may or may not remain intact and where the system is described in terms of the actual components and their functional relationships; and
- the RT behavioral level, where timing is not given and where the properties of the system are specified in terms of the input/output relationships among variables.

If the exact timing relationships of the system are to be studied, then a model at the RT structural level should be used. If exact timing of events is not important, then a functional- or behavioral-level model will suffice.

Models for digital systems are usually either graphical models or register-transfer language models. A graphical model is the Petri-net.[5] An RT language model is a list of statements describing the register transfers and the logical conditions that initiate them. RT languages are either procedural or nonprocedural[4]—a procedural language requires an explicit sequential ordering of the statements in the language; a nonprocedural language attaches no meaning to the ordering of statements. A statement is associated with a "label" that specifies the condition under which the operation specified by the statement is activated. Hence, a nonprocedural language is best for modeling hardware which has concurrent operations.

The user of microcomputer components often has trouble understanding the operation of a component because circuit level, switching circuit level, and register-transfer level models of the component are not available. The user must have enough information so that he can analyze any interface design for proper operation. Many users are forced to experiment with the part to learn how it works.

Models for microcomputer components are usually specified at the programming level or the RT behavioral level. Data that the user can use to do an analysis of his interfacing approach are generally not given in RT form. Consider, for example, the 6520 PIA—the data sheets and instruction manual use words, tables, charts, timing diagrams, a pin-out, and a block diagram to "model" the device. First-time users and students often have difficulty understanding the words and memorizing the tables and diagrams. Unfortunately, the details they want are often buried in such words or diagrams. Textbook writers usually repeat (or even say less than) what is in the manuals.

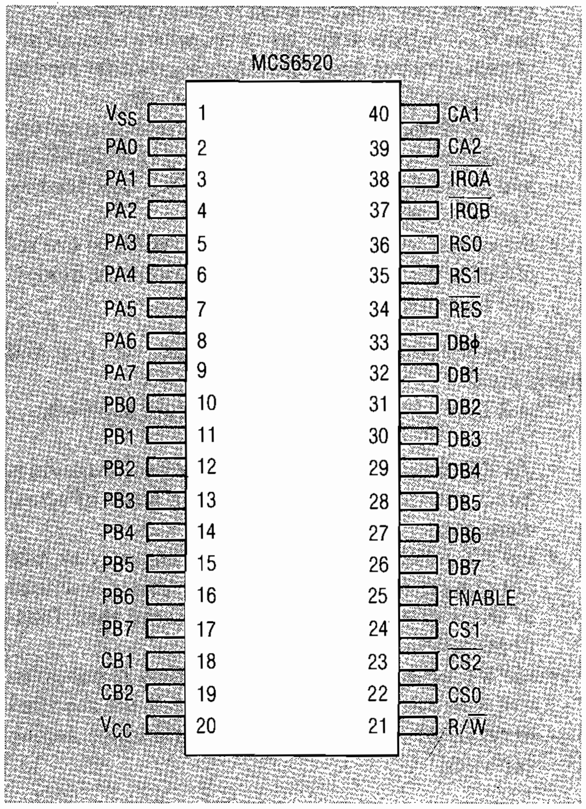


Figure 1. 6520 pin-out.

By employing a nonprocedural RT structural-level model (similar to those used by Chu[6] and Mano[7]) along with the manufacturer's timing diagram, our approach gives users easy access to operational detail. The circuit-level and switching-circuit-level blocks used in the design of the 6520 are given along with their RT structural-level equivalents.

A new model for the 6520 PIA

The 6520 was designed by Synertek, Inc., of Santa Clara, California. Figure 1 is the manufacturer's pin-out for the 6520. Figure 2 is a new block diagram. The control logic block shows many inputs and outputs; twelve control input pins and two output pins ($\overline{IRQA}$ and $\overline{IRQB}$) appear at the left. The control inputs and eight timing control flip-flops provide select signals for reading or writing the internal registers. The data I/O bus is on the top and two I/O ports, A and B, are on the right. External I/O pins are R/$\overline{W}$ (read/write control); CS0, CS1, $\overline{CS2}$ (chip selects); $\overline{RESET}$; DBn (data bus pin n for $n = 0, 1, 2, 3, 4, 5, 6, 7$); PA$n$ (port A pin n); PBn (port B pin n); RS1 and RS0 (register selects); $\overline{IRQA}$ and $\overline{IRQB}$ (interrupt request output pins for ports A and B); CA1, CB1, CA2, and CB2 (interrupt input and handshaking); and ENABLE (= E)—the Φ2 clock pin when hooked up in the usual way.

There are seven internal registers and two internal buses. The first register is the DIR—data input register, which is loaded with the external data bus (DB0-DB7) whenever ENABLE (Φ2) is asserted (high). The DIR then outputs this data onto the internal input bus to the remaining six registers—the A control register AC, the B control register BC, the A data direction register AD, the B data direction register BD, the A output register AO, and the B output register BO. Each register has an internal load, clear, and bus enable line associated with it. Whenever a register's load control is asserted (high), the register is loaded with whatever is on the internal input bus. Whenever the $\overline{RESET}$ (clear) line is asserted (low), all registers are reset to zero. Whenever a bus enable line is asserted (high), data from the selected register is placed on the internal output bus. This data is put on the external data bus (DB0-DB7) only if the tristate enable associated with the tristate buffers of the data bus becomes true (high).

Circuit level. At the circuit level, the registers consist of latches made from enhancement-mode MOSFETs with depletion-mode load devices (Figure 3a). More load or reset inputs can be added by simply paralleling more MOSFETs. Figure 3b represents the equivalent of Figure 3a at the switching circuit level.

Figure 4 shows the switching-circuit-level model of the circuit used for bit 7 of AC and BC. RESET is the complement of the reset signal coming into the 6520. The output of AC7 can be shown to be

$$AC7 = \overline{\overline{AC7} \cdot \overline{L} + R + RESET}.$$

CONTROL SIGNAL NOTATION

DBOE	=	$CS0 \cdot CS1 \cdot (\overline{CS2})' \cdot R/\overline{W} \cdot \overline{RESET} \cdot E$	Data bus output enable
ADS	=	$RS1' \cdot RS0' \cdot AC2'$	A direction select
AS	=	$RS1' \cdot RS0' \cdot AC2$	A select
ACS	=	$RS1' \cdot RS0$	A control select
BDS	=	$RS1 \cdot RS0' \cdot BC2'$	B direction select
BS	=	$RS1 \cdot RS0' \cdot BC2$	B select
BCS	=	$RS1 \cdot RS0$	B control select
ADIE	=	$RS1D' \cdot RS0D' \cdot AC2' \cdot \overline{RESET} \cdot W \cdot E'$	A direction input enable
AIE	=	$RS1D' \cdot RS0D' \cdot AC2 \cdot \overline{RESET} \cdot W \cdot E'$	A input enable
ACIE	=	$RS1D' \cdot RS0D \cdot \overline{RESET} \cdot W \cdot E'$	A control input enable
BDIE	=	$RS1D \cdot RS0D' \cdot BC2' \cdot \overline{RESET} \cdot W \cdot E'$	B direction input enable
BIE	=	$RS1D \cdot RS0D' \cdot BC2 \cdot \overline{RESET} \cdot W \cdot E'$	B input enable
BCIE	=	$RS1D \cdot RS0D \cdot \overline{RESET} \cdot W \cdot E'$	B control input enable

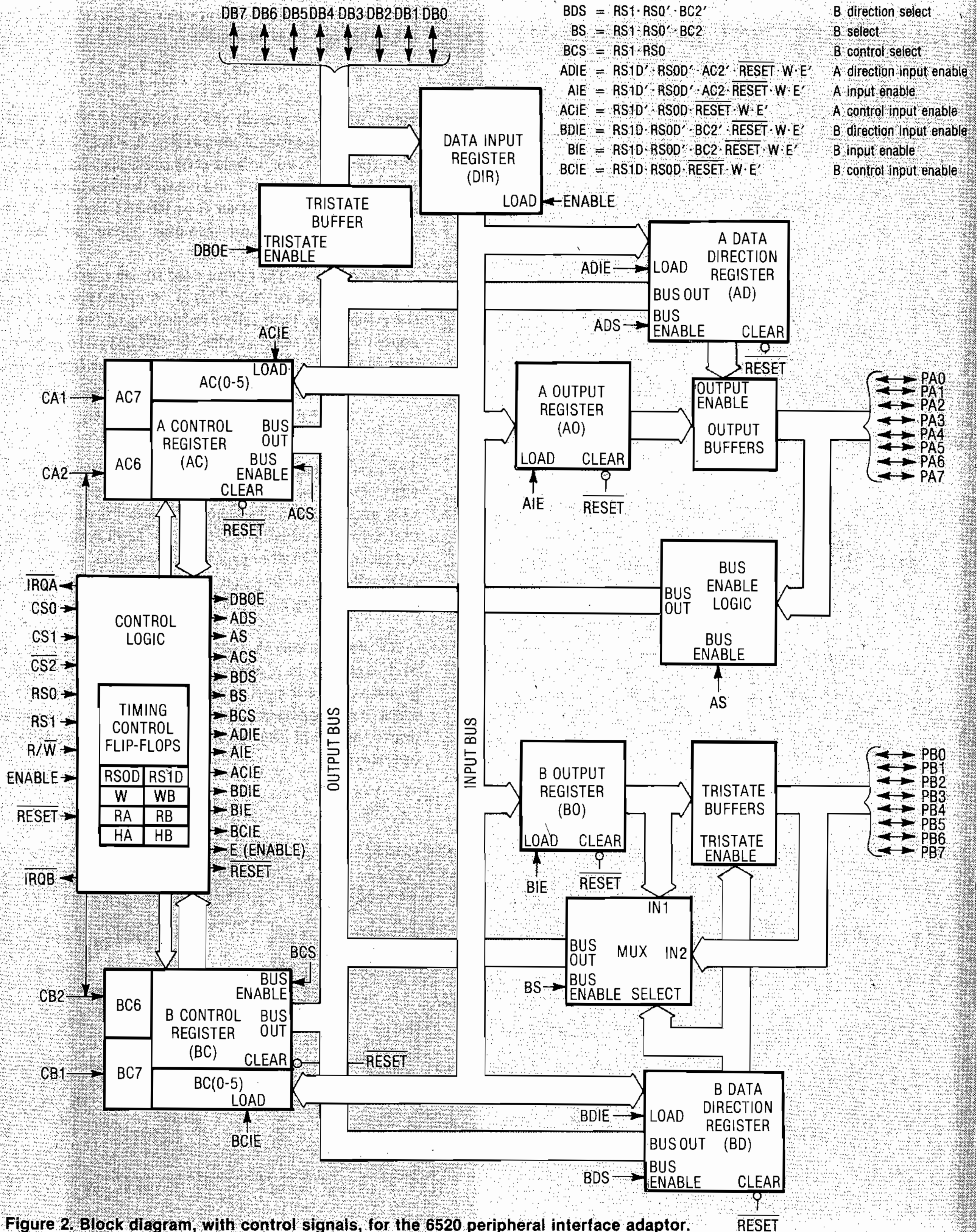


Figure 2. Block diagram, with control signals, for the 6520 peripheral interface adaptor.

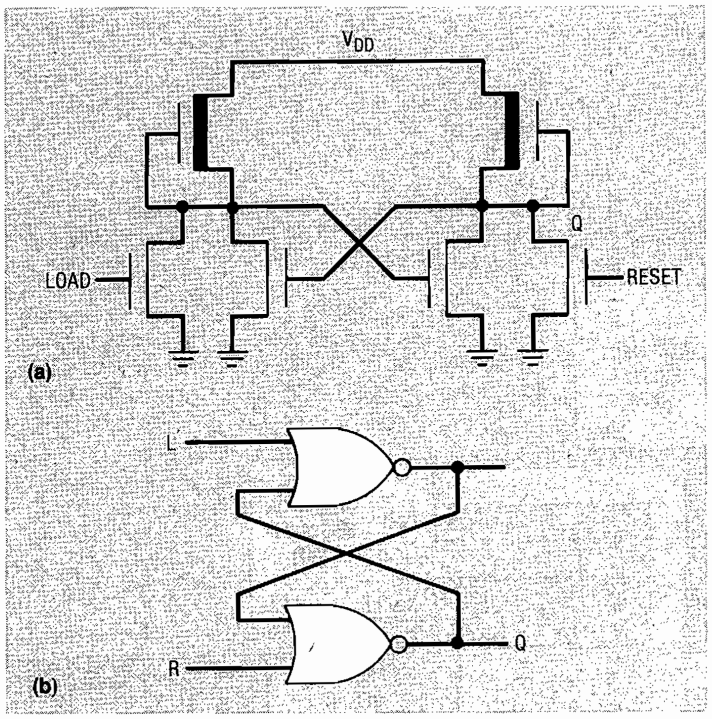


Figure 3. Circuit-level latch (a); switching-circuit-level latch (b).

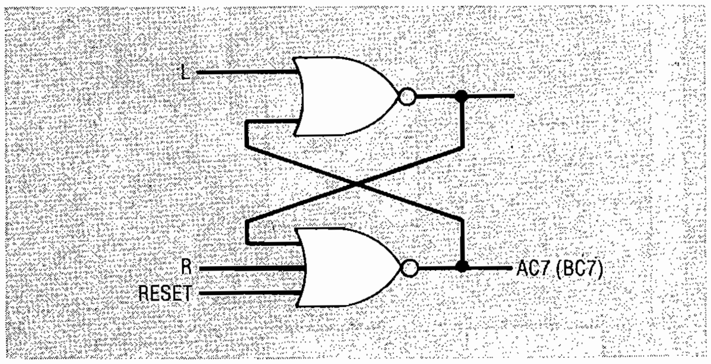


Figure 4. Circuit for AC7 (and BC7).

If R = 1 or RESET = 1, then AC7 = 0. If R = 0 and RESET = 0, then

$$AC7 = AC7 + L.$$

This says that if L = 0, then AC7 is held at its previous value. Otherwise, if AC7 = 0 and L becomes 1, then AC7 goes to 1. This can be summarized by the nonprocedural RT structural-level statement:

$$\begin{aligned} R'\cdot RESET'\cdot L:&\quad AC7 \leftarrow 1\\ R+RESET:&\quad AC7 \leftarrow 0. \end{aligned}$$

Here, primes represent the complement of a variable. This simply says that as long as R = 0, RESET = 0, and L = 0, then AC7 will be held at its previous value. If L goes to 1, then AC7 goes to 1 and is held there as long as R = 0 and RESET = 0. If either R or RESET becomes 1, then AC7 goes to 0 and remains there as long as R′·RESET′·L = 0.

Figure 5 shows the circuit used for bit *n* in the DIR and for the same bit of AD. This is simply a type of master-slave flip-flop, with selective loading of the slaves. The control input L selects which slave (if any) is to receive the data in the DIR. The RT statements for this hardware are

$$\begin{aligned} E:&\quad DIRn \leftarrow DBn\\ RESET'\cdot L\cdot E':&\quad ADn \leftarrow DIRn\\ RESET:&\quad ADn \leftarrow 0 \end{aligned}$$

The first statement says that bit *n* of DIR will be loaded with DB*n* as long as E is high. When E goes low, DIR*n* is held until E goes high again. The second statement says that as long as RESET = 0 and L = 1, bit *n* of DIR will be loaded into AD*n* whenever E goes low. If L = 0, however, AD*n* will be held at its current value. The last statement says that as long as RESET = 1, AD*n* is held at zero.

RT description. A complete RT description for the 6520 appears in Table 1, which lists five statement groups—one for port operation (PA and PB), one for register-read operation, one for register-write operation, one for interrupt flag operation, and one for handshaking operation.

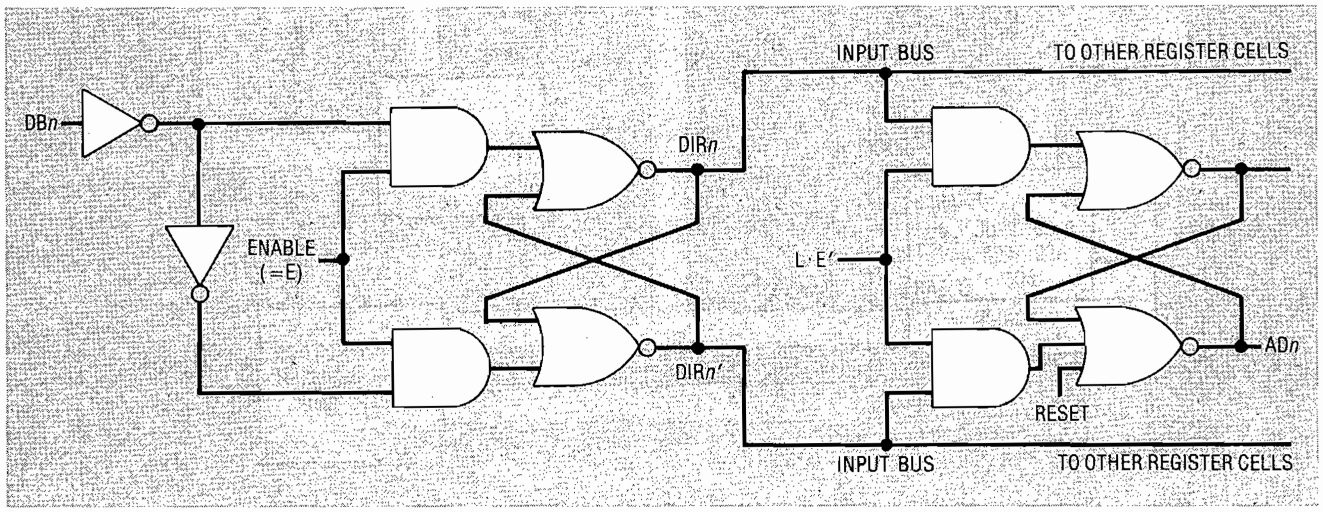


Figure 5. DIR*n* and AD*n* switching-circuit-level circuitry.

Table 1.
6520 operation.

6520 PORT OPERATION (PA, PB)		
AD*n*:	PA*n* = AO*n*	
BD*n*:	PB*n* = BO*n*	
AD*n*′ :	PA output driver (*n*) → high Z	
BD*n*′ :	PB tristate (*n*) → high Z	*n* = 0,1,2,3,4,5,6,7
6520 REGISTER-READ OPERATION		
R/$\overline{\text{W}}$·CS0·CS1·($\overline{\text{CS2}}$)′·$\overline{\text{RESET}}$·E:	DB*n* = RS1′·RS0′·AC2′·AD*n* ∨ RS1′·RS0′·AC2·(AO*n*′·AD*n*)′ ·PA*n** ∨ RS1′·RS0 · AC*n* ∨ RS1·RS0′·BC2′·BD*n* ∨ RS1·RS0′·BC2 ·(PB*n*·BD*n*′ ∨ BO*n*·BD*n*) ∨ RS1·RS0·BC*n*	
(R/$\overline{\text{W}}$·CS0·CS1·($\overline{\text{CS2}}$)′·$\overline{\text{RESET}}$·E)′:	DB tristate (*n*) → high Z	
6520 REGISTER-WRITE OPERATION		
E:	DIR*n* ← DB*n*	*n* = 0,1,2,3,4,5,6,7
E:	RS0D ← RS0	
E:	RS1D ← RS1	
(R/$\overline{\text{W}}$)′·$\overline{\text{RESET}}$·CS0·CS1·($\overline{\text{CS2}}$)′·E:	W ← 1	
[CS0·CS1·($\overline{\text{CS2}}$)′]′·E + ($\overline{\text{RESET}}$)′:	W ← 0	
RS1D′·RS0D′·AC2′·$\overline{\text{RESET}}$·W·E′:	AD ← DIR	
RS1D′·RS0D′·AC2·$\overline{\text{RESET}}$·W·E′:	AO ← DIR	
RS1D′·RS0D·$\overline{\text{RESET}}$·W·E′:	AC(0-5) ← DIR(0-5)	
RS1D·RS0D′·BC2′·$\overline{\text{RESET}}$·W·E′:	BD ← DIR	
RS1D·RS0D′·BC2·$\overline{\text{RESET}}$·W·E′:	BO ← DIR	
RS1D·RS0D·$\overline{\text{RESET}}$·W·E′:	BC(0-5) ← DIR(0-5)	
($\overline{\text{RESET}}$)′:	AD,AO,AC,BD,BO,BC ← 0	
6520 INTERRUPT FLAG OPERATION		
AC0·AC7 + AC3·AC6:	$\overline{\text{IRQA}}$ = 0	
BC0·BC7 + BC3·BC6:	$\overline{\text{IRQB}}$ = 0	
(AC0·AC7 + AC3·AC6)′:	$\overline{\text{IRQA}}$ output driver → high Z	
(BC0·BC7 + BC3·BC6)′:	$\overline{\text{IRQB}}$ output driver → high Z	
CS0·CS1·($\overline{\text{CS2}}$)′·RS1′·RS0′·AC2·R/$\overline{\text{W}}$·$\overline{\text{RESET}}$·E:	RA ← 1	
[CS0·CS1·($\overline{\text{CS2}}$)′]′·E:	RA ← 0	
CS0·CS1·($\overline{\text{CS2}}$)′·RS1·RS0′·BC2·R/$\overline{\text{W}}$·$\overline{\text{RESET}}$·E:	RB ← 1	
[CS0·CS1·($\overline{\text{CS2}}$)′]′·E:	RB ← 0	
RA′·$\overline{\text{RESET}}$·(AC1′·CA1↓ + AC1·CA1↑):	AC7 ← 1	
RA + ($\overline{\text{RESET}}$)′:	AC7 ← 0	
AC5′·RA′·$\overline{\text{RESET}}$·(AC4′·CA2↓ + AC4·CA2↑):	AC6 ← 1	
RA + AC5 + ($\overline{\text{RESET}}$)′:	AC6 ← 0	
RB′·$\overline{\text{RESET}}$·(BC1′·CB1↓ + BC1·CB1↑):	BC7 ← 1	
RB + ($\overline{\text{RESET}}$)′:	BC7 ← 0	
BC5′·RB′·$\overline{\text{RESET}}$·(BC4′·CB2↓ + BC4·CB2↑):	BC6 ← 1	
RB + BC5 + ($\overline{\text{RESET}}$)′:	BC6 ← 0	
6520 HANDSHAKING OPERATION (CA2, CB2)		
AC5:	CA2 = HA	
BC5:	CB2 = HB	
AC5′ :	CA2 output driver → high Z	
BC5′ :	CB2 tristate → high Z	
W·RS1D·RS0D′·BC2·E′ :	WB ← 1	
W′·E′ :	WB ← 0	
AC4′·AC3′·AC7 + AC4′·AC3·RA′·E′ + AC4·AC3 + AC5′ :	HA ← 1	
AC5·(AC4′·AC3′·AC7′·RA·E′ + AC4′·AC3·RA·E′ + AC4·AC3′):	HA ← 0	
BC4′·BC3′·BC7 + BC4′·BC3·WB′·E + BC4·BC3 + BC5′ :	HB ← 1	
BC5·(BC4′·BC3′·BC7′·WB·E + BC4′·BC3·WB·E + BC4·BC3′):	HB ← 0	

*PA*n* (PB*n*) is the logic level of the external voltage applied to port A(B) pin "*n*." When PA*n* is configured as an output, a read operation samples the data on PA*n*. No multiplexer is used, unlike the set-up for port B in Figure 2. Should input data be placed on an output pin, a low on PA*n* will override the reading of AO*n*. For normal output operation (when input data is not placed on an output pin), one should let PA*n* = 1 in this equation.

LEGEND

DB*n*	Data bus (*n*)
DIR	Data input register
RS0D	Register-select 0 delayed flip-flop
RS1D	Register-select 1 delayed flip-flop
W	Write-control flip-flop
RA	Read A flip-flop
RB	Read B flip-flop
WB	Write B flip-flop
HA	Handshake on CA2 flip-flop
HB	Handshake on CB2 flip-flop

Port operation. The first group describes the effect that the data-direction-register bits AD*n* and BD*n* have on bit *n* of port A's output drivers and bit *n* of port B's tristates. If AD*n* = 1, then the port A output driver (open drain with pull-up) is active, and the logic level of AO*n* (Figure 2) is placed on the output pin PA*n*. If AD*n* = 0, then the port A output driver is disabled and PA*n* functions as an input. A similar result holds for port B. Thus, the individual bits of the data-direction-registers AD and BD can be used to program the individual lines of PA and PB to be either inputs or outputs.

Register-read operation. The second group is composed of two RT statements. The first says that if the control expression before the colon is true, then the expression to the right of the equal sign is output to the data bus. The control expression is the DBOE signal (see legend for Figure 2) shown hooked to the tristate enables on the data bus tristate buffers (Figure 2). Data are always present on the input to these buffers (even if the chip is not selected) and are transferred to the data bus only if DBOE becomes true and enables the tristate buffer outputs. The output bus in Figure 2 is made up of eight 6 × 1 multiplexers with special select codes; the logical expressions for these codes are labeled ADS, AS, ACS, BDS, BS, and BCS in the legend. Only one of these selects can be logical 1 at any one time, as the ANDing of any select expression with any other select expression demonstrates. Zero is always the result. That all possible cases are covered by these six selects can be shown by ORing them:

$$\mathrm{ADS} \vee \mathrm{AS} \vee \mathrm{ACS} \vee \mathrm{BDS} \vee \mathrm{BS} \vee \mathrm{BCS} = 1$$

In Figure 2, one can see that if ADS = 1, then AD is put on the output bus, and if DBOE is also logical 1, then AD is put on the data bus. Similarly, ACS, BDS, and BCS put AC, BD, and BC, respectively, on the output bus. Ports A and B are not identical, since there are eight 2 × 1 multiplexers in port B which are not in port A. When bit *n* of port A is configured as an output by AD*n*, AO*n* will be read from the logic state of the voltage on PA*n* and not from AO*n* directly (as is the case for port B). Thus, if AS = 1, then $(\mathrm{AO}n' \cdot \mathrm{AD}n)' \cdot \mathrm{PA}n$ will be placed on bit *n* of the output bus. Similarly, if BS = 1, then $(\mathrm{PB}n \cdot \mathrm{BD}n' \vee \mathrm{BO}n \cdot \mathrm{BD}n)$ will be put on the output bus.

Register-write operation. This group of statements describes how data are written into registers. The first statement says that as long as ENABLE (= E) is high, the data input register DIR samples the data on the data bus. The falling edge of E latches the signal on the data bus into DIR. The next two statements provide delayed signals for register select 0 and 1. RS0 and RS1 are usually attached to the A0 and A1—address 0 and 1—signals from the 6502 microprocessor. E is usually interfaced to the system Φ2 clock. A0 and A1 exist from the beginning of one Φ1 to the beginning of the next Φ1, and RS0D and RS1D exist from the beginning (0 to 1 transition) of one Φ2 to the beginning of the next Φ2 (This is why we call them delayed signals.) The next two statements show the conditions for setting and clearing the internal W flip-flop. Whenever the chip is selected with R/$\overline{\mathrm{W}}$ low, W is set to 1 when E goes high. If the chip is not selected, then W is cleared when E goes high. Hence, W is also a Φ2-to-Φ2 signal. The last seven statements show how data in DIR are transferred to one of the other six internal registers when W is high and E goes low. The action of DIR is like that of the master in a master-slave flip-flop arrangement; the other six registers behave like selectively addressed slaves. The last six equations in the legend for Figure 2 describe the relationships of the control conditions to the load controls shown in Figure 2. Note that AC6, AC7, BC6, and BC7 cannot be altered by writing through the data bus.

Interrupt flag operation. This statement group gives the conditions under which the interrupt flags AC6, AC7, BC6, and BC7 can be set or cleared. The output drivers for pins 38 and 37—$\overline{\mathrm{IRQA}}$ and $\overline{\mathrm{IRQB}}$, respectively—are two open-drain MOSFETs. The first two statements show the conditions under which these MOSFETs are in low-impedance output conditions and the next two statements show the conditions under which $\overline{\mathrm{IRQA}}$ and $\overline{\mathrm{IRQB}}$ are in high-impedance states. The circuit diagram for $\overline{\mathrm{IRQA}}$ is shown in Figure 6. The next four statements show the conditions under which the RA and RB—read port A and read port B—internal flip-flops are set or cleared. The last eight statements show the conditions under which AC7, AC6, BC7, and BC6 are set or cleared—for example, when AC1 = 0 and input CA1 falls from 1 to 0, or when AC1 = 1 and input CA1 rises from 0 to 1, then AC7 is set to 1. AC7 can be cleared only if data are read out of port A or if $\overline{\mathrm{RESET}}$ is brought low. The conditions for AC6, BC7, and BC6 are similar.

Handshaking operation. The final group of statements describes the operation of the handshaking lines CA2 and CB2. Bit 5 of each control register controls the direction of signal flow on CA2 and CB2. The first four statements say that if AC5(BC5) = 1, then CA2(CB2) takes on the value of the internal flip-flop HA(HB). Otherwise, if AC5(BC5) = 0, then CA2(CB2) is programmed as an interrupt input. The next two statements show the conditions under which the internal flip-flop WB (write port B) is set or cleared. WB is a Φ1-to-Φ1 signal that is one clock period behind the R/$\overline{\mathrm{W}}$ signal. The last four statements show the conditions for setting and clearing the internal flip-flops HA and HB. (Tables 2 and 3 show the operation of these flip-flops.) The first case presented in Table 2 gives a handshake on reading port A, and the first case in Table 3 gives a handshake on writing port B. The second case in Table 2 outputs a short negative pulse upon

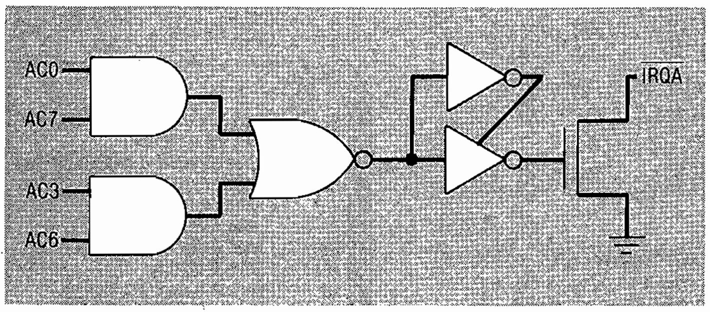


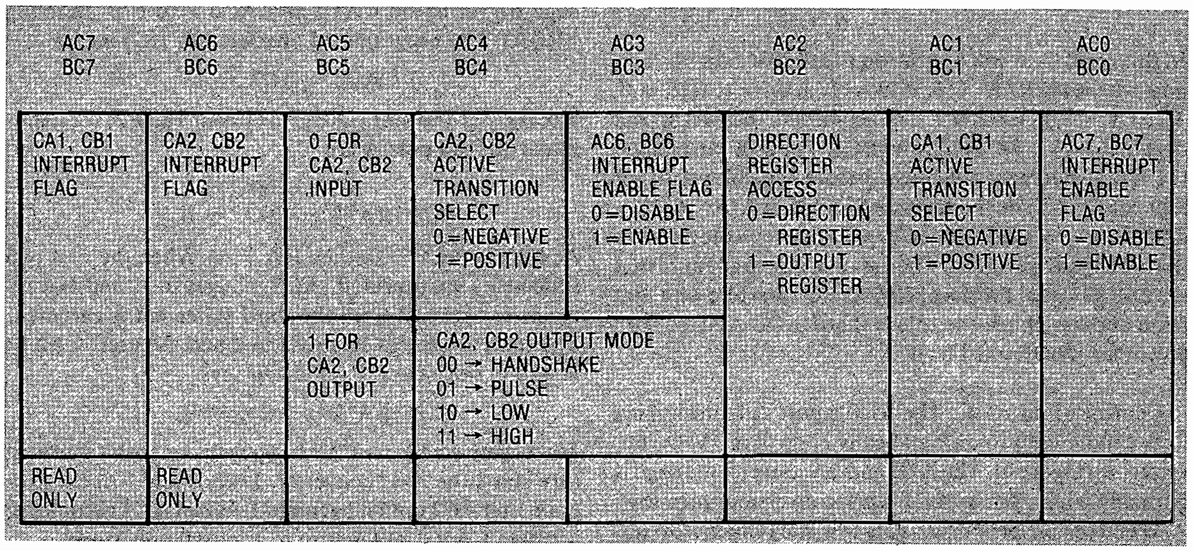
Figure 6. Circuit diagram for $\overline{\mathrm{IRQA}}$.

Table 2.
Operation of HA.

AC4	AC3	ACTION
0	0	AC7′·RA·E′: HA ← 0; AC7: HA ← 1
0	1	RA·E′: HA ← 0; RA′·E′: HA ← 1
1	0	HA ← 0
1	1	HA ← 1

Table 3.
Operation of HB.

BC4	BC3	ACTION
0	0	BC7′·WB·E: HB ← 0; BC7: HB ← 1
0	1	WB·E: HB ← 0; WB′·E: HB ← 1
1	0	HB ← 0
1	1	HB ← 1



AC7 BC7	AC6 BC6	AC5 BC5	AC4 BC4	AC3 BC3	AC2 BC2	AC1 BC1	AC0 BC0
CA1, CB1 INTERRUPT FLAG	CA2, CB2 INTERRUPT FLAG	0 FOR CA2, CB2 INPUT	CA2, CB2 ACTIVE TRANSITION SELECT 0=NEGATIVE 1=POSITIVE	AC6, BC6 INTERRUPT ENABLE FLAG 0=DISABLE 1=ENABLE	DIRECTION REGISTER ACCESS 0=DIRECTION REGISTER 1=OUTPUT REGISTER	CA1, CB1 ACTIVE TRANSITION SELECT 0=NEGATIVE 1=POSITIVE	AC7, BC7 INTERRUPT ENABLE FLAG 0=DISABLE 1=ENABLE
		1 FOR CA2, CB2 OUTPUT	CA2, CB2 OUTPUT MODE 00 → HANDSHAKE 01 → PULSE 10 → LOW 11 → HIGH				
READ ONLY	READ ONLY						

Figure 7. A control-register programming form.

reading port A, and the second case in Table 3 outputs a short negative pulse upon writing port B. The last two cases in each table are manual low and high outputs. (Figure 7 shows a useful control-register programming form.)

The RT-level model presented in Table 1, when used with the manufacturer's timing data, enables the designer to analyze his hardware and software schemes for errors. Tracey[8] points out that an RT structural-level description, such as that in Table 1, can be tedious. We assume, however, that the designer would use the table as a reference tool only, to help him analyze a particular operation or hook-up. Such a model should also be helpful to anyone needing to test a 6520.

Use of the model for analysis of interfacing schemes

The RT statements in Table 1 can serve as a tool for analyzing interfacing designs—here, we will discuss interfacing between a 6502 and a 6520, and between a 6520 and an external A/D converter.

6502-to-6520 interface. Consider, for example, the interfacing situation shown in Figure 8. The 6520 operation for this case can be obtained simply by substituting the 6502 outputs into the corresponding 6520 RT statements.

Z-80 FORTH $50.00

Z-80 FORTH—a complete program development system. Full fig-FORTH and FORTH-79 vocabularies, plus extensions for access to all CP/M (tm) functions. Uses standard CP/M compatible random access disk files for screen storage. Package includes: interpreter/compiler with virtual memory management, line editor, screen editor, Z-80 Assembler, decompiler, utilities, demonstration programs, and 65 page user manual. System requirements: Z-80 microcomputer, 48 kbytes RAM, CP/M 2.2 or MP/M 1.1 $50.00

Z-80 FORTH WITH NAUTILUS SYSTEMS CROSS-COMPILER. Extend/modify the FORTH runtime system, recompile on a host computer for a different target computer, generate headerless code, generate ROMable code with initialized variables. Supports forward referencing to any word or label. Produces load map and list of unresolved symbols. 170 page manual. System requirements as for Z-80 FORTH above. $200.00

MACHINE TEST PROGRAM PACKAGE FOR Z-80 systems. Includes memory, floppy disk, printer, and terminal tests with all source code. Requires CP/M 2.2 $50.00

All software distributed on eight-inch soft sectored single density diskettes. Prices include shipping by first class or UPS within USA or Canada. COD charges extra. Purchase orders accepted at our discretion. (CP/M and MP/M are registered trademarks of Digital Research, Inc. Z-80 is a registered trademark of Zilog, Inc.)

Laboratory Microsystems
4147 Beethoven Street
Los Angeles, CA 90066
(213) 390-9292

RS No. 4 for Z80 Forth **RS No. 5 for Test Programs**

Doing this for the register-read operation, one obtains

$R/\overline{W} \cdot A13 \cdot A14 \cdot A15' \cdot \overline{RESET} \cdot \Phi2$:

$$\begin{aligned}
DBn = {} & A1' \cdot A0' \cdot AC2' \cdot ADn \\
\vee {} & A1' \cdot A0' \cdot AC2 \cdot (AOn' \cdot ADn)' \cdot PAn \\
\vee {} & A1' \cdot A0 \cdot ACn \\
\vee {} & A1 \cdot A0' \cdot BC2' \cdot BDn \\
\vee {} & A1 \cdot A0' \cdot BC2 \cdot (PBn \cdot BDn' \vee BOn \cdot BDn) \\
\vee {} & A1 \cdot A0 \cdot BCn.
\end{aligned}$$

$$n = 0, 1, 2, 3, 4, 5, 6, 7$$

In this case, data will appear on the 6520's data bus outputs whenever $R/\overline{W} = 1$, $A13 = 1$, $A14 = 1$, $A15 = 0$, $\overline{RESET} = 1$, and $\Phi2 = 1$. The actual data that will appear depends on A1, A0, AC2, and BC2. The user can analyze the DBOE (control statement before the colons) to see that for hexadecimal addresses \$6XXX or \$7XXX (X = don't care), data from the 6520 will appear on the data bus during a read operation. The user can also make sure that the DBOE signal for the 6520 does not interfere with that from another chip—the rule is that DBOE ANDed with any other DBOE must be 0.

Control of an ADC. The 6520's port A handshaking lines can be used to automatically control an A/D converter[9,10] (Figure 9). The A/D end-of-conversion signal is normally high. When A/D start conversion goes low, A/D end of conversion also goes low and remains low until conversion is completed. If CA1 is programmed to set AC7 to 1 on a positive edge (AC1 = 1), then AC7 can be polled to determine when conversion is complete. A simple AIM-65 program was used to run the A/D converter:

```
         LDA #26     ;Handshake on read port A
                     ;and
         STA 9801    ;positive edge triggering on
                     ;CA1
         LDA 9800    ;sets CA2 to 0 to trigger one-
                     ;shot and start conversion.
LOOP:    BIT 9801    ;Through converting?
         BPL LOOP    ;No, LOOP again.
         LDA 9800    ;Yes, reset AC7 and start
                     ;conversion again.
         JMP LOOP    ;Go to LOOP.
```

The first two lines in the above program put AC5 = 1 (for CA2 output), AC4 = 0, AC3 = 0 (for handshaking on read), AC2 = 1 (for port A access), AC1 = 1 (for positive edge triggering on CA1), and AC0 = 0 (for disabling $\overline{IRQA}$). The programmed operation of AC7 is now

$$CA1\uparrow: \quad AC7 \leftarrow 1.$$

The register-transfer statements (from Table 2) for handshake on read port A are

$$\begin{aligned} AC7: &\quad HA \leftarrow 1 \\ AC7' \cdot RA \cdot E': &\quad HA \leftarrow 0. \end{aligned}$$

As long as AC5 = 0, then HA = 1. When the first two statements are executed, AC5 changes to 1, and HA remains at logical 1 until the third statement is executed. The third statement sets AC7 to 0 and then sets CA2 = HA to 0, since the RT statement says

$$AC7' \cdot RA \cdot E': \quad HA \leftarrow 0.$$

The negative edge on CA2 triggers the one-shot which in turn starts the A/D converter. Lines 4 and 5 of the program poll AC7 until it becomes 1. When AC7 becomes 1, it makes CA2 = HA return to a logical one, since

$$AC7: \quad HA \leftarrow 1.$$

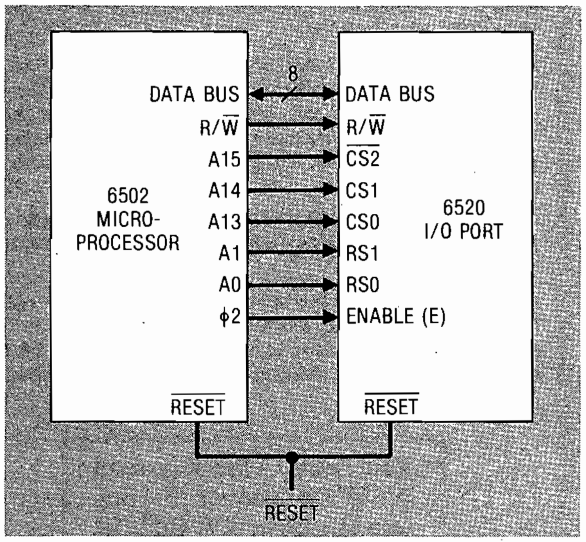


Figure 8. An interface hook-up.

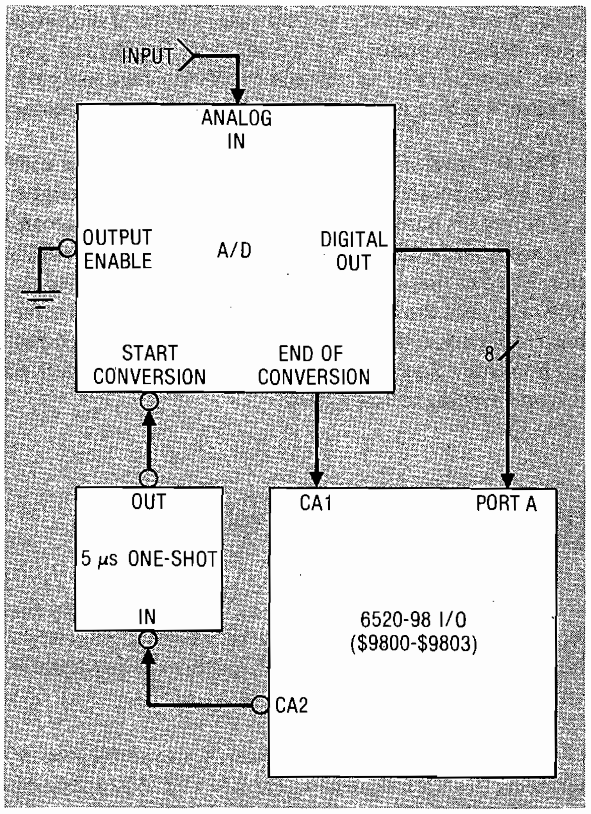


Figure 9. A/D converter connections.

Line 6 resets AC7 to 0 and sets CA2 = HA to 0, which starts the converter going again. Thus, the program runs the A/D converter automatically.

A nonprocedural RT structural-level model provides a unified approach to understanding 6520 hardware, interfacing, and software. By having available a model at this level of detail, the new user or student needs less "consultant-level" assistance—he has more confidence in the interfaces he designs. ■

References

1. Jordan, H. F., and B. J. Smith, "The Assignment Statement in Hardware Description Languages," *Computer,* Vol. 10, No. 6, June 1977, pp. 43-49.
2. Bell, C. G., and A. Newell, *Computer Structures: Readings and Examples,* McGraw-Hill Book Co., New York, 1971.
3. Bell, C.G., J. C. Mudge, and J. E. McNamara, *Computer Engineering, A DEC View of Hardware Systems Design,* Digital Press, Maynard, Mass., 1978.
4. Barbacci, M. R., "A Comparison of Register Transfer Languages for Describing Computers and Digital Systems," *IEEE Trans. Computers,* Vol. C-24, No. 2, Feb. 1975, pp. 137-150.
5. Peterson, J. L., "Petri-Nets," *Computing Surveys,* Vol. 9, No. 3, Sept. 1977, pp. 223-252.
6. Chu, Y., "An ALGOL-Like Computer Design Language," *Comm. ACM,* Vol. 8, No. 10, Oct. 1965, pp. 607-615.
7. Mano, M. M., *Computer System Architecture,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976.
8. Tracey, J. H., "Difficulties in Applying Register Transfer Languages to Microcomputer System Design," *Proc. 21st Midwest Symposium on Circuits and Systems,* Hale and Michel, eds., Ames, Iowa, Aug. 1978, pp. 158-161.
9. Hanson, D. F., "A Microprocessor Laboratory for Electrical Engineering Seniors," *IEEE Trans. Education,* Vol. E-24, No. 1, Feb. 1981, pp. 8-14.
10. Hanson, D. F., "A Microprocessor Laboratory Course Based on an AIM-65 Solderless Interfacing Unit," *Proc. 1981 Southeastern Section Ann. Meeting: Computers in Engineering and Technology Education,* University of Tennessee at Chattanooga, American Soc. for Eng. Education, Apr. 1981, pp.21-28.

Donald F. Hanson is an assistant professor of electrical engineering at the University of Mississippi, University, Mississippi. His research interests include the development of mathematical and numerical techniques for solving boundary-value problems of electromagnetic theory, and digital and analog electronics applications, including microprocessors and microcomputer interfacing. He was previously an assistant professor of electrical engineering at Iowa State University, Ames.

Hanson received the BS, MS, and PhD degrees in electrical engineering from the University of Illinois, Urbana, in 1969, 1972, and 1976, respectively. He is member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu.

Hanson's address is University of Mississippi, School of Engineering, Electrical Engineering Department, University, MS 38677.