TANGERINE

TANGERINE
COMPUTER SYSTEMS LIMITED

FOREWORD

TANEX is the second board in the **microtan** system. On this single board there are sufficient facilities to satisfy the needs of most users.

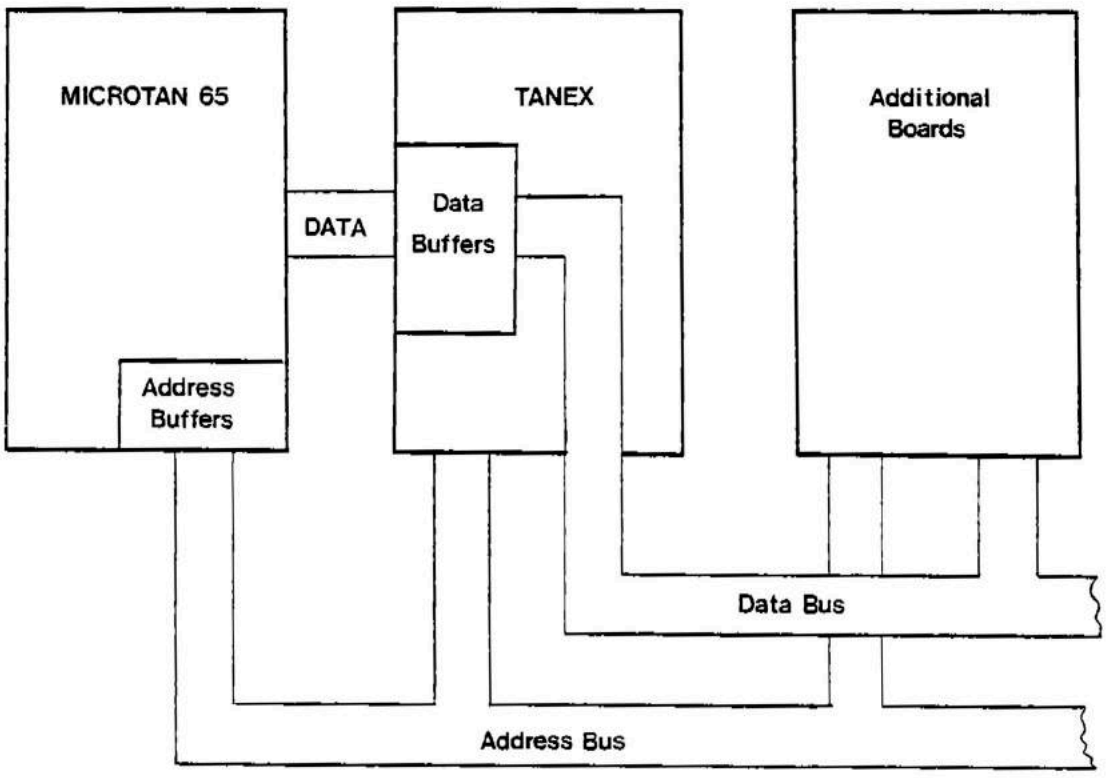
Featuring 7K RAM, 6K ROM, an 1ØK microsoft BASIC interpreter, parallel and serial I/O ports, and all the essential logic for memory mapping and data bus buffering, TANEX turns the **microtan** 65 into a very powerful and usable system.

If you have purchased TANEX in kit form then read chapter 8 before assembly; better still read the whole manual!

CHAPTER 1

Data Bus Buffering

The data bus on the microtan 65 is directly connected to the 65Ø2 microprocessor. In order to drive a complete system, it is necessary to buffer this bus, but to have done this on the microtan 65 would have increased the cost of production, apart from the more obvious problem of insufficient space on the printed circuit board. Data bus buffers are therefore placed on TANEX, as they are only a necessity when microtan is expanded into a system, and this results in the system shown in the block diagram below.



The data bus buffers are controlled by the memory mapping logic on TANEX as follows.

When the 65Ø2 microprocessor addresses memory located on the microtan 65 card, the buffers are disabled. Conversely, when memory which is external to the microtan 65 card is addressed, the buffers are enabled and the direction of data flow is determined by the

state of the R/W line.

The only restriction imposed by this method of bus buffering is that a device capable of Direct Memory Access (DMA), such as TANDISC, cannot read or write directly to the RAM, ROM, or I/O ports on the microtan 65 card. Examined in more depth, this restriction is of little consequence, since a DMA device is unlikely to want to read the TANBUG ROM, or to access any of the I/O ports on the microtan 65. In addition, this means that there is no way that a DMA device can accidentally overwrite TANBUG's system parameters, the processor stack, or the display memory, as these are located in the RAM on the microtan 65. The microtan system is therefore safe from system crashes which might otherwise be caused by a DMA transfer of data into this reserved memory area.

CHAPTER 2

Memory Mapping

Referring back to the microtan 65 manual, page 3-3, it is described there how microtan 65 operates with a very simple memory map of only three segments. When used in conjunction with TANEX, a more complex memory map is generated by the logic circuitry included on TANEX, which reconfigures the mapping as shown in the accompanying figure. In order to continue to use the RAM, ROM, and I/O on the microtan 65, three control lines are generated by TANEX; these are called $\overline{\text{RAME}}$, ROME and $\overline{\text{IOE}}$.

When the microprocessor addresses memory between locations Ø and 3FF (i.e. the lowest 1K of memory), the control line $\overline{\text{RAME}}$ is activated. By cutting the link LKRAM on the microtan 65, the 1K RAM on microtan can then be enabled by $\overline{\text{RAME}}$. When any other memory address is selected, $\overline{\text{RAME}}$ is inactive, and the microtan 65 RAM is disabled.

Addresses F8ØØ to FFFF (the top 2K of memory) cause ROME to be active in a similar way, and by cutting link LKROM on the microtan 65, this signal allows the ROM on microtan (TANBUG) to be enabled.

The third control line is $\overline{\text{IOE}}$, which become active for addresses in the range BFFØ to BFFF - the top 16 I/O addresses. Again, cutting link LKIO on the microtan 65 allows this signal to enable microtan's I/O ports. Note that $\overline{\text{IOE}}$ enables 16 I/O addresses although microtan only uses 4 I/O port addresses. Users should not use any of these 16 I/O addresses for self constructed peripherals. As described in the microtan manual, user peripherals should use addresses starting at BCØØ and work upwards.

Thus by cutting the three links LKRAM, LKROM and LKIO on the microtan 65, and connecting TANEX, full memory mapping of the microtan system is achieved. The RAM, ROM and I/O on TANEX itself are controlled by TANEX with no dependence on external

signals. The addresses of each of these segments of TANEX are described in the appropriate section.

To aid the decoding of I/O addresses, there is a control line on TANBUS called I/O, which is bussed to all slots on the system motherboard. This signal is active (TTL high) when any I/O port is addressed, and therefore obviates the need for other peripheral devices to decode a full 16 bit address - only the lower order ten bits need to be decoded.

ADDRESS	FUNCTION
FFFF F8ØØ	2K ROM (TANBUG)
F7FF FØØØ	2K ROM (XBUG)
EFFF E8ØØ	2K ROM (SPACE)
E7FF EØØØ	1ØK BASIC INTERPRETER TANEX
DFFF DØØØ	
CFFF CØØØ	
BFFF BCØØ	BFFØ - BFFF MICROTAN 65 I/O 1K I/O PORTS
BBFF 2ØØØ	4ØK RAM TANRAM
1FFF Ø4ØØ	7K RAM TANEX
Ø3FF ØØØØ	1K RAM ON MICROTAN 65

Fig 2·1 Full Memory Map of the Microtan System

CHAPTER 3

Random Access Memory

TANEX has provision for 7K of static RAM on board using the popular 2114 1K x 4 static RAM chip. The memory occupies addresses in the range 4ØØ to 1FFF.

Only two 2114's (i.e. 1K x 8) are supplied with TANEX in the minimum configuration, and these two chips should be inserted into locations N7 and N14 on the circuit board. Subsequent 2114's should be inserted in the ascending order indicated in Fig. 3.1, as this ensures that the available RAM is always in a contiguous block with the 1K of RAM located on the microtan 65. (No harm will be done by not following this recommendation if the user has particular need for non contiguous memory).

ADDRESS	FUNCTION	IC LOCATIONS	
1FFF ICØØ	1K TANEX RAM	N1	N8
18FF 18ØØ	1K TAMEX RAM	N2	N9
17FF 14ØØ	1K TANEX RAM	N3	N10
13FF 1ØØØ	1K TANEX RAM	N4	N11
ØFFF Ø8ØØ	1K TANEX RAM	N5	N12
ØAFF Ø8ØØ	1K TANEX RAM	N6	N13
Ø7FF Ø4ØØ	1K TANEX RAM	N7	N14
Ø3FF Ø2ØØ	VDU DISPLAY MEMORY	MICROTAN 65 RAM (1K) STACK OPERATES DOWN- WARDS FROM Ø1FF	
Ø1FF ØØ4Ø	MICROTAN 65 USER RAM		
ØØ3F ØØØØ	RESERVED FOR TANBUG		

Fig 3·1 RAM Segment Memory Map

CHAPTER 4

Read Only Memory

TANEX can accept up to three 2K x 8 EPROMS and the two 4K x 8 ROMS that contain microsoft BASIC. The BASIC interpreter (in two 2332 ROMS) is located in the address space from CØØØ to DFFF, and these two ROMS should be inserted into IC locations H2 and J2. Each ROM must be placed in its correct socket for the BASIC interpreter to operate, though no damage will ensue if they are accidentally transposed. Full information on this is included with the BASIC ROMS.

The three 2K byte EPROM positions accept the industry standard 2716 EPROM in its 5 volt only version. The IC locations and appropriate address range are shown in Fig. 4.1.

In order to make use of EPROM in the microtan system, the EPROM must be programmed with useful software. This could be the users own software, programmed into EPROM using an EPROM programmer, which itself may be either the microtan system programmer, or the users own. Alternatively, EPROMS containing proprietary software can be obtained from TANGERINE.

Interaction with TANBUG

The microtan system has been designed from the outset to be expandable, and TANBUG is no exception to this. Already powerful as a 1K monitor, it contains within itself the necessary code to expand by a further 2K, when used in conjunction with TANEX.

Referring back to the microtan 65 manual, page 6-21, there is described TANBUG's error linking procedure. When TANBUG receives a command that is not in its repetoire, the program executes a jump to location F7F7, which is in the top EPROM on TANEX. If no EPROM is installed, and the link LK1 is present, location F7F7 is decoded as FFF7, and the program continues, generating a '?' on the VDU. If the link LK1 is broken, address F7F7 is correctly decoded, and TANBUG's action will then depend upon the content of the EPROM in board location G2. If the user

wishes to use an EPROM in this socket, but without expanding TANBUG, then locations F7F7, F7F8 and F7F9 must be programmed with the instruction:

```
JMP FFF7
```

This will return TANBUG to the correct point. If the user does wish to expand TANBUG, then location F7F7 should have the instruction:

```
JMP USERBUG
```

where 'USERBUG' is the start address of the users expansion software that will action any new commands. At the end of the users expansion software, a normal return to TANBUG should be executed using the instruction:

```
RTS
```

This will generate a carriage return/linefeed and re-enter TANBUG. Alternatively, the user expansion software can end with the following instructions:

```
PLA
PLA
JMP FC37
```

This will return directly to TANBUG with no change to the display. If, however, an error situation is the result, for example, because of an illegal command, the program should execute the instruction:

```
JMP FC89
```

This will print a question mark and carriage return and restart TANBUG appropriately.

The methods given above all presume that the users software has left the stack pointer in the same position as it was when the entry was made via address F7F7 i.e. that there are an equal number of JSR and RTS instructions in every flow path of the users

expansion software.

Note that TANGERINE Computer Systems reserve the right to use this top EPROM location for the expanded system monitor, 'XBUG'.

ADDRESS	FUNCTION	LOCATION
FFFF F8ØØ	2K TANBUG	ON MICROTAN 65
F7FF FØØØ	2K EPROM	TANEX G2 RESERVED FOR "XBUG"
EFFF E8ØØ	2K EPROM	TANEX E2
E7FF EØØØ	2K BASIC EXTENSION	TANEX D3
DFFF CØØØ	8K BASIC INTERPRETER	TANEX H2 and J2

Fig 4·1 ROM Segment Memory Map

CHAPTER 5

The 6522 Versatile Interface Adaptor

TANEX uses the 6522 VIA to implement the following facilities:

2 8 bit bi-directional parallel data ports.

2 16 bit programmable timer/counters.

1 serial TTL data port.

In its minimum configuration, TANEX is supplied with one 6522, which should be located in IC socket A2. A second 6522 may be fitted as an option into IC socket C2. In the description that follows, references to IC locations and memory addresses will be given as in the following examples:

A) The 6522 in location A2 (C2).

B) The register at memory address BFCA (BFEA).

The first number will always relate to the standard 6522; the number in brackets refers to the second, optional, 6522.

It should be noted that the standard 6522 in location A2 is also used to implement the cassette recorder interface described in chapter 7 of this manual, although this does not restrict the user from any of the 6522's facilities when the cassette interface is not actually in use.

A block diagram of the 6522 is shown in Fig. 5.1, and it can be seen that despite the number of functions implemented within the 6522, the number of input/output pins available to the user is kept to a minimum by allowing a degree of flexibility. Each eight bit parallel data port has two control lines associated with it, available to the user to implement handshaking, input latching etc. When using the serial output, two of the control lines become serial I/O ports; one acting as the data pin, the other as a shift clock. In addition, one of the timer/counters can control one data line, for the generation of anything from simple square waves or pulses, up to complex programmable pulse waveforms. The other timer/counter can, in addition to normal timing functions, count pulses applied to one of the data lines.

To allow full control of all the functions, the 6522 has sixteen

internal registers; the detailed function of each of these is described in subsequent sections. An overall memory map of the registers, giving their memory addresses and mnemonic names is given in Fig. 5.2.

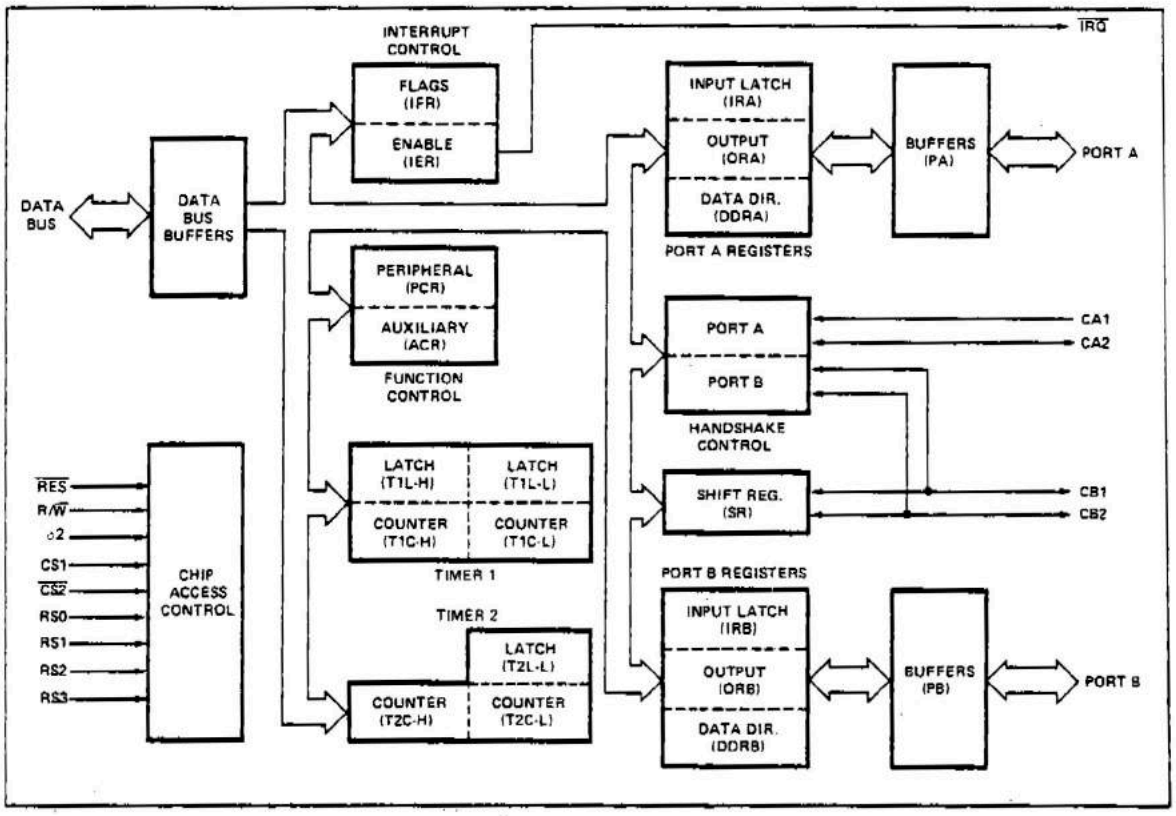


Fig 5·1 Block Diagram of 6522

USER CONNECTIONS

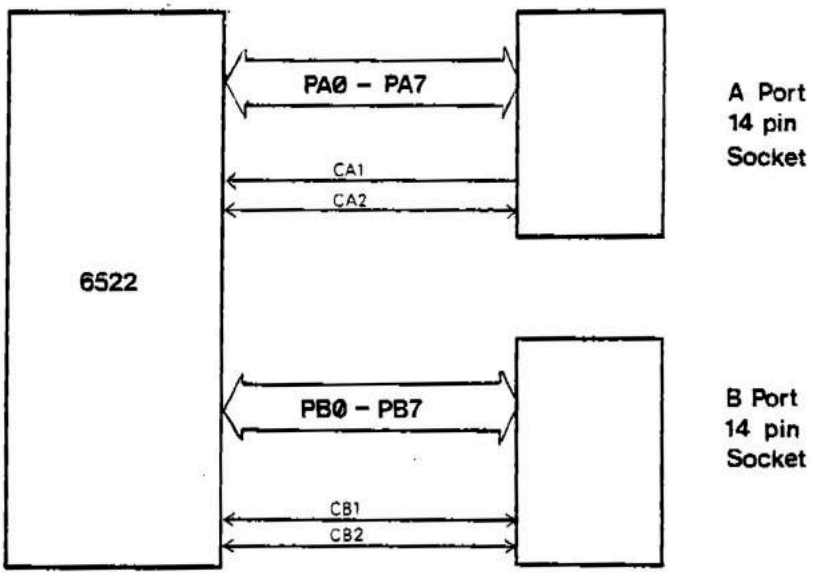
The user can connect his/her hardware to the 6522 using two 14 pin IC sockets in locations A1 and B1 (C1 and D1). A block diagram of the output signals, and the pin connections in the sockets is given in Fig. 5.3, and the signals are described in outline below.

PAØ-PA7 (Peripheral A Port)

The Peripheral A port consists of 8 lines which can be individually programmed to act as inputs or outputs under control of a Data Direction Register. The polarity of output pins is controlled by

MEMORY ADDRESS		READ	WRITE	MNEMONIC
BFCF	(BFEF)	Same as IRA without handshake	Same as ORA without handshake	
BFCE	(BFEE)	Interrupt Enable Register	Interrupt Enable Register	IER
BFCD	(BFED)	Interrupt Flag Register	Interrupt Flag Register	IRF
BFCC	(BFEC)	Peripheral Control	Peripheral Control	PCR
BFCB	(BFEB)	Auxiliary Control	Auxiliary Control	ACR
BFCA	(BFEA)	Shift Register	Shift Register	SR
BFC9	(BFE9)	Timer 2 Counter High Byte	Timer 2 Counter High Byte	T2C-H
BFC8	(BFE8)	Timer 2 Counter Low Byte	Timer 2 Latch Low Byte	T2C-L
BFC7	(BFE7)	Timer 1 Latch High Byte	Timer 1 Latch High Byte	T1L-H
BFC6	(BFE6)	Timer 1 Latch Low Byte	Timer 1 Latch Low Byte	T1L-L
BFC5	(BFE5)	Timer 1 Counter High Byte	Timer 1 Counter High Byte	T1C-H
BFC4	(9FE4)	Timer 1 Counter Low Byte	Timer 1 Latch Low Byte	T1C-L
BFC3	(BFE3)	Data Direction Port A	Data Direction Port A	DDRA
BFC2	(BFE2)	Data Direction Port B	Data Direction Port B	DDRB
BFC1	(BFE1)	Parallel Port A	Parallel Port A	IRA/ORA
BFCØ	(BFEØ)	Parallel Port B	Parallel Port B	IRB/ORB

Fig 5·2 Register Addresses of the 6522



BOTH SOCKETS HAVE THE SAME PIN-OUTS, AND ARE AS SHOWN BELOW.

PIN NUMBER	'A' SIGNALS at socket A1 (C1)	'B' SIGNALS at socket B1 (D1)	SIGNAL SENSE
1	+5 volts	+5 volts	-
2	PAØ	PBØ	In/Out
3	PA1	PB1	In/Out
4	PA2	PB2	In/Out
5	PA3	PB3	In/Out
6	PA4	PB4	In/Out
7	Ground	Ground	
8	Ground	Ground	
9	PA5	PB5	In/Out
1Ø	PA6	PB6	In/Out
11	PA7	PB7	In/Out
12	CA2	CB2	In/Out
13	CA1	CB1	Input only In/Out
14	+5 volts	+5 volts	

Fig 5·3 Parallel I/O Connections

an Output Register and input data may be latched into an internal register under control of the CA1 line. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. The A port is socket location A1 (C1).

CA1, CA2 (Peripheral A Control Lines)

These two control lines have multiple functions. Both lines can act as interrupt inputs; with corresponding interrupt flags and enable bits. In addition, CA1 controls the latching of input data on PAØ-PA7, and CA1 with CA2 can implement a handshake control for data exchange. CA2 may also be 'manually' set to high or low in the manner of a data output. CA1 is a high impedance input. CA2 is one standard TTL load when in input mode, and will drive one TTL load when an output.

PBØ-PB7 (Peripheral B Port)

The Peripheral B port consists of eight bi-directional lines which are controlled by an output register and a data direction register in much the same manner as the PA port. In addition, the polarity of the PB7 output signal can be controlled by one of the interval timers while the second timer can be programmed to count pulses on the PB6 pin. Peripheral B lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. In addition, they are capable of sourcing 3.ØmA at 1.5VDC in the output mode to allow the outputs to directly drive Darlington transistor circuits. The B port is socket location B1 (D1).

CB1, CB2 (Peripheral B Control Lines)

The Peripheral B control lines act as interrupt inputs or as handshake outputs. As with CA1 and CA2, each line controls an interrupt enable bit. In addition, these lines act as a serial port under control of the Shift Register. These lines represent one standard TTL load in the input mode and will drive one standard TTL load in the output mode. Unlike PBØ-PB7, CB1 and CB2 cannot drive Darlington transistor circuits.

PARALLEL DATA PORT OPERATION

Referring to the block diagram, Fig. 5.1, it can be seen that each 8 bit peripheral port has a Data Direction Register (DDRA, DDRB) for specifying whether the peripheral pins are to act as inputs or outputs. A Ø in a bit of the Data Direction Register causes the corresponding peripheral pin to act as an input. A 1 causes the pin to act as an output.

Each peripheral pin is also controlled by a bit in an Output Register (ORA, ORB) and an Input Register (IRA, IRB). When the pin is programmed as an output, the voltage on the pin is controlled by the corresponding bit of the Output Register. A 1 in the Output Register causes the output to go high, and a Ø causes the output to go low. Data may be written into Output Register bits corresponding to pins which are programmed as inputs. In this case, however, the output pin is unaffected.

Reading a peripheral port causes the contents of the Input Register (IRA, IRB) to be transferred onto the Data Bus. With input latching disabled, IRA will always reflect the levels on the PA pins. With input latching enabled, IRA will reflect the levels on the PA pins at the time that CA1 received the latching signal.

The IRB register operates similarly to IRA, however, for pins programmed as outputs there is a slight difference. Whereas reading IRA results in the actual level at the PA pin being sensed, reading IRB results in the relevant bits of the output register, ORB, being sensed. The data direction registers and input/output registers are depicted in Fig. 5.4a and Fig. 5.4b.

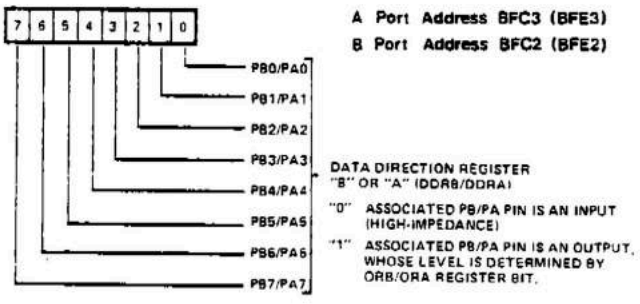
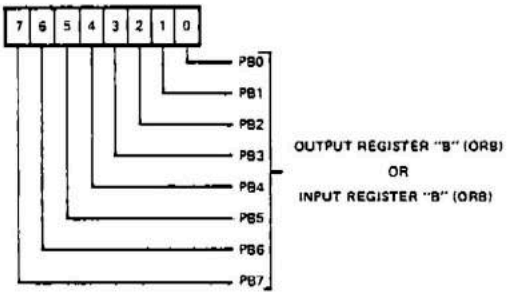


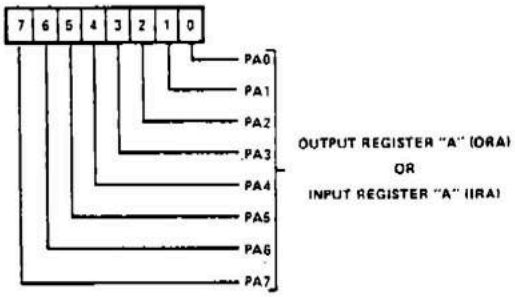
Fig 5·4a Data Direction Register

ORB/IRB Address BFC0 (BFE0)



Pin Data Direction Selection	WRITE	READ
DDRB = "1" (OUTPUT)	MPU writes Output Level (ORB)	MPU reads output register bit in ORB. Pin level has no affect.
DDRB = "0" (INPUT) (Input latching disabled)	MPU writes into ORB, but no effect on pin level, until DDRB changed.	MPU reads input level on PB pin.
DDRB = "0" (INPUT) (Input latching enabled)		MPU reads IRB bit, which is the level of the PB pin at the time of the last CB1 active transition.

ORA/IRA Address BFC1 (BFE1)



Pin Data Direction Selection	WRITE	READ
DDRA = "1" (OUTPUT) (Input latching disabled)	MPU writes Output Level (ORA).	MPU reads level on PA pin.
DDRA = "1" (OUTPUT) (Input latching enabled)		MPU reads IRA bit which is the level of the PA pin at the time of the last CA1 active transition.
DDRA = "0" (INPUT) (Input latching disabled)	MPU writes into ORA, but no effect on pin level, until DDRA changed.	MPU reads level on PA pin.
DDRA = "0" (INPUT) (Input latching enabled)		MPU reads IRA bit which is the level of the PA pin at the time of the last CA1 active transition

Fig 5·4b Input/Output Registers

Handshake Control of Data Transfers

The 6522 allows positive control of data transfers to and from peripheral devices through the operation of "handshake" lines. Port A lines (CA1, CA2) handshake data on both a read and a write operation while the Port B lines (CB1, CB2) handshake on a write operation only.

Read Handshake:- Positive control of data transfers from peripheral devices can be accomplished very effectively using Read Handshaking. In this case, the peripheral device must generate a "Data Ready" signal signifying that valid data is present on the peripheral port. This signal normally interrupts the microprocessor, which then reads the data, causing generation of a "Data Taken" signal. The peripheral device responds by making new data available. This process continues until the data transfer is complete.

In the 6522, automatic "Read" Handshaking is possible on the Peripheral A port only. The CA1 interrupt input pin accepts the

"Data Ready" signal and CA2 generates the "Data Taken" signal. The "Data Ready" signal will set an internal flag which may interrupt the processor or which may be polled under program control. The "Data Taken" signal can either be a pulse, or a level which is set low when the microprocessor reads the input data and is cleared by the next "Data Ready" signal. These options are shown in Fig. 5.5 which illustrates the normal Read Handshaking sequence.

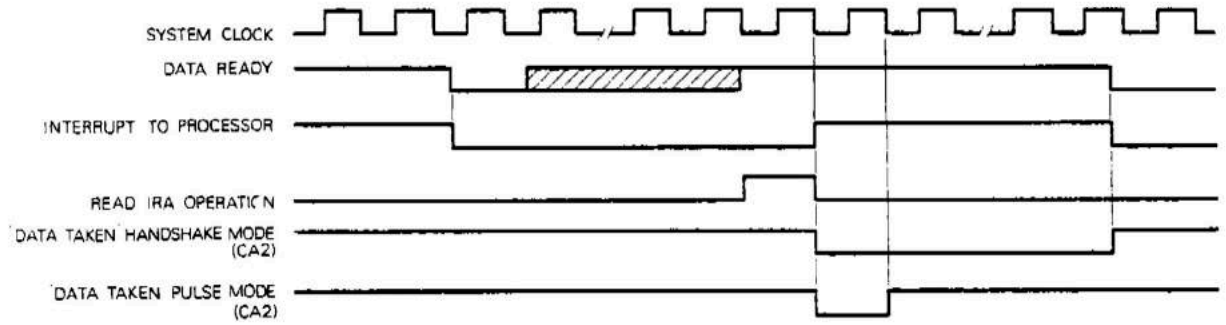
Write Handshake:- The sequence of operations which allows handshaking data from microprocessor to a peripheral device is very similar to that described for Read Handshaking. For Write Handshaking, the 6522 generates the "Data Ready" signal and the peripheral device must respond with the "Data Taken" signal. This can be accomplished on both the PA port and the PB port on the 6522. CA2 or CB2 act as a "Data Ready" output in either the handshake mode or pulse mode and CA1 or CB1 accept the "Data Taken" signal from the peripheral device, which sets the interrupt flag and clears the "Data Ready" output. This sequence is shown in Fig. 5.5.

Selection of operating modes for CA1, CA2, CB1, and CB2 is accomplished by the Peripheral Control Register, and by the two bits in the Auxiliary Control Register. The format of all these registers is given in Fig. 5.6 and Fig. 5.7.

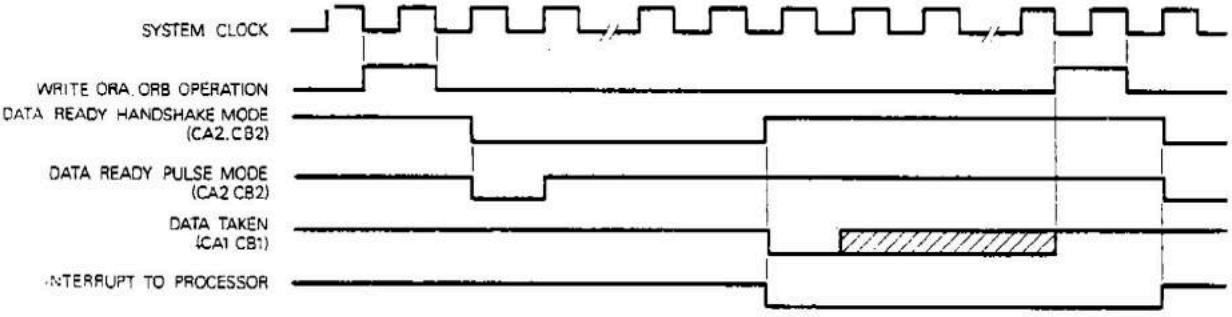
Summary of Parallel Port Operating Modes

1) As a simple output with DDRA(B) = FF
2) As an unlatched input with DDRA(B) = Ø, ACRØ and ACR1 = Ø
3) As a latched input using CA1 with DDRA(B) = Ø, ACRØ and ACR1 = 1, positive or negative edge latched (PCRØ, PCR4)
4) As a handshake interface in two modes (level handshake or pulse handshake) using CA1 and CA2 (CB1 and CB2)
5) If CA2 (or CB2) is not in use for handshaking, it can be used as an additional data output, or as an independent interrupt input.

Note: Use of CB1 and CB2 as input/output controls assumes that the shift register is disabled i.e. ACR4,3,2 are set to Ø.



5·5a Read Handshake Timing (Port A Only)



5·5b Write Handshake Timing (Port A and Port B)

AUXILIARY CONTROL REGISTER - ADDRESS BFCB (BFEB)

BIT0 PA INPUT LATCH CONTROL: 0 = Latch disabled. 1 = Latch enabled.
BIT1 PB INPUT LATCH CONTROL: 0 = Latch disabled. 1 = Latch enabled.
BITS 4, 3 and 2 SHIFT REGISTER CONTROL:
(000) disabled.
(001) shift in, timed by timer 2.
(010) shift in, timed by system clock.
(011) shift in, timed by external clock.
(100) shift out, free running at timer 2 rate.
(101) shift out, timed by timer 2.
(110) shift out, timed by system clock.
(111) shift out, timed by external clock.
BIT5 TIMER 2 CONTROL: 0 = Single timed interrupt. 1 = Count down with pulses input on PB6.
BITS 7 and 6 TIMER 1 CONTROL:
(00) single timed interrupt, PB7 output disabled.
(01) continuous timed interrupts, PB7 output disabled.
(10) single timed interrupt, single pulse output to PB7*.
(11) continuous interrupts, PB7 inverted on each interrupt*.

* Note: For timer 1 to output to PB7, both DDR87 and ACR7 must be set. The level of ORB7 has no effect under these conditions.

Fig 5·6 Auxiliary Control Register

PERIPHERAL CONTROL REGISTER - ADDRESS BFCC (BFEC)

BITØ CA1 INTERRUPT CONTROL: Ø = Negative active edge. 1 = Positive active edge.
BITS 3, 2 and 1 CA2 CONTROL:

(ØØØ)	Input: set the CA2 interrupt flag on negative edge. Clear flag by reading IRA/ORA.
(ØØ1)	Input: set flag CA2 on negative edge. Clear flag by writing to IFR register.
(Ø1Ø)	Input: as for code ØØØ, with positive edge.
(Ø11)	Input: as for code ØØ1, with positive edge.
(1ØØ)	Output: handshake mode (active low). Reset by active transition of CA1 input.
(1Ø1)	Output: pulse mode (active low).
(11Ø)	Manual Output: CA2 held low.
(111)	Manual Output: CA2 held high.

BIT4 CB1 INTERRUPT CONTROL: Code as for CA1.
BITS 7, 6 and 5 CB2 CONTROL: Codes as for CA2.

Note 1: The specified active edge for CA1, CB1 controls both the active edge for interrupt and the active edge for input latching when enabled.

Note 2: Codes ØØ1 and ØØ3 for CA2 and CB2 allow this input to be used as a completely independent interrupt; the Shift Register must be disabled to use CB1/CB2 as input/output.

Fig 5·7 Peripheral Control Register

TIMER-COUNTER OPERATION

Timer T1 Operation

Timer T1 consists of two 8 bit latches and a 16 bit counter (see Fig. 5.1). The latches are used to store data which is to be loaded into the counter. After loading, the counter decrements at the system clock rate of $\frac{3}{4}$MHz. Upon reaching zero, an interrupt flag will be set, and the microprocessor will be interrupted if the appropriate enable bit is set in the IER register. The timer will then disable any further interrupts, or will automatically transfer the contents of the latches into the counter and will continue to decrement. In addition, the timer may be programmed to invert the output signal on a peripheral pin each time it "times-out". Each of these modes is selected via 2 bits in the ACR (Fig. 5.5), and are fully described below.

The T1 registers are depicted in Fig. 5.8, and are summarised

in the table below:

Address	Mnemonic	Write	Read
BFC4	T1C-L	Load counter to low byte of latch.	Read low byte of counter. reset T1 interrupt flag.
BFC5	T1C-H	Load high byte latch and transfer count from both latches to counter (16 bits) and initiate counting. reset T1 flag.	Read high byte of counter.
BFC6	T1L-L	Load low byte latch (same as BFC4).	Read low byte of data latch.
BFC7	T1L-H	Load high byte of data latch.	Read high byte of data latch.

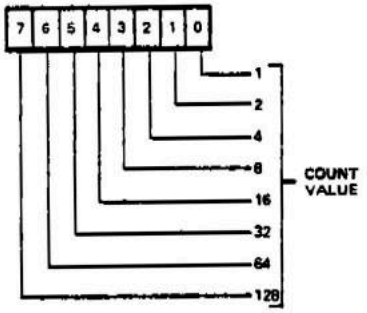
Timer 1 - One Shot Mode

The interval timer one-shot mode allows generation of a single interrupt for each timer load operation. As with any interval timer, the delay between the "write T1C-H" operation and generation of the processor interrupt is a direct function of the data loaded into the timing counter. In addition to generating a single interrupt, Timer 1 can be programmed to produce a single negative pulse on the PB7 peripheral pin. With the output enabled (ACR7 = 1 and DDRB7 = 1), a "write T1C-H" operation will cause PB7 to go low. PB7 will return high when Timer 1 times out. The result is a single, programmable width, pulse.

In the one-shot mode, writing into the high order latch has no effect on the operation of Timer 1. However, it will be necessary to ensure that the low order latch contains the proper data before initiating the count-down with a "write T1C-H" operation. When the processor writes into the high order counter, the T1 interrupt flag will be cleared, the contents of the low order latch will be transferred into the low order counter, and the timer will begin to decrement at the system clock rate of $\frac{3}{2}$MHz. If the PB7 output

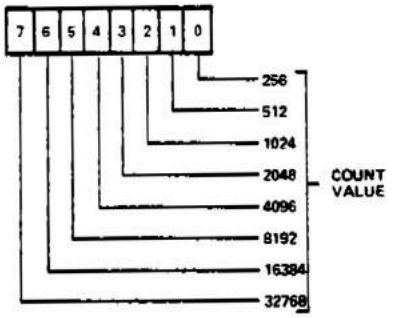
Address BFC4 (BFE4)

REG 4 – TIMER 1 LOW-ORDER COUNTER



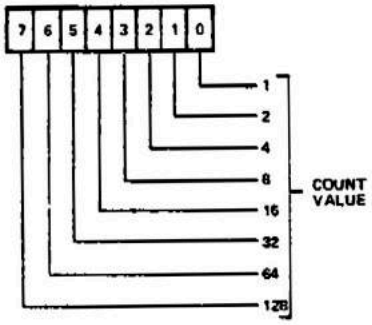
Address BFC5 (BFE5)

REG 5 – TIMER 1 HIGH-ORDER COUNTER



Address BFC6 (BFE6)

REG 6 – TIMER 1 LOW-ORDER LATCHES



Address BFC7 (BFE7)

REG 7 – TIMER 1 HIGH-ORDER LATCHES

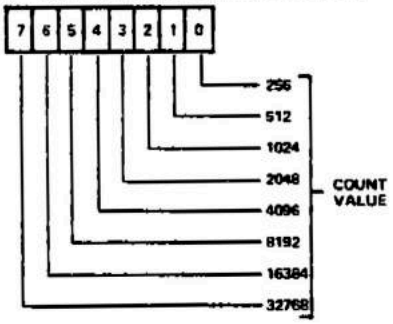


Fig 5·8 Timer 1 Registers

is enabled, this signal will go low on the phase two following the write operation. When the counter reaches zero, the T1 interrupt flag will be set, and the processor interrupted (if interrupt is enabled), and the signal on PB7 will go high. At this time the counter will continue to decrement at system clock rate. This allows the processor to read the contents of the counter to determine the time since interrupt. However, the T1 interrupt flag cannot be set again unless it has been cleared as described in this specification.

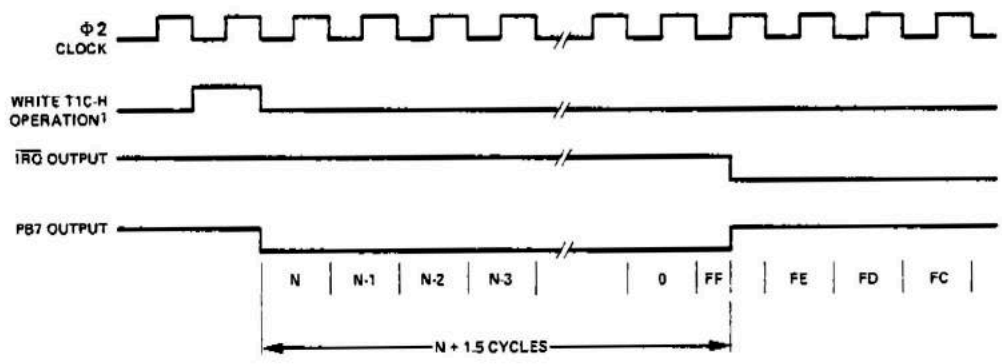
Timing for the one-shot mode is shown in Fig. 5.9.

Timer 1 Free-Run Mode

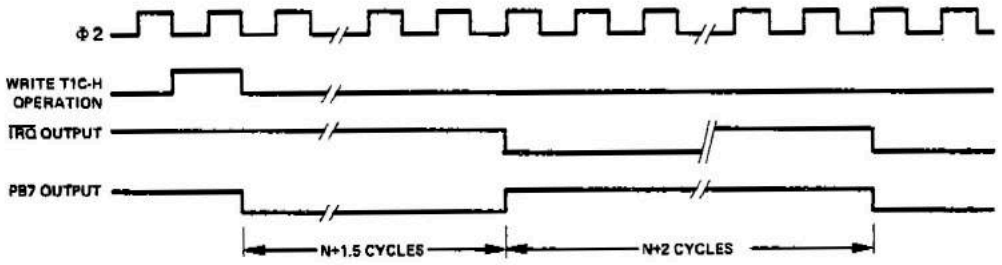
The most important advantage associated with the latches in T1 is the ability to produce a continuous series of evenly spaced interrupts and the ability to produce a square wave on PB7 whose frequency is not affected by variations in the processor interrupt response time. This is accomplished in the "free-running" mode.

In the free-running mode, 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 latches into the counter (16 bits) and continues 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 later. However, it is not necessary to rewrite to the timer to enable setting the interrupt flag for the next time-out.

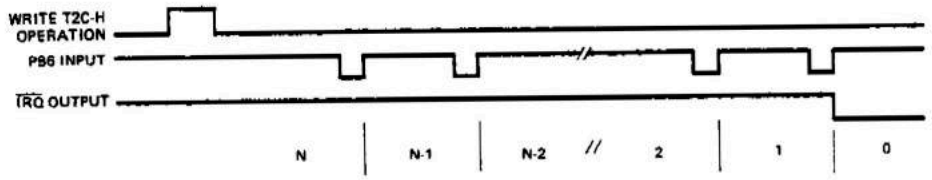
Both interval timers in the 6522 are "re-triggerable", that is re-writing to the counter will always re-initialize the time-out period. In fact, the time-out can be prevented completely if the processor continues to rewrite the timer before it reaches zero. Timer 1 will operate in this manner if the 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



INTERVAL TIMER "ONE-SHOT" MODE TIMING SEQUENCE



TIMER 1 "FREE-RUNNING" MODE



TIMER 2 PULSE COUNTING MODE

Fig 5·9 Counter-Timer Operating Modes

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 alter the period of the next half cycle during each half cycle of the output signal on PB7. In this manner, very complex waveforms can be generated. Timing for the free-running mode is shown in Fig. 5.9.

Timer 2 Operation

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 has data registers 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 the system clock rate. Fig. 5.10 illustrates the T2 Counter Registers.

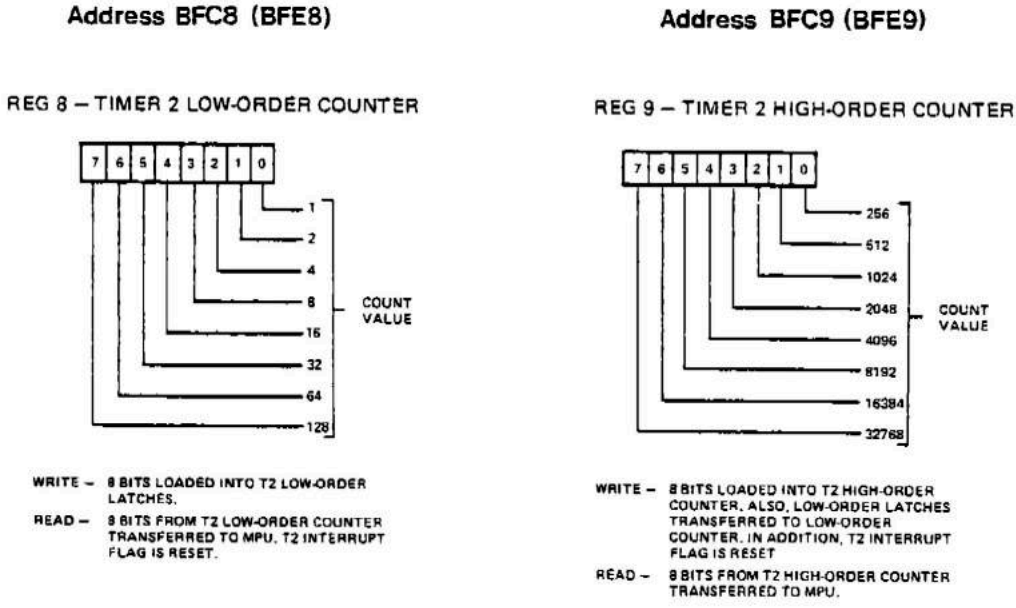


Fig 5·10 T2 Counter Registers

Timer 2 One-Shot Mode

As an interval timer, T2 operates in the "one-shot" mode similarly 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 the initial time-out so that it will not be set by the counter continuing to decrement through zero. The processor must rewrite T2C-H, which re-loads the counter, and re-enables the T2 interrupt. The T2 interrupt flag is cleared by reading T2C-L or by writing T2C-H. Timing for this operation is shown in Fig. 5.9.

Timer 2 Pulse Counting Mode

In the pulse counting mode, T2 serves primarily to count a predetermined number of low 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 to rewrite T2C-H to allow the interrupt flag to set the next time T2 reaches zero. Timing for this mode is shown in Fig. 5.9. The input pulse must be low on the leading edge of the system clock (i.e. it must be wider than the period of the Ø2 clock).

APPLICATIONS OF THE INTERVAL TIMERS

A Real-Time Clock

The function of a real-time clock is to allow the microprocessor to keep a count of hours, minutes and seconds. With such knowledge of "real-time", the microprocessor can then control any task, or peripheral equipment, that needs to operate at a particular time, or times, of day. The microprocessor could be programmed, for example, to turn on the television, or radio, at certain times of the day; to control the house lights; or to put messages on the VDU screen, such as "GOOD MORNING", "COFFEE TIME", "TIME TO MAKE LUNCH" etc.

In order to do this, Timer 1 would be used in its free running mode to provide interrupts at fixed intervals which are easily related to seconds; in addition, the interrupts need to be fairly slow so as not to tie up the processor inordinately. If 2Ø milliseconds is chosen as the time interval, then Timer 1 needs a count of 15ØØØ (decimal) with microtan's system clock rate of $\frac{3}{2}$MHz. Thus the user would set the registers as follows:-

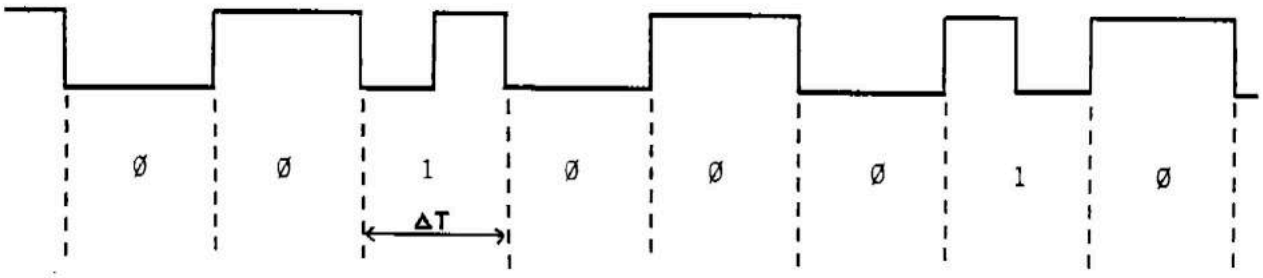
Auxiliary Control Register: bit 7 = Ø, bit 6 = 1 (continuous interrupts).
Timer 1 T1L-L = 8A (Hex) low byte of period.
Timer 1 T1L-H = 3D (Hex) high byte of period.
Timer 1 T1C-H = 3D loads counters and commences timer operation.

In order for interrupts to occur, the user must set the appropriate bit in the interrupt enable register IER. The users interrupt routine must then perform the actual conversion to hours, minutes, seconds by maintaining a count in locations in memory.

Bi-phase Encoded Data

As an example of using the T1 output via PB7, consider serial output of data in the following format:

Binary data ØØ1ØØØ1Ø to be output.



As can be seen, each bit occupies a time ΔT, and at the end of each bit, the signal changes state. If the bit is a 1, an additional state change occurs in the middle of the bit period. This is called bi-phase, or two frequency data, and is popular for use on disc and tape memories. The method would work as follows. Timer 1 is set to free running mode with output to PB7 enabled, and period ΔT set in T1L-L. If the first bit is zero, as here,

T1C-H is loaded with zero, which triggers the timer for a count of ΔT. During this interval, the second bit is tested; if it is zero, (as here) no action need be taken, as the timer will reload the period ΔT. If the subsequent bit is a 1, then the processor loads period ΔT/2 into T1L-L, so that output PB7 changes state at twice the rate. In the latter part of this cycle, the next bit is inspected, and so the process goes on. The process is thus:

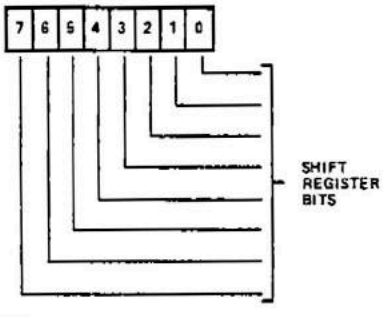
1) Set up timer for first bit period.
2) Inspect next bit: if it has changed load T1L-L with new count.
   : if it is the same as the previous one, leave T1L-L alone.

SHIFT REGISTER OPERATION - TTL SERIAL I/O PORT

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 external devices.

The control bits which select the various shift register operating modes are located in the Auxiliary Control Register, as shown in Fig. 5.6. The shift register itself is depicted below.

Address BFCA (BFEA)



NOTES:
1. WHEN SHIFTING OUT, BIT 7 IS THE FIRST BIT OUT AND SIMULTANEOUSLY IS ROTATED BACK INTO BIT 0.
2. WHEN SHIFTING IN, BITS INITIALLY ENTER BIT 0 AND ARE SHIFTED TOWARDS BIT 7.

The eight modes of operation of the shift register are now described.

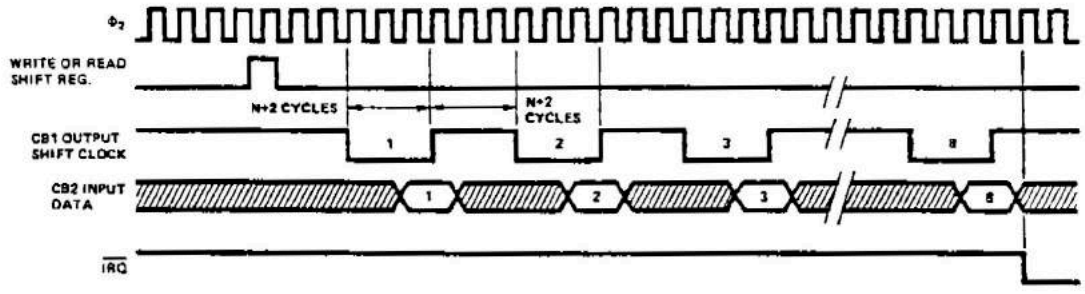
SR Disabled (ØØØ)

The ØØØ mode is used to disable the Shift Register. In this mode the microprocessor can write or read the SR, but the shifting operation 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 Ø).

Shift in Under Control of T2 (ØØ1)

In the ØØ1 mode the shifting rate is controlled by the low order 8 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 (N).

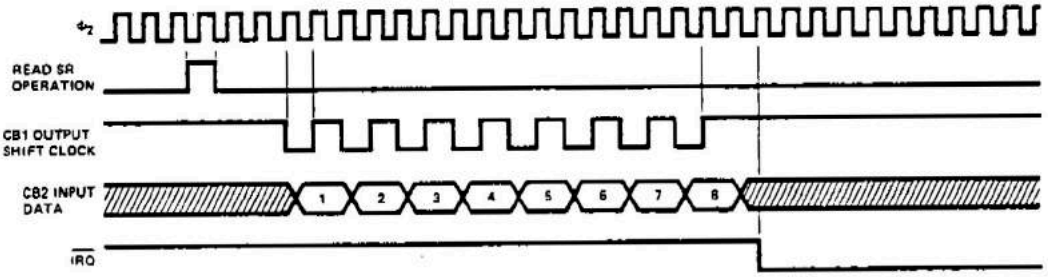
The shifting operation is triggered by writing or reading the shift register. Data is shifted first into the low order bit of SR and is then shifted into the next higher order bit of the shift register on the negative-going edge of each clock pulse. The input data should change before the positive-going edge of the CB1 clock pulse. This data is shifted into the shift register during the phase 2 clock cycle following the positive-going edge of the CB1 clock pulse. After 8 CB1 clock pulses, the shift register interrupt flag will be set and IRQ will go low.



Shift in Under Control of System Clock (Ø1Ø)

In mode Ø1Ø the shift rate is a direct function of the system clock frequency. CB1 becomes an output which generates shift pulses for controlling external devices. Timer 2 operates as an inde-

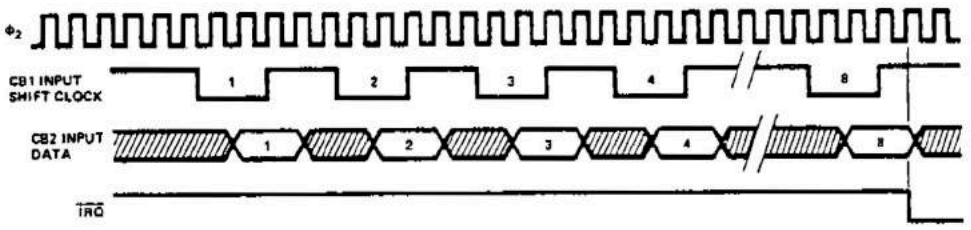
pendent interval timer and has no effect on SR. The shifting operation is triggered by reading or writing the Shift Register. Data is shifted first into bit Ø and is then shifted into the next higher order bit of the shift register on the trailing edge of each phase 2 clock pulse. After 8 clock pulses, the shift register interrupt flag will be set, and the output clock pulses on CB1 will stop.



Shift in Under Control of External CB1 Clock (Ø11)

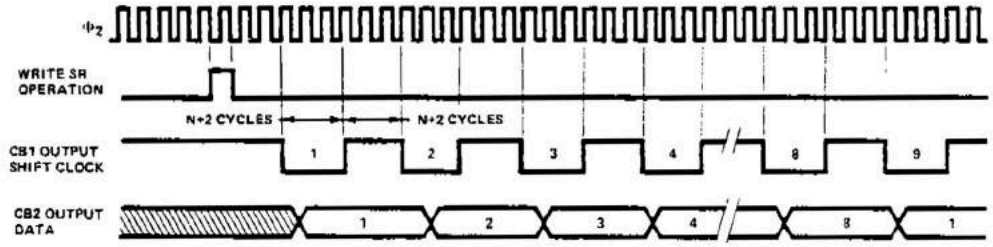
In mode Ø11 CB1 becomes an input. This allows an external device to load the shift register at its own pace. The shift register counter will interrupt the processor each time 8 bits have been shifted in. However, the shift register counter does not stop the shifting operation; it acts simply as a pulse counter. Reading or writing the Shift Register resets the interrupt flag and initializes the SR counter to count another 8 pulses.

Note that this data is 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.



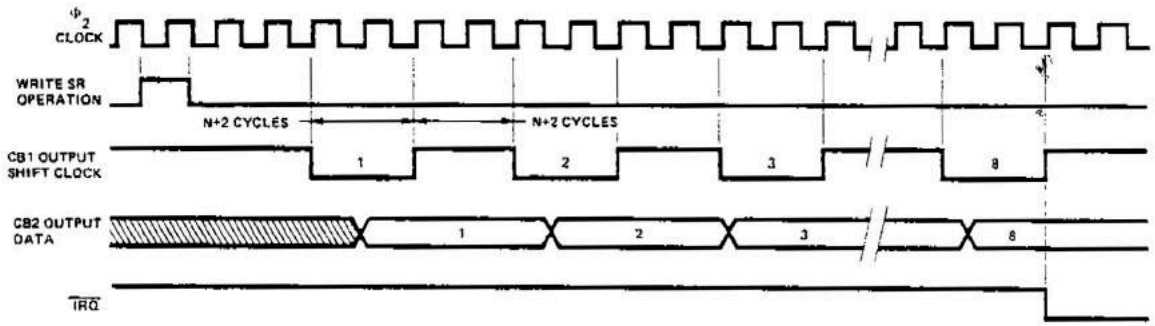
Shift Out Free-Running at T2 Rate (1ØØ)

Mode 1ØØ is very similar to mode 1Ø1 in which the shifting rate is set by T2. However, in mode 1ØØ the SR Counter does not stop the shifting operation. Since the Shift Register bit 7 (SR7) is recirculated back into bit Ø, the 8 bits loaded into the shift register will be clocked onto CB2 repetitively. In this mode the shift register counter is disabled.



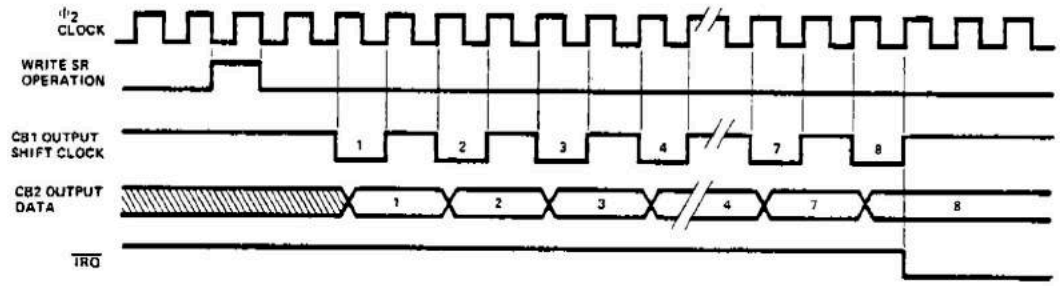
Shift Out Under Control of T2 (1Ø1)

In mode 1Ø1 the shift rate is controlled by T2 (as in the previous mode). 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, 8 shift pulses are generated on CB1 to control shifting in external devices. After the 8 shift pulses, the shifting is disabled, the SR Interrupt Flag is set and CB2 remains at the last data level.



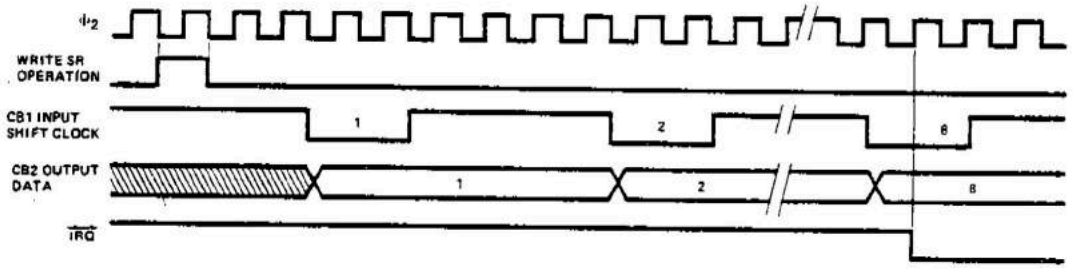
Shift Out Under Control of System Clock (11Ø)

In mode 11Ø, the shift rate is controlled by the phase 2 system clock.



Shift Out Under Control of External CB1 Clock (111)

In mode 111 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 8 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 counting the next 8 shift pulses on pin CB1. After 8 shift pulses, the interrupt flag is set. The microprocessor can then load the shift register with the next byte of data.



INTERRUPT OPERATION

Controlling interrupts within the 6522 involves three principle operations. These are flagging the interrupts, enabling interrupts and signaling to the processor that an active interrupt exists within the chip. Interrupt flags are set by interrupting conditions which exist within the 6522 or on inputs to the 6522. These flags normally remain set until the interrupt has been serviced. To determine the source of an interrupt, the microprocessor

must examine the flag register bits in order from highest to lowest priority. This is accomplished by reading the flag register 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. These can be set or cleared by the processor to allow the corresponding interrupt flag to interrupt the processor. 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 (IRQ) will go active. The microtan 65 will respond to this interrupt as described in the microtan manual, page 6-19, and the user must modify the interrupt linkage as described there in order to respond to the 6522.

In the 6522, the seven 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 allows the microprocessor to ascertain the existence of an interrupt condition with a single conditional branch instruction. The interrupt registers are shown below:

	7	6	5	4	3	2	1	Ø	Address
IFR	IRQ	T1	T2	CB1	CB2	SR	CA1	CA2	BFCD (BFED)
IER	Set/ Clear	T1	T2	CB1	CB2	SR	CA1	CA2	BFCE (BFEE)

The conditions for setting/clearing the IFR flags are shown below:

BIT	SET BY	CLEARED BY
Ø	An Active Transition on CA2	Read/Write to ORA/IRA*
1	An Active Transition on CA1	Read/Write to ORA/IRA
2	Completion of 8 shifts	Read/Write to SR
3	Active Transition on CB2	Read/Write to ORB/IRB*
4	Active Transition on CB1	Read/Write to ORB/IRB
5	Time out of T2	Read T2C-L, or write T2C-H
6	Time out of T1	Read T1C-L, or write T1C-H
7	Any enabled interrupt	Clearing all interrupts

* except in certain modes (see Fig. 5.7, PCR)

In addition to the flag clearing methods shown above, any flag may be cleared by writing a 1 to the appropriate bit of the IFR.

Interrupt Control

To enable the selected interrupts, the respective bits in the enable register must be set. This is accomplished by writing to the IER with a 1 in bit 7, and a 1 in each bit which is to be enabled.

e.g. 1ØØ1Ø1ØØ enables the CB1 and SR interrupts.

To disable these, the same byte is written to IER, but with bit 7 zero.

e.g. ØØØ1Ø1ØØ disables the CB1 and SR interrupts.

Having enabled interrupts, an operation may now occur in the 6522 which results in an interrupt flag being set, and this flag may or may not be an enabled interrupt. If it is enabled, the IRQ bit will be set, and the processor interrupted. Before the interrupt routine can identify the flag which has caused the interrupt, the microprocessor must first mask off any flag bits corresponding to disabled interrupts. This can be done simply as follows:

```
LDA  IFR     ; read flag register.
AND  IER     ; and accumulator with enable bits.
note: IER bit 7 will always be read as a zero.
```

The interrupt service routine can now correctly identify the interrupt condition and take the appropriate action, to clear the interrupt flag(s), before returning from the service routine. If it is required to clear all the interrupt flags in "one go", the following simple operation will suffice:

```
LDA  IFR     ; active flags are logic one.
STA  IFR     ; writing back a "one" clears the flag.
```

CHAPTER 6

Serial I/O Option

The serial I/O option allows the user to implement a full asynchronous serial data port, connecting, for example, to a professional computer terminal. The data port can operate as TTL, 20mA current loop, or RS232C mode, as selected by the user, with full modem control if required. The kit consists of a 6551 programmable UART (Universal Asynchronous Receiver/Transmitter), a 1.8432MHz crystal, and an RS232C interface driver. Note that the RS232C driver requires an additional power supply of +12 volts and -12 volts at approximately 15 milliamps. The data port is brought out via a 14 pin socket at location E1, the pin out of which is shown in Fig. 6.1.

In order to operate the UART, its mode of operation must be programmed from software, using its four internal registers. The significance of these registers, and their memory addresses, are shown in Figs. 6.2 to 6.5.

PIN	SIGNAL	DIRECTION	COMMENT
1	20mA Loop + Transmit	Output	
2	20mA Loop - Transmit	Output	
3	RS232C out	Output	
4	TTL Input	Input	Remove LK2 to operate with a TTL input.
5	$\overline{DTR}$	Output	Modem control
6	TTL Output	Output	
7	Ground	Input	
8	20mA Receive - $\overline{CTS}$	Input	Modem control*
9	$\overline{RTS}$	Output	Modem control
10	$\overline{DCD}$	Input	Modem control
11	$\overline{DSR}$	Input	Modem control
12	RS232C in	Input	
13	20mA Receive +	Input	Pair with pin 7
14	+5 volts	Output	

*Note: CTS must be held low to enable the transmitter if no modem is used.

Fig 6·1 Serial I/O Socket Pinout

Writing a byte of data to the UART data register loads the UART transmitter register with that data and starts transmission, providing that the necessary control bits have been set in the other registers. If the UART receives serial data, it may be read as an eight bit byte from the same memory address. The UART is organised internally with entirely separate data registers for transmitting and receiving, and it is simply for convenience in operation that they are accessed using the same memory address.

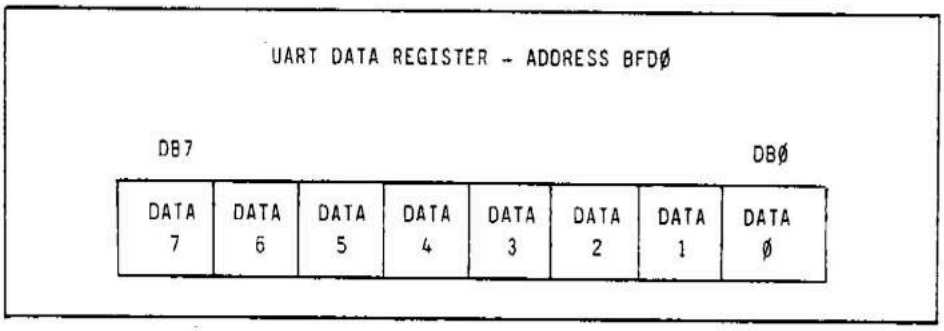
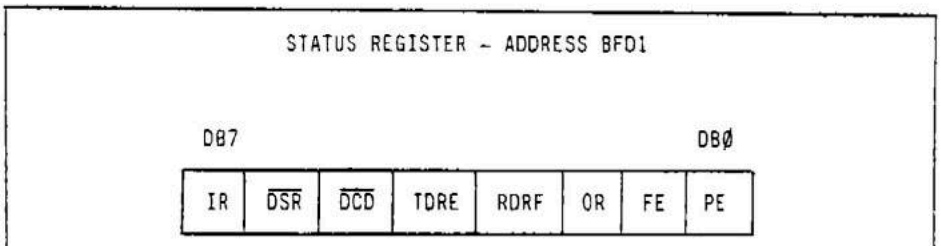


Fig 6·2 UART Data Register



This register is READ ONLY, and reflects the status of the UART.
NOTE: WRITING TO THIS ADDRESS CAUSES THE UART TO BE RESET.

PE = PARITY ERROR: This bit will be set (high) if a received byte of data has incorrect parity.
FE = FRAMING ERROR: This bit is set (high) if the received data did not frame correctly. (This is indicative of slightly mis-matched baud rates).
OR = OVERRUN: Set (active high) if the received data overruns.
RDRF = RECEIVER DATA REGISTER FULL: Active high. Indicates that data has been received. Generates an interrupt if set.
TDRE = TRANSMITTER DATA REGISTER EMPTY: Active high. Indicates that the next byte of data to be transmitted may be loaded into BDFØ. Generates an interrupt if set.
$\overline{DCD}$ = DATA CARRIER DETECTED: Active low. Indicates the presence of a carrier signal when the 6551 is used with a modem. Generates an interrupt when set.
$\overline{DSR}$ = DATA SET READY: Active low. For a modem control, indicating that data transfer can take place. Generates an interrupt when active.
IR = INTERRUPT REQUEST: Active high. This bit is set if the UART is generating an interrupt.

Fig 6·3 Status Register

COMMAND REGISTER - ADDRESS BFD2

D87							DBØ
PMC1	PMCØ	PME	REM	TIC1	TICØ	IRD	$\overline{DTR}$

Writing to this register loads the UART command register; a read may be performed to check the register contents.

$\overline{DTR}$ = DATA TERMINAL READY: Active low. For modem control - used to indicate the terminal is ready. Must be Ø to enable transmitter.
IRD = INTERRUPTS DISABLE: Active high. Disables the UART interrupt when active.
TIC1 and TICØ = TRANSMITTER INTERRUPT CONTROL.
(ØØ) RTS output = high, transmit interrupt disabled.
(Ø1) RTS output = low, transmit interrupt enabled.
(1Ø) RTS output = low, transmit interrupt disabled.
(11) RTS output = low, transmit interrupt disabled and transmit BREAK (i.e. continuous zero).
REM = RECEIVER ECHO MODE: Active high. When active, the transmitter automatically transmits (echoes) a received character - also called half-duplex mode. Normal operation is with this bit low.
PME = PARITY MODE ENABLE: Active high. When active, the UART generates a parity bit on transmission, and checks parity on reception.
PMC1 and PMCØ = PARITY MODE CONTROL.
(ØØ) Odd parity transmitted and received.
(Ø1) Even parity transmitted and received.
(1Ø) High parity bit transmitted but no parity check.
(11) Low parity bit transmitted but no parity check.

Fig 6·4 Command Register

Applying the serial I/O option

Although the above register details may make this option seem complex to use, it should be remembered that the Control and Command Registers will, for any given application, only require to be set up once. After that it will only be necessary to use the Status Register and Data Register for the actual communication, an example of which now follows. Suppose it is desired to use this port to connect to a standard computer terminal, which has the following characteristics: Baud rate 12ØØ, word length 8 bits, 2 stop bits and no parity, full duplex mode. Since this is not a modem the signals $\overline{CTS}$, $\overline{DCD}$, $\overline{DSR}$, $\overline{DTR}$ and $\overline{RTS}$ may all be ignored, though $\overline{CTS}$ must be connected to ground to enable the transmitter. For the sake of simplicity, we will also ignore the three error conditions FE, OR and PE, and we will operate the UART without interrupts. Thus the first thing the users soft-

CONTROL REGISTER - ADDRESS BFD3

DB7							DBØ
SBN	WL1	WLØ	RCS	SBR3	SBR2	SBR1	SBRØ

SBRØ to SBR3 - SELECT BAUD RATE.

(ØØØØ)	External clock - not available to the user.	
(ØØØ1)	5Ø	Baud
(ØØ1Ø)	75	Baud
(ØØ11)	11Ø	Baud
(Ø1ØØ)	135	Baud
(Ø1Ø1)	15Ø	Baud
(Ø11Ø)	3ØØ	Baud
(Ø111)	6ØØ	Baud
(1ØØØ)	12ØØ	Baud
(1ØØ1)	18ØØ	Baud
(1Ø1Ø)	24ØØ	Baud
(1Ø11)	36ØØ	Baud
(11ØØ)	48ØØ	Baud
(11Ø1)	72ØØ	Baud
(111Ø)	96ØØ	Baud
(1111)	192ØØ	Baud

RCS = RECEIVER CLOCK SOURCE: A logic one selects the internal clock source, and terminal RXC on the 6551 becomes a clock output. Logic zero selects external clock. Must always be a one.

WL1 and WLØ = SELECT WORD LENGTH

(ØØ) 8 bits (Ø1) 7 bits
(1Ø) 6 bits (11) 5 bits

SBN = STOP BIT NUMBER: When a logic zero, one stop bit is transmitted and received. When a logic one, one and a half stop bits are transmitted and received for word length of 5 bits with no parity, one stop bit for word length of 8 bits with parity and two stop bits for all other combinations.

Fig 6·5 Control Register

ware must do is to initialise the Command and Control Registers to produce an interface to match the specification given.

In the Command Register PMC1 and PMCØ should be 1Ø, selecting high parity with no check, PME should be Ø to inhibit parity check, REM a Ø to select full duplex mode, TIC1 and TICØ both Ø to disable transmitter interrupt, IRD set to 1 to disable interrupts and $\overline{\text{DTR}}$ set to Ø to enable the transmitter. The hex code to be sent to the Command Register is therefore 83. In the Control Register SBN is set to 1 for 2 stop bits, WL1 and WLØ to ØØ to select 8 bit word length, RCS set to 1 to select internal clock and SBR3 to SBRØ set to 1ØØØ to select 12ØØ baud. The hex code to be sent to the Control Register is therefore 98.

Thus, the initialisation routine becomes simply:

```
LDA  #83
STA  BFD2      ; sends command byte.
LDA  #98
STA  BFD3      ; sends control byte.
```

In addition, the user needs two subroutines, one to receive a data byte from the terminal, and one to transmit a data byte. The following examples use three RAM locations as temporary storage areas:

TRCHAR: contains a character for transmission.
RDCHAR: contains a received character.
CHWAIT: is a marker flag.

Since the example assumes full duplex mode, it is possible for a character to be received at the same time as one is being transmitted. The transmit character routine therefore always looks to see if a character has been received, and if so, sets CHWAIT to be non-zero, indicating that a received character is awaiting attention. The users main program can then take the necessary action if it inspects CHWAIT.

TRANSMIT CHARACTER ROUTINE

```
TCH:  LDA   BFD1      ; read UART status.
      PHA             ; save it.
      AND   #Ø8       ; test RDRF.
      ORA   CHWAIT    ; OR in marker.
      STA   CHWAIT    ; and store it.
      PLA             ; get status again.
      AND   #1Ø       ; test TDRE.
      BEQ   TCH       ; loop until not busy.
      LDA   TRCHAR    ; fetch character.
      STA   BFDØ      ; send it.
      RTS             ; and return.
```

RECEIVE CHARACTER ROUTINE

```
RCH:     LDA   CHWAIT   ; marker.
         BNE   RCH2Ø    ; branch if char waiting.
RCH1Ø:   LDA   BFD1     ; status.
         AND   #Ø8      ; test RDRF.
         BEQ   RCH1Ø    ; loop if none.
RCH2Ø:   LDA   #Ø
         STA   CHWAIT   ; clear marker.
         LDA   BFDØ     ; read data.
         STA   RDCHAR   ; store it.
         RTS            ; and return.
```

If it is preferred to use the port under interrupt control, the user must provide the software to modify TANBUG's interrupt linking as described in the microtan 65 manual page 6-19, and the interrupt routine must then read the UART status word to find the condition that has caused the interrupt, and take appropriate action.

CHAPTER 7

Cassette Recorder Interface

This interface allows the user to store programs on an ordinary domestic cassette recorder, and to reload them from cassette for subsequent use. The interface makes use of the 6522 Versatile Interface Adaptor described in section 5, using Timer T1 to generate serial data to the PB7 output for recording, and using the CB2 input as the signal input for reading in recorded programs. Note that these cassette connections are common to those of the parallel I/O socket B1.

The software to operate this interface is listed at the end of this section, and must, initially, be manually entered into RAM using TANBUG. Two programs are provided, called LOADCØ, to load programs from cassette, and DUMPCØ to dump programs to cassette, and these normally reside in the bottom 1K of RAM (on microtan). With these programs installed in microtan, only memory locations 44 to 4F are available to the user, plus approximately 2Ø (decimal) locations available for the stack; however, dumping the dump program itself to cassette allows the use of memory addresses 146 to 1E8 as temporary store, since the load program can then reload 'dump' when required.

Connecting to the cassette recorder

The user must provide a suitable lead to connect to his/her tape recorder. The terminal block on TANEX has four terminals. which should be connected as follows:

CAS O/P	:	Connect to cassette recorder input.
OV	:	Earth connection for shield.
CAS I/P	:	Connect to cassette recorder output.
OV	:	Earth connection for shield.

The user may now manually load the cassette software into microtan using TANBUG, in the locations indicated on the listing provided, and can immediately dump the 'DUMPCØ' program to tape for future use.

Dumping a program to tape

Subroutine DUMPCØ dumps the contents of a specified memory segment onto the tape as follows:

a) Using TANBUG's 'M' command, enter the memory address of where the dump is to start into locations 4Ø (low byte of address) and 41 (high byte).

b) Similarly, enter the end address into locations 42 (low byte) and 43 (high byte). For example, if the users program that is to be dumped starts at 4ØØ and ends at 4DE then after steps a) and b) above, the four locations should be as follows:

Location	Contents	
4Ø	ØØ	start address low byte.
41	Ø4	start address high byte.
42	DE	end address low byte.
43	Ø4	end address high byte.

c) Start the program by using TANBUG's 'G' command from location DUMPCØ (146). The VDU will give a carriage return and cursor prompt.

d) Start the cassette tape running in record mode and adjust the recording level if required. Make a note of the tape counter on the recorder for later use.

e) Let the tape run for at least 5 seconds to record the lead in tone, then hit any keyboard key to commence recording. (The cursor will also disappear from the VDU screen).

f) The contents of the specified memory area will now be recorded onto tape.

g) When the recording is complete, the cursor will reappear on the screen.

h) The cassette recorder can now be stopped. Note the tape counter indication.

Loading a program from tape

The LOADCØ program loads a program into the same memory area from which it was dumped.

a) Load the appropriate cassette into the cassette recorder, and position the tape at the start of the 5 second lead in tone recorded in step d) above, leaving the cassette on 'play' with 'pause' set.

b) Set the load program ready to run by using TANBUG's 'G' command from location LOADCØ (5Ø) but do not depress CR yet!

c) Let the tape run by releasing the pause button, and then press CR on the microtan keyboard.

d) The program will now load from tape, and the cursor will disappear from the VDU screen.

e) When loading is complete the cursor will reappear on the VDU screen, and the cassette can be stopped.

The load program may generate three error messages, these are parity error, framing error and checksum error. If a parity error occurs the program displays a 'P' and the location at which the error occurred. Program loading still continues however. The user may then correct the location that suffered the parity error. If a framing error occurs the load program completely loses synchronisation with the data on the tape and loading is aborted. 'F' is displayed along with the location where the framing error occurred. An attempt should be made to reload the tape. If the framing error occurs in the same position then there is most likely to be a drop-out on the tape, but if loading is successful then the original framing error may have been due to mains borne interference. The third error condition is an incorrect checksum, indicated by 'C' being displayed. An incorrect checksum may be caused by a double error in a data byte which will not produce a parity error. A checksum error means that the user must check his/her original listing to find and correct the "bad" data bytes.

```
                ; Program LOADCØ loads an absolute file from the cassette.
                ; The code is automatically dumped into the area of store specified
                ; on the tape at dump time.
                ; RESET recovers monitor in case of lock-up.
                ; If the user wishes to relocate code in a different area
                ; of memory (e.g. for text) he/she should change the
                ; STA 4Ø, STA 41 and STA 4Ø,Y instructions at the start
                ; of LOADCØ to NOP.
                ; Locations 4Ø and 41 should be set to the required start
                ; address and locations 42 and 43 to the required end address.

                ; Define I/O label addresses.

                IOIER  = BFCE
                IOACR  = BFCB
                IODDRB = BFC2
                IOT1LL = BFC6
                IOT1LH = BFC7
                IOORB  = BFCØ
                IOT1CL = BFC4
                IOIFR  = BFCD
                T2CH   = BFC9
                T1CH   = BFC5
                IOPCR  = BFCC
                T2LL   = BFC8

                ; Define TANBUG label addresses.

                POLLKB = FDFA
                OUTPCR = FE73
                OPCHR  = FE75
                VDUC   = Ø3EØ
                HEXPNT = FFØB
                RC1    = FC4B

ØØ5Ø  2ØBAØØ  LOADCØ: JSR INITL        ; First initialise the 6522.
ØØ53  A9ØØ            LDA #Ø          ; Clear souce address and checksum.
ØØ55  854Ø            STA 4Ø
ØØ57  8541            STA 41
ØØ59  8544            STA 44
ØØ5B  8DCBBF          STA IOACR       ; Set timer 2 mode.
ØØ5E  A97F            LDA #7F
ØØ6Ø  8DCEBF          STA IOIER       ; Disable 6522 interrupts.
ØØ63  ØA              ASL A
ØØ64  8DC8BF          STA T2LL        ; Load low byte of timer latch.
ØØ67  78              SEI             ; Disable interrupts.
ØØ68  A94Ø     FORLB: LDA #4Ø         ; Set CB2 mode.
ØØ6A  8DCCBF          STA IOPCR       ; Load 6522 peripheral control register.
ØØ6D  2Ø1CØ1          JSR TMTNCØ      ; Synchronise on tape leader.
ØØ7Ø  AØØ3            LDY #3          ; Set count.
ØØ72  2ØC3ØØ  LOAD1:  JSR RDWDCØ      ; Get status byte.
ØØ75  7Ø32            BVS FERR        ; Frame check.
ØØ77  9Ø3Ø            BCC FERR        ; Fatal error on status read?
ØØ79  994ØØØ          STA 4Ø,Y        ; Else store till done.
ØØ7C  88              DEY
ØØ7D  1ØF3            BPL LOAD1
ØØ7F  C8              INY
```

```
                ; We have now set status up in locations 40-43 as in the
                ; DUMP subroutine.
                ; Proceed to get characters and store them.
                ; Y = 0 here.

0080  20C300    LOAD3:  JSR RDWDC0          ; Get the character.
0083  7024              BVS FERR            ; Check for fatal frame error.
0085  B007              BCS LOAD2           ; Branch if parity O.K.
0087  48                PHA
0088  A950              LDA #50             ; Else set up P.
008A  20AD00            JSR ERRP            ; Call error print.
008D  68                PLA
008E  9140      LOAD2:  STA (40),Y          ; Store the number.
0090  18                CLC                 ; Clear carry.
0091  6544              ADC 44              ; Add checksum.
0093  8544              STA 44              ; Restore it.
0095  203301            JSR RECCMP          ; Increment count via subroutine.
0098  D0E6              BNE LOAD3           ; If carry clear done, else next.
009A  20C300            JSR RDWDC0          ; Get checksum.
009D  C544              CMP 44              ; Compare with actual checksum.
009F  F005              BEQ LOAD4           ; If equal then O.K.
00A1  A943              LDA #43             ; Else set up C.
00A3  20AD00    LOAD5:  JSR ERRP            ; Error.
00A6  4C48FC    LOAD4:  JMP RC1             ; Return to TANBUG.
00A9  A946      FERR:   LDA #46             ; Frame error - set F.
00AB  D0F6              BNE LOAD5           ; Unconditional branch.

                ; ERRP is the load routine error printer.

00AD  2075FE    ERRP:   JSR OPCHR           ; Output error type.
00B0  A541              LDA 41              ; Output high byte of address.
00B2  2000BFF           JSR HEXPNT
00B5  A540              LDA 40              ; Output low byte of address.
00B7  2000BFF           JSR HEXPNT
00BA  2073FE    INITL:  JSR OUTPCR          ; Carriage return.
00BD  A920      OBLCR:  LDA #20             ; Obliterate cursor.
00BF  8DE003            STA 3E0
00C2  60                RTS

                ; Subroutine RDWDC0 reads a byte from the tape.
                ; On entry it assumes that the interface is set up and running,
                ; and that initially the tape is running with 2400Hz.
                ; Answer is placed in the accumulator.

00C3  98        RDWDC0: TYA                 ; Save Y.
00C4  48                PHA
00C5  201C01    STBIT:  JSR TMTNC0          ; Keep looking for a start bit.
00C8  B0FB              BCS STBIT           ; Read first cycle.
00CA  20F600            JSR RDBTC0          ; Test start bit.
00CD  5004              BVC RDWD1           ; No framing error.
00CF  2469      RDWD2:  BIT FORLB+1         ; Redundant - set V.
00D1  701F              BVS RDWD5           ; Else return with F error.
00D3  B0FA      RDWD1:  BCS RDWD2           ; If carry set then a framming error.
00D5  A900              LDA #0              ; Else get 8 bits and clear accumulator.
00D7  48                PHA                 ; Save parity.
00D8  A008              LDY #8              ; Count of 8.
00DA  20F600    RDWD3:  JSR RDBTC0          ; Read a bit.
00DD  6A                ROR A               ; Shift into accumulator.
00DE  AA                TAX                 ; Save accumulator in X.
00DF  0A                ASL A               ; Get bit back in carry.
00E0  68                PLA                 ; Pick up checksum.
```

```
ØØE1  69ØØ            ADC #Ø              ; Add in carry.
ØØE3  48               PHA                 ; Push it on stack.
ØØE4  8A               TXA                 ; Recover A.
ØØE5  88               DEY                 ; Decrement count.
ØØE6  DØF2             BNE RDWD3           ; Till 8 bits.
ØØE8  2ØF6ØØ           JSR RDBTCØ          ; Get parity.
ØØEB  AA               TAX
ØØEC  68               PLA
ØØED  7ØEØ             BVS RDWD2           ; Error if frame.
ØØEF  69ØØ             ADC #Ø              ; Add parity bit.
ØØF1  4A               LSR A
ØØF2  68       RDWD5:  PLA
ØØF3  A8               TAY
ØØF4  8A               TXA
ØØF5  6Ø               RTS

                ; Subroutine RDBTCØ reads one bit from cassette via PB2.
                ; On entry it assumes subroutine TMTNCØ to have been used
                ; to set timer 2.
                ; Note on entry, one transition of the current data bit
                ; has been read.
                ; On exit, one transition of the next data bit will
                ; have been read.
                ; If all transitions are logic high, the result is a
                ; one, and the carry is set.
                ; If all are zero, the result is a zero and the carry
                ; is clear, otherwise the overflow bit is set as a
                ; framing error flag.
                ; Using two bits makes cassette recorder phase irrelevant.

ØØF6  48       RDBTCØ: PHA                 ; Save accumulator.
ØØF7  98               TYA                 ; Save Y.
ØØF8  48               PHA
ØØF9  A9ØØ             LDA #Ø              ; Clear accumulator.
ØØFB  AØØ2             LDY #2              ; Set Y = 2.
ØØFD  2Ø1CØ1           JSR TMTNCØ          ; Read two bits.
Ø1ØØ  2A               ROL A
Ø1Ø1  2Ø1CØ1           JSR TMTNCØ
Ø1Ø4  2A               ROL A
Ø1Ø5  B8               CLV                 ; Clear frame error flag.
Ø1Ø6  FØØ8             BEQ RDBT1           ; If Ø a zero, clear carry and V.
Ø1Ø8  AØØ6             LDY #6              ; Else set Y = 6.
Ø1ØA  C9Ø3             CMP #3              ; Compare accumulator with 3.
Ø1ØC  FØØ2             BEQ RDBT1           ; If equal to 1 carry set.
Ø1ØE  2469             BIT FORLB+1         ; Else set frame error V flag.
Ø11Ø  Ø8       RDBT1:  PHP                 ; Save V and C flags.
Ø111  2Ø1CØ1   REMCH:  JSR TMTNCØ          ; Read remaining characters.
Ø114  88               DEY
Ø115  DØFA             BNE REMCH
Ø117  28               PLP                 ; Recover PSW.
Ø118  68               PLA                 ; Recover Y and A, C and V unaffected.
Ø119  A8               TAY
Ø11A  68               PLA
Ø11B  6Ø               RTS                 ; Return.

                ; Subroutine TMTNCØ obtains the time (in processor clock
                ; cycles via timer 2) between two successive low to
                ; high transitions of the input on CB2 pin.
                ; On entry it expects the following conditions to be met:
                ; (a) Timer 2 to have been loaded at the previous low
```

```
                ; to high transition.
                ; (b) CB2 to already be set up to receive.
                ; On exit, the flag is   for a logic 1 transition,   for a logic Ø.
                ; Note this subroutine investigates one waveform cycle
                ; and not a complete bit.
                ; Cassette input CB2 working in positive edge interrupt.
                ; Register X is used.

Ø11C  48        TMTNCØ:  PHA                    ; Save accumulator.
Ø11D  ADCØBF             LDA IOORB              ; Clear interrupt flag bit 3.
Ø12Ø  ADCDBF    TMT1:    LDA IOIFR              ; Examine interrupt flag.
Ø123  29Ø8               AND #8                 ; Compare bit 3.
Ø125  FØF9               BEQ TMT1               ; Wait for the transition.
Ø127  AEC9BF             LDX T2CH               ; Read timer 2 high counter.
Ø12A  A9FF               LDA #FF                ; Restart the counter.
Ø12C  8DC9BF             STA T2CH
Ø12F  EØFE               CPX #FE                ; Set up carry. 1 for 24ØØ, Ø for 12ØØ.
Ø131  68                 PLA                    ; Reset accumulator.
Ø132  6Ø                 RTS

                ; Subroutine RECCMP compares the address held in locations
                ; 4Ø and 41 with that held in locations 42 and 43.
                ; If they are equal it exits with zero flag clear, otherwise
                ; it increments the address in 4Ø and 41 and exits with
                ; zero flag set.
                ; Note the user must initially set 4Ø and 41 to a lower
                ; value than 42 and 43.

Ø133  A543      RECCMP:  LDA 43
Ø135  C541               CMP 41                 ; Compare high bytes.
Ø137  DØØ6               BNE RCMP1
Ø139  A542               LDA 42
Ø13B  C54Ø               CMP 4Ø                 ; Compare low bytes.
Ø13D  FØØ6               BEQ RCMP2
Ø13F  E64Ø      RCMP1:   INC 4Ø                 ; Increment address.
Ø141  DØØ2               BNE RCMP2
Ø143  E641               INC 41
Ø145  6Ø        RCMP2:   RTS

                ; The program DUMPCØ is used to dump an area of memory to
                ; cassette tape.
                ; The program is written to reside in pages Ø and 1, but may
                ; be relocated by suitably changing addresses.
                ; All subroutines used must be present in RAM with the
                ; exception of POLLKB, OPCHR and OUTPCR which are in TANBUG.
                ; Once the dump program and its relevant subroutines have
                ; been entered and are working, the DUMPCØ program may be
                ; used to dump itself to cassette.
                ; In future it is only necessary to re-key in the loader
                ; program to recover the dump program after a power fail.
                ; Note that the monitor BPTCOD store is used for the dump.
                ; All breakpoints must be removed before using the dump routine.
                ; The checksum is stored in location 44.

Ø146  A2ØØ      DUMPCØ:  LDX #Ø                 ; First initialise the interface.
Ø148  8644               STX 44                 ; Clear checksum.
Ø14A  A97F               LDA #7F                ; Clear 6522 interrupts.
Ø14C  8DCEBF             STA IOIER
Ø14F  A98Ø               LDA #8Ø                ; Set port B for output.
Ø151  8DC2BF             STA IODDRB
Ø154  A9CØ               LDA #CØ                ; Set ACR for PB7 square wave.
```

```
Ø156  8DCBBF            STA IOACR
Ø159  A99C              LDA #9C             ; Load T1 counter.
Ø15B  8DC4BF            STA IOT1CL
Ø15E  8EC5BF            STX T1CH
Ø161  2Ø73FE            JSR OUTPCR
Ø164  2ØFAFD            JSR POLLKB          ; Look at keyboard.
Ø167  78                SEI                 ; Disable interrupts.
Ø168  AØØ3              LDY #3              ; Set count.
Ø16A  B94ØØØ   DUMPC1:  LDA 4Ø,Y            ; Record the 4 status words.
Ø16D  2Ø9ØØ1            JSR RECORD
Ø17Ø  88                DEY
Ø171  1ØF7              BPL DUMPC1
Ø173  2Ø8DØØ            JSR OBLCR           ; Obliterate cursor.
Ø176  C8                INY
Ø177  B14Ø     DUMPC2:  LDA (4Ø),Y          ; Get the first character.
Ø179  48                PHA                 ; Save it.
Ø17A  18                CLC                 ; Add in checksum.
Ø17B  6544              ADC 44
Ø17D  8544              STA 44              ; Save it again.
Ø17F  68                PLA                 ; Recover character.
Ø18Ø  2Ø9ØØ1            JSR RECORD          ; Transmit it.
Ø183  2Ø33Ø1            JSR RECCMP          ; Increment address and compare.
Ø186  DØEF              BNE DUMPC2          ; If carry set do next.
Ø188  A544              LDA 44              ; Else record checksum.
Ø18A  2Ø9ØØ1            JSR RECORD
Ø18D  4C4BFC            JMP RC1             ; Return to TANBUG with CR.

                ; Subroutine RECORD records a complete 8 bit byte of data
                ; present in the accumulator on entry.
                ; Format of data is logic Ø start bit, 8 bits of data lsb first,
                ; even parity bit, two logic 1 stop bits.
                ; The logic 1 stop bits may extend to longer than two bit periods.
                ; Y index is used to count the number of bits.

Ø19Ø  AA        RECORD  TAX
Ø191  98                TYA                 ; Accumulator in X.
Ø192  48                PHA                 ; Save Y.
Ø193  2CCØBF    RCRD1:  BIT IOORB           ; Is PB7 set to a 1?
Ø196  1ØFB              BPL RCRD1           ; If not wait.
Ø198  2CCØBF    RCRD2:  BIT IOORB           ; Now wait for zero transition.
Ø19B  3ØFB              BMI RCRD2
Ø19D  AØØ9              LDY #9              ; Set index count for output.
Ø19F  A9ØØ              LDA #Ø              ; Set stack to compare parity.
Ø1A1  48                PHA
Ø1A2  8A                TXA                 ; Get accumulator back.
Ø1A3  18                CLC                 ; Output start bit.
Ø1A4  2ØC6Ø1    RCRD5:  JSR OPBTCØ
Ø1A7  88                DEY                 ; Now output each bit.
Ø1A8  FØØC              BEQ RCRD3           ; When zero give parity.
Ø1AA  3ØØF              BMI RCRD4           ; When negative give stop bits.
Ø1AC  AA                TAX                 ; Save accumulator in X.
Ø1AD  4A                LSR A               ; Set up carry.
Ø1AE  68                PLA                 ; Recover parity.
Ø1AF  69ØØ              ADC #Ø              ; Add in the parity.
Ø1B1  48                PHA
Ø1B2  8A                TXA                 ; Recover accumulator.
Ø1B3  4A        RCRD6:  LSR A               ; Reset carry.
Ø1B4  1ØEE              BPL RCRD5           ; Unconditional jump to output.
Ø1B6  68        RCRD3:  PLA                 ; Recover parity.
Ø1B7  49Ø1              EOR #1              ; Invert lower bit.
```

```
Ø1B9  1ØF8            BPL RCRD6          ; Unconditional jump to output.
Ø1BB  38       RCRD4: SEC
Ø1BC  2ØC6Ø1          JSR OPBTCØ
Ø1BF  38              SEC
Ø1CØ  2ØC6Ø1          JSR OPBTCØ
Ø1C3  68              PLA                ; Recover Y.
Ø1C4  A8              TAY
Ø1C5  6Ø              RTS

                ; Subroutine OPBTCØ outputs one bit to CØ.
                ; On entry it is assumed that a minimum of 1ØØuSec logic low
                ; is available to reset timer 1 latches, and that the timer
                ; is already running except on start-up.
                ; If carry is set on entry to this subroutine, a logic 1
                ; (8 cycles of 24ØØHz) is output, otherwise a logic Ø is
                ; output (4 cycles of 12ØØHz).
                ; An exit occurs immediately after the last transition but one
                ; to give the program time to set up the 6522 timer 1
                ; latches before the next transition.
                ; On exit, the interface is still oscillating.
                ; Note LDA and LDX instructions do not affect the carry.

Ø1C6  48       OPBTCØ: PHA               ; Save accumulator.
Ø1C7  A99C             LDA #9C           ; Set time codes for 24ØØHz.
Ø1C9  A2ØØ             LDX #Ø            ; A = low byte. X = high byte.
Ø1CB  BØØ3             BCS OPBT1         ; If carry set skip the next bit.
Ø1CD  ØA               ASL A             ; Else set for 12ØØHz.
Ø1CE  E8               INX
Ø1CF  18               CLC
Ø1DØ  8DC6BF   OPBT1:  STA IOT1LL        ; Set up the timer.
Ø1D3  8EC7BF           STX IOT1LH        ; Write low and high.
Ø1D6  A9Ø8             LDA #8            ; Set accumulator for 8½ low cycles.
Ø1D8  9ØØ1             BCC OPBT2         ; If carry clear skip.
Ø1DA  ØA               ASL A             ; Else double for 24ØØHz.
Ø1DB  AA       OPBT2:  TAX               ; Get in X.
Ø1DC  ADC4BF   OPBT3:  LDA IOT1CL        ; Clear timer time-out, read low timer.
Ø1DF  2CCDBF   OPBT4:  BIT IOIFR         ; Test for it to become set again.
Ø1E2  5ØF8             BVC OPBT4         ; Can use overflow flag.
Ø1E4  CA               DEX               ; When set decrement index.
Ø1E5  DØF5             BNE OPBT3         ; Till all done.
Ø1E7  68               PLA               ; Then exit with at least 2ØØuSec.
Ø1E8  6Ø       PRGEND: RTS


                END
```

CHAPTER 8

Assembly and Construction

Although this chapter is intended for those who have purchased TANEX in kit form, it is recommended that purchasers of ready assembled TANEX also read through this, particularly the last section -"Connecting to Microtan 65". Before beginning to assemble TANEX, please read right through the instructions carefully so that you understand all the operations involved.

Preparing Your Work Surface

For assembling TANEX you will require a miniature soldering-iron, thin multicored solder, pliers and wirecutters (both of the small variety).

As in the Microtan 65, many of the integrated circuits are of the MOS type and can be damaged by static electricity, and we recommend you take the following precautions. Do not wear nylon clothes; ensure that your soldering iron is properly earthed; spread a sheet of aluminium foil (cooking foil) over your working area and earth it to a radiator or water pipe with a piece of wire.

Unpacking and Identifying the Parts

Unpack the kit and identify all the parts as listed below:

P.C.B. MT002

IC Sockets

2 off	-	8 pins
11 off	-	14 pins
3 off	-	16 pins
14 off	-	18 pins
5 off	-	24 pins
1 off	-	28 pins
2 off	-	40 pins
2 off	-	20 pins

Integrated Circuits - Minimum Configuration TANEX

M1	74LS3Ø	K2	74LSØØ	
L1	74LS21	M4	74LS244	
K1	74LS138	L3	74LS32	
J1	74LS138	A3	LM358	
H1	74LSØ4	N7	2114	(MOS)
G1	74LS139	N14	2114	(MOS)
M3	74LS244	A2	6522	(MOS)
L2	74LS74			

Optional Integrated Circuits

C2	6522	additional VIA integrated circuit.	(MOS)
F1	6551	serial I/O option.	(MOS)
D2	75150	serial I/O option.	
N1	N6, N8 - N13	additional RAM (see Fig. 3.1)	(MOS)

Resistors

(See Microtan 65 Manual, page 2 - 4 for identification method).

R1 1ØK (brown,black,orange)

R2)
R3) No longer required.

R4 1K (brown,black,red)
R5 1ØK (brown, black, orange)
R6 1K (brown, black, red)
R7 12ØK (brown, red, yellow)
R8 22K (red, red, orange)
R9 2K2 (red, red, red)
R1Ø 2K2 (red, red, red)
R11 22ØR (red, red, brown)
R12 1ØK (brown, black,orange)
R13 1K (brown, black, red)
R14 22ØR (red, red, brown)

R15 1K (brown, black, red)
R16 2K2 (red, red, red)
R17 1ØK (brown, black, orange)
R18 1K (brown, black, red)
R19 1ØK (brown, black, orange)

Capacitors

C1 - C19 and C21 - C23 47n
C2Ø 1ØØ µF 1ØV

Transistors

Tr1 - Tr3 BC184

Crystal

1.8432 MHz serial I/O option.

Connectors

14 Pin DIL Header serial I/O option.
64 pin Eurocard style edge connector.
4 way screw terminal block.

ASSEMBLING TANEX

a) Fit and solder the IC sockets, ensuring that they are the correct way round, i.e. that the pin 1 identifier on the socket is at the same end as the identifier mark on the printed circuit board. A1, B1, C1, D1, E1 should be fitted with 'RN' sockets.

b) Fit and solder the resistors into their positions. Note R2 and R3 no longer fitted.

c) Fit and solder the capacitors into their positions. C2Ø is an electrolytic, connect positive terminal to

the "O" in "C20" legend.

d) Fit and solder the transistors into their positions.

e) Fit and solder links LK1,LK2 and LK3 using the excess wire cut off from one of the resistors.

f) Fit and solder the screw terminal for the cassette interface, and the edge connector.

g) If you have purchased the serial I/O option, fit the crystal using the "sticky pad" provided and solder in place. Plug Header into E1 when serial I/O not being used.

h) Insert the integrated circuits, ensuring that they are the correct way round, into their respective sockets, leaving the MOS devices until last.

Assembly is now complete, but carefully double check to ensure that there are no solder blobs or bridges anywhere.

Further Assembly Hints

1) The circuit board is of the plated through hole type. This makes it difficult to remove a component once it has been soldered in place, so please double check that you have the right component in the right place before soldering it.

2) Do not apply pressure to the printed circuit board when soldering, as this may cause the tracks to lift and break.

3) A good solder joint is made by 'wetting' the tip of the soldering iron with solder, and then placing the tip of the iron against the leg of the component where it emerges from the circuit board, and then feeding a small amount of solder against the tip,

allowing it to flow around the area of the joint.

4) Wash your hands. Dirt and grease on the circuit board will make soldering difficult and unreliable.

5) Components are inserted on the side of the board with the white printing, and soldered ONLY on the opposite side.

6) Most important of all: DO NOT HURRY.

CONNECTING TO MICROTAN 65

In order to allow Microtan 65 and TANEX to operate together, the following procedure, which assumes that the user has purchased a motherboard, must be followed:

1) Turn off the power supplies, and disconnect the Microtan 65.

2) On Microtan 65, cut through the three links marked LKRAM, LKROM and LKIO.

3) On TANEX, LK3 has been added for DMA operations. When connected LK1 must be disconnected and DMA operations may then be carried out. For normal use leave LK3 disconnected until XBUG has been fitted and LK1 cut.

4) Now plug the two boards into the mini-motherboard or system motherboard. Note that the edge connectors of the two boards are offset at different positions to prevent confusion. The track side of the motherboard, or mini-motherboard indicates the correct positions for the printed circuit boards.

5) Reconnect to you power supply, and begin to enjoy the full power of your expanded Microtan System.

TANGERINE
COMPUTER SYSTEMS LIMITED
Forehill Works Forehill Ely Cambs England