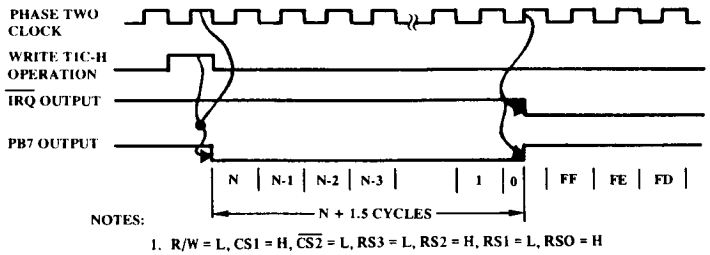


Interval Timer "One-Shot" Mode Timing Sequence
FIGURE 6-7

the ability to produce a square wave on PB7 whose frequency is not affected by variations in the processor interrupt response time. This is accomplished in the "free-running" mode.

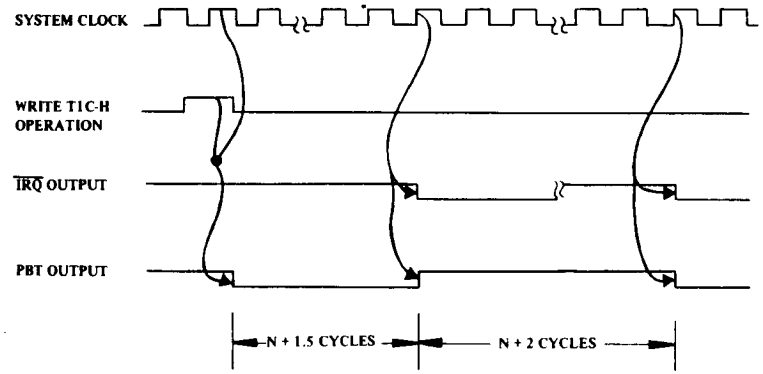
In the free-running mode (ACR6 = 1), the interrupt flag is set and the signal on PB7 is inverted each time the counter reaches zero. However, instead of continuing to decrement from zero after a time-out, the timer automatically transfers the contents of the latch into the counter (16 bits) and continues to decrement from there. The interrupt flag can be cleared by writing T1C-H, by reading T1C-L, or by writing directly into the flag as described below. However, it is not necessary to rewrite the timer to enable setting the interrupt flag on the next time-out.

All interval timers in the R6500 family devices are "re-triggerable." Rewriting the counter will always re-initialize the time-out period. In fact, the time-out can be prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the processor writes into the high-order counter (T1C-H). However, by loading the latches only, the processor can access the timer during each down-counting operation without affecting the time-out in process. Instead, the data loaded into the latches will determine the length of the next time-out period. This capability is particularly valuable in the free-running mode with the output enabled. In this mode, the signal on PB7 is inverted and the interrupt flag is set with each time-out. By responding to the interrupts with new data for

the latches, the processor can determine the period of the next half cycle during each half cycle of the output signal on PB7. In this manner, very complex pulse width modulated waveforms can be generated. Timing for the free-running mode is shown in Figure 6-8.

6.4.6 Timer 2

Timer 2 operates as an interval timer (in the "one-shot" mode only), or as a counter for counting negative pulses on the PB6 peripheral pin. A single control bit is provided in the Auxiliary Control Register to select between these two modes. This timer is comprised of a "write-only" low-order latch (T2L-L), a "read-only" low-order counter and a read/write high-order counter. The counter registers act as a 16-bit counter which decrements at $\phi 2$ rate.



Timer I Free-Running Mode
FIGURE 6-8

Timer 2 addressing can be summarized as follows:

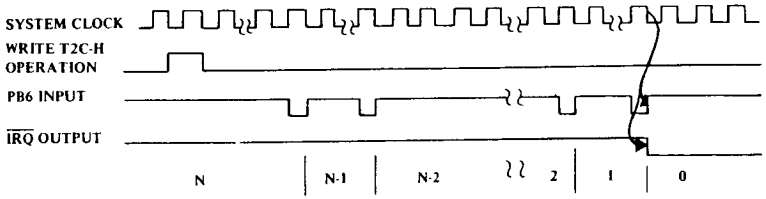
RS3	RS2	RS1	RS0	R/W = 0	R/W = 1
H	L	L	L	Write T2L-L	Read T2C-L Clear Interrupt flag
H	L	L	H	Write T2C-H Transfer T2L-L to T2C-L Clear Interrupt flag	Read T2C-H

TIMER 2 INTERVAL TIMER MODE

As an interval timer, T2 operates in the "one-shot" mode similar to Timer 1. In this mode, T2 provides a single interrupt for each "write T2C-H" operation. After timing out, the counter will continue to decrement. However, setting of the interrupt flag will be disabled after initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H to enable setting of the interrupt flag. The interrupt flag is cleared by reading T2C-L or by writing T2C-H. Timing for this operation is shown in Figure 6-9.

TIMER 2 PULSE COUNTING MODE

In the pulse counting mode, T2 serves primarily to count a predetermined number of negative-going pulses on PB6. This is accomplished by first loading a number into T2. Writing into T2C-H clears the interrupt flag and allows the counter to decrement each time a pulse is applied to PB6. The interrupt flag will be set when T2 reaches zero. At this time the counter will continue to decrement with each pulse on PB6. However, it is necessary



Timer 2 Pulse Counting Mode
FIGURE 6-9

to rewrite T2C-H to allow the interrupt flag to set on subsequent down-counting operations. Timing for this mode is shown in Figure 6-10.

6.4.7 Shift Register

The Shift Register (SR) performs serial data transfers into and out of the CB2 pin under control of an internal modulo-8 counter. Shift pulses can be applied to the CB1 pin from an external source or, with the proper mode selection, shift pulses generated internally will appear on the CB1 pin for controlling shifting in external devices.

The control bits which allow control of the various shift register operating modes are located in the Auxiliary Control Register. These bits can be set and cleared by the system processor to select one of the operating modes discussed in the following paragraphs.

SHIFT REGISTER INPUT MODES

Auxiliary Control Register ACR4 selects the shift register input or output mode. There are three input modes and four output modes, differing primarily in the source of the pulses which control the shifting operation. With ACR4=0 the input modes are selected by ACR3 and ACR2 as follows:

ACR4	ACR3	ACR2	
0	0	0	Shift Register Disabled
0	0	1	Shift in under Control of Timer 2
0	1	0	Shift in at system clock rate
0	1	1	Shift in under Control of External Input Pulses

All Shift Register inputs are sampled into the Shift Register during the Ø2 low immediately following the detection of the shift clock rising transition. This detection occurs during Ø2 high.

MODE 000 - SHIFT REGISTER DISABLED

The 000 mode is used to disable the Shift Register. In this mode the microprocessor can write or read the SR, but the shifting operating is disabled and operation of CB1 and CB2 is controlled by the appropriate bits in the Peripheral Control Register (PCR). In this mode the SR Interrupt Flag is disabled (held to a logic 0).

MODE 001 - SHIFT IN UNDER CONTROL OF TIMER 2

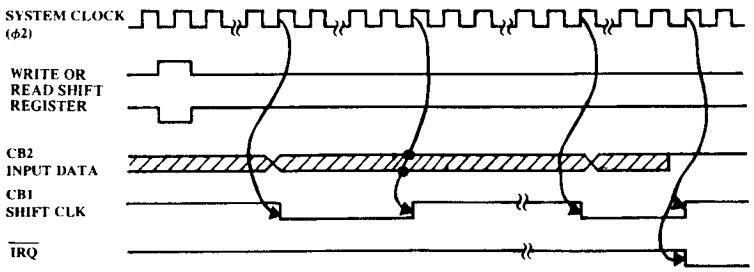
In this mode the shifting rate is controlled by the low-order eight

bits of T2. Shift pulses are generated on the CB1 pin to control shifting in external devices. The time between transitions of this output clock is a function of the system clock period and the contents of the low-order T2 latch.

The shifting operation is triggered by writing or reading the Shift Register. Data are shifted first into the low-order bit of SR and are then shifted into the next-higher-order bit of the Shift Register on the trailing edge of each clock pulse. As shown in Figure 6-10, the input data should change on the negative edge of the clock pulse. These data are loaded into the Shift Register during the system clock cycle following the positive edge of the clock pulse. After eight clock pulses, the Shift Register Interrupt Flag will be set.

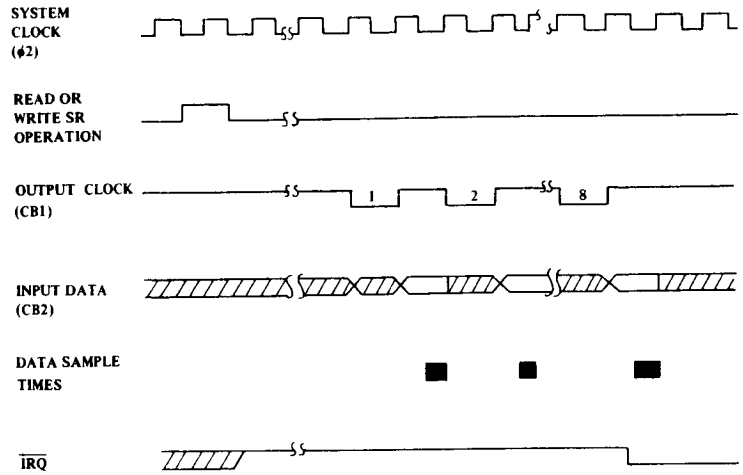
MODE 010 - SHIFT IN AT SYSTEM CLOCK RATE

In this mode the shift rate is a direct function of the system clock frequency. CB1 becomes an output which generates shift pulses for controlling external devices. The shifting operation is triggered by reading or writing the Shift Register. Data are shifted first into bit 0 and are then shifted into the next-higher-order bit of the Shift Register on the positive edge of each clock pulse. After nine clock pulses, the Shift Register Interrupt Flag will be set, and the output clock pulses on CB1 will stop. Figure 6-11 illustrates this timing.



Shifting in Under Control of T2

FIGURE 6-10



Timing Sequence for Shifting in at System Clock Rate

FIGURE 6-11

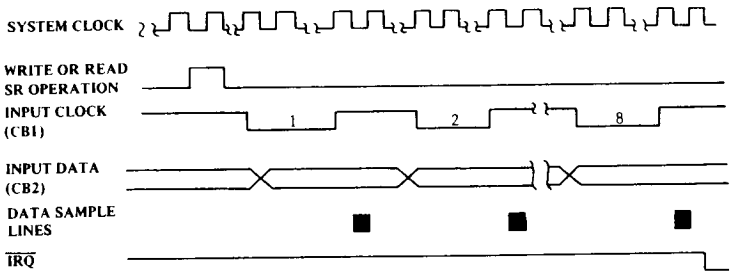
MODE 011 - SHIFT IN UNDER CONTROL OF EXTERNAL CLOCK

In this mode CB1 becomes an input. This allows an external device to load the shift register at its own pace. The shift register counter will interrupt the processor each time 8 bits have been shifted in. However, the shift register counter does not stop the shifting operation; it acts simply as a pulse counter. Reading or writing the Shift Register resets the Interrupt flag and initializes the SR counter to count another eight pulses.

Note that data are shifted during the first system clock cycle following the positive edge of the CB1 shift pulse. For this reason, data must be held stable during the first full cycle following CB1 going high. Timing for this operation is illustrated in Figure 6-12.

SHIFT REGISTER OUTPUT MODES

The four shift register output modes are selected by setting the input/output control bit (ACR4) to a logic 1 and then selecting the specific output mode with ACR3 and ACR2. In each of these modes the shift register shifts data out of bit 7 to the CB2 pin. At the same time the contents of bit 7 are shifted back into bit 0. As in the input modes, CB1 is used either



Timing Sequence for Shifting in Under Control of External Clock

FIGURE 6-12

as an output to provide shifting pulses out or as an input to allow shifting from an external pulse. The four modes are as follows:

ACR4	ACR3	ACR2	Mode
1	0	0	Shift Out - Free-Running Mode. Shift Rate Controlled by T2.
1	0	1	Shift Out - Shift Rate Controlled by T2.
1	1	0	Shift Out at System Clock Rate.
1	1	1	Shift Out Under Control of an External Pulse.

All shift register outputs are set during Ø2 low immediately following the transition (falling) of the shift clock 1. This occurs during Ø2 high.

MODE 100 FREE-RUNNING OUTPUT

This mode is very similar to mode 101 in which the shifting rate is set by T2. However, in mode 100 the SR Counter does not stop the shifting operation. Since the Shift Register bit 7 (SR7) is recirculated back into

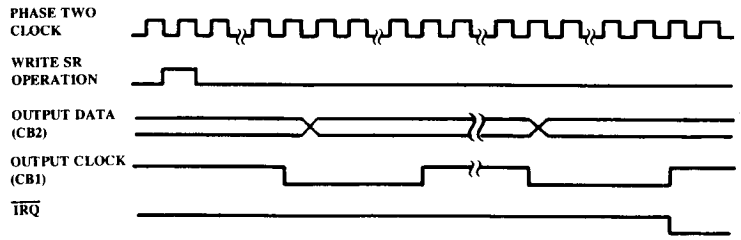
bit 0, the eight bits loaded into the Shift Register will be clocked onto CB2 repetitively. In this mode the shift register counter is disabled.

MODE 101 - SHIFT OUT UNDER CONTROL OF TIMER 2

In this mode the shift rate is controlled by Timer 2. However, with each read or write of the Shift Register the SR Counter is reset and 8 bits are shifted onto CB2. At the same time, eight shift pulses are generated on CB1 to control shifting in external devices. After the eight shift pulses, the shifting is disabled, and the SR Interrupt Flag is set. If the Shift Register is reloaded before the last time-out, the shifting will continue. This sequence is illustrated in Figure 6-13.

MODE 110 - SHIFTING OUT AT SYSTEM CLOCK RATE

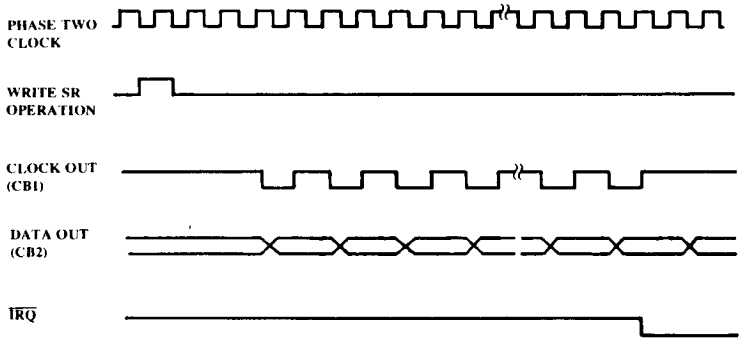
In this mode the shift register operation is similar to that of mode 101. However, the shifting rate is a function of the system clock on the chip enable pin (Ø2) and is independent of T2. Timer 2 resumes its normal function as an independent interval timer. Figure 6-14 illustrates the timing sequence for mode 110.



NOTE: DATA OUT DETERMINED BY CB2 CONTROL IN PCR:

Shifting Out Under Control of T2

FIGURE 6-13



Shifting Out Under Control of System Clock

FIGURE 6-14

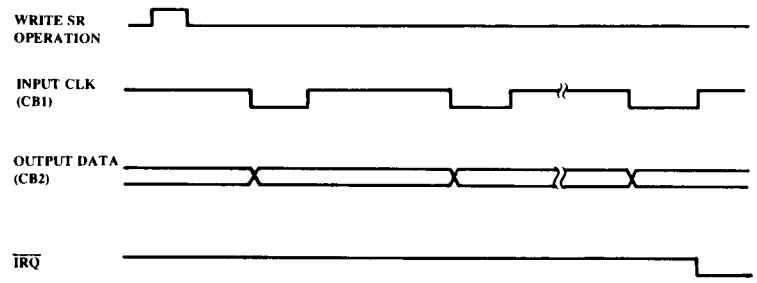
MODE 111 - SHIFT OUT UNDER CONTROL OF AN EXTERNAL PULSE

In this mode, shifting is controlled by pulses applied to the CB1 pin by an external device. The SR counter sets the SR Interrupt flag each time it counts eight pulses but it does not disable the shifting function. Each time the microprocessor writes or reads the shift register, the SR Interrupt flag is reset and the SR Counter is initialized to begin couting the next eight shift pulses on pin CB1. After eight shift pulses, the interrupt flag is set. The microprocessor can then load the shift register with the next byte of data.

Figure 6-15 illustrates the timing sequence for mode 111.

6.4.8 Interrupt Control

Controlling interrupts within the R6522 involves three principal operations: flagging the interrupts, enabling interrupts, and signalling to the processor that an active interrupt exists within the chip. Interrupt flags are set by interrupting conditions which exist within the chip or on inputs to the chip. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the microprocessor must



Shifting Out Under Control of External Clock

FIGURE 6-15

examine these flags in order from highest to lowest priority. This is accomplished by reading the flag reigster into the processor accumulator, shifting this register either right or left and then using conditional branch instructions to detect an active interrupt.

Associated with each interrupt flag is an interrupt enable bit. This bit can be set or cleared by the processor to enable interrupting the processor from the corresponding interrupt flag. If an interrupt flag is set to a logic 1 by an interrupting condition, and the corresponding interrupt enable bit is set to a 1, the Interrupt Request Output ($\overline{IRQ}$) will go low. $\overline{IRQ}$ is an "open-collector" output which can be "wire-or'ed" with other devices in the system to interrupt the processor.

In the R6522, all the interrupt flags are contained in one register. In addition, bit 7 of this register will be read as a logic 1 when an interrupt exists within the chip. This permits very convenient polling of several devices within a system to locate the source of an interrupt.

	7	6	5	4	3	2	1	0
Interrupt Flag Register	IRQ	T1	T2	CB1	CB2	SR	CA1	CA2
Interrupt Enable Register	Set/ clear control	T1	T2	CB1	CB2	SR	CA1	CA2

INTERRUPT FLAG REGISTER

The IFR is a read/limited write register. When the proper chip select and register signals are applied to the chip, the contents of this register are placed on the data bus. Bit 7 indicates the status of the $\overline{IRQ}$ output. This bit corresponds to the logic function: IRQ = IFR6 X IER6 + IFR5 X IER5 + IFR4 X IER4 + IFR3 X IER3 + IFR2 X IER2 + IFR1 X IER1 + IFR0 X IER0.

Note: X = logic AND, + = Logic OR.

Bits six through zero are latches which are set and cleared as follows:

Bit #	Set by	Cleared by
0	Active transition of the signal on the CA2 pin.	Reading or writing the A Port Output Register (ORA) using address 0001.
1	Active transition of the signal on the CA1 pin.	Reading or writing the A Port Output Register (ORA), using address 0001.
2	Completion of eight shifts	Reading or writing the Shift Register.
3	Active transition of the signal on the CB2 pin.	Reading or writing the B Port Output Register.
4	Active transiton of the signal on the CB1 pin.	Reading or writing the B Port Output Register.
5	Time-out of Timer 2.	Reading T2 low order counter or writing T2 high order counter.
6	Time-out of Timer 1.	Reading T1 low order counter or writing T1 high order latch

In addition to the clearing operations shown in the table, individual bits in the IFR can be cleared by writing into the register. A logic 1 in the data word written into the IFR will clear the corresponding interrupt flag. A zero in this word will leave the corresponding flag untouched. Setting the flags occurs only from interrupting conditions within the chip.

The IFR bit 7 is not a flag. Therefore, this bit is not directly cleared by writing a logic 1 into it. It can only be cleared by clearing all the flags in the register or by disabling all the active interrupts as discussed in the next section.

INTERRUPT ENABLE REGISTER (IER)

For each interrupt flag in IFR, there is a corresponding bit in the Interrupt Enable Register (IER). The system processor can set or clear selected bits in this register to facilitate controling individual interrupts without affecting others. If bit 7 of the data placed on the system data bus during the write operation is a 0, each 1 in bits 6 through 0 clears the corresponding bit in the Interrupt Enable Register. For each zero in bits 6 through 0, the corresponding bit is unaffected.

Setting selected bits in the Interrupt Enable Register is accomplished by writing to the same address with bit 7 in the data word set to a logic 1. In this case, each 1 in bits 6 through 0 will set the corresponding bit. For each zero, the corresponding bit will be unaffected. This individual control of the setting and clearing operations allows very convenient control of interrupts during system operation.

In addition to setting and clearing IER bits, the processor can read the contents of this register. Bit 7 will be read as a logic 0.

6.4.9 Function Control

Control of the various functions and operating modes within the R6522 is accomplished primarily through two registers, the Peripheral Control Register (PCR), and the Auxiliary Control Register (ACR). The PCR is used primarily to select the operating mode for the four peripheral control pins. The Auxiliary Control Register selects the operating mode for the interval timers (T1, T2), and the serial port (SR).

PERIPHERAL CONTROL REGISTER

The Peripheral Control Register is organized as follows:

Bit No.	7	6	5	4	3	2	1	0
Function	CB2 Control			CB1 Control	CA2 Control			CA1 Control

Each of these functions is discussed in detail below.

1. CA1 Control

Bit 0 of the Peripheral Control Register selects the active transition of the input signal applied to the CA1 interrupt input pin. If this bit is a logic 0, the CA1 interrupt flag will be set by a negative transition (high to low) of the signal on the CA1 pin. If PCR0 is a logic 1, the CA1 interrupt flag will be set by a positive transition (low to high) of this signal.

2. CA2 Control

The CA2 pin can be programmed to act as an interrupt input or as a peripheral control output. As an input, CA2 operates in two modes, differing primarily in the methods available for resetting the interrupt flag. Each of these two input modes can operate with either a positive or a negative active transition as described above for CA1.

In the output mode, the CA2 will perform either a "Read" or a "write" handshake operation. The CA2 operating modes are selected as follows:

PCR3	PCR2	PCR1	Mode
0	0	0	CA2 Negative Edge Interrupt (IFR0/ORA Clear) Mode -- Set CA2 interrupt flag (IFR0) on a negative transition of the input signal. Clear IFR0 on a read or write of the Peripheral A Output Register (ORA) or by writing logic 1 into IFR0.
0	0	1	CA2 Negative Edge Interrupt (IFR0 Clear) Mode -- Set IFR0 on a negative transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag. Clear IFR0 by writing logic 1 into IFR0.
0	1	0	CA2 Positive Edge Interrupt (IFR0/ORA Clear) Mode -- Set CA2 interrupt flag on a positive transition of the CA2 input signal. Clear IFR0 with a read or write of the Peripheral A Output Register.
0	1	1	CA2 Positive Edge Interrupt (IFR0 Clear) Mode -- Set IFR0 on a positive transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag. Clear IFR0 by writing logic 1 into IFR0.
1	0	0	CA2 Handshake Output Mode -- Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	CA2 Pulse Output Mode -- CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	CA2 Output Low Mode -- The CA2 output is held low in this mode.
1	1	1	CA2 Output High Mode -- The CA2 output is held high in this mode.

In the interrupt-IFR0 clear input mode, writing or reading the ORA register has no effect on the CA2 interrupt flag. This flag must be cleared by writing a logic 1 into the appropriate IFR bit. This mode allows the processor to handle interrupts which are independent of any operations taking place on the peripheral I/O ports.

The handshake and pulse output modes have been described previously. Note that the timing of the output signal varies slightly depending on whether the operation is initiated by a read or a write.

3. CB1 Control

Control of the active transition of the CB1 input signal operates in exactly the same manner as that described above for CA1. If PCR4

is a logic ϕ, the CB1 interrupt flag (IFR4) is a logic 1, the CB1 interrupt flag (IFR4) will be set by a negative transition of the CB1 input signal and cleared by a read or write of the ORB register. If PCR4 is a logic one, IFR4 will be set by a positive transition of CB1.

If the Shift Register function has been enabled, CB1 will act as an input or output for the shift register clock signals. In this mode, the CB1 interrupt flag will still respond to the selected transition of the signal on the CB1 pin.

4. CB2 Control

With the serial port disabled, operation of the CB2 pin is a function of the three high-order bits of the PCR. The CB2 modes are very similar to those described previously for CA2. These modes are selected as follows:

PCR7	PCR6	PCR5	Mode
0	0	0	CB2 Negative Edge Interrupt (IFR3/ORB Clear) Mode -- Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the Peripheral B Output Register (ORB) or by writing logic 1 into IFR3.
0	0	1	CB2 Negative Edge Interrupt (IFR3 Clear) Mode -- Set IFR3 on a negative transition of the CB2 input signal. Reading or writing ORB does not clear the interrupt flag. Clear IFR3 by writing logic 1 into IFR3.
0	1	0	CB2 Positive Edge Interrupt (IFR3/ORB Clear) Mode -- Set CB2 input signal. Clear the CB2 interrupt flag on a read or write of ORB or by writing logic 1 into IFR3.
0	1	1	CB2 Positive Edge Interrupt (IFR3 Clear) Mode -- Set IFR3 on a positive transition of the CB2 input signal. Reading or writing ORB does not clear the CB2 interrupt flag. Clear IFR3 by writing logic 1 into IFR3.
1	0	0	CB2 Handshake Output Mode -- Set CB2 low on a write ORB operation. Reset CB2 high with an active transition of the CB1 input signal.
1	0	1	CB2 Pulse Output Mode -- Set CB2 low for one cycle following a write ORB operation.
1	1	0	CB2 Manual Output Low Mode -- The CB2 output is held low on this mode.
1	1	1	CB2 Manual Output High Mode -- The CB2 output is held high in this mode.

AUXILIARY CONTROL REGISTER

Many of the functions in the auxiliary control register have been discussed previously. However, a summary of this register is presented here as a convenient reference for the R6522 user. The auxiliary control register is organized as follows:

Bit No.	7	6	5	4	3	2	1	0
Function	T1 Control		T2 Control	Shift Register Control			PB Latch Enable	PA Latch Enable

1. PA Latch Enable

The R6522 provides input latching on both the PA and PB ports. In this mode, the data present on the peripheral A input pins will be latched within the chip when the CA1 interrupt flag is set. Reading the PA port will result in these latches being transferrred into the processor. As long as the CA1 interrupt flag is set, the data on the peripheral pins can change without affecting the data in the latches. This input latching can be used with any of the CA2 input or output modes.

It is important to note that on the PA port, the processor always reads the data on the peripheral pins (as reflected in the latches). For output pins, the processor still reads the latches. This may or may not reflect the data currently in the ORA. Proper system operation requires careful planning on the part of the system designer if input latching is combined with output pins on the peripheral ports.

Input latching is enabled by setting bit 0 in the Auxiliary Control Register to a logic 1. As long as this bit is a 0, the latches will directly reflect the data on the pins.

2. PB Latch Enable

Input latching on the PB port is controlled in the same manner as that described for the PA port. However, with the peripheral B port the input latch will store either the voltage on the pin or the contents of the Output Register (ORB) depending on whether the pin

is programmed to act as an input or an output. As with the PA port, the processor always reads the input latches.

3. Shift Register Control

The Shift Register operating mode is selected as follows:

ACR4	ACR3	ACR2	Mode
0	0	0	Shift register disabled.
0	0	1	Shift in under control of Timer 2.
0	1	0	Shift in under control of Ø2 pulses.
0	1	1	Shift in under control of external clock pulses.
1	0	0	Free-running output at rate determined by Timer 2.
1	0	1	Shift out under control of Timer 2.
1	1	0	Shift out under control of the Ø2 pulses.
1	1	1	Shift out under control of external clock pulses.

4. T2 Control

Timer 2 operates in two modes. If ACR5 = 0, T2 acts as an interval timer in the one-shot mode. If ACR5 = 1, Timer 2 acts to count a predetermined number of pulses on pin PB6.

5. T1 Control

Timer 1 operates in the one-shot or free-running mode with the PB7 output control enabled or disabled. These modes are selected as follows:

ACR7	ACR6	Mode
0	0	One-Shot Mode -- Output to PB7 disabled.
0	1	Free-Running Mode -- Output to PB7 disabled.
1	0	One-Shot Mode -- Output to PB7 enabled.
1	1	Free-Running Mode -- Output to PB7 enabled.

6.5 R6522 APPLICATION NOTES

The R6522 represents a significant advance in general-purpose microprocessor I/O. Unfortunately, its many powerful features coupled with a set of very flexible operating modes, cause this device to appear to be very complex at first glance. However, a detailed analysis will show that the VIA is organized to allow convenient control of these powerful features. This section seeks to assist the system designer in his understanding of the R6522 by illustrating how the device can be used in microprocessor-based systems.

6.5.1 Control of R6522 Interrupts

Organization of the R6522 interrupt flags into a single register greatly facilitates the servicing of interrupts from this device. Since there is only one IRQ output for the seven possible sources of interrupt within the chip, the processor must examine these flags to determine the cause of an interrupt. This is best accomplished by first transferring the contents of the flag register into the accumulator. At this time it may be necessary to mask off those flags which have been disabled in the Interrupt Enable Register. This is particularly important for the edge detecting inputs where the flags may be set whether or not the interrupting function has been enabled. Masking of those flags can be accomplished by performing an AND operation between the IER and the accumulator or by performing an "AND IMMEDIATE." The second byte of this AND # instruction should specify those flags which correspond to interrupt functions which are to be serviced.

If the N flag is set after these operations, an active interrupt exists within the chips. This interrupt can be detected with a series of shift and branch instructions.

Clearing interrupt flags is accomplished very conveniently by writing a logic 1 directly into the appropriate bit of the Interrupt Flag Register. This can be combined with an interrupt enable or disable operation as follows:

```
                LDA #%10010000 ; initialize accumulator
                STA IFR        ; clear interrupt flag
                STA IER        ; set interrupt enable flag

or              LDA #%00001000 ; initialize accumulator
                STA IFR        ; clear interrupt flag
                STA IER        ; disable interrupt
```

Another very useful technique for clearing interrupt flags is simply to transfer the contents of the flag register back into this register as follows:

```
LDA IFR ; transfer IFR to accumulator
STA IFR ; clear flags corresponding to active interrupts
```

After completion of this operation the accumulator will still contain the interrupt flag information. Most importantly, writing into the flag register clears only those flags which are already set. This eliminates the possibility of inadvertently clearing a flag while it is being set.

6.5.2 Use of Timer 1

Timer 1 represents one of the most powerful features of the R6522 The ability to generate very evenly spaced interrupts and the ability to control the voltage on PB7 makes this timer particularly valuable in various timing, data detection and waveform generation applications.

TIME-OF-DAY CLOCK APPLICATIONS

An important feature of many systems is the time-of-day clock. In microprocessor-based systems the time of day is usually maintained in memory and is updated in an interrupt service routine. A regular processor interrupt will then assure that this time of day will always be available when it is needed in the main program.

Generating very regular interrupts using previously available timers presented difficulties because of the need to reload the timer for each interrupt. Unfortunately, the time between the interrupts will fluctuate due to variations in the interrupt response time. This problem is eliminated in the Timer 1 "free-running" mode. The accuracy of these "free-running" interrupts is only a function of the system clock and is not affected by interrupt response time.

ASYNCHRONOUS DATA DETECTION

The extraction of clock and data information from serial asynchronous ASCII signals or from any single channel data recording device relies on the ability to establish accurate strobes. As discussed previously, the period of these strobes can be seriously affected by the interrupt response time using conventional timers. However, T1 again allows generation of very accurate interrupts. The processor responds to these interrupts by strobing

the input data. The ability to reload the T1 latches without affecting the count-down in progress is very useful in this application. This allows the strobe time to be doubled or halved during data detection.

Figure 6-16 is an example of the use of this timer with asynchronous serial data.

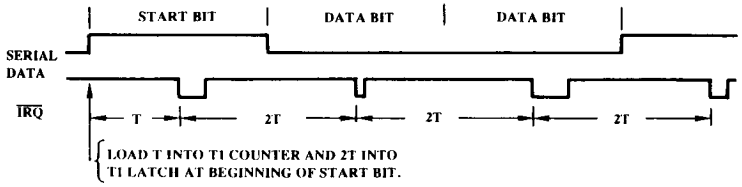
WAVEFORM GENERATION WITH TIMER 1

In addition to generating processor interrupts, Timer 1 can be used to control the output voltage on peripheral pin PB7 (output mode). In this mode a single negative pulse can be generated on PB7 (one-shot mode) or, in the free-running mode, a continuous waveform can be generated. In this latter mode the voltage on PB7 will be inverted each time T1 times out.

A single solenoid can be triggered by simply writing to T1C-H in the one-shot mode if the PB7 signal is used to control the solenoid directly.

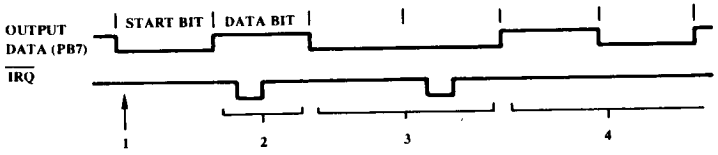
Generating very complex waveforms can be a simple problem if T1 is used to control PB7 in the free-running mode. During any count-down process the latches can be loaded to determine the length of the next count-down period.

Figure 6-17 shows this timing sequence for generating ASCII serial data.



Asynchronous Data Detection Using Timer 1

FIGURE 6-16



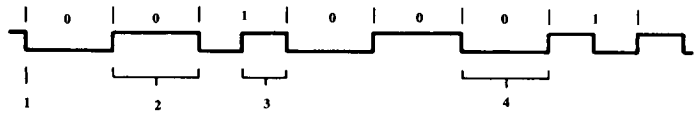
1. LOAD T INTO T1 COUNTER AND LATCH. LOAD T INTO T2 TO TRIGGER T1 LATCH RELOAD.
2. LOAD 2T INTO T1 LATCH DURING THIS BIT TIME.
3. LOAD T INTO T1 LATCH ANYTIME DURING THIS PERIOD. LOAD NT INTO T2. N = NUMBER OF 1'S AND 0'S WHICH FOLLOW.
4. A SERIES OF 1'S AND 0'S WILL BE GENERATED UNTIL THE T1 LATCH IS AGAIN CHANGED. NOTE THAT THE USE OF T2 TO CONTROL RELOADING THE T1 LATCH ELIMINATES THE NEED TO INTERRUPT ON EACH TRANSITION.

ASCII Serial Data Generation Using T1

FIGURE 6-17

An application where this mode of operation is also very powerful is in the generation of biphase encoded data for tape or disk storage. This encoding technique and the sequence of operations which would take place illustrated in Figure 6-18.

These applications represent only a tiny portion of the potential T1 applications. Some other possibilities are pulse width modulation waveforms, sound generation for video games, and A/D techniques requiring very accurate pulse widths.



1. LOAD T1 COUNTER AND LATCH.
2. SHIFT T1 LATCH ONE BIT TO THE RIGHT DURING THIS PERIOD.
3. SHIFT T1 LATCH LEFT DURING THIS PERIOD.
4. SHIFT T1 LATCH RIGHT DURING THIS PERIOD.

NOTE THAT T1 MUST BE ACCESSED ONLY WHEN THE OUTPUT DATA CHANGES. A STRING OF 1'S OR 0'S CAN BE GENERATED WITHOUT PROCESSOR INTERVENTION.

Generating Biphase Encoded Data

FIGURE 6-18

SECTION 7

R6530 ROM — RAM — I/O TIMER (RRIOT)

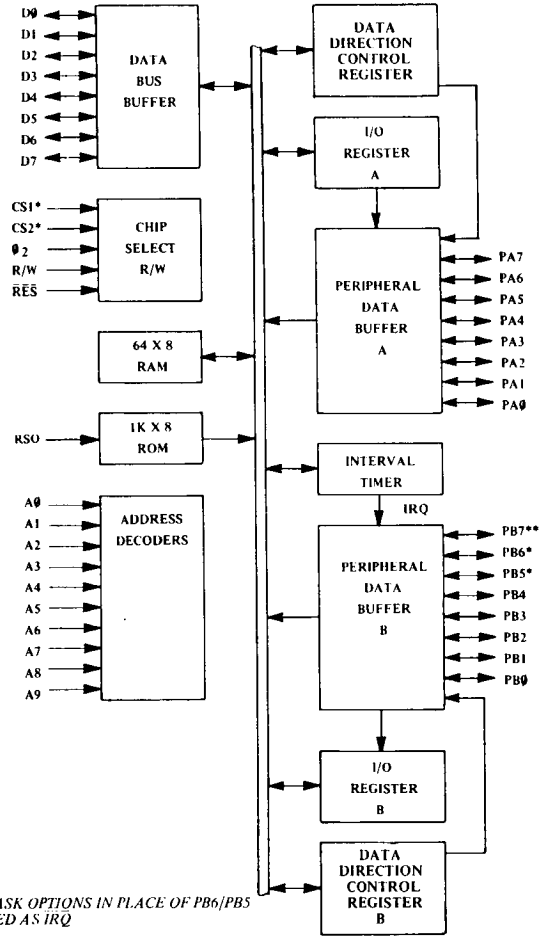
The R6530 is designed to operate in conjunction with the R650X Microprocessor (CPU). It is comprised of a mask-programmable 1024 x 8 ROM; a 64 x 8 RAM; two 8-bit bidirectional ports, capable of directly interfacing the Microprocessor unit and peripheral devices; and a programmable interval timer with interrupt, capable of timing in various intervals from 1 to 262,144 clock periods.

The I/O configuration, the interval timer and interrupt capability are under software control.

- 8-bit bidirectional Data Bus for communication with the microprocessor unit
- Two 8-bit bidirectional ports for direct interface to peripherals
- Two Programmable Data Direction Registers
- Programmable Interval Timer from 1 to 256 x 1024 clock periods.
- Programmable Interval Timer Interrupt
- TTL and CMOS-Compatible Peripheral Lines
- Peripheral Pins with Direct Transistor Drive Capability
- Three-State Data Pins
- Up to 7K contiguous ROM with no external decoding
- 1024 x 8 ROM
- 64 x 8 Static RAM

7.1 R6530 ORGANIZATION

A block diagram of the internal architecture is shown in Figure 7-1 The R6530 is divided into four basic sections, RAM, ROM, I/O and TIMER. The RAM and ROM interface directly with the microprocessor through the system data bus and address lines. The I/O section consists of two 8-bit halves. Each half contains a Data Direction Register (DDR) and an I/O Register. The DDR controls the peripheral output buffers. A "1" written into the DDR sets up the corresponding peripheral buffer as an output buffer -- that is,



**CS1/CS2 ARE MASK OPTIONS IN PLACE OF PB6/PB5*
***PB6 MAY BE USED AS IRQ*

R6530 Internal Architecture

FIGURE 7-1

anything then written into the I/O Register will appear on that corresponding peripheral pin. A "0" written into the DDR inhibits the output buffer from transmitting data from the I/O Register. The output buffer remains in the high state, making it ready to receive data on the peripheral lines.

It should be noted that the microprocessor, when reading the I/O Register, is in fact reading the Peripheral Pin and not the I/O Register. The only way the I/O Register data can be changed is by a microprocessor Write operation. The Register is not affected by the data on the Peripheral Pin.

7.1.1 ROM -- 1K Byte (8K Bits)

The 8K ROM is in a 1024 x 8 configuration. Address lines A0-A9, as well as RSØ are needed to address the entire ROM. With the addition of CS1 and CS2, up to seven R6530s may be addressed, giving 7168 x 8 bits of contiguous ROM.

7.1.2 RAM -- 64 Bytes (512 Bits)

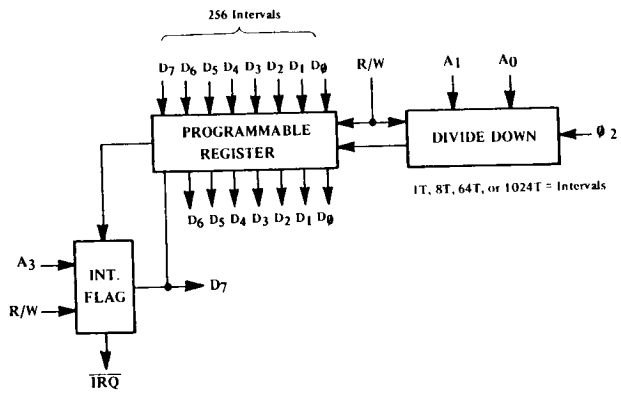
A 64 x 8 static RAM is contained on the R6530. It is addressed by A0-A5 (Byte Select), RS0, A6, A7, A8, A9 and, depending on the number of chips in the system, CS1 and CS2.

7.1.3 Internal Peripheral Registers

There are four internal registers, two data direction registers and two output registers. The two data direction registers (A side and B side) control the direction of data into and out of the peripheral pins. For example, a "1" loaded into data direction register A, position 3 sets up peripheral pin PA3 as an output. If a "0" had been loaded instead, PA3 would be configured as an input. The two output registers are used to latch data from the data bus during a Write operation until the peripheral device can read the data supplied by the microprocessor unit. Although during a Read operation the microprocessor unit reads the peripheral pin, the address is the same as the register. For those pins programmed as outputs by the data direction registers, the data on the pins will be the same as in the I/O register.

7.1.4 Interval Timer

The Timer section of the R6530 contains three basic parts: preliminary divide down register, programmable 8-bit register and interrupt logic. These are illustrated in Figure 7-2.



Basic Elements of Interval Timer

FIGURE 7-2

The interval timer can be programmed to count up to 256 time intervals. Each time interval can be either 1T, 8T, 64T or 1024T increments, where T is the system clock period. When a full count is reached, an interrupt flag is set to a logic "1." After the interrupt flag is set the internal clock begins counting down to a maximum of -255T. Thus, after the interrupt flag is set, a Read of the timer will tell how long since the flag was set up to a maximum of 255T.

When writing to the timer, the high-order 8 bits of the timer are written by the system data bus. If a count of 52 time intervals were to be counted, 0 0 1 1 0 1 0 0 would be written into the timer section. The time intervals of 1, 8, 64 or 1024T are decoded from address lines AØ and A1 at this same time. Address line A3, if high during this write operation, enables the interrupt flag onto pin PB7. PB7 should be programmed as an input if it is to be used as a interrupt pin. PB7 goes low when an interrupt occurs. When the timer is read prior to the interrupt flag being set, the number of time intervals remaining will be read, i.e., 51, 50, 49, etc.

Should the timer be read when interrupt occurs, the value read would be 1 1 1 1 1 1 1 1. After interrupt, the timer register decrements at a divide by "1" rate of the system clock. If, after interrupt, the timer is read and a value of 1 1 1 0 0 1 0 0 is read, the time since interrupt is 28T. The value read is in two's complement.

Value read = 1 1 1 0 0 1 0 0
Complement = 0 0 0 1 1 0 1 1
ADD 1 = 0 0 0 1 1 1 0 0 = 28.

Thus, to arrive at the total elapsed time, merely do a two's complement add to the original time written into the timer. Again, assume time written as 0 0 1 1 0 1 0 0 (= 52). With a divide by 8, total time to interrupt is (52 x 8) + 1 = 417T. Total elapsed time would be 416T + 28T = 444T, assuming the value read after interrupt was 1 1 1 0 0 1 0 0.

After interrupt, whenever the timer is written or read the interrupt is reset. However, the reading or writing of the timer at the same time interrupt occurs will not reset the interrupt flag.

Figure 7-3 illustrates an example of interrupt.

When reading the timer after an interrupt, A3 should be low to disable the $\overline{IRQ}$ pin. This is done to avoid future interrupts until after another Write timer operation.

7.2 INTERFACE LINES

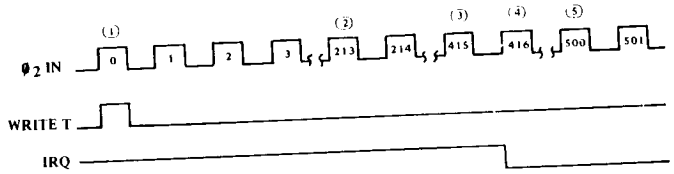
Figure 7-4 is the pinout diagram of the R6530.

7.2.1 Reset (RES)

During system initialization a Logic "0" on the $\overline{RES}$ input will cause a zeroing of all I/O registers. This in turn will cause all I/O buses to act as inputs, thus protecting external components from possible damage and erroneous data while the system is being configured under software control. The Data Bus Buffers are put into an OFF-STATE during Reset. Interrupt is disabled when reset. The $\overline{RES}$ signal must be held low for at least one clock period when reset is required.

7.2.2 Input Clock

The input clock is a system Phase Two clock which can be either a low-level clock ($V_{IL} < 0.4$, $V_{IH} > 2.4$) or a high-level clock ($V_{IL} < 0.2$, $V_{IH} = Vcc \begin{smallmatrix}+0.3\\-0.2\end{smallmatrix}$).



SHOULD THE PROGRAMMABLE TIMER REGISTER BE READ AT THE TIMES NOTED ON THE DIAGRAM ABOVE, IT WOULD CONTAIN:

(1) Data written into interval timer is $00110010 = 52_{10}$ A divide by 8 pre scale is used.
(2) $00011001 = 25_{10}$ $52 - \frac{213}{8} - 1 = 52 - 26 - 1 = 25$
(3) $00000000 = 0_{10}$ $52 - \frac{8 \cdot 415}{8} - 1 = 52 - 51 - 1 = 0$
(4) Interrupt has occurred at Ø2 pulse #416
(5) 10101100 Two's complement = $01010100 = 84_{10}$ $84 + (52 \times 8) = 500_{10}$

Example of Interrupt Generated by Interval Timer

FIGURE 7-3

7.2.3 Read/Write (R/W)

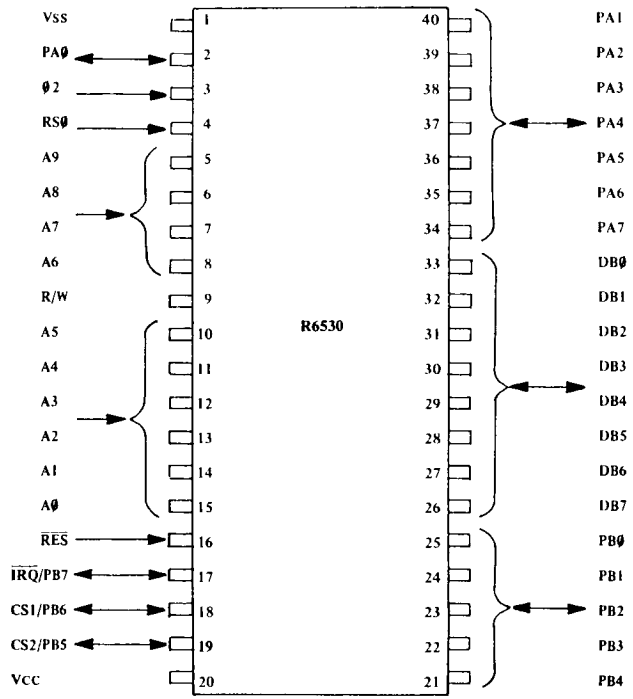
The R/W signal is supplied by the microprocessing unit and is used to control the transfer of data to and from the microprocessing unit and the R6530. A high on the R/W pin allows the processor to read (with proper addressing) the data supplied by the R6530. A low on the R/W pin allows a write (with proper addressing) to the R6530.

7.2.4 Interrupt Request (IRQ)

The $\overline{IRQ}$ pin is an interrput pin from the interval timer. This same pin, if not used as an interrupt, can serve as a peripheral I/O pin (PB7). When used as an interrupt, the pin should be set up as an input by the data direction register. The pin will be normally high with a low indicating an interrupt from the R6530. An external pull-up device is not required; however, if collector-OR'd with other devices, the internal pull-up may be omitted with a mask option.

7.2.5 Data Bus (D0-D7)

The R6530 has eight bidirectional data pins (D0-D7). These pins connect to the system's data lines and allow transfer of data to and from



R6530 Pinout Designation

FIGURE 7-4

the microprocessor unit. The output buffers remain in the "off" state except when a Read operation occurs.

7.2.6 Peripheral Data Ports

The R6530 has 16 pins available for peripheral I/O operations. Each pin is individually software programmable to act as either an input or an output. The 16 pins are divided into two 8-bit ports, PA0-PA7 and PB0-PB7. PB5, PB6 and PB7 also have other uses which will be discussed in Section 7.3. The pins are set up as an input by writing a "0" into the corresponding bit in the data direction register. A "1" into the data direction register will cause its corresponding bit to be an output. When in the input mode, the peripheral output buffers are in the "1" state and a pull-up device acts as less than one TTL load to the peripheral data lines. On a Read operation, the microprocessor unit reads the peripheral pin. When the peripheral device gets information from the R6530 it receives data stored in the data register. The microprocessor will read correct information if the peripheral lines are greater than 2.0 volts for a "1" and less than 0.8 volt for a "0" as the peripheral pins are all TTL compatible. Pins PA0 and PB0 are also capable of sourcing 3 ma at 1.5V, thus making them capable of Darlington drive.

7.2.7 Address Lines (A0-A9)

There are 10 address pins. In addition to these 10, there is the ROM SELECT pin. The above pins, A0-A9 and ROM SELECT, are always used as addressing pins. There are two additonal pins which are mask-programmable and can be employed either individually or together as CHIP SELECTs. They are pins PB5 and PB6. When used as peripheral data pins they cannot be used as chip selects.

7.3 ADDRESSING

Addressing of the R6530 offers many variations to the user for greater flexibility. The user may configure his system with RAM in lower memory, ROM in higher memory, and I/O registers with interval timers between the extremes. There are 10 address lines (A0-A9). In addition, there is the possibility of three additional address lines to be used as chip-selects and to distinguish between ROM, RAM, I/O and interval timer. Two of the additional lines are chip-selects 1 and 2 (CS1 and CS2). The chip-select pins can also be PB5 and PB6. Whether the pins are used as chip-selects or peripheral I/O pins is a mask option and must be specified when ordering

the part. Both pins act independently of each other in that either or both pins may be designated as a chip-select. The third additional address line is RS0. The R6502 and R6530 in a 2-chip system would use RS0 to distinguish between ROM and non-ROM sections of the R6530. With the addressing pins available, a total of 7K contiguous ROM may be addressed with no external decode. Below is an example of a 1-chip and a 7-chip R6530 Addressing Scheme.

7.3.1 One-Chip Addressing

Figure 7-5 illustrates a 1-chip system decode for the R6530.

7.3.2 Seven-Chip Addressing

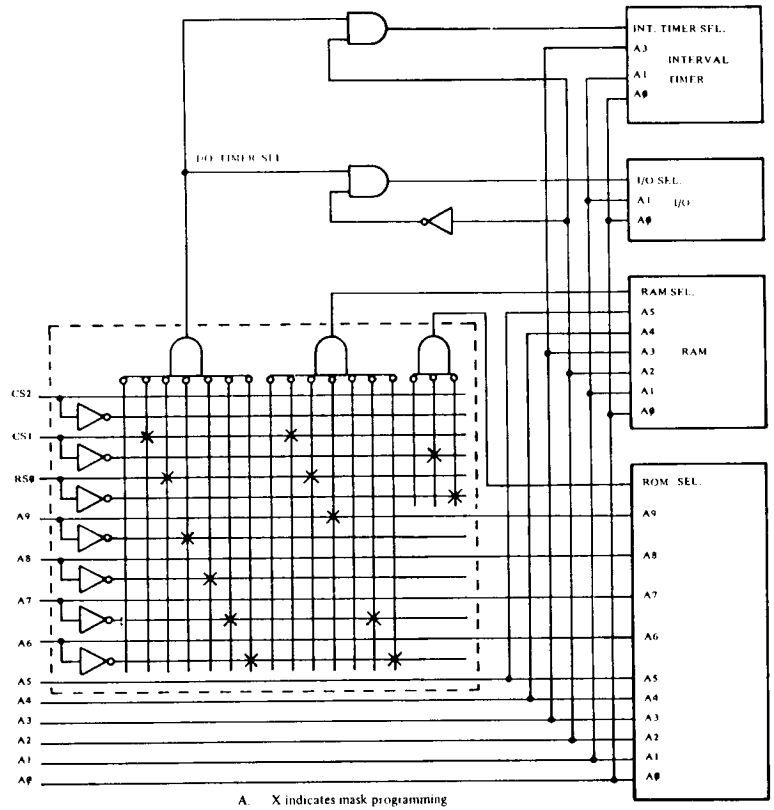
In the 7-chip system the objective would be to have 7K of contiguous ROM, with RAM in low order memory. The 7K of ROM could be placed between addresses 65,536 and 1024. For this case, assume A13, A14 and A15 are all 1 when addressing ROM, and 0 when addressing RAM or I/O. This would place the 7K ROM between Addresses 65,535 and 57,367. The 2 pins designated as chip-select or I/O would be masked programmed as chip-select pins. Pin RS0 would be connected to address line A10. Pins CS1 and CS2 would be connected to address lines A11 and A12, respectively. (See Table 7-1).

The two examples given would permit addressing of the ROM and RAM; however, once the I/O timer has been addressed, further decoding is necessary to select which of the I/O registers are desired, as well as the coding of the interval timer.

7.3.3 I/O Register -- Timer Addressing

Table 7-2 illustrates the addressing decoding for the internal elements and timer programming. Address line A2 distinguishes I/O registers from the timer. When A2 is low and I/O timer select is high, the I/O registers are addressed. Once the I/O registers are addressed, address lines A1 and A0 decode the desired register.

When the timer is selected, A1 and AØ decode the divide by matrix. This is discussed further in the Timer Selection. In addition, Address A3 is used to enable the interrupt flag to PB7.



A. X indicates mask programming
i.e. ROM select = $\overline{CS1}$•RSO
RAM select = $\overline{CS1}$•$\overline{RSO}$•A9•A7•A6
I/O TIMER SELECT = $\overline{CS1}$•RSO•A9•A8•A7•A6

B. Notice that A8 is a "don't care" for
RAM select

C. CS2 can be used as PB5 in this example.

R6530 One Chip Address Encoding Diagram
FIGURE 7-5

The addressing of the ROM select, RAM select and I/O Timer select lines would be as follows:

TABLE 7-1
Addressing Decode for I/O Register and Timer

		CS2 A12	CS1 A11	RSØ A10	A9	A8	A7	A6
R6530 #1,	ROM SELECT	0	0	1	X	X	X	X
	RAM SELECT	0	0	0	0	0	0	0
	I/O TIMER	0	0	0	1	0	0	0
R6530 #2,	ROM SELECT	0	1	0	X	X	X	X
	RAM SELECT	0	0	0	0	0	0	1
	I/O TIMER	0	0	0	1	0	0	1
R6530 #3,	ROM SELECT	0	1	1	X	X	X	X
	RAM SELECT	0	0	0	0	0	1	0
	I/O TIMER	0	0	0	1	0	1	0
R6530 #4,	ROM SELECT	1	0	0	X	X	X	X
	RAM SELECT	0	0	0	0	0	1	1
	I/O TIMER	0	0	0	1	0	1	1
R6530 #5,	ROM SELECT	1	0	1	X	X	X	X
	RAM SELECT	0	0	0	0	1	0	0
	I/O TIMER	0	0	0	1	1	0	0
R6530 #6,	ROM SELECT	1	1	0	X	X	X	X
	RAM SELECT	0	0	0	0	1	0	1
	I/O TIMER	0	0	0	1	1	0	1
R6530 #7,	ROM SELECT	1	1	1	X	X	X	X
	RAM SELECT	0	0	0	0	1	1	0
	I/O TIMER	0	0	0	1	1	1	0

* RAM select for R6530 #5 would read = $\overline{A12}$•$\overline{A11}$•$\overline{A10}$•$\overline{A9}$•A8•$\overline{A7}$•$\overline{A6}$

TABLE 7-2

R6530 Seven-Chip Addressing Scheme

	ROM SELECT	RAM SELECT	I/O TIMER SELECT	R/W	A3	A2	A1	A0
READ ROM	1	0	0	1	X	X	X	X
WRITE RAM	0	1	0	0	X	X	X	X
READ RAM	0	1	0	1	X	X	X	X
WRITE DDRA	0	0	1	0	X	0	0	1
READ DDRA	0	0	1	1	X	0	0	1
WRITE DDRB	0	0	1	0	X	0	1	1
READ DDRB	0	0	1	1	X	0	1	1
WRITE PER. REG. A	0	0	1	0	X	0	0	0
READ PER. REG. A	0	0	1	1	X	0	0	0
WRITE PER. REG. B	0	0	1	0	X	0	1	0
READ PER. REG. B	0	0	1	1	X	0	1	0
WRITE TIMER								
÷ 1T W/IRQ to PB7	0	0	1	0	1	1	0	0
÷ 8T WO/IRQ to PB7	0	0	1	0	0	1	0	1
÷ 64T W/IRQ to PB7	0	0	1	0	1	1	1	0
÷ 1024T WO/IRQ to PB7	0	0	1	0	0	1	1	1
READ TIMER DISABLE IRQ TO PB7	0	0	1	1	0	1	X	0
READ INTERRUPT FLAG	0	0	1	1	X	1	X	1

SECTION 8

R6532 RAM — I/O TIMER (RIOT)

The R6532 is designed to operate in conjunction with the R6500 Microcomputer System's microprocessor (CPU) family. It is comprised of a 128 x 8 static RAM; two software-controlled, 8-bit bidirectional data ports allowing direct interfacing between the microprocessor unit and peripheral devices; a software programmable interval timer with interrupt, capable of timing in various intervals from 1 to 262,144 clock periods; and a programmable edge-detecting circuit.

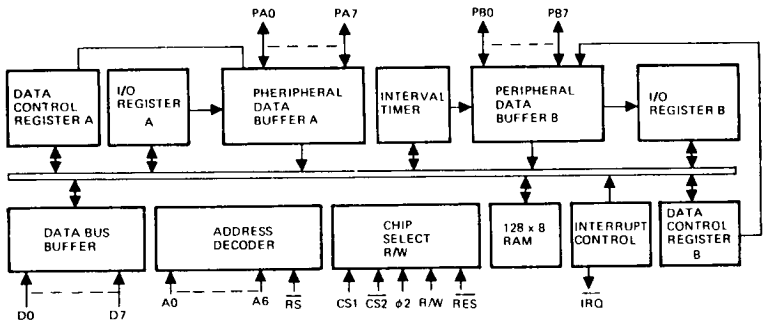
- 8-bit bidirectional Data Bus for direct communication with the microprocessor
- 128 x 8 static RAM
- Two 8 bit bidirectional data ports for interface to peripherals
- Two programmable Data Direction Registers
- Programmable Interval Timer with Interrupt
- TTL & CMOS compatible peripheral lines
- Peripheral pins with Direct Transistor Drive Capability
- High Impedance Three-State Data Pins
- Programmable edge-sensitive interrupt

8.1 R6532 ORGANIZATION

A block diagram of the internal architecture is presented in Figure 8-1. The R6532 is divided into four basic sections, RAM, I/O, Timer, and Interrupt Control. The RAM interfaces directly with the microprocessor through the system data bus and address lines. The I/O section consists of two 8-bit halves, each of which contains a Data Direction Register (DDR) and an I/O Register.

8.1.1 RAM - 128 Bytes (1024 Bits)

The 128 x 8 Read/Write memory acts as a conventional static RAM. Data can be written into the RAM from the microprocessor by selecting the chip (CS1 = 1, $\overline{CS2}$ = Ø) and by setting $\overline{RS}$ to a logic Ø (0.4V). Address lines A0 through A6 are then used to select the desired byte of storage.



R6532 Internal Architecture

FIGURE 8-1

8.1.2 Internal Peripheral Registers

The Peripheral A I/O port consists of eight lines which can be individually programmed to act as either an input or an output. A logic 0 in a bit of the Data Direction Register (DDRA) causes the corresponding line of the PA port to act as an input. A logic 1 causes the corresponding PA line to act as an output. The voltage on any line programmed to be an output is determined by the corresponding bit in the Output Register (ORA).

Data are read directly from the PA pins during any read operation. For any output pin, the data transferred into the processor will be the same as those contained in the Output Register if the voltage on the pin is allowed to go to 2.4V for a logic one. Note that for input lines, the processor can write into the corresponding bit of the Output Register. This will not affect the polarity of the pin until the corresponding bit of DDRA is set to a logic 1 to allow the peripheral pin to act as an output.

The operation of the Peripheral B Input/Output port is exactly the same as the normal I/O operation of the Peripheral A port. The eight lines can each be programmed to act as either an input or as an output by placing a 0 or a 1 into the Data Direction register (DDRB). In the output mode, the voltage on a peripheral pin is controlled by the Output Register (ORB).

The primary difference between the PA and the PB ports is in the operation of the output buffers which drive these pins. The buffers are push-pull devices which are capable of sourcing 3 ma at 1.5V. This allows these pins to directly drive transistor switches. To ensure that the microprocessor will read proper data on a "Read PB" operation, sufficient logic is provided in the chip to allow the microprocessor to read the Output Register instead of reading the peripheral pin as on the PA port.

8.1.3 Edge-Detecting Interrupt

In addition to acting as a peripheral I/O line, the PA7 line can serve as an edge-detecting input. In this mode, an active transition will set the internal interrupt flag (bit 6 of the Interrupt Flag register). Setting the interrupt flag will cause IRQ output to go low if the PA7 interrupt has been enabled.

Control of the PA7 edge-detecting mode is accomplished by writing to one of four addresses. In this operation, A0 controls the polarity of the active transition and A1 acts to enable to disable interrupting of the processor. The data placed on the Data Bus during this operation are discarded and have no effect on the control of PA7.

Setting of the PA7 interrupt flag will occur on an active transition even if the pin is being used as a normal input or as a peripheral control output. The flag will also be set by an active transition if interrupting from PA7 is disabled. The reset signal ($\overline{RES}$) will disable the PA7 interrupt and will set the active transition to negative (high to low). During the system initialization routine, it is possible to set the interrupt flag by a negative transition. It may also be set by changing the polarity of the active interrupt. It is therefore recommended that the interrupt flag be cleared before enabling interrupting from PA7.

Clearing of the PA7 Interrupt Flag occurs when the microprocessor reads the Interrupt Flag Register.

8.1.4 Interval Timer

The Timer section (Figure 8-2) of the R6532 contains a preliminary divide-down register, a programmable 8-bit register, and interrupt logic.

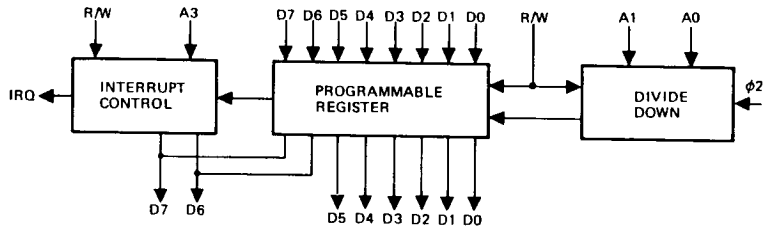
The Interval Timer can be programmed to count up to 255 time intervals. Each time interval can be either 1T, 8T, 64T or 1024T increments, where T is the system clock period. When a full count is reached, an interrupt flag is set to a logic 1. After the interrupt flag is set the internal clock begins counting down to a maximum of -255T. Thus, after the interrupt flag is set, a Read of the Timer will tell how long since the flag was set up to a maximum of 255T.

The 8-bit system Data Bus is used to transfer data to and from the Interval Timer. If a count of 52 time intervals were to be counted, the pattern 0 0 1 1 0 1 0 0 would be put on the Data Bus and written into the Interval Time register.

At the same time that data are being written to the Interval Timer, the counting intervals of 1, 8, 64, 1024T are decoded from address lines A0 and A1. During a Read or Write operation address line A3 controls the interrupt capability of PB7, i.e. A_3 = 1 enables $\overline{IRQ}$, A_3 = 0 disables $\overline{IRQ}$. When the the timer is read prior to the interrupt flag being set, the number of time intervals remaining will be read, i.e., 51, 50, 49, etc.

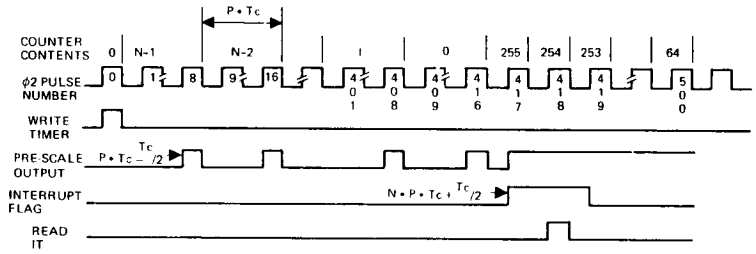
When the timer has counted thru 0 0 0 0 0 0 0 0 on the next count time an interrupt will occur and the counter will read 1 1 1 1 1 1 1 1. After interrupt, the timer register decrements at a divide by "1" rate of the system clock. If after interrupt, the timer is read and a value of 1 1 1 0 0 1 0 0 is read, the time since interrupt is 27T. The value read is two's complement, but it should be remembered that interrupt occurred on count number -1 and we must, therefore, subtract 1:

Value read = 1 1 1 0 0 1 0 0
Complement = 0 0 0 1 1 0 1 1
ADD 1 = 0 0 0 1 1 1 0 0 = 28 Equals two's complement of register
SUB 1 = 0 0 0 1 1 0 1 1 = 27



Basic Elements of Interval Timer

FIGURE 8-2



ASSUME 52 LOADED INTO TIMER WITH A DIVIDE BY 8.
THE COUNTER CONTENTS AND THE CLOCK PULSE NUMBERS WILL COINCIDE.

Interval Timer Example

FIGURE 8-3

Thus, to arrive at the total elapsed time, one merely carries out a two's complement add to the original time written into the timer. Again, time is to be assumed to be written as 0 0 1 1 0 1 0 0 (=52). With a divide by 8, total time to interrupt is (52 x 8) + 1 = 417T. Total elapsed time would be 417T + 27T = 444T, assuming the value read after interrupt was 1 1 1 0 0 1 0 0.

After the interrupt, whenever the timer is written or read the interrupt is reset. However, the reading of the timer at the same time the interrupt occurs will not reset the interrupt flag. When the interrupt flags are read (DB7 for the timer, DB6 for edge detect) data bus lines D0-D5 go to zero.

When reading the timer after an interrupt, A3 should be low tc disable the $\overline{IRQ}$ pin. This is done to avoid future interrupts until after another Write timer operation.

8.2 INTERFACE LINES

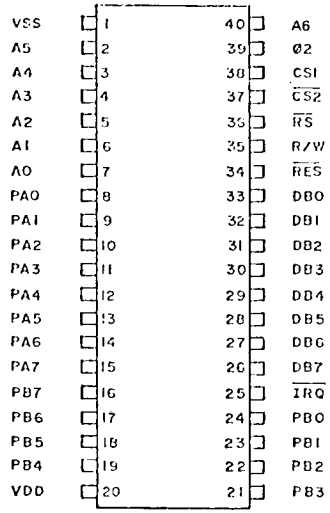
Figure 8-4 is the pinout diagram of the R6532

8.2.1 Peset (RES)

During system initialization a Logic "0" on the RES input will cause a zeroing of all four I/O registers. This, in turn, will cause all I/O buses to act as inputs, thus protecting external components from possible damage and erroneous data while the system is being configured under software control. The Data Bus Buffers are put into an OFF-STATE during Reset. Interrupt capability is disabled with the $\overline{RES}$ signal. The $\overline{RES}$ signal must be held low for at least one clock period when reset is required.

8.2.2 Input Clock

The input clock is a system Phase 2 clock which can be either a low-level clock (V_{IL} < 0.4, V_{IH} > 2.4) or a high-level clock (V_{IL} < 0.2, V_{IH} = Vcc $^{+0.3}_{-0.2}$).



R6532 Pinout Designation

FIGURE 8-4

8.2.3 Read/Write (R/W)

The R/W signal is supplied by the microprocessor array and is used to control the transfer of data to and from the microprocessor array and the R6532. A high on the R/W pin allows the processor to read (with proper addressing) the data supplied by the R6532. A low on the R/W pin allows a write (with proper addressing) to the R6532.

8.2.4 Interrupt Request (IRQ)

The $\overline{IRQ}$ pin is an interrupt pin from the interrupt control logic. The pin will be normally high, with a low indicating an interrupt from the R6532. An external pull-up device is required. The $\overline{IRQ}$ pin may be activated by a transition on PA7 or timeout of the interval timer.

8.2.5 Data Bus (D0-D7)

The R6532 has eight bidirectional data pins (D0-D7). These pins connect to the system's data lines and permit transfer of data to and from the microprocessor array. The output buffers remain in the "off" state except when a Read operation occurs.

8.2.6 Peripheral Data Ports

The R6532 has 16 pins available for peripheral I/O operations. Each pin is individually software-programmable to act as either an input or an output. The 16 pins are divided into two 8-bit ports, PA0-PA7 and PB0-PB7. PB7 also has other uses which are discussed elsewhere. The pins are set up as an input by writing a "0" into the corresponding bit of the data direction register. A "1" into the data direction register will cause its corresponding bit to be an output. When in the input mode, the peripheral output buffers are in the "1" state and a pull-up device acts as less than one TTL load to the peripheral data lines. On a Read operation, the microprocessor unit reads the peripheral pin. When the peripheral device gets information from the R6532 it receives data stored in the data register. The microprocessor will read correct information if the peripheral lines are greater than 2.0 volts for a "1" and less than 0.8 volt for a "0" as the peripheral pins are all TTL compatible. Pins PB0-PB7 are also capable of sourcing 3 ma at 1.5V, thus making them capable of Darlington drive.

8.2.7 Address Lines (A0-A6)

There are 7 address pins. In addition to these 7, there is the $\overline{\text{RAM SELECT}}$ pin. The above pins, A0-A6 and $\overline{\text{RAM SELECT}}$, are always used as addressing pins. There are two additional pins which are used as CHIP SELECTs. They are pins CS1 and $\overline{CS2}$.

8.3 ADDRESSING

Table 8-1 summarizes addressing decoding for the R6532.

TABLE 8-1

Addressing Decode for R6532

RAM ADDRESSING

$\overline{RS}$ = 0
RW = 1 to read, 0 to write
AØ-A6 select RAM address

I/O ADDRESSING

$\overline{RS}$ = 1 A2 = 0
RW = 1 to read, 0 to write

	A1	AØ
PA data	0	0
PA data direction	0	1
PB data	1	0
PB data direction	1	1

WRITE EDGE-DETECTION CONTROL

$\overline{RS}$, A2 = 1 R/W, A4 = 0

A1 = 1, enable interrupt from PA7
A1 = 0, disable interrupt from PA7
A0 = 1, positive edge detect
A0 = 0, negative edge detect

READ AND CLEAR INTERRUPT FLAG

$\overline{RS}$, R/W, A2, AØ = 1

Bit 7 = Timer Flag
Bit 6 = PA7 Flag

WRITE COUNT TO INTERVAL TIMER

$\overline{RS}$, A4, A2 = 1 R/W = 0

	A1	AØ
: 1	0	0
: 8	0	1
: 64	1	0
: 1024	1	1

A3 = 1 enable timer interrupt
A3 = 0 disable timer interrupt

NOTE: For all operations CS1 = 1, $\overline{CS2}$ = 0.

APPENDIX A

SUMMARY OF SINGLE-CYCLE EXECUTION

This section contains an outline of the data on both the address bus and the data bus for each cycle of the various processor instructions. It tells the system designer exactly what to expect while single-cycling through a program.

Note that the processor will not stop in any cycle where R/W is a 0 (WRITE cycle). Instead, it will go right into the next READ cycle and stop there. For this reason, some instructions may appear to be shorter than indicated here.

All instructions begin with T0 and the fetch of the OP CODE and continue through the required number of cycles until the next T0 and the fetch of the next OP CODE.

While the basic terminology used in this appendix is discussed in the Programming Manual, it has been defined below for ease of reference while studying Single-Cycle Execution.

OP CODE--The first byte of the instruction containing the operator and mode of address.

OPERAND--The data on which the operation specified in the OP CODE is performed.

BASE ADDRESS--The address in Indexed addressing modes which specifies the location in memory to which indexing is referenced. The high-order byte of the base address (AB08 to AB15) is BAH (Base Address High) and the low-order byte of the base address (AB00 to AB07) is BAL (Base Address Low).

EFFECTIVE ADDRESS--The destination in memory in which data are to be found. The effective address may be loaded directly as in the case of Page Zero and Absolute Addressing or may be calculated as in Indexing operations. The high-order byte of the effective address (AB08 to AB15) is ADH and the low-order byte of the effective address (AB00 to AB07) is ADL.

INDIRECT ADDRESS--The address found in the operand of instructions utilizing (Indirect),Y which contains the low-order byte of the base address. IAH and IAL represent the high- and low-order bytes.

JUMP ADDRESS--The value to be loaded into Program Counter as a result of a Jump instruction.

A. 1. SINGLE-BYTE INSTRUCTIONS

ASL	DEX	NOP	TAX	TYA
CLC	DEY	ROL	TAY	
		ROR		
CLD	INX	SEC	TSX	
CLI	INY	SED	TXA	
CLV	LSR	SEI	TXS	

These single-byte instructions require two cycles to execute. During the second cycle the address of the next instruction in program sequence will be placed on the address bus. However, the OP CODE which appears on the data bus during the second cycle will be ignored. This same instruction will be fetched on the following cycle, at which time it will be decoded and executed. The ASL, ROL and LSR instructions apply to the accumulator mode of address.

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T0	PC + 1	OP CODE	1	Next Instruction

A. 2. INTERNAL EXECUTION ON MEMORY DATA

ADC	CMP	EOR	LDY
AND	CPX	LDA	ORA
BIT	CPY	LDX	SBC

The instructions listed above will execute by performing operations inside the microprocessor using data fetched from the effective address. This total operation requires three steps. The first step (one cycle) is the OP CODE fetch. The second (zero to four cycles) is the calculation of an effective address. The final step is the fetching of the data from the effective address. Execution of the instruction takes place during the fetching and decoding of the next instruction.

A. 2.1. Immediate Addressing (2 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.2. Zero Page Addressing (3 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Effective Address
T2	00, ADL	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.3. Absolute Addressing (4 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Low-Order Effective Address Byte
T2	PC + 2	ADH	1	Fetch High-Order Effective Address Byte
T3	ADH, ADL	Data	1	Fetch Data
T0	PC + 3	OP CODE	1	Next Instruction

A. 2.4. Indirect, X Addressing (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + X	ADL	1	Fetch Low-Order Byte of Effective Address
T4	00, BAL + X + 1	ADH	1	Fetch High-Order Byte of Effective Address
T5	ADH, ADL	Data	1	Fetch Data
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.5. Absolute, X or Absolute, Y Addressing (4 or 5 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Low-Order Byte of Base Address
T2	PC + 2	BAH	1	Fetch High-Order Byte of Base Address
T3	ADL: BAL + Index Register	Data*	1	Fetch Data (No Page Crossing)
	ADH: BAH + C			Carry is Ø or 1 as Required from Previous Add Operation
T4*	ADL: BAL + Index Register	Data	1	Fetch Data from Next Page
	ADH: BAH + 1			
T0	PC + 3	OP CODE	1	Next Instruction

*If the page boundary is crossed in the indexing operation, the data fetched in T3 is ignored. If page boundary is not crossed, the T4 cycle is bypassed.

A. 2.6. Zero Page, X or Zero Page, Y Addressing Modes (4 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
TI	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + Index Register	Data	1	Fetch Data (No Page Crossing)
T0	PC + 2	OP CODE	1	Next Instruction

A. 2.7. Indirect, Y Addressing Mode (5 or 6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Page Zero Indirect Address
T2	00, IAL	BAL	1	Fetch Low-Order Byte of Base Address
T3	00, IAL + 1	BAH	1	Fetch High-Order Byte of Base Address
T4	ADL: BAL + Y ADH: BAH + C	Data*	1	Fetch Data from Same Page Carry is 0 to 1 as Required from Previous Add Operation
T5*	ADL: BAL + Y ADH: BAH + 1	Data	1	Fetch Data from Next Page
T0	PC + 2	OP CODE	1	Next Instruction

*If page boundary is crossed in indexing operation, the data fetch in T4 is ignored. If page boundary is not crossed, the T5 cycle is bypassed.

A. 3. STORE OPERATIONS

STA
STX
STY

The specific steps taken in the Store Operations are very similar to those taken in the previous group (internal execution on memory data). However, in the Store Operation, the fetch of data is replaced by a WRITE (R/W = 0) cycle. No overlapping occurs and no shortening of the instruction time occurs on indexing operations.

A. 3.1. Zero Page Addressing (3 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Zero Page Effective Address
T2	00, ADL	Data	0	Write Internal Register to Memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.2. Absolute Addressing (4 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Low-Order Byte of Effective Address
T2	PC + 2	ADH	1	Fetch High-Order Byte of Effective Address
T3	ADH, ADL	Data	0	Write Internal Register to Memory
T0	PC + 3	OP CODE	1	Next Instruction

A. 3.3. Indirect, X Addressing (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	00, BAL + X	ADL	1	Fetch Low-Order Byte of Effective Address
T4	00, BAL + X + 1	ADH	1	Fetch High-Order Byte of Effective Address
T5	ADH, ADL	Data	0	Write Internal Register to Memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.4. Absolute, X or Absolute, Y Addressing (5 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Low-Order Byte of Base Address
T2	PC + 2	BAH	1	Fetch High-Order Byte of Base Address
T3	ADL: BAL + Index Register ADH: BAH + C	Data (Discarded)	1	
T4	ADH, ADL	Data	0	Write Internal Register to Memory
T0	PC + 3	OP CODE	1	Next Instruction

A. 3.5. Zero Page, X or Zero Page, Y Addressing Modes (4 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded)	1	
T3	ADL: BAL + Index Register	Data	0	Write Internal Register to Memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 3.6. Indirect, Y Addressing Mode (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Page Zero Indirect Address
T2	00, IAL	BAL	1	Fetch Low-Order Byte of Base Address
T3	00, IAL + 1	BAH	1	Fetch High-Order Byte of Base Address
T4	ADL: BAL + Y ADH: BAH	Data (Discarded)	1	
T5	ADH, ADL	Data	0	Write Internal Register to Memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 4. READ -- MODIFY -- WRITE OPERATIONS

ASL	LSR
DEC	ROL
INC	ROR

The Read -- Modify -- Write operations involve the loading of operands from the operand address, modification of the operand and the resulting modified data being stored in the original location.

A. 4.1. Zero Page Addressing (5 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Page Zero Effective Address
T2	00, ADL	Data	1	Fetch Data
T3	00, ADL	Data	0	
T4	00, ADL	Modified Data	0	Write Modified Data Back into Memory
T0	PC + 2	OP CODE	1	Next Instruction

A. 4.2. Absolute Addressing (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Low-Order Byte of Effective Address
T2	PC + 2	ADH	1	Fetch High-Order Byte of Effective Address
T3	ADH, ADL	Data	1	
T4	ADH, ADL	Data	0	
T5	ADH, ADL	Modified Data	0	Write Modified Data Back into Memory
TØ	PC + 3	OP CODE	1	Next Instruction

A. 4.3. Zero Page, X Addressing (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Page Zero Base Address
T2	00, BAL	Data (Discarded	1	
T3	ADL: BAL + X (Without Carry)	Data	1	Fetch Data
T4	ADL: BAL + X (Without Carry)	Data	0	
T5	ADL: BAL + X (Without Carry)	Modified Data	0	Write Modified Data Back into Memory
TØ	PC + 2	OP CODE	1	Next Instruction

A. 4.4. Absolute, X Addressing (7 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	BAL	1	Fetch Low-Order Byte of Base Address
T2	PC + 2	BAH	1	Fetch High-Order Byte of Base Address
T3	ADL: BAL + X ADH: BAH + C	Data (Discarded)	1	
T4	ADL: BAL + X ADH: BAH + C	Data	1	Fetch Data
T5	ADH, ADL	Data	0	
T6	ADH, ADL	Modified Data	0	Write Modified Data Back into Memory
T0	PC + 3	OP CODE	1	New Instruction

A. 5. MISCELLANEOUS OPERATIONS

BCC	BRK	PHP
BCS	BVC	PLA
BEQ	BVS	PLP
BMI	JMP	RTI
BNE	JSR	RTS
BPL	PHA	

A. 5.1. Push Operation -- PHP, PHA (3 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Pointer*	Data	0	Write Internal Register into Stack
T0	PC + 1	OP CODE	1	Next Instruction

*Hereafter referred to as "Stack Ptr."

A. 5.2. Pull Operations -- PLP, PLA (4 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	OP CODE (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr. + 1	Data	1	Fetch Data from Stack
T0	PC + 1	OP CODE	1	Next Instruction

A. 5.3. Jump to Subroutine -- JSR (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Low-Order Byte of Subroutine Address
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr.	PCH	0	Push High-Order Byte of Program Counter to Stack
T4	Stack Ptr. - 1	PCL	0	Push Low-Order Byte of Program Counter to Stack
T5	PC + 2	ADH	1	Fetch High-Order Byte of Subroutine Address
T0	Subroutine Address (ADH, ADL)	OP CODE	1	

A. 5.4. Break Operation -- (Hardware Interrupt)-BRK (7 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch BRK OP CODE (or Force BRK)
T1	PC + 1 (PC on Hardware Interrupt)	Data (Discarded)	1	
T2	Stack Ptr.	PCH	0	Push High-Order Byte of Program Counter to Stack
T3	Stack Ptr. - 1	PCL	0	Push Low-Order Byte of Program Counter to Stack
T4	Stack Ptr. - 2	P	0	Push Status Register to Stack
T5	FFFE (NMI-FFFA) (RES-FFFC)	ADL	1	Fetch Low-Order Byte of Interrupt Vector
T6	FFFF (NMI-FFFB) (RES-FFFD)	ADH	1	Fetch High-Order Byte of Interrupt Vector
T0	Interrupt Vector (ADH, ADL)	OP CODE	1	Next Instruction

A. 5.5. Return from Interrupt -- RTI (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr. + 1	Data	1	Pull P from Stack
T4	Stack Ptr. + 2	Data	1	Pull PCL from Stack
T5	Stack Ptr. + 3	Data	1	Pull PCH from Stack
T0	PCH, PCL	OP CODE	1	Next Instruction

A. 5.6. Jump Operation -- JMP

A.5.6.1. Absolute Addressing Mode (3 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	ADL	1	Fetch Low-Order Byte of Jump Address
T2	PC + 2	ADH	1	Fetch High-Order Byte of Jump Address
T0	ADH, ADL	OP CODE	1	Next Instruction

A.5.6.2. Indirect Addressing Mode (5 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	IAL	1	Fetch Low-Order Byte of Indirect Address
T2	PC + 2	IAH	1	Fetch High-Order Byte of Indirect Address
T3	IAH, IAL	ADL	1	Fetch Low-Order Byte of Jump Address
T4	IAH, IAL + 1	ADH	1	Fetch High-Order Byte of Jump Address
T0	ADH, ADL	OP CODE	1	Next Instruction

A. 5.7. Return from Subroutine -- RTS (6 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
T0	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Data (Discarded)	1	
T2	Stack Ptr.	Data (Discarded)	1	
T3	Stack Ptr. + 1	PCL	1	Pull PCL from Stack
T4	Stack Ptr. + 2	PCH	1	Pull PCH from Stack
T5	PCH, PCL (from Stack)	Data (Discarded)	1	
T0	PCH, PCL + 1	OP CODE	1	Next Instruction

A. 5.8. Branch Operation -- BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS (2, 3, or 4 Cycles)

Tn	Address Bus	Data Bus	R/W	Comments
TO	PC	OP CODE	1	Fetch OP CODE
T1	PC + 1	Offset	1	Fetch Branch Offset
T2*	PC + 2 + Offset (w/o) Carry)	OP CODE	1	Offset Added to Program Counter
T3**	PC + 2 + Offset (with Carry)	OP CODE	1	Carry Added

*Skip if branch not taken.

**Skip if branch not taken; skip if branch operation does not cross page boundary.

PLACE STAMP HERE

Documentation Manager
D727, RC55
Rockwell International
MICROELECTRONIC DEVICES
3310 Miraloma Ave.
Anaheim, CA 92803