PPLICATIONS INFORMATION AN5

MARCH 1982

SY6522
Versatile Interface Adapter
Microprocessor Products

SYNERTEK

CONTENTS

The Synertek 6522 is truly a versatile I/O device. Its many operating modes and ease of programming make its inclusion in a design very desirable.

SCOPE

The purpose of this note is not to reiterate the information contained in the data sheet, but to answer the most frequently asked questions. The SY6522 data sheet contains all the device's electrical and timing characteristics and descriptions of the control register organization. This note will occasionally reference the data sheet and direct the reader to specific pages and figures.

1.0 Interrupt Flag Register (IFR) and Interrupt Enable Register (IER)

Understanding these two registers and their interaction with each is imperative for efficiently using the 6522.

The IER is basically seven switches linking the seven interrupt flags of the IFR to the IRQ output and to IFR bit 7. Writing 7F (H) to the IER disables (opens the switches) all external interrupts. Neither the IRQ output nor IFR bit 7 will go true. However, bits 0-6 of the IFR will still reflect the interrupt state of their respective 6522 function.

By writing 1XXX XXXX (binary value where any x = 1) to the IER, the corresponding bit in the IFR will be linked to the IRQ output and to IFR bit 7.

Notes: Corrections to earlier data sheets and catalogues.

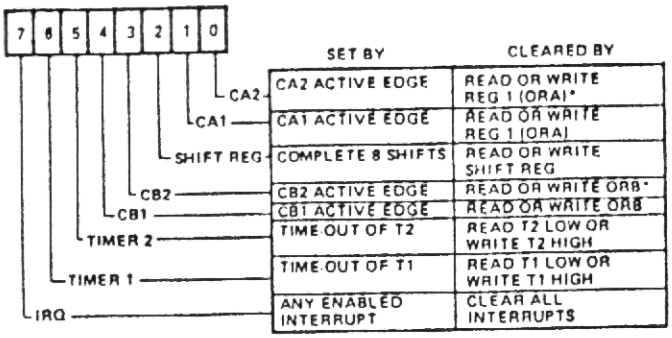
1. Clearing IFR bits 0 and 3. If "Independent Interrupt" is selected in the PCR these bits can be reset ONLY by writing a "1" to their bit position in the IFR.
2. When reading the IER bit 7 will be a "1" not a "0".

2.0 Port A and Port B

These two 8-bit ports are fully described in the data sheet and need little clarification. Operation in the "latch" mode has presented a few questions that are not fully covered in the early data sheets. When either bits 0 or 1 of the ACR are set to a logic "1" their respective ports are put in the latch mode. As described in the data sheet, the port lines to be used must be programmed as inputs. At this point a read of either input register will effectively be a read of the port lines. Not until CA1 or CB1 has transitioned (as programmed in the PCR) will any data be latched. Now a read of the input register will reflect the data that was on the port line at the time of the latching transition. After this read the input registers will again appear transparent until the next CA1 or CB1 transition. See Figure 2 below for an illustration of reading in the latch mode.

Another area relating to the latch mode not specified in earlier data sheets is TAL, (CA1, CB1 set up, prior to the triggering transition). The new data catalog specs this at 300ns which is for a 2 MHz $\phi 2$ clock only. The formula for

REG 13 – INTERRUPT FLAG REGISTER



	SET BY	CLEARED BY
CA2	CA2 ACTIVE EDGE	READ OR WRITE REG 1 (ORA)*
CA1	CA1 ACTIVE EDGE	READ OR WRITE REG 1 (ORA)
SHIFT REG	COMPLETE 8 SHIFTS	READ OR WRITE SHIFT REG
CB2	CB2 ACTIVE EDGE	READ OR WRITE ORB*
CB1	CB1 ACTIVE EDGE	READ OR WRITE ORB
TIMER 2	TIME-OUT OF T2	READ T2 LOW OR WRITE T2 HIGH
TIMER 1	TIME-OUT OF T1	READ T1 LOW OR WRITE T1 HIGH
IRQ	ANY ENABLED INTERRUPT	CLEAR ALL INTERRUPTS

REG 14 – INTERRUPT ENABLE REGISTER

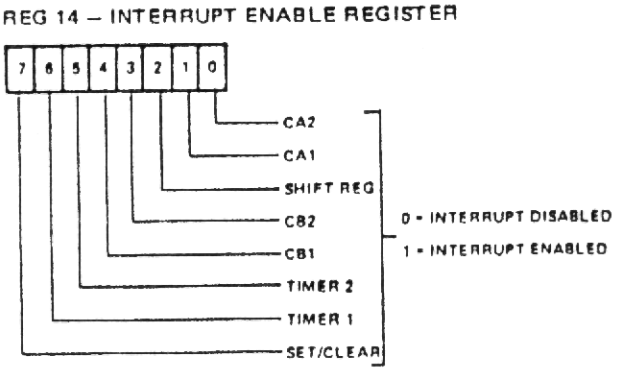
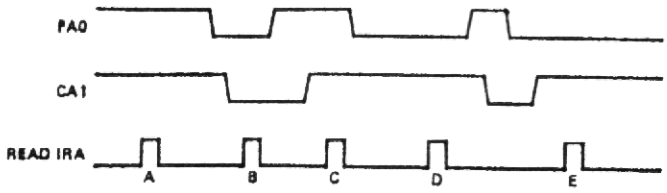


Figure 1.



ASSUMPTIONS: 1. LATCHING ON NEGATIVE TRANSITION.
2. NO TRANSITIONS HAD OCCURRED PRIOR TO READ 'A'.

READ OPERATION	DATA BUS BIT 0
A	1
B	0
C	1
D	0
E	1

Figure 2.

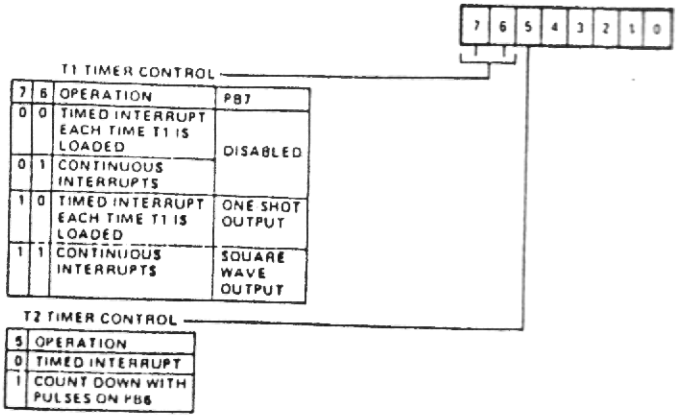


Figure 3. Timer Modes of Operation

determining this value is Tcy/2 + 50ns. This means: if the user has programmed the 6522 for latch mode, latching to occur on the negative edge of CA1, then CA1 must be high for a minimum TAL before the triggering edge occurs.

3.0 The Timers

The 6522 timers have generated a lot of customer calls due to the not totally clear information available. First a quick review of their associated registers, describing what happens when they are read or written. Then a discussion of how the modes operate and programming sequences to get them started and keep them running.

3.1 Timer 1

T1 LOW-ORDER COUNTER (T1-CL)

Writing to the T1-CL is effectively a write to the low-order latch. The data is held in the latch until the high-order counter is written: at this time the data is transferred to the counter. A read T1-CL transfers the counter's contents to the data bus and if a T1 interrupt has occurred the read operation will clear the IFR flag and reset $\overline{IRQ}$.

T1 LOW-ORDER LATCH (T1L-L)

Writing T1L-L stores an 8-bit count value into the latch. Effectively the same as a write to T1C-L. A read of T1L-L transfers the latch's contents to the data bus; it has no affect on the T1 interrupt flag.

T1 HIGH-ORDER COUNTER (T1C-H)

A write to T1C-H loads both the high-order counter and high-order latch with the same value; simultaneously the T1L-L contents are transferred to the low-order counter and the count begins. If PB7 has been programmed as a TIMER 1 output it will go low on the $\phi 2$ following the write operation. Additionally, if the T1 interrupt flag has already been set, the write operation will clear it. A read of T1C-H transfers the counter's contents to the data bus.

T1 HIGH-ORDER LATCH (T1L-H)

A write to T1L-H loads an 8-bit count value into the latch. A read of T1L-H transfers the contents of the latch to the data bus. Neither operation has an affect on the interrupt flag.

3.2 T1 Operation

3.2.1 Timed Interrupts

When both ACR bits 6 and 7 are zero, T1 will generate one timed interrupt whenever T1C-H is written.

As an example:

1. Write 00 (H) to ACR — select T1 mode.
2. Write C0 (H) to IER — enable T1 interrupts.
3. Write 0A (H) to T1C-L — (decimal 10).
4. Write 00 (H) to T1C-H — (decimal 0).

The write to T1C-H initiates the countdown on the next $\phi 2$. The counter decrements on each succeeding $\phi 2$ from 10 to 0 and then one half $\phi 2$ cycle later IRQ goes active. The countdown is 11 and a half cycles. Or, the timed period to interrupt is equal to N (where N is the combined count value of T1C-L and T1C-H) plus one and one half cycles. See Figure 4.

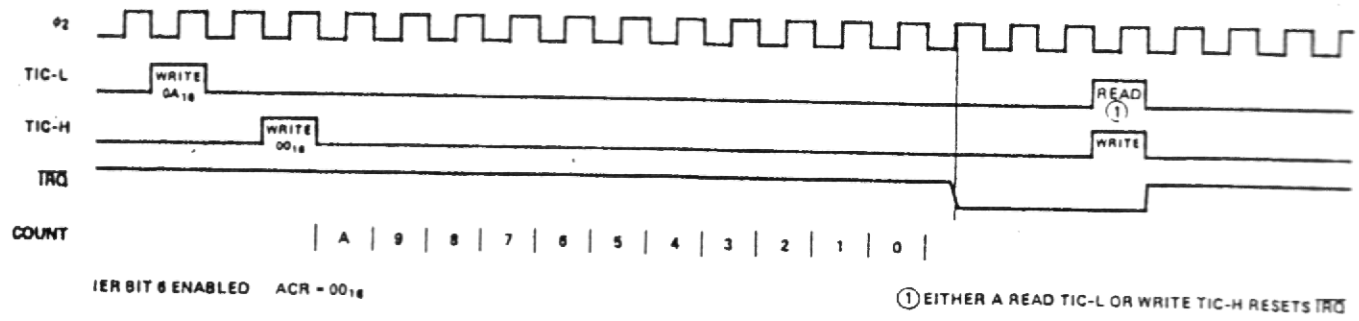


Figure 4. Timed Interrupt

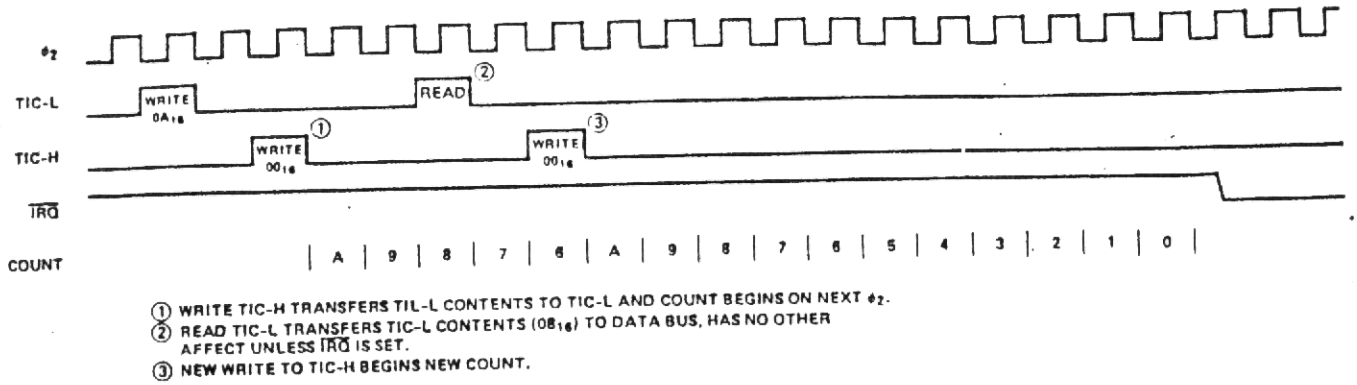


Figure 5. Timed Interrupt

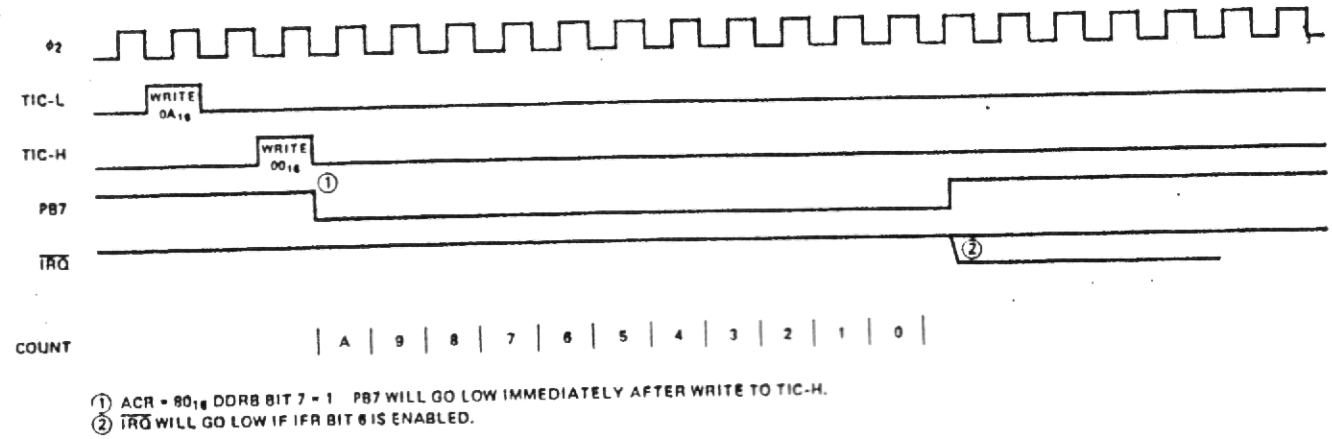


Figure 6. One-Shot Output

In this mode the $\overline{IRQ}$ output can be prevented from going low if T1C-H is written before time out occurs. A user can let $\overline{IRQ}$ act as an alarm if a task of known length takes too long. If $\overline{IRQ}$ ever did set, the user can then go to a fault routine. (refer to Figure 5)

3.2.2 One-Shot Output

By setting bit 7 in the ACR to a one, PB7 will be enabled as a one-shot output. PB7 will go low immediately after writing T1C-H. The duration of the pulse is equal to N + one and one half (where N equals the count value) to guarantee a valid output level on PB7. Bit 7 of DDRB must also be set to a one. ORB bit 7 will NOT affect the level on PB7 (refer to Figure 6)

3.2.3 Continuous Interrupts

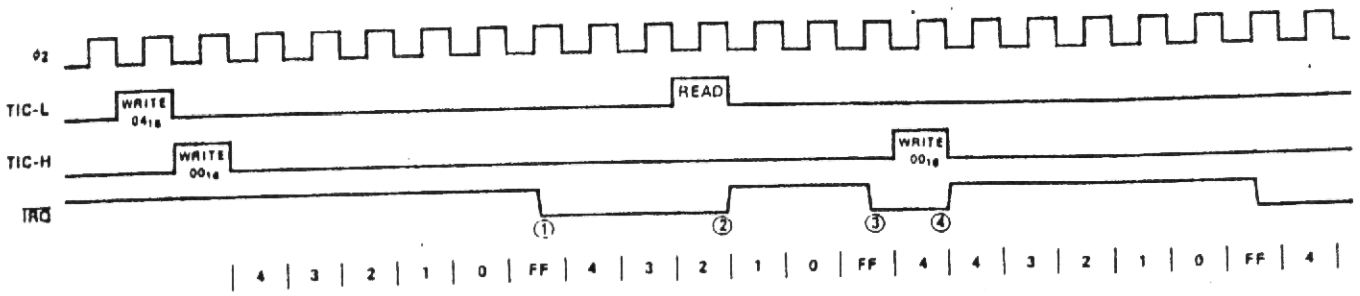
T1 will generate continuous interrupts when bit 6 of the ACR is a one. In effect, this bit provides a link between the latches and counter; automatically loading the counters from the latches when time out occurs. Note: when in this mode and $\overline{IRQ}$ is enabled, $\overline{IRQ}$ will go low after the first (and each succeeding) time out and stay low until either TIC-L is read or T1C-H is written. (Refer to Figure 7).

3.2.4 Square Wave Output

When both ACR bits 6 and 7 are a one, PB7 will be enabled, and a square wave output will be generated. IRQ can be either enabled or disabled and will have no affect on the PB7 output. Immediately after writing T1C-H PB7 will go low for N + one and a half cycles. Beginning with the following high half-phase of PB7 each succeeding half-phase will be N + 2 ϕ2 cycles. (See Figure 8).

By taking advantage of the Timer's flexibility, a complex waveform can be generated on PB7. By writing new values into the latches (not the counters), the new values will automatically be loaded into the counters when time out of the previous value occurs. The following sequence can be used.

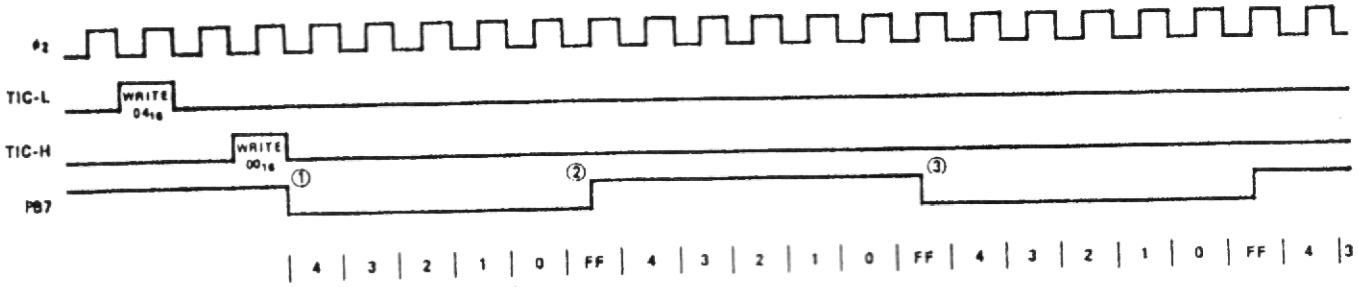
1. Load value A into T1C-L, T1C-H.
2. Load value B into T1L-L, T1L-H.
3. Bit test IFR bit 6, when set value B will be loaded into the counters and value C can be loaded into the latches.
4. Read T1C-L to reset IFR bit 6.
5. Repeat steps 3 and 4 loading a new value into the latches each time bit 6 gets set.



ACR = 40_{16} IER BIT 6 ENABLED

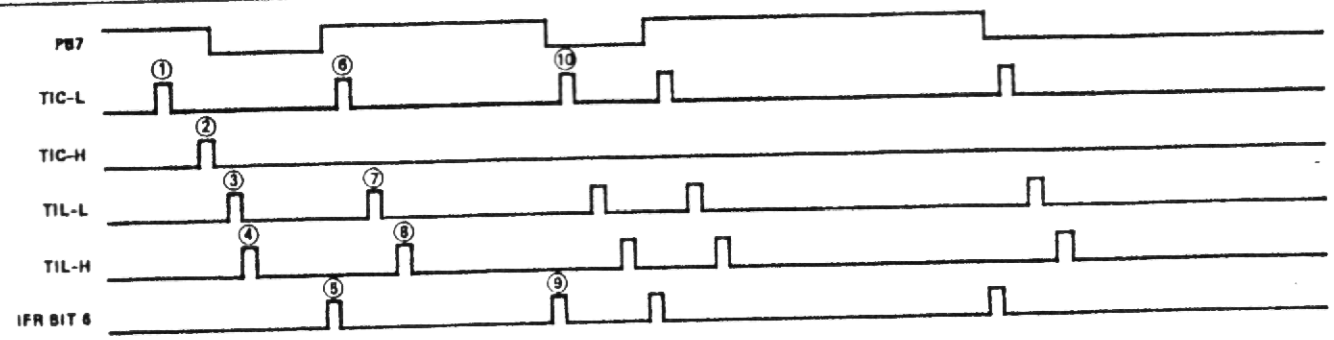
① FIRST COUNT COMPLETE, TIMEOUT = N + 1½, NEXT ϕ_2 RELOADS COUNTERS FROM LATCHES.
② READ OF TIC-L TRANSFERS COUNTER'S CONTENTS TO BUS AND RESETS $\overline{IRQ}$.
③ IRQ SET, NOTE COUNT FROM ① TO ③ IS 6 (N+2).
④ WRITE TO TIC-H RESETS $\overline{IRQ}$ AND RESTARTS COUNT.

Figure 7. Continuous Interrupts



IER BIT 6 DISABLED, ACR = C0
NOTE: TIME BETWEEN TRANSITIONS ① AND ② EQUALS N + 1½
ALL OTHERS EQUAL N + 2 AS BETWEEN ② AND ③ .

Figure 8. PB7 Square Wave Output



① WRITE TIC-L COUNT A LOW.
② WRITE TIC-H COUNT A HIGH, PB7 GOES LOW, COUNT BEGINS.
③ WRITE TIL-L COUNT B LOW.
④ WRITE TIL-H COUNT B HIGH.
⑤ COUNT A COMPLETE, COUNT B LOADED INTO COUNTERS, IFR BIT 6 HIGH.
⑥ READ TIL-C, RESETS IFR BIT 6.
⑦ WRITE TIL-L COUNT C LOW.
⑧ WRITE TIL-H COUNT C HIGH.
⑨ COUNT B COMPLETE COUNT C LOADED, IFR BIT 6 HIGH.
⑩ READ TIC-L, RESETS IFR BIT 6 AND SO ON.

Figure 9. PB7 Variable Pulse Output

4.0 Timer 2

Timer 2 operates as an interval timer, similar to T1 timed interrupt mode, and as a negative pulse counter. Bit 5 of the ACR selects the mode of operation, (see Figure 3). IFR bit 5 is the T2 interrupt flag, and IER bit 5 is the T2 $\overline{IRQ}$ control bit.

There are only two addressable T2 registers.

REGISTER 8 T2 LOW-ORDER COUNTER/LATCH (T2C-L)

Writing T2C-L effectively stores an 8-bit byte in a write only latch where it will be held until the count is initiated. A read of T2C-L transfers the contents of the low-order counter to the data bus, and if a T2 interrupt has occurred, the read operation will clear the T2 interrupt flag and reset $\overline{IRQ}$.

REGISTER 9 T2 HIGH-ORDER COUNTER/ LATCH (T2C-H)

Writing T2C-H loads an 8-bit byte into the high-order counter and latch and simultaneously loads the low-order latch into the low-order counter, and the count down is initiated. If a T2 interrupt has occurred, the write operation will clear the T2 interrupt flag and reset $\overline{IRQ}$. A read of T2C-H transfers the contents of the high-order counter to the data bus.

4.1 Timed Interrupts

T2 timed interrupts operate similarly to T1 timed interrupts. First the low-order value is loaded into T2C-L, then the high-order value is loaded into T2C-H, and the count down begins. When count N + 1 1/2 is reached, IFR bit 5 and/or IRQ are set. One additional feature of T2 is, that on reaching 0000(H) the counter rolls over to FFFF(H) and continues to decrement. This allows the user to read the counter and determine exactly when the interrupt occurred.

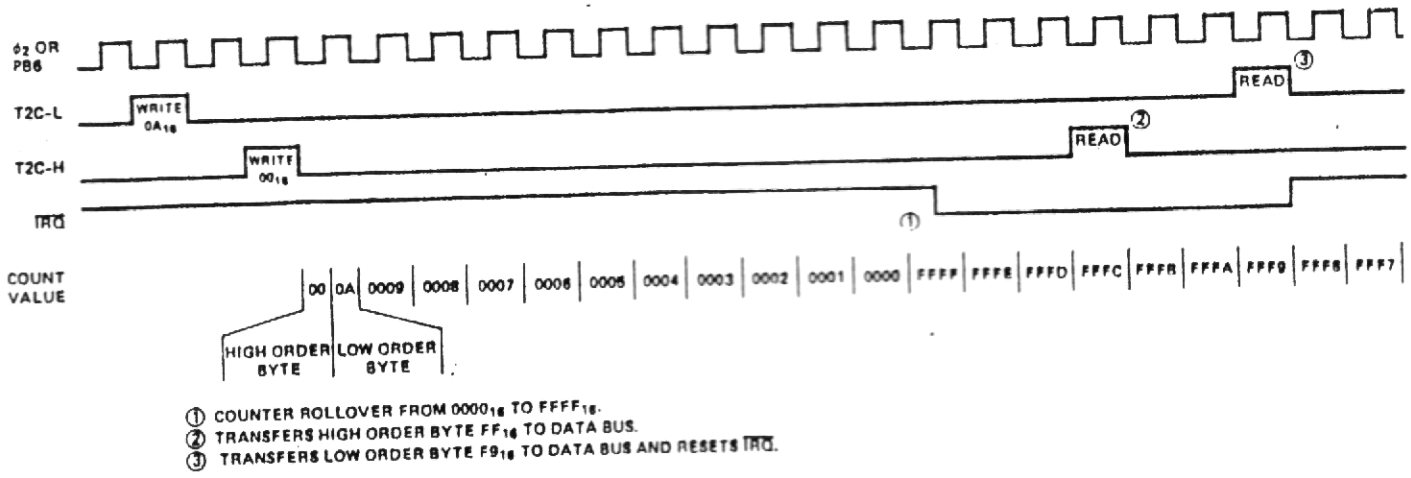
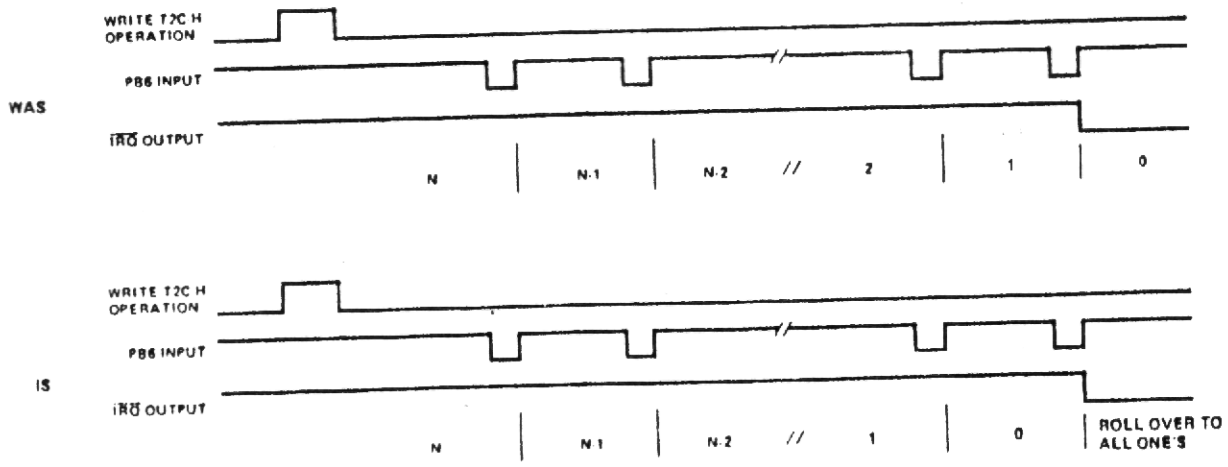


Figure 10. T2 Timed Interrupts Timer 2 Pulse Counting Mode



CORRECTION TO FIG. 21 IN DATA SHEET. THE 'IS' CONDITION IS CORRECT.

Figure 11.

4.2 Pulse Counting

Timer 2 can also be used to count negative pulses on the PB6 line. There is a minimum high and low period required on the pulse stream on PB6, both the high period and low pulse must be a minimum 2xTcy. A correction to all data sheets is required. $\overline{IRQ}$ does not set when the count reaches zero. $\overline{IRQ}$ or bit 5 of the IFR will set when the counter rolls over to FFFF(H). For example, loading decimal 2 into the counter requires 3 pulses to set $\overline{IRQ}$. Therefore, if you are using PB6 as an additional interrupt input, load all zeroes into T2 and a single negative pulse on PB6 will cause an interrupt.

4.3 Timer 2 and the Shift Register

There is one other operating mode for T2. It can be used as the shift clock for the shift register. A more detailed description will be given in the next section. Briefly, T2 timeouts can be used to clock data in or out of the shift register. In these modes of operation only the low-order 8 bits of the counter are applicable. Although not stated in the data sheet T2 can be configured to be clocked externally, making it relatively easy to shift data asynchronously at a standard 16x clock.

5.0 Shift Register

The shift register operation is straight forward and covered quite thoroughly in the data sheet. One thing to keep in mind if using it for serial communication, the MSB is shifted in/out first. The standard convention is to shift the LSB first. If trying to communicate with another device using a standard transmission scheme two alternatives are available: one hardware and the other software. Refer to Figure 12 below for a hardware technique of swapping the byte end-to-end.

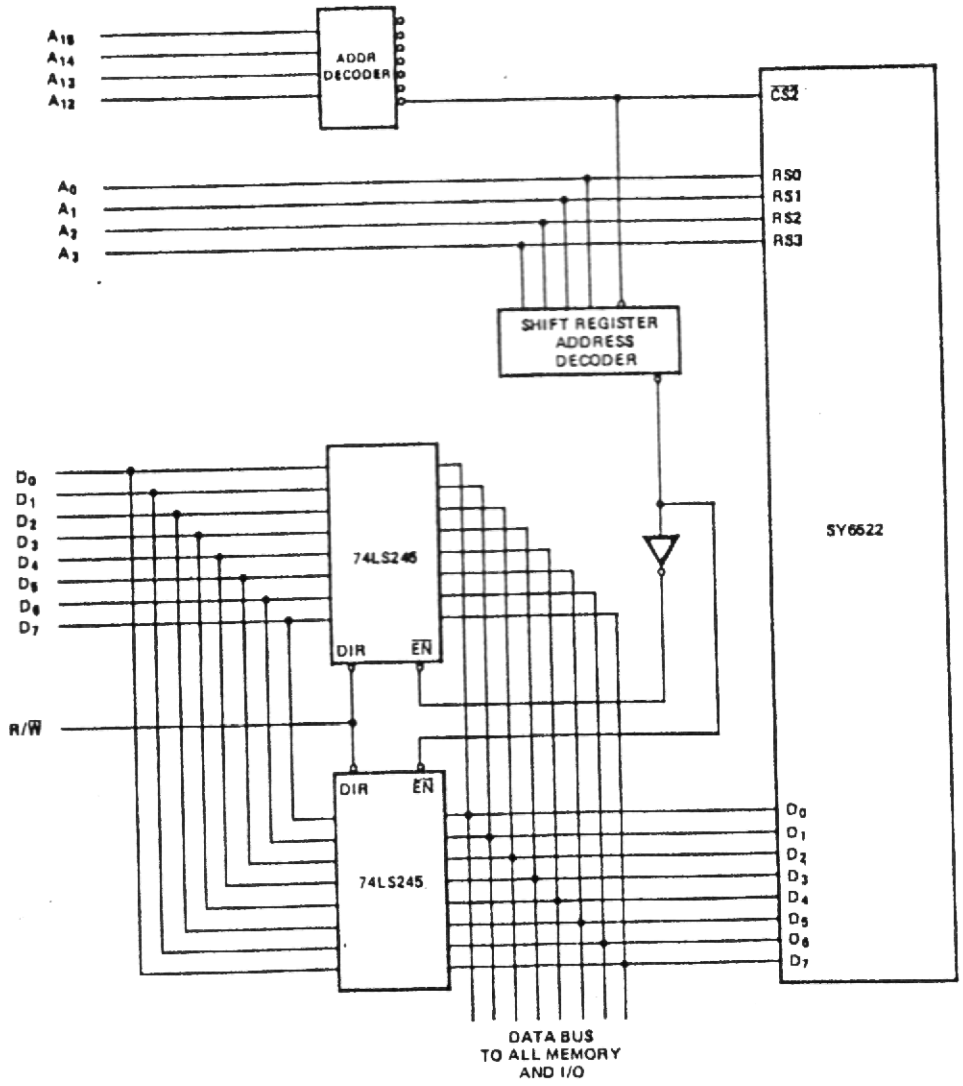


Figure 12.

5.1 Shift Register Warnings

MODE 010

We receive numerous calls reporting that the shift-in under control of $\phi 2$ outputs 9 shift pulses. Synertek corrected this problem a number of years ago. However, as of this writing, the other two manufacturers still have this problem. It was a design error that Synertek believed serious enough to correct.

MODES 011 AND 111

When shifting data with an external clock (CB1) more than 8 pulses may occur before $\overline{IRQ}$ is set. In one instance this is caused by an idiosyncrasy of the device; in the other it is caused by user implementation.

In both the shift-in and shift-out modes a timing condition may occur when the SY6522 does not detect the shift pulse. This no-shift condition occurs when CB1 and $\phi 2$ are asynchronous and their edges coincide as illustrated in Figure 13.

This no-shift condition occurs in a 6522 regardless of the manufacturer. The problem can be remedied in a number of ways. Where convenient, CB1 can be derived from $\phi 2$, transitioning on the $\phi 2$ rising edge. When CB1 cannot be derived in this manner it can be synced to $\phi 2$ by the use of a 'D' type filp-flop.

The other failure is not a 'no-shift', but an extra ninth shift, and can be attributed to a lack of information in the data sheet. The failure occurs when shifting data out under control of CB1, where CB1 is free-running. This usually occurs when a user tries to implement the SY6522 as a UART.

With CB1 free-running, $\overline{IRQ}$ goes low after the eighth negative transition on CB1. Generally, the interrupt is serviced by fetching the next byte and storing it in the shift register. The store operation resets $\overline{IRQ}$ and the shift register counter. If the store to the shift register and a negative transition on CB1 are coincident, a simultaneous store and shift occur before $\overline{IRQ}$ and the shift register counter are reset. The result is bit 7 (MSB) is immediately shifted out on CB2, then 8 more bits are shifted before $\overline{IRQ}$ is set. Figure 14 below illustrates the fault condition and the resultant error.

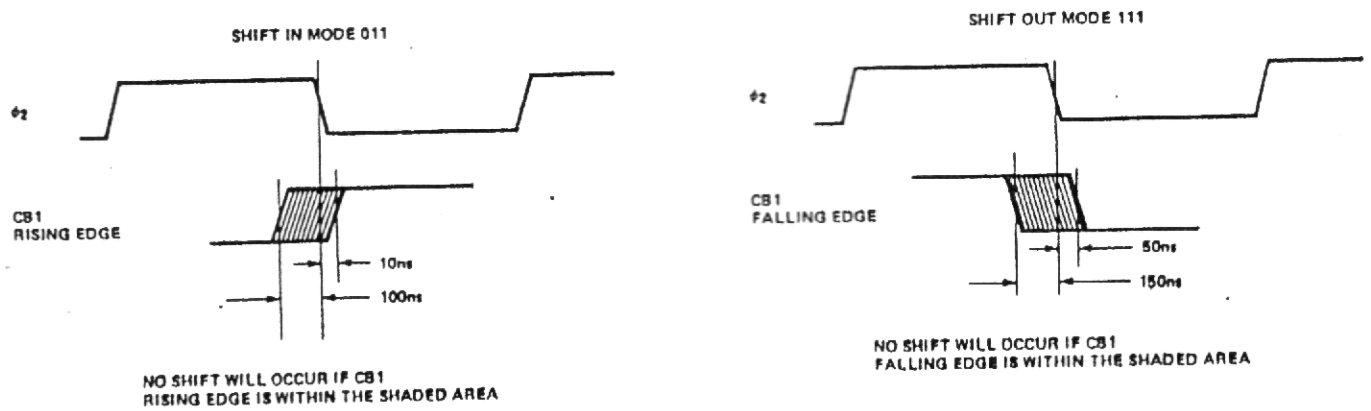
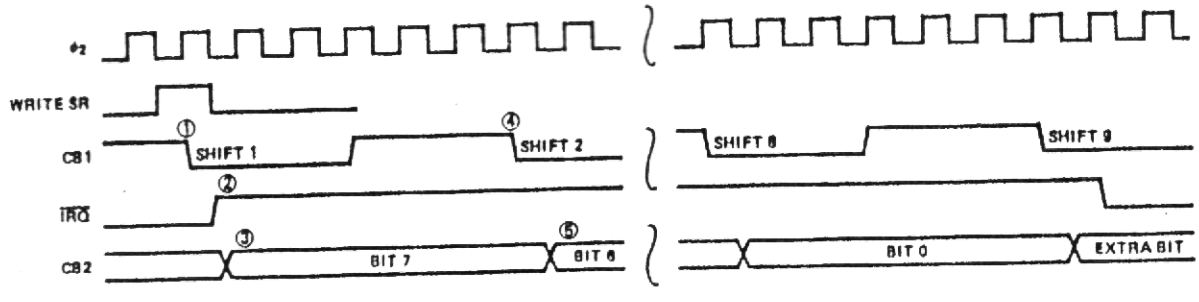


Figure 13. "No Shift" Condition



① DATA WRITTEN TO SR AND SHIFTED BY CB1
② $\overline{IRQ}$ RESET, SHIFT REGISTER COUNTER RESET
③ BIT 7 TRANSFERRED TO CB2
④ SECOND SHIFT OCCURS, SHIFT COUNTER COUNTS 1
⑤ BIT 6 TRANSFERRED TO CB2 AND SO ON UNTIL 9 SHIFTS HAVE OCCURRED

Figure 14.

This shouldn't be much of a problem at low transmission rates unless the interrupt routine is extremely long. However, at high transmission rates timing does become critical and the user will have to ensure the CB1 negative edge cannot occur during the Write cycle.

This can be accomplished by counting interrupt routine cycle times (best and worst case) and adjusting CB1 so it will not transition low again until the write shift register operation is complete and $\overline{IRQ}$ has returned high. Where adjusting CB1 is not possible, the user could gate CB1 with $\overline{IRQ}$ disabling all transitions until the shift register counter is reset.

5.2 8 Bits Only

A word of caution. The shift register shifts 8 bits. Users have tried to transmit less than 8 bits and have run into trouble. You cannot shift 7 bits and then write another byte to the shift register hoping for 7 bit transmission. The 8th bit in the byte will be transmitted.

5.3 Starting the Shift Register

First of all the ACR must be programmed to enable the shift register. Starting a shift out operation is accomplished by writing a byte to the shift register. To start a shift in operation either a read or write of the shift register is required. Shifting in the next byte is enabled by reading the last byte received.

APPLICATIONS

6.0 SY6522 Interrupt Control

Organization of the SY6522 interrupt flags into a single register greatly enhances the servicing of interrupts from this device. Since there is only one $\overline{IRQ}$ output for the seven possible sources of interrupt within the chip, the processor must examine these flags to determine the cause of the interrupt. There are a number of methods of doing this; one method on sensing an interrupt is to immediately read the IFR contents into an accumulator (or register) and test bit 7. If set then the SY6522 was the interrupting device, if not set then check for another cause. With a SY6502 this is easily done using the BPL or BMI instructions. If the SY6522 was the interrupting device a masking of the unwanted bits can be performed by ANDing the accumulator with the IER, leaving as ones only those sources of interrupts that were enabled. Now simple shift routines can be performed to determine which one of the bits is set to further direct the CPU for correct action.

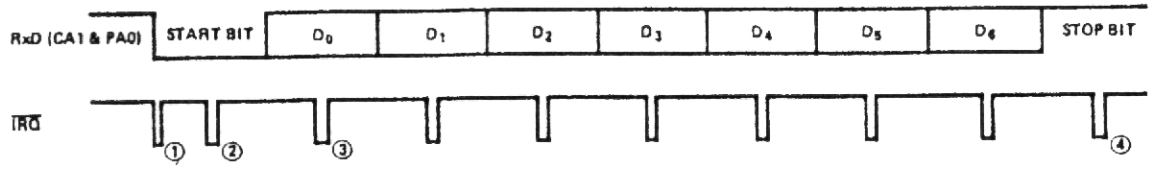
7.0 Serial Communications

The SY6522 can support very limited serial communications. In new design-ins where board space and cost can be justified a dedicated device such as the SY6551 (ACIA) or SY2661 (EPCI) would be a designer's first choice. If, however, you're limited in space and the 6522 is already incorporated in the design serial communications can be supported.

Asynchronous communications formats can take many forms. There will generally be a start bit, followed by 5 to 8 data bits, then optionally a parity bit and 1, 1-1/2 or 2 stop bits. The information cell can be up to 11 bits in length. This creates obvious problems for the shift register which can only handle 8 bits of data.

However, by using T1, a transition-sensitive input (CA1, CB1, CA2 or CB2) and any port line, serial communications can be established. In this example we'll use CA1 and PA0, by tying them together we can form an edge-sensitive RxD input. To begin operation CA1 would be programmed to generate an interrupt on a negative transition to detect a start bit. Upon detection of a transition CA1 would be disabled, T1 started to time out after one half bit time. With a T1 interrupt PA0 would be read to guarantee it was still zero (a true start bit); if not zero restart routine, if zero start T1 count down for full bit time.

On each successive T1 interrupt read PA0 store and shift until the data and parity if selected are received. Then a check for stop bits to detect any framing error in the receipt of the data. Now start over again searching for the next start bit. As can be seen this is very software intensive taking a lot of CPU time, and therefore practical for only low baud rates.



① CA1 INTERRUPT, SET T1 TO HALF BIT TIMEOUT, DISABLE CA1.
② T1 HALF BIT TIME, VERIFY START BIT VALID, START FULL BIT TIMEOUTS.
③ FULL BIT TIMEOUT, READ PA0 FOR DATA, RESET $\overline{IRQ}$
④ FINAL $\overline{IRQ}$, CHECK FOR VALID STOP BIT, RE-ENABLE CA1.

Figure 15.

8.0 Shift Register Serial Communication

As mentioned above the shift register is limited as an asynchronous communications device. It can still be used quite effectively. The shift register was designed primarily as a synchronous serial communications port for distributed systems. These systems can be either single processor with distributed peripheral controllers or distributed processor systems. The most important characteristic of the shift register in these applications is its ability to transfer information at relatively slow data rates to allow the use of R-C noise suppression techniques. The transfers can be accomplished while the processor is servicing other aspects of the system. A simplified 2 processor distributed system is shown in the drawing below. Use of the SY6522 shift register allows effective communication between the two systems without the use of complex asynchronous communications techniques.

In a system with distributed peripherals, the shift register can be used to transfer data to the peripheral interface devices. This is illustrated in Figure 17 for a system with a number of distributed status displays. These displays are serviced by stand-alone controllers which actuate the lamps in the status displays with simple drivers. The data and clock lines are wired in parallel to each unit. In addition, a single SY6522 peripheral port allows selection of the display to be updated. With the system shown, the status display can be updated at any time by simply selecting the desired display and then writing to the shift register.

Remote input devices can be serviced in much the same manner by shifting data into the shift register under control of a peripheral port output as shown in Figure 17. Each set of input switches can be polled by first selecting the set to be read then triggering the shift operation with a shift register read. A shift register interrupt can be used to cause the processor to read the input information after shifting is complete.

The techniques described above can be utilized to expand I/O capabilities in a very cost-effective manner. A control system requiring an extensive amount of I/O can be implemented with one SY6522 and numerous TTL shift registers as illustrated in Figure 18.

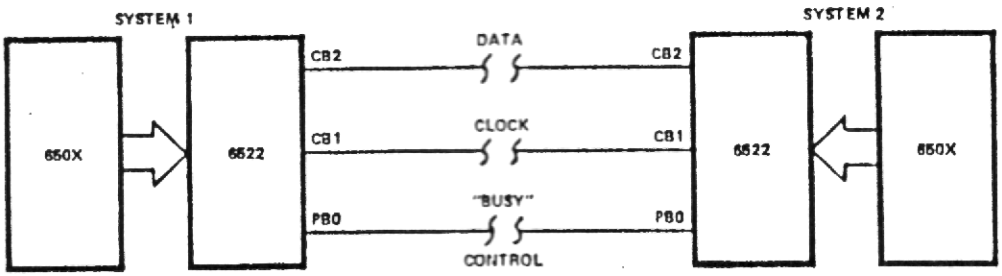


Figure 16. Shift Register for 2 Processor Communications

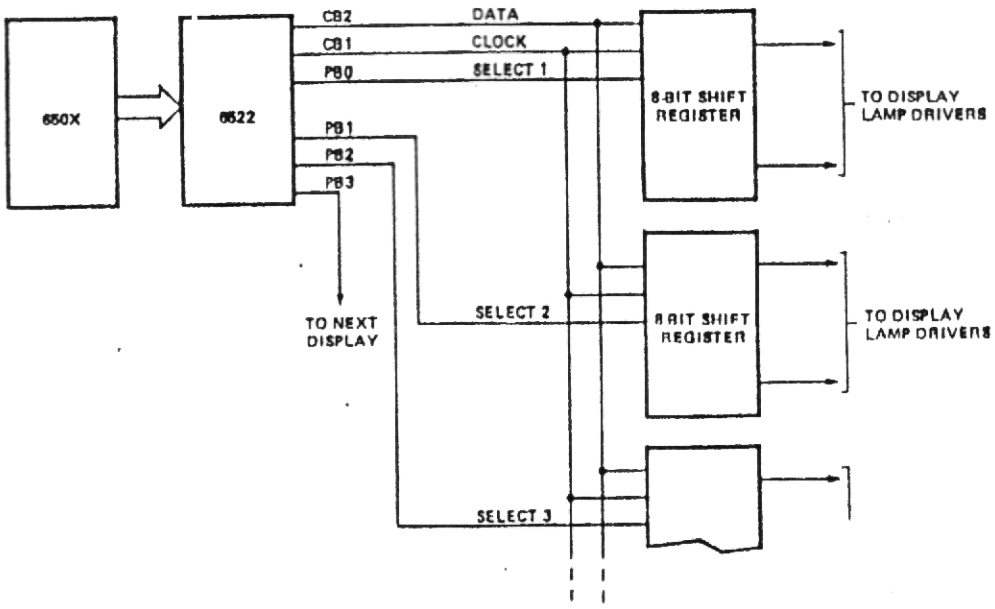


Figure 17. Shift Register for Remote Displays

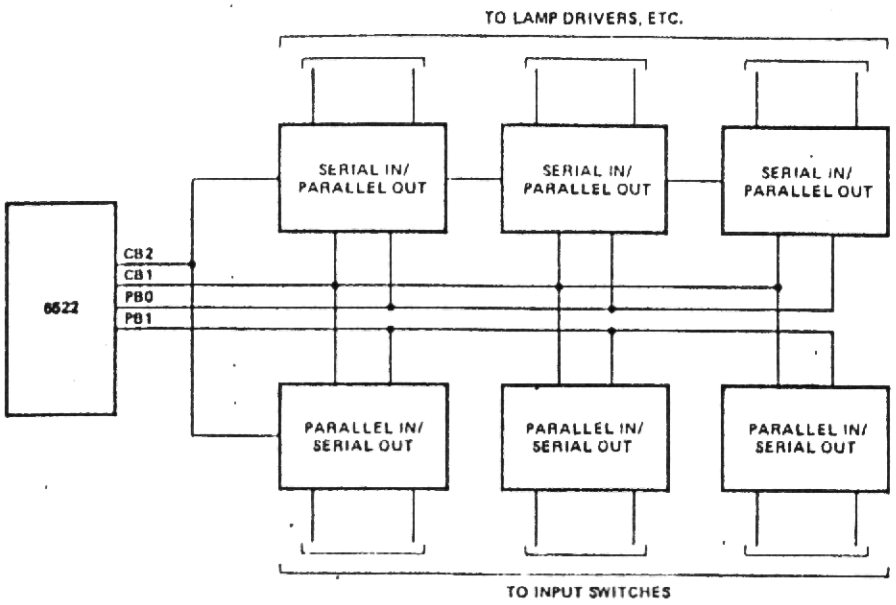


Figure 18. Using the Shift Register to Expand I/O Capabilities

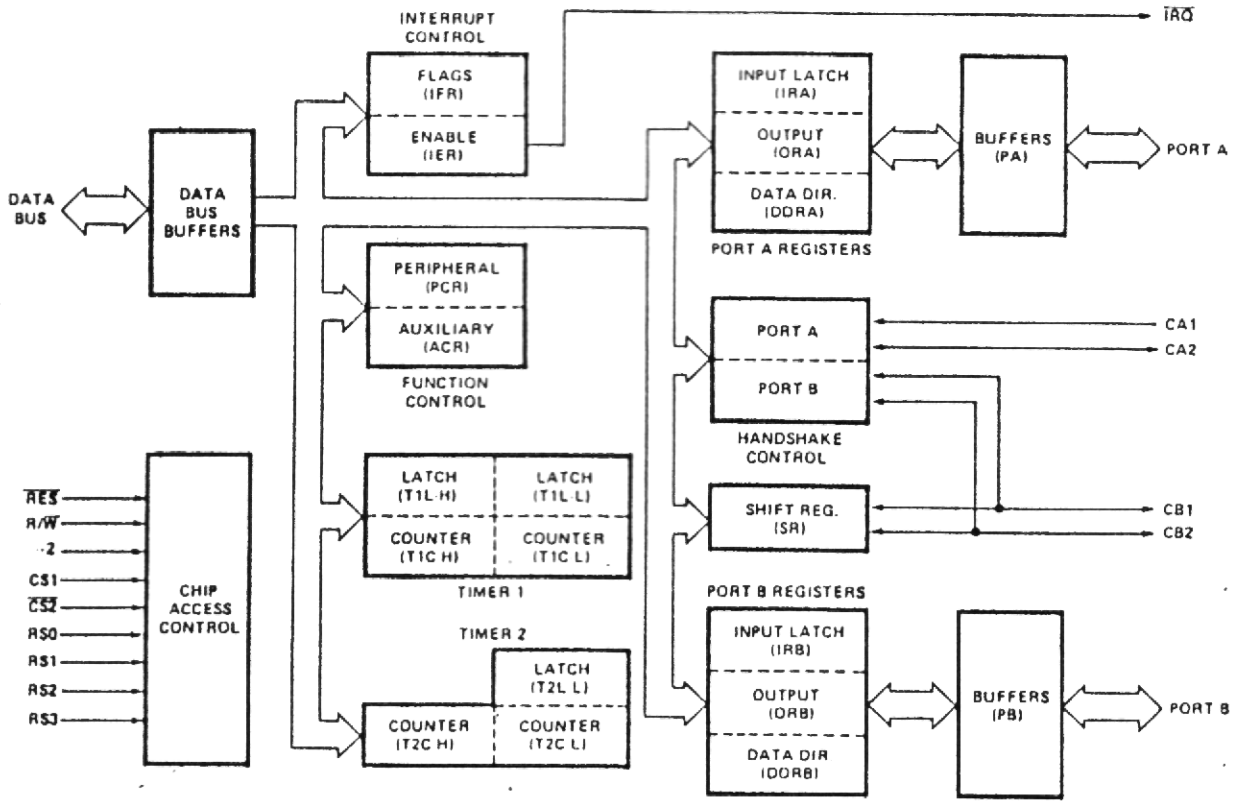


Figure 19.

Register Number	RS Coding				Register Desig.	Description	
	RS3	RS2	RS1	RS0		Write	Read
0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
2	0	0	1	0	DDRB	Data Direction Register "B"	
3	0	0	1	1	DDRA	Data Direction Register "A"	
4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
5	0	1	0	1	T1C-H	T1 High-Order Counter	
6	0	1	1	0	T1L-L	T1 Low-Order Latches	
7	0	1	1	1	T1L-H	T1 High-Order Latches	
8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
9	1	0	0	1	T2C-H	T2 High-Order Counter	
10	1	0	1	0	SR	Shift Register	
11	1	0	1	1	ACR	Auxiliary Control Register	
12	1	1	0	0	PCR	Peripheral Control Register	
13	1	1	0	1	IFR	Interrupt Flag Register	
14	1	1	1	0	IER	Interrupt Enable Register	
15	1	1	1	1	ORA/IRA	Same as Reg 1 Except No "Handshake"	

SY6522 Internal Register Summary

For further applications assistance direct inquiries to:

MICROPROCESSOR APPLICATIONS
Synertek
P.O. Box 552 MS60
Santa Clara, CA 95052
or Telephone (408) 988-5614.