Texas Instruments

TMS 3556

3556 Video Display Processor

User's Manual

TMS3556
FAMILY
USERS
MANUAL

M GODDARD

22.03.84

IMPORTANT

NOTICES

This product was developed by Texas Instruments under contract to the Centre Commun d'Etudes de Television et Telecommunications (CCETT).

Numero de marche : M4643W.

Information contained in this publication is believed to be accurate and reliable. However, responsibility is assumed neither for its use nor for any infringement of patents or rights of others that may result from its use. No licence is granted by implication or otherwise under any patent or patent right of Texas Instruments or others.

P R E F A C E

This document is a guide to the use of the TMS3556 family of Video Display Processors (VDP's).

It is intended to be autonomous by containing all pertinent details of specification, operation, application and characteristics of the VDP family of devices. With the possible exception certain documents relating to specific detailed applications it is intended that this users manual contain all necessary information for the system engineer designing around one of the VDP family devices. As such, it replaces all preceeding documention, and may in some cases render the latter obselete, erroneous or both.

CONTENTS

LIST OF FIGURES

FIGURE TITLE PAGE

LIST OF TABLES

0 PIN ASSIGNEMENT

The pin assignement of the TMS3556 Video Display Processor is shown in Figure 0.1 and a brief functional description of each pin is given in Table 0.1.

These two items are placed immediately at the start of this users guide for ease of reference.

FIGURE 0.1 : TMS3556 VDP PIN ASSIGNEMENT

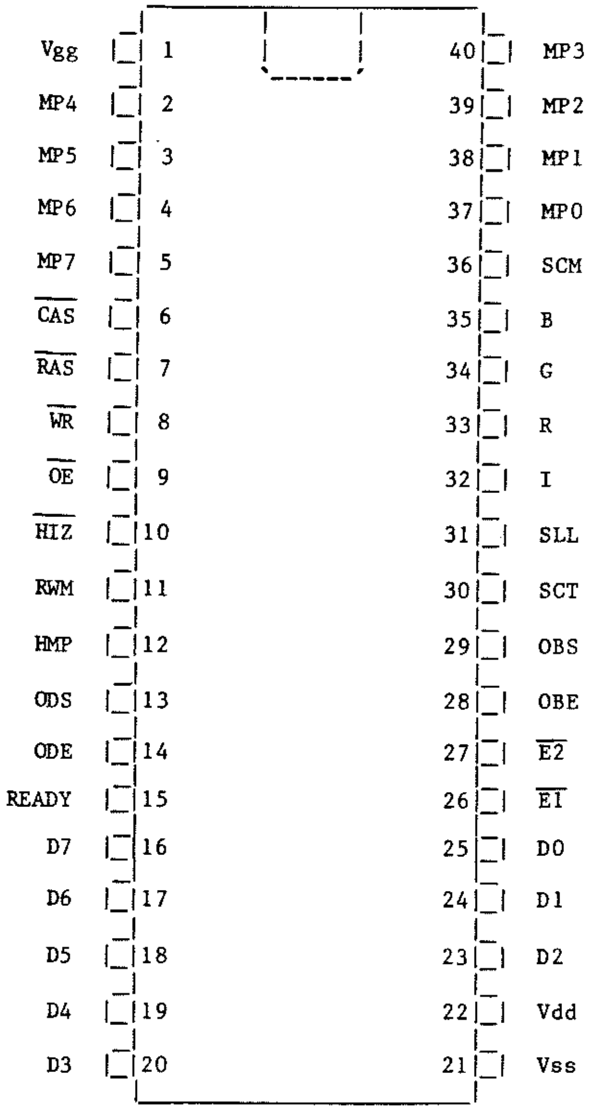


TABLE 0.1: TMS3556 VDP PIN DESCRIPTION

PIN NAME	PIN NO.	DESCRIPTION
MP0	37	
MP1	38	
MP2	39	CPU to VDP address
MP3	40	and data bus
MP4	2	
MP5	3	MP0 is the MOST
MP6	4	significant bit
MP7	5	
E1	26	CPU to VDP access
E2	27	control signals
RWM	11	CPU data write stobe
READY	15	VDP to CPU ready signal
B	35	Blue output
G	34	Green output
R	33	Red output
I	32	Incrustation output
SCM	36	Composite sync output
SLL	31	Line sync input or output
SCT	30	Frame sync input or output
D0	25	
D1	24	
D2	23	VDP to memory
D3	20	address and data bus
D4	19	
D5	18	D0 is the MOST
D6	17	significant bit
D7	16	
RAS	7	Row address strobe
CAS	6	Column address strobe
WR	8	Memory write strobe
OE	9	Memory output enable
HMP	12	Data provider access request
HIZ	10	Data provider access grant
ODE	14	DMA oscillator input
ODS	13	DMA oscillator output
OBE	28	Time base oscillator input
OBS	29	Time base oscillator output
Vgg	1	5 volt power supply
Vdd	22	3 volt power supply
Vss	21	Power supply ground

1 INTRODUCTION

1.1 Description

The TMS 3556 Video Display Processor (VDP) is designed to enable the construcion of low cost, high performance video display terminals including those used in teletext and vidoetex type applications.

The VDP generates individual red, green, blue (RGB) and synchronisation (SCM) signals suitable for display on a raster scan colour or monochrome video display monitor or domestic television set. A fast switching, or incrustation signal (I) is also generated to enable mixing of the VDP output with an external video signal such as an off-air broadcast, VCR output or another VDP.

Modes of operation, display and addressing are directly programmable by on chip control registers. Characteristics of displayed characters, screen format, video signal timing etc. are programmable by a single mask level change. Several of these mask programmed variants exist as standard products and are described in section 11. The body of this document deals exclusivly with the 'standard' VDP identified by the term 'moat programming XA 8628'.

The VDP supervises and controls a video display memory (VRAM) of up to 64K bytes. Automatic control signal generation and transparent refresh of dynamic random access memories (DRAM's) allow large memories at very low cost. High speed request and access grant signals are provided to allow fast external data input to the memory block without affecting VDP display operation or picture quality.

Manipulation of data within the VDP's memory and programmation of internal VDP control registers is accomplished via the independant processor interface. This is designed to be transparent to VDP display operation and does not affect picture stability.

1.2 Display formats

1.2.1 Bit-mapped mode

Bit-mapped mode is designed for terminals producing alpha-geometric type displays.

Each dot on the display screen, typically 320 points by 250 lines, is individually programmable in one of eight colours.

1.2.2 Text mode

Text mode is designed for terminals producing alpha-mosaic type displays.

The screen is composed of typically 25 rows of 40 characters. The characters may be alpha-numeric or block graphic. Characters from up to four independant character generators each containing 128 different characters are allowed on one display screen.

Characters may be flashed, inversed, underlined, masked, displayed in double width, double height or double size. They may also be superimposed onto an automatically synchronised external video signal.

1.2.3 Mixed mode

The above two modes of operation may be mixed on a line by line basis on the same screen. Since text mode requires less memory space than bit-mapped mode, this mode enables economy of VRAM in applications where full bit-map is not required over the whole screen.

1.2.4 Subtitling mode

Subtitling mode is a mode in which the vertical size of the active area of the display is greatly reduced, typically to three lines of text, and is displayed at the bottom of the active frame. This would normally be used in conjunction with the incrustation attribute to overlay text in a subtitling manner on to an external video image.

The display memory required for this mode of operation is greatly reduced as a consequence.

1.3 Features

- Single chip Video Display Processor
- Generation of RGB signals for TV display
- Composite sync output for standard TV's
- Frame and line sync outputs for direct drive CRT's
- External syncronisation capability
- Direct control and refresh for DRAM'S
- Transparent asynchronous direct memory access
- Asynchronous transparent CPU access to display memory
- High speed memory input channel for teletext data
- Up to 64K bytes of directly addressable video memory
- Text, bit-mapped, mixed and subtitling modes
- Compatible with 4, 8 and 16 bit processors
- 3 and 5 volt power supplies
- 40 pin plastic package

1.4 Typical applications

- Teletext/Videotex receivers and adaptors
- Home computers
- Games terminals
- Point-of-sale terminals
- 40/80 character video display terminals
- Low/medium resolution CAD/graphics terminals
- Process control terminals

2 SYSTEM ARCHITECTURE

Figure 1 shows a block diagram of a typical application of the TMS3556 VDP as it could be used in a teletext receiver.

The interface between the VDP and the external system can be divided into six major sections: CPU, memory, data provider, display, oscillators and power supply.

This section describes each of these interfaces in turn.

2.1 CPU Interface

2.1.1 Control signals

The signals comprising the CPU/VDP interface are as follows:

$\overline{\text{E1}}$ & $\overline{\text{E2}}$ - VDP interface enable signals.

These two signals combined enable the CPU/VDP interface and define the type of access cycle being performed by the CPU as shown in Table 2.1.

TABLE 2.1 - CPU/VDP ACCESS CONTROL FOR DATA TRANSFERS

<table>
<tr><th>Operation</th><th>E1</th><th>E2</th><th>RWM</th><th>MP bus state</th></tr>
<tr><td>Inactive</td><td>H</td><td>H</td><td>X</td><td>High-z</td></tr>
<tr><td>Read VDP register</td><td>L</td><td>H</td><td>_ H _</td><td>Output</td></tr>
<tr><td>Write VDP register</td><td>L</td><td>H</td><td>_|_|_</td><td>Input</td></tr>
<tr><td>Write VDP memory</td><td>H</td><td>L</td><td>|_|</td><td>Input</td></tr>
<tr><td>Read VDP memory</td><td>L</td><td>L</td><td>H</td><td>Output</td></tr>
</table>

X indicates any state, including transitions

RWM - CPU data write strobe.

This signal is used to strobe data into the VDP from the data bus for CPU write cycles.

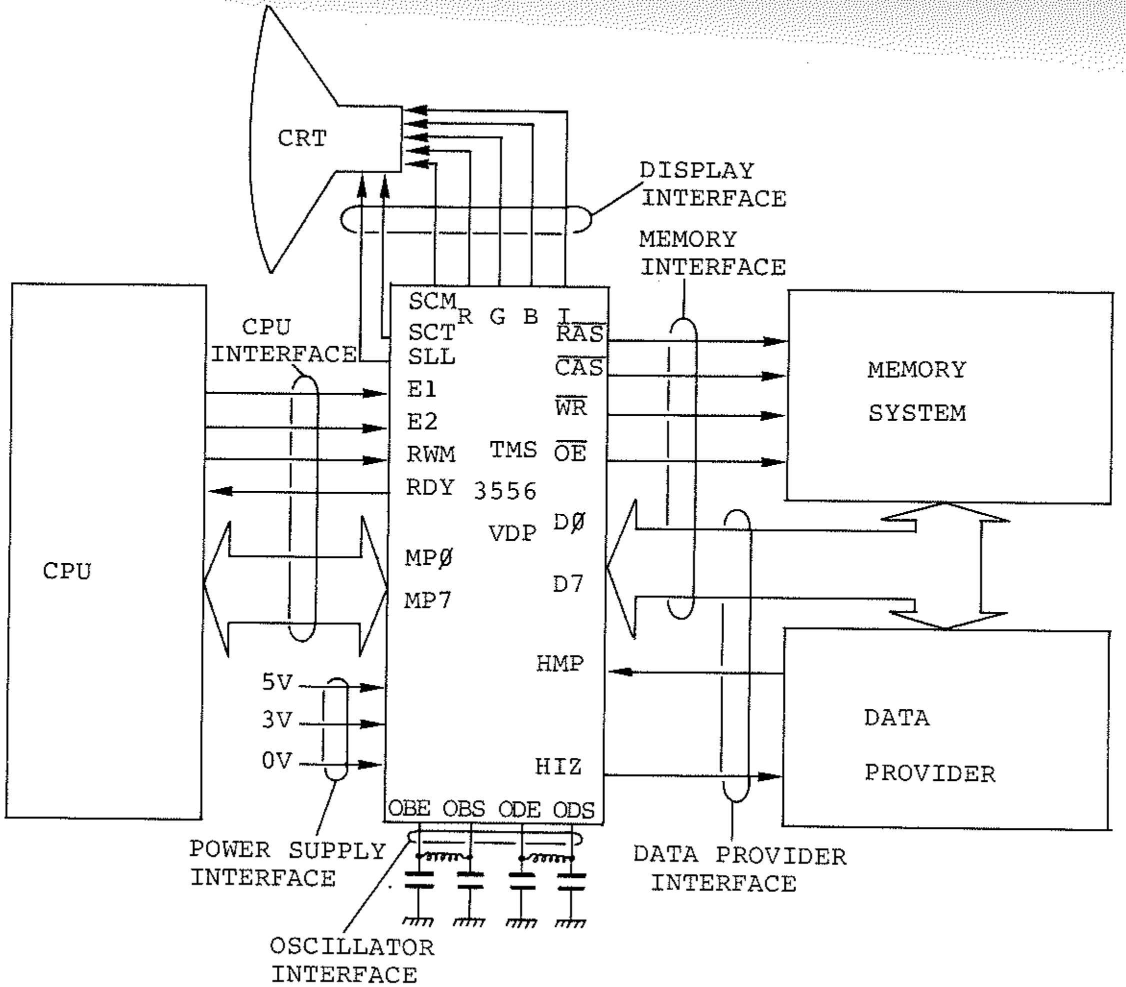


FIGURE 2.1 : SYSTEM BLOCK DIAGRAM

MP0-7 - Microprocessor/CPU to VDP 8 bit data bus.

These eight signal lines are used for bi-directional transfer of data and addresses between the CPU and the VDP. The bus carries data from the CPU destined for either internal VDP registers or VDP memory in the case of a write cycle, or data from the VDP's memory or status register in the case of a read cycle.

NOTE : Contrary to the convention used in some processor based systems, MP0 is the MOST significant bit of the CPU-VDP data bus, and MP7 the LEAST significant.

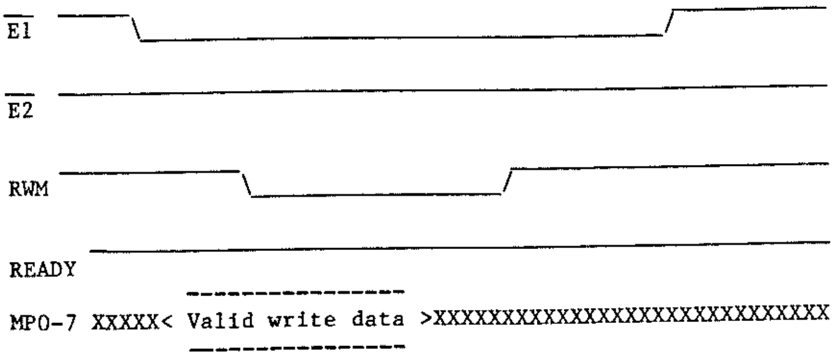
READY - VDP to CPU ready acknowledge signal.

This line is used by the VDP to signal the CPU that data on the MP bus is valid in the case of a read cycle or that the VDP is ready for a new access cycle following a write to VDP memory or base address register.

Figures 2.2, 2.3, 2.4, 2.5 & 2.6 show the action of these signals for each type of access cycle.

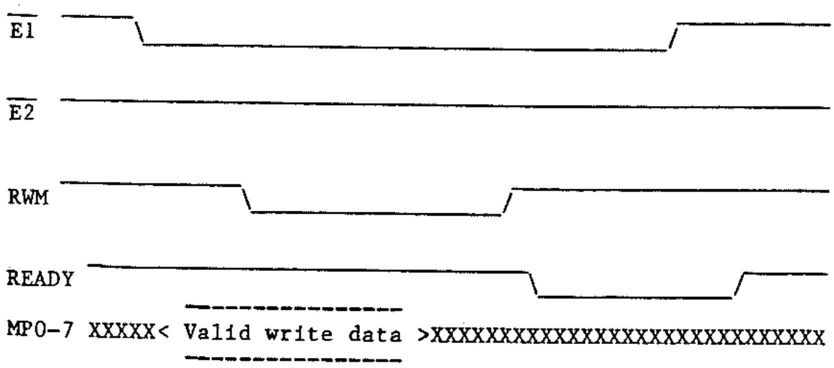
2.1.2 Write to control registers

FIGURE 2.2 - WRITE TIMING TO CONTROL REGISTERS


E1
E2
RWM
READY
MP0-7 XXXXX< Valid write data >XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

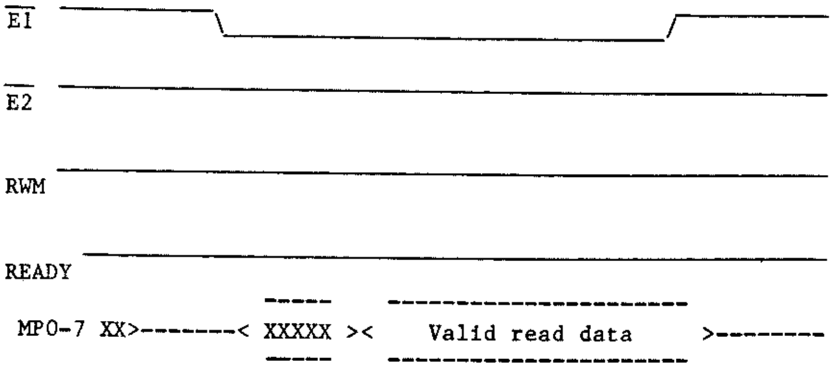
2.1.3 Write to base address registers

FIGURE 2.3 - WRITE TIMING TO BASE ADDRESS REGISTERS


E1
E2
RWM
READY
MP0-7 XXXXX< Valid write data >XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

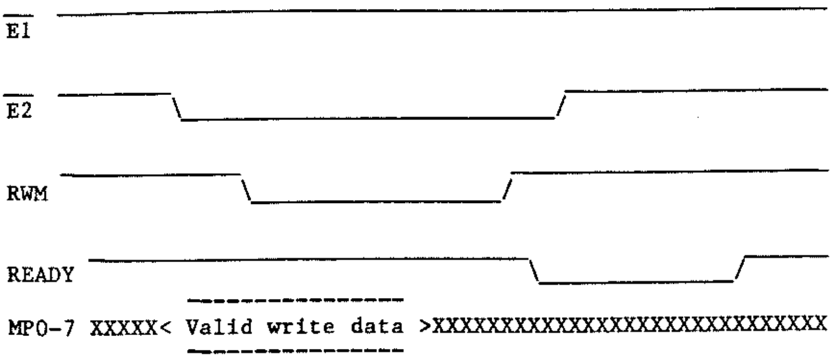
2.1.4 Read from status register

FIGURE 2.4 - READ TIMING FROM STATUS REGISTER



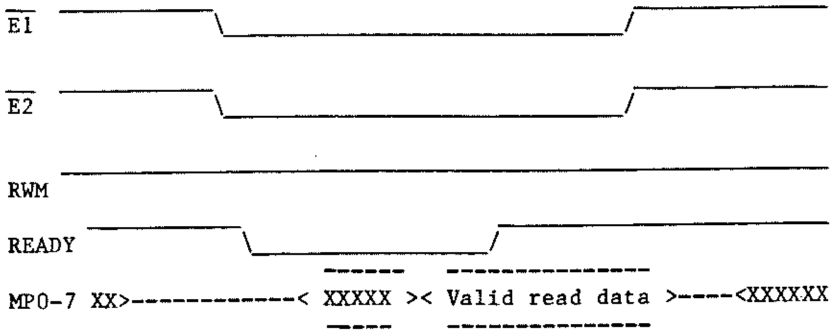
2.1.5 Write to VDP memory

FIGURE 2.5 - WRITE TIMING TO VDP MEMORY


E1
E2
RWM
READY
MP0-7 XXXXX< Valid write data >XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

2.1.6 Read from VDP memory

FIGURE 2.6 - READ TIMING FROM VDP MEMORY


E1
E2
RWM
READY
MP0-7 XX>------------< XXXXX >< Valid read data >----<XXXXXX

2.1.7 VDP initialisation

The VDP is initialised by programming control register CM1. There is no hardware reset implemented in the device. Programming CM1 (with any value) has the effect of halting all VDP operation - i.e the display is disabled and the VDP ceases to refresh DRAM. The various line, row and frame counters are also reset to zero and held in this state although both the time base and DMA oscilators continue to run.

VDP operation is restarted when any of the base address registers are programmed.

VDP initialisation would typicaly continue with the programation of the other control registers CM2 to CM4 followed by initialisation of the relevant base address registers.

VDP initialisation is discussed in more detail in section 3.

2.2 VDP/memory interface

2.2.1 Control signals

The control signals comprising the VDP interface to RAM are essentially designed for Dynamic Random Access Memory (DRAM) although with suitable adaption they can be used to drive static RAM, ROM or PROM.

The signals are $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, $\overline{\text{WR}}$, $\overline{\text{OE}}$ and D0-D7 and are described below:

$\overline{\text{RAS}}$ - Row Address Strobe.

The low going edge of this signal indicates that a valid row address is present on the D0 to D7 bus and is used to latch this address into the memory. The active low level thereafter acts as a chip enable signal for the memories.

$\overline{\text{CAS}}$ - Column Address Strobe.

The low going edge of this signal indicates that a valid column address is present on the D0 to D7 bus and is used to latch this address into the memory. The active low level thereafter acts as an output enable for some memory systems.

$\overline{\text{WR}}$ - Write strobe.

The low going edge of this signal indicates that valid write data is present on the D0 to D7 data bus and is used to latch this data into memory.

$\overline{\text{OE}}$ - Output Enable.

The active low level of this signal indicates that the VDP has placed its D0 to D7 bus in the high impedence input mode ready to accept data from the memory.

D0 - D7 - Memory address/data bus

The 8 bit multiplexed address and data bus used to transfer the row and column adresses to the memory and the data to and from the memory.

NOTE : Contrary to the convention used in some processor based systems, D0 is the MOST significant bit of the VDP - memory bus, and D7 the LEAST significant.

2.2.2 Memory types

The TMS3556 VDP interface control signals as described in paragraph 2.1 are directly compatible with most currently available dynamic memories including the 4164 (64K by 1), the 4416 (16K by 4) the 4408 (8K by 4) and the 4116 (16K by 1). With a suitable interface some static memories may also be used, as described in section 5, although except in the case of relatively small memory requirements this alternative is rarely cost effective.

The differences in internal VDP register programing to comply with the differing address structure of the above memories are described in section 3.

2.3 VDP/data provider interface

2.3.1 General

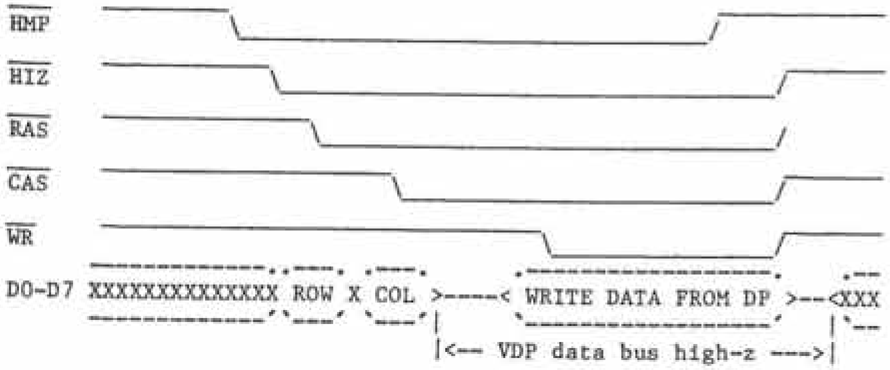
The term 'data provider' is used in this users guide in a general sense to indicate any device or system used to input data directly to the VDP's memory system. This action takes place under the full supervision of the VDP and control of the CPU. The interface can be considered similar to a DMA channel in a normal processor-memory system, although it may only write data one byte at a time and into a zone defined by the VDP.

The data provider itself may be a teletext packet identifier, videotex UART, or another processor.

2.3.2 Interface control signals

The data provider uses the same data bus as the VDP memory system. Two handshake lines HMP and HIZ control the VDP/data provider interface. HMP is the data provider access request line and signals the VDP that the former has a valid byte of data ready for transfer into the VDP's memory. The VDP, upon granting the access request signals this fact with a low level on the HIZ line. The timing of the data provider write cycle is shown in figure 2.7

Figure 2.7: VDP/Data Provider Interface Timing



2.4 VDP/TV interface

The VDP/TV interface carries the red, green and blue (RGB) colour picture information from the VDP to the television or CRT monitor.

The interface also carries three synchronisation signals, two of which are bidirectional and are used to synchronise the display CRT to the VDP or vice versa. These signals are described below:

SCM - Composite sync output.

SCM is an output only signal and carries CCIR compatible composite line and frame synchronisation information when the VDP is programmed in internal sync mode. This signal can either be used directly to drive the sync input of a monitor or TV or combined with the RGB information via a suitable colour encoder to generate a composite video signal.

The state of SCM is undefined when the VDP is programmed in external sync mode.

SCM is an open drain output.

SLL - Line sync input/output.

SLL is a bidirectional signal carrying the line synchronisation information to or from the TV depending upon the programmed mode of the VDP.

SLL is an open drain output or high impedance input.

SCT - Frame sync input/output or composite sync input.

SCT is a bidirectional signal which either carries frame synchronisation information to or from the TV or composite sync from the TV depending upon the programmed mode of the VDP.

SCT is an open drain output or high impedance input.

The final signal making up the VDP/TV interface is the fast switching or Incrustation signal I.

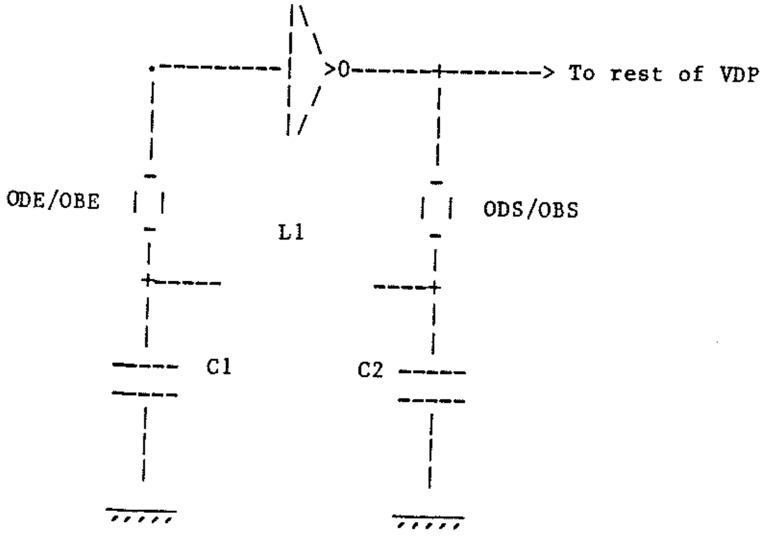
The state of this signal is controled by information contained in the page display memory and depends upon the programmed mode of the VDP. It can be used as a video overlay control signal to superimpose or incrust RGB information from the VDP into an external video signal as in the case of teletext, or it may be used to switch between two possible colour palette memories for greater colour control.

Use and programmation of this signal is described in section 4.

2.5 Oscilators

The VDP contains two on-chip oscillator circuits: DMA and the Time Base. Both oscillators are configured as Colpitts LC oscillators as shown in figure 2.8.

Figure 2.8: Simplified Oscillator structure



Used in this way with an inductor and two'capacitors the nominal frequency can be calculated from the formula below:

$$f = \frac{1}{2\pi\sqrt{\frac{LC1C2}{C1+C2}}}$$

As an alternative to using inductors and capacitors a ceramic resonator may be used or the input sides (OBE & ODE) of the oscillators may be driven from an external clock. For more details refer to section 6.

2.6 Power supply

The VDP requires two independant power supplies (although one can be generated from the other externally).

The large majority of functional elements in the VDP are powered by the 3 volt Vdd supply. The latter is less than the conventional 5 volts in order to reduce the devices power dissipation and hence operating temperature.

Two sections of the VDP are powered by the 5 volt Vgg supply; these are the input/output buffers and the DMA and memory controller. The former is to maintain TTL signal level compatibility, and the latter is because these sections operate at much higher frequencies than the rest of the device.

3 VDP INTERNAL ARCHITECTURE

3.1 General

Figure 3.1 shows a block diagram of the internal architecture of the VDP. The latter is composed of four major sections: CPU control and interface, display controler and attribute decoder, DMA controler and memory timing generator, and the timebase and synchronisation logic.

3.2 CPU interface section

All comunication between the CPU and the VDP passes via the CPU/VDP interface as described in paragraph 2.1.

The CPU needs to access the VDP for one of the following reasons:

- Program VDP operational parameters (command registers)
- Program VDP memory organisation (base addresses)
- Read from the data buffer or page memory
- Write to the page memory or character generators
- Read VDP status

3.3 On-chip registers

The VDP contains 16 registers accessable by the CPU. One of these is the read only status register, the others being write only. Table 3.1 shows the registers together with their access addresses.

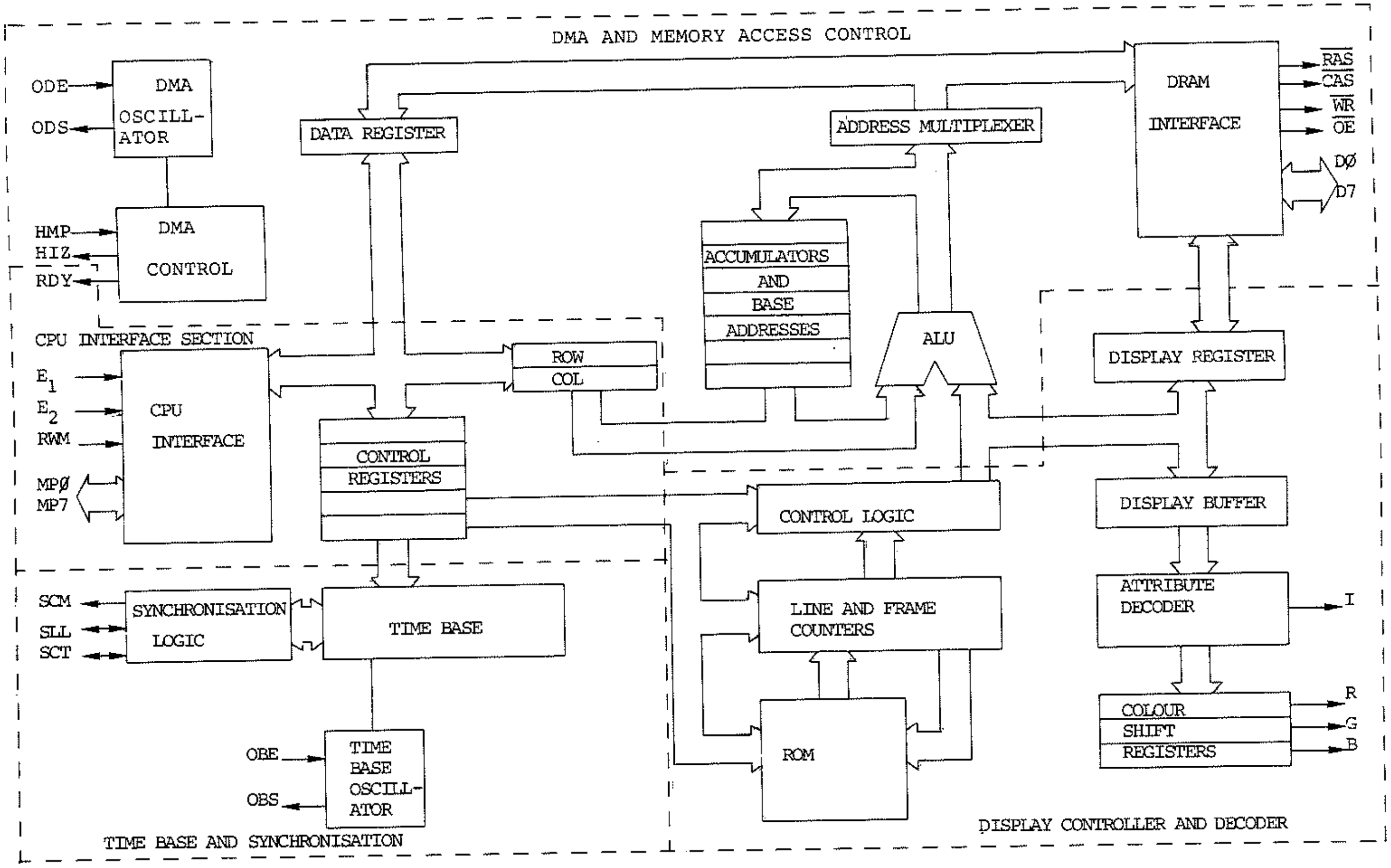


FIGURE 3.1 : BLOCK DIAGRAM OF TMS 3556 VDP

TABLE 3.1: ON-CHIP REGISTERS

ADDRESS	REGISTER	CONTENTS
0	REGISTER POINTER	REGISTER ADDRESS
1	COL	LS BYTE ADR BUFFER
2	ROW	MS BYTE ADR BUFFER
3	STATUS	VDP STATUS INFO
4	CM1	TIME BASE CONTROL
5	CM2	DECODER CONTROL
6	CM3	MODE & MEMORY CTL
7	CM4	FULL PAGE ATTRIBS
8	BAMT	START OF BUFFER
9	BAMP	CPU ADDRESS REG
10	BAPA	START OF DISPLAY
11	BAGC0	CHARACTER GEN NO.0
12	BAGC1	CHARACTER GEN NO.1
13	BAGC2	CHARACTER GEN NO.2
14	BAGC3	CHARACTER GEN NO.3
15	BAMTF	END OF BUFFER

Essentially the CPU may make two types of access to the VDP: access to internal registers or access to the external RAM. The type of access is defined by the code on the E1 and E2 enable lines as shown in Table 3.2.

TABLE 3.2 : CPU - VDP ACCESS TYPES

E1	E2	OPERATION
1	1	Interface inactive (MP bus high-z)
0	1	Access to on-chip registers
1	0	Write to external VDP memory
0	0	Read from external VDP memory

In addition, any operation involving a transfer of data from the CPU to the VDP, i.e. a write to an internal register or VDP memory is accompanied by a low-going pulse on the CPU write data stobe line RWM.

Timimg diagrams for each type of access appear in section 2.

The 16 on-chip registers are all accessed via the first register in the block - the register pointer. Any access to the on chip registers, i.e. an access with E1=0 and E2=1, will access the register indicated by the contents of this pointer. The latter is shown in Figure 3.2

FIGURE 3.2: REGISTER POINTER CONTENTS

RT8	RT4	RT2	RT1	AC8	AC4	AC2	AC1

The least significant nibble, AC8-AC1, contains the address of the register to be accessed. Note that if the value of these four bits is zero, then the pointer will in fact be pointing to itself. This condition is necessary if the pointer contents are to be changed.

The most significant nibble, RT8-RT1, contains what is termed the 'return' address. These four bits are copied into the least significant nibble following the access and are normaly programed to zero so that the pointer points to itself after the access allowing its contents to be modified.

Any memory access performed by the CPU, i.e. either of the accesses with E2=0, resets all eight bits of the register pointer to zero.

3.3.1 Programming registers 1 to 7

Registers 1 (COL) to 7 (CM4) are all eight bit registers and may be programmed (excluding register 3 - the STATUS register) by setting the pointer to address the desired value and then writing the required data to the VDP.

As an example, consider the action of writing to registers COL (address=1) followed by ROW (address=2). Assume that the contents of the pointer are initially zero. The sequence of events is as follows:

	E1	E2	RWM	MP bus	Pointer
1. Initial state	1	1	1	high-z	>00
2. Perform a register access with the value of >01 on the data bus (Pointer now points to COL)	0	1	‾|_|‾	>01	>01
3. Return to inact. state	1	1	1	high-z	>01
4. Perform a register access with the value of COL on the data bus	0	1	‾|_|‾	<col>	>01
5. Return to inact. state (Pointer returns to zero)	1	1	1	high-z	>00
6. Register access with >02 on data bus (Pointer now points to ROW)	0	1	‾|_|‾	>02	>02
7. Inactive state	1	1	1	high-z	>02
8. Register access with the value of ROW on the data bus	0	1	‾|_|‾	<row>	>02
9. Inactive state (Pointer returns to zero)	1	1	1	high-z	>00

The register COL and ROW now contain the desired values and the pointer has returned to zero ready for a new access.

The sequence shown above uses the return address RT8-RT1 to clear the pointer after each access. Although this is the simplest technique of register programming, it is also the slowest. The fact that the contents of RT8-RT1 are copied into AC8-AC1 following each access may be used to reduce the number of operations required. Consider the following sequence:

	E1	E2	RWM	MP bus	Pointer
1. Initial state	1	1	1	high-z	>00
2. Perform a register access with the value of >21 on the data bus (Pointer now points to COL)	0	1	‾‾_‾‾	>21	>21
3. Return to inact. state	1	1	1	high-z	>21
4. Perform a register access with the value of COL on the data bus	0	1	‾‾_‾‾	<col>	>21
5. Return to inact. state (RT8-RT1 have been copied into AC8-AC1 and the pointer now points to ROW)	1	1	1	high-z	>22
6. Register access with the value of ROW on the data bus	0	1	‾‾_‾‾	<row>	>22
7. Inactive state	1	1	1	high-z	>22

The same result has been obtained i.e. COL and ROW now contain the desired data, but with fewer accesses to the VDP. The difference is that the pointer is now 'locked' with a value of >22 and no longer points to itself ready for further accesses. The only alternative is to perform an access to the VDP's external memory, i.e. either of the accesses with E2=0, which automatically resets the pointer to zero:

	E1	E2	RWM	MP bus	Pointer
7. Initial state (last state of preceding sequence)	1	1	1	high-z	>22
8. Perform an external memory access	0	0	1	<data>	>00
9. Return to inact. state	1	1	1	high-z	>00

A read access is shown for the memory access in step 8 but a write access would have had the same effect on the pointer. The pointer value is now zero enabling further register accesses.

3.3.2 Reading the Status register

The Status register is accessed in a similar manner to the other eight bit registers with the exception that its contents may only be read. This implies that the CPU's data strobe line RWM remains inactive high during the access and that the CPU must place its data bus (MP0-MP7) in the high impedance input mode in order to read the data output by the VDP.

The sequence of operations shown below is typical of a Status register read:

	E1	E2	RWM	MP bus	Pointer
1. Initial state	1	1	1	high-z	>00
2. Perform a register access with the value of >03 on the data bus (Pointer now points to STATUS)	0	1	‾|_|‾	>03	>03
3. Return to inact. state	1	1	1	high-z	>03
4. Perform a register access with the data bus in the input mode	0	1	1	<status>	>03
5. Return to inactive state	1	1	1	high-z	>00

The 'fast' register access mode (i.e. when RT8-RT1 are not zero) may be used to read the Status register continuously. If a value of >33 is written to the pointer then as long as E1 is held low, the contents of the Status register will be continuously updated on the data bus. In addition, if the E1 and E2 lines are returned to their inactive high states, it is sufficient to simply take E1 low again and the current contents of the status register will be immediately placed on the MP bus by the VDP:

	E1	E2	RWM	MP bus	Pointer
1. Initial state	1	1	X	high-z	>00
2. Perform a register access with the value of >33 on the data bus (Pointer now points to STATUS)	0	1	‾|_|‾	>33	>33
3. Return to inact. state	1	1	X	high-z	>33
4. Perform a register access with the CPU data bus in the input mode	0	1	1	<status>	>33
5. Hold E1 low	0	1	1	<new> <status>	>33
6. Hold E1 low	0	1	1	<new> <status>	>33
7. Return to inact. state	1	1	X	high-z	>33
8. Take E1 low : :	0	1	1	<new> <status>	>33
9. Return to inactive state	1	1	X	high-z	>33

At the end of the operation the pointer is 'locked' with a value of >33 and a memory access must be performed to return its value to zero as before.

3.3.3 Programming registers 8 to 15 (Base adresses)

Registers 8 to 15 are all 16 bit write only registers containing the base addresses which define the external memory structure as described in paragraph 3.4.

Since the CPU to VDP data bus is only 8 bits wide, these registers may only be programmed indirectly.

The desired address is first placed in COL and ROW and then transfered to the relevant base address. ROW contains the most significant byte of the address and COL the least significant.

COL and ROW are programmed as described in paragraph 3.3.1 and the data is then transfered to the base address register by writing the register address to the pointer twice.

The sequence for programming the base address BAPA is shown below as an example. Assume that COL and ROW already contain the least and most significant bytes of the address to be programmed respectivly, and that the pointer contains a value of zero.

	E1	E2	RWM	MP bus	Pointer
1. Initial state	1	1	X	high-z	>00
2. Write >0A to the pointer. Pointer now points to BAPA	0	1	‾‾|_|‾‾	>0A	>0A
3. Return to inact. state	1	1	X	high-z	>0A
4. Perform a dummy write to the pointer. ROW is copied into the MS byte of BAPA and COL into the LS byte of BAPA	0	1	‾‾|_|‾‾	don't care	>0A
5. Return to inact. state (Pointer returns to 0)	1	1	1	high-z	>00

Following this sequence, BAPA contains a copy of COL and ROW and the pointer contains a value of zero.

When addresses are transferred from COL and ROW to a base address register an unavoidable incrementation of their value occurs. This incrementation is by two for character generator base addresses BAGC0 to BAGC3 and by one for the other base address registers. Thus to programme BAPA with a value of >8000 say, COL and ROW must be programmed to >7FFF. To programme BAGC0 with >1500 COL and ROW must be programmed with >14FE.

3.3.4 Status register

Register 3 is an eight bit register containing a variety of status information for use by the CPU. All status bits are active high. The status register contents are depicted in Figure 3.3.

FIGURE 3.3 : STATUS REGISTER

ST1	ST2	ST3	ST4	ST5	ST6	ST7	ST8

ST1 - Not used - always zero

ST2 - Frame sync

This bit is active high during the VDP frame synchronisation time. The source of the information depends on the mode of operation of the VDP as shown in table 3.3

TABLE 3.3: SOURCE OF FRAME SYNC SIGNAL FOR ST2

DC1	BT4	SIGNALS ON SLL & SCT	ST2 SOURCE
0	0	Input line & frame	See note 1
0	1	Input line & composite	See note 2
1	0	Input line & composite	Generated from SLL & SCT
1	1	Output line & frame	Copy of signal on SCT

Note 1: The state of ST2 is undefined when the VDP is programmed to input line and frame sync. i.e. DC1=0 and BT4=0.

Note 2: The state of ST2 will only indicate frame sync in this mode if the VDP is synchronised with the incoming signals. If this is not the case, ST2 will remain inactive low. Refer to paragraph 3.8.3 for more details.

ST3 - Vertical display interval

This bit is active high during all scan lines which display active information. It goes active high at the bottom of the upper border and goes inactive at the top of the lower border.

ST4 - Not used: indeterminate value

ST5 - Ready

The state of this bit is a direct copy of the signal on the VDP Ready output line.

ST6 - Horizontal display interval

This bit goes active high during the active portion of each line. I.e. from the right hand edge of the left border to the left hand edge of the right border.

ST7 - Buffer memory overflow

This bit goes active high whenever the input buffer becomes full i.e. the Data Provider write pointer ACMT catches up with the CPU read pointer ACMP. This bit is cleared when the buffer is initialised by programing the base address of the start of the buffer BAMT.

ST8 - Buffer memory empty

This bit goes active high whenever the input buffer becomes empty i.e. the CPU read pointer ACMP catches up with the Data Provider write pointer ACMT. This bit is set when the buffer is initialised by programing the base addresss of the start of the buffer BAMT.

3.3.5 Command register 1 - CM1

CM1 is an eight bit write only register whose contents define the time base operation.

Its contents are shown in Figure3.4 and the significance of each individual bit is described in Table 3.4.

FIGURE 3.4: COMMAND REGISTER CM1 CONTENTS

```
.-----------------------------------------------.
| BT1 | BT2 | BT3 | BT4 | BT5 |  X  |  X  |  X  |
`-----------------------------------------------'
 MSB                                        LSB
```

TABLE 3.4: TIME BASE COMMAND REGISTER (CM1)

NAME	STATE	FUNCTION
BT1	0	VDP display output non-interlaced
	1	VDP display output interlaced (Note 1)
BT2	0	Display standard 1
	1	Display standard 2 (Note 2)
BT3	0	Not used - Must be zero (Note 3)
BT4		Combined with DC1 to control SLL and SCT synchronisation mode (See Table 3.5)
BT5	0	Normal display mode
	1	Subtitling mode (Note 4)

1. In interlaced frame mode, each alternate frame is composed of an odd number of half lines causing the display device to have a half line vertical offset on every other frame thus doubling the effective number of scan lines. The information displayed remains the same, however for both even and odd frames. In non-interlaced mode, all frames have the same even number of scan lines and are displayed identically.

It should be noted that in external synchronisation mode, the VDP will output its scan format, i.e. interlaced or not, controled uniquely by the style of the input sync signals and the state of BT1 will effectivly be ignored. If the VDP is programmed in external sync mode and no sync signals are input however, the VDP will free run in the mode defined by the state of BT1.

2. As described in paragraph 4.1, the VDP contains effectively two independant ROMs defining two different display formats. The state of BT2 defines which ROM is used to control the display timing.

3. BT3 is a bit used during the production of the device to programme the time base in a high speed test mode of operation. This mode is unsuitable for normal display devices and should be disabled by programming BT3 with a zero value.

4. Subtitling mode is described in paragraph 1.2.4 and 4.4.

TABLE 3.5: Control of SLL and SCT modes

DC1	BT4	I/O	SLL	SCT
0	0	Input	Line sync.	Frame sync.
0	1	Input	Line sync.	Composite sync.
1	0	Input	Line sync.	Composite sync.
1	1	Output	Line sync.	Frame sync.

N.B. - Although SLL and SCT are shown as inputing external synchronisation signals when DC1 = 1 and BT4 = 0 it should be noted that the VDP is nevertheless in internal sync mode with the timebase free running. The input signals in this case are used solely to generate the status bit ST2.

3.3.6 Command register 2 - CM2

CM2 is an eight bit write only register whose contents define the display decoder operation.

Its contents are shown in Figure 3.5 and the significance of each individual bit is described in Table 3.6.

FIGURE 3.5 : COMMAND REGISTER CM2 CONTENTS

DC1	DC2	DC3	DC4	DC5	DC6	DC7	DC8
MSB							LSB

TABLE 3.6 : DISPLAY COMMAND REGISTER (CM2)

NAME	STATE	FUNCTION
DC1	0 1	External synchronisation mode Internal synchronisation mode Combined with BT4, DC1 also defines SLL & SCT mode (Note 1)
DC2	0 1	Row 0 replaced by border colour Row 0 displayed normaly (Note 2)
DC3	0 1	Masking attribute disabled Masking attribute enabled
DC4	0 1	Incrustation attribute disabled Incrustation attribute enabled
DC5	0 1	Chr gen 3 treated as alphanumeric Chr gen 3 treated as alphamosaic (Note 3)
DC6	0 1	No grid display Grid display (Note 4)
DC7	0 1	Underlining attribute disabled Underlining attribute enabled
DC8	0 1	Normal incrustation mode (Note 5) Superimposition incrustation mode

1. Synchronisation modes are discussed in paragraph 3.8 and are shown in Table 3.5.

2. DC2 allows the display of Row 0 to be suppressed. Row 0 is the uppermost line of text of the display and when disabled is displayed with the uniform border colour and the contents of the page memory for this row is ignored. This allows a VDP which would normally output a 25 line display to instead output only 24 lines compatible with certain teletext and videotex standards.

3. Refer to section 4.2 for a description of the difference between an alpha-geometric and an alpha-mosaic character generator.

4. If the state of DC6 is 1 a grid will be displayed over the normal image. The grid is generated by forcing the first pixel of each line of each character and all of the last line of each character to be displayed in their complementary colour (i.e. the RGB outputs are inverted). This mode is usefull for editing or page composition terminals where the position of each character cell needs to be visible. Grid display is active only in text mode.

5. In normal incrustation mode, any character with this attribute applied will be accompanied with an active high level on the incrustation output signal I for the whole of the horizontal width of the character. This enables both the background and foreground parts of the character cell to be overlayed onto an external video image. In superimposition mode, however, the I output only goes active during the foreground part of the character causing incrusted characters to be overlayed onto external video without a surrounding 'box' of background colour. See section 4.1.

3.3.7 Command register 3 - CM3

CM3 is an eight bit write only register whose contents define the display mode and the memory parameters.

Its contents are shown in Figure 3.6 and the significance of each individual bit is described in Table 3.7.

FIGURE 3.6 : COMMAND REGISTER CM3 CONTENTS

CT1	CT2	CT3	CT4	CT5	CT6	X	X
MSB							LSB

TABLE 3.7 : MODE AND MEMORY REGISTER (CM3)

NAME	STATE	FUNCTION
CT1,CT2		Display mode
	CT1 CT2	
	0 0	Display disabled
	0 1	Text mode
	1 0	Bit-mapped mode
	1 1	Mixed mode
CT3	0	Not used - Must be zero
CT4	0	Normal CPU memory accesses
	1	CPU access inhibited during active display interval. (Note 1)
CT5	0	Memory access timing type 2 (Note
	1	Memory access timing type 1 2)
CT6	0	Address multiplexing for 64K DRAMs
	1	Address multiplexing for 16K DRAMs (Note 3)

Notes:

1. See paragraph 3.7.4

2. See paragraph 3.7.5

3. The 16 bit memory address is multiplexed onto the D0 - D7 address/data lines during the falling edges of RAS and CAS. Two different multiplexing schemes are selectable in order to comply with the differing row and column address formats of 16K and 64K DRAMs. The state of CT6 selects which type of multiplexing is used as shown in table 3.8 where A0 to A15 are the MSB and LSB resectively of the 16 bit logical address as seen by the CPU.

TABLE 3.8: ADDRESS MULTIPLEXING OPTIONS FOR 16K AND 64K DRAMS

Memory	CT6	ROW	ADDRESS							COLUMN	ADDRESS						
type	state	D0	D1	D2	D3	D4	D5	D6	D7	D0	D1	D2	D3	D4	D5	D6	D7
64K/1 or 16K/4	0	A8	A9	A10	A11	A4	A5	A6	A7	A0	A15	A2	A3	A12	A13	A14	A1
16K/1	1	0	A9	A10	A11	A5	A6	A7	A8	A1	A15	A3	A4	A12	A13	A14	A2

3.3.8 Command register 4 - CM4

CM4 is an eight bit write only register whose contents define full screen attributes used in the text mode of operation only. The contents and use of this register are described in paragraph 4.2.6.

3.4 Memory organisation and base address registers

The external memory associated with the VDP is of a totally generalised uncommited nature. This means that both the ammount of memory and its detailed segmentation and use is defined entirely by the user via the command and base address registers.

As far as memory size is concerned, the VDP can directly address up to 64K bytes without external paging hardware. There is no minimum limit to the memory size other than practical operational considerations and the user may choose the exact ammount to suit the particular application.

At power up the memory is totally unnasigned. Various areas of memory are allocated to specific functions by programmation of the base address registers as described below.

The various base address registers are listed in Table 3.9 for reference.

TABLE 3.9 BASE ADDRESS REGISTERS

Name	Address	Function
BAMT	8	Start of data buffer
BAMP	9	CPU base address
BAPA	A	Start of display
BAGC0	B	Start of chr gen 0
BAGC1	C	Start of chr gen 1
BAGC2	D	Start of chr gen 2
BAGC3	E	Start of chr gen 3
BAMTF	F	End of data buffer

The contents of BAMT and BAMTF define the start and finish of the area of memory allocated to the data input buffer. BAMT points to the first byte of the buffer and BAMTF points to the byte following the last byte of the buffer. The DMA controler uses this buffer area to store data input from the data provider. For description of buffer operation refer to paragraph 3.6.

The contents of BAMP are used in conjunction with the CPU auto-incrementing address pointer ACMP. Operation and use of these two registers is described in paragraph.3.5.3.

The contents of BAPA defines the start of the page display memory. A base address is not required to indicate the end of the page memory since the size of the latter will always be implicitly defined by the display mode selected. There may be any number of zones allocated to displayable information limited only by the total size of available memory - the VDP will simply display the zone currently indicated by the contents of BAPA.

Due to the double cycle technique used by the VDP to read the page memory, BAPA must always point to an even address.

The contents of BAGC0 to BAGC3 define the first bytes of the character generators 0 to 3 respectively. A maximum of four generators may be defined for any one page displayed, however, as for the page memory, there is no limit to the total number stored apart from the size of the memory itself, the four currently required being indicated by the contents of the base address registers. Each character generator can contain a maximum of 128 characters implying a size of 1280 bytes for each generator at a rate of 10 bytes per character. If four character generators are not required then there is no need to define the unused ones in the memory or even initialise their base address registers. Also, if the generators are not full then they may be overlapped to economise on memory space or even occupy the same physical area of memory.

3.5 Auto-incrementing addresss pointers

The VDP contains five sixteen bit auto-incrementing address pointers used for addressing the VDP memory. These are shown in table 3.10 and their use in relation to the base address registers and memory organisation is shown in Figure 3.22.

Table 3.10: AUTC-INCREMENTIMG ADDRESS POINTERS

POINTER	FUNCTION
ACMPxy	CPU PRIMARY ACCESS
ACMP	CPU SECONDARY ACCESS
ACPA	PAGE DISPLAY ACCESS
ACMT	BUFFER INPUT ACCESS
ACREF	REFRESH COUNTER

All accesses to the external memory system use one of these address pointers. Also, in a general sense, any access using one of these pointers will cause the value of that pointer to be incremented by one. Some of the pointers also undergo limit testing following the incrementation to determine whether their value has exceeded a predetermined range. This is explained more fully in the discussion that follows.

3.5.1 Accumulators associated with the CPU

Two auto-incrementing accumulators are associated with CPU accesses to the external memory. The pointers concerned are ACMP and ACMPxy. Their names are arbitary. Any CPU access will use one of these two pointers and, as explained above, the pointer used will have its value incremented by one following the access.

Obviously, before any access is made using one of these pointers, it must first be programmed with the address of the first memory location to be read or written. All subsequent accesses to the memory of the same type i.e. read or write will then use the pointer implicitly to access contiguous memory locations.

Figure 3.7 shows an imaginary diagram of the internal VDP hardware associated with these two pointers. Note that this is not necessarily the actual physical implementation in the VDP but simply a functionally equivalent structure for explanatory purposes.

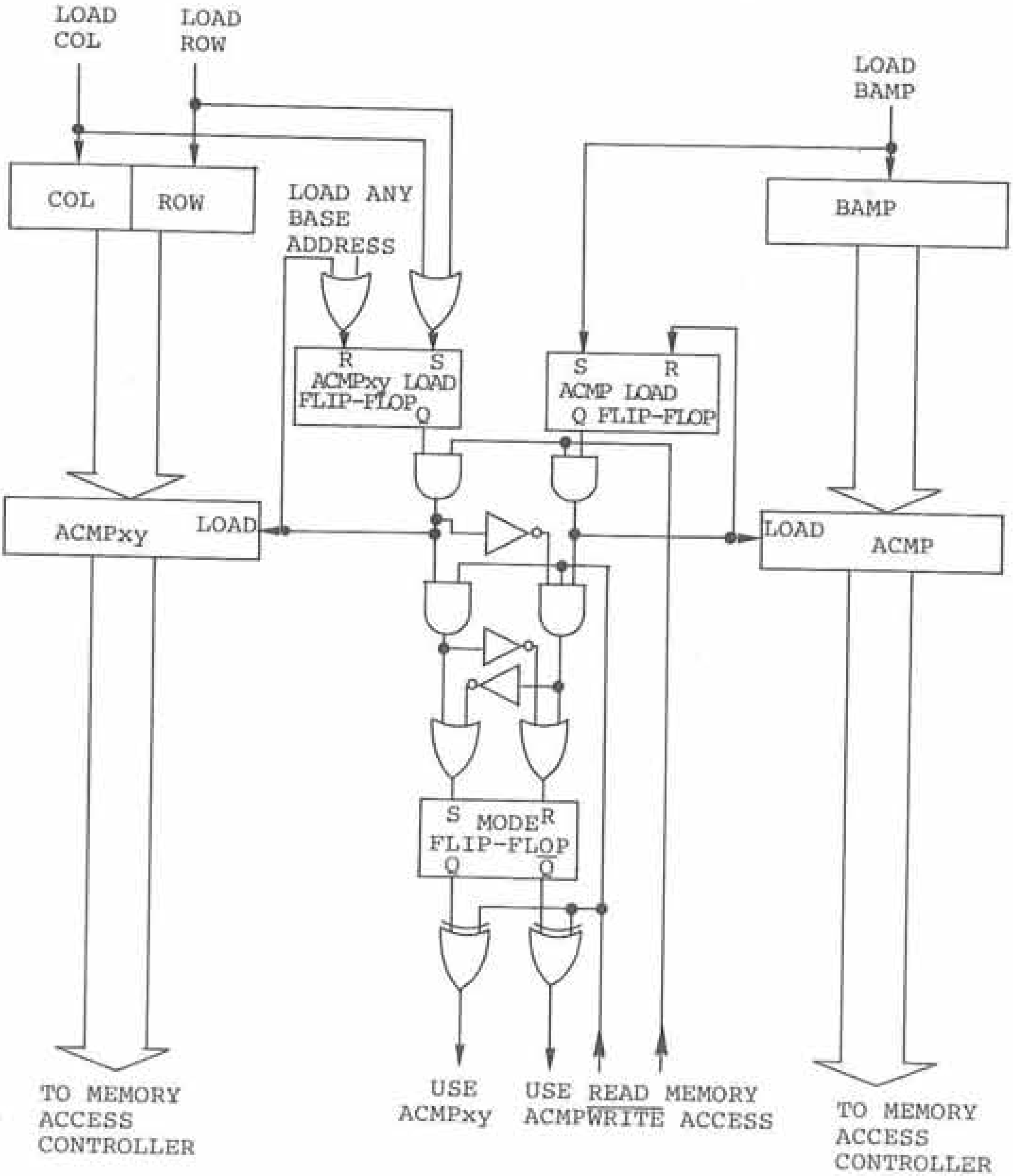
Three flip-flops are associated with the two pointers, each of the latter have what we will will term a 'load' flip-flop and a third 'mode' flip-flop is shared between the two.

Any time that either of the registers COL or ROW are programmed (using the technique explained in section 3.3.1) then the ACMPxy load flip-flop is set. Any time that a base address is programmed (using the technique explained in section 3.3.3) then the ACMPxy load flip-flop is reset. If the ACMPxy load flip-flop is found to be set at the time of a CPU memory access then the contents of both the registers COL and ROW will be transferred into ACMPxy and the latter will be used for the memory access with the consequential incrementation of its value afterwards. Additionally, two other actions occur. Firstly the ACMPxy load flip-flop is reset; this prevents COL and ROW from again being transferred into ACMPxy on the next access. Secondly, the mode flip-flop is set to attribute ACMPxy to the mode of access just performed i.e. if the memory access was a read, then ACMPxy is attributed to all subsequent read accesses and conversely its partner ACMP is attributed to all subsequent write accesses; if the memory access was a write, however, then ACMPxy will be attributed to all subsequent write accesses and ACMP to all subsequent read accesses. Note that the memory access itself and whether it be a read or write access is decoded by the VDP from the state of the E1 and E2 CPU interface lines as explained in paragraph 2.1.

Initialisation of ACMP is performed in a slightly different manner and makes use of the base address register BAMP.

Programming BAMP (using the technique explained in paragraph 3.3.3) has the effect of setting the ACMP load flip-flop; it also resets the ACMPxy load flip-flop because BAMP is a base address register as explained above. Subsequent accesses to other base addresses, however, will not reset the ACMP load flip-flop, thus once set, short of a total power down of the VDP, it will remain set until ACMP is loaded. The latter action takes place on the first following memory access at which time ACMP is loaded with the value of BAMP and is used for the access itself with the consequential incrementation of its value following the access. Additionaly, the ACMP load flip-flop is reset preventing ACMP from being re-loaded on subsequent memory accesses and the mode flip-flop is set to attribute ACMP to all

FIGURE 3.7 : SYMBOLIC CPU POINTER HARDWARE



subsequent accesses of the same type i.e. read or write, and ACMPxy to all subsequent accesses of the opposite type.

Once initialised in this way, further memory accesses may be made without explicitly redefining the desired access addresses. The accesses will use either ACMP or ACMPxy to supply the address information and increment the contents of the pointer used following the access. The pointer used will be the one attributed to the type of access performed - one pointer will always be attributed to read accesses and the other consequently to write accesses.

It is possible to initialise both ACMP and ACMPxy in one access operation:

COL and ROW are first programmed with the value desired for ACMP. This value is then transfered to BAMP causing the ACMP load flip-flop to be set and the ACMPxy load flip-flop to be reset as explained above. Now, instead of performing a memory access (which would load ACMP and leave ACMPxy untouched), COL and ROW are programmed with the value desired for ACMPxy. This causes the ACMPxy load flip-flop to be set as described above, but does not reset the ACMP flip-flop. Now, if a memory access is performed, we have both of the load flip-flops set with the result that both ACMPxy and ACMP will be loaded with the contents of COL and ROW and BAMP respectivly. We now have an apparent contention as to which pointer will be used for the access itself, however, the pointers are prioritized such that in this particular case ACMPxy will be used for the access and ACMP will be updated but not used. Both load flip-flops are reset following the loading of the pointers and the mode flip-flop is set to attribute ACMPxy to the type of access just performed and ACMP to the opposite type.

3.5.2 Accumulator associated with the display decoder

One of the auto-incrementing address pointers is associated with the display decoder. This is the pointer ACPA.

ACPA points to the current byte being read out of the page display memory. ACPA is initialised automaticaly by the internal VDP hardware with a copy of the base address BAPA whenever the VDP is reset by programmation of command register CM1.

ACPA is reset equal to the page display base address BAPA at the start of the active display area of each frame. Thereafter ACPA is incremented after each display access to the page memory thereby always pointing to the next byte to be accessed by the display controler. When ACPA reaches the address following the last byte in the page display memory it is reloaded with the value contained in BAPA and therefore points to the first byte of the page memory.

3.5.3 Accumulator associated with the data buffer

ACMT is the auto-incrementing accumulator used for data provider write accesses to the data input buffer. The base address registers BAMT and BAMTF define the begining and end of the buffer respectively.

During an access request by the data provider, it is the address contained in ACMT that is used for the access. ACMT is initialised whenever the base address BAMT is programmed by the CPU in which case it also receives the same address thus setting it to point to the first byte of the buffer. Remember that the value programmed into BAMT, and hence ACMT, will be one more than the value programmed in COL and ROW by the CPU. All data provider accesses use the contents of ACMT to address the memory and ACMT is then incremented so that it always points to the next address to be written by the data provider.

After each data provider access and consequential incrementation of ACMT the contents of the latter are compared to the base address BAMTF which contains the address of the byte following the last byte in the buffer. If the two values are found to be equal then ACMT is reprogrammed with the contents of BAMT and so points to the first byte in the buffer. In this way ACMT circulates between the two base addresses BAMT and BAMTF minus one.

The CPU auto-incrementing address pointer ACMP besides being a general purpose CPU pointer is also associated with the data buffer and is used by the CPU to read the data previously input by the data provider. For this reason the pointer ACMT besides being compared to BAMTF following every write is also compared to ACMP. If equality is found then the write pointer has caught up with the read pointer and the buffer is full. This is the buffer overflow condition and is indicated by the buffer overflow flag ST7 in the Status register. Further data provider access requests are thereafter ignored until the buffer is reset by programming BAMT whereupon ST7 is reset, ACMT reset equal to BAMT and data provider access requests are once more recognised.

To determine when the buffer is empty, each read from the buffer using ACMP is followed by a comparison between the latter and the write pointer ACMT. If equality is found then the read pointer has caught up with the write pointer and the buffer is empty. This condition is indicated to the CPU by the buffer empty flag ST8 in the Status register.

To determine when the CPU read pointer has reached the end of the designated buffer zone, ACMP is also compared to BAMTF following each access. If equality is found then ACMP is reprogrammed automatically with the contents of BAMP. Under normal circumstances BAMP will contain the same address as BAMT and so ACMP will then point to the first byte in the buffer.

It should be noted that the VDP has no way of distinguishing when ACMP is being used as a general purpose CPU pointer or as the buffer read pointer. For this reason it always tests for equality between it and the base address BAMTF following any access using ACMP with the consequential update of its value to BAMP if a match is found. This action could well cause unpredictable behaviour of ACMP if BAMTF contains an indeterminate value as it may well do in systems which do not make use of the data input buffer and so apparently have no need to programme BAMTF.

3.5.4 Auto-incrementing accumulator associated with refresh

ACREF is the address pointer used for refresh accesses to the dynamic RAM's. ACREF counts from zero to >FF00 in increments of >0100 therby scanning the 256 possible row addresses.

3.6 Buffer operation

As described above, a zone of memory is dedicated (not necessarily exclusively) to the data provider input by programing base address registers BAMT and BAMTF. The data provider may request a write access to this zone using the HMP handshake line. The VDP processes this request as any other memory access and starts the data provider access by initiating a DRAM write cycle. The cycle differs from a normal write cycle in that during the time the VDP normally outputs write data to the memories it instead tristates its data bus to enable the data provider to place its own data on the bus. This action is signaled by a low level on the return handshake line HIZ. The VDP then strobes the data into the memories with the WR line and returns HIZ to its inactive high state and the data provider in turn tristates its data bus completing the access cycle.

3.7 DMA controler

3.7.1 General

There are four possible subsystems which require access to the memory associated with the VDP: the display controler, refresh controler, data provider and the CPU.

The DMA controler is responsable for overseeing all accesses to the VDP's memory. This entails recognising the access requests, prioritising them relative to other possible pending accesses, synchronising them with the DMA clock and eventually, performing the access itself to VDP memory. A general block diagram of the memory access request controler appears in figure 3.8. Each of the various functions performed are described in the following paragraphs.

Requests for access to the VDP memory are input to the DMA controler via one of four channels corresponding to data provider, CPU, refresh, and display. The first two of these channels are accessible externally, the last three are internal to the VDP.

The data provider access request is initiated by the falling edge of the HMP input. For CPU accesses, a read access is timed from the falling edge of E2 and a write access from the rising edge of RWM.

3.7.2 Priority control

The various requests pass via a priority decision circuit which encodes the highest priority pending request based on the order shown in table 3.11.

TABLE 3.11

MEMORY ACCESS PRIORITY

PRIORITY	CHANNEL
1	Data provider
2	CPU
3	Display
3	Refresh

Normally, the CPU has priority over the display controller when accessing memory. Under some circumstances, e.g. with a high speed data provider input and a high density of CPU accesses, or, using the slow memory timing option, the display controller may miss accesses to the memory causing visual interference to the displayed image. To alleviate this situation to a certain extent, it is possible, by programming bit CT4 in command register CM3 to one, to disallow CPU accesses during the active part of each scan line (the latter is the only time that the display controller needs to access the memory). The CPU can determine whether an access will be allowed by testing bits ST3 and ST6 in the status register; if both of these bits are zero then the VDP is not in an active part of the display and the CPU may make an access. The ready output line and the ST5 status bit maintain their function in this mode and indicate that the VDP has completed the last requested write cycle, or, that valid data is present on the MP bus in the case of a read cycle.

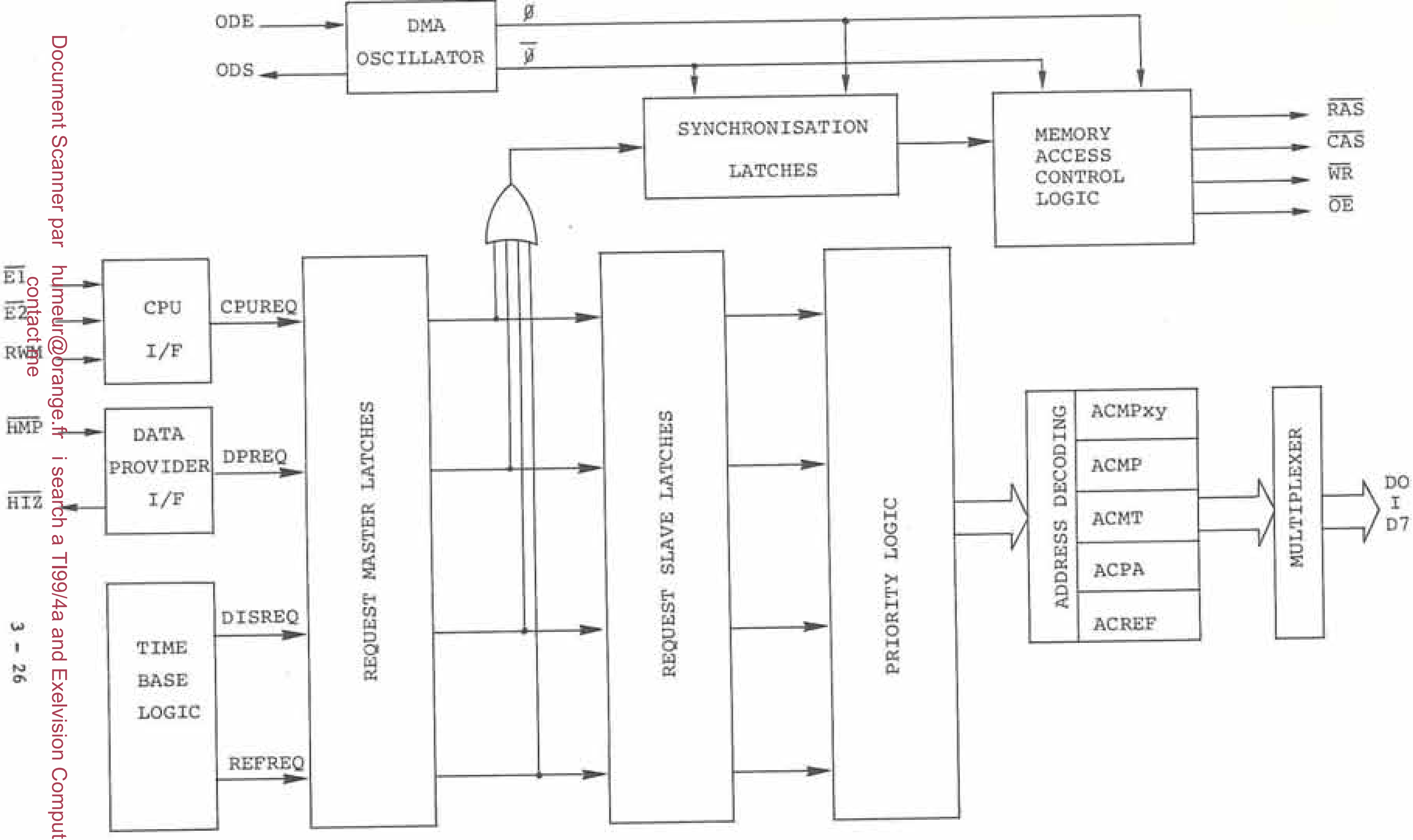


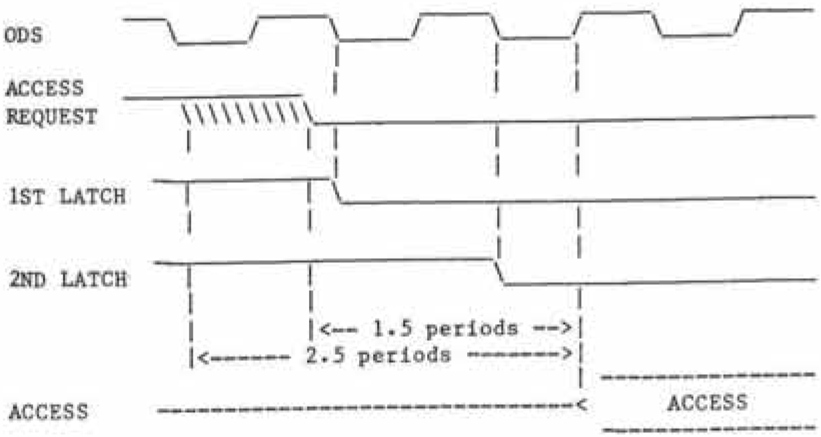
FIGURE 3.8 : MEMORY ACCESS REQUEST CONTROL

Although both display and refresh accesses have the same priority, they can never be requested at the same time as described in paragraph 3.7.6.

3.7.3 Request synchronisation

The access requests are sampled on each falling edge of the internal DMA clock as shown in figure 3.9. Since the access requests are typically asynchronous with respect to this clock it is quite possible that the minimum set up and hold time for the synchronising latch are violated causing a momentary indeterminate level at its output. To prevent this occurence affecting the following circuitry, the output of the first latch is sampled on the next falling edge of the internal DMA clock by which time it should have stabilised at one or other level. Although it is theoretically possible that a certain timing relationship between the access request and the DMA clock cause an indeterminate level to propagate through to the output of the second latch, the probability is so small as to be insignificant.

FIGURE 3.9 : REQUEST SYNCHRONISATION



3.7.4 Memory access control

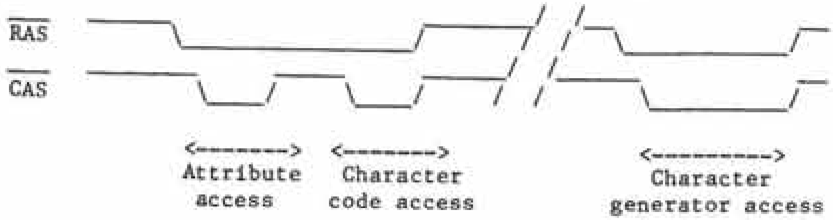
The section of the DMA controler dealing with the memory accesses themselves looks at the state of the access synchronisation latches on each rising edge of the internal DMA clock. In the cases where no access request is active the controler will simply enter an idle state for one clock cycle. As soon as an access request is detected the controler immediatly starts the DRAM access cycle and disregards further requests until the current cycle is finished whereupon it recommences sampling the latch outputs on each rising edge. If another access request becomes active more than one DMA cycle before the end of the current access, then the next access will be executed immediately following the current one. In this way any on-going cycle will always complete normally, and the DMA controler will then execute the highest priority pending access.

The access itself may be one of several types depending upon the channel requesting access. Notably, to increase memory througput, the DMA controler uses a double or page mode access for one of the request channels ascociated with the display controller.

As described in section 4, the VDP needs to read three bytes from the page memory in order to display eight points on the screen. The sequence of accessing these bytes depends upon the programmed display mode.

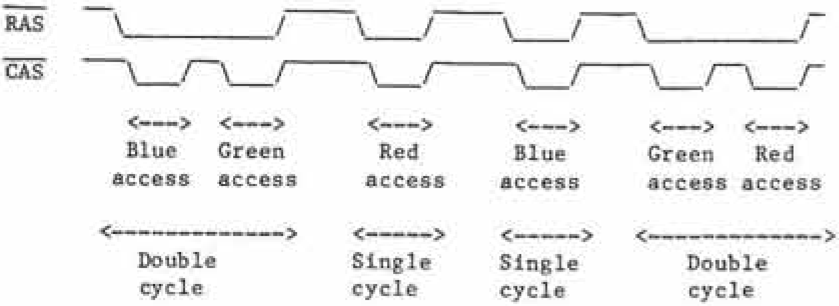
In text mode, for each line of each character displayed, the VDP first uses a double read cycle to access the two bytes containing the attributes and the character code followed by a single read access to the character generator as shown in Figure 3.10.

FIGURE 3.10: PAGE MEMORY ACCESS SEQUENCE IN TEXT MODE



In bit-mapped mode, each odd access to the memory employs a double cycle to read the blue and green bytes followed by a single cycle to read the red byte. Each even access uses a single cycle to read the blue byte followed by a double cycle to read the green and red bytes as shown in Figure 3.11. This alternating technique is used in order to ensure that restricting BAPA to an even address is sufficient to ensure that a double read cycle never crosses a 128 byte boundary.

FIGURE 3.11: Page memory access in bit-mapped mode



3.7.5 Memory access timing

Two memory timing options are available in the VDP. With CT5=1 the timing is suitable for standard DRAM's. Under some circumstances such as when slower memory is used, or with static memory, it may be desirable to lengthen the memory access cycles. This is effected by programming bit CT5 in command register CM3 to zero. Table 3.12 lists the memory timing parameters which differ between the two options together with their nominal values expressed in numbers of DMA clock periods. For the values of the parameters which are not functions of the memory timing option chosen refer to section 9.

TABLE 3.12: Memory timing options

PARAMETER	SYMBOL	CT5=1	CT5=0
RAS low pulse width	tw(RL)	3	5
CAS low pulse width	tw(CL)	2	4
WR low pulse width	tw(W)	1	3
OE low pulse width	tw(OE)	1	3
WR low before CAS high	tsu(WCH)	1	3
OE low before CAS high	tsu(OCH)	1	3
WR low before RAS high	tsu(WRH)	1	3
OE low before RAS high	tsu(ORH)	1	3
Data hold after WR low	th(WLD)	1	3
Single read cycle time	tc(RD)	5	7
Single write cycle time	tc(W)	5	7
CAS double read cycle	tc(P)	3	5
Double read cycle time	tc(D)	8	12
HIZ low pulse width	tw(HIZ)	4	6

The various accesses and the timing of the interface signals for the standard memory timing option are shown in figures 3.12, 3.13 and 3.14.

FIGURE 3.12 : SINGLE READ ACCESS (CPU OR DISPLAY)

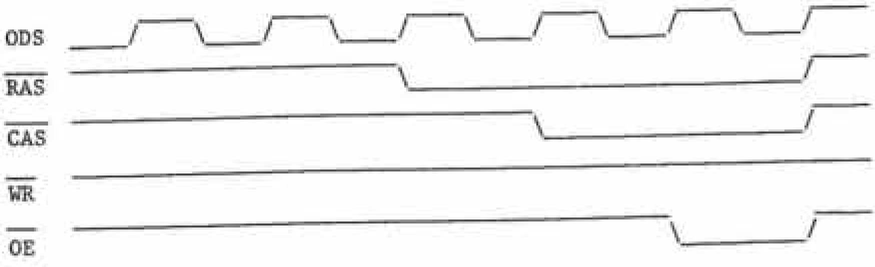


FIGURE 3.13: SINGLE WRITE ACCESS (CPU OR DATA PROVIDER)

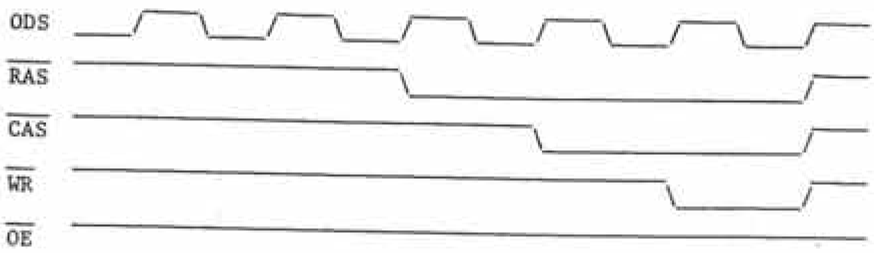
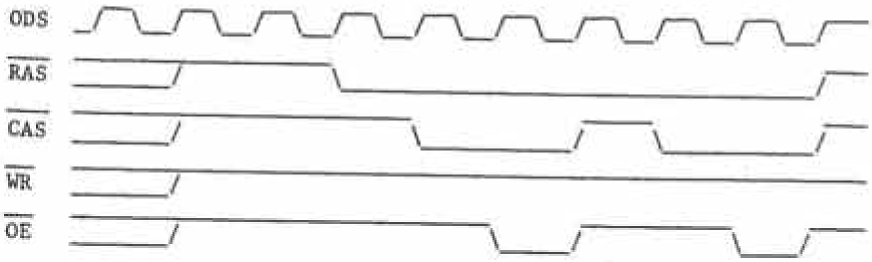


FIGURE 3.14 : DOUBLE READ ACCESS (DISPLAY)



3.7.6 Refresh cycles

The refresh cycles necessary for the correct operation of dynamic memories are controled by the VDP and carried out transparently to the other elements in the system.

To simplify the internal hardware of the VDP, the latter uses ordinary read cycles to refresh the memory with both RAS and CAS going active during the cycle (strictly, only RAS needs to go low for a refresh cycle, but a normal read cycle works just as well). In addition, since the highest density of memory accesses occurs during the active parts of each display line, so as not to increase the number of accesses during this worst case period, the refresh cycles are performed during the line blanking periods when the display controller is not accessing the memory. The centre of the line blanking period is indicated by the line sync pulse and five refresh cycles are performed during each one. With a typical line duration of 64us, all 256 rows will be refreshed every 3.28ms.

3.7.7 Access response times

From an access time point of view, it can be seen from figure 3.9 that a minimum of from 1 1/2 and 2 1/2 cycles are necessary between the start of the external access request and the start of the access itself. The variation of one DMA cycle depends upon the phase of the internal DMA clock at the time of the initial external request. In addition to this start up delay any access may further be delayed until the completion of any on-going cycle and the servicing of any higher priority access requests which may become active in the mean time. Table 3.13 shows the typical number of DMA clock cycles for each type of access.

TABLE 3.13: Length of various access cycles

ACCESS TYPE	ACCESS PRIORITY	DMA CYCLES PER ACCESS	
		Standard	Slow
DATA PROVIDER	1	5	7
CPU	2	5	7
DOUBLE DISPLAY	3	8	12
SINGLE DISPLAY	3	5	7
REFRESH	3	5	7

As an example, consider the calculation of the maximum access time for the CPU in a system which also contains a data provider:

Although the refresh cycles are shown having the same access priority as display requests, they can normally be disregarded for worst case access time calculations because they are always synchronised in such a way as to occur outside of the active part of each line as described in paragraph 3.7.6.

From table 3.13 it can be seen that with the standard memory timing option the longest access cycle is that for the double display request taking 8 DMA cycles. If the CPU access request falls just later than one DMA period before the start of this access, the former will have to wait 9 DMA cycles for it to complete. In addition, a higher priority data provider request may become active in the mean time, causing a further 5 cycle delay to execute it. Finally, there will be the delay of 5 cycles for the CPU access itself. The total delay time is therefore as follows:

Possible on-going double display access ..	9	cycles
Possible data provider access	5	cycles
CPU access itself	5	cycles
Total maximum access delay	19	cycles

In the case of a system without a data provider, the maximum wait time reduces to 14 DMA periods.

The minimum access time occurs when the CPU request falls just early enough so that the synchronisation delay is the minimum 1.5 periods and no other DMA access is on-going. The delay will be then 6.5 DMA periods:

Synchronisation and prioritising.........	1.5 cycles
CPU access itself	5 cycles
Total minimum access delay	6.5 cycles

3.7.8 Internal DMA cycles

The above description of access request, recognition and operation applies to accesses to the memory associated with the VDP. The same mechanism, however, also applies to CPU accesses to any of the base address registers. This means that the access time and operation of the interface signals should be calculated in the same way. Notably, the ready line will behave in exactly the same way for a CPU write to a base address as for a CPU write to VDP memory.

3.8 Time base

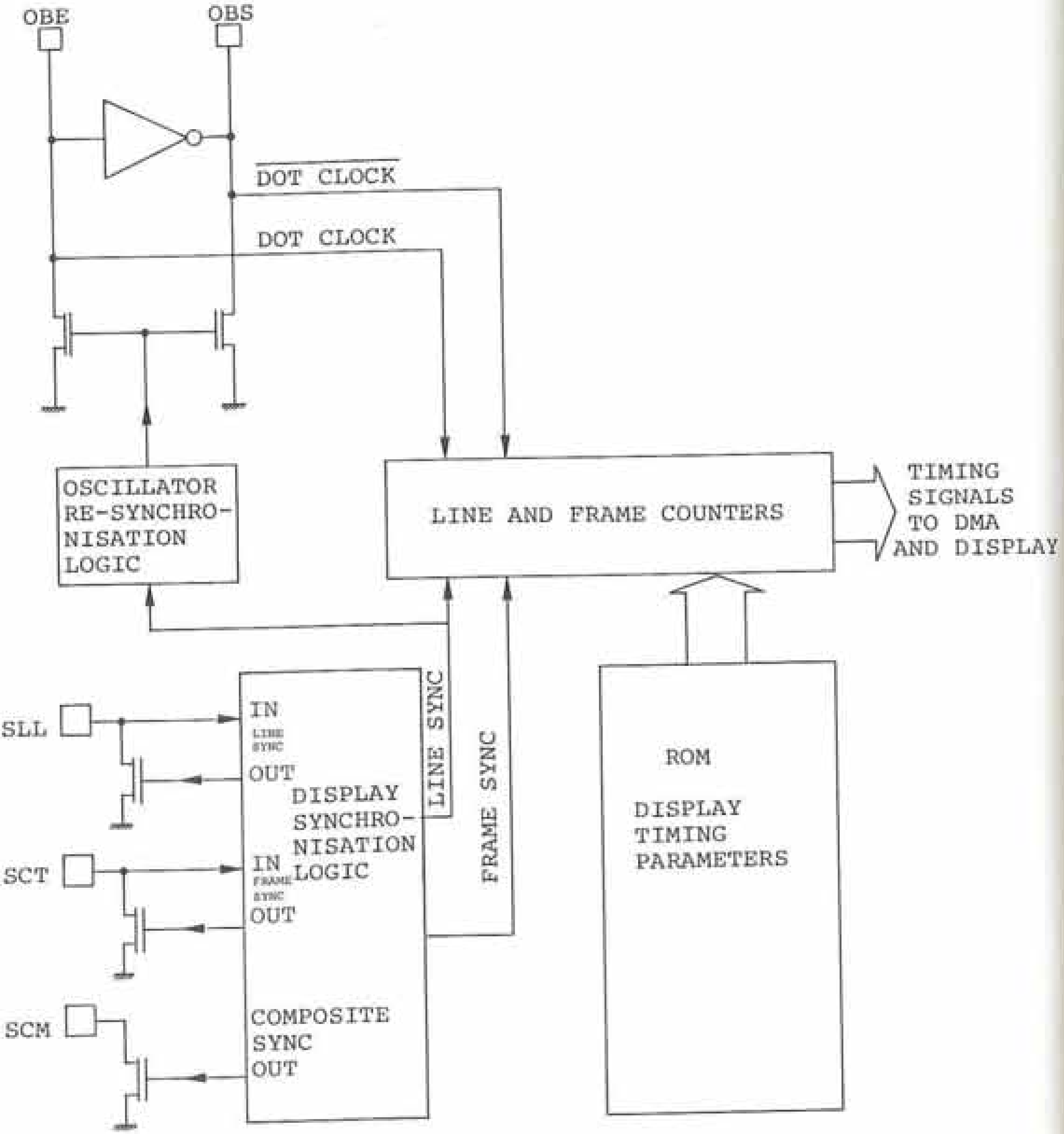
Figure 3.15 shows a block diagram of the Time Base. This is composed of four major sections: master oscillator, synchronisation logic, display counters and two mask programable areas of read only memory (ROM). The latter contain the display timing parameters enabling the VDP to switch between two different TV standards under software control. The ROM contents may be defined as required prior to fabrication. For more details refer to the section on TMS3556 family variants.

3.8.1 Display timing

The display timing is derived from the two complementary outputs of the master time base oscillator. The frequency is selected to be equal to the dot clock required; i.e. the duration of one displayed pixel will be exactly equal to one cycle of the time base oscillator.

It should be noted that since the various elements of the time base are wholly synchronous, any variation of the frequency of the master oscillator will be translated directly into a proportional variation of the parameters of the display timing. Thus to guarantee a certain maximum variation of the

FIGURE 3.15 : TIME BASE BLOCK DIAGRAM



display parameters it is sufficient to apply the same maximum variation limit in percentage terms to the master time base oscillator. If OBE is being driven by an externally derived clock there is little problem. If an LC network is used however, it should be remembered that the variation in output frequency will be roughly equal to the variation in the values of the components; for example, if components with a maximum variation of 10% are used, then the maximum variation of time base frequency will also be 10%.

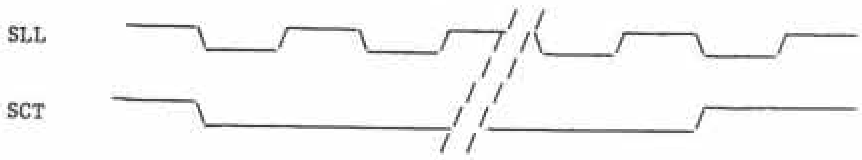
3.8.2 Display synchronisation

To permit the display CRT or monitor to display a stable picure synchronised with the RGB outputs from the VDP, both devices must be perfectly synchronised together. There are two basic modes of accomplishing this: either the CRT synchronises itself to the signals provided by the VDP or vice versa. The former is internal sync mode, the latter external. The two alternatives are selected under software control by the CPU by programming the relevant values into the on chip control registers.

In internal sync mode, two alternative methods of synchronising the TV or monitor are available depending upon the requirements of the display device being used. The first is to use the composite sync signal output on the SCM pin. The various timing parameters of this signal are defined by the contents of the time base ROM's. It's usual general form is illustrated in figure 3.16

The second alternative is to drive the CRT directly from the individual line and frame sync outputs SLL and SCT. Again the exact timing of these signals is controled by information stored in the time base ROM's; typical forms for these signals are shown in figure 3.17

FIGURE 3.17: TYPICAL LINE AND FRAME OUTPUT SIGNALS



Since these signals are derived by programable division of the master dot clock, they will be perfectly synchronous with the latter and their respective periods will be exact integer multiples of the dot clock period.

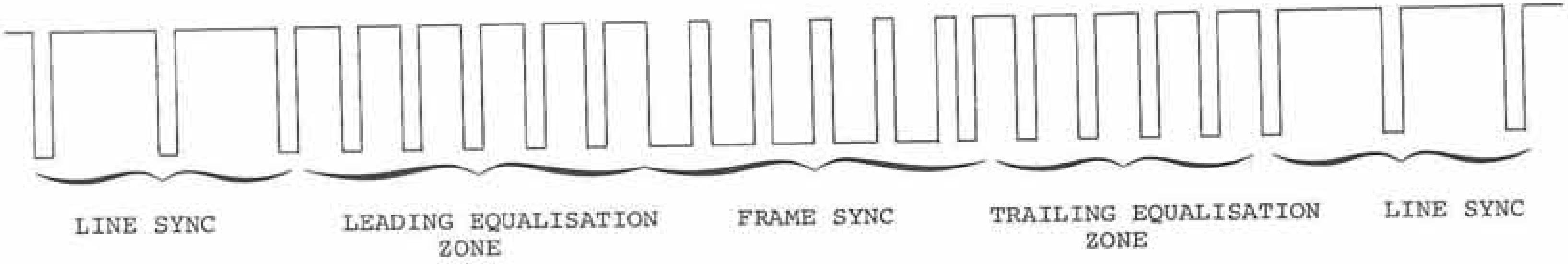


FIGURE 3.16 : TYPICAL COMPOSITE SYNC OUTPUT SIGNAL

3.8.3 External synchronisation

In external sync mode the VDP display output is synchronised to signals provided by the display device. Two variants of this mode are possible: either line and frame sync are input or alternativly line and composite sync.

In the first case, the falling edges of the two input signals are used to directly reset the line and frame counters to zero. To allow for possible phase offset of the two input signals, and to ensure that both counters are reset simultaneously, the falling edge of the line sync input generates a pulse whichs gates the frame sync pulse onto the chip. Thus the falling edge of the frame sync pulse must occur prior to, or coincident with the falling edge of the line sync pulse. Additionally, it should be noted that the status bit ST2 is undefined in this mode.

In the case where composite sync is input on SCT, the VDP uses the line sync signal input on SLL directly to reset the line counters, and also to detect the leading and trailing equalisation pulses present in the composite sync signal. If this event occurs outside of the active vertical portion of the display, i.e. whilst the status bit ST3 is zero, the status bit ST2 is set active high, the frame counter is reset, and the next frame commences one line after the trailing equalisation period, i.e. at the end of the first line found without an intermediate half line pulse as shown in figure 3.18. If the leading and trailing equalisation pulses occur during the active vertical part of the display however, they will be ignored, the VDP will not reset its vertical counters and the status bit ST2 will remain inactive low.

In both modes of external synchronisation the line sync signal input on SLL is used directly to reset the horizontal counters and the VDP immediately commences the next scan line.

In addition, the VDP resynchronises the time base oscillator at the start of each new line, i.e. on each falling edge of the SLL input. This is accomplished by momentarily pulling both oscilator pins OBE and OBS to ground thereby removing any stored energy in the external LC network. This is done so that the phase of the VDP's dot clock is held constant with respect to the input line sync and is independant of the phase of the dot clock at the end of the preceeding line. If this were not done, each line would start with a random phase of the dot clock oscillator translating to pixel flutter in the displayed image.

It should be noted that the composite sync output signal on SCM is derived directly from the display timebase. Since the latter is asynchronously reset by external signals at the start of each line and frame, the form and timing of SCM may

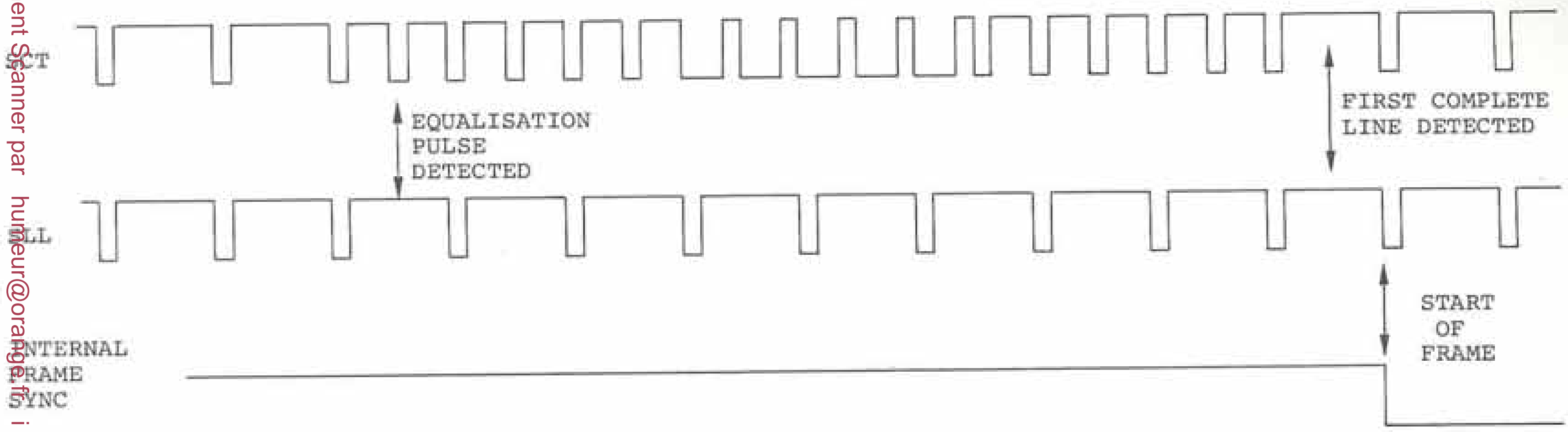


FIGURE 3.18 : EXTERNAL SYNCHRONISATION WITH LINE AND COMPOSITE INPUTS

be considerably altered in external sync mode and will in some cases be unsuitable for display synchronisation. A sync signal derived from the same source as the sync signals input to the VDP should instead be used.

Even though the VDP is programmed in external sync mode it nevertherless continues to generate sync signals internally which are overidden by the external ones. If the external sync signals are removed, the VDP will continue to run at its natural frequency defined by its timebase oscillator and output RGB and SCM signals. Care must be taken when reapplying external sync since the phase of the latter with respect to the internal sync may mean that the first line sync pulse occurs during the active part of a displayed line with the resultant possible corruption of on-chip registers as described in paragraph 6.4.

Reference should be made to section 6 for further hardware considerations to be taken into account when operating the VDP in external synchronisation mode.

3.8.4 Display decoder timing

It is of interest to examine the timing of the display decoder and when display accesses are actually made to memory.

The timing of the display logic varies slightly between text and bit-mapped mode and will be discussed separtately.

As described in paragraph 3.7.3 in text mode the display decoder requests a double access and a single access to the page memory once every character period. Since the display decoder cannot process the information instantaneously, a pipeline system is used as shown in Figure 3.19. At the start of each character period a double access is requested to read the attribute and character code bytes of the next character to be processed. The character code thus read is combined with the character generator base address to generate the address to be used for the next single access request. The latter is made at the start of the next character time. Thus at the start of each character time, the decoder requests a character generator access for character n and an attribute and code access for character n+1 as shown in Figure 3.20.

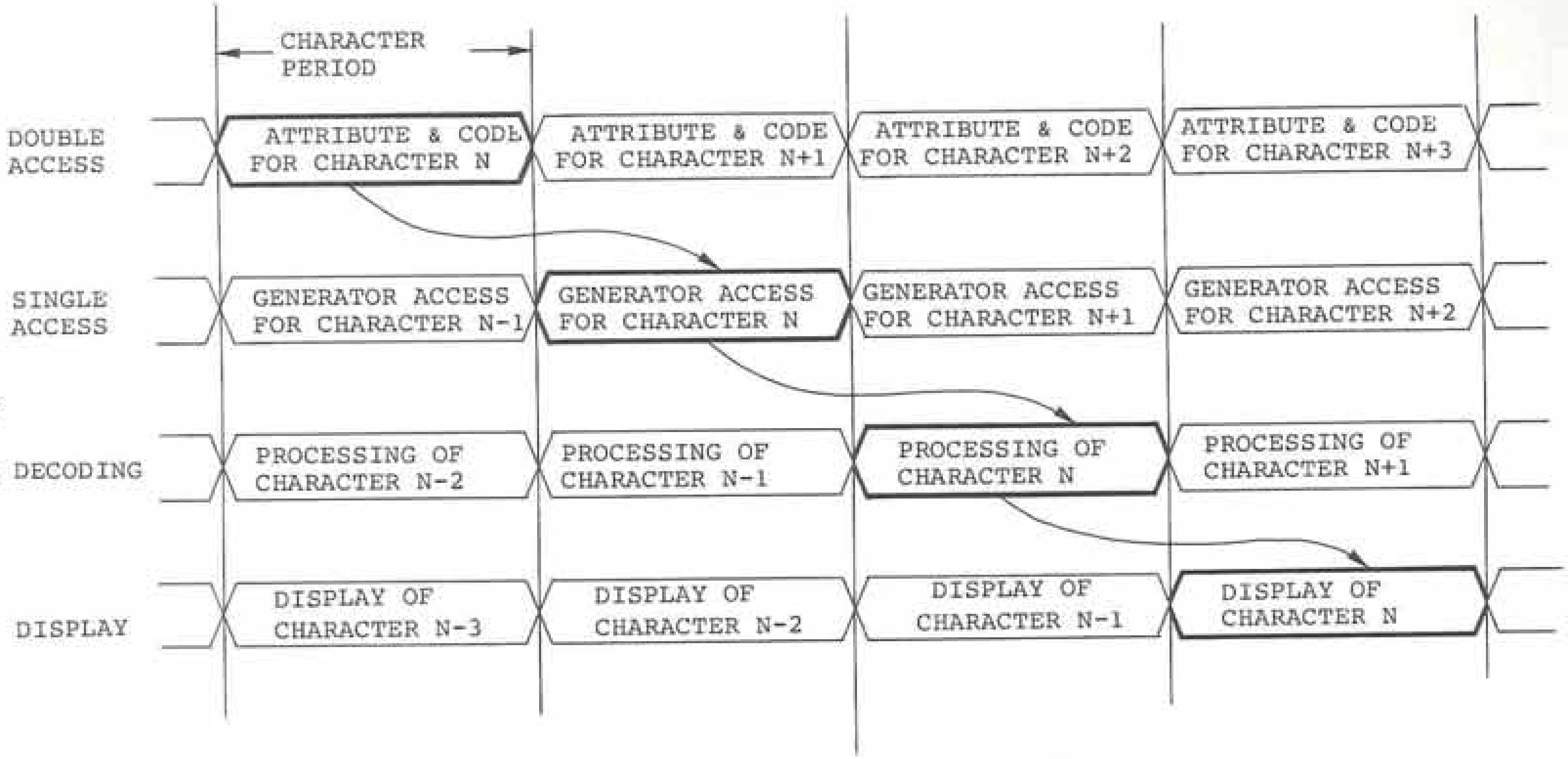
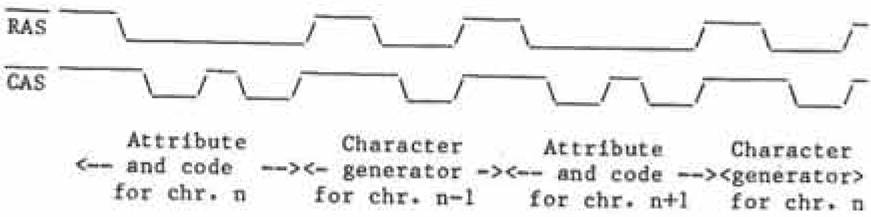

CHARACTER PERIOD
DOUBLE ACCESS
SINGLE ACCESS
DECODING
DISPLAY
ATTRIBUTE & CODE FOR CHARACTER N
ATTRIBUTE & CODE FOR CHARACTER N+1
ATTRIBUTE & CODE FOR CHARACTER N+2
ATTRIBUTE & CODE FOR CHARACTER N+3
GENERATOR ACCESS FOR CHARACTER N-1
GENERATOR ACCESS FOR CHARACTER N
GENERATOR ACCESS FOR CHARACTER N+1
GENERATOR ACCESS FOR CHARACTER N+2
PROCESSING OF CHARACTER N-2
PROCESSING OF CHARACTER N-1
PROCESSING OF CHARACTER N
PROCESSING OF CHARACTER N+1
DISPLAY OF CHARACTER N-3
DISPLAY OF CHARACTER N-2
DISPLAY OF CHARACTER N-1
DISPLAY OF CHARACTER N

FIGURE 3.19 : PIPELINE DISPLAY TIMING IN TEXT MODE

FIGURE 3.20: Memory access sequence in text mode



At the start of the next character time the attributes and character generator information are passed to the decoder itself which processes the information during the next character time and finally outputs the resultant RGB information one character later as shown in Figure 3.19. Thus the time from the first access request concerning a charcter (to access the attribute and code byte) to the start of that character being output on the RGB lines is three character periods.

In the case of bit-mapped mode, the sequence is slightly simpler since no point processing is required. Figure 3.21 shows the access timing and indicates a delay of the equivalent of two character times from the start of the accesses to the output on RGB.

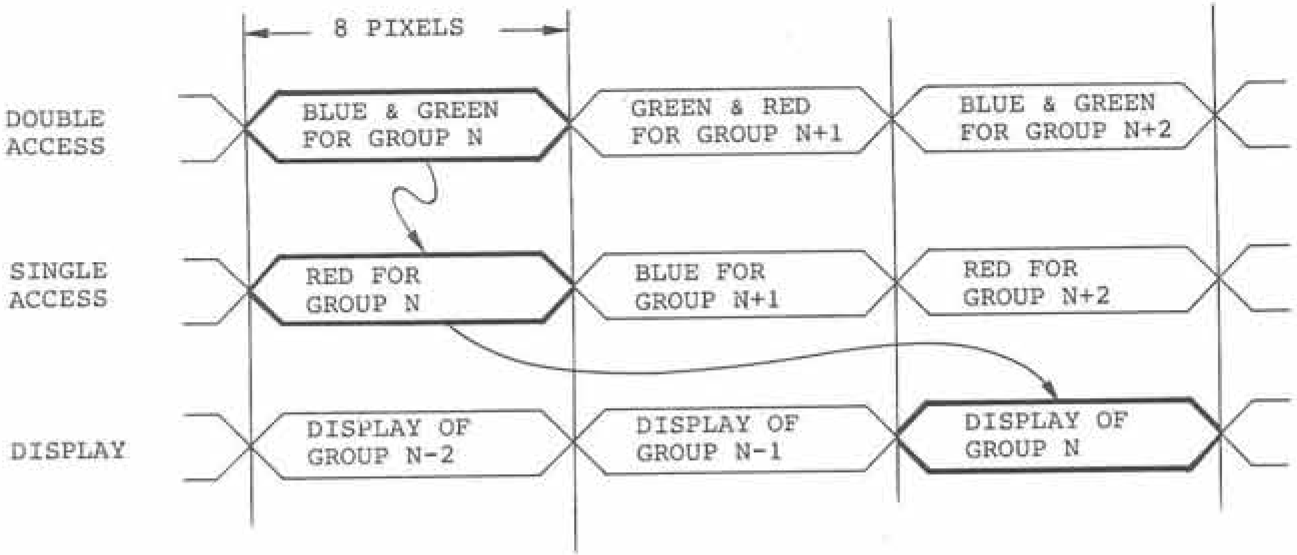
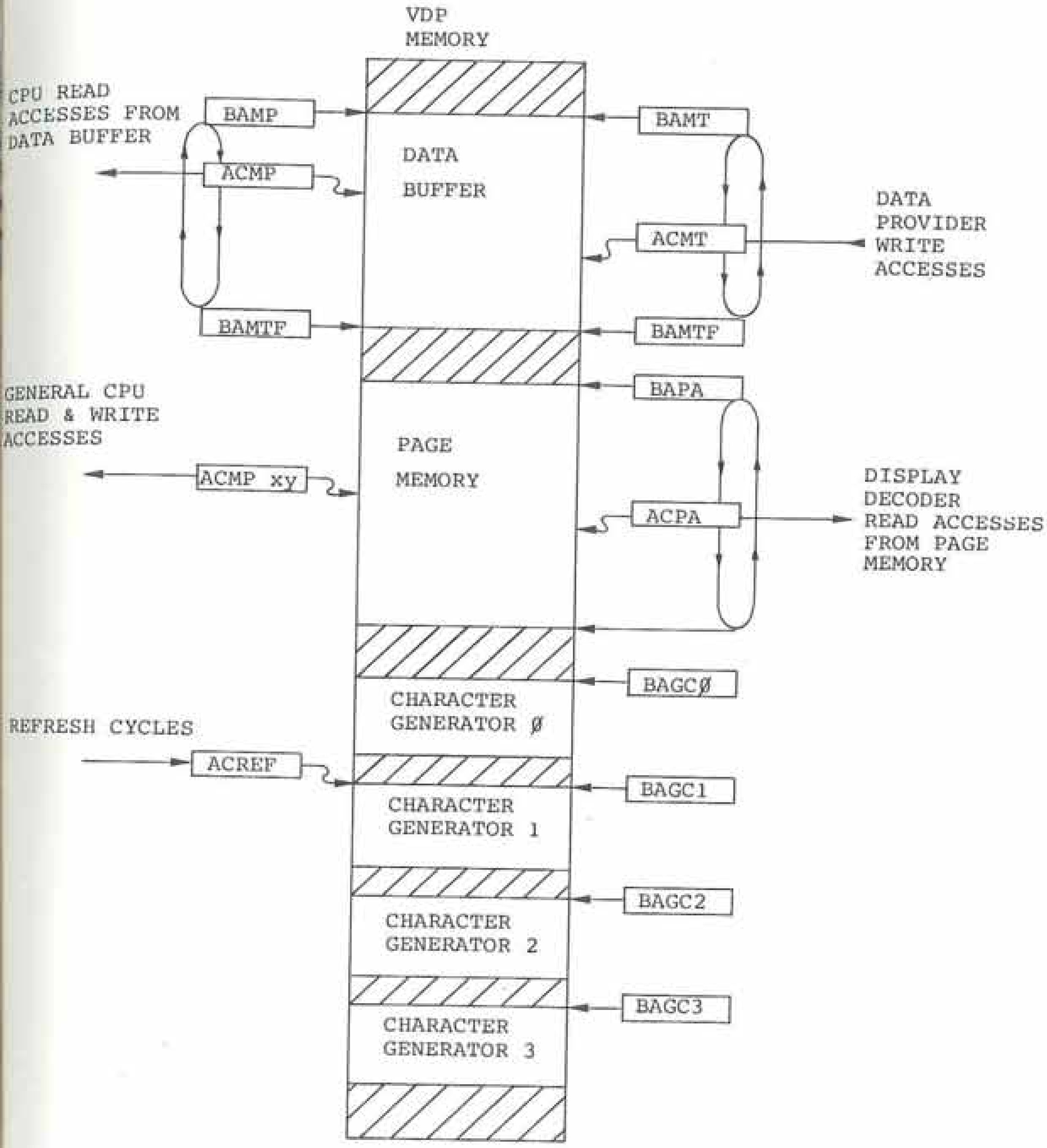


FIGURE 3.21 : PIPELINE DISPLAY TIMING IN BIT-MAPPED MODE

FIGURE 3.22 : BASE ADDRESS AND AUTO-INCREMENTING POINTER MAP



4 OPERATION

The base address BAPA is programmed with an address in the VDP memory pointing to the first byte of an area of memory containing the information needed by the VDP to generated the displayed image. This memory area is read continuously by the VDP display controller in synchronism with the scanning of the display device. The size of this page display memory and the exact nature of the information stored in it are functions of the mode of operation of the VDP programmed in the on-chip command registers.

4.1 Display standard options

As previously described, the VDP contains two areas of ROM who's contents are defined by a mask during the production of the device. These two ROMs contain the timing parameters of two independant display formats.

The standard VDP, identified by the code 'XA8628' contains the parameters for standard 625 and 525 line television displays as shown in Table 4.1. The ROM used by the time base is selected by the state of bit BT2 in command register CM1. Typical time-base oscillator frequencies are 7.25MHz in 625 line mode and 6.047MHz in 525 line mode giving line durations of 64us and 63.5us respectively.

For other TMS3556 variants, refer to section 11.

TABLE 4.1: TIME BASE PARAMETERS FOR STANDARD (XA8628) VDP

PARAMETER	Programmed times		UNIT
	BT2=0	BT2=1	
Vertical sync period (frame length)			
non-interlaced mode	626	524	Half-lines
interlaced mode	625	525	----"----
Character width	8	8	Dots
Horizontal sync period (line length)	58	48	Characters
Horizontal sync pulse width	4	4	----"----
Blanked area after falling edge			
of horizontal sync pulse	9	7	----"----
Left hand border	4	4	----"----
Horizontal display area	40	32	----"----
Right hand border	4	4	----"----
Blanked area before falling edge			
of horizontal sync pulse	1	1	----"----
Leading equalisation duration or blanked area before vertical sync pulse			
non-interlaced	6	6	Half-lines
interlaced - even frames	5	7	----"----
- odd frames	6	6	----"----
Vertical sync pulse width	5	5	----"----
Trailing equalisation duration			
non-interlaced	5	6	----"----
interlaced - even frames	5	6	----"----
- odd frames	4	5	----"----
Blanked area after falling edge of			
vertical sync pulse			
non-interlaced	40	40	-----"----
interlaced - even frames	40	40	----"----
odd frames	39	39	----"----
Upper border height - normal mode	36	32	----"----
- subtitling mode	476	392	----"----
Vertical display area - normal mode	500	420	----"----
- subtitling mode	60	60	----"----
Lower border height			
non-interlaced	44	26	----"----
interlaced - even frames	44	26	----"----
- odd frames	44	28	----"----
SLL output low level pulse width	29	24	Characters
SCT output low level pulse width			
non-interlaced	40	40	Half-lines
interlaced - even frames	40	40	----"----
- odd frames	39	39	----"----

4.2 Text mode

The VDP is placed in text mode by programming the mode control bits CT1 and CT2 in command register CM3 to zero and one respecively.

4.2.1 Page memory

In text mode, the display is made up of regular equally sized cells which contain a displayed character or part of a displayed character.

With the standard timebase programmation of the VDP there are 1000 of these character cells on one display screen, organised as 25 rows of 40 characters in 625 line mode or 672 characters organised as 21 rows of 32 characters in 525 line mode as shown in figure 4.1.

One of the unique features of text mode, sometimes refered to as typographic mode, is that only characters of a predefined shape may be displayed on the screen. The predefined characters themselves are stored in an area of memory separate to that of the page display memory called a character generator. The VDP can display characters from up to four of these predefined generators each containing up to 128 characters giving a maximum possibility of 512 different characters on one displayed screen. Each character is defined in a matrix of 8 pixels horizontally by 10 pixels vertically as shown in figure 4.2. The four character generator tables may, like the page display memory itself, be located anywhere in the available VDP memory that the user choses, the first entry of each being indicated to the VDP by the programmation of the on-chip character generator base addresses BAGC0, BAGC1, BAGC2 and BAGC3. This in itself allows any number of generator tables to be stored in the memory available, four of which can be refered to on any one page.

Additionally, although the character generators are refered to as 'predefined', because they reside in RAM their contents may be dynamically modified at any time - even while the VDP is using their contents for display. The feature in videotex terminals known as Dynamically Redefinable Character Sets (DRCS) is therefore intrinsically compatible with the VDP's own operation.

The page memory itself contains two bytes of information to define almost all the visual features of each character cell. The exceptions are serial attributes and are described later. With the normal page structure of 25 rows of 40 characters this means that a total of 2000 bytes are required to define one page in text mode as shown in figure 4.3.

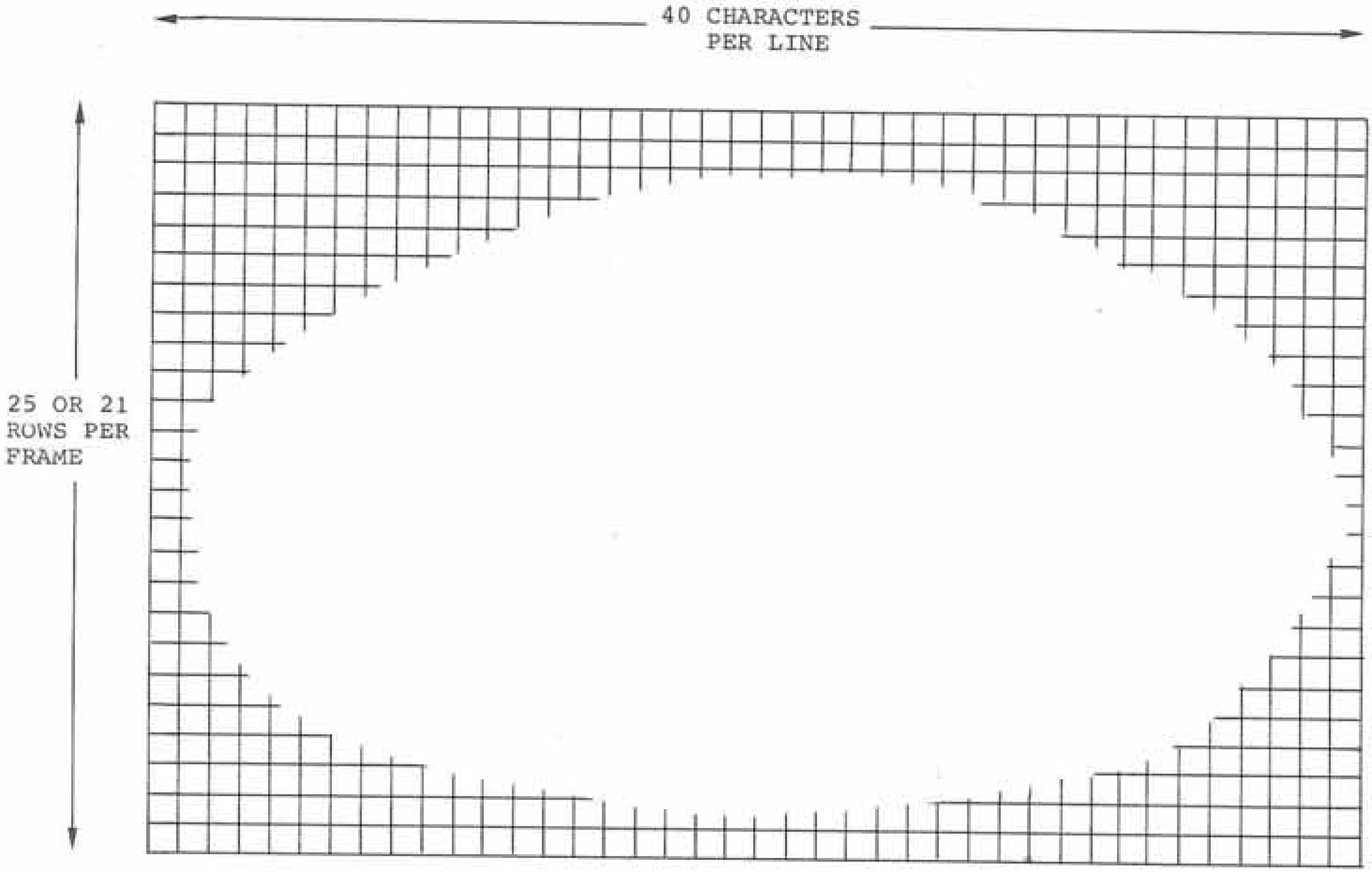
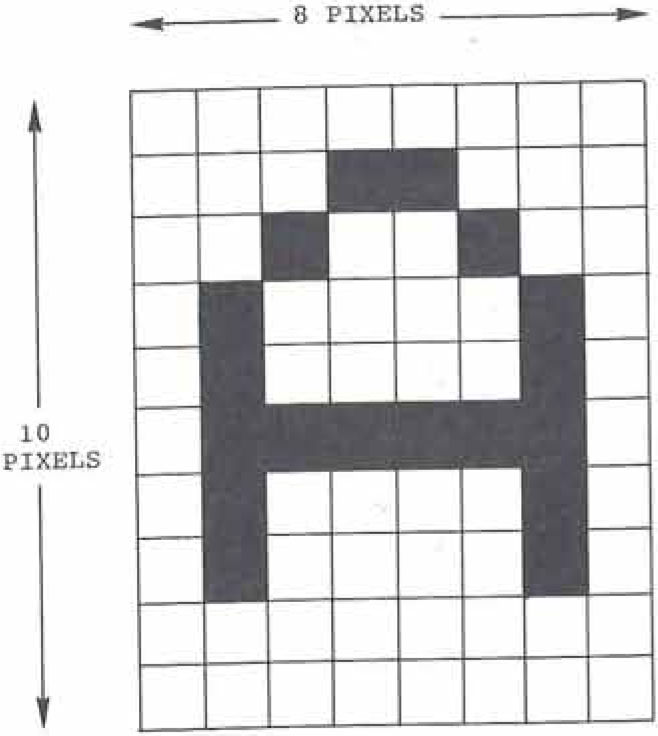
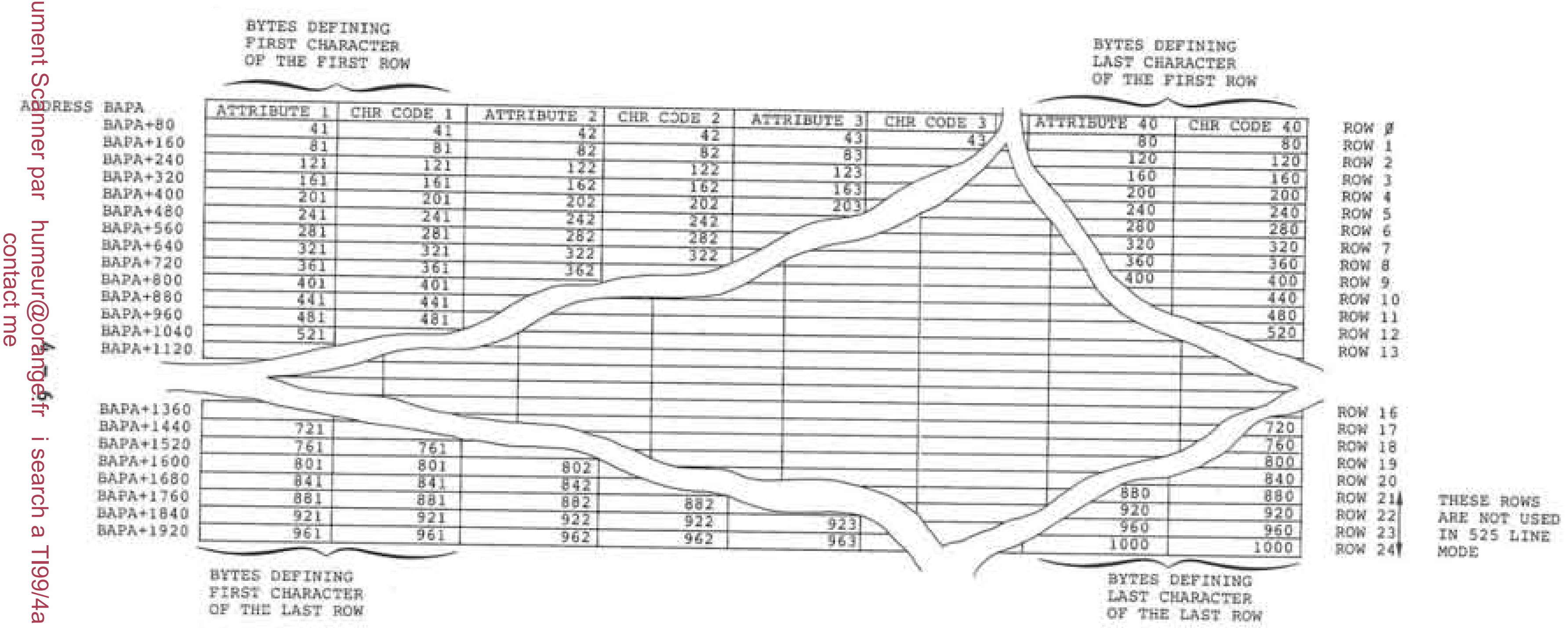


FIGURE 4.1 : TEXT MODE DISPLAY FORMAT (625 OR 525 LINES)

FIGURE 4.2 : CHARACTER MATRIX




BYTES DEFINING FIRST CHARACTER OF THE FIRST ROW
BYTES DEFINING LAST CHARACTER OF THE FIRST ROW
BYTES DEFINING FIRST CHARACTER OF THE LAST ROW
BYTES DEFINING LAST CHARACTER OF THE LAST ROW
THESE ROWS ARE NOT USED IN 525 LINE MODE

ADDRESS BAPA	ATTRIBUTE 1	CHR CODE 1	ATTRIBUTE 2	CHR CODE 2	ATTRIBUTE 3	CHR CODE 3	ATTRIBUTE 40	CHR CODE 40	ROW Ø
BAPA+80	41	41	42	42	43	43	80	80	ROW 1
BAPA+160	81	81	82	82	83		120	120	ROW 2
BAPA+240	121	121	122	122	123		160	160	ROW 3
BAPA+320	161	161	162	162	163		200	200	ROW 4
BAPA+400	201	201	202	202	203		240	240	ROW 5
BAPA+480	241	241	242	242			280	280	ROW 6
BAPA+560	281	281	282	282			320	320	ROW 7
BAPA+640	321	321	322	322			360	360	ROW 8
BAPA+720	361	361	362				400	400	ROW 9
BAPA+800	401	401						440	ROW 10
BAPA+880	441	441						480	ROW 11
BAPA+960	481	481						520	ROW 12
BAPA+1040	521								ROW 13
BAPA+1120									
									ROW 16
BAPA+1360								720	ROW 17
BAPA+1440	721							760	ROW 18
BAPA+1520	761	761						800	ROW 19
BAPA+1600	801	801	802					840	ROW 20
BAPA+1680	841	841	842				880	880	ROW 21↓
BAPA+1760	881	881	882	882			920	920	ROW 22
BAPA+1840	921	921	922	922	923		960	960	ROW 23
BAPA+1920	961	961	962	962	963		1000	1000	ROW 24↑

FIGURE 4.3 : PAGE MEMORY STRUCTURE IN TEXT MODE

4.2.2 Attributes

The two bytes per character give a total of 16 bits containing the information required to define the contents of each character cell. The first 9 bits, i.e. the whole of the first byte and the first, or most significant bit of the second byte contain what are called the 'attributes' of the character. These indicate, amongst other things, from which of the four character generators the shape of the character is to be taken. The 7 remaining bits define which of the 128 characters of the defined generator is to be displayed.

Figure 4.4 shows the contents of the two bytes defining the character cell.

Figure 4.4: PAGE MEMORY CONTENTS DESCRIBING ONE CHARACTER CELL

```
                     Even byte                               Odd byte
         ______________/\_______________________         _______/\______
        /                                       \       /               \
 .---------------------------------------------------.  .----------------.
 | BF | GF | RF | CG1 | CG0 | INV | DH | DW |  | FLS | CHR CODE |
 `---------------------------------------------------'  `----------------'
```

Table 4.2 describes the significance of each bit.

Table 4.2 : BIT SIGNIFICANCE OF TEXT MODE PAGE MEMORY CONTENTS

Bit	Description
BF GF RF	Foreground character colour.
CG1 CG0	Character generator containing the character to be displayed. CG1 is the most significant bit.
INV	Inversion
DH DW	DH DW Character size 0 0 Single 0 1 Double width 1 0 Double height 1 1 Double size
FLS	Flashing
CHR CODE	Character number in the defined generator (7 bits)

As described below, for each entry in the character generator, the latter contains a pattern of 1's and 0's defining the shape of the character. The 1's in the definition represent the foreground of the character and, in the absence of other overideing attributes, are displayed with the colour defined by the bits B, G and R.

If both of the bits DH and DW are zero, the character is displayed in single size, that is to say that its representation on the screen occupies one character cell. If one or both of these bits are a one, then the character will occupy two or four adjacent cells on the display screen. All of the byte pairs referring to the cells occupied must be programmed with the same content definition as shown in Figure 4.5. Additionally, any row of the displayed screen may only contain either the top halves of double height characters or the bottom halves, but not both at the same time. This is shown pictorially in figure 4.6. To ensure correct alignement of characters with and without the double height attribute applied, the first row of all double height characters is displayed three times, and the last only once, the others being displayed twice, as shown in Figure 4.7.

If the INV bit in the attribute definition is set to one, the character will be displayed with the foreground and background colours interchanged, that is, the 1's in the character definition will be displayed with the current background colour, and the 0's with the foreground colour defined by the B, G and R bits as shown in Figure 4.8.

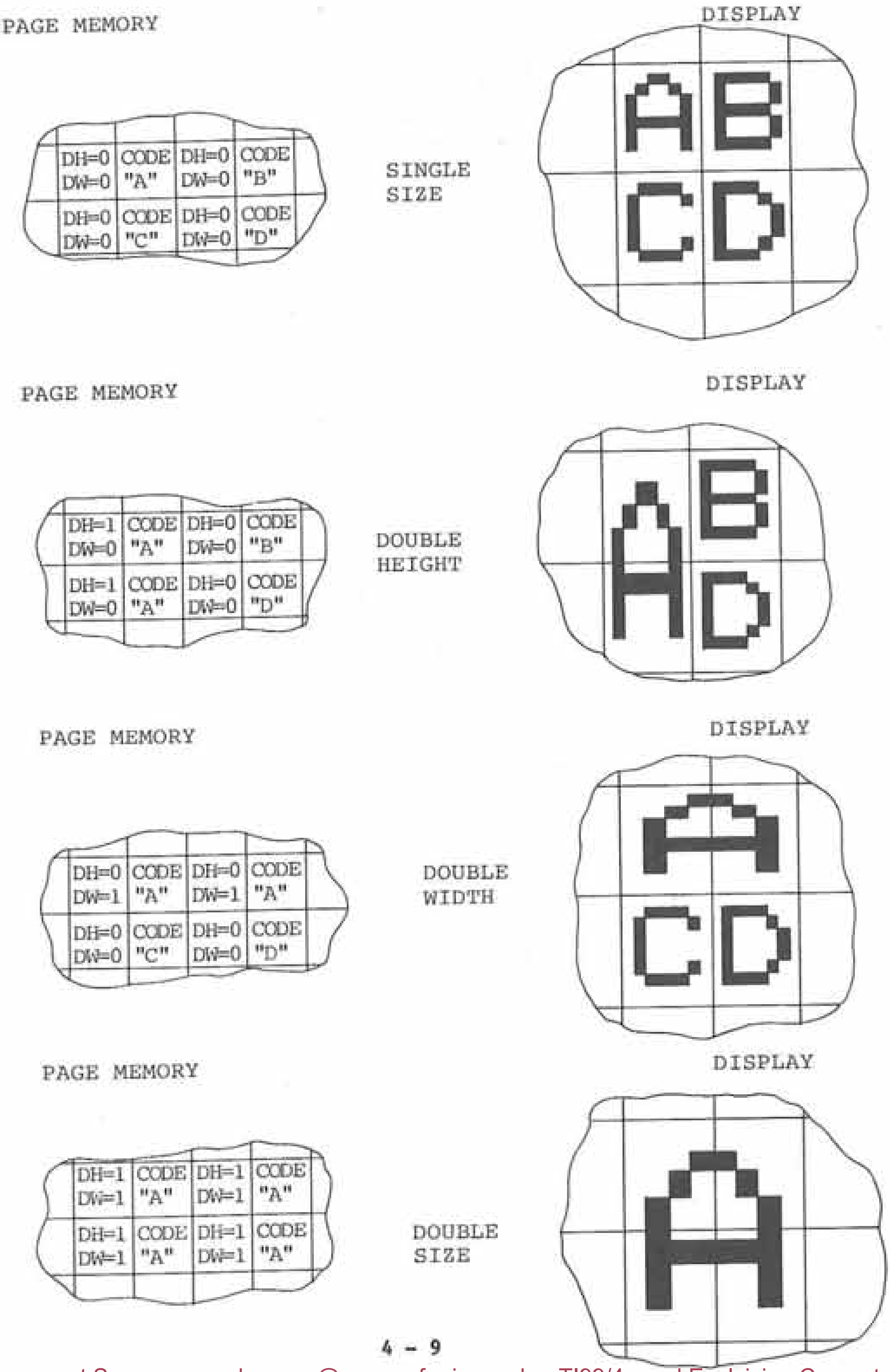
If the FLS bit is set to one, the character will be flashed at a rate of approximately twice per second. This is effected by alternately displaying the foreground parts of the character in the background colour for half of the flashing period so that the character appears to flash on and off repetitively.

4.2.3 Character generators

As explained in paragraph 4.2.1, the VDP can display characters from up to four different character generators on any displayed page, each generator containing up to 128 different characters. The character generators themselves contain, for each character definition a pattern of 1's and 0's indicating to the VDP the foreground and background portions of each character respectively as shown in figure 4.9. The structure of each character generator is as follows:

The first byte of the generator (at the address indicated by the contents of the character generator base address BAGCn, where n=0,1,2 or 3) contains the definition of

IGURE 4.5 : APPLICATION OF DOUBLE HEIGHT AND DOUBLE WIDTH ATTRIBUTES



PAGE MEMORY

DH=0
DW=0
	CODE
	"A"
		DH=0
		DW=0
			CODE
			"B"

DH=0
DW=0
	CODE
	"C"
		DH=0
		DW=0
			CODE
			"D"

SINGLE SIZE

DISPLAY

PAGE MEMORY

DH=1
DW=0
	CODE
	"A"
		DH=0
		DW=0
			CODE
			"B"

DH=1
DW=0
	CODE
	"A"
		DH=0
		DW=0
			CODE
			"D"

DOUBLE HEIGHT

DISPLAY

PAGE MEMORY

DH=0
DW=1
	CODE
	"A"
		DH=0
		DW=1
			CODE
			"A"

DH=0
DW=0
	CODE
	"C"
		DH=0
		DW=0
			CODE
			"D"

DOUBLE WIDTH

DISPLAY

PAGE MEMORY

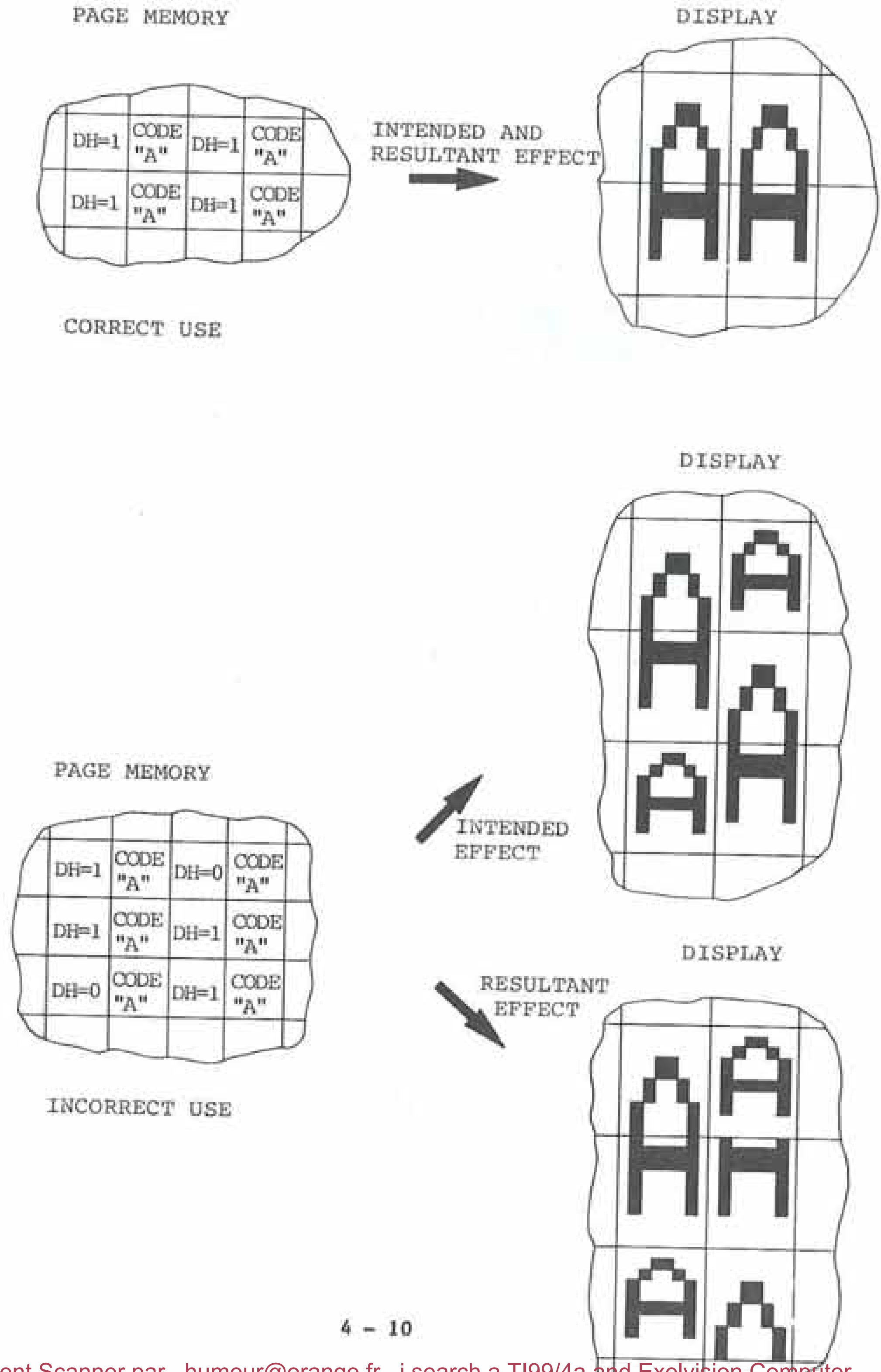
DH=1
DW=1
	CODE
	"A"
		DH=1
		DW=1
			CODE
			"A"

DH=1
DW=1
	CODE
	"A"
		DH=1
		DW=1
			CODE
			"A"

DOUBLE SIZE

DISPLAY

FIGURE 4.6 : LIMITATION ON USE OF DOUBLE HEIGHT ATTRIBUTE



DH=1	CODE "A"	DH=1	CODE "A"
DH=1	CODE "A"	DH=1	CODE "A"

DH=1	CODE "A"	DH=0	CODE "A"
DH=1	CODE "A"	DH=1	CODE "A"
DH=0	CODE "A"	DH=1	CODE "A"

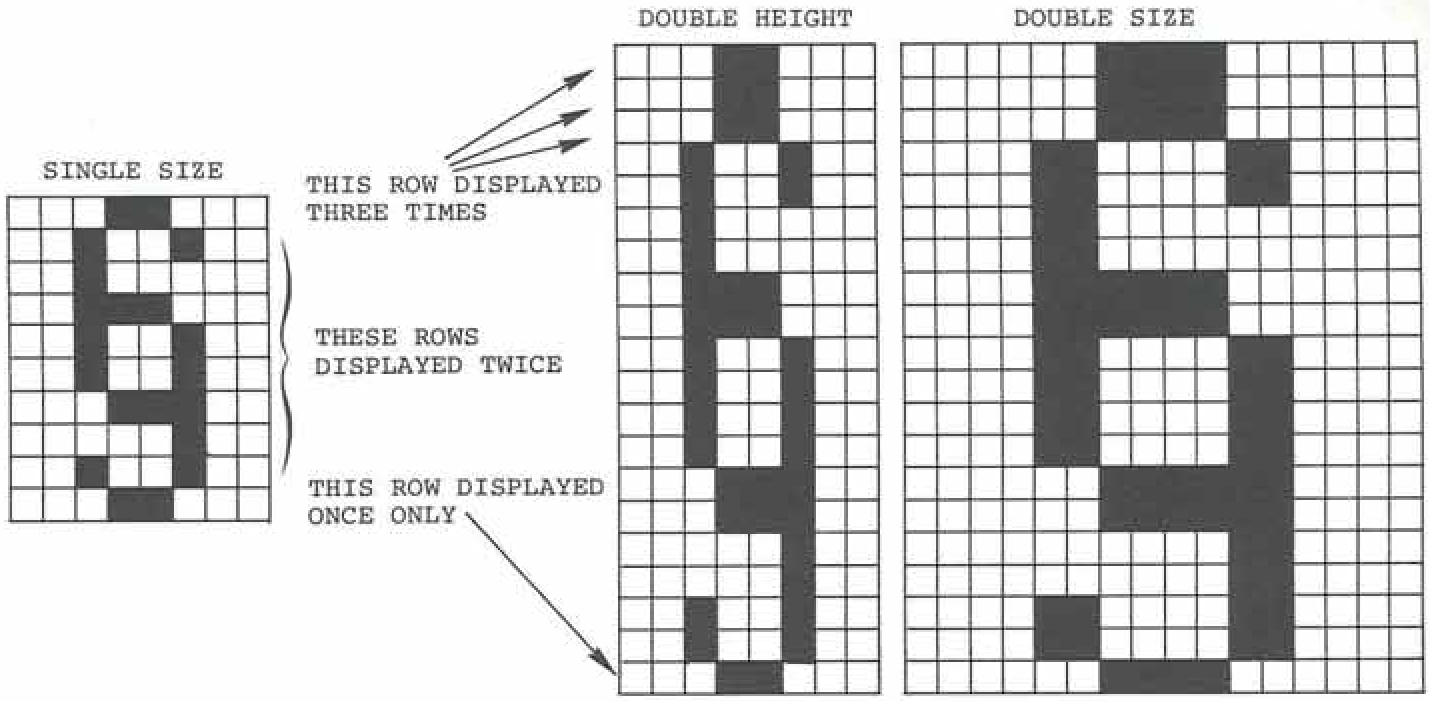

SINGLE SIZE
THIS ROW DISPLAYED THREE TIMES
THESE ROWS DISPLAYED TWICE
THIS ROW DISPLAYED ONCE ONLY
DOUBLE HEIGHT
DOUBLE SIZE

FIGURE 4.7 : DISPLAY OF DOUBLE HEIGHT ATTRIBUTE

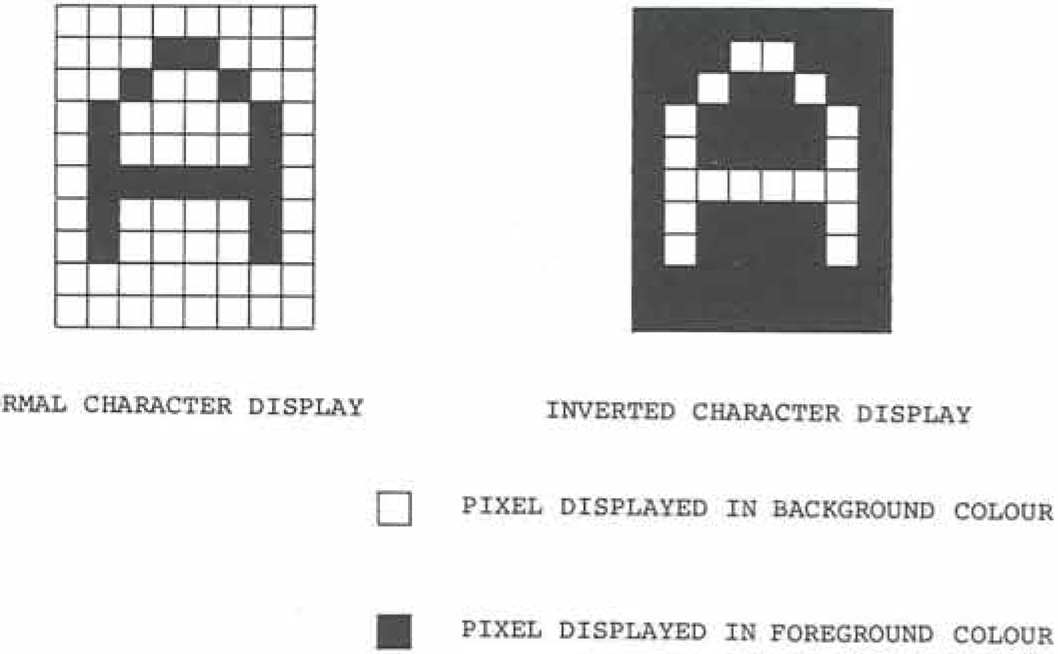
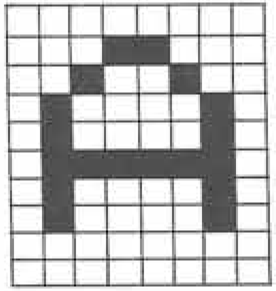

NORMAL CHARACTER DISPLAY
INVERTED CHARACTER DISPLAY
□ PIXEL DISPLAYED IN BACKGROUND COLOUR
■ PIXEL DISPLAYED IN FOREGROUND COLOUR

FIGURE 4.8 : EFFECT OF INVERSION ATTRIBUTE

DESIRED CHARACTER SHAPE



CHARACTER GENERATOR CONTENTS

ROW Ø	0	0	0	0	0	0	0	0	> 00
ROW 1	0	0	0	1	1	0	0	0	> 18
ROW 2	0	0	1	0	0	1	0	0	> 24
ROW 3	0	1	0	0	0	0	1	0	> 42
ROW 4	0	1	0	0	0	0	1	0	> 42
ROW 5	0	1	1	1	1	1	1	0	> 7E
ROW 6	0	1	0	0	0	0	1	0	> 42
ROW 7	0	1	0	0	0	0	1	0	> 42
ROW 8	0	0	0	0	0	0	0	0	> 00
ROW 9	0	0	0	0	0	0	0	0	> 00

HEX VALUE
OF EACH BYTE

■ FOREGROUND OF CHARACTER

□ BACKGROUND OF CHARACTER

FIGURE 4.9 : DEFINING CHARACTER SHAPE IN GENERATOR

the last line of the first character, i.e. the character refered to with a code of 0. The following byte in the memory contains the definition of the last line of the second character in the generator, i.e. that refered to with a character code of 1, and so on, up to the 128th byte of the table (at the address BAGCn + 127) which contains the definition of the last line of the last (128th) character. The structure then repeats with the next byte in the generator containing the definition of the next to last line of the first character and so on, up to the 256th byte (at address BAGCn + 255) which contains the definition of the next to last line of the last character. This pattern continues until all the lines of all the characters have been defined. This structure is shown in figure 4.10. A convenient formula for finding the address of the Nth line of the Mth character is:

Address = BAGCn + ((10 - N) x 128) + (M - 1)

with: 1 N 10 and 1 M 128

4.2.4 Serial attributes

As we have seen, the first 9 bits of the page memory contents defining one character contain the description of the character foreground colour, from which character generator its shape is to be taken, its size, whether it is to be inverted and whether it is to be flashed. These features are termed parallel attributes, and since each character is associated with its own set of these attributes, they can be defined on a character by character basis and only affect the display of the character cell to which they refer. Other possibilities exist for altering the visual representation of the characters which can be applied to a group of characters on the screen. These are termed serial or zone attributes. They include: background colour, masking, incrustation and underlining.

Serial attributes are controled by writing a special delimiter character at each end of the zone to be affected as shown in figure 4.11.

FIGURE 4.11: APPLICATION OF SERIAL ATTRIBUTES WITH TWO DELIMITERS

```
.-------------------------------------------------------------.
| Delimiter |   Series of normal characters | Delimiter |
`-------------------------------------------------------------'
     /\       \_______________  _______________/       /\
 Start of                     \/                    End of first
 serial zone     Zone to which the serial           zone and start
                 attributes will be applied         of next
```

ADDRESS	BAGCn	BAGCn+1	BAGCn+2		BAGCn+124	BAGCn+125	BAGCn+126	BAGCn+127	
BAGCn	ROW 9 OF CHR Ø	ROW 9 OF CHR 1	ROW 9 OF CHR 2		ROW 9 OF CHR 124	ROW 9 OF CHR 125	ROW 9 OF CHR 126	ROW 9 OF CHR 127	ROW 9
BAGCn+128	" 8 " Ø	" 8 " 1	" 8 " 2		" 8 " 124	" 8 " 125	" 8 " 126	" 8 " 127	ROW 8
BAGCn+256	" 7 " Ø	" 7 " 1	" 7 " 2		" 7 " 124	" 7 " 125	" 7 " 126	" 7 " 127	ROW 7
BAGCn+384	" 6 " Ø	" 6 " 1	" 6 " 2		" 6 " 124	" 6 " 125	" 6 " 126	" 6 " 127	ROW 6
BAGCn+512	" 5 " Ø	" 5 " 1	" 5 " 2		" 5 " 124	" 5 " 125	" 5 " 126	" 5 " 127	ROW 5
BAGCn+640	" 4 " Ø	" 4 " 1	" 4 " 2		" 4 " 124	" 4 " 125	" 4 " 126	" 4 " 127	ROW 4
BAGCn+768	" 3 " Ø	" 3 " 1	" 3 " 2		" 3 " 124	" 3 " 125	" 3 " 126	" 3 " 127	ROW 3
BAGCn+896	" 2 " Ø	" 2 " 1	" 2 " 2		" 2 " 124	" 2 " 125	" 2 " 126	" 2 " 127	ROW 2
BAGCn+1024	" 1 " Ø	" 1 " 1	" 1 " 2		" 1 " 124	" 1 " 125	" 1 " 126	" 1 " 127	ROW 1
BAGCn+1152	" Ø " Ø	" Ø " 1	" Ø " 2		" Ø " 124	" Ø " 125	" Ø " 126	" Ø " 127	ROW Ø

8 bytes defining
character Ø

8 bytes defining
character 127

FIGURE 4.10 : STRUCTURE OF CHARACTER GENERATORS

The two delimiters at each end of the zone are in fact identical, the trailing one indicates the end of the zone implicitly simply by defining the start of another zone.

One character in each generator is reserved to indicate the zone delimiter. This is the 32nd character which has the character code >20 in hex. This value has been chosen because it is the ASCII code for the space character, and since the latter, by definition has no foreground component, none of the normal parallel attributes have any effect on its visual representation (for example, a space cannot be flashed).

The delimiter character when found by the VDP is displayed as a space, that is a character cell with a uniform colour. The delimiter itself is stored in the page memory with the same two byte structure as normal characters, the difference being that the VDP interprets the first 9 attribute bits differently, and specifically to indicate the attributes of the following zone, up to the next serial delimiter. The two bytes stored in the page memory are shown in figure 4.12.

Figure 4.12: PAGE MEMORY CONTENTS FOR A DELIMITER CHARACTER

```
                     Even byte                                Odd byte
   /-------------------/\---------------------\           /------/\------\
 .--------------------------------------------------.   .------------------.
 | BF| GF| RF| MSK | INC | BB | GB | RB |             | UNL | hex >20  |
 '--------------------------------------------------'   '------------------'
```

Table 4.3 describes the significance of each bit.

Table 4.3 : Bit significance of delimiter attributes

Bit	Description
BF GF RF	Colour of the delimiter character.
MSK	Masking
INC	Incrustation
BB GB RB	Background colour of the zone
UNL	Underlining

The three bits BF, GF, and RF define the colour of the whole of the delimiter character cell irrespective of the contents of the character generators at character address >20. If the masking attribute was active in the zone preceeding the delimiter, and masking is enabled, however, the colour of the delimiter will be the same as the background colour of the preceeding zone.

The three bits BB, GB and RB define the background colour for the following characters up to the next delimiter or the end of the row whichever occurs first. The background colour is the colour of those pixels defined at 0 in the character generator. Note that this will be the case irrespective of whether the preceeding zone was masked or not.

The MSK bit controls the masking attribute. For this bit to have an effect, the masking enable bit DC3 in command register CM2 must be set. If DC3 is reset, the state of MSK will be ignored. The MSK bit, when set with DC3 set, causes all following character cells, up to the next delimiter or the end of the row, whichever comes first, to be displayed as spaces. The colour of the following spaces will be that defined by BB, GB and RB. Figure 4.13 shows the effect of the masking attribute.

The INC bit controls the incrustation attribute. For this bit to have an effect, the incrustation enable bit DC4 in command register CM2 must be set. If DC4 is reset, the state of INC will be ignored. Additionally, if the VDP is programmed in internal sync mode, i.e. the sync mode control bit DC1 in command register CM2 is set, the Incrustation output signal I will be forced to a continuous high level irrespective of the incrustation attribute. The INC bit then, when set to one, with DC4 set and DC1 reset, causes the I output signal to go high at the start of the delimiter and remain high until the next delimiter with INC set to zero or the end of the row whichever occurs first. If additionally the superimposition control bit DC8 is also set, the I output signal will only go active high during the foreground parts of the characters and not during the background parts or during any of the delimiters included in the incrusted zone. This is shown in Figure 4.14.

The R, G and B output signals are forced to a low level whenever the I output signal is low.

The UNL bit controls the underlining attribute. For this bit to have an effect, the underlining enable bit DC7 in command register CM2 must be set. If DC7 is reset, the state of UNL will be ignored. The effect of the underlining attribute depends on whether the character affected is alphanumeric or alphamosaic. As far as the VDP is concerned, whether the character is one or other of the two sorts depends entirely upon which character generator it comes from. Character generators 0 and 1, i.e. those indicated by base addresses BAGC0 and BAGC1

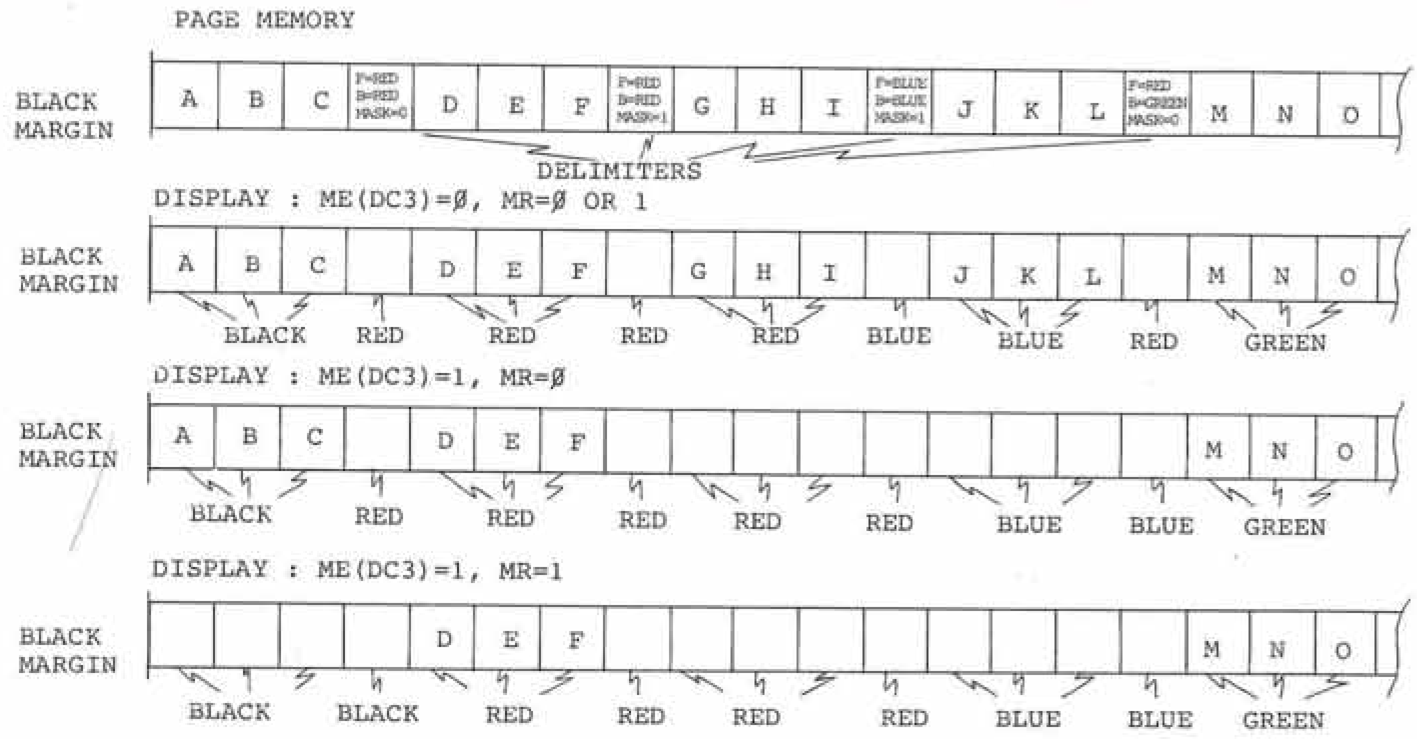
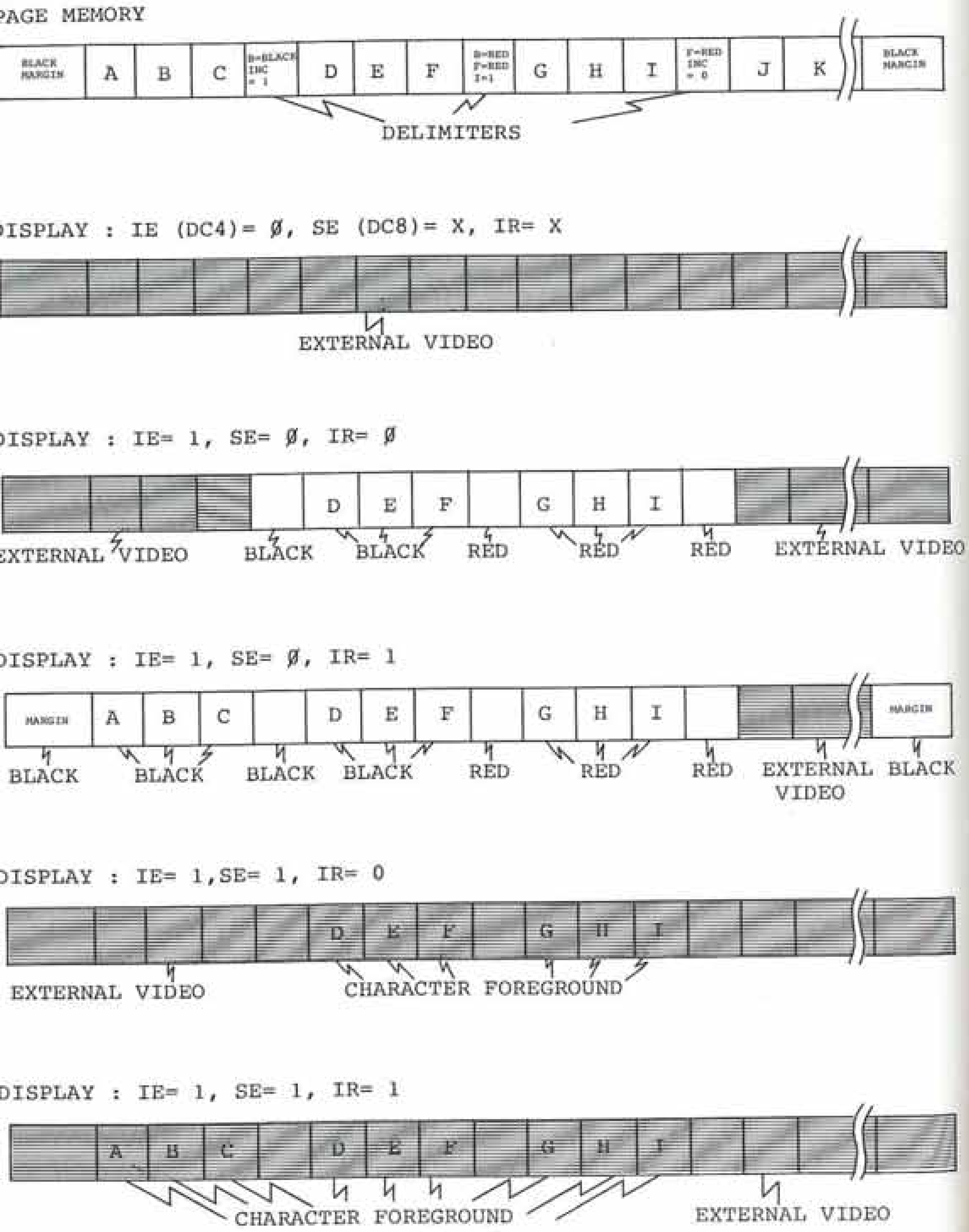

PAGE MEMORY
BLACK MARGIN
A B C
F=RED B=RED MASK=0
D E F
F=RED B=RED MASK=1
G H I
F=BLUE B=BLUE MASK=1
J K L
F=RED B=GREEN MASK=0
M N O
DELIMITERS
DISPLAY : ME(DC3)=Ø, MR=Ø OR 1
BLACK MARGIN
A B C D E F G H I J K L M N O
BLACK RED RED RED RED BLUE BLUE RED GREEN
DISPLAY : ME(DC3)=1, MR=Ø
BLACK MARGIN
A B C D E F M N O
BLACK RED RED RED RED RED BLUE BLUE GREEN
DISPLAY : ME(DC3)=1, MR=1
BLACK MARGIN
D E F M N O
BLACK BLACK RED RED RED RED BLUE BLUE GREEN

FIGURE 4.13 : EFFECT OF THE MASKING ATTRIBUTE

FIGURE 4.14 : EFFECT OF INCRUSTATION AND SUPERPOSITION



respectively, are always considered to contain alpha-numeric characters. Character generator 2, indicated by the contents of base address BAGC2, is always considered to contain alpha-mosaic characters. The fourth character generator, indicated by BAGC3, can be defined as either containing alpha-numeric or alpha-mosaic characters by programming bit DC5 in command register CM2 to zero or one respectively.

For alpha-numeric characters, the effect of the underlining attribute is to display the whole of the last line of the character in the foreground colour defined for that character irespective of the bit pattern contained in the character generator. For alpha-mosaic characters, the effect of underlining is to display the third, seventh and tenth rows, together with the first and fifth columns in the background colour irespective of the contents of the character generator. These two effects are shown pictorially in figure 4.15. The delimiters themselves are not underlined even if they are preceeded by a zone in which this attribute is active.

4.2.5 Alpha-mosaic attributes

The nine attribute bits associated with characters taken from an alpha-mosaic character generator are interpreted in a slightly different fashion from those of a normal alpha-numeric character generator.

Figure 4.16 shows the contents of the two bytes defining the alpha-mosaic character cell.

Figure 4.16: PAGE MEMORY CONTENTS DESCRIBING A MOSAIC CHARACTER

```
                         Even byte                            Odd byte
           ____________________/\____________________      ________/\________
          /                                          \    /                  \
 .-------------------------------------------------.  .------------------.
 | BF | GF | RF | CG1 | CG0 | BB | GB | RB |  | FLS | CHR CODE |
 '-------------------------------------------------'  '------------------'
```

Table 4.4 describes the significance of each bit.

FIGURE 4.15 : EFFECT OF UNDERLINING ATTRIBUTE

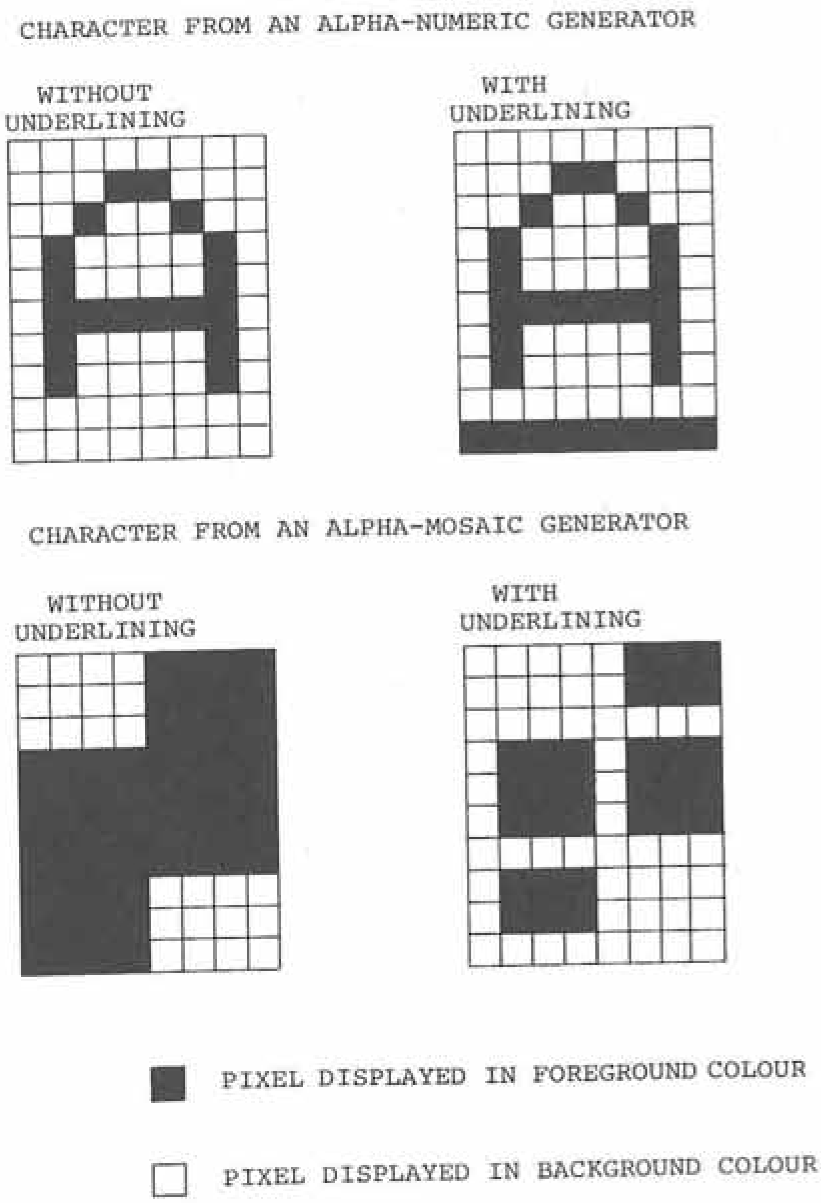

CHARACTER FROM AN ALPHA-NUMERIC GENERATOR
WITHOUT UNDERLINING
WITH UNDERLINING
CHARACTER FROM AN ALPHA-MOSAIC GENERATOR
WITHOUT UNDERLINING
WITH UNDERLINING
■ PIXEL DISPLAYED IN FOREGROUND COLOUR
□ PIXEL DISPLAYED IN BACKGROUND COLOUR

Table 4.4 : BIT SIGNIFICANCE OF ALPHA-MOSAIC ATTRIBUTES

Bit	Description
BF GF RF	Foreground character colour.
CG1 CG0	Character generator defined to contain alpha-mosaic characters.
BB GB RB	Background character colour
FLS	Flashing
CHR CODE	Character number in the defined generator (7 bits)

As for normal characters the three bits BF, GF and RF define the foreground colour of the character.

The character generator select bits CG1 and CG0 must indicate a generator containing alpha-mosaic characters as described above.

The three bits BB, GB and RB define the background colour of the character itself and also the background colour of the following alpha-numeric characters. This means that as far as background colour is concerned, an alpha-mosaic character acts just like a zone delimiter.

The FLS bit has an identical effect in the case of an alpha-mosaic character as in the case of an alpha-numeric one.

Note also that the underlining attribute is also interpreted differently for alpha-mosaic characters as described in paragraph 4.2.4.

4.2.6 Command register CM4 and full page attributes

All of the serial attributes described in the preceeding paragraphs are active for zones of characters on the screen. The end of the zones are indicated either by another delimiter, which has the effect of defining the start of a new zone, or, the end of the row whichever occurs first. The end of each row, and consequentially, the begining of the next row acts just like a delimiter with the state of the serial attributes at the start of each row defined by the contents of command register CM4.

Command register CM4 is an eight bit on chip register accesses in exactly the same way as the other command registers. Its contents are shown in Figure 4.17 and the significance of each individual bit is described in Table 4.5.

FIGURE 4.5: COMMAND REGISTER CM4 CONTENTS

BM	GM	RM	X	MR	LR	IR	X
MSB							LSB

TABLE 4.5: DISPLAY ATTRIBUTE REGISTER (CM4)

NAME	FUNCTION
BM	\ Margin colour and
GM	> background colour at
RM	/ the start of each row
X	Not used - don't care
MR	Masking
LR	Underlining
IR	Incrustation
X	Not used - don't care

The three bits BM, GM and RM define the colour of the margin around the whole screen and also the background colour at the start of each row.

The three bits MR, LR and IR define the state of the masking, underlining and incrustation attributes, respectively at the start of each row. Note that for these attributes to have any effect, the relevant enable bits, DC3, DC7 and DC4 in command register CM2 must be set just as for normal delimiters.

Additionally, if IR is set, DC4 is set and the superimposition control bit DC8 is reset, the I output signal will also be high during the display of the margin, including the upper and lower borders, irrespective of any serial delimiters in the active part of the page.

Figures 4.13, 4.14, and 4.15 show the effect of MR, IR and LR.

4.3 Bit-mapped mode

The VDP is placed in bit-mapped mode by programming the mode control bits CT1 and CT2 in command register CM3 to one and zero respecively.

In bit-mapped, or graphics, mode the colour of each individual pixel of the displayed image is independantly definable. The latter is made up of 320 pixels horizontally by 250 vertically in 625 line mode, or 256 pixels by 210 in 525 line mode, each one of which can be displayed in one of eight colours. There are no character generators associated with the bit-mapped mode and so their base addresses BAGC0 to BAGC3 are not used. It is still possible to define the margin colour and control the state of the I output signal, however, as is described below.

4.3.1 Page memory

In the same way as for text mode, BAPA points to an area of memory containing the information necessary for the VDP to generate the displayed image. Since each pixel can be defined to take one of eight colours, three bits are required for this purpose. The red, green and blue bits for each group of eight pixels are contained in three consecutive bytes in the page memory as shown in figure 4.18.

FIGURE 4.18: PAGE MEMORY STRUCTURE FOR 8 PIXELS IN BIT-MAPPED MODE

```
 .-----------------.  .-----------------.  .-----------------.
 | B B B B B B B B |  | G G G G G G G G |  | R R R R R R R R |
 '-----------------'  '-----------------'  '-----------------'
   |             |      |             |      |             |
 Blue of     Blue of  Green of    Green of  Red of       Red of
 Nth point   N+7th    Nth point   N+7th    Nth point    N+7th
             point                point                 point
```

40 groups of these triplets define the colour of the 320 points on each line making 120 bytes per line as shown in figure 4.19.

4.3.2 Control of the I output and margin colour

Following each group of 120 bytes are two extra bytes the contents of which are shown in figure 4.20.

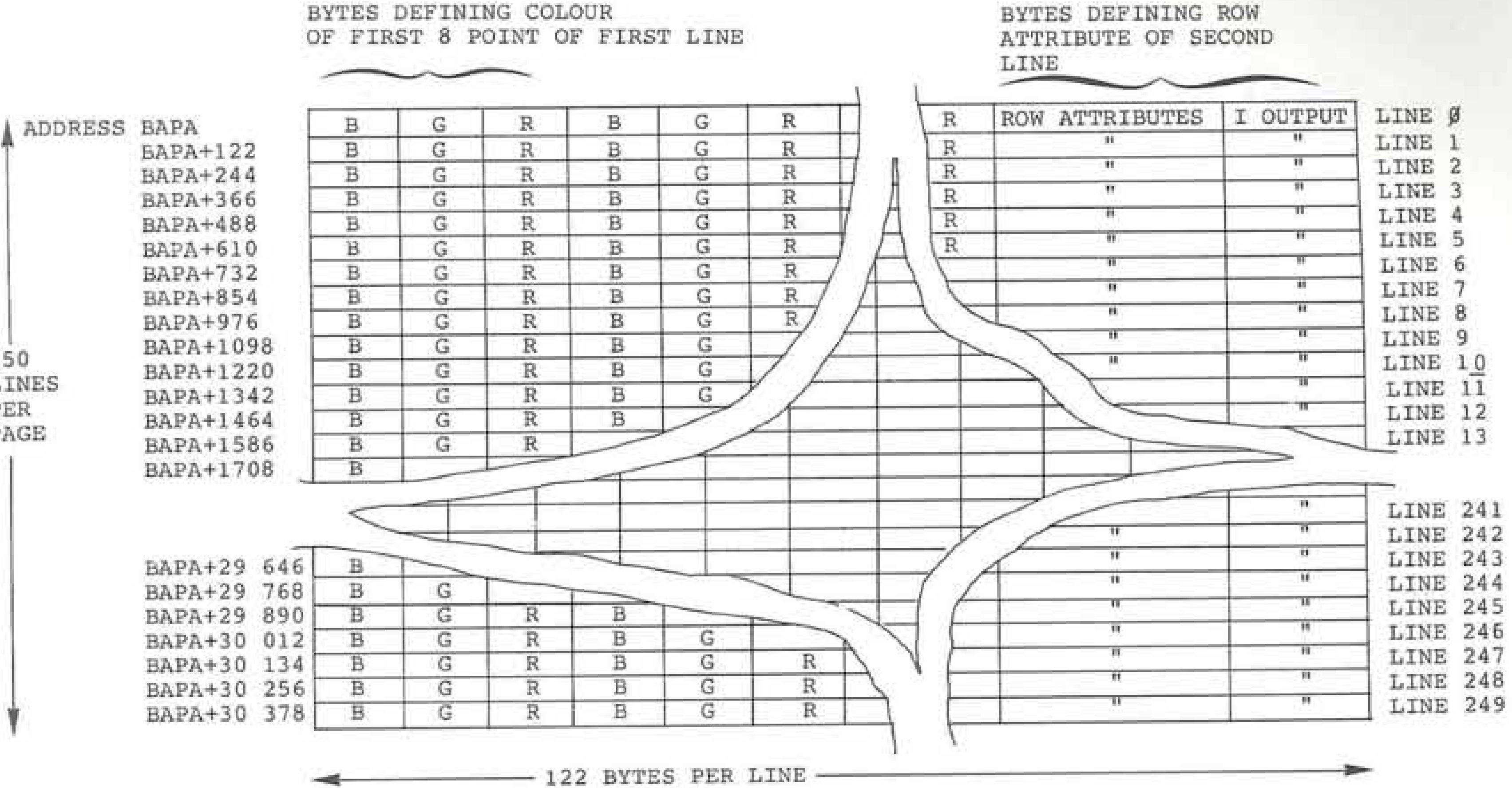


FIGURE 4.19 : PAGE MEMORY STRUCTURE IN BIT MAPPED MODE

FIGURE 4.20: Contents of 121st and 122nd byte of each line

```
                121st byte                         122nd byte
.---------------------------------------.  .------------------.
| BM | GM | RM | X | MR | LR | IR | X |  | I output control |
'---------------------------------------'  '------------------'
```

The contents of the 121st byte are copied in to the CM4 command register at the end of each active line. The three bits BM, GM and RM define the colour of the left and right parts of the margin of the following line. The contents of the 121st byte of the last line of the display define the colour of the upper and lower margin areas of the display together with the left and right parts of the margin of the first active line of the display.

The MR bit, when set, causes the whole of the following line to be displayed in the border colour as defined by the BM, GM and RM bits. Note that the border itself will not be displayed unless DC1=1 or DC1=0 with IR=1.

The LR bit is not intented for use in bit-mapped mode and so should always be programmed to zero to avoid unpredictable results.

In internal sync mode (DC1=1), the IR bit has no effect on the display. In external sync mode (DC1=0) the IR bit, when set if DC4 is set, causes the I output signal to go high for the whole of the next line, including the border, irrespective of the contents of the 122nd byte.

The contents of the 122nd byte on each line controls the state of the I output signal for the following line displayed. Their action depends on the state of the synchronisation mode control bit DC1 in command register CM2. Each line displayed is divided up into 8 groups of 40 points and each bit in the 122nd byte of every line is associated with one of these groups; i.e. the Nth bit in the 122nd byte controls the state of the I output for the Nth group of 40 points of the following line. The 122nd byte of the last line of the display controls the I output for the first line of the display.

If DC1 is zero, the I output signal will be low during the whole of the margin area and will only change to a high level during the group of 40 points which have a one programmed in the corresponding bit position of the 122nd byte of the preceeding line. Additionally, the RGB outputs will be forced to a low level whenever the I output is low. If DC1 is at a one state, the I output signal will be high during the whole of the margin area, and will only go low during a group of 40 points which have the corresponding bit position in the 122nd byte of the preceeding line programmed to one. In this case, the RGB outputs are not blanked when I is low.

Since there are 122 bytes needed to define one line of the bit-mapped display, and there are 250 active lines on the screen, the definition of one page occupies 30500 bytes of VDP memory.

4.4 Mixed mode

The VDP is placed in mixed mode by programming both of the mode control bits CT1 and CT2 in command register CM3 to one.

Mixed mode, as its name implies allows both text and bit-mapped display on the same screen. These two modes may be mixed in any way the user choses on a line by line basis with the sole limitation that the display must start with a row of text. Lines programmed in bit mapped mode may be grouped individually, a row of text will, of course, take up 10 lines of the displayed image.

4.4.1 Page memory

Each line of graphics or row of text is followed in the page memory by two extra bytes. These would be the 81st and 82nd bytes of a text row, or the 121st and 122nd bytes of a bit-mapped line. Their contents is shown in figure 4.21.

FIGURE 4.21: CONTENTS OF THE BYTES FOLLOWING EACH LINE OR ROW

81st or 121st byte								82nd or 122nd byte
BM	GM	RM	C/G	MR	LR	IR	X	I output control

The contents of the 81st or 121st byte are transfered into command register CM4 at the end of each active line or row in the same manner as in pure bit-mapped mode. The three bits BM, GM and RM define the colour of the left and right portions of the margin of the following line or row. These three bits in the 81st or 121st byte of the last row or line on the screen define the colour of the upper and lower areas of the margin together with the colour of the left and right portions of the margin of the first row. They also define the background colour at the start of the first row.

The C/G bit defines the mode of the next part of the display. If programmed to one, a line of graphics will follow, and if programmed to zero a row of text will follow. If a row of text is requested, the contents of the 82nd or 122nd byte will be ignored. The state of the C/G bit will be ignored for the 81st or 121st byte of the last line or row of the display since the latter always starts with a row of text.

The three bits MR, LR and IR control the state of the masking, underlining and incrustation attributes at the start of the next row, if the latter is in text mode, in exactly the same way as the same bits of command register CM4 are used in pure text mode. If a bit-mapped line is programmed, these three bits have the same effect as the corresponding bits in the 121st byte in pure bit-mapped mode.

The contents of the 82nd or 122nd byte of each line controls the state of the I output signal during the next line, if the latter is in bit-mapped mode, in exactly the same way as the contents of the 122nd byte of every line in pure bit-mapped mode.

For rows defined in text mode, the other attributes, including the text mode control bits contained in command register CM2 retain the same significance as in pure text mode.

The size of the page memory in mixed mode depends on the mix of text and graphics and can be calculated from the following formula:

Total memory size = (T x 82) + (G x 122)

where T and G are the number of text rows
and lines of graphics respectively.

4.5 Subtitling mode

Subtitling mode is a mode with a reduced area of active display, the upper margin being extended to cover the rest of the screen.

This area is equivalent to the last three rows of text on the screen. The page memory for this mode then occupies only 240 bytes of VDP memory. All of the other parameters and functions of text mode are retained. Note that the contents of the page memory base address BAPA will still point to the start of the information to be displayed even though this occurs on o lower part of the screen.

5 OPERATIONAL SOFTWARE CONSIDERATIONS

5.1 Initialisation

5.1.1 Register pointer

As explained in paragraph 3.3, all operational mode controls and memory organisation are controled by on chip registers the contents of which are defined by the CPU. To write to any of these registers the CPU makes use of the register pointer which on power up will contain an indeterminate value and so must first be cleared. The only sure way of effecting this operation is to perform an access to external VDP memory i.e. either of the accesses with the enable signal E2 low. The internal hardware of the VDP is constructed in such a way that the register pointer is always cleared following a memory access. The data read from, or written to, the memory will doubtless be irrelavent for this first access.

5.1.2 Control registers

Following the clearing of the register pointer, the control registers themselves, and the base address registers can be initialised. There is no intrinsic limitation to the order in which the registers are programmed, or even an absolute need to programme all of them, however, certain considerations should be observed:

Programming control register CM1 has the side-effect of halting the operation of the VDP timebase. This means that the display decoder ceases to make accesses to the page memory and, in text mode to the character generators, and the R, G and B outputs are held low resulting in a blank display.

Additionally, refresh of the DRAM's will also stop since this is synchronised to the display. During the first initialisation phase following power up, this presents no problem since the memory will not contain any data needing retention. However, if CM1 is programmed during normal VDP operation then the time base should be re-started as soon as possible afterwards to ensure correct data retention in the dynamic memories.

The timebase is restarted when any of the base address registers, i.e. registers 8 to 15, are programmed. It should be noted that the timebase will not simply carry on from the state in which it was halted, but with the various frame, line and character counters, and the auto-incrementing page display accumulator, ACPA, all zeroed. Thus the display restarts with the first line of the upper border.

A logical and suitable sequence of initialising the on-chip registers is therefore simply in ascending numerical order, commencing by command register CM1 followed by the other command registers, and finally the base addresses.

5.1.3 Character generators

It is advisable to initialise the character generator tables only after the on chip registers have been programmed. This is because on power up the timebase may not be running correctly and therefore correct DRAM refresh and hence data retention is not assured. Also, the sequence of initialising the command registers, as described above, results in a momentary timebase halt causing a break in the normal continuous refresh action, which, although probably too short to cause problems, is best avoided when the DRAM's contain useful data.

The character generator tables themselves are only accessed by the display decoder due to an explicit reference to their contents contained in the page display memory. Thus, if certain character codes are never used in the construction of the page display memory, it is unnecessary to define the contents of the character generator table for those characters (it may nevertheless be convenient to do so). If no characters at all are referenced from a particular character generator, then it is not necessary to define its base address register contents on chip.

5.1.4 Page display memory

For the same reason as for the character generators, it is advisable only to write to the page display memory following initialisation and start-up of the timebase.

Typically, page display memory initialisation will effectivly entail 'clearing' it, i.e. writing a uniform colour to all pixels in bit-mapped mode, or writing spaces to all character positions in text mode. In the latter case, two alternatives present themselves:

In the first case, the page display memory is filled with delimiter characters (code >20) in all cells. As described in paragraph 4.1.4, the attribute byte associated with each of these delimiters contains the colour of the space that is displayed as a result thereby allowing the page to be cleared to any uniform colour desired. The particularity of this technique of clearing the page display memory is that the colour of each character cell is defined explicitly by its attribute byte and the programmation of neighbouring characters will not affect it. Put otherwise, the page memory has been filled with end of zone delimiters so that any zone in which serial attributes are active which is subsequently written on the screen will automatically terminate at the first untouched character on the row containing the zone.

In the second case, the page display memory is filled with a character not having the code >20 and therefore not demonstrating the properties of a serial zone delimiter (the code >00 is typically used for this). The character generator definition for this character should be initialised with all zeroes so that the display appears in uniform background colour. Since there are no serial delimiters in the page display memory, the background colour displayed will be the default background colour at the start of each row as defined in command register CM4. The border area around the active part of the display will also be in the same colour as a consequence. The difference now is that the page display memory no longer contains zone delimiters so that if the former is overwritten at some location by a start of zone delimiter (code >20) with consequential zone attributes, the zone thus defined, and hence the consequential visual effect, background colour etcetera, will propagate implicitly all the way to the end of the modified row.

5.1.5 Data input buffer

As described in paragraph 3.6 the data input buffer is initialised by programming the base address register defining the start of the allocated zone, BAMT. This will happen automatically when the base address registers are initilaised following power up, although it may be desirable to re-initialise the buffer during normal operation (following a buffer overflow condition for example).

The effect of initialising the data buffer is to reset the data provider write pointer ACMT to be equal to BAMT, i.e. the start of the buffer, and to set and reset respectively, the buffer empty and buffer overflow flags ST7 and ST8 in the status register. If ST8 was set prior to initialisation indicating that the buffer had overflowed and that data provider accesses were inhibited, programming BAMT will remove this condition and data provider accesses will once more be allowed.

Because ACMT is reset equal to BAMT and the buffer empty flag in the status register is unconditionally set when the buffer is initialised it is thereafter not possible to determine how much data had been written into the buffer by the data provider but not read out of it by the CPU. Thus, care must be taken when initialising the buffer that the latter does not contain required data unless the CPU has accurate knowledge of how many bytes are left in it (i.e. where ACMT was at the moment of initialisation). It should also be noted that following a buffer reset, although ST7 indicates a buffer empty condition, the two pointers ACMT and ACMP will not necessarily be equal since the latter is only programmable explicitly by the CPU. Furthermore, it should be noted that under no circumstances is the CPU inhibited from reading the contents of the data buffer; this means that if the read and write pointers are equal with the buffer empty flag ST7 set indicating that there are no new bytes to be read by the CPU, the latter can still perform a read access causing the two pointers to become unequal and therby reseting the buffer empty flag. This could cause CPU misoperation if, for example, an independant part of the sofware checks ST7, since the latter now apperently indicates that there is useful data in the buffer even though no data provider accesses may have taken place since the buffer became empty. Also, if a data provider access does take place in this situation ACMT will be incremented following the access and become equal to ACMP. This is the buffer overflow condition and ST8 will be set to indicate it and further data provider accesses inhibited even though in fact there is only one unread byte in the buffer.

5.2 Memory address pointer manipulation

Paragraph 3.5.1 describes the operation of the on chip auto-incrementing address pointers associated with the CPU. The following paragraphs describe their use in more practical terms.

5.2.1 Safe pointer manipulation

As described in paragraph 3.5.1 the VDP contains two pointers associated with CPU accesses: ACMPxy and ACMP. The former is always a general purpose pointer for CPU reads and writes anywhere in the available VDP memory, the second is normally reserved for CPU reads from the data provider input buffer, but can be used like ACMPxy in systems not containing a data provider.

In systems without a data provider, and hence an input buffer, the simplest pointer manipulation technique from a software point of view, and hence less exposed to software errors, is to use the primary pointer ACMPxy for all accesses to the VDP memory. This is accomplished by always re-programming registers COL and/or ROW prior to any access which is either to a memory location other than the one immediately following the last location accessed, or is in the oposite mode to the last access performed, i.e. a read following a write or vice versa.

The advantage of this technique is that no confusion arises as to which mode (read or write) is attributed to which pointer, and therefore which pointer will be used for the access, ACMPxy being used for them all.

The disadvantage is a small software overhead since the application programme must repeatedly reprogramme COL and ROW for all accesses which are not both contiguous and in the same mode. If software execution time is not critical, then this method is recommended as a permanent technique and certainly for initial users of the VDP who are not fully familiar with CPU pointer manipulation.

5.2.2 Pointer manipulation with the data buffer

In sytems implementing a data provider the pointer ACMP is reserved for CPU reads from the buffer. It must not be used for any other access since this will destroy its contents and therefore the address of the next byte to be read out of the buffer will be lost. As explained in paragraph 3.6 ACMP is initialised indirectly via the base address register BAMP and so the latter must be programmed with the address of the start of the buffer contained in the base address BAMT. Provided that ACMP is in the read mode, the first read will copy the contents of BAMP into ACMP and the latter will be used for the access, thus reading the first byte of the buffer. Its value will be incremented following this and all subsequent accesses until its value becomes equal to the base address register BAMTF wherupon it will be automatically reset equal to BAMP. Thus once initialised ACMP can thereafter be relied upon to always point to the next byte to be read.

The fact that ACMP is totally dedicated to buffer use means that ACMPxy must be used for all accesses of a general nature to the rest of VDP memory. This essentially means using the technique described in paragraph 5.2.1 however, an additional precaution is necessary:

As described above, ACMP must always be in the read mode for accesses to the data buffer. However its mode at any particular instant will be the complemernt of the mode of ACMPxy. Performing memory reads with ACMPxy will place the latter in the read mode and consequentially ACMP in the write mode. This change of mode for ACMP does not alter its contents but does mean that it cannot be immediately used for a buffer access - it must first be placed in the read mode. This could be effected directly by programming BAMP and performing a read access as described in paragraph 3.5.1, unfortunatly this action will also modify its contents which is, of course, unacceptable. The only alternative is to change its mode indirectly by placing ACMPxy in the write mode by programming COL or ROW and performing a memory write. This means, in sum, that ACMPxy must spend most of its time in the write mode, being complemented to the read mode only when necessary, and always being returned to the write mode prior to any buffer access.

5.2.3 Dual pointer manipulation without a data buffer

In systems not implementing a data buffer, both CPU pointers are available for general memory data manipulation, both reads and writes. This can lead to far greater software efficiency with carefully strucured use.

As an example, consider the steps necessary to move a block of data from one location in VDP memory to another. Assume that 100 bytes are to be moved from a location starting at address 'ORIG' to a location starting at address 'COPY'.

Using the 'simple' techinique, the steps would be as shown in Figure 5.1, and using the 'fast' technique as shown in Figure 5.2.

FIGURE 5.1: Block copy in VDP memory using 'simple' technique

STEP	ACTION	RESULT
1	Write 'ORIG' to COL/ROW	COL/ROW = 'ORIG'
2	Memory read	Read byte of original
3	Write 'COPY' to COL/ROW	COL/ROW = 'COPY'
4	Memory write	Write byte of copy
5	Increment 'ORIG'	ORIG = ORIG + 1
6	Increment 'COPY'	COPY = COPY + 1
7	Is ORIG = 'ORIG' + 100?	Check for all bytes copied
8	Loop back to 1 if no	
9	Stop if yes	

FIGURE 5.2: Block copy in VDP memory using the 'fast' technique

STEP	ACTION	RESULT
1	Write 'COPY' to COL/ROW	COL/ROW = 'COPY'
2	Transfer COL/ROW to BAMP	BAMP = 'COPY'
3	Write 'ORIG' to COL/ROW	COL/ROW = 'ORIG'
4	Memory read	Read byte using ACMPxy
5	Memory write	Write byte using ACMP
6	Is ORIG = 'ORIG' + 100?	Check for all bytes copied
7	Loop back to 4 if no	
8	Stop if yes	

Examining these two figures shows that the loop size for the 'simple' technique is 7 operations (including the test for all bytes transferred), whereas it reduces to only 3 using the 'fast' technique thereby greatly increasing processor throughput.

5.3 Multipage and scrolling

The address of the first byte in the page display memory is programmed in the base address BAPA as described in section 4. The value of this register can be modified at any time by the application software to indicate any zone in the memory containing displayable information. This gives an intrinsic mutipage storage capability, where any number of distinct pages can be stored in VDP memory, the one to be displayed being indicated by the contents of BAPA.

As far as scrolling is concerned, two techniques present themselves:

The first technique is to simply reprogramme the base address BAPA with its old value plus the number of bytes needed in page memory to display one row of text; this will normaly be 80 in pure text mode. The result of this is that the display then starts with what was the second row of the display and all the other rows are scrolled up by one with the last row displaying data which previously off the bottom of the screen. This last row would typically be cleared before the scroll took place. The advantage of this technique is twofold: firstly, little time is necessary to perform it - the time taken to reprogramme BAPA and possibly erase the bottom row, and secondly the text scrolled off the top of the screen is not lost so that the reverse operation can be performed and the display will revert to its preceeding contents. The disadvantage is that more and more memory is used each time that the display is scrolled resulting in an eventual limit as to how many times the display is scrolled.

The second technique does not alter the value of BAPA but performs a block move of the whole of the page one row upwards in memory so that the first row is overwritten by the second, the second by the third and so on. After the penultimate row is rewritten by the last row, the latter is erased. This gives exactly the same visual effect as the first technique with the advantage that it may be effected as many times as desired since the actual page memory zone never moves. The disadvantages are that unless first saved elsewhere in the memory the old contents of the first row are destroyed and that it takes much longer since the whole of the page memory must be copied.

5.4 Compliance with teletext and videotex standards

Although the VDP is a very flexible device allowing control over nearly all of its display characterstics, certain features of some national teletext and videotex standards are not realisable by the hardware of the VDP but must be overcome by judicious software manipulation. The following paragraphs indicate mode of these small divergencies together with a recommended technique for overcoming them.

5.4.1 Antiope

Antiope is the generic name for the French teletext standard.

Two particularities arrise when decoding this standard with the VDP:

The first concerns the masking attribute. As described in paragraph 4.1.4 a serial delimiter written in page display memory will be displayed in the colour specified in its attribute byte except when preceeded by a zone in which the masking attribute is active i.e. a previous delimiter has requested masking and the global masking enable bit is set. In the latter case the delimiter will be displayed in the same colour as the background of the preceeding masked zone if masking is enabled or in its own defined colour if masking is disabled. Figure 4.13 shows the effect of the masking attribute. This is a nonconformity of the Antiope standard which specifies that a serial delimiter, used for example to define a new background, must always be displayed in that new background colour irrespective of any possible masking attributes.

The solution to the problem is simply to replace all serial delimiters which occur during a masked zone, and which also define a new background colour, with a character from an alpha-mosaic character generator which is defined as all foreground - in Antiope this would be the character >5F. Since alpha-mosaic characters are also serial delimiters as far as background colour is concerned, the substitute character has the desired effect of defining a new background colour, but does not suffer from the colour change when inside a masked zone.

The second particularity occurs only with a potential (at the time of writing) change to the Antiope standard. This is the case of separated graphics treated as a parallel attribute. The VDP treats separated graphics as a serial attribute and normally, a serial delimiter is required. The solution is to create the alpha-mosaic character generator in two halves. The first contains all the normal alpha-mosaic characters and the second, the same characters but defined as they would appear with the separated attribute applied. Then, instead of writing the normal character in the page memory and applying the underlining attribute, its 'underlined' equivalent is instead written in its place and the undelining attribute is not applied.

Since only 64 alpha-mosaic characters are defined in the Antiope standard and the character generators in the VDP contain up to 128 characters, the same character generator can be used to contain both the normal and underlined versions of the characters.

5.4.2 Ceefax

5.4.3 Prestel

5.4.4 Teletel

The discrepencies with the French videotex standard Teletel when implemented with the VDP are identical to those discussed for Antiope in paragraph 5.6.1.

6 OPERATIONAL HARDWARE CONSIDERATIONS

6.1 DMA frequency and timing considerations

As explained in section 3, the DMA controller is responsible for sychronising, prioritising and finally executing all access requests to VDP memory. The densest period of memory activity is during the active part of the displayed line, since, in addition to any possible data provider and CPU accesses, the display controller will be regularly reading the page memory and character generators in order to generate the RGB outputs.

The actual DMA frequency used in any particular system is widely variable and the user may chose a value to suit the particular application; for example, the frequency of a clock already used elsewhere in the system. There are however, some considerations to be taken into account, and these are discussed in the paragraphs that follow.

6.1.1 Maximum DMA frequency

The theoretical maximum DMA frequency is a parameter limited pure and simply by the maximum speed capability of the VDP which is a function of the technology used in its fabrication. As the VDP is fabricated using shrunk scaled NMOS (SMOS), this limit is fairly high and is situated above 18MHz. However, it should not be forgotten that the various memory access cycles are all timed from this master DMA clock and therefore the maximum speed capabilities of the memory devices themselves are more likely to limit the real maximum DMA frequency than the VDP itself. This, of course, is the correct way round since it allows improvements in memory technology to be immediately appreciated in VDP based systems.

As an example calculation of the maximum DMA frequemcy useable in a system, consider the case of a VDP connected to standard TMS4164-15 DRAM's as shown in Figure 7.11. Table 6.1 lists the various TTL propagation times for the interface shown and Table 6.1 gives a list of timing parameters for memory accesses for both the minimum memory requirements and what results from a variety of DMA frequencies. It has been assumed that the standard VDP is used in 625 line mode (BT2 = 0) and normal memory timing mode (CT5 = 1). Some of the timing parameters have different values from those given in section 9 because account has been taken here of the effect and propagation times of the TTL interface shown. The need and effect of the latter is discussed in paragraph 6.4.

TABLE 6.1: Memory timing parameters with varying DMA frequency

TIMING	VDP SPEC		VALUES AT DMA FREQS:			4164-15
PARAMETER	DMA	ELEC	16MHz	17MHz	18MHz	MEMORY
SYMBOL	CYCLES	VAR[ns]	[ns]	[ns]	[ns]	SPEC[ns]
Tc(DMA)	1	0	62.5	58.8	55.6	
Tw(RL)	3					150
Tw(CL)	2					100
Tw(WL)	1					45
Tsu(WCH)	1					60
Tw(RH)	2					100
Tw(CH)	1					50
Tsu(CA)	0.5					-5
Tsu(RA)	1					0
Tsu(D)	0.5					0
Th(CLCA)	0					45
Th(RA)	0.5					20
Th(WLD)	1					45
Tc(R/W)	5					260
Tc(P)	3					160
Ta(R)	3					150
Ta(C)	2					100

6.1.2 Minimum DMA frequency

There is no intrinsic minimum DMA frequency limit due to hardware limitations of the VDP itself. The real minimum frequency comes from the need for the display controller to access the page display memory a fixed number of times during each active display line. The actual calculation of the minimum frequency will therefore depend upon the duration of each individual access in terms of DMA cycles, the number of accesses to be made during the active part of the displayed line, and finally, the length of the active part of the line.

As an example, consider a system using the standard VDP programmed in 625 line mode with standard memory timing and displaying a page of pure text. Assume also that the dot clock, or timebase oscillator is running at its typical frequency for this mode of 7.25MHz giving a standard CCIR line duration of 64us. With these conditions, the VDP will be accessing the memory once every 1.085us to display one character. The display decoder needs to read three bytes from the memory in this time: a double access taking 8 DMA cycles to read the attribute and code bytes followed by a single access taking 5 DMA cycles to read the character generator giving 13 cycles in all. Since these 13 cycles must be performed in a maximum of 1.085us, a simple calculation of 1.085us/13 gives the maximum DMA period allowable of 83.5ns equivalent to a DMA frequency of 11.98MHz.

Thus any frequency below this will result in the display decoder not being able to read all the information necessary and picture degradation will result. Additionally, in this situation, the memories are occupied 100% of the duration of the active part of each line leaving no time for either data provider or CPU accesses. If either of these two devices request an access it will however be acknowledged since they have both higher priority than the display requests. The resulting access will reduce the time available for display accesses, which, being already minimum, will again cause display errors.

Under normal circumstances, therefore, it is unadvisable to use the theoretical minimum DMA frequency, unless neither the CPU nor the data provider ever make accesses the memory during the active part of the displayed lines. If absolutly necessary, it is possible to assure this latter condition by firstly setting bit CT4 to one in command register CM3, thus disallowing CPU accesses during the active part of each line, and secondly synchronising the VDP timebase to the incomming video signal so that in the case of VBI teletext operation, the data provider only requests accesses during the frame blanking period. Since both of these restrictions are not without their disadvantages, it is much better, if at all possible, to use the maximum practical DMA frequency, or at least one which allows the display decoder, CPU and data provider to access the memory asynchronously at their respective maximum rates without exceeding 100% memory ocupation and therefore degrading the displayed image quality. It should, however, be pointed out that missed display accesses from time to time, say less than one per frame on average, will be hardly noticable visually.

To determine the minimum DMA frequency when either CPU or data provider accesses are allowed during the active parts of each line it is necessary to calculate the worst case delay experienced by the display decoder.

In text mode the decoder requests a double and a single access at the start of each character time. If no other accesses are on-going at that time, a resynchronisation delay of up to 2.5 DMA cycles occurs as explained in section 3. If, during this time a CPU or data provider access request becomes active, because it is of a higher priority it is executed first causing an additional delay of 5 DMA cycles. The double display access is then performed taking a further 8 DMA cycles giving a total delay of 15.5 DMA cycles. For the information to be useable by the decoder, the cycle must complete in less than 9 dot clock periods. A simple calculation then gives the minimum DMA frequency to be 15.5/9 or 1.72 times the dot clock.

Performing the same calculation for the single display access gives a maximum delay of 20.5 DMA cycles. This must take less than the maximum delay of 11 dot clock periods available to the decoder giving a minimum DMA frequency of 1.86 times the dot clock.

Thus in text mode:

Minimum DMA frequency = 1.86 x Time base frequency

In bit-mapped mode the calculation is slightly different because of the alternating nature of double and single accesses. The worst case condition turns out to be when a CPU or data provider access request occurs between a single followed by a double display access. In this case the maximum delay experienced by the first byte of the double access is: 2.5 cycles for synchronisation, 5 cycles for the single display access, 5 cycles for the CPU or data provider access and 5 cycles for the first part of the double display access giving 17.5 DMA cycles in all. As for text mode operation, the first two bytes must be available to the decoder within 9 dot clock periods giving a minimum DMA frequency of 17.5/9 or 1.95 times the dot clock.

Thus in bit-mapped mode:

Minimum DMA frequency = 1.95 x Time base frequency

6.2 Time base considerations

The time base is responsible for generating all the timing signals required by the display decoder. These are derived from the master time base oscillator. The latter is also often reffered to as the dot clock since the duration of each displayed pixel is equal to one period of the oscillator.

In internal synchronisation mode, i.e. DC1 in command register CM2 is zero, the line and frame counters are perfectly synchronised to the dot clock resulting in a fixed length for each line which is an integer number of dot clock periods, and a fixed number of lines per frame as defined in Table 4.1. As a result, any variation in the oscillator frequency will be directly translated into an equivalent variation, in percentage terms, of the lengths of the scan lines and frames.

In general, it is possible to say that the maximum allowable variation in the duration of these two characteristics will be far less for the display device employed than the VDP itself; i.e. the monitor or TV specification should be used to determine the allowable variation in the dot clock frequency. The only real limit to timebase frequency variation, as far as the VDP is concerned in the absolute maximum frequency given in section 9.

6.3 Clock generation

As described in section 2, there are several techniques available to generate the two required clocks - DMA and timebase. The clocks may either be generated externally to the VDP and input on pins OBE and ODE for the time base and DMA oscillators respectively, or resonant circuits may be connected between pins OBE and OBS for the timebase, and ODE and ODS for the DMA oscillator. The resonant circuits themselves may either take the form of a ceramic resonator or an LC tank circuit as described in section 2. If a ceramic resonator is used, the manufacturers instructions should be observed as to any possible additional components to be included. It should be remembered, however, that an LC tank circuit must be used for the timebase in external synchronisation mode due to the need to resynchronise the timebase oscillator at the start of each scan line. Use of quartz crystals is not recommended for either oscillator due to their much higher Q and resultant start-up difficulties.

6.4 External synchronisation

In external synchronisation mode (DC1 = 0), the VDP uses the line and frame, or line and composite signals input on pins SLL and SCT to synchronise the line and frame counters as described in paragraph 3.8.3. Notably, the falling edge of the line sync signal present on SLL is used to directly reset the character or pixel counter and a new line is immediatly started. As described in paragraph 3.7.6 each new scan line starts immediately with five refresh cycles, the first of which starts immediately on the falling edge of SLL. To perform the refresh cycle, the VDP accesses the internal refresh accumulator ACREF which is physically located in the same memory block as the other on chip accumulators and base address registers. Under normal circumstances, no problem will arise since no internal RAM location is accessed during the line retrace period apart from ACREF. If, however the externally supplied sync signal on SLL happens to occur whilst the VDP is in the active part of the line, and hence accessing other internal registers, due to the internal delay times in the VDP the refresh cycle may well start before the preceeding internal access was complete resulting in corruption of the accumulator or base address register being accessed at the time and a modification to its value.

As a consequence, under no circumstances must a line sync pulse occur on SLL in external sync mode during the active part of any line unless the base addresses are reprogrammed afterwards. Notably, care must be taken in teletext receivers when changing from one received channel to another since in most cases, it is unlikley that the two channels will have line sync occuring at the same instant. In this case, the VDP must be reinitialised by reprogramming its base address registers following the application of the new synchronisation signals on SLL and SCT.

6.5 System interfaces

6.5.1 Memory interface

Since the VDP is designed to work with standard DRAMs there are few hardware considerations to be observed when designing the VDP to memory interface. Nevertheless, two particular considerations should be taken into account:

Firstly, reference to section 9 shows that the column address hold time after CAS goes low is fixed at 25ns, and is neither a function of the memory timing option selected nor the frequency of the DMA clock. Since this time is too short for most DRAMs, external hardware, in the form of a latch, must be added to comply with memory specifications. The TMS4164-15 for example specifies a minimum thCLCA of 45ns.

Secondly, all VDP to memory write cycles, including data provider accesses, are ordinary write cycles as oposed to what is termed 'early write cycles'. Reference to DRAM data sheets shows that this type of cycle results in the Q outputs of the memory going low impedance for part of the time when a write is performed. As a result, it is not possible to simply connect the D input to the Q output for each memory, as is done in simple memory systems, and a buffer must be used to isolate the two signal lines.

The recommended hardware implementation of these points is given in section 7.

6.5.2 Data provider interface

The data provider interface is composed of the two handshake lines HMP and HIZ and the eight bit VDP/memory data bus D0 to D7 as described in paragraph 2.3.2. When the data provider has a byte of data to be written into VDP memory, it first takes its access request line HMP active low. From paragraph 3.7.7 we can calculate that the minimum respose time, in the case when no memory cycles are on going, is 1.5 DMA cycles following which, the data provider access will commence.

In the worst case situation, a double display cycle will just have started when the access is requested and a delay of 7 (at least one cycle must have occured) DMA cycles (with standard memory timing selected) will result. The delay from the start of the cycle to the falling edge of HIZ is fixed at 1 DMA cycle making the total delay from the falling edge of HMP to the falling edge of HIZ variable from 2.5 to 8 cycles as shown in Figure 6.1.

FIGURE 6.1: Data provider access timing

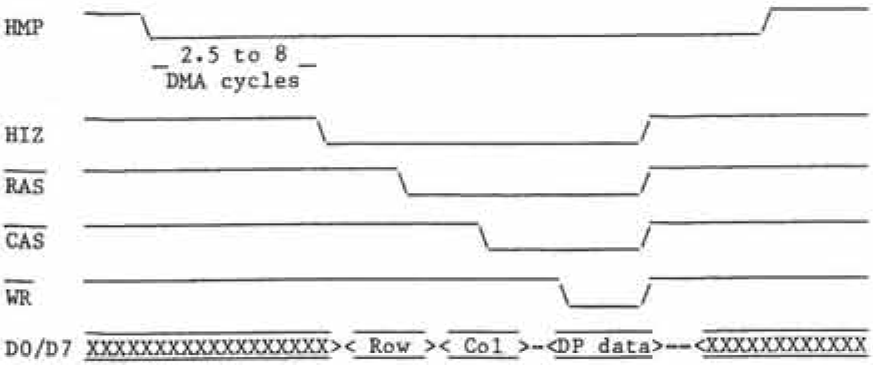


Figure 6.1 shows that the data provider must not immediately output its data onto the D0 to D7 bus as soon as it sees HIZ low because the VDP will not yet have strobed the row and column addresses into memory. Rather, it must wait for the VDP to tristate its D0 to D7 bus following the falling edge of CAS. Since the time delay from CAS falling to D0/D7 tristate is relativley short, a logical combination of HIZ and CAS is adequate in most cases to generate an output enable signal for the data provider.

Note that there is no maximum HMP low pulse width specified. This is because HMP is an edge triggered input and only the falling edge of HMP is recognised and latched internally in the the VDP. One idiosyncrasy of VDP operation should however be considered if HMP is to be left low for an extended period. This is that if the data input buffer is reset by programming the base address BAMT whilst HMP is low, the VDP internal logic generates a false data provider access request with the consequential undesirable modification to the various data buffer pointers, and in particular the input address pointer ACMT. Since the buffer is by definition empty after a reset, a false access request will cause the buffer empty flag to reset indicating that one byte has been written by the data provider whereas no access has in fact taken place.

7 APPLICATIONS EXAMPLES

7.1 CPU - VDP interfaces

7.1.1 TMS7000 Single chip interface

By far the simplest method of interfacing the TMS7000 CPU to the VDP from a hardware point of view is shown in Figure 7.1. The required VDP control lines, E1, E2 and RWM being simply derived from one of the output ports, in this case PORT B, and the eight bit bidirectional bus MP0 to MP7 is connected to a bidirectional port, in this case PORT C.

The TMS7000 is configured in Single Chip mode so that the applications software, contained in the on chip program ROM, has full and independant control over the state of each of the 11 interface lines.

Note that the RDY signal from the VDP is not used in this configuration, because in all cases the theroretical calculation shows that the VDP will always react faster than the interface can read or write data.

Typical TMS7000 routines for using this type of interface to read or write VDP internal registers or external RAM are given in section 8.

7.1.2 TMS7000 semi memory mapped interface

The interface described in paragraph 7.1.1 although being the simplest from a hardware point of view does suffer from the fact that the applications software must execute one instruction each time that one of the signal lines changes state. For example, to output a low going pulse on RWM, two instructions are required, one to send RWM low, and one to return it high again. This need to explicitly define the states of all interface signals means that the routines for communicating with the VDP are relatively time consuming. In many applications this presents no problem since a high rate of data transfer to and from the VDP is not required. For example, in conventional videotex terminals the recieved characters arrive at a rate of 120 per second giving ample time for any consequential CPU - VDP communication. In some cases however, notably in teletext applications where the input data rate is much higher, of the order of 1Mbyte per second, CPU - VDP communication must take place much faster and, in general, accesses to the memory will be much more frequent since, in addition to the much higher data rate, the CPU has to read the incomming data from the data input buffer in addition to writing the page display memory.

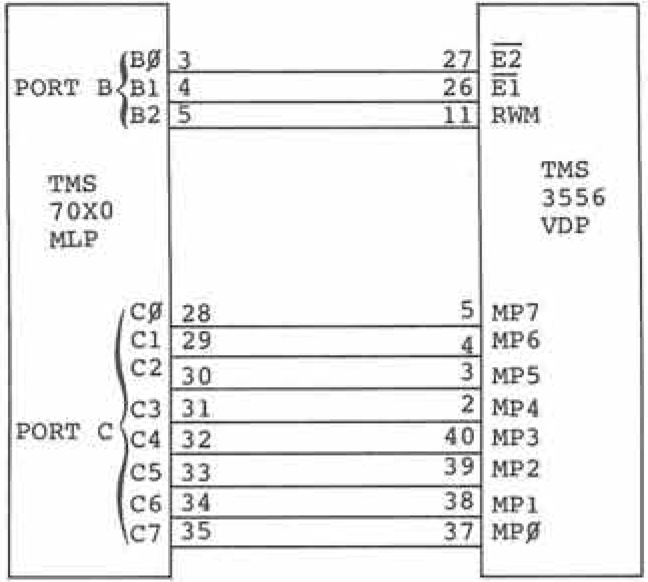


FIGURE 7.1 : TMS 70X0 SINGLE CHIP MODE INTERFACE

The interface shown in Figure 7.2 goes some way towards solving the problem. Termed a 'semi memory-mapped' interface it operates in a mixture of software driven port control and hardware driven memory bus control. The TMS7000 for this interface is configured in full expansion mode wherin some of the lines of the B port are used to output hardware generated memory control signals which, suitable modified, are used to generate some of the CPU - VDP interface signals. In addition the C port becomes the CPU's data bus in this mode and this is connected via a buffer to the VDP's MP bus. The software routines to access the VDP are now both shorter, in terms of object code, and more importantly, execute much faster.

Various examples of suitable software routines to access the VDP using this interface are given in section 8.

7.1.3 Full memory-mapped interfaces

The technique described in paragraph 7.1.2 of using the hardware generated CPU memory control signals to generate some of the VDP control signals can be taken further to a point where all of the required signals, including the actual reading or writing of the data to or from the MP bus, are generated from the CPU's memory control signals. This represents the ultimate performance achievable in terms of speed since, in most cases, only a single instruction is required to perform a read or write access to or from the VDP.

The general strucure of this type of interface is shown in Figure 7.3. The actual hardware realisation of the interface will depend very much on the processor used, but the principle remains the same: the E1 and E2 control signals are generated from the CPU's address bus - different access addresses being used by the CPU for each of the various VDP access types, RWM is generated from the CPU's R/W or WE output line with perhaps some timing modification to respect VDP requirements and the MP bus is simply tied directly to the CPU's data bus. Note that since this interface will result in accesses being performed in a much shorter time than with the two interfaces described above, it is possible that with high performance processors, the state of the RDY line must also be taken into account, and notably account must be taken of the fact that the ready line does not react identically for read and write cycles.

One example of a full memory-mapped interface used with a TMS7000 appears in Figure 7.4. In this case a PROM is used to replace some of the logic required in order to reduce the chip count. Because the TMS7000 does not provide a ready input, and that in some cases the read access time of the VDP will be superior to that required, a technique is employed wherby two consecutive read cycles are performed by the CPU to read one byte of data from the memory. The first starts the read access

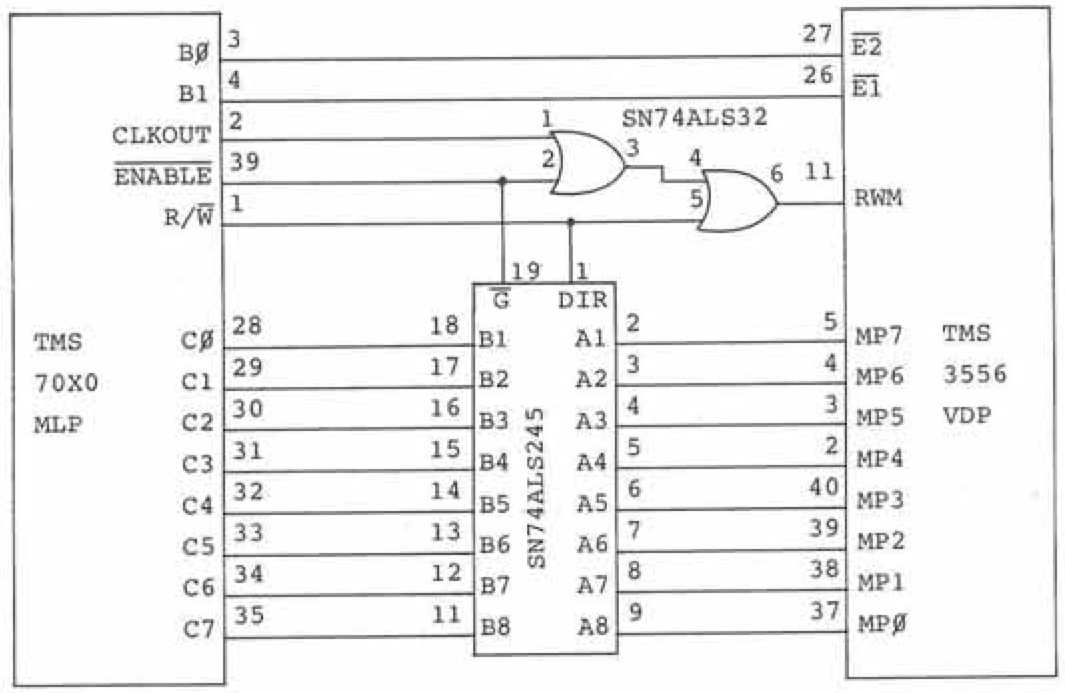


FIGURE 7.2 : TMS 70X0 SEMI MEMORY MAPPED INTERFACE

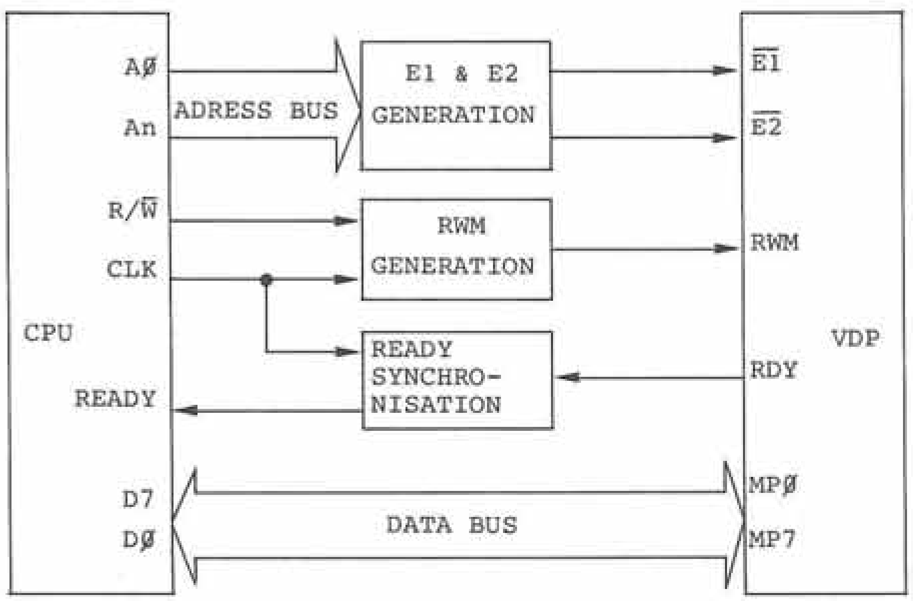


FIGURE 7.3 : GENERALISED MEMORY MAPPED INTERFACE

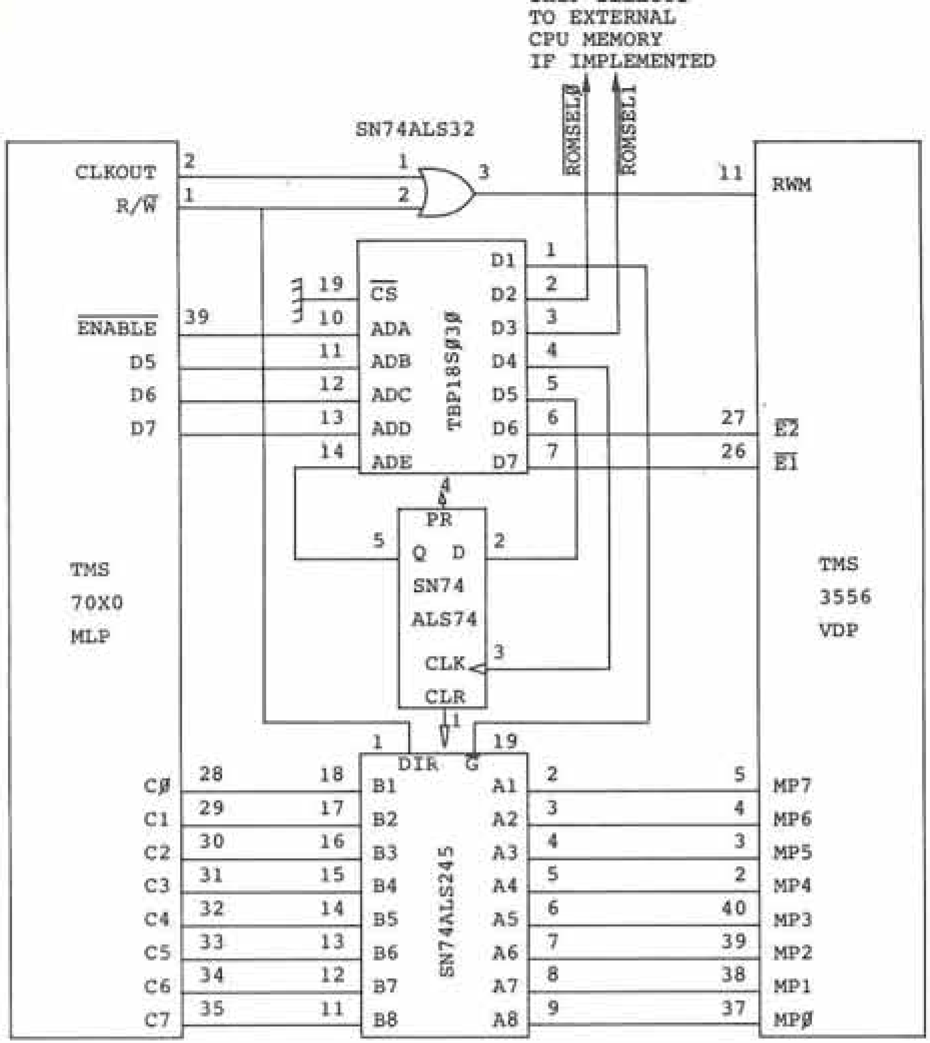


TABLE 7.1
PROM CONTENTS

TBP185Ø3Ø			
ADDRESS	DATA	ADDRESS	DATA
> 00	>1F	>10	> 7F
> 01	>1F	>11	> 7F
> 02	>ØE	>12	> 6E
> 03	>ØE	>13	> 66
> 04	>1E	>14	> 5E
> 05	>1E	>15	> 7E
> 06	>16	>16	> 7E
> 07	>1E	>17	> 7E
> 08	>1E	>18	> 3E
> 09	>1E	>19	> 7E
> 0A	>1E	>1A	> 7E
> 0B	>1E	>1B	> 7E
> 0C	>1D	>1C	> 7D
> 0D	>1F	>1D	> 7F
> 0E	>1B	>1E	> 7B
> 0F	>1F	>1F	> 7F

FIGURE 7.4 - TMS 70X0 FULL MEMORY MAPPED INTERFACE

FIGURE 7.4 : TMS 70X0 FULL MEMORY MAPPED INTERFACE

and the second retrieves the read data and puts the VDP interface back in the inactive state. This is not required for write cycles since the VDP's write access and cycle times (these are not the same) are sufficiently short in all cases.

Example software for this interface is given in section 8.

7.1.4 INTEL 8048 interface

A single chip interface between the VDP and an Intel 4048 processor appears in Figure 7.5. This operates in a similar manner to the TMS7000 single chip interface.

7.1.4 MOTOROLA 6801 interface

A single chip interface between the VDP and a Motorola 6801 processor appears in Figure 7.6. This operates in a similar manner to the TMS7000 single chip interface.

7.2 VDP - Data Provider interface

7.2.1 TMS3534

Figure 7.7 shows the recommended interface between the VDP and the TMS3534 data provider.

7.3 VDP - Memory interfaces

7.3.1 2 x TMS4416

The recommended circuit for interfacing the VDP to two TMS4416 16K by 4 DRAM's is shown in Figure 7.8. The ALS573 latch is required in the address lines to extend the ThCLCA time of the VDP which would otherwise be too short for the memories.

Note that the interface would be identical in the case of TMS4408 8K by 4 memories.

7.3.2 4 x TMS4416

The recommended circuit for interfacing the VDP to four TMS4416 16K by 4 DRAM's is shown in Figure 7.9. In this case the column address strobe CAS is decoded to select one pair of memories depending upon the state of the VDP's A1 address line which is output on D7 just prior to CAS going low.

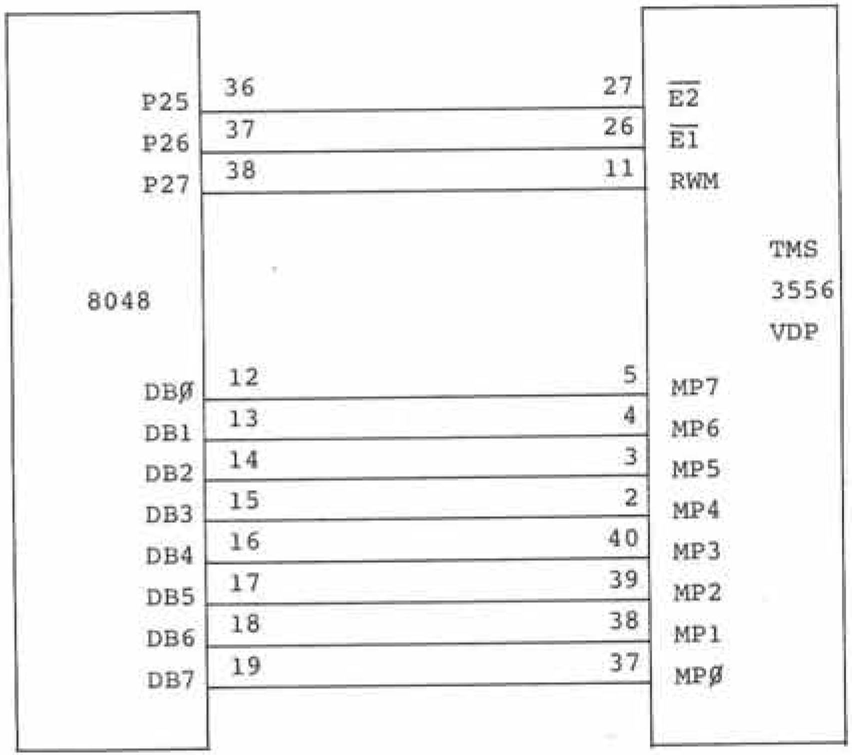


FIGURE 7.5 : INTEL 8Ø48 INTERFACE

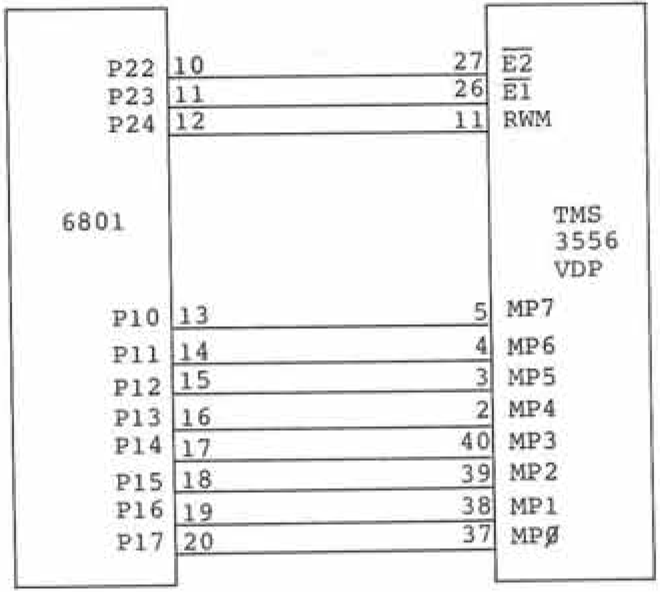


FIGURE 7.6 : MOTOROLA 68Ø1 INTERFACE

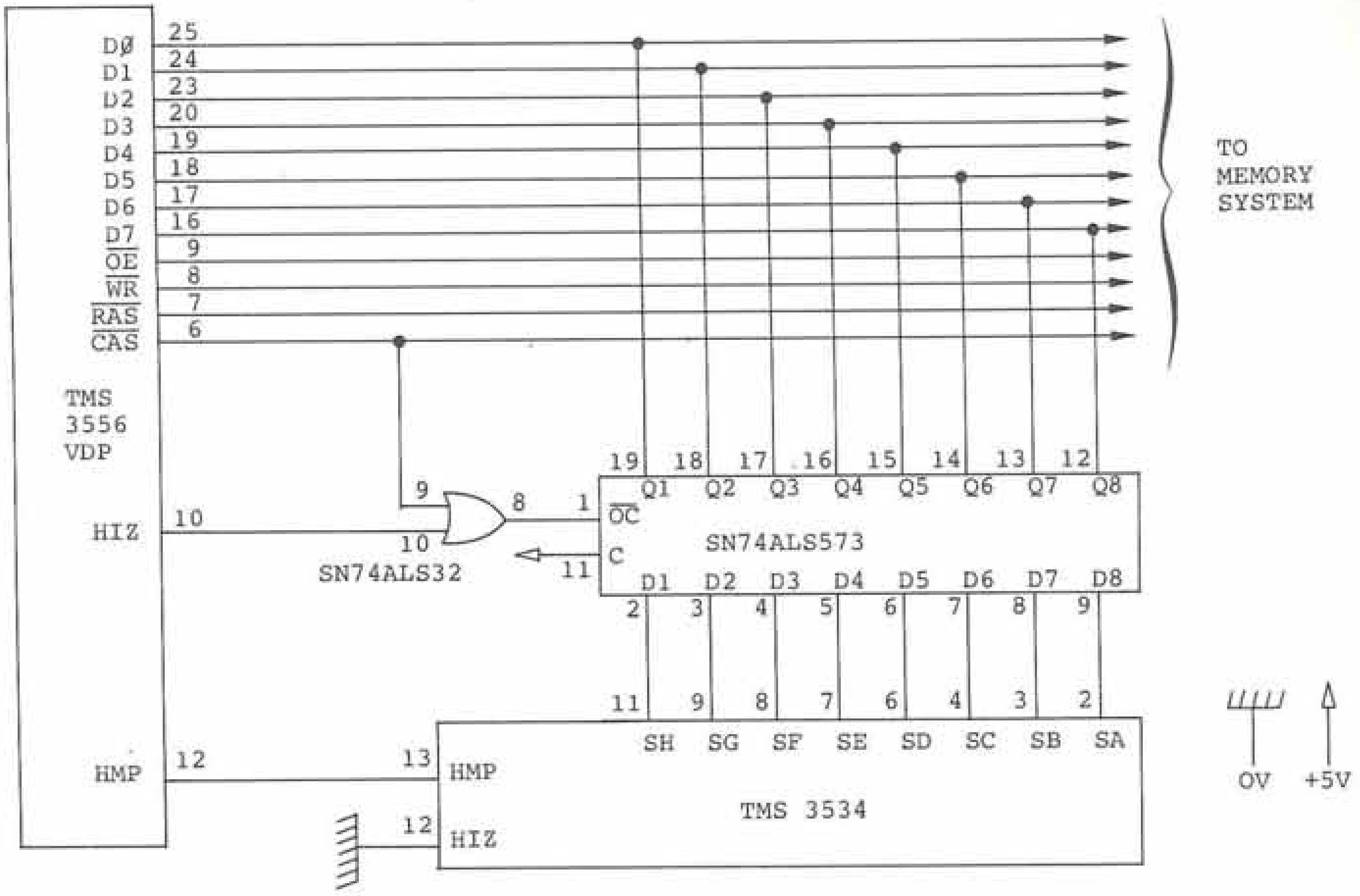
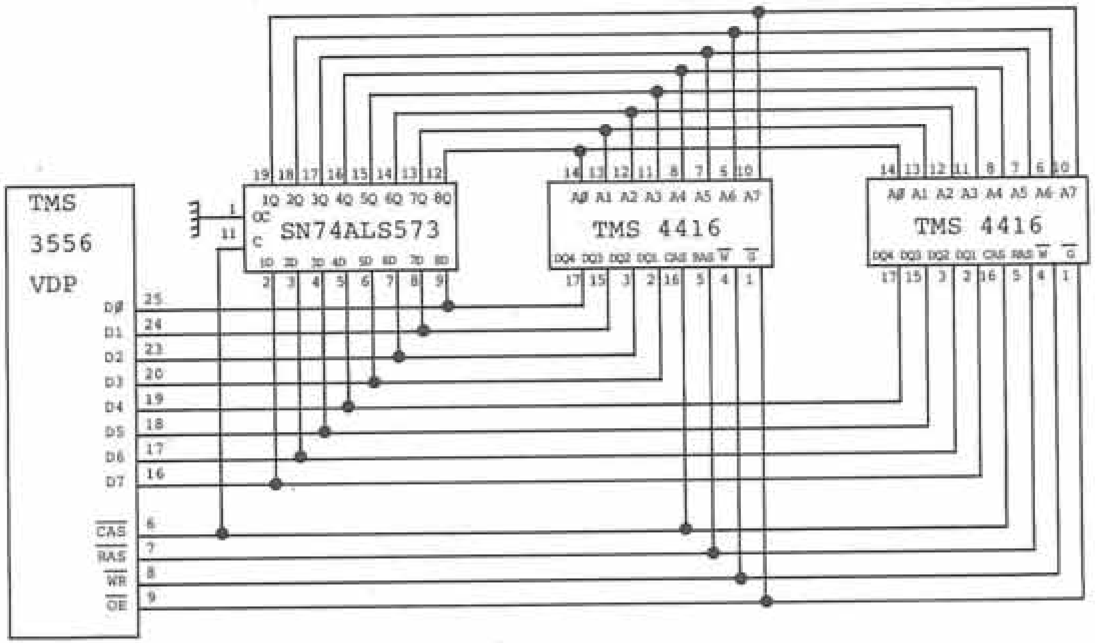


FIGURE 7.7 : TMS 3534 INTERFACE



— FIGURE 7.8 : MEMORY INTERFACE FOR 2 X TMS 4416 —

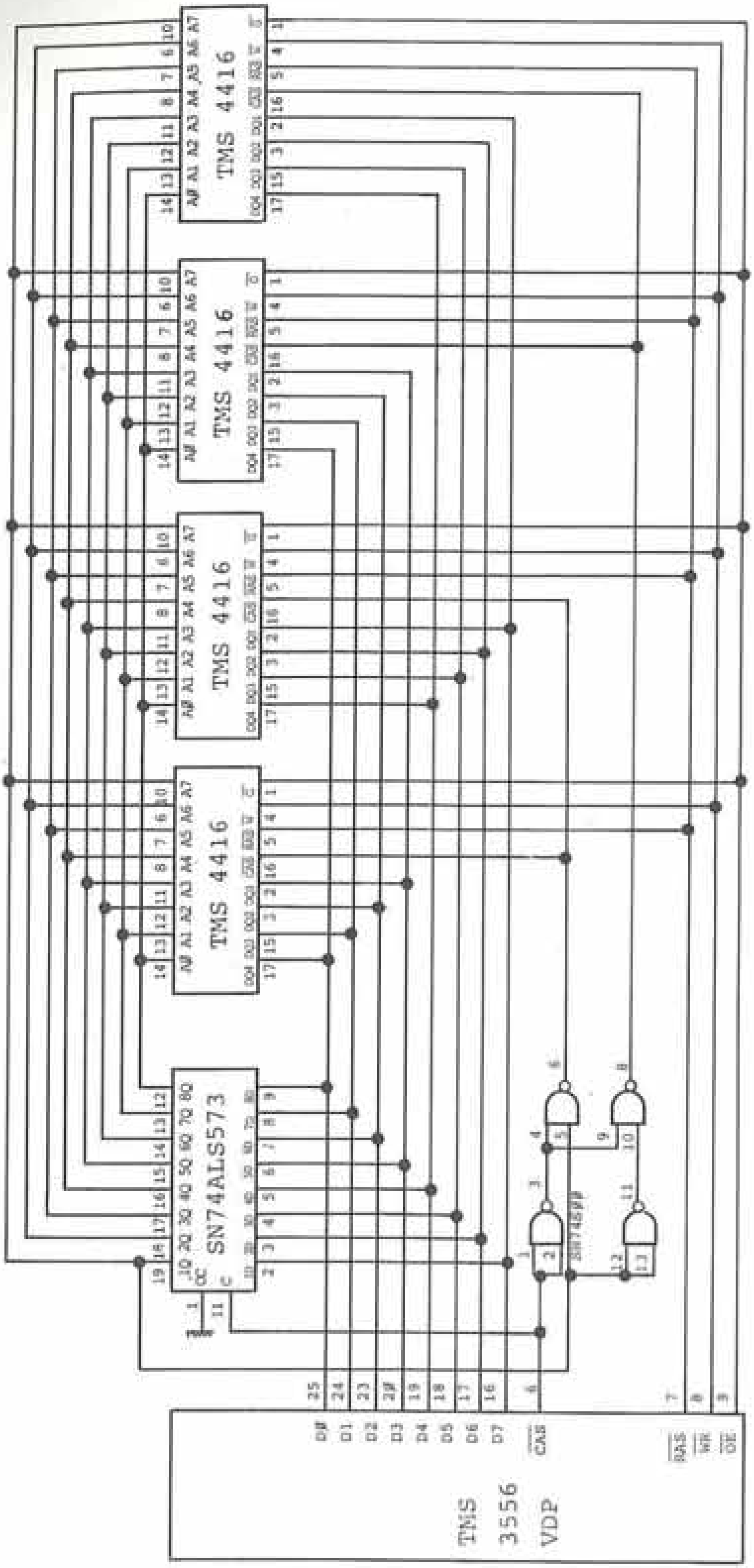


FIGURE 7.9 : MEMORY INTERFACE FOR 4 X TMS 4416

7.3.3 6 or 8 x TMS4416

The recommended circuit for interfacing the VDP to six or eight TMS4416 16K by 4 DRAM's is shown in Figure 7.10. In this case the column address strobe CAS is decoded to select one pair of memories depending upon the state of the VDP's A0 and A1 address lines which are output on D0 and D7 respectively just prior to CAS going low.

7.3.4 8 x TMS4164

The recommended circuit for interfacing the VDP to eight TMS4164 64K by one DRAM's is shown in Figure 7.11. No CAS decoding is required in this case, but an extra latch or buffer must be used to separate the D inputs from the Q outputs of the memories. This is because the VDP only performs ordinary write cycles, as opposed to early write cycles, and the Q outputs will go active even during a write cycle.

7.4 VDP - TV interface

7.4.1 Domestic TV standard RGB interface

Figure 7.12 shows an example interface between the VDP and a standard RGB domestic TV connector.

7.4.2 Grey scale interface

Figure 7.13 shows an example interface used to generate an eight level grey scale in addition to the normal eight colours. This interface employs two PROMs to either pass the RGB signals directly to the outputs, or to convert them via a simple resistive D to A converter to an 8 level analogue signal. The choice between colour or grey scale is controled by the state of the I output signal from the VDP; this interface can therefore not be used in systems which require the I output signal for incrustation purposes.

The programmed contents of the PROM's are given in Table 7.1.

7.4.3 Colour palette

Figure 7.14 shows the diagram of a circuit which provides the possiblity of generating up to 16 colours on one display screen out of a total possible of 4096 using one VDP. The circuit is composed of two separate palettes of eight colours each. The palette used for each section of the display is selected by the state of the I output signal. The interface to allow the CPU to modify the palette contents is also shown in the same diagram.

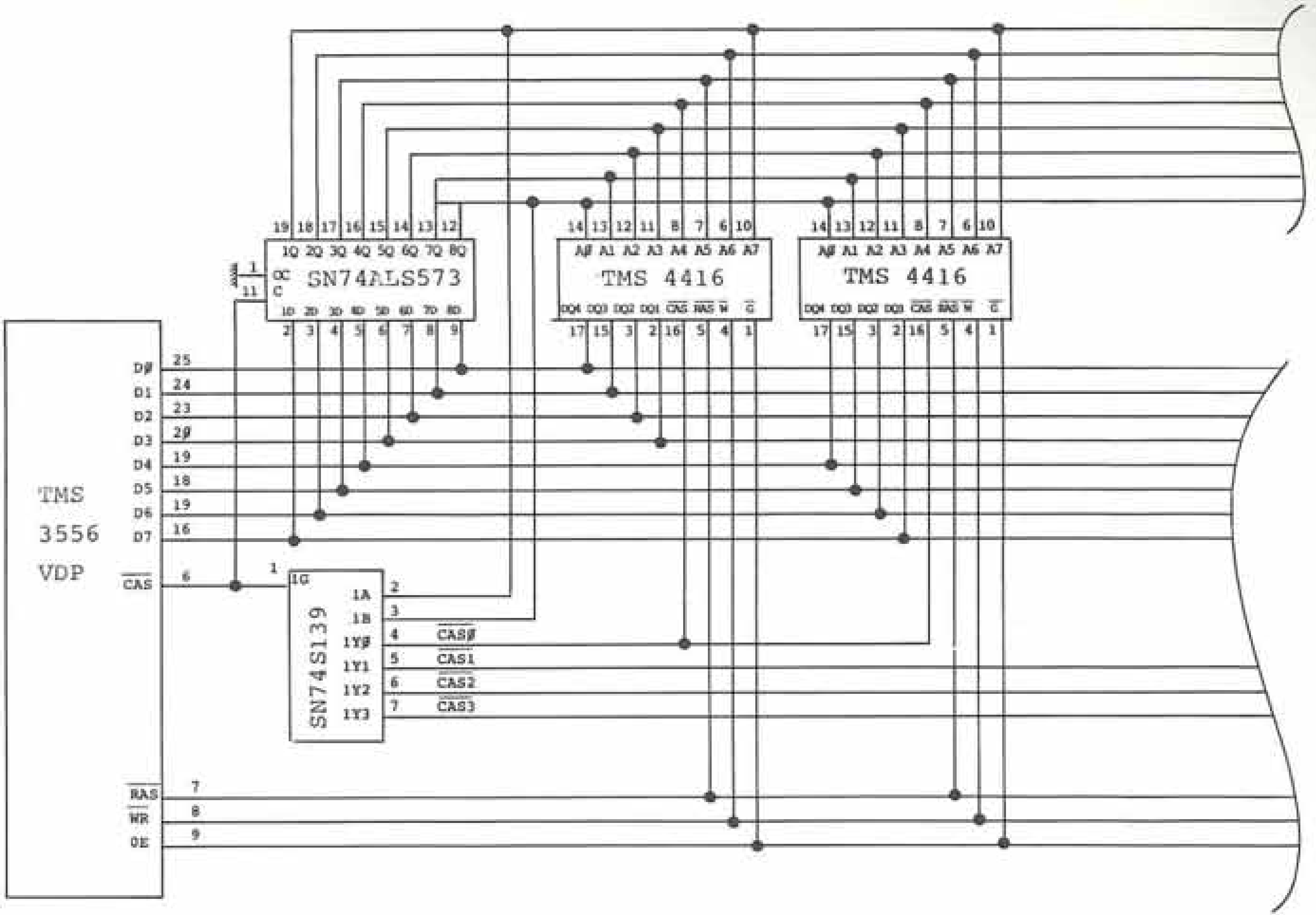


FIGURE 7.10 : MEMORY INTERFACE FOR 6 OR 8 X TMS 4416

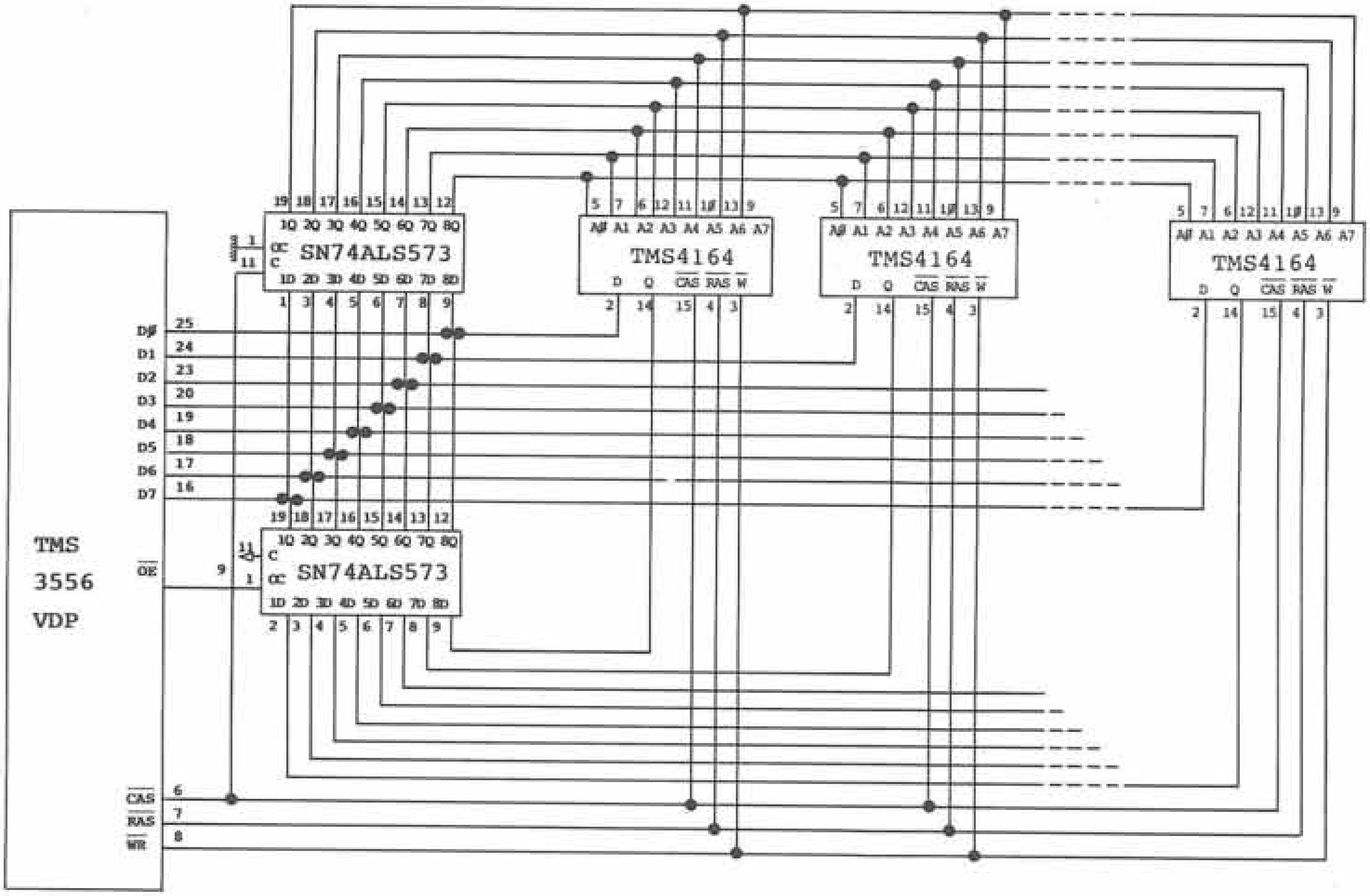


FIGURE 7.11 : MEMORY INTERFACE FOR 8 X TMS 4164

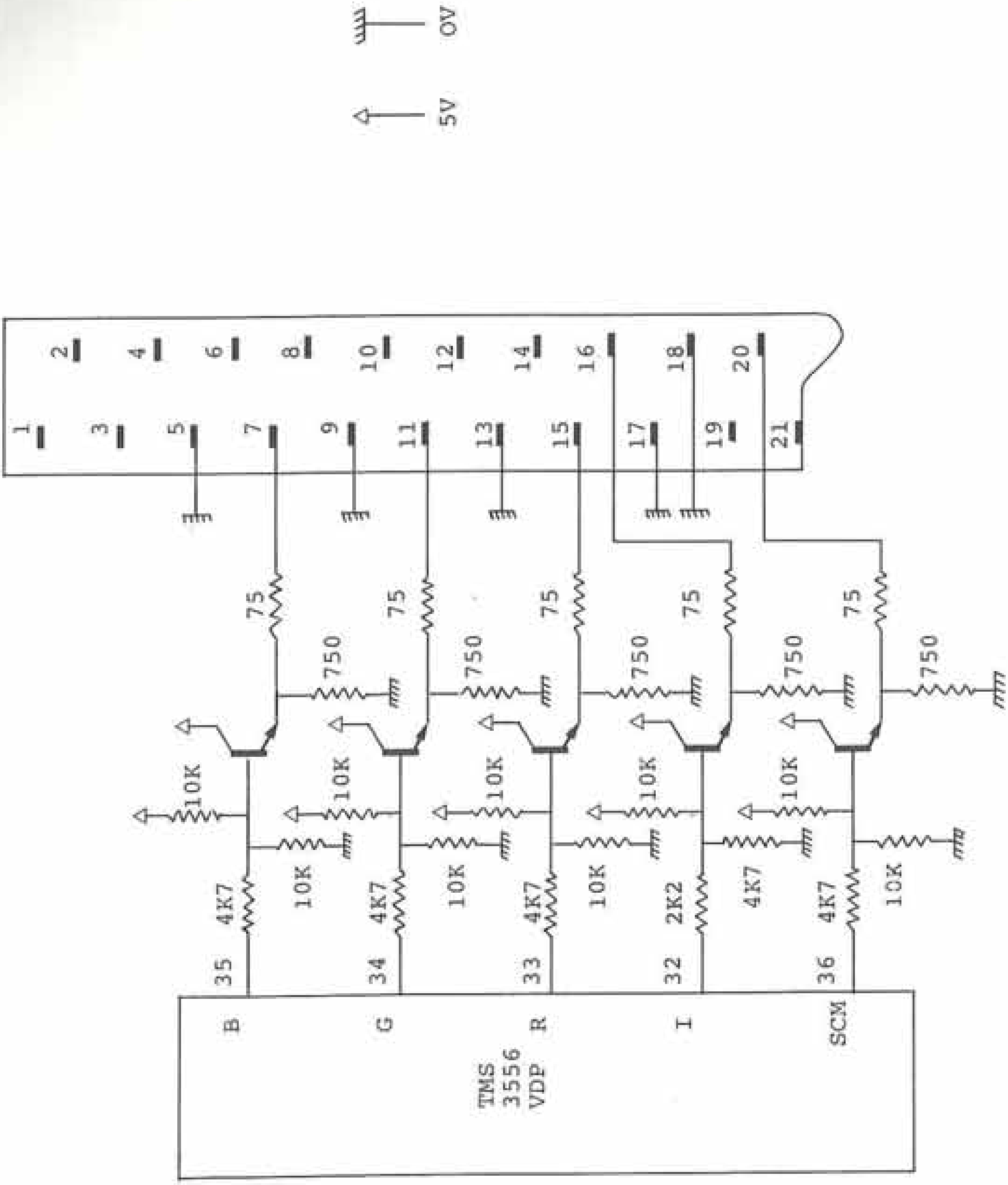


FIGURE 7.12 : DOMESTIC TV RGB INTERFACE

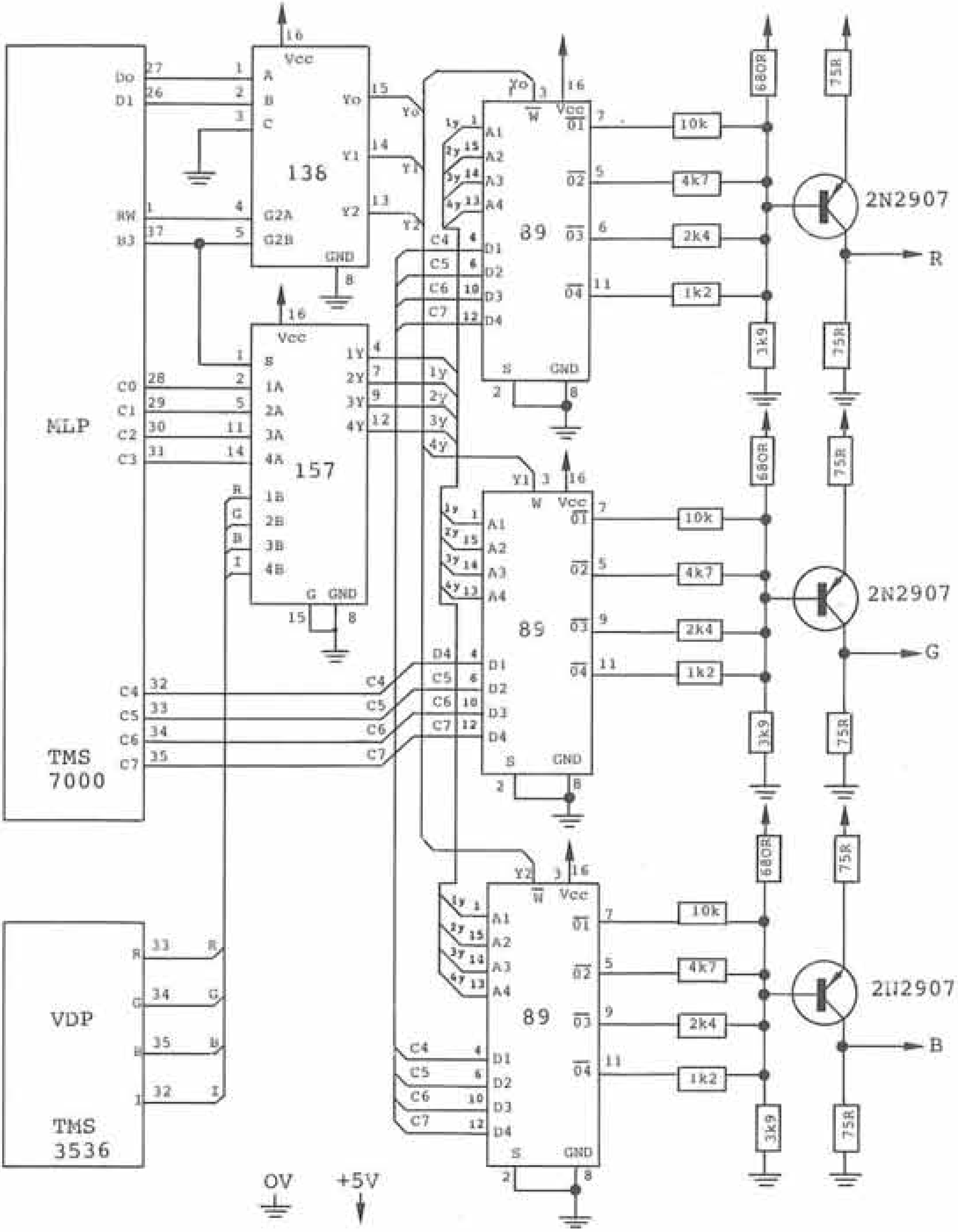


FIGURE 7.14 : 16 COLOUR PALETTE CIRCUIT

7.5 Vidoetex terminal examples

The basic videotex terminal structure is shown in Figure 7.15. One of the advantages of a system based on the TMS3556 VDP is that the same basic structure is used for a whole range of systems. The only changing elements are the memory size, and possibly the processor used.

Discussion of the details of the telephone line interface is outside the scope of this document.

7.5.1 Minimum alpha-mosaic terminal

The minimum alpha-mosaic terminal uses the basic structure shown in Figure 7.15 with a minimum memory system. A suitable memory arangement would be that shown in Figure 7.8 using either two TMS4408's or TMS4416's giving 8K or 16K bytes of memory.

7.5.2 Multipage alpha-mosaic terminal

The multipage alpha-mosaic terminal is identical to that described in paragraph 7.5.1 with the exception of increased memory size. The actual ammount of memory used depends upon the ammount of memory occupied by the character generators etc. plus the number of extra pages that it is wished to store. Note that in many cases it will take less room to store non displayed pages in their coded form as they are received as opposed to their decoded form. In addition, compression algorithms can be used to further reduce the space required to store non displayed pages.

7.5.3 Alpha-geometric terminal

The alpha-geometric terminal again uses the same basic architecture as the alpha-mosaic terminals. In this case however, the VDP operates in bit-mapped mode with the consequential increase in minimum memory requirement. Typically 30500 bytes of memory will be required for each page stored, although as for multipage mosaic terminals, compression algorithms may be used to reduce the size of non displayed pages, or the latter may be stored in non-decoded form if this takes up less room.

7.6 Teletext application

The difference between a videotex terminal and a teletext one lies in the source of the data to be displayed. For teletext, the data arrives via an off-air video signal and must first be extracted via a teletext data demodulator such as the SN96533 and a prefix processor or demultiplexer such as the TMS3534. These two circuits taken together form what is termed in the rest of this users guide the 'data provider'. Additionally, the teletext terminal will implement a remote control receiver as opposed to a local keyboard. The interface with the remote control will vary widely from terminal to terminal depending on the manufacturers choice of system, and it is immpossible to give typical examples of implementation.

The circuit diagram of a typical teletext terminal is shown in Figure 7.16. It should be noted that this is the circuit diagram of an evaluation system designed to demonstrate the capabilities of a teletext system based on the TMS3556, and as such is not necessarily the optimum implementation in all applications.

7.7 Dual VDP's

The TMS3556 VDP contains all of the features required to satisfy the needs of the vast majority of colour video terminal applications such as teletext and vidoetex as we now know them. However, it is possible to construct terminals using the TMS3556 with features and performance going beyond those currently required by these applications. This is effected by the use of two or more VDP's configured in parallel giving, amongst other things, a greater number of colours, higher resolution or both. This section describes the hardware considerations required to implement dual VDP based terminals.

7.7.1 Synchronisation considerations

If two VDP's are to be used in parallel they must necessarily be synchronised together as far as their respective display timing is concerned. This implies that they share a common dot clock and common line and frame synchronisation signals.

The first of these two points is easily satisfied by supplying bith VDP's with a common external time base clock as shown in Figure 7.17.

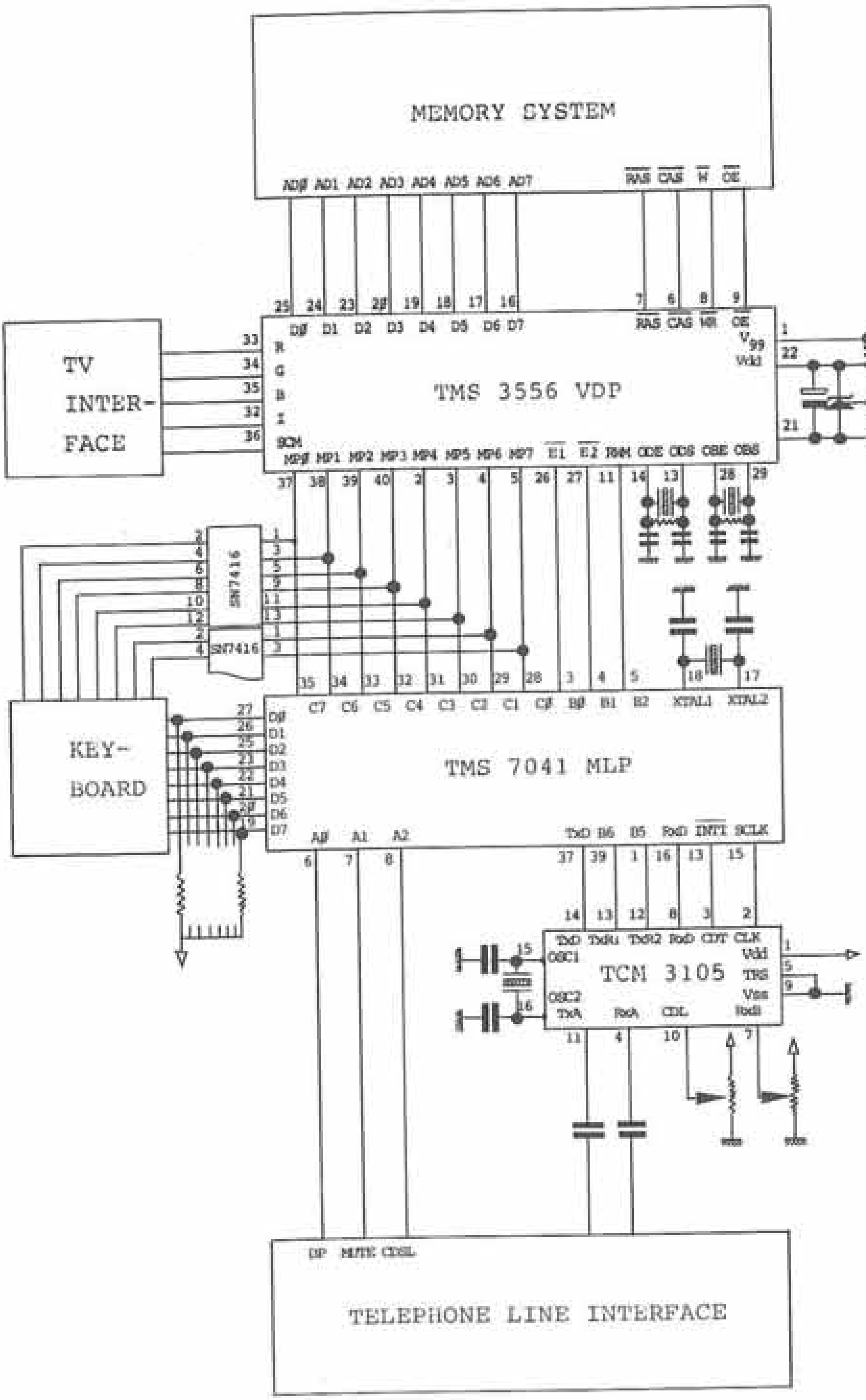


FIGURE 7.15 : VIDEOTEX TERMINAL STRUCTURE

FIGURE 7.17: Dual VDP time base synchronisation with common external clock

```
                          .-----------.
                          | External  |
                          | oscill-   |
                          |   ator    |
                          `-----------'
                                |
                     -----------|-----------
                    |                       |
                    V                       V
        .----------------.       .------------------.
        |        OBE     |       |       OBE        |
        |    VDP   1     |       |     VDP   2      |
        `----------------'       `------------------'
```

Alternatively, if the dot clock is not generated externally, the timebase oscillator output from one VDP can be used to supply an external clock signal to the second VDP as shown in Figure 7.18.

FIGURE 7.18: Time base synchronisation with master and slave VDP's

```
       .------------------------------.
       |       Master    VDP          |
       |      OBE              OBS    |
       `------------------------------'
                |              |
                |  .---------. |
                |-|Resonator |-|
                   `---------' |
                               |
                               V
       .------------------------------.
       |                       OBE    |
       |      Slave    VDP            |
       `------------------------------'
```

The method of synchronising the two VDP's from a line and frame point of view are similar: i.e. either line and frame sync signals can be generated externally and fed to both VDP's, or one VDP is designated the 'master' and outputs sync signals to the other VDP designated the 'slave'. In the first case, both VDP's are configured in external synchronisation mode, and in the second, the master VDP is configured in internal synchronisation mode, and the slave in external synchronisation mode. The diagram for the second configuration appears in Figure 7.19.

FIGURE 7.19: Display synchronisation with master and slave VDP's

```
         .-------------------------------.
         |        Master   VDP           |
         |      SLL              SCT     |
         .-------------------------------.
                |                |
                |                |
                V                V
         .-------------------------------.
         |       SLL             SCT     |
         |       Slave    VDP            |
         .-------------------------------.
```

Using the configuration shown in Figure 7.19, although simplifying the display synchronisation does complicate slightly the synchronisation of the two time bases. This arises because the slave VDP, being in external synchronisation mode, tries to resynchronise its time base oscillator following each falling edge of its SLL input indicating the start of a new scan line as described in paragraph 3.8.3. The slave VDP does this by pulling its OBE input low, and therefore overloading the OBS output of the master VDP if the latter is trying to drive a high level. This could be avoided by including a series limiting resistor in the line from OBS to OBE but in fact this is not necessary as described below.

Unfortunately, the oscillator resynchronisation period described in paragraph 3.8.3 during which OBE and OBS are held low is of an indeterminate duration and varies considerably as a function of temperature and between diferent VDP's. The period itself will be of the order of several dot clock periods. Since the slave VDP is not recognising the dot clock input during this time, when it eventually removes the low level on OBE and OBS it will start its next scan line with an indeterminate delay with respect to the master VDP.This is because the latter, being in internal synchronisation mode, does not resynchronise its time base oscillator and so starts a new scan line simultaneously with the falling edge of its SLL output.

Additional to the unknown delay of the slave VDP when in external synchronisation mode is a fixed delay of two clock periods due to the internal resynchronisation logic. This must also be taken into account be the resynchronisation hardware as described below.

There are several methods of overcoming the problem of unknown slave delay. The method used depends upon the source of the dot clock itself. The methods described below use the technique of halting the slave VDP purposly for a known time immediately following the detection of the falling edge of SLL output from the master VDP. The length of time is chosen to be superior in all cases to that during which the slave VDP would have halted itself. The result of this action is that the resultant delay of the slave VDP with respect to the master VDP is then accurately known, and steps may be taken to compensate for it.

The first technique is useable when the master VDP is generating its own dot clock and also the dot clock for the slave VDP. Figure 7.20 shows the principle of the technique used. Immediately upon detecting the falling edge of the masters SLL output, two consecutive pulses are removed from the dot clock signal input to the slave device. During this time the slave VDP tries to resynchronise its oscillator and then prepares itself for further pulses on its OBE input. These two 'lost' dot clock cycles, added to the two cycles lost due to the internal logic as described above, add up to a total delay of four dot clock cycles between the master and slave VDP's. This delay is compensated for by delaying the RGB and I outputs of the master VDP by four dot clock periods so that both VDP's are perfectly synchronised as far as the display device is concerned.

FIGURE 7.20: Synchronisation of dual VDP's with local clock

```
.------.                                 .------.
|      |                                 |      |
|  SCT |--------->----------------->-----|SCT   |
|      |                                 |      |
|  SLL |--------->-----------+----->-----|SLL   |
|      |                     |           |      |
|      |               .-----------.     |      |
|  OBS |--->---+--->---| -2 pulses |-->--|OBE   |
|      |       |       `-----------'     |      |
|Master|  .---------.                    |Slave |
| VDP  |  |resonator|                    | VDP  |
|      |  `---------'                    |      |
|      |       |                         |      |
|  OBE |-------'                         |      |
|      |  .--------.                     |      |
|     R|--|   4    |-----> Red           |     R|------> Red
|     G|--| period |-----> Green         |     G|------> Green
|     B|--| delay  |-----> Blue          |     B|------> Blue
|     I|--|        |-----> Incrust       |     I|------> Incrust
`------'  `--------'                     `------'
```

The second technique is useable when the dot clock supplied to both VDP's is generated externally. The technique is similar to that described above with the difference that the dot clock signal supplied to the master is also modified following the fallin edge of SLL by the removal of four clock pulses. Two of these serve to compemsate for the two lost by the slave VDP during its resynchronisation time, and the other two compensate for the intrinsic two dot delay of the slave VDP. Thus the RGB outputs of both VDP's are then synchronised directly at their respective outputs without need for further timing modification. Figure 7.21 shows the principle involved and Figure 7.22 shows a circuit diagram of a suitable harware arangement to perform this function.

FIGURE 7.21: Synchronisation of two VDP's with external clocks

```
.-------.                                                 .-------.
|       |                                                 |       |
|   SCT |--------->----------------->-----|SCT    |
|       |                                                 |       |
|   SLL |---->-----+------->------+--->-----|SLL    |
|       |          |                  |                   |       |
|       |   .------------.    .------------.              |       |
|   OBE |-<-| -4 pulses  |    | -2 pulses  |->-|OBE    |
|       |   `------------'    `------------'              |       |
|Master |          |                  |                   |Slave  |
| VDP   |          `---<--+-->---'                   | VDP   |
|       |                 |                               |       |
|      R|-----> Red    External                          |      R|-----> Red
|      G|-----> Green    dot                             |      G|-----> Green
|      B|-----> Blue    clock                            |      B|-----> Blue
|      I|-----> Incrust                                  |      I|-----> Incrust
`-------'                                                 `-------'
```

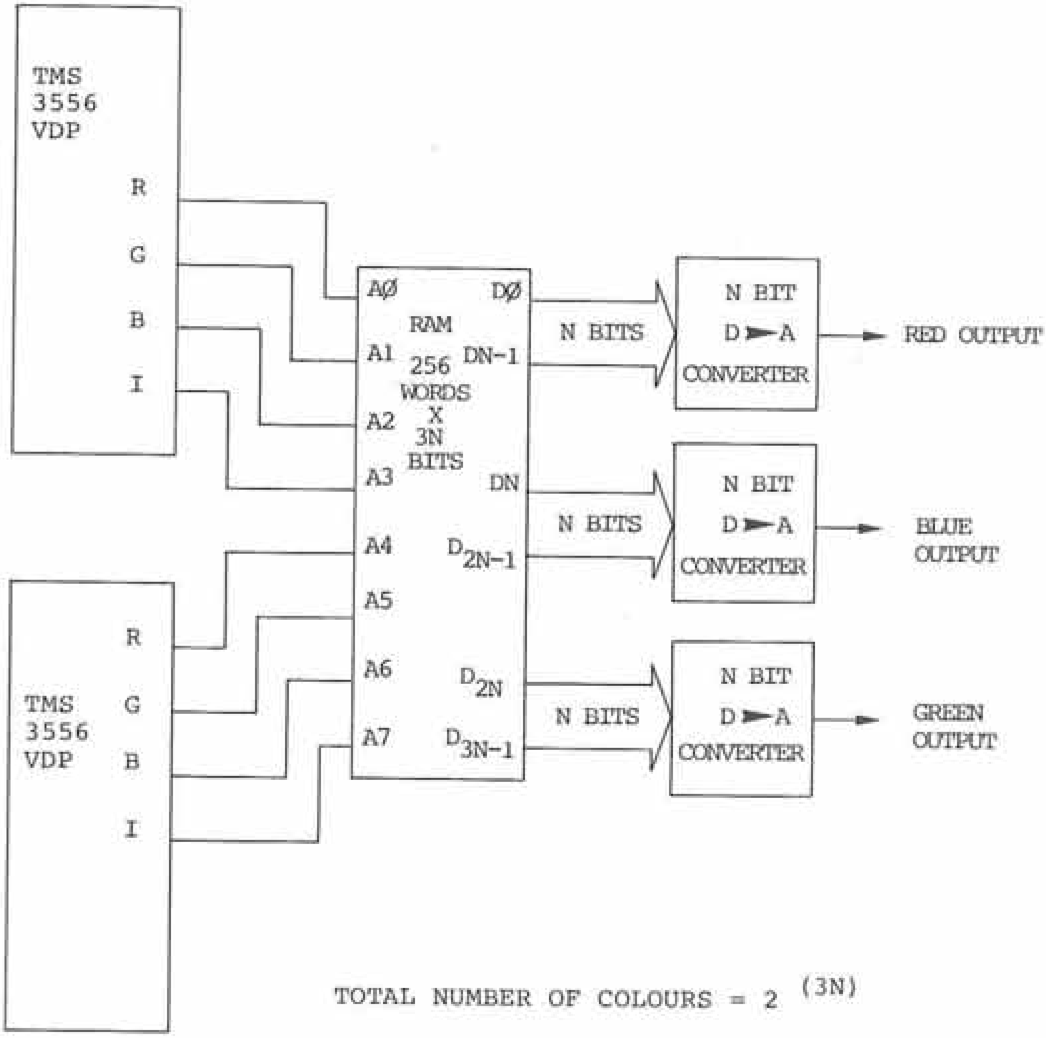
7.7.2 Colour palette

The two sets of RGB and I outputs resulting from the strucures described above can be used for a variety of purposes. One use is to increase both the total number of colours available for image creation and the maximum number of colours possible on on displayed page at any one time.

This is effected by using the VDP outputs not as direct signals for the display device, as with a single VDP, but as addresses for a colour look-up memory driving, in turn three digital to analogue converters as shown in Figure 7.22.

The maximum number of colours on any one displayed screen is a function of the number of outputs from the VDP's. Here we have six independat colour outputs giving a choice of 64 different colours for each pixel or character of the display. If, in addition, the I outputs are also used to address the colour RAM, the total number of colours per screen increases to 256 although the serial nature of the incrustation attribute restricts the way in which these colours are mixed to a certain extent.

FIGURE 7.22 : PALETTE CIRCUIT FOR 2 VDP's



TOTAL NUMBER OF COLOURS = $2^{(3N)}$

The total number of colours possible is simply a function of the word length of the colour memory and analogue to digital converters. For example if 4 bits per primary are stored in the memory, sixteen distinct levels per primary result, giving a total choice of 4096 colours.

7.7.3 Double resolution

Another use of dual VDP's is to increase the resolution of the displayed image, or the number of characters per line. For this, the VDP outputs are time multiplexed in a suitable manner so that each VDP supplies alternate pexels or characters of the displayed image. Figure 7.23 shows a diagram of two VDP's used to generate a bit-mapped display with a resolution of 640 by 250 pixels.

7.8 Monochrome VDU type terminal

If the display device used in the VDP based system is monochrome only, then it would appear that two out of the three available output signals are unuseable. This is not the case, since the outputs may be mixed via a suitable interface to give a three level signal: black, white and half intensity, with twice the normal definition. Figure 7.23 shows the principle involved in the construction of an 80 character by 25 row and 640 by 250 pixel dual intensity monochrome terminal using a single VDP. It should be noted however, that with the simple interface shown the VDP is restricted to bit-mapped mode operation only.

FIGURE 7.23 : DOUBLE RESOLUTION CIRCUIT
FOR 2 VDP's

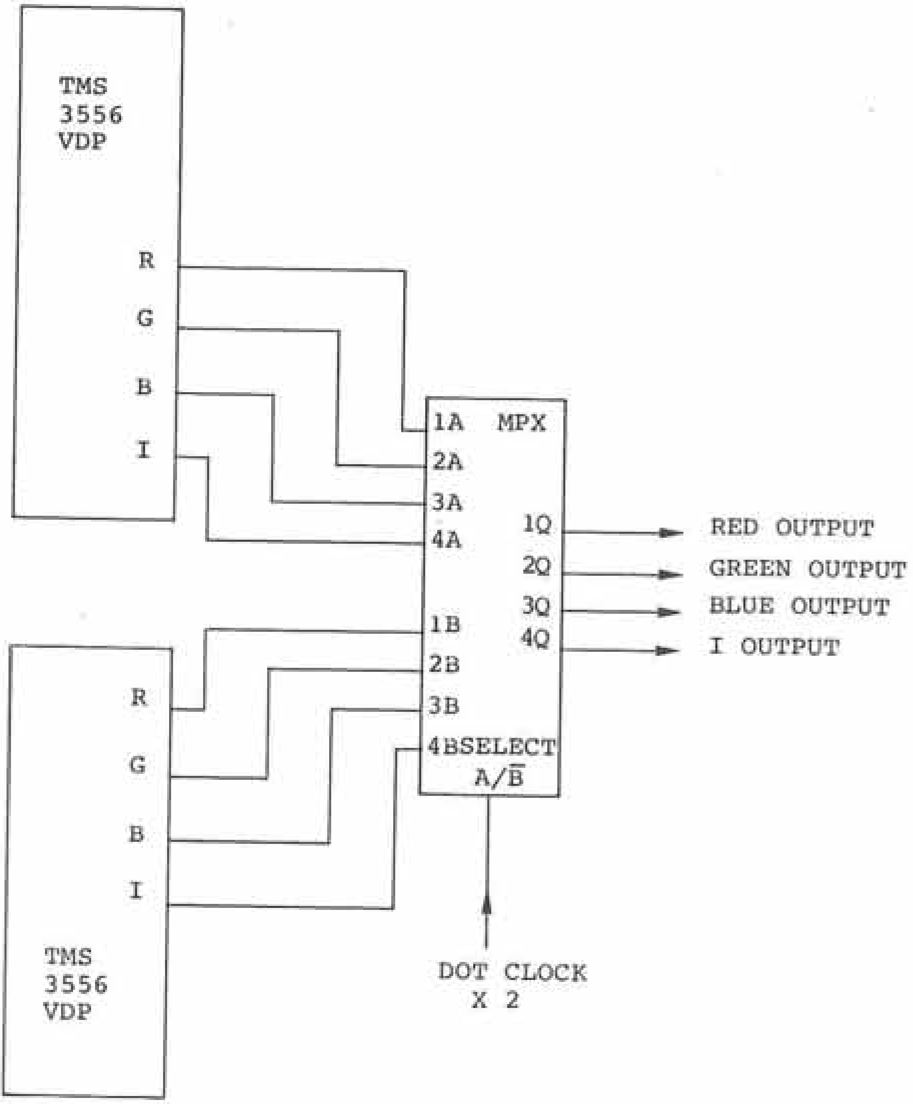


FIGURE 7.24 : 80 CHARACTER MONOCHROME VDU TYPE TERMINAL

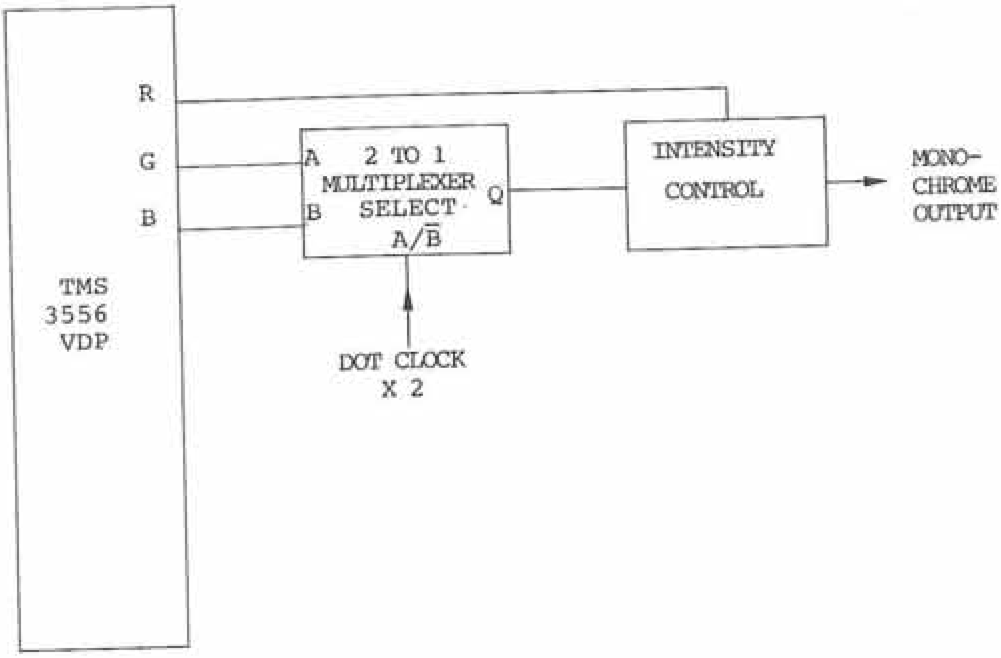


TABLE 7.2: PROM CONTENTS FOR GREY SCALE INTERFACE

GREY SCALE

ADDRESS	CONTENTS	ADDRESS	CONTENTS
0	>FE	8	>EF
1	>FF	9	>FF
2	>FD	A	>DF
3	>FF	B	>FF
4	>FB	C	>BF
5	>FF	D	>FF
6	>F7	E	>7F
7	>FF	F	>FF

COLOURS

ADDRESS	CONTENTS	ADDRESS	CONTENTS
0	>FF	8	>F7
1	>FF	9	>F6
2	>FF	A	>F5
3	>FF	B	>F4
4	>FF	C	>F3
5	>FF	D	>F2
6	>FF	E	>F1
7	>FF	F	>F0

8 SOFTWARE EXAMPLES

8.1 General description

A variety of software modules have been developed by Texas Instruments for the TMS7000 family of 8-bit microcomputers to control the TMS3556 VDP in a wide range of applications. These include Teletext (both VBI and Full-Field), Videotex (Prestel and Minitel/Teletel) and bit-mapped Graphics (NAPLPS).

The software is written in modular form so that it can be easily added to or reduced to suit individual applications. Parameters which are constants at assembly time are assigned names which are defined in an equalities file. Initial values of registers are defined in a separate table and use is made of the TRAP feature of the TMS7000 to call frequently used subroutines. Also all RAM registers are given names relative to their usage to aid readability of the routines. These are defined in another separate file. Finally, the routines which control the interface to the VDP are in the Handler file.

Thus the files used to create an application are:

1. RAM definitions
2. Equalities definitions
3. Specific application source
4. Handler routines
5. Initialisation tables
6. Trap definitions

Examples of each of these files follow.

8.2 RAM definition file.

This file reserves RAM locations for commonly used parameters so that their values can be passed between subroutines without having to know, define or calculate, their current values unless they are modified within the routine. This makes it easier to write source code which is transportable between different applications.

```
************************************************************
* ALLOCATION OF THE INTERNAL REGISTERS OF THE MLP          *
************************************************************
         AORG      >0002     Addresses 0 & 1 = regs A & B

LENGTH   BSS       2         Number of points-1 in a line
*                            (Graphics)
COL      BSS       1         Character position on a line (0-39)
TEINTE   BSS       1         Colour of a pixel or character
*                            Or grey scale value
BIT      BSS       1         Bit position of a Pixel within an
*                            eight-points data byte in VDP RAM
RAN      BSS       1         Text row number (0 to 24)
INDADR   BSS       2         VDP RAM address pointer - 16 bits
*                            Indadr = MSByte, Indadr+1 = LSByte
************************************************************
HEIGHT   BSS       1         Vertical length of a line (Graphics)
FLAG     BSS       1         Byte containing 8 independent flags
KFLAG1   BSS       1         4 flags used by keyboard decode
A0       BSS       1         ] General purpose pointer MSByte
A1       BSS       1         ] General purpose pointer LSByte
A4       BSS       1         ) General purpose pointer MSByte
A5       BSS       1         ) General purpose pointer LSbyte
LIGNE    BSS       1         Vertical position of a point (0-249)
CMTAB    BSS       4         Ram image of VDP registers CM1 - CM4
POINT    BSS       2         Horizontal position of a pixel
*                            (0 to 319)
*                            Point = MSByte, Point+1 = LSByte
STBIT    BSS       1         Bit position of 1st Pixel of a line
PRIMAI   BSS       1         Primary colour number (1, 2 or 4)
FEN      BSS       1         Window type interpretation number
*                            (0 or 1)
CCODE    BSS       1         Character code number (0 to 127)
KCODE    BSS       1         Character code number (0 to 127)
BBIT     BSS       1         Temporary store for bit position
LLONG    BSS       2         Temporary store for LENGTH & used in
*                            ROLLDN
COUNT    BSS       2         16-bit counter for iterations etc.
RBAPA    BSS       2         Current page base address
PBYTE    BSS       3         Tempory store for 3 bytes (graphics)
FOND     BSS       1         Background colour
COUNTA   BSS       1         g.p. counter
COUNTB   BSS       1         g.p. counter
ATTRIB   BSS       1         Parallel attributes for a character
ATTSER   BSS       1         Current serial attributes in text
                             mode
TEMP     BSS       2         g.p. store
*
************************************************************
*        End of RAM file definition                        *
*        Last Register = 45 ( >002D )                      *
************************************************************
Note: The BSS directive allocates the  specified  number
      of bytes within the ram area.
```

8.3 Equalities definition file

This file allocates names to constants used within the software modules. It makes the software easier to read and understand and enables all occurrences of a particular constant to be modified by changing only one line of code hence ensuring all occurrences are undisputedly changed at the next assembly.

```
*         Colour Equivalences
*
NOIR      EQU       0         * Black
BLEU      EQU       4         * Blue
ROUGE     EQU       1         * Red
VIOLET    EQU       5         * Magenta
VERT      EQU       2         * Green
CYAN      EQU       6         *
JAUNE     EQU       3         * Yellow
BLANC     EQU       7         * White
*
*         Memmory Size/Configuration
*
RAM64K    EQU       0         * 64K RAM
RAM16K    EQU       2         * 16K RAM
RAMROM    EQU       3         * Mixed RAM/ROM
*
NBCOL     EQU       4         * Number of Keyboard columns
*
*         I/O Port definitions for TMS 7000
*
IOCTRL     EQU       0          *  I/O  Port  to  control  Memory
expansion mode
*
SINGLE    EQU       0         * Single-Chip mode
FULLEX    EQU       >80       * Full Expansion mode
*
PORTA     EQU       4         * Input Lines
PORTB     EQU       6         * Output Lines
PORTC     EQU       8         * Adrress/Data AD7/AD0
CDIR      EQU       9         * PortC Direction control for each
                              * bit
*                             * 1=out, 0=in.
PORTD     EQU       >A        * PortD Data
DDIR      EQU       >B        * PortD Direction control for each
                              * bit
*
* VDP Access E1   = PORT B BIT 0
*            E2   = PORT B BIT 1
*            R/-W = PORT B BIT 2
*            DATA = PORT C
*
```

```
*       Programmable Timer/Event Counter Equivalences
*
TIMER   EQU     2         * Timer Data Register
TIMCTL  EQU     3         * Timer Control Register
DEBLOW  EQU     255       * Areg. value for delay in ENCODE
DEBHI   EQU     10        * Breg. value for delay in ENCODE
*
*       Equivalences used by PROVDP  - See 'INITAB' for details
*
ENTRLA  EQU    >70
STDRD1  EQU    >60
BT3     EQU    >50
BT4     EQU    >40
STITRE  EQU    >30
SYNTHE  EQU    >71
ROW00   EQU    >61
MASQUA  EQU    >51
INCRUS  EQU    >41
ALPHA4  EQU    >31
GRILLE  EQU    >21
LIGNAG  EQU    >11
SURIMP  EQU    >01
GRAPHI  EQU    >72
TEXTE   EQU    >62
VDPAGE  EQU    >52
PRICPU  EQU    >42
CHRMEM  EQU    >32
CT6     EQU    >22
CT7     EQU    >12
PASADR  EQU    >02
*
*       VDP Register numbers used to program Base Addresses
*
ADBAMT  EQU     8
ADBAMP  EQU     9
ADBAPA  EQU     >A
ADBAG0  EQU     >B
ADBAG1  EQU     >C
ADBAG2  EQU     >D
ADBAG3  EQU     >E
ADBATF  EQU     >F
*
***********************************************************
*       End of equalities file                            *
***********************************************************
```

8.4. VDP Handiers file

This file contains the software routines which control the communication between the CPU and the VDP. Its contents will depend upon the actual interface chosen between the two devices - single chip, semi memory mapped or full memory mapped.

Example software for each type of interface is given in the paragraphs that follow.

8.4.1 Single chip interface

The most cost effective interface to the TMS3556 is the "Single Chip" mode of the TMS7000 as described in paragraph 7.1.1. This type of interface is also valid for other microprocessors which have a "single chip" mode. The software will not be directly transferable to another microprocessor due to different internal archetectures, in particular the internal RAM structures and operating techniques; nevertheless the same sequence of events and control signal manipulation can be used.

As described in section 3, the CPU may make one of four types of access to the VDP. These are listed in Table 8.1 together with the names of the routines shown in this paragraph to effect each access operation.

TABLE 8.1: Routines accessing the VDP in single chip mode

Access type	Name	Parameters passed
Write to VDP register	AVDP	Data in A reg in B
Read VDP status register	STAT	Value returned in A
Write to external VDP RAM	ERAM	Value written in A
Read from external VDP RAM	LRAM	Value read in A

The source code for each of the routines listed in Table 8.1 appears below.

```
********************************************************************
* AVDP                                                             *
*        Access to VDP registers when in 'single chip'             *
*        Enter with register address in A and data in B            *
*        Called by TRAP 18                                         *
********************************************************************

AVDP    MOVP   %>FF,CDIR                  * PORT C to output mode
*                                         *        (if necessary)
        ANDP   %?11111101,PORTB           * E1 low for reg access
        MOVP   A,PORTC                    * Register address to Pt C
        ANDP   %?11111011,PORTB           * Strobe RWM low
        ORP    %?00000100,PORTB           * Return RWM high
        ORP    %?00000011,PORTB           * Return E1 and E2 high
AVDPA   ANDP   %?11111101,PORTB           * E1 low for reg access
        MOVP   B,PORTC                    * Data to Port C
        ANDP   %?11111011,PORTB           * Strobe RWM low
        ORP    %?00000100,PORTB           * Return RWM high
        ORP    %?00000011,PORTB           * Return E1 and E2 high
        RETS                              * Return to main program

**********************************************************
* LRAM                                                   *
*        Read one byte of data from VDP ram              *
*        Data read from RAM is returned in A             *
*        Called by TRAP 23(single-chip mode)             *
**********************************************************

LRAM    MOVP   %>00,CDIR                  * PORT C to input mode
*                                         *       (if necessary)
        ANDP   %?11111100,PORTB           * E1 & E2 low for RAM read
        MOVP   PORTC,A                    * Read data from PORT C
        ORP    %?00000011,PORTB           * Return E1 & E2  high
        RETS                              * Return to calling prog

**********************************************************
* ERAM                                                   *
*        Write one byte of data to VDP RAM               *
*        Data to be written is passed in A               *
*        Called by TRAP 17                               *
**********************************************************

ERAM    MOVP   %>FF,CDIR                  * PORT C to output mode
*                                         *        (if necessary)
        ANDP   %?11111110,PORTB           * E2 low for RAM write
ERAMA   MOVP   A,PORTC                    * Data written to PORT C
        ANDP   %?11111011,PORTB           * Strobe RWM low
        ORP    %?00000100,PORTB           * Return RWM high
        ORP    %?00000011,PORTB           * Return E2 high
        RETS                              * Return to calling prgm.
```

```
********************************************************
* STAT                                                 *
*         Read the VDP status register                 *
*         Value read is returned in A                  *
********************************************************
STAT    MOV    %>03,B                    * Addr. of status reg to B
        MOVP   %>FF,CDIR                 * PORT C to output mode
        CALL   @AVDPA                    * Set Pointer to Reg. 3
        MOVP   %>00,CDIR                 * PORT C to input mode
        ANDP   %?11111101,PORTB          * E1 low for reg access
        MOVP   PORTC,A                   * Read data from PORT C
        ORP    %?00000011,PORTB          * Return E1 and E2 high
        RETS
```

8.4.2 Semi memory mapped interface

The routines described in this paragraph are designed to operate with the semi memory mapped interface as described in paragraph 7.1.2.

For these routines, the TMS7000 must be in full expansion mode. The address 'VDPADR' is given a suitable value outside of the on chip program memory address range and is decoded from the external address bus if the system contains supplementary external memory.

```
*****************************************************************
* AVDP                                                          *
*         Access to VDP registers                               *
*         Enter with register address in A and data in B        *
*         Called by TRAP 18                                     *
*****************************************************************

AVDP    MOVP   %FULLEX,IOCTRL            * To full expansion mode
*                                        *        (if necessary)
        ANDP   %?11111101,PORTB          * E1 low for reg access
        STA    @VDPADR                   * Write register address
        ORP    %?00000011,PORTB          * Return E1 high
        XCHB   A                         * Swap data into A
AVDPA   ANDP   %?11111101,PORTB          * E1 low for reg access
        STA    @VDPADR                   * Write data to register
        ORP    %?00000011,PORTB          * Return E1 high
        RETS                             * Return to main program
```

```
************************************************************
* LRAM                                                     *
*         Read one byte of data from VDP ram               *
*         Data read from RAM is returned in A              *
*         Called by TRAP 23                                *
************************************************************

LRAM    MOVP   %FULLEX,IOCTRL                * To full expansion mode
*                                            *       (if necessary)
        ANDP   %?11111100,PORTB              * E1 & E2 low for RAM read
        LDA    @VDPADR                       * Read data into A
        ORP    %?00000011,PORTB              * Return E1 & E2  high
        RETS                                 * Return to calling prog

************************************************************
* ERAM                                                     *
*         Write one byte of data to VDP RAM                *
*         Data to be written is passed in A                *
*         Called by TRAP 17                                *
************************************************************

ERAM    MOVP   %FULLEX,IOCTRL                * To full expansion mode
*                                            *        (if necessary)
        ANDP   %?11111110,PORTB              * E2 low for RAM write
ERAMA   STA    @VDPADR                       * Write data to VDP
        ORP    %?00000011,PORTB              * Return E2 high
        RETS                                 * Return to calling prgm.

************************************************************
* STAT                                                     *
*         Read the VDP status register                     *
*         Value read is returned in A                      *
************************************************************
STAT    MOV    %>03,A                        * Addr. of status reg to A
        CALL   @AVDPA                        * Set Pointer to Reg. 3
        ANDP   %?11111101,PORTB              * E1 low for reg access
        LDA    @VDPADR                       * Read data from VDP
        ORP    %?00000011,PORTB              * Return E1 high
        RETS
```

8.4.3 Full memory mapped interface

The routines described in this paragraph are designed to operate with the full memory mapped interface as described in paragraph 7.1.3. Obviously, the routines are so short as to render it unecessary to define them as routines at all. The relavent instructions would typicaly be included in the main program where necessary. They are shown in this paragraph in subroutine form simply to demonstrate the functional similarity to their equivalent counterparts used with the other two interfaces.

The various addresses: 'VDPREG', 'VDPWR', 'VDPRD1' and 'VDPRD2' are defined to be different from each other and from any other address used in the system. Their values are chosen such that their presence on the address bus during the access can be detected by the external interface hardware, and cause the latter to set the correct states on the E1 and E2 control lines depending on the type of access.

For these routines, the TMS7000 must be in full expansion mode. The address 'VDPADR' is given a suitable value outside of the on chip program memory address range and is decoded from the external address bus if the system contains supplementary external memory.

```
******************************************************************
* AVDP                                                           *
*        Access to VDP registers                                 *
*        Enter with register address in A and data in B          *
*        Called by TRAP 18                                       *
******************************************************************

AVDP   MOVP  %FULLEX,IOCTRL            * To full expansion mode
*                                      *        (if necessary)
       STA   @VDPREG                   * Write register address
       XCHB  A                         * Swap data into A
AVDPA  STA   @VDPREG                   * Write data to register
       RETS                            * Return to main program

*********************************************************
* LRAM                                                  *
*        Read one byte of data from VDP ram             *
*        Data read from RAM is returned in A            *
*        Called by TRAP 23                              *
*********************************************************

LRAM   MOVP  %FULLEX,IOCTRL            * To full expansion mode
*                                      *       (if necessary)
       LDA   @VDPRD1                   * 1st read to start access
       LDA   @VDPRD2                   * 2nd read to get data
       RETS                            * Return to calling prog
```

```
********************************************************
* ERAM                                                 *
*        Write one byte of data to VDP RAM             *
*        Data to be written is passed in A             *
*        Called by TRAP 17                             *
********************************************************

ERAM   MOVP  %FULLEX,IOCTRL           * To full expansion mode
*                                     *        (if necessary)
ERAMA  STA   @VDPWR                   * Write data to VDP
       RETS                           * Return to calling prgm.

********************************************************
* STAT                                                 *
*        Read the VDP status register                  *
*        Value read is returned in A                   *
********************************************************

STAT   MOV   %>03,A                   * Addr. of status reg to A
       STA   @VDPREG                  * Set Pointer to Reg. 3
       LDA   @VDPREG                  * Read data from VDP
       RETS
```

8.5 General VDP control handlers

The routines listed in this section are used to perform various VDP related functions. They may not all be needed in a particular VDP application but are listed here as typical application subroutine examples.

All the routines listed below interface to the VDP via one or several of the CPU to VDP communications routines described in paragraph 8.4.

```
********************************************************
* PROVON                                               *
*        Program VDP Internal Registers with           *
*        desired feature                               *
* PROVX                                                *
*        Program VDP Internal Registers with           *
*        desired features - Called by TRAP 14          *
* PROVOF                                               *
*        Switch off feature in VDP internal registers  *
********************************************************
PROVON MOV   A,TEMP
       LDA   @VDPREG(B)
       OR    TEMP,A
PROVX  STA   @VDPREG(B)
       TRAP  18               *AVDP
       RETS
```

```
PROVOF INV   A
       MOV   A,TEMP
       LDA   @VDPREG(B)
       AND   TEMP,A
       JMP   PROVX

********************************************************
* COLRAN                                               *
*        Store the address in INDADR,INDADR+1 in       *
*        VDP registers COL & RAN                       *
*        Called by TRAP 22                             *
********************************************************
COLRAN MOV   %2,B
       MOV   INDADR,A         *INDADR to VDP reg.2
       TRAP  18               *AVDP
       MOV   %1,B
       MOV   INDADR+1,A
       TRAP  18               *AVDP
       RETS

********************************************************
* BASADR                                               *
*        Program Base Addresses into VDP internal      *
*        registers 8 to F.                             *
********************************************************
BASADR PUSH  B                *Save VDP Reg. number
       TRAP  22       *CLRN   *Store INDADR+1 in VDP reg.1
*                             *Store INDADR in VDP reg.2
       POP   B
       MOV   B,A
       TRAP  18       *AVDP   *Double access to VDP to move
*                             *reg.s 1&2 to register number
       RETS                   *contained in Areg.

********************************************************
* ACMP                                                 *
*        Set ACMP pointer via VDP reg 9 - BAMP         *
*        Called by TRAP 20                             *
*        Does not switch mode (single-chip to full ex.)*
********************************************************
ACMP   MOV   %9,B             *VDP register 9
       MOV   B,A              *VDP register 9
       TRAP  18
       RETS

********************************************************
* TPSCRS (TRAP 15)                                     *
*        Position read or write pointer in VDPram      *
********************************************************
TPSCRS TRAP  11               *TPGADR
       TRAP  22               *COLRAN
```

```
*********************************************************
* INIT                                                  *
*        Initialisation of the VDP Mode.                *
*        Initialisation of the VDP Internal Registers   *
*********************************************************
*
INIT    MOV    %3,B              *1st VDP register to be proggrammed
NXVDP   LDA    @INTCM1(B)        *INTCM1=Start of table of values
        PUSH   B
        ADD    %4,B
        CALL   @PROVX            *Sub. to program VDP registers
        POP    B
        DEC    B
        JC     NXVDP
*
*         Program VDP Base Addresses
*
        MOV    %15,B             * 8 registers to program
        MOV    B,COUNTA          * COUNTA = register to program
NXBAS   LDA    @BASDAT(B)        *Get LSByte of Address
        MOV    A,INDADR+1        *Store LSByte in INDADR+1
        DEC    B
        LDA    @BASDAT(B)        *Get MSByte of Address
        MOV    A,INDADR          *Store MSByte in INDADR
        PUSH   B                 *
        MOV    COUNTA,B          *
        CALL   @BASADR
        DEC    COUNTA
        POP    B                 *
        DEC    B                 *
        JC     NXBAS             *Jump if carry, i.e. more Addresses
*
        MOV    %BAPALO,RBAPA+1
        MOV    %BAPAHI,RBAPA     *Store Page Base Address in MLP RAM
*
        RETS

*********************************************************
* GENCAR                                                *
*         Program Character Generators into VDPram      *
*       Base address in VDPram should be in TEMP,TEMP+1 *
*       Starting address in ROM of generator should be  *
*       in A4,A5.                                       *
*       HEX value of 1st character should be in CCODE   *
*********************************************************
GENCAR MOVD   TEMP+1,INDADR+1   *Base address in VDPram (=BAGCx+2)
       TRAP   22                *COLRAN
       MOVD   %(128*10)-1,A1    *A0,A1 = number of bytes in
*                               *generator
       CLR    A
NXCLR0 TRAP   17                *A = data to be written
       DECD   A1                *decrement loop count
       JC     NXCLR0            *Clear all lines of GENE0 in VDP
                                * ram
        ADD   %128,TEMP+1      *start at line 8
        ADC   %0,TEMP          *
        MOVD  TEMP+1,INDADR+1  *TEMP+1 = BASE address in VDP ram
        ADD   CCODE,INDADR+1   *start at character CCODE
        ADC   %0,INDADR        *
        MOV   %128,COUNT       *Max number of characters = 128
        SUB   CCODE,COUNT      *CCODE = 1st character in generator
        MOV   %7,COUNTA        *COUNTA=number of lines per
                               * character
GENELP  ADD   %128,INDADR+1    *Point to next line
        ADC   %0,INDADR        *
        MOVD  A5,A1            *A0,A1 = ROM address of start of
                               *line
        ADD   COUNT,A5         *A4,A5 = ROM address of end of line
        ADC   %0,A4            *
        CALL  @ROMVDP          *Recopy GEN0 into VDPram
        DJNZ  COUNTA,GENELP    *Loop until COUNTA = 0
*
        MOV   %2,COUNTA        *COUNTA = number of bytes to write
        MOVP  %FULLEX,IOCTRL
        LDA   *A5              *Read byte from ROM
        MOVP  %SINGLE,IOCTRL
        CALL  @NXLIN           *
*
        MOV   %1,COUNTA        *COUNTA = no. of bytes per
                               *character
        ADD   %4,TEMP          *TEMP = 9th line of gene
        CALL  @NXLIN           *
        RETS
*
NXLIN   MOV   A,COUNT          *COUNT = number of characters
NXLIN1  MOV   COUNTA,COUNTB    *
        MOVD  TEMP+1,INDADR+1  *TEMP,TEMP+1=BASE address of char
                               *gen
        CALL  @LNREAD          *Read next byte from ROM
        ADD   A,INDADR+1       *Set pointer register for line 7 of
        ADC   %0,INDADR        *this character
NXLIN2  TRAP  22               *COLRAN - set pointer in VDP ram
        CALL  @LNREAD          *Read next byte from ROM
        TRAP  17               *A = data to write to VDP ram
        SUB   %128,INDADR+1    *Set pointer register to next line
        SBB   %0,INDADR        *
        DJNZ  COUNTB,NXLIN2    *Loop until COUNTB = 0
        DJNZ  COUNT,NXLIN1     *Loop until COUNT = 0
        CALL  @LNREAD          *Read next byte from ROM
        DJNZ  COUNTA,NXLIN     *Loop until COUNTA = 0
        RETS
*
LNREAD  INC   A5               *Increment ROM address
        ADC   %0,A4            *
        MOVP  %FULLEX,IOCTRL
        LDA   *A5              *Read byte from ROM
        MOVP  %SINGLE,IOCTRL
        RETS
```

```
*************************************************************
* GENE2                                                     *
*         Generate GENE2 - Block Graphics                   *
*************************************************************
GENE2  MOVD  %BAGC2+2,INDADR+1
       TRAP  22               *COLRAN
       MOV   %>FF,A
       MOVD  %1280,A1
GENCLR TRAP  17
       DECD  A1
       JC    GENCLR            *Write >FF to all GENE2
       MOV   %>A,LIGNE
       MOV   %>20,COUNTB
NXLINE CLR   B
NXCOLL ADD   COUNTB,INDADR+1
       ADC   %0,INDADR
       PUSH  B
       TRAP  22                *COLRAN
       POP   B
NXCHAR CLR   A
       CMP   %>8,LIGNE
       JNC   INF8
       BTJZ  %?00100000,B,SUL10 *Lines 10 to 8
       ADD   %>0F,A
SUL10  BTJZ  %?00010000,B,INCCAR
       JMP   INCCRX
INF8   CMP   %>4,LIGNE
       JNC   INF4
       BTJZ  %?00001000,B,SUL8 *Lines 7 to 4
       ADD   %>0F,A
SUL8   BTJZ  %?00000100,B,INCCAR
       JMP   INCCRX
INF4   BTJZ  %?00000010,B,SUL4 *Lines 3 to 1
       ADD   %>0F,A
SUL4   BTJZ  %?00000001,B,INCCAR
INCCRX ADD   %>F0,A
INCCAR TRAP  17               * ERAM
       INC   B                *Next character
       CMP   %>20,B
       JEQ   INCADD
       CMP   %>40,B
       JNE   NXCHAR
       MOV   %>40,COUNTB
       DJNZ  LIGNE,NXLINE     *Preceding line
```

```
        MOVD  %BAGC2+2,INDADR+1
        MOV   %>7F,COUNTB
        MOV   %10,COUNTA
CHAR5F  ADD   COUNTB,INDADR+1
        ADC   %0,INDADR
        TRAP  22                  *COLRAN
        CLR   A
        TRAP  17                  *ERAM
        MOV   %>80,COUNTB
        DJNZ  COUNTA,CHAR5F       *Write >00 to all rows of char. 7F
        RETS

INCADD  MOV   %>40,COUNTB
        JMP   NXCOLL
*
************************************************************
*COLOUR                                                    *
*       Program character colour in ATTRIB                 *
************************************************************
COLOUR  MOV   TEINTE,A
        SWAP  A
        RL    A
        AND   %>1F,ATTRIB
        OR    A,ATTRIB            *F4,F3,F2 = B,G,R
        RETS

************************************************************
* MRGCLR                                                   *
*       Program border colour                              *
************************************************************
MRGCLR  MOV   TEINTE,A            *TEINTE = colour
        SWAP  A                   *
        RL    A                   * A = A*32
        PUSH  A                   *save A
        MOV   %CM4,B              *B = 7 = VDPreg CM4
        TRAP  6                   *PROVON
        POP   A                   *restore A
        INV   A                   *invert A
        AND   %>E0,A              *set unwanted bits to 0
        MOV        %CM4,B         *B = 7 = VDPreg CM4
        TRAP       5              *PROVOF
        RETS

************************************************************
* TPGADR (TRAP 11)                                         *
*       Calculate address in VDP ram                       *
************************************************************
TPGADR  CLR   INDADR
        MOV   RAN,INDADR+1        *Address = RAN
        PUSH  A
        PUSH  B
        MPY   %80,INDADR+1        * = 80*RAN
        MOVD  B,INDADR+1
        ADD    RBAPA+1,INDADR+1
        ADC    RBAPA,INDADR        * = BAPA + 80*RAN
        MOV    COL,A
        RL     A
        ADD    A,INDADR+1
        ADC    %>00,INDADR         * = BAPA + 80*RAN + 2*COL
        POP    B
        POP    A
        RETS

*********************************************************
* TPXYE (TRAP 10)                                       *
*        Position ACMPxy in write mode at current       *
*        cursor position                                *
*********************************************************
*
TPXYE   DEC    INDADR+1
        SBB    %0,INDADR            *Backspace pointer address
        TRAP   22                   *Reposition ACMPxy
        TRAP   23                   *Read 1 byte
        PUSH   A
        TRAP   22                   *Read 1 byte
        POP    A                    *A=data to be written
        TRAP   17                   *Write 1 byte
        INC    INDADR+1
        ADC    %0,INDADR            *Forwardspace pointer address
        RETS

*********************************************************
* TWRPAG (TRAP 12)                                      *
*        Write a character at the current writing       *
*        position                                       *
*********************************************************
TWRPAG MOV    ATTRIB,A             *character attribute
       TRAP   17                   *ERAM
       MOV    CCODE,A              *character code
       TRAP   17                   *ERAM
       RETS

*********************************************************
* TRDPAG (TRAP 13)                                      *
*        Read a character code and its attribute at     *
*        the current read pointer                       *
*********************************************************
TRDPAG TRAP   23                   *LRAM
       MOV    A,ATTRIB
       TRAP   23                   *LRAM
       MOV    A,CCODE
       RETS
```

```
**********************************************************
* UPDATE                                                 *
*        Position the cursor at the next character       *
**********************************************************
UPDATE INC   COL                * Update COL,RAN:
       CMP   %40,COL
       JEQ   OVFH
       RETS
OVFH   CLR   COL                *COL=40-->COL=0 and RAN+1
       INC   RAN
       CMP   %25,RAN
       JNE   NOOVF
       MOV   %1,RAN             *RAN=25-->RAN=1
NOOVF  TRAP  15                 *TPSCRS acmpxy=bapa
       RETS

**********************************************************
* TCLERO                                                 *
*        Fill from the current writing position to the  *
*        end of the row with the same character          *
**********************************************************
TCLERO CALL  @TWRPAG
       CALL  @UPDATE
       CMP   %0,COL
       JNE   TCLERO
       RETS

**********************************************************
* TCLEPG                                                 *
*        Fill from the current writing position to the  *
*        screen with the same character                  *
**********************************************************
TCLEPG CMP   %24,RAN
       JEQ   LASTLN
       CALL  @TCLERO
       JMP   TCLEPG
LASTLN CALL  @TCLERO
       RETS

**********************************************************
* TCLPAG                                                 *
*        Fill the whole screen with the same character  *
**********************************************************
TCLPAG CLR   RAN
       CLR   COL
       TRAP  15                 *TPSCRS
       CALL  @TCLEPG
       RETS
```

```
*************************************************************
* TCLEAR                                                    *
*        Clear screen - fill with delimiters                *
*************************************************************
TCLEAR MOV   FOND,ATTRIB
       SWAP  ATTRIB
       RL    ATTRIB
       MOV   %>20,CCODE
       CLR   RAN
       CLR   COL
       MOVD  RBAPA+1,INDADR+1
       TRAP  22
       MOVD  %2050,A1
       CLR   A
TCL1   TRAP  17
       DECD  A1
       JC    TCL1
       RETS

*************************************************************
* ROMVDP                                                    *
*        Copy a block of memory from ROM in MLP             *
*        to VDP RAM                                         *
*************************************************************
ROMVDP TRAP  22                *CLRAN-Set pointer ACMP
NXMOV  MOVP  %FULLEX,IOCTRL    *Change to FULL-EXPANSION
       LDA   *A1               *A1 contains start address of data
       MOVP  %SINGLE,IOCTRL    *Change to SINGLE-CHIP
       TRAP  17                *Write Data byte to VDP RAM
       INC   A1
       JNC   LOW
       INC   A0
LOW    CMP   A0,A4             *A4 contains END address MSB
       JNE   NXMOV
       CMP   A1,A5             *A5 contains END address LSB
       JNE   NXMOV
       RETS

*************************************************************
* VDPVDP (ROLL-UP)                                          *
*        Move a block of data within the VDP ram            *
*        Address of start of original in A0 & A1            *
*        Address of start of copy in INDADR & INDADR+1      *
*        Address of end of original in A4 & A5              *
*        A0,A1 must be > INDADR,INDADR+1                    *
*        This routine could also be used to Roll-down       *
*        if INDADR,INDADR+1 > A4,A5                         *
*************************************************************
VDPVDP PUSH  INDADR+1
       PUSH  INDADR
       SUB   A1,A5
       SBB   A0,A4             *A4,A5 contain no. of bytes-1
       MOVD  A1,INDADR+1
       TRAP  22                *COLRAN
```

```
        TRAP   23               *ACMPxy=READ CURSOR
        POP    INDADR           *Restore 'write' start address
        POP    INDADR+1
        PUSH   A                *Put 1st byte back on stack
        DECD   INDADR+1         *(BAMP incremented when loaded)
        TRAP   22               *Load COL & RAN with write pointer
        TRAP   20               *Set ACMP to 'write'
        POP    A
        JMP    NXMOV1           *jump to write 1st byte
NXMOVU  TRAP   23               *LRAM - reads byte onto stack
NXMOV1  TRAP   17               *ERAM - writes value already on
                                *stack
        DECD   A5
        JC     NXMOVU
        INC    INDADR+1
        ADC    %0,INDADR        *Restore original value of INDADR+1
        RETS
```

```
********************************************************
* ROLLDN                                               *
*        Move a block of data within the VDP ram       *
*        Address of start of original in A0 & A1       *
*        Address of start of copy in INDADR & INDADR+1 *
*        Address of end of original in A4 & A5         *
*        A0,A1 must be < INDADR,INDADR+1               *
********************************************************
ROLLDN PUSH  INDADR+1
       PUSH  INDADR
       PUSH  A1
       PUSH  A0
       PUSH  A5
       PUSH  A4
       SUB   A1,A5            *Calculate no. of bytes -1
       SBB   A0,A4            *to be moved
       MOVD  A5,LLONG+1       *LLONG+1 = byte count-1
       MOVD  A5,A1            *A1,A0  = byte count-1
       POP   A4               *Restore 'read' start address
       POP   A5               *Restore 'read' start address
       ADD   INDADR+1,LLONG+1 *LLONG becomes 'write' start
                              *address
       ADC   INDADR,LLONG
MOVBYT MOVD  A5,INDADR+1
       TRAP  22               *COLRAN-position 'read' pointer
       TRAP  23               *read one byte
       PUSH  A
       MOVD  LLONG+1,INDADR+1
       TRAP  22               *COLRAN-positin 'write' pointer
       POP   A
       TRAP  17               *write one byte
       DECD  A5
       DECD  LLONG+1
       DECD  A1               *decrement byte count
       JC    MOVBYT
       POP   A0               *Restore original values
       POP   A1
       POP   INDADR
       POP   INDADR+1
       RETS
```

```
**********************************************************
* VDPRAM                                                 *
*         Copy a Block of data from VDPRAM to MLPRAM     *
*         This routine is effectively replaced by        *
*         Custom Micro-Instruction LRAM                  *
*         This routine is useful to move data to         *
*         registers other than the stack                 *
**********************************************************
*         Address of 1st byte in VDPRAM should be        *
*         Stored in INDADR+1.                            *
*         Address of 1st register in MLPRAM to receive   *
*         the data should be stored in B reg.            *
*         Address of last register should be in A5       *
**********************************************************
VDPRAM TRAP   22                    *COLRAN-Set ACMPxy Read Pointer
       SUB    B,A5                  *calculate no. of bytes to read
       INC    A5                    *no. = start address also
NXMOV3 TRAP   23                    *Read into Areg.
       STA    @0(B)                 *Store data in RAM reg
       INC    B
       DJNZ   A5,NXMOV3             *Test if last data read
*                                   *More data to read, jump to NXMOV3
       RETS

**********************************************************
* RAMVDP                                                 *
*         Copy a Block of Memory From MLP RAM to VDPRAM  *
*         This routine is effectively replaced by the    *
*         Custom Micro-Instruction ERAM                  *
*         This routine is useful to transfer data from   *
*         registers other than the stack                 *
**********************************************************
*         Store address of 1st register to copy in Breg  *
*         Store address of Last register in A5           *
**********************************************************
RAMVDP TRAP   22                    *COLRAN-Set Write Pointer ACMPxy
       SUB    B,A5                  *Calculate no. of bytes to move
       INC    A5                    *inclusive of limits
NXMOV4 LDA    @0(B)                 *0=regA, B holds 1st register to
                                    *copy
       TRAP   17                    *ERAM
       INC    B
       DJNZ   A5,NXMOV4             *test for last byte read
       RETS
```

```
**************************************************************
* RAMINF                                                     *
*         Copy a block of data within the MLP RAM            *
*         Start Address for data to be copied is             *
*         contained in register A1.                          *
*         Start Address for copied data is in INDADR+1       *
*         End Address for data to be copied is in A5         *
*         This routine is used when Start Address of         *
*         original is greater than Start Address of          *
*         Copied data                                        *
**************************************************************
RAMINF CLR    A0                *MSB address in RAM = 0
       CLR    INDADR            *MSB address in RAM = 0
NXMOV5 LDA    *A1               *Get 1st byte of data at RAM adr A1
       STA    *INDADR+1         *Store data in RAM add. given by
                                *INDADR
       INC    INDADR+1          *Increment the REGISTER no for
                                *received
       INC    A1                *Increment the REGISTER no for
                                *sending
       CMP    A1,A5             *Test if last data moved
       JC     NXMOV5            *If not, jump to NXMOV5
       RETS

**************************************************************
* RAMSUP                                                     *
*         Used to copy data within the MLP RAM when          *
*         the Start Address of the Copied data is            *
*         greater than the Start Address of the original*
**************************************************************
RAMSUP CLR    A4                *MSB address in RAM = 0
       CLR    INDADR            *MSB address in RAM = 0
       PUSH   A5                *save end address
       SUB    A1,A5             *calculate no. of bytes-1
       ADD    A5,INDADR+1       *calc. end address of copy
       MOV    A5,A1
       POP    A5                *move no. of bytes-1 to A1
       INC    A1                *A1=no. of bytes
NXMOV6 LDA    *A5               *Get 1st byte of data at RAM add.A5
       STA    *INDADR+1         *Store data in RAM add. given by
                                *INDADR
       DEC    INDADR+1          *Decrement Address for next write
                                *data
       DEC    A5                *Decrement Address for next read
                                *data
       DJNZ   A1,NXMOV6         *Test for last data copied
       RETS
```

```
************************************************************
* DELAI                                                    *
*         Routine to give programmable delay in the        *
*         range 49uSec to 25MSec                           *
************************************************************
*         Enter routine with value >00 to >FF in Areg      *
*         Enter routine with value >00 to >1F in Breg      *
*         Delay = [3.2 * A * (B mod32)] + 39.6uSec         *
************************************************************
DELAI  PUSH  ST               *Save STATUS reg contents
       DINT                   *Disable Interrupts
       MOVP  A,TIMER          *Preset Timer Latch
       AND   %>1F,B           *Ensure B is not greater than 32
       OR    %>80,B           *Set Start Control bit
       MOVP  B,TIMCTL         *Loab B value to Prescale & start
                              *Timer
NXTIME MOVP  TIMER,A          *Read current Timer value
       JNE   NXTIME           *Test Status for INT2 active
       MOVP  %>00,TIMCTL      *INT2 received, Stop Timer
       POP   ST               *Restore STATUS
       RETS

************************************************************
*         This program is the Master Handler for use       *
*         with applications using the DATA-PROVIDER        *
************************************************************
* MODETV                                                   *
*         Program the full screen TV mode in the VDP       *
************************************************************
*
MODETV MOVP  %>F,CDIR
       MOVP  %?1110,PORTC
       TRAP  4                *LATCEN
       MOV   %BT4,A           *Program BT4=1
       MOV   %CM1,B
       TRAP  6
       MOV   %SYNTHE,A        *Program VDP for full-screen
       MOV   %CM2,B
       TRAP  5                 analogue image
       MOV   %ENTRLA,A        *Non-interlaced frame
       MOV   %CM1,B
       TRAP  5
       MOV   %INCRUS,A        *Supression of insertion
       MOV   %CM2,B
       TRAP  5
       MOVD  %BAPA,INDADR+1
       MOV   %>A,B
       CALL  @BASADR          *Reprogram page base address
       RETS
```

```
**************************************************************
* MODEST                                                     *
*         Program ANTIOPE mode in VDP                        *
**************************************************************
MODEST MOVP  %>F,CDIR          *Set PORT C direction to 'output'
       MOVP  %?0110,PORTC      *Commutation Lente = Active
       TRAP  4                 *Store value in Addressable Latch
       MOV   %BT4,A            *BT4=0
       MOV   %CM1,B            *BT4=0
       TRAP  5                 *Program BT4 OFF in VDP
       MOV   %SYNTHE,A         *DC1=1
       MOV   %CM2,B            *
       TRAP  6                 *Program DC1 ON in VDP
       MOV   %ENTRLA,A         *BT1=0   i.e. Non-Interlaced
       MOV   %CM1,B            *BT4=0
       TRAP  5                 *Program BT1 OFF in VDP
       MOV   %ROW00,A          *Enable ROW 0 display
       MOV   %CM2,B            *
       TRAP  6                 *Program ROW 0 ON in VDP
       MOVD  %BAPA,INDADR+1
       MOV   %>A,B
       CALL  @BASADR           *Reprogram page base address
       RETS

**************************************************************
*         End of 'HANDLER'                                   *
**************************************************************
```

8.6 TRAP Definitions

```
**********************************************************
*      Trap Vectors 4 to 23 are used by the 'Handlers' *
*      to call frequently used subroutines, thereby    *
*      saving memory space.                            *
**********************************************************
*      Any TRAPs not defined in either the HANDLERS or *
*      application programme are assigned the same     *
*      address as the RESET vector i.e. ROMBEG         *
**********************************************************
*                                                      *
       AORG  >FFD0
       DATA  LRAM              * TRAP 23
       DATA  COLRAN            * TRAP 22
       DATA  ROMBEG            * TRAP 21 <-- Available for
       DATA  ACMP              * TRAP 20     application program
       DATA  ROMBEG            * TRAP 19 <--
       DATA  AVDP              * TRAP 18
       DATA  ERAM              * TRAP 17
       DATA  ROMBEG            * TRAP 16 <--
       DATA  TPSCRS            * TRAP 15
       DATA  PROVX             * TRAP 14
       DATA  TRDPAG            * TRAP 13
       DATA  TWRPAG            * TRAP 12
       DATA  TPGADR            * TRAP 11
       DATA  ROMBEG            * TRAP 10 <--
       DATA  ROMBEG            * TRAP  9 <--
       DATA  ROMBEG            * TRAP  8 <--
       DATA  ROMBEG            * TRAP  7 <--
       DATA  ROMBEG            * TRAP  6 <--
       DATA  ROMBEG            * TRAP  5 <--
       DATA  ROMBEG            * TRAP  4 <--
**********************************************************
*        INTERRUPT VECTORS                             *
*        Can be called under normal S/W control or     *
*        activated by external interrupts, but not     *
*        both  ways in the same program since an       *
*        external hardware interrupt routine must be   *
*        terminated by an RETI instruction, whereas    *
*        a S/W Trap is terminated by RETS instr.       *
**********************************************************
       DATA  (INT 3 vector)  * TRAP  3
**********************************************************
*        TRAP 2 is the TIMER rollover interrupt        *
**********************************************************
       DATA  (INT 2 vector)  * TRAP  2
**********************************************************
       DATA  (INT 1 vector)  * TRAP  1
**********************************************************
*        TRAP 0 is the RESET VECTOR                    *
**********************************************************
RESET  DATA  ROMBEG            * TRAP  0
       END
```

9 ELECTRICAL SPECIFICATION

9.1 ABSOLUTE MAXIMUM RATINGS

Supply voltage Vgg	-0.3 to 10 V
Supply voltage Vdd	-0.3 to 10 V
All input voltages	-0.3 to 10 V
Continuous power dissipation	1 W
Operating free air temperature range	0 to 60 C
Storage temperature range	-55 to +140 C
Maximum short circuit output current	mA

All voltages are with respect to Vss

Stresses beyond those listed under 'Absolute Maximum Ratings' may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the 'Recommended Operating Conditions' section of this users guide is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

9.2 RECOMMENDED OPERATING CONDITIONS

PARAMETER	MIN	NOM	MAX	UNIT
Supply voltage Vgg	4.75	5	5.25	V
Supply voltage Vdd	2.85	3	3.15	V
Supply voltage Vss		0		V
High-level input voltage Vih (all inputs except OBE and ODE)	2.2		Vgg	V
High-level input voltage (OBE and ODE inputs)	2.7		Vgg	V
Low level input voltage Vil (all inputs except OBE and ODE)	0		0.8	V
Low level input voltage (OBE and ODE inputs)			0.6	V
Low level output current (SCM, SLL & SCT outputs)				mA
Operating free air temp. Ta	0		60	C

9.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (UNLESS OTHERWISE NOTED)

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
Voh	High-level Output voltage	RAS,CAS,OE WR,D(0-7)	Ioh= 1.5mA Cl=50pF	2.4			V
		MP(0-7) HIZ	Ioh= 500uA Cl= 50pF	2.4			V
		READY	Ioh= 250uA Cl= 50pF	2.4			V
		R,G,B,I	Ioh= 500uA Cl= 50pF	2.4			V
Vol	Low level Output voltage	RAS,CAS,OE WR,D(0-7)	Iol= 3.5mA Cl= 50pF			0.6	V
		MP(0-7) HIZ	Iol= 1.0mA Cl= 50pF			0.6	V
		READY	Iol= 500uA Cl= 50pF			0.6	V
		R,G,B,I	Iol= 2.0mA Cl= 50pF			0.6	V
		SCM,SCT, SLL	Iol= 2.0mA Cl= 50pF			0.6	V
Iih	High level input current	D(0-7)	Vi = 5.25V all other pins at 0V			200	uA
		MP(0-7)				100	uA
		All others				20	uA
Iil	Low level input current	D(0-7)	Vi = 0V all other pins at 5.25V			-200	uA
		MP(0-7)				-100	uA
		All others				-20	uA
Iozh	Off-state output current, high level voltage applied	D(0-7)	Vo= 5.25V			100	uA
		MP(0-7)	Vo= 5.25V			200	uA

Iozl	Off-state output current, low level voltage applied	D(0-7)	Vo= 0.4V		-100	uA
		MP(0-7)	Vo= 0.4V		-200	uA
Igg	Supply current from Vgg		Vgg = MAX	105	140	mA
Idd	Supply current from Vdd		Vdd = MAX	70	100	mA

Ci	Input capacitance	MP(0-7)
		D(0-7)
			f=1 MHz		15	pF

		All others	f=1 MHz		10	pF

Co	Output capacitance	MP(0-7)
		D(0-7)
			f=1 MHz		15	pF

		All others	f=1 MHz		10	pF

all typical values are at Vgg = 5V, Vdd=3V, Ta=25 C

NOTE: All voltages are with respect to Vss unless otherwise noted

9.4 TIMING REQUIREMENTS OVER FULL RANGES OF RECOMMENDED OPERATING CONDITIONS

PARAMETER		MIN	TYP	MAX	UNIT
tc(ODE)	DMA oscillator cycle time				ns
tc(OBE)	Time base oscillator cycle time	110			ns
tc(E)	E1,E2 cycle time	See para. 3.7.7			
tc(HMP)	HMP cycle time	See para. 3.7.7			
td(RDHEL)	delay time RDY high to E1,E2 low	0			ns
tsu(ERWL)	E1,E2 set up time before RWM low	10			ns
th(RWHE)	E1,E2 hold time after RWM high	10			ns
twh(E)	E1,E2 pulse width E1 and E2 high	110			ns
twl(RWM)	pulse width, RWM low	110			ns
tsu(MPRWL)	data set up time before RWM low	5			ns
th(RWLMP)	data hold time after RWM low	40			ns
tsu(ELODL)	set up time, E2 low to ODE low~				ns
tsu(RWHODL)	set up time, RWM high to ODE low~				ns
tsu(DCH)	data set up time before CAS high				ns
th(CHD)	data hold time after CAS high	0			ns
tdis(CHD)	data disable time after CAS high			35	ns
twl(HMP)	pulse width HMP low	220			ns

NOTE: Timing measurements are made at the 10% and 90% points of input and clock transitions. In addition, Vil max and Vih min must be met at the 10% and 90% points.

~ This is a system timing parameter only and need not be respected in all cases; see paragraph 3.7.7.

9.5 SWITCHING CHARACTERISTICS OVER FULL RANGES OF RECOMMENDED OPERATING CONDITIONS (Independent of DMA oscillator frequency)

PARAMETER		MIN	TYP	MAX	UNIT
tac(ST)	status register access from E1 low			140	ns
ten(ELMP)	MP bus enable time after E1,E2 low	40			ns
tsu(MPRDH)	data set up time before RDY high	5			ns
th(EHMP)	data hold time after E1,E2 high	50			ns
td(ELRDL)	delay time, E2 low to RDY low	45		90	ns
td(RWHRDL)	delay time, RWM high to RDY low	45		90	ns
tdis(CLD)	disable time, CAS low to Dn high z				ns
th(CLCA)	column address hold time after CAS low	15			ns
ten(HZHD)	enable time, HIZ high to Dn driven	20			ns
td(CHHZH)	delay time, CAS high to HIZ high			15	ns
td(DMP)	delay time, Dn valid to MPn valid				ns
td(ODLRL)	delay time ODE low to RAS low	0			ns
td(OBHIH)	delay time OBE high to I high	0			ns
td(OBLIL)	delay time OBE low to I low	0			ns
td(OBLRGB)	delay time OBE low to R,G,B valid				ns
th(OBLRGB)	R,G,B hold time after OBE low				ns
td(OBLSL)	delay time OBE low to SLL low	0		90	ns

~ See paragraph x.x.x.

9.6 SWITCHING CHARACTERISTICS OVER FULL RANGES OF RECOMMENDED OPERATING CONDITIONS (Dependant on DMA oscillator frequency)

PARAMETER		DMA CLOCK CYCLES		ELECTRICAL VARIATION			UNIT
		OPT1	OPT2	MIN	TYP	MAX	
tc(W)	single write cycle time	5	7		0		ns
tc(RD)	single read cycle time	5	7		0		ns
tc(P)	CAS read double cycle time	3	5		0		ns
tc(D)	RAS read double cycle time	8	12		0		ns
tw(CH)	pulse width, CAS high (double cycle)	1	1	-14			ns
tw(CL)	pulse width, CAS low	2	4	-11			ns
tw(RH)	pulse width, RAS high	2	2	-21			ns
tw(RL)	pulse width, RAS low	3	5	-15			ns
tw(W)	pulse width, WR low	1	3	-10			ns
tw(OE)	pulse width, OE low	1	3	-10			ns
tsu(CA)	column address set up time	0.5	0.5	-27			ns
tsu(RA)	row address set up time	1	1	-45			ns
tsu(D)	data set up time before WR low	0.5	0.5	-20			ns
tsu(WRH)	WR set up time before RAS high	1	3	-15			ns
tsu(WCH)	WR set up time before CAS high	1	3	-15			ns
tsu(ORH)	OE set up time before RAS high	1	3	-15			ns
tsu(OCH)	OE set up time before CAS high	1	3	-15			ns
th(RA)	row address hold time after RAS low	0.5	0.5	0			ns
th(WLD)	data hold time after WR low	1	3	30			ns
td(RLCL)	delay time, RAS low to CAS low	1	1	-10			ns
td(CLWL)	delay time, CAS low to WR low	1	1	-15			ns
td(HZLCL)	delay time, HIZ low to CAS low	2	2	15			ns
tac(E)	read access time from E2 low			See para. 3.7.7			
td(ELRL)	delay time, E2 low to RAS low			See para. 3.7.7			
td(RWHRL)	delay time, RWM high to RAS low			See para. 3.7.7			
td(HPLHZL)	delay time, HMP low to HIZ low			See para. 3.7.7			

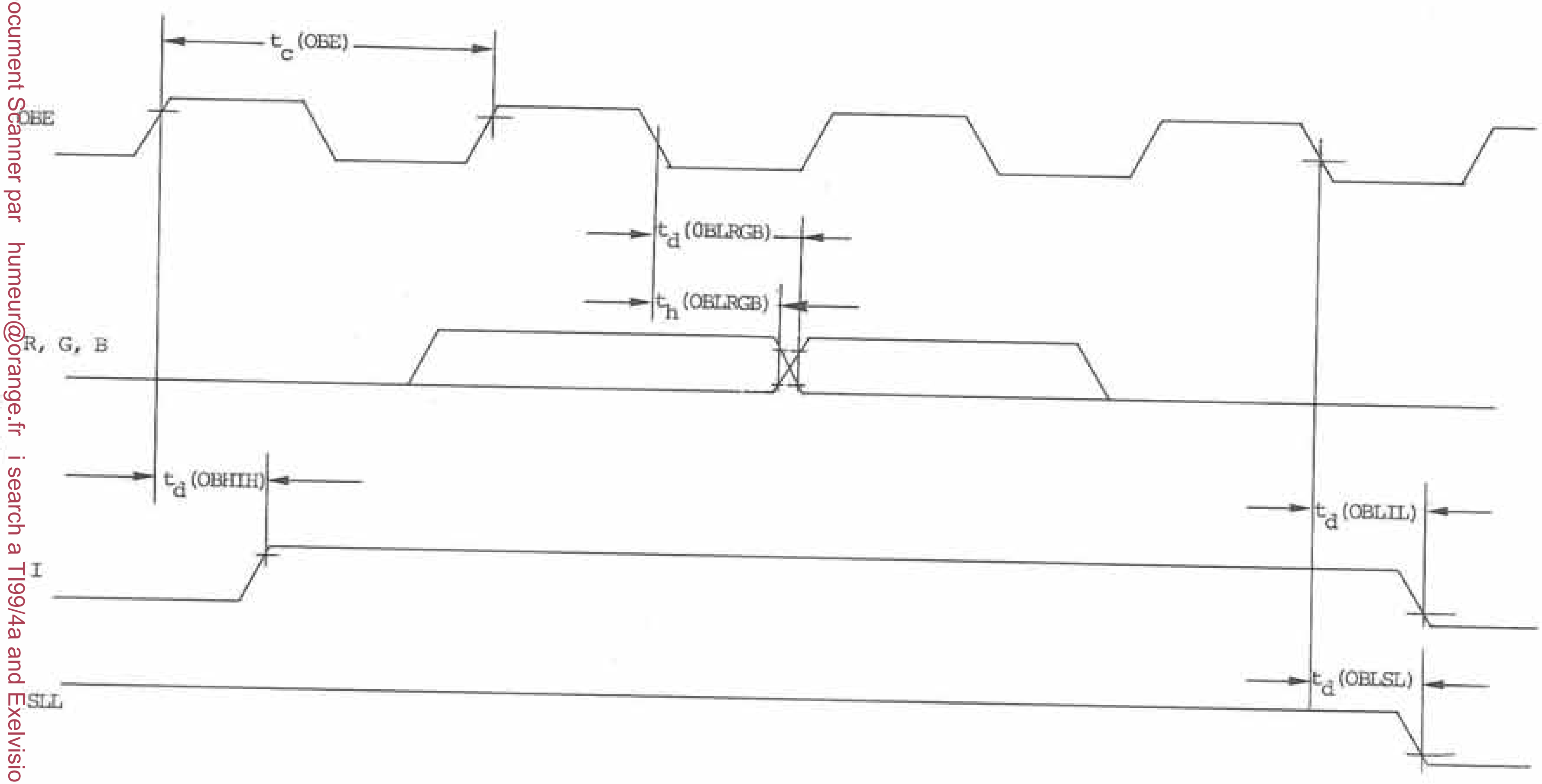
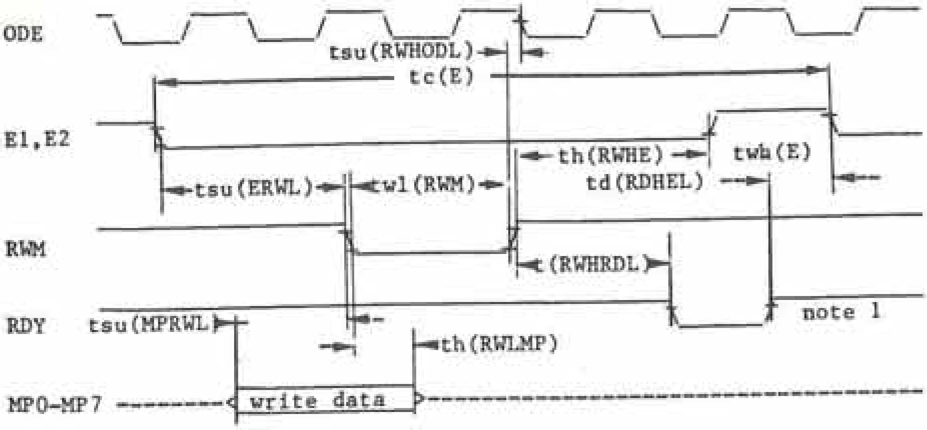


FIGURE 9.1 : R, G, B & I SWITCHING CHARACTERISTICS

FIGURE 9.2: CPU - VDP WRITE CYCLE TIMING



Note 1: Ready stays high for accesses to registers 0 to 7

FIGURE 9.3: CPU - VDP MEMORY READ CYCLE TIMING

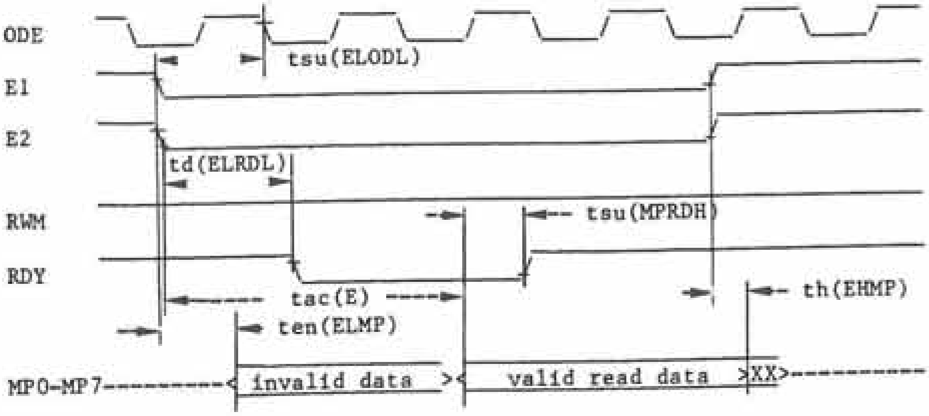


FIGURE 9.4: CPU - VDP STATUS READ CYCLE TIMING

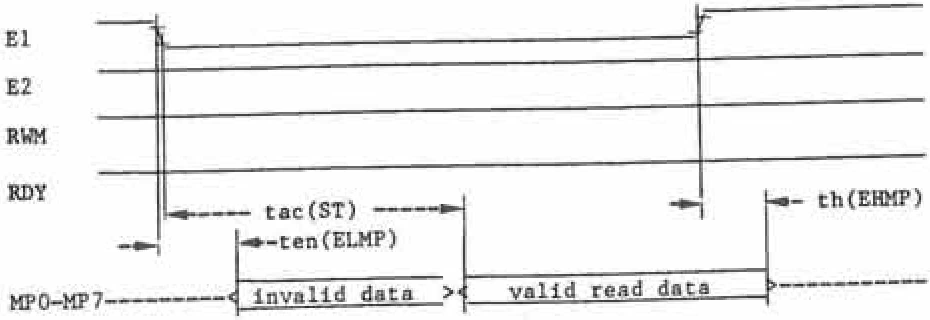
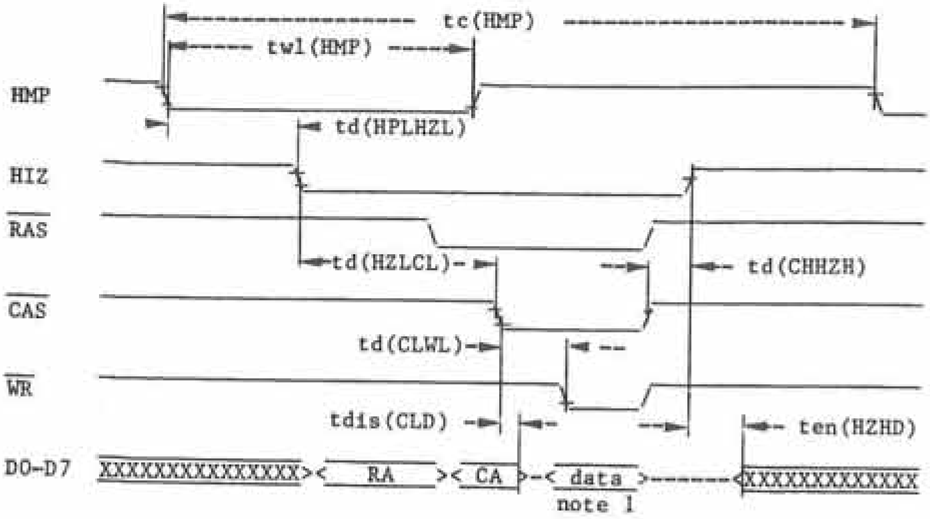


FIGURE 9.5: DATA PROVIDER WRITE CYCLE TIMING



Note 1: VDP MP bus is high impedance, data is output by the data provider. Data set-up and hold times with respect to WR are determined by memory timing requirements.

FIGURE 9.6: VDP - DRAM WRITE CYCLE TIMING

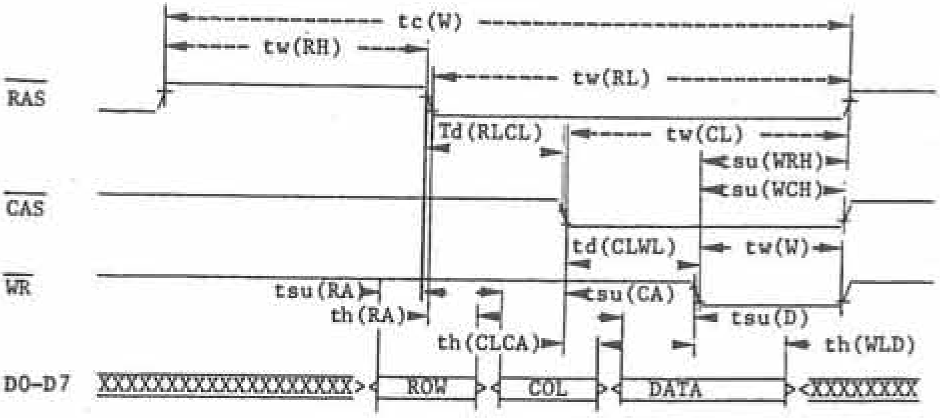
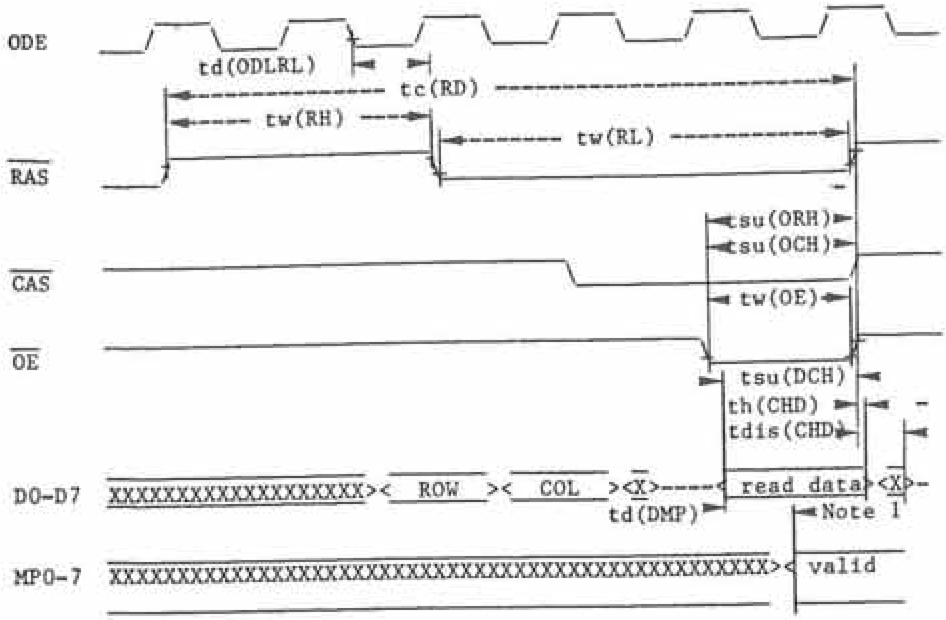
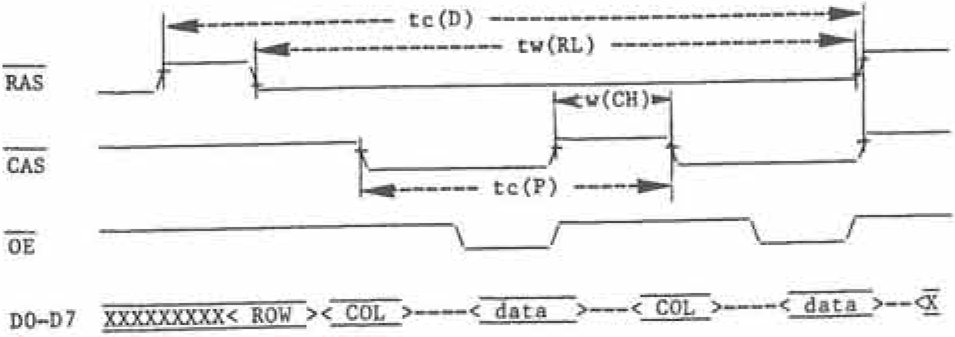


FIGURE 9.7: VDP - DRAM READ CYCLE TIMING



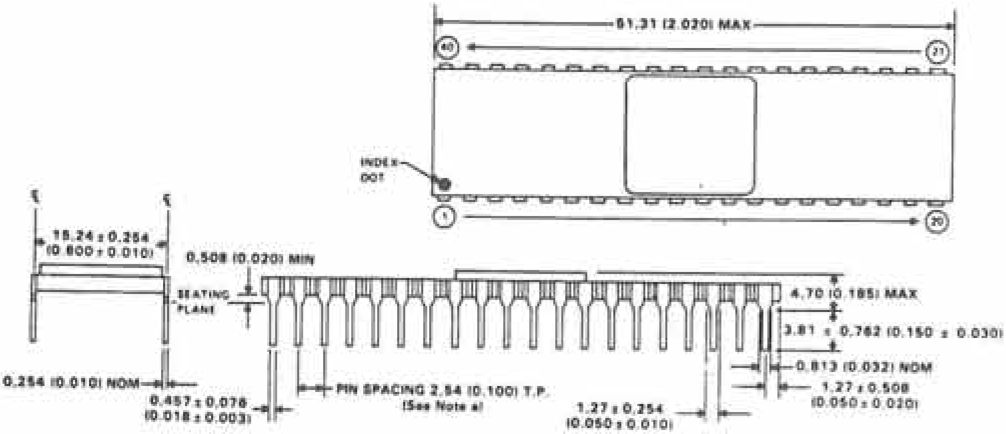
Note 1: td(DMP) applies to CPU memory read cycles only

FIGURE 9.8: VDP - DRAM DOUBLE READ CYCLE TIMING



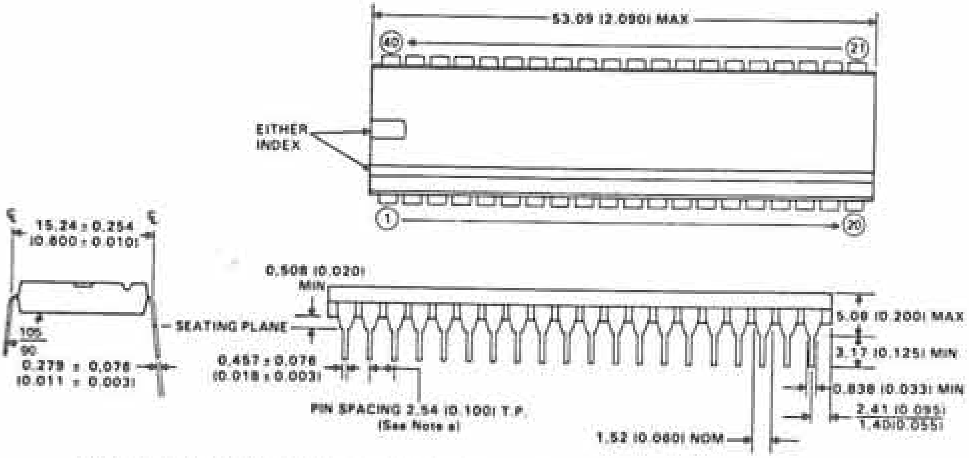
10 MECHANICAL DATA

40-PIN CERAMIC PACKAGE



NOTES: a. Each pin centerline is located within 0.254 (0.010) of its true longitudinal position.
b. All linear dimensions are in millimeters and parenthetically in inches. Inch dimensions govern.

40-PIN PLASTIC PACKAGE



NOTES: a. Each pin centerline is located within 0.254 (0.010) of its true longitudinal position.
b. All linear dimensions are in millimeters and parenthetically in inches. Inch dimensions govern.

11. MASK PROGRAMMED TIMEBASE STANDARD OPTIONS

As described in section 4 the VDP contains two areas of mask programmable ROM defining the timebase parameters. Table 11.1 gives the values of these parameters for a the dual 320 point option VDP. The latter carries the programming reference: XA8627. It diifers from the XA8328 version essentially in the fact that both 625 and 525 line modes display 40 characters per line.

For the 525 line mode, a typical dot clock would be 7.15MHz giving a line duration of 63.78us.

TABLE 11.1: TIME BASE PARAMETERS FOR DUAL 320 POINT VDP (XA8627)

PARAMETER	Programmed times BT2=0	 BT2=1	UNIT
Vertical sync period (frame length)			
non-interlaced mode	626	524	Half-lines
interlaced mode	625	525	----"----
Character width	8	8	Dots
Horizontal sync period (line length)	58	57	Characters
Horizontal sync pulse width	4	4	----"----
Blanked area after falling edge of horizontal sync pulse	9	8	----"----
Left hand border	4	4	----"----
Horizontal display area	40	40	----"----
Right hand border	4	4	----"----
Blanked area before falling edge of horizontal sync pulse	1	1	----"----
Leading equalisation duration or blanked area before vertical sync pulse			
non-interlaced	6	6	Half-lines
interlaced - even frames	5	7	----"----
- odd frames	6	6	----"----
Vertical sync pulse width	5	5	----"----
Trailing equalisation duration			
non-interlaced	5	6	----"----
interlaced - even frames	5	6	----"----
- odd frames	4	5	----"----
Blanked area after falling edge of vertical sync pulse			
non-interlaced	40	40	-----"----
interlaced - even frames	40	40	----"----
odd frames	39	39	----"----
Upper border height - normal mode	36	32	----"----
- subtitling mode	476	392	----"----
Vertical display area - normal mode	500	420	----"----
- subtitling mode	60	60	----"----
Lower border height			
non-interlaced	44	26	----"----
interlaced - even frames	44	26	----"----
- odd frames	44	28	----"----
SLL output low level pulse width	29	28	Characters
SCT output low level pulse width non-interlaced	40	40	Half-lines
interlaced - even frames	40	40	----"----
- odd frames	39	39	----"----