6 Cache, Write Buffer and Coprocessors

The chapter describes the ARM processor instruction and data cache, and its write buffer.

6.1 Instruction and Data Cache (IDC)

ARM processor contains a 4Kbyte mixed instruction and data cache. The IDC has 256 lines of 16 bytes (4 words), organised as a 4-way set associative cache, and uses the virtual addresses generated by the processor core. The IDC is always reloaded a line at a time (4 words). It may be enabled or disabled via the ARM processor Control Register and is disabled on **nRESET**.

The operation of the cache is further controlled by the *Cacheable* or C bit stored in the Memory Management Page Table (see the Memory Management Unit chapter). For this reason, in order to use the IDC, the MMU must be enabled. The two functions may however be enabled simultaneously, with a single write to the Control Register.

6.1.1 Cacheable bit

The *Cacheable* bit determines whether data being read may be placed in the IDC and used for subsequent read operations. Typically main memory will be marked as Cacheable to improve system performance, and I/O space as Non-cacheable to stop the data being stored in ARM7500's cache. [For example if the processor is polling a hardware flag in I/O space, it is important that the processor is forced to read data from the external peripheral, and not a copy of initial data held in the cache]. The Cacheable bit can be configured for both pages and sections.

6.1.2 IDC operation

In the ARM processor the cache will be searched regardless of the state of the C bit, only reads that miss the cache will be affected.

Cacheable Reads	C = 1
	A linefetch of 4 words will be performed and it will be randomly placed in a cache bank.
Uncacheable Reads	C = 0
	An external memory access will be performed and the cache will not be written.

6.1.3 IDC validity

The IDC operates with virtual addresses, so care must be taken to ensure that its contents remain consistent with the virtual to physical mappings performed by the Memory Management Unit. If the Memory Mappings are changed, the IDC validity must be ensured.

Software IDC Flush

The entire IDC may be marked as invalid by writing to the ARM processor IDC Flush Register (Register 7). The cache will be flushed immediately the register is written, but note that the next two instruction fetches may come from the cache before the register is written.

6.1.4 Doubly mapped space

Since the cache works with virtual addresses, it is assumed that every virtual address maps to a different physical address. If the same physical location is accessed by more than one virtual address, the cache cannot maintain consistency, since each virtual address will have a separate entry in the cache, and only one entry will be updated on a processor write operation. To avoid any cache inconsistencies, both doubly-mapped virtual addresses should be marked as uncacheable.

6.2 Read-Lock-Write

The IDC treats the Read-Locked-Write instruction as a special case. The read phase always forces a read of external memory, regardless of whether the data is contained in the cache. The write phase is treated as a normal write operation (and if the data is already in the cache, the cache will be updated). Externally the two phases are flagged as indivisible by asserting the **LOCK** signal.

6.3 IDC Enable/Disable and Reset

The IDC is automatically disabled and flushed on **nRESET**. Once enabled, cacheable read accesses will cause lines to be placed in the cache.

6.3.1 To enable the IDC

To enable the IDC, make sure that the MMU is enabled first by setting bit 0 in Control Register, then enable the IDC by setting bit 2 in Control Register. The MMU and IDC may be enabled simultaneously with a single control register write.

6.3.2 To disable the IDC

To disable the IDC, clear bit 2 in the Control Register and perform a flush by writing to the flush register.

6.4 Write buffer (WB)

The ARM processor write buffer is provided to improve system performance. It can buffer up to 8 words of data, and 4 independent addresses. It may be enabled or disabled via the W bit (bit 3) in the ARM processor Control Register and the buffer is disabled and flushed on reset.

The operation of the write buffer is further controlled by one bit, B, or Bufferable, which is stored in the Memory Management Page Tables. For this reason, in order to use the write buffer, the MMU must be enabled.

The two functions may however be enabled simultaneously, with a single write to the Control Register. For a write to use the write buffer, both the W bit in the Control Register, and the B bit in the corresponding page table must be set.

6.4.1 Bufferable bit

This bit controls whether a write operation may or may not use the write buffer. Typically main memory will be bufferable and I/O space unbufferable. The Bufferable bit can be configured for both pages and sections.

6.4.2 Write buffer operation

When the CPU performs a write operation, the translation entry for that address is inspected and the state of the B bit determines the subsequent action. If the write buffer is disabled via the ARM processor Control Register, bufferable writes are treated in the same way as unbuffered writes.

Bufferable write

If the write buffer is enabled and the processor performs a write to a bufferable area, the data is placed in the write buffer at FCLK speeds and the CPU continues execution. The write buffer then performs the external write in parallel. If however the write buffer is full (either because there are already 8 words of data in the buffer, or because there is no slot for the new address) then the processor is stalled until there is sufficient space in the buffer.

Unbufferable writes

If the write buffer is disabled or the CPU performs a write to an unbufferable area, the processor is stalled until the write buffer empties and the write completes externally, which may require synchronisation and several external clock cycles.

Read-lock-write

The write phase of a read-lock-write sequence is treated as an Unbuffered write, even if it is marked as buffered.

Note: *A single write requires one address slot and one data slot in the write buffer; a sequential write of n words requires one address slot and n data slots. The total of 8 data slots in the buffer may be used as required. So for instance there could be 3 non-sequential writes and one sequential write of 5 words in the buffer, and the processor could continue as normal: a 5th write or an 6th word in the 4th write would stall the processor until the first write had completed.*

To enable the write buffer

To enable the write buffer, ensure the MMU is enabled by setting bit 0 in the Control Register, then enable the write buffer by setting bit 3 in the Control Register. The MMU and write buffer may be enabled simultaneously with a single write to the Control Register.

To disable the write buffer

To disable the write buffer, clear bit 3 in the Control Register.

Note: *Any writes already in the write buffer will complete normally.*

6.5 Coprocessors

ARM processor has no external coprocessor bus, so it is not possible to add external coprocessors to this device.

The ARM processor still has an internal coprocessor designated #15 for internal control of the device. All coprocessor operations except MCR or MRC to registers 0 to 7 on coprocessor #15 will cause the undefined instruction trap to be taken.

7 ARM Processor MMU

This chapter describes the ARM processor Memory Management Unit.

7.1 Introduction

The MMU performs two primary functions: it translates virtual addresses into physical addresses, and it controls memory access permissions. The MMU hardware required to perform these functions consists of a Translation Look-aside Buffer (TLB), access control logic, and translation table walking logic.

The MMU supports memory accesses based on Sections or Pages. Sections are comprised of 1MB blocks of memory. Two different page sizes are supported: Small Pages consist of 4Kb blocks of memory and Large Pages consist of 64Kb blocks of memory. (Large Pages are supported to allow mapping of a large region of memory while using only a single entry in the TLB.) Additional access control mechanisms are extended within Small Pages to 1Kb Sub-Pages and within Large Pages to 16Kb Sub-Pages.

The MMU also supports the concept of domains - areas of memory that can be defined to possess individual access rights. The Domain Access Control Register is used to specify access rights for up to 16 separate domains.

The TLB caches 64 translated entries. During most memory accesses, the TLB provides the translation information to the access control logic.

If the TLB contains a translated entry for the virtual address, the access control logic determines whether access is permitted. If access is permitted, the MMU outputs the appropriate physical address corresponding to the virtual address. If access is not permitted, the MMU signals the CPU to abort.

If the TLB misses (it does not contain a translated entry for the virtual address), the translation table walk hardware is invoked to retrieve the translation information from a translation table in physical memory. Once retrieved, the translation information is placed into the TLB, possibly overwriting an existing value. The entry to be overwritten is chosen by cycling sequentially through the TLB locations.

When the MMU is turned off (as happens on reset), the virtual address is output directly onto the physical address bus.

7.2 MMU program-accessible registers

The ARM processor provides several 32-bit registers which determine the operation of the MMU. The format for these registers and a brief description is shown in *Figure 7-1: MMU register summary* on page 7-3. Each register will be discussed in more detail within the section that describes its use.

Data is written to and read from the MMUs registers using the ARM CPU's MRC and MCR coprocessor instructions.

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1 write	**0**	**Control**																					**R**	**S**	**B**	**1**	**D**	**P**	**W**	**C**	**A**	**M**
2 write	**Translation Table Base**																															
3 write	15		14		13		12		11		10		Domain Access Control 9		8		7		6		5		4		3		2		1		0	
5 read	**Fault Status**																				**0**	**0**	**0**	**0**	**Domain**				**Status**			
5 write	**Flush TLB**																															
6 read	**Fault Address**																															
6 write	**TLB Purge Address**																															

Figure 7-1: MMU register summary

Translation Table Base Register

The Translation Table Base Register holds the physical address of the base of the translation table maintained in main memory. Note that this base must reside on a 16KB boundary.

Domain Access Control Register

The Domain Access Control Register consists of sixteen 2-bit fields, each of which defines the access permissions for one of the sixteen Domains (D15-D0).

Note: *The registers not shown are reserved and should not be used.*

Fault Status Register

The Fault Status Register indicates the domain and type of access being attempted when an abort occurred. Bits 7:4 specify which of the sixteen domains (D15-D0) was being accessed when a fault occurred. Bits 3:1 indicate the type of access being attempted. The encoding of these bits is different for internal and external faults (as indicated by bit 0 in the register) and is shown in *Table 7-4: Priority encoding of fault status* on page 7-12. A write to this register flushes the TLB.

Fault Address Register

The Fault Address Register holds the virtual address of the access which was attempted when a fault occurred. A write to this register causes the data written to be treated as an address and, if it is found in the TLB, the entry is marked as invalid. (This operation is known as a TLB purge). The Fault Status Register and Fault Address Register are only updated for data faults, not for prefetch faults.

7.3 Address translation

The MMU translates virtual addresses generated by the CPU into physical addresses to access external memory, and also derives and checks the access permission. Translation information, which consists of both the address translation data and the access permission data, resides in a translation table located in physical memory. The MMU provides the logic needed to traverse this translation table, obtain the translated address, and check the access permission.

There are three routes by which the address translation (and hence permission check) takes place. The route taken depends on whether the address in question has been marked as a section-mapped access or a page-mapped access; and there are two sizes of page-mapped access (large pages and small pages). However, the translation process always starts out in the same way, as described below, with a Level One fetch. A section-mapped access only requires a Level One fetch, but a page-mapped access also requires a Level Two fetch.

7.4 Translation process

7.4.1 Translation table base

The translation process is initiated when the on-chip TLB does not contain an entry for the requested virtual address. The Translation Table Base (TTB) Register points to the base of a table in physical memory which contains Section and/or Page descriptors. The 14 low-order bits of the TTB Register are set to zero as illustrated in *Figure 7-2: Translation table base register*; the table must reside on a 16Kb boundary.

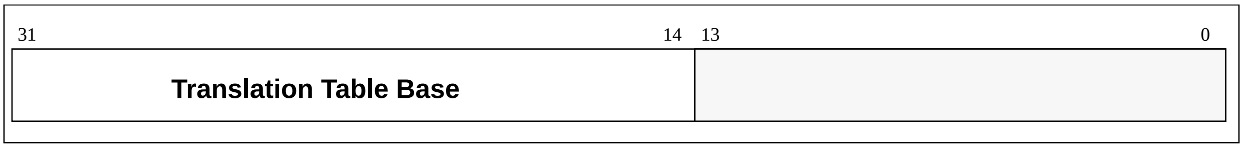


Figure 7-2: Translation table base register

7.4.2 Level one fetch

Bits 31:14 of the Translation Table Base register are concatenated with bits 31:20 of the virtual address to produce a 30-bit address as illustrated in *Figure 7-3: Accessing the translation table first level descriptors* on page 7-5. This address selects a four-byte translation table entry which is a First Level Descriptor for either a Section or a Page (bit1 of the descriptor returned specifies whether it is for a Section or Page).

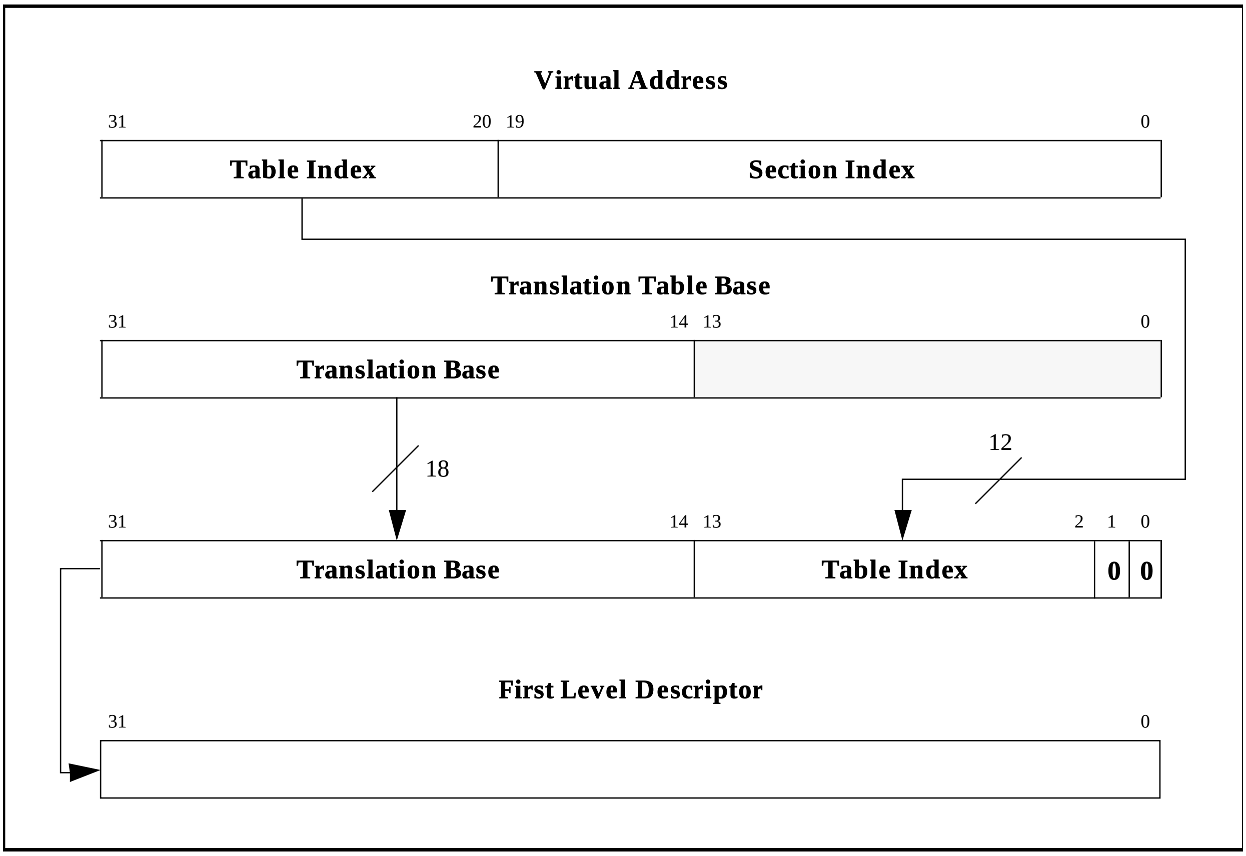


Figure 7-3: Accessing the translation table first level descriptors

7.4.3 Level one descriptor

The Level One Descriptor returned is either a Page Table Descriptor or a Section Descriptor, and its format varies accordingly. The following figure illustrates the format of Level One Descriptors.

31 – 20	19 – 12	11 – 10	9	8 – 5	4	3	2	1	0	
								0	0	**Fault**
Page Table Base Address				**Domain**	**1**			0	1	**Page**
Section Base Address		**AP**		**Domain**	**1**	**C**	**B**	1	0	**Section**
								1	1	**Reserved**

Figure 7-4: Level one descriptors

The two least significant bits indicate the descriptor type and validity, and are interpreted as shown below

Value	Meaning	Notes
0 0	Invalid	Generates a Section Translation Fault
0 1	Page	Indicates that this is a Page Descriptor
1 0	Section	Indicates that this is a Section Descriptor
1 1	Reserved	Reserved for future use

Table 7-1: Interpreting level one descriptor bits [1:0]

7.4.4 Page table descriptor

Bits 3:2 are always written as 0.

Bit 4 should be written to 1 for backward compatibility.

Bits 8:5 specify one of the sixteen possible domains (held in the Domain Access Control Register) that contain the primary access controls.

Bits 31:10 form the base for referencing the Page Table Entry. (The page table index for the entry is derived from the virtual address as illustrated in *Figure 7-7: Small page translation* on page 7-10).

If a Page Table Descriptor is returned from the Level One fetch, a Level Two fetch is initiated as described below.

7.4.5 Section descriptor

Bits 3:2 (C, & B) control the cache- and write-buffer-related functions as follows:

C - Cacheable: indicates that data at this address will be placed in the cache (if the cache is enabled).

B - Bufferable: indicates that data at this address will be written through the write buffer (if the write buffer is enabled).

Bit 4 should be written to 1 for backward compatibility.

Bits 8:5 specify one of the sixteen possible domains (held in the Domain Access Control Register) that contain the primary access controls.

Bits 11:10 (AP) specify the access permissions for this section and are interpreted as shown in *Table 7-2: Interpreting access permission (AP) bits* on page 7-7. Their interpretation is dependent upon the setting of the S and R bits (control register bits 8 and 9). Note that the Domain Access Control specifies the primary access control; the AP bits only have an effect in client mode. Refer to section on access permissions.

AP	S	R	Supervisor permissions	User permissions	Notes
00	0	0	No Access	No Access	Any access generates a permission fault
00	1	0	Read Only	No Access	Supervisor read only permitted
00	0	1	Read Only	Read Only	Any write generates a permission fault
00	1	1	Reserved		
01	x	x	Read/Write	No Access	Access allowed only in Supervisor mode
10	x	x	Read/Write	Read Only	Writes in User mode cause permission fault
11	x	x	Read/Write	Read/Write	All access types permitted in both modes.
xx	1	1	Reserved		

Table 7-2: Interpreting access permission (AP) bits

Bits 19:12 are always written as 0.

Bits 31:20 form the corresponding bits of the physical address for the 1MByte section.

7.5 Translating section references

Figure 7-5: Section translation illustrates the complete Section translation sequence. Note that the access permissions contained in the Level One Descriptor must be checked before the physical address is generated. The sequence for checking access permissions is described below.

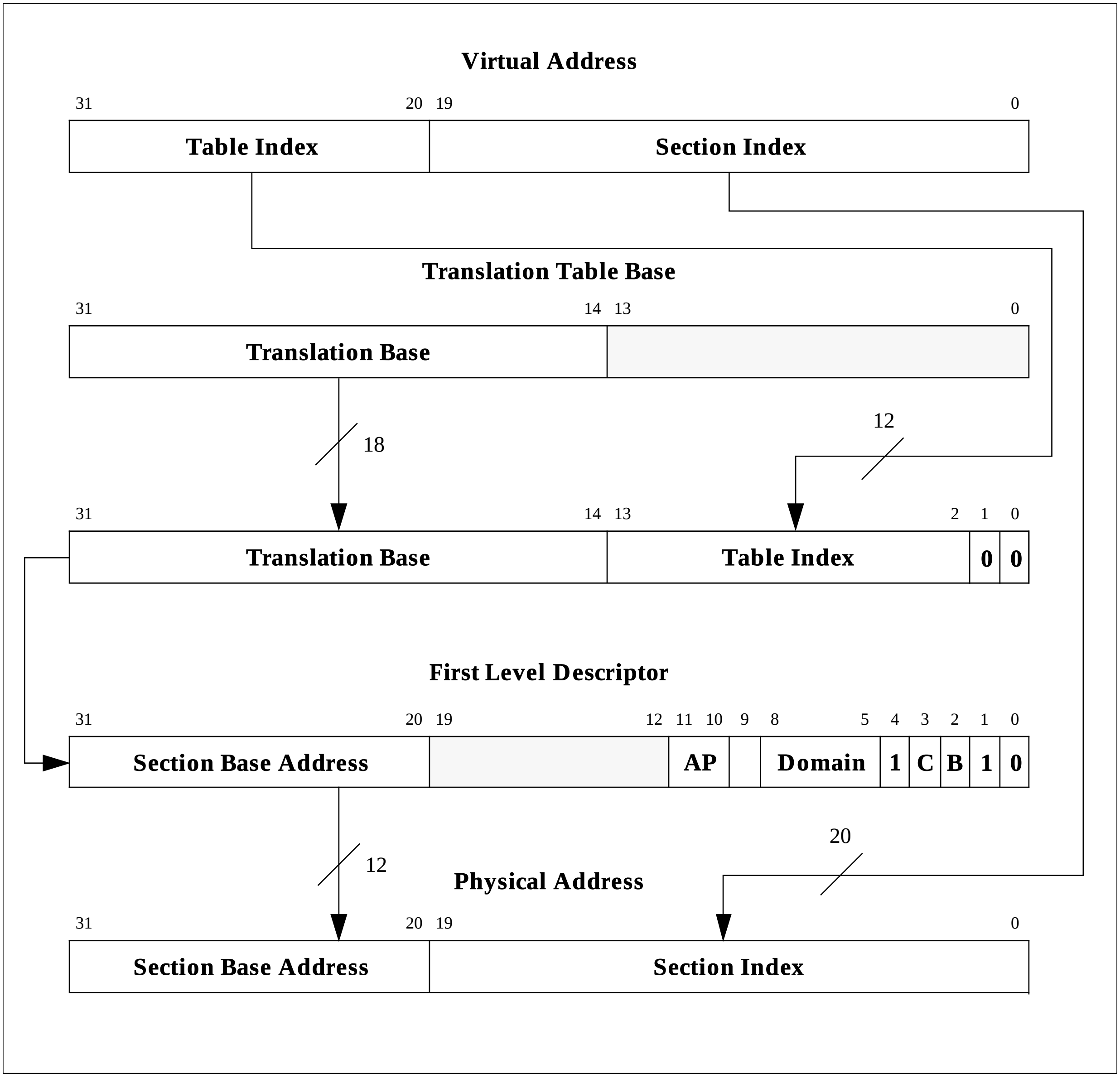
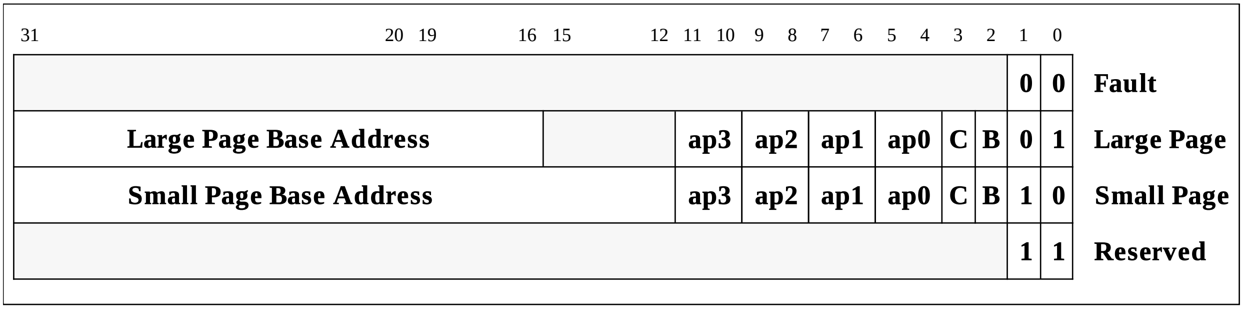


Figure 7-5: Section translation

7.5.1 Level two descriptor

If the Level One fetch returns a Page Table Descriptor, this provides the base address of the page table to be used. The page table is then accessed as described in *Figure 7-7: Small page translation*, and a Page Table Entry, or Level Two Descriptor, is returned. This in turn may define either a Small Page or a Large Page access. *Figure 7-6: Page table entry (level two descriptor)* on page 7-9 shows the format of Level Two Descriptors.



31 – 20	19 – 16	15 – 12	11 – 10	9 – 8	7 – 6	5 – 4	3	2	1	0	
									0	0	**Fault**
Large Page Base Address			**ap3**	**ap2**	**ap1**	**ap0**	**C**	**B**	0	1	**Large Page**
Small Page Base Address			**ap3**	**ap2**	**ap1**	**ap0**	**C**	**B**	1	0	**Small Page**
									1	1	**Reserved**

Figure 7-6: Page table entry (level two descriptor)

The two least significant bits indicate the page size and validity, and are interpreted as follows.

Value	Meaning	Notes
0 0	Invalid	Generates a Page Translation Fault
0 1	Large Page	Indicates that this is a 64 Kb Page
1 0	Small Page	Indicates that this is a 4 Kb Page
1 1	Reserved	Reserved for future use

Table 7-3: Interpreting page table entry bits 1:0

Bit 2 B - Bufferable: indicates that data at this address will be written through the write buffer (if the write buffer is enabled).

Bit 3 C - Cacheable: indicates that data at this address will be placed in the IDC (if the cache is enabled).

Bits 11:4 specify the access permissions (ap3 - ap0) for the four sub-pages and interpretation of these bits is described earlier in *Table 7-1: Interpreting level one descriptor bits [1:0]* on page 7-6.

For large pages, **bits 15:12** are programmed as 0.

Bits 31:12 (small pages) or bits **31:16** (large pages) are used to form the corresponding bits of the physical address - the physical page number. (The page index is derived from the virtual address as illustrated in *Figure 7-7: Small page translation* on page 7-10 and *Figure 7-8: Large page translation* on page 7-11).

7.6 Translating small page references

Figure 7-7: Small page translation illustrates the complete translation sequence for a 4kB Small Page. Page translation involves one additional step beyond that of a section translation: the Level One descriptor is the Page Table descriptor, and this is used to point to the Level Two descriptor, or Page Table Entry. (Note that the access permissions are now contained in the Level Two descriptor and must be checked before the physical address is generated. The sequence for checking access permissions is described later).

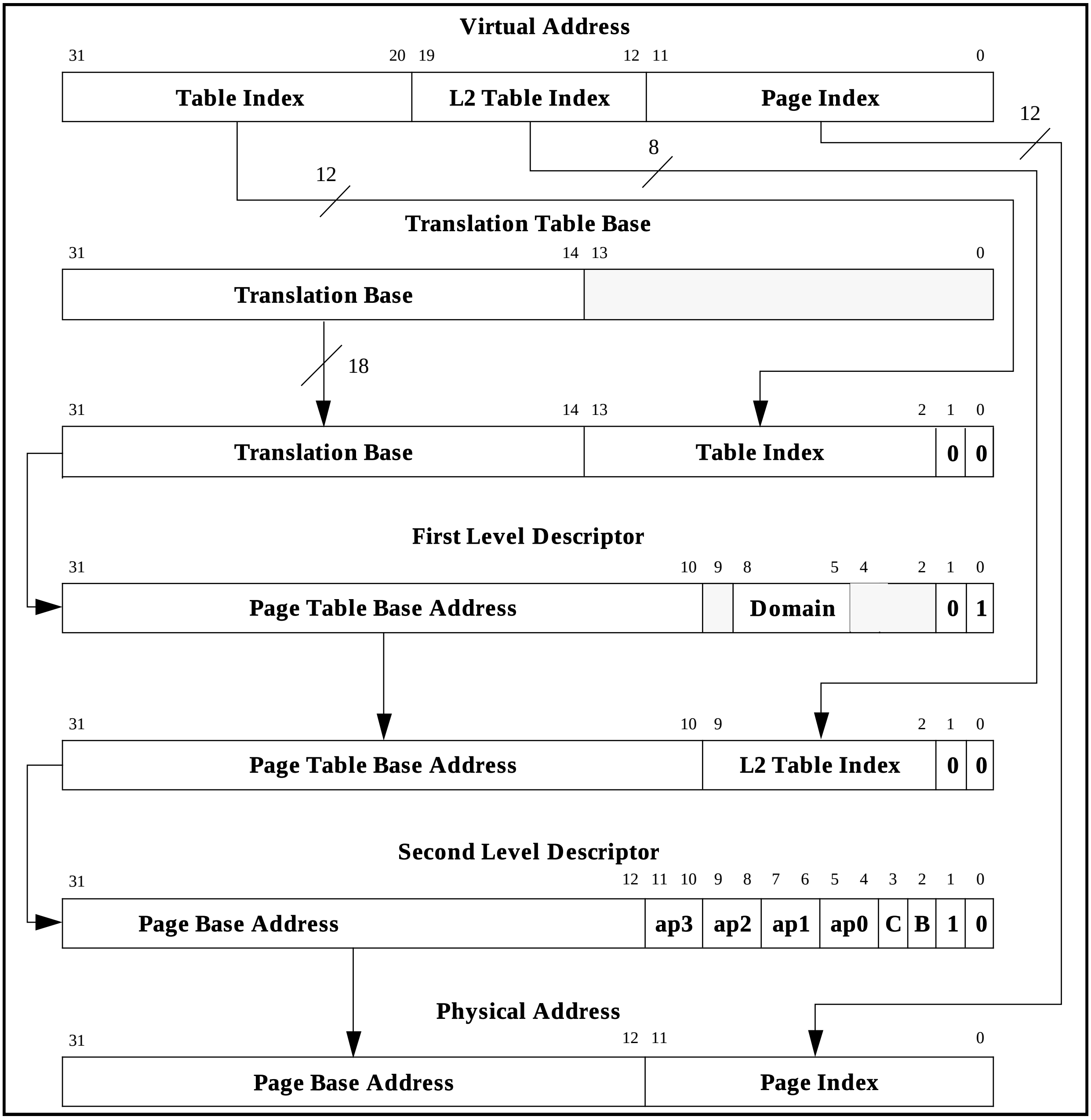


Figure 7-7: Small page translation

7.7 Translating large page references

Figure 7-8: Large page translation illustrates the complete translation sequence for a 64Kb Large Page. Note that since the upper four bits of the Page Index and low-order four bits of the Page Table index overlap, each Page Table Entry for a Large Page must be duplicated 16 times (in consecutive memory locations) in the Page Table.

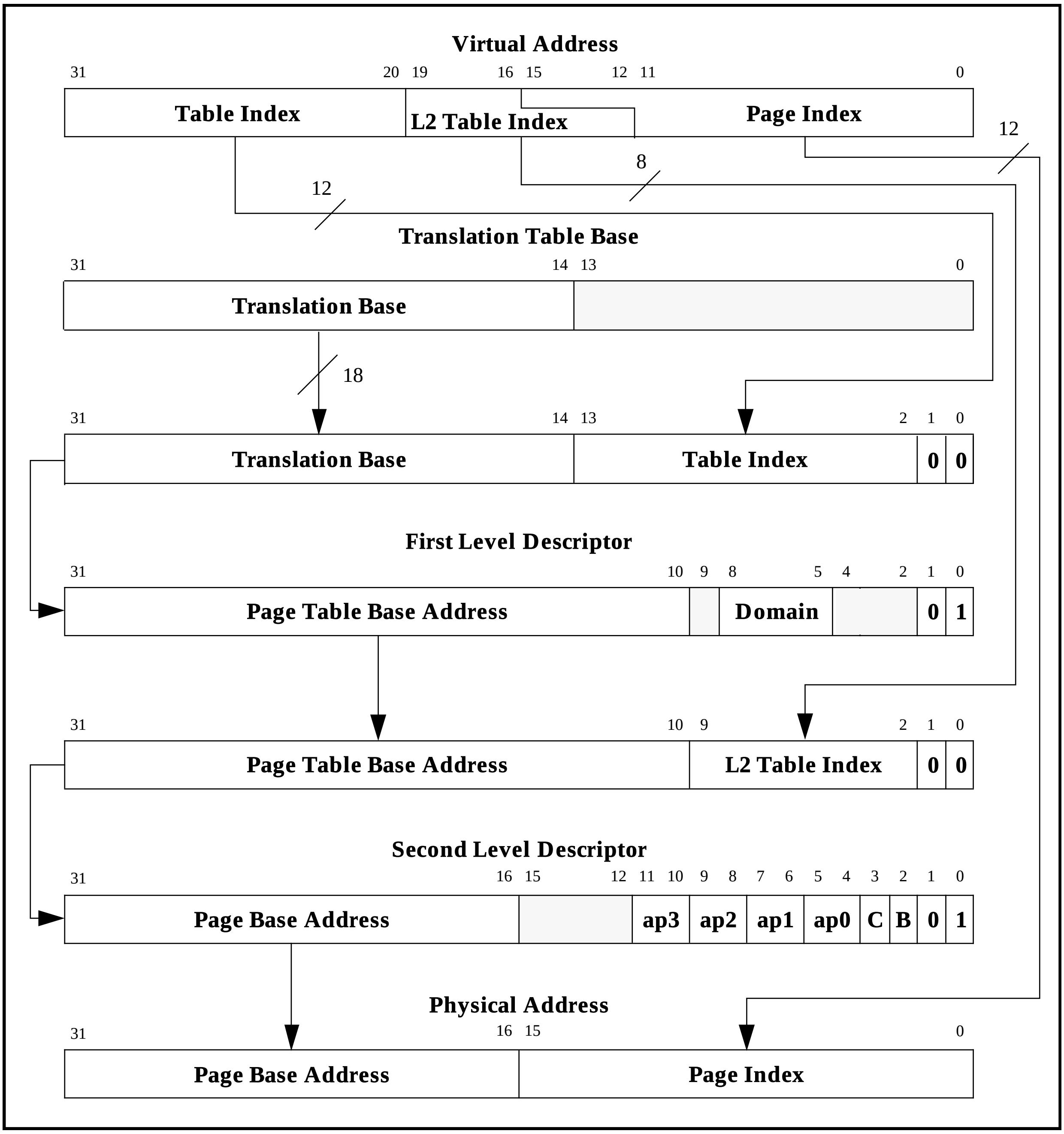


Figure 7-8: Large page translation

7.8 MMU faults and CPU aborts

The MMU generates four types of faults:

Alignment Fault

Translation Fault

Domain Fault

Permission Fault

The access control mechanisms of the MMU detect the conditions that produce these faults. If a fault is detected as the result of a memory access, the MMU will abort the access and signal the fault condition to the CPU. The MMU is also capable of retaining status and address information about the abort. The CPU recognises two types of abort: data aborts and prefetch aborts, and these are treated differently by the MMU.

If the MMU detects an access violation, it will do so before the external memory access takes place, and it will therefore inhibit the access.

7.9 Fault Address & Fault Status Registers (FAR & FSR)

Aborts resulting from data accesses (data aborts) are acted upon by the CPU immediately, and the MMU places an encoded 4 bit value FS[3:0], along with the 4 bit encoded Domain number, in the Fault Status Register (FSR). In addition, the virtual processor address which caused the data abort is latched into the Fault Address Register (FAR). If an access violation simultaneously generates more than one source of abort, they are encoded in the priority given in ❯*Table 7-4: Priority encoding of fault status* on page 7-12.

CPU instructions on the other hand are prefetched, so a prefetch abort simply flags the instruction as it enters the instruction pipeline. Only when (and if) the instruction is executed does it cause an abort; an abort is not acted upon if the instruction is not used (i.e. it is branched around). Because instruction prefetch aborts may or may not be acted upon, the MMU status information is not preserved for the resulting CPU abort; for a prefetch abort, the MMU does not update the FSR or FAR.

The sections that follow describe the various access permissions and controls supported by the MMU and detail how these are interpreted to generate faults.

Priority	Source	FS[3210]	Domain [3:0]	FAR
Highest	Alignment	00x1	x	valid
	Translation (Section)	0101	Note 2	valid
	Translation (Page)	0111	valid	valid
	Domain (Section)	1001	valid	valid
	Domain (Page)	1011	valid	valid

Table 7-4: Priority encoding of fault status

Priority	Source	FS[3210]	Domain [3:0]	FAR
	Permission (Section)	1101	valid	valid
Lowest	Permission (Page)	1111	valid	valid

Table 7-4: Priority encoding of fault status (Continued)

x is undefined, and may read as 0 or 1

Notes: *Any abort masked by the priority encoding may be regenerated by fixing the primary abort and restarting the instruction.*
In fact this register will contain bits[8:5] of the Level 1 entry which are undefined, but would encode the domain in a valid entry.

7.10 Domain access control

MMU accesses are primarily controlled via domains. There are 16 domains, and each has a 2-bit field to define it. Two basic kinds of users are supported: Clients and Managers. Clients use a domain; Managers control the behaviour of the domain. The domains are defined in the Domain Access Control Register. ➲*Figure 7-9: Domain access control register format* illustrates how the 32 bits of the register are allocated to define the sixteen 2-bit domains.

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 7-9: Domain access control register format

➲*Table 7-5: Interpreting access bits in domain access control register* defines how the bits within each domain are interpreted to specify the access permissions.

Value	Meaning	Notes
00	No Access	Any access will generate a Domain Fault.
01	Client	Accesses are checked against the access permission bits in the Section or Page descriptor.
10	Reserved	Reserved. Currently behaves like the no access mode.
11	Manager	Accesses are NOT checked against the access Permission bits so a Permission fault cannot be generated.

Table 7-5: Interpreting access bits in domain access control register

7.11 Fault checking sequence

The sequence by which the MMU checks for access faults is slightly different for Sections and Pages. The figure below illustrates the sequence for both types of accesses. The sections and figures that follow describe the conditions that generate each of the faults.

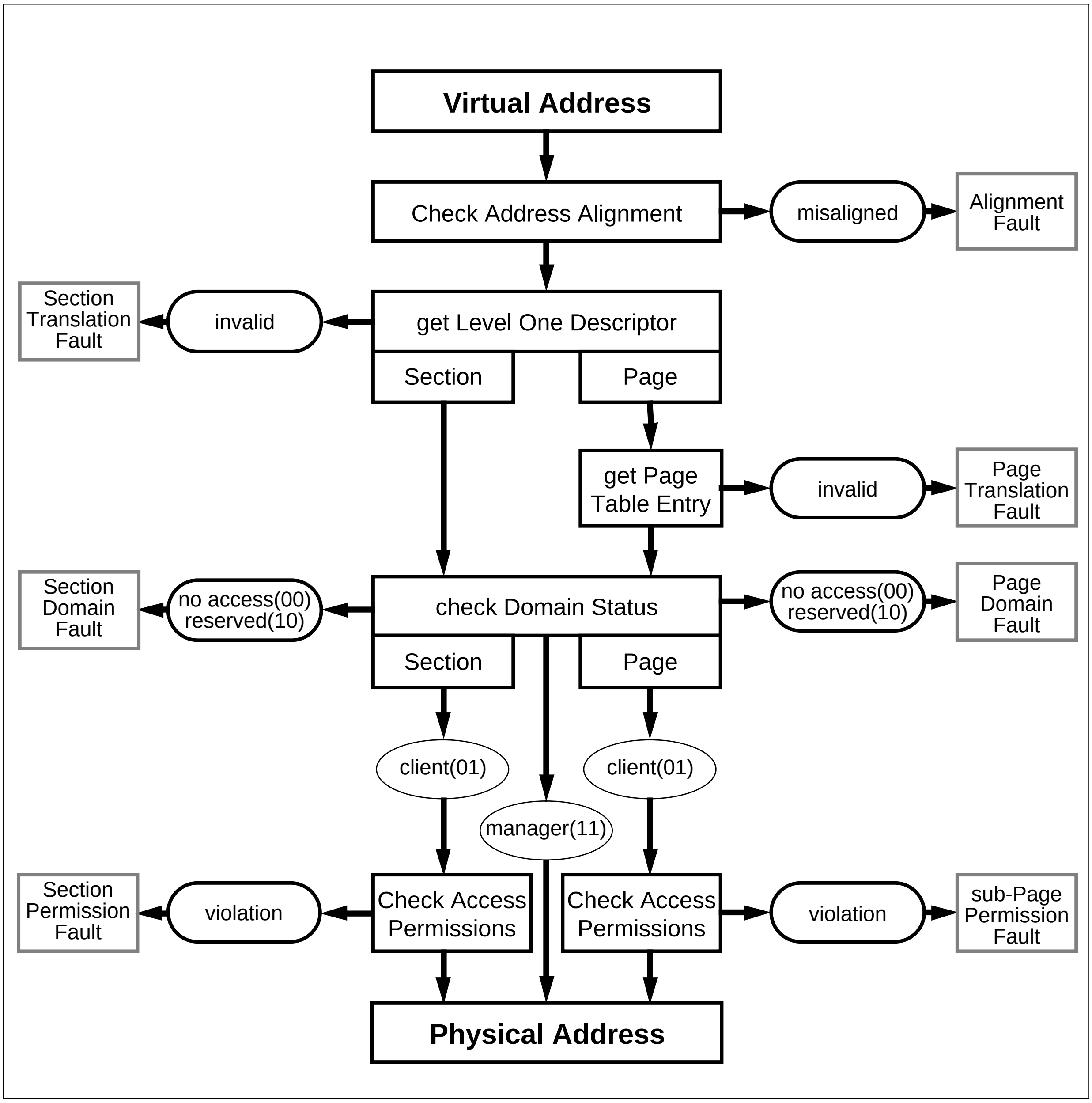


Figure 7-10: Sequence for checking faults

7.11.1 Alignment fault

If Alignment Fault is enabled (bit 1 in Control Register set), the MMU will generate an alignment fault on any data word access the address of which is not word-aligned irrespective of whether the MMU is enabled or not; in other words, if either of virtual address bits [1:0] are not 0. Alignment fault will not be generated on any instruction fetch, nor on any byte access. Note that if the access generates an alignment fault, the access sequence will abort without reference to further permission checks.

7.11.2 Translation fault

There are two types of translation fault: section and page.

1 A Section Translation Fault is generated if the Level One descriptor is marked as invalid. This happens if bits[1:0] of the descriptor are both 0 or both 1.

2 A Page Translation Fault is generated if the Page Table Entry is marked as invalid. This happens if bits[1:0] of the entry are both 0 or both 1.

7.11.3 Domain fault

There are two types of domain fault: section and page. In both cases the Level One descriptor holds the 4-bit Domain field which selects one of the sixteen 2-bit domains in the Domain Access Control Register. The two bits of the specified domain are then checked for access permissions as detailed in ➲*Table 7-2: Interpreting access permission (AP) bits* on page 7-7. In the case of a section, the domain is checked once the Level One descriptor is returned, and in the case of a page, the domain is checked once the Page Table Entry is returned.

If the specified access is either No Access (00) or Reserved (10) then either a Section Domain Fault or Page Domain Fault occurs.

7.11.4 Permission fault

There are two types of permission fault: section and sub-page. Permission fault is checked at the same time as Domain fault. If the 2-bit domain field returns client (01), then the permission access check is invoked as follows:

Section

If the Level One descriptor defines a section-mapped access, then the AP bits of the descriptor define whether or not the access is allowed according to ➲*Table 7-2: Interpreting access permission (AP) bits* on page 7-7. Their interpretation is dependent upon the setting of the S bit (Control Register bit 8). If the access is not allowed, then a Section Permission fault is generated.

Sub-page

If the Level One descriptor defines a page-mapped access, then the Level Two descriptor specifies four access permission fields (ap3..ap0) each corresponding to one quarter of the page. Hence for small pages, ap3 is selected by the top 1Kb of the page, and ap0 is selected by the bottom 1kB of the page; for large pages, ap3 is selected by the top 16Kb of the page, and ap0 is selected by the bottom 16Kb of the page. The selected AP bits are then interpreted in exactly the same way as for a section (see *Table 7-2: Interpreting access permission (AP) bits* on page 7-7), the only difference being that the fault generated is a sub-page permission fault.

7.12 External aborts

The ARM7500 does not support external aborts.

7.12.1 Interaction of the MMU, IDC and write buffer

The MMU, IDC and WB may be enabled/disabled independently. However there are only five valid combinations. There are no hardware interlocks on these restrictions, so invalid combinations will cause undefined results.

MMU	IDC	WB
off	off	off
on	off	off
on	on	off
on	off	on
on	on	on

Table 7-6: Valid MMU, IDC, and WB combinations

The following procedures must be observed.

To enable the MMU:

1 Program the Translation Table Base and Domain Access Control Registers

2 Program Level 1 and Level 2 page tables as required

3 Enable the MMU by setting bit 0 in the Control Register.

Note: *Care must be taken if the translated address differs from the untranslated address as the two instructions following the enabling of the MMU will have been fetched using “flat translation” and enabling the MMU may be considered as a branch with delayed execution. A similar situation occurs when the MMU is disabled. Consider the following code sequence:*

```
MOV             R1, #0x1
MCR             15,0,R1,0,0    ; Enable MMU
Fetch Flat
Fetch Flat
Fetch Translated
```

To disable the MMU

1. Disable the WB by clearing bit 3 in the Control Register.
2. Disable the IDC by clearing bit 2 in the Control Register.
3. Disable the MMU by clearing bit 0 in the Control Register.

Note: *If the MMU is enabled, then disabled and subsequently re-enabled the contents of the TLB will have been preserved. If these are now invalid, the TLB should be flushed before re-enabling the MMU.*

Disabling of all three functions may be done simultaneously.

7.13 Effect of reset

See *Chapter 4: ARM Processor Programmer's Model* .

8 The Video and Sound Macrocell

This chapter introduces the ARM7500 video and sound system.

8.1 Introduction

The ARM7500 single chip computer contains a high performance video and sound controller, capable of meeting the requirements of a wide range of configurations.

The video and sound macrocell handles all the video processing aspects of the ARM7500 functionality, making the ARM7500 suitable for incorporation into a wide range of end products ranging from portable hand-held LCD systems through to higher performance SuperVGA desktop products.

The flexible bus interface provides hardware support for interfacing to DRAM memory systems in conjunction with the ARM7500 memory controller. The video and sound macrocell obtains data from external DRAM under DMA control. The macrocell also incorporates two stereo sound systems - an 8-bit (logarithmic) system, featuring up to eight channels, each with its own stereo position, and an serial sound output port suitable for connection to an external CD DAC.

Features include:

- VGA, SuperVGA, XGA resolution
- three 8-bit DACs giving 16M colours
- direct driving of LCD or CRT screens
- 1, 2, 4, 8, 16, 32 bits per pixel modes
- up to 120MHz pixel rate
- very low power consumption

8.2 Features

8.2.1 Flexible video system

The video and sound macrocell contains 296 write-only registers which offer a high degree of flexibility to the system programmer. 256 of these are used as the 28-bit video palette entries. These are programmed via an auto-incrementing address pointer. The remaining registers are specific control registers and allow the user to program the display parameters.

8.2.2 Hardware cursor

The video and sound macrocell has a hardware cursor for all its display modes - Normal, Hi-Res, and LCD. By offering cursor support on chip the designer benefits in terms of speed and lower software overhead. The cursor is 32 pixels wide and any number of pixels high and can be displayed in 4 colours including transparent from its own 28-bit wide palette. In this way a cursor of any shape and size can be defined within the 32-pixel wide limit.

8.2.3 Palette

The video subsystem has a 28-bit wide 256-entry palette where each entry uses 8 bits for Red, 8 for Green and 8 for Blue, and 4 bits for external data. These external bits may be used outside the chip for a variety of purposes such as supremacy, fading, Hi-Res and LCD driving.

Look Up Tables (LUT) allow for logical to physical translation and gamma correction.The Red Green and Blue LUTs each drive their respective DACs, and the Ext LUT is normally configured to drive the 4-bit output port.

There are three 8-bit linear monotonic DACs (Red, Green and Blue) which give a total of 16M possible colours. The DACs are designed to operate up to 120 MHz and drive doubly-terminated 75Ω lines directly.

8.2.4 Pixel clock

The ARM7500 is capable of generating a display at any pixel rate up to 120MHz. The pixel clock may be selected from one of 3 sources, and then the selected frequency of this clock may be further divided down by a factor of between 1 and 8.

The video and sound macrocell contains an on-chip phase comparator which, when used in conjunction with an external Voltage Controlled Oscillator (VCO), forms a Phase Locked Loop. This configuration allows a single reference clock to generate all the required frequencies for any display mode thus obviating the need for multiple external crystals.

8.2.5 Display modes

Irrespective of the memory configuration used, the video subsystem is capable of many different display formats. In addition to the normal linear CRT display, the video subsystem can generate a display suitable for either very high resolution displays, single or dual-panel LCDs.

For CRT displays, the video and sound macrocell is capable of operating in a variety of pixel modes - 1,2,4,8,16,32 bits/pixel, and can also directly drive LCD displays in 1,2 or 4 bits per pixel via an internal 16-level grey scaler. The grey scaler algorithm adopted is patented.

8.2.6 Power management

The macrocell is designed for power sensitive applications and incorporates design features to minimise power consumption. A *power down mode* allows power savings to be made when the device is not in use, for example, in conjunction with a battery powered LCD system. Additional power sensitive features include the powering down of functions of the device currently not in use, such as the video DACs, sound DACs and the LCD grey scaler. In addition the palette design has been segmented such that only one eighth of the palette is enabled and clocked at any one time. The power-down mode can be used in conjunction with the ARM7500's STOP mode to ensure minimum power consumption when clocks are stopped.

8.2.7 On-chip sound system

The ARM7500 supports two systems - an 8-bit (logarithmic) system using an internal dedicated DAC, featuring up to eight channels each with its own stereo position, and a 32-bit serial sound output suitable for driving external CD DACs.

In the 8-bit mode the device can work with 1,2,4 or 8 stereo channels, using time division multiplexing to synthesise left and right outputs. The sample rate is programmable through the Sound Frequency Register.

Enhanced 32-bit stereo sound is offered by the serial sound output which consists of a three pin serial interface. Each 32-bit sample consists of 16 bits for the left channel and 16 bits for the right channel.

8.3 Block diagram

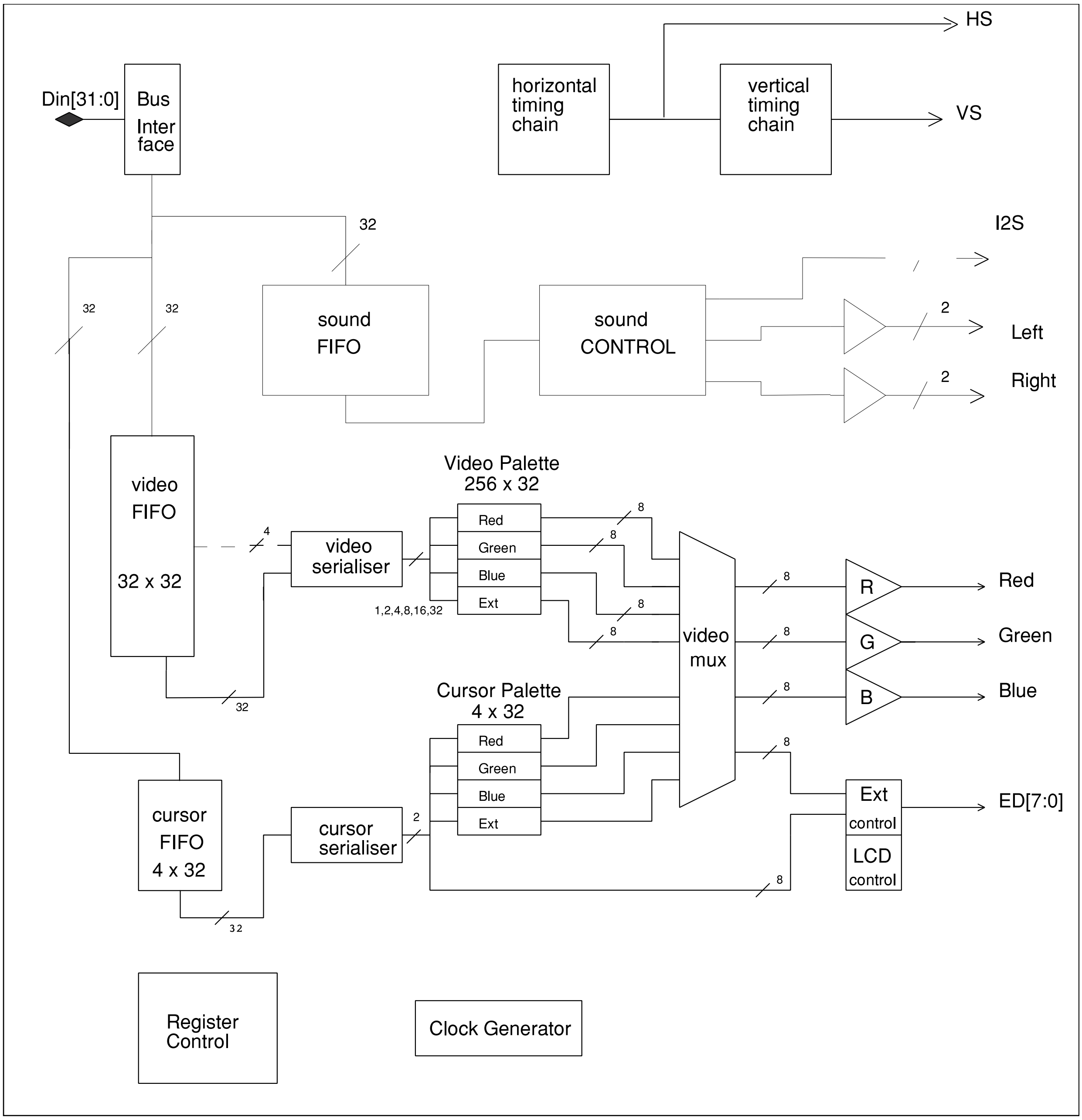


Figure 8-1: Video and sound macrocell block diagram

9 Video and Sound Programmer's Model

This chapter details the video and sound macrocell programmable registers.

9.1 The video and sound macrocell registers

The video and sound macrocell contains 296 write-only registers. These are split into 2 categories; the 256 28-bit video palette entries, and the remaining control registers. The video palette entries are written via an auto-incrementing address pointer. All the other registers (including the 28-bit cursor palette) are written directly with the address encoded in the top 4 or 8 bits of the data word. To program the registers, the ARM7500 address bus should be set to between 0x03400000 and 0x034FFFFF, and the data word written should include the individual register address in the upper 4 or 8 bits, as appropriate.

In order to define the display format correctly, eleven registers need to be programmed as shown in the diagram below:

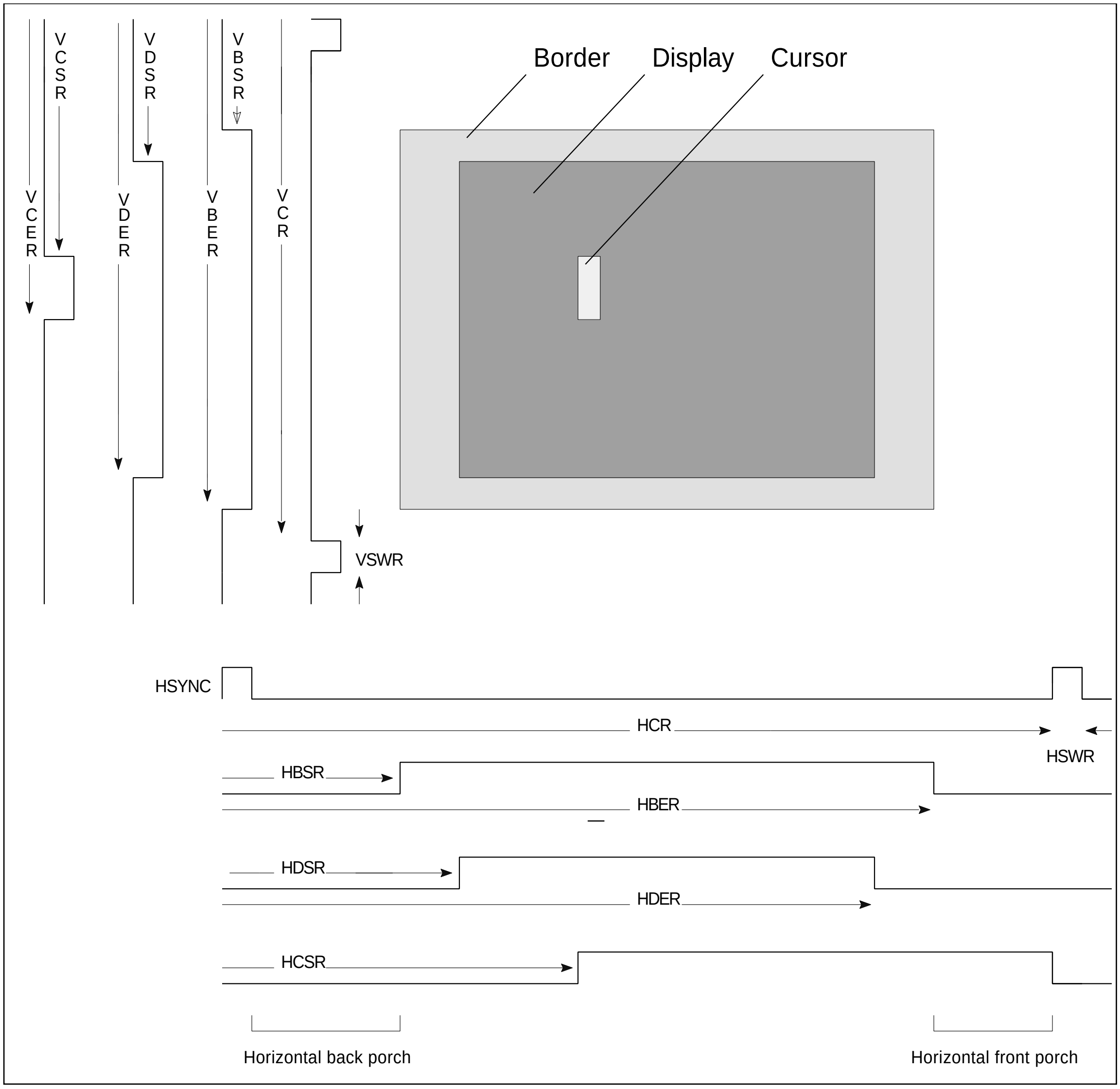


Figure 9-1: The video and sound macrocell display format definitions

The register allocation is shown in the following table. An x denotes the actual data field, and any unused bit should be programmed with a logic zero. Do not access any register at any location other than that shown as the actual register map is multiple-mapped.

The External Register, Control Register, Sound Control Register and Data Control Register all contain bits that are not initialised at power up, and so must be programmed before the video and sound macrocell will operate correctly.

Address(hex)	Register
0xxxxxxx	Video Palette
100000xx	Video Palette Address Register
20000000	RESERVED
300000xx	LCD Offset register 0
310000xx	LCD offset register 1
4xxxxxxx	Border Colour Register
5xxxxxxx	Cursor Palette logical colour 1
6xxxxxxx	Cursor Palette logical colour 2
7xxxxxxx	Cursor Palette logical colour 3
8000xxxx	Horizontal Cycle Register
8100xxxx	Horizontal Sync Width Register
8200xxxx	Horizontal Border Start Register
8300xxxx	Horizontal Display Start Register
8400xxxx	Horizontal Display End Register
8500xxxx	Horizontal Border End Register
8600xxxx	Horizontal Cursor Start Register
8700xxxx	Reserved
8800xxxx	Test Register
8C00xxxx	Test Register
9000xxxx	Vertical Cycle Register
9100xxxx	Vertical Sync Width Register
9200xxxx	Vertical Border Start Register
9300xxxx	Vertical Display Start Register
9400xxxx	Vertical Display End Register

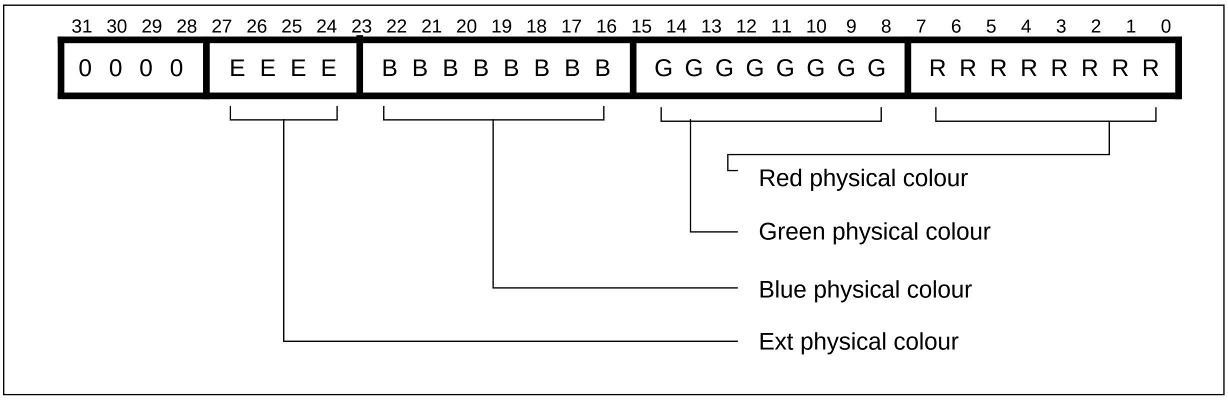
Table 9-1: The video and sound macrocell register allocation

Address(hex)	Register
9500xxxx	Vertical Border End Register
9600xxxx	Vertical Cursor Start Register
9700xxxx	Vertical Cursor End Register
9800xxxx	Test Register
9A00xxxx	Test Register
9C00xxxx	Test Register
A000000x : A700000x	Stereo Image Registers
B00000x	Sound Frequency Generator
B10000x	Sound Control Register
C00xxxxx	External Register
D000xxxx	Frequency Synthesis Register
E00xxxxx	Control Register
F000xxxx	Data Control Register

Table 9-1: The video and sound macrocell register allocation (Continued)

9.2 Video palette: Address 0x0

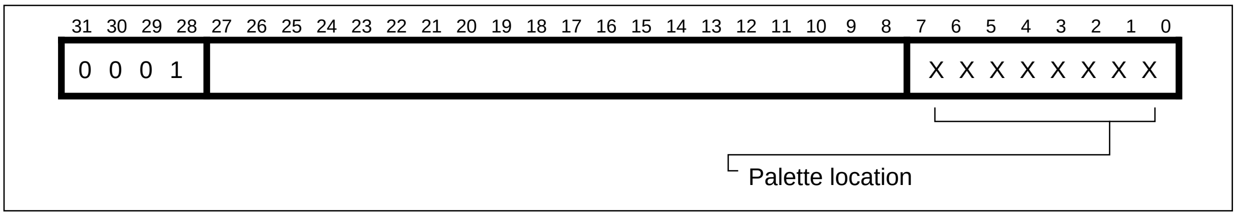
All entries of the video palette are written at address 0. In order to write any or all of the palette locations, the address pointer must first be written, as described below. The palette is programmed with a 28-bit word representing the physical data field.



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	E	E	E	E	B	B	B	B	B	B	B	B	G	G	G	G	G	G	G	G	R	R	R	R	R	R	R	R

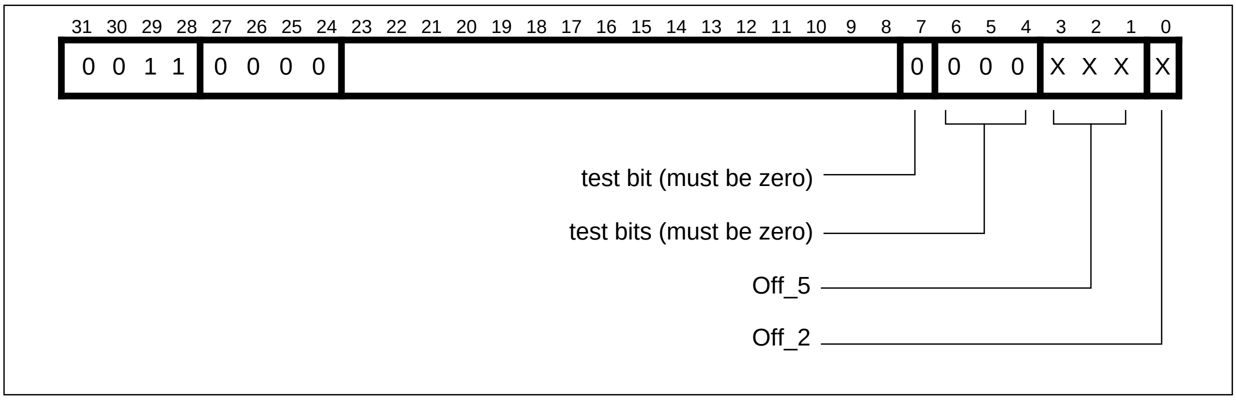
9.3 Video palette address pointer: Address 0x1

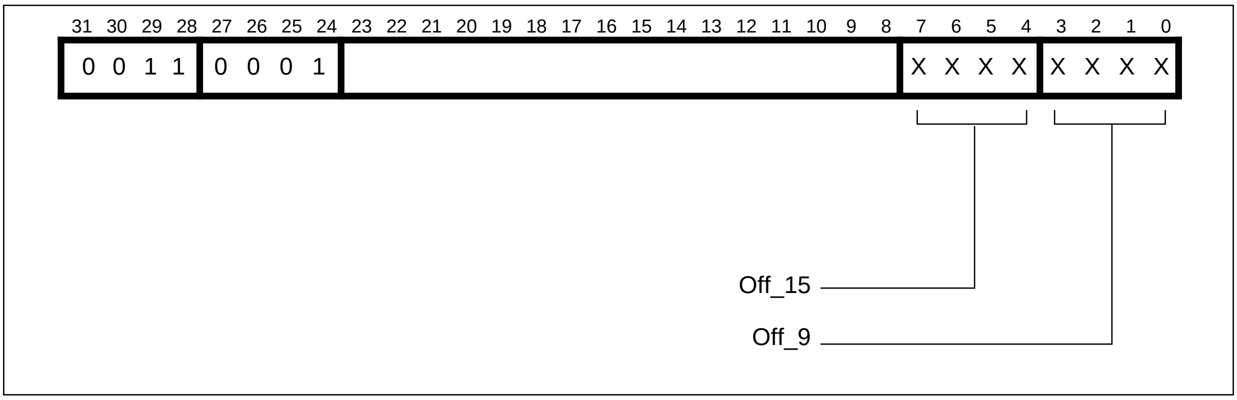
The address pointer is programmed at address 1, and it may be programmed to any value from 0 to 255. The first write to the palette will then occur at this location, and the address pointer will post-increment so that the next palette write will occur to the following location. The counter will wrap around from 255 to 0. Once the address pointer has been written, any number of palette locations can be programmed, and the pointer can be reprogrammed at any time if only part of the whole palette is to be updated.



9.4 LCD offset registers: Addresses 0x30 and 0x31

These two, 8-bit registers define the offsets required for driving a dual panel LCD screen. Register 0 defines the offsets for the five and two frame duty cycle grey scales, as well as reset and test mode bits. Register 1 defines the offsets for the nine and fifteen frame duty cycle grey scales.





The registers values are dependent upon the size of the LCD screen to be driven, and are calculated in the following way:

Off_15 = (3xL + 8) mod 15
Off_9 = (7xL + 4) mod 9
Off_5 = (1xL + 3) mod 5
Off_2 = 0

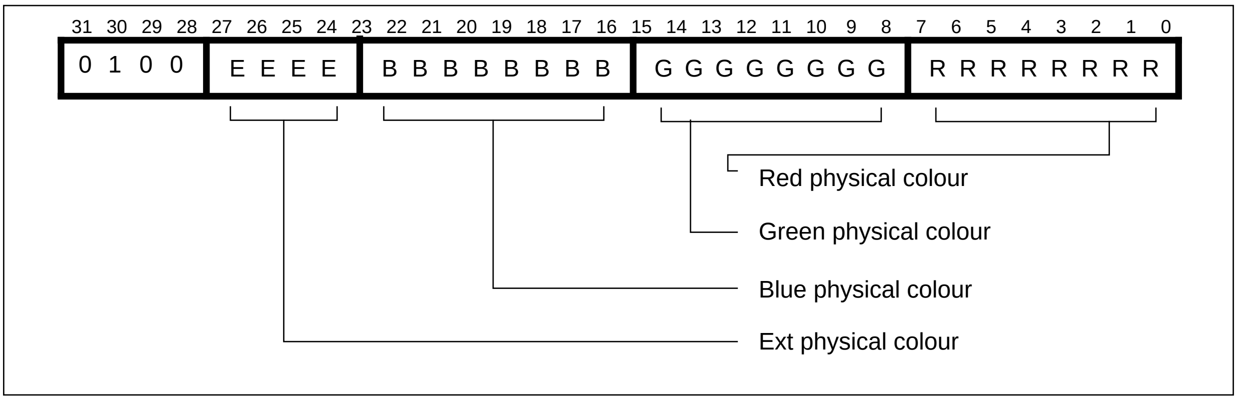
Where L is the number of lines in the upper panel of the dual panel LCD screen.

Bits 7-4 of register 0 are only used in test mode, and must all be set to zero in normal operation.

msel[2:0] are test bits and should be programmed LOW.

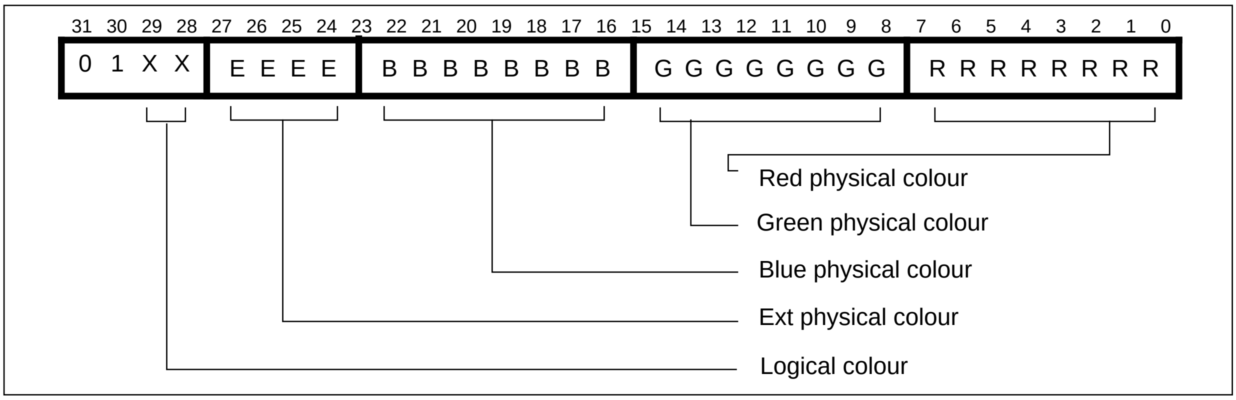
9.5 Border colour register: Address 0x4

This register defines the physical border colour, and is programmed with a 28-bit word. Note that this register is programmed directly, independent of the value of the video palette address pointer.



9.6 Cursor palette: Addresses 0x5-0x7

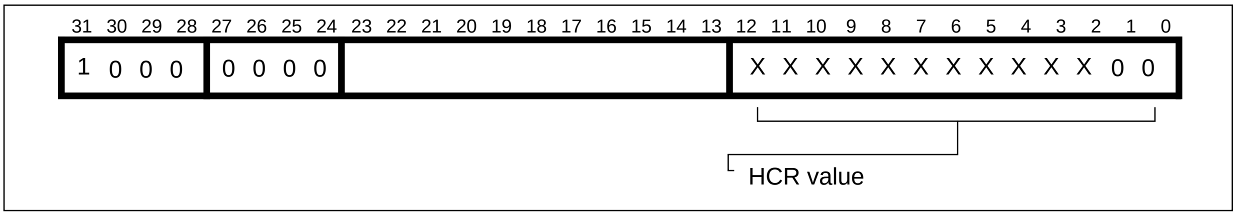
These three registers are programmed with the physical colour of the three logical cursor colours. Note that cursor logical colour 00 is defined as being transparent (i.e. no cursor display), and its location is used for the Border Colour Register above.



9.7 Horizontal cycle register (HCR): Address 0x80

This register defines the period, in pixels, of the horizontal scan, i.e. display time + retrace time.

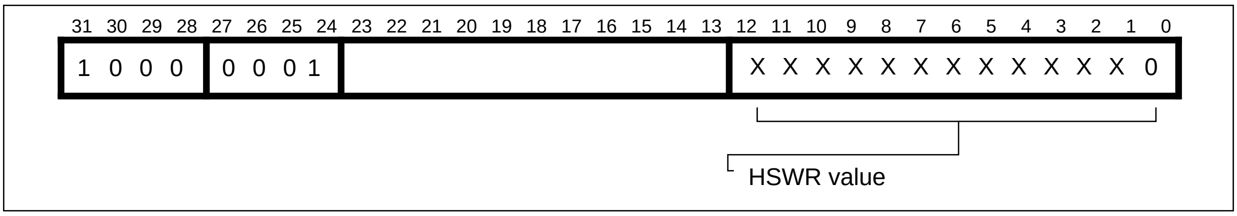
This is a 14-bit register of which the bottom 2 bits must be programmed to 0. If N pixels are required in the horizontal scan period, then value (N-8) should be programmed into the HCR. (N must be a multiple of 4).



9.8 Horizontal sync width register (HSWR): Address 0x81

This register defines the period, in pixels, of the **HSYNC** pulse.

This is a 14-bit register of which the bottom bit must be programmed to 0. If N pixels are required in the **HSYNC** pulse, then value (N-8) should be programmed into the HSWR. (N must be a multiple of 2).

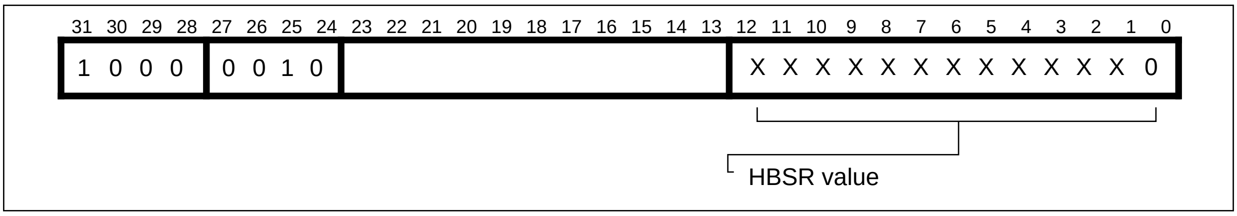


9.9 Horizontal border start register (HBSR): Address 0x82

This register defines the time, in pixels, from the start of the **HSYNC** pulse to the start of the border display.

This is a 14-bit register of which the bottom bit must be programmed to 0. If N pixels are required in this time, then value (N-12) should be programmed into the HBSR. (N must be a multiple of 2).

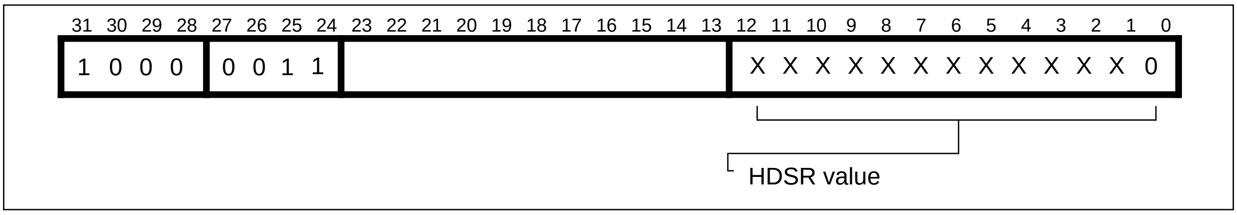
Note that this register must always be programmed, even when a border is not required. If a border is not required, then the value in the HBSR must be such as to start the border in the same place as the display start. i.e. $N_{HBSR}= N_{HDSR}$.



9.10 Horizontal display start register (HDSR): Address 0x83

This register defines the time, in pixels, from the start of the **HSYNC** pulse to the start of the video display.

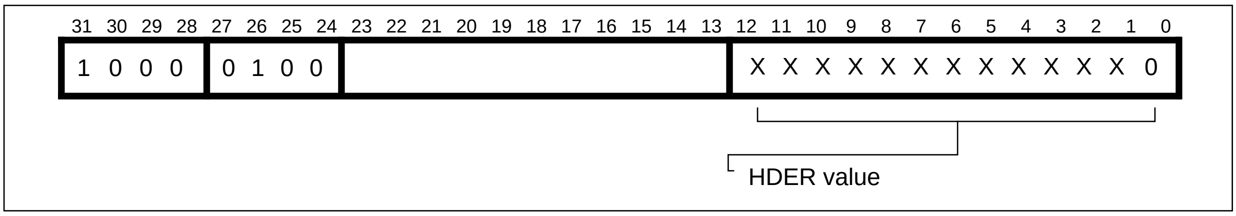
This is a 14-bit register of which the bottom bit must be programmed to 0. If N pixels are required in this time, then value (N-18) should be programmed into the HBSR. (N must be a multiple of 2).



9.11 Horizontal display end register (HDER): Address 0x84

This register defines the time, in pixels, from the start of the **HSYNC** pulse to the end of the video display. (i.e. the first pixel which is not display).

This is a 14-bit register of which the bottom bit must be programmed to 0. If N pixels are required in this time, then value (N-18) should be programmed into the HBER. (N must be a multiple of 2)

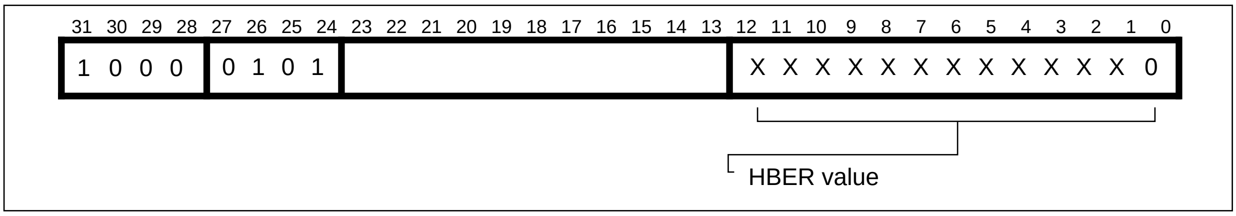


9.12 Horizontal border end register (HBER): Address 0x85

This register defines the time, in pixels, from the start of the **HSYNC** pulse to the end of the border display. (i.e. the first pixel which is not border).

This is a 14-bit register of which the bottom bit must be programmed to 0. If N pixels are required in this time, then value (N-12) should be programmed into the HBER. (N must be a multiple of 2).

Again, if no border is required, this register must still be programmed such that $N_{HBER} = N_{HDER}$.

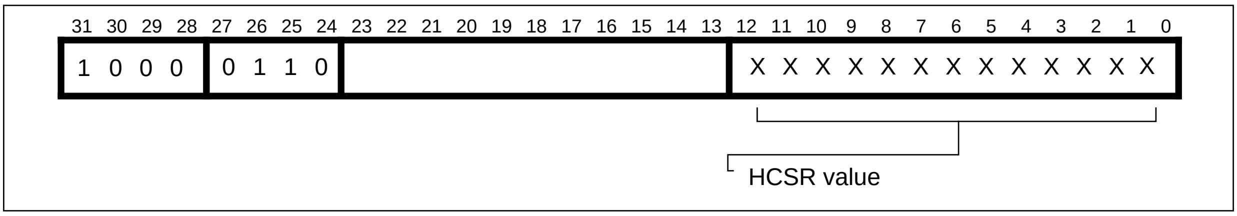


9.13 Horizontal cursor start register (HCSR): Address 0x86

This register defines the time, in pixels, from the start of the **HSYNC** pulse to the start of the cursor display.

This is a 14-bit register of which all bits may be programmed. If N pixels are required in this time, then value (N-17) should be programmed into the HCSR. The cursor can thus be programmed to start on any pixel. In HiRes mode, the cursor can still only be programmed to start on a normal pixel boundary. However, because the resolution of the cursor can be defined to a micro-pixel, by using different cursor images it is possible to position the cursor to any micro-pixel.

Note that only the cursor start position needs to be defined, as the cursor is automatically disabled after 32 pixels in normal mode, or 16 pixels in HiRes mode. If a cursor smaller than this is required, then the remaining bits in the cursor pattern should be programmed to logical colour 00 (transparent).



9.14 Horizontal interlace register (HIR): Address 0x87

Address 87H is reserved. Do not attempt to program this register.

9.15 Horizontal test registers: Addresses 0x88 & 0x8H

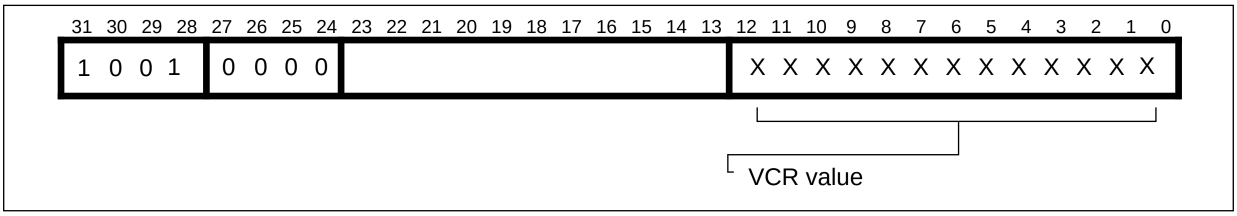
Two registers are provided for testing the chip in production. Neither of these registers are intended to be used during normal operation of the device.

9.16 Vertical cycle register (VCR): Address 0x90

This 13-bit register defines the period, in units of a raster, of the vertical scan. i.e. display time + flyback time.

If N rasters are required in a complete frame, then value (N-2) should be programmed into the VCR.

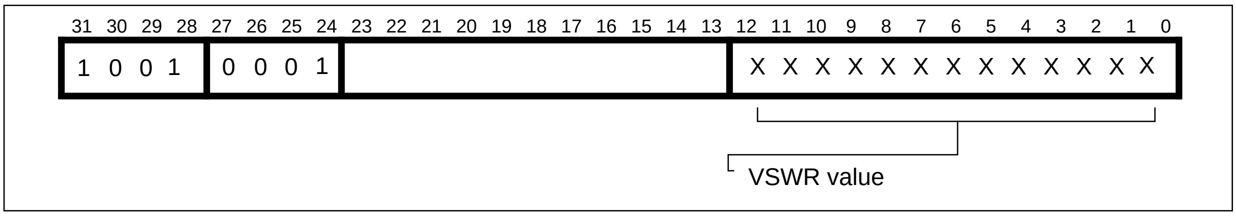
If an interlaced display is selected, (N-3)/2 must be programmed into the VCR. [N must be odd]. Here N is still the number of rasters in a complete frame, *not* a field.



9.17 Vertical sync width register (VSWR): Address 0x91

This 13-bit register defines the width, in units of a raster, of the **VSYNC** pulse.

If N rasters are required in the **VSYNC** pulse, then value (N - 2) should be programmed into the VSWR. The minimum value allowed for N is 2.

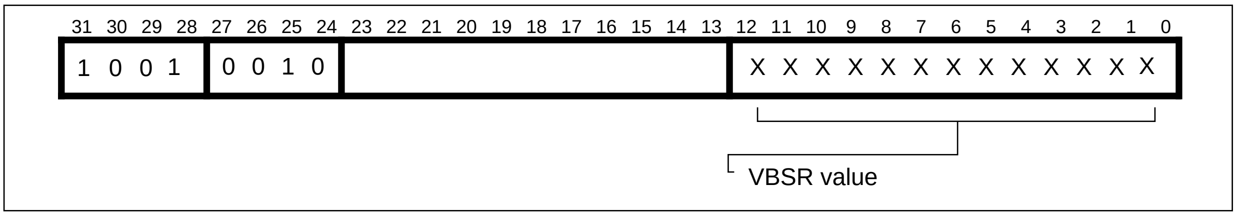


9.18 Vertical border start register (VBSR): Address 0x92

This 13-bit register defines the time, in units of a raster, from the start of the **VSYNC** pulse to the start of the border display.

If N rasters are required in this time, then value (N-1) should be programmed into the VBSR.

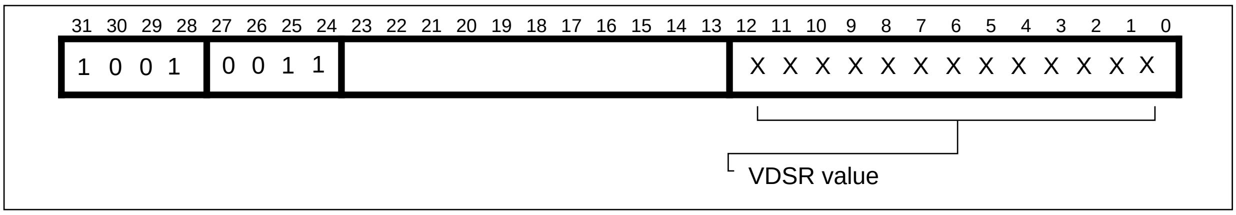
If no border is required, this register must still be programmed, in this case to the same value as the VDSR.



9.19 Vertical display start register (VDSR): Address 0x93

This 13-bit register defines the time, in units of a raster, from the start of the **VSYNC** pulse to the start of the video display.

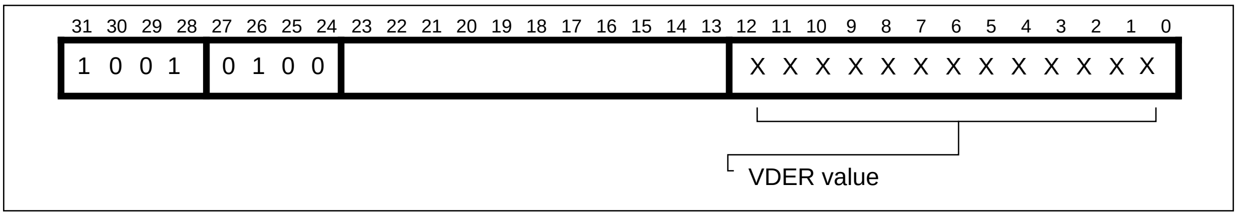
If N rasters are required in this time, then value (N-1) should be programmed into the VDSR.



9.20 Vertical display end register (VDER): Address 0x94

This 13-bit register defines the time, in units of a raster, from the start of the **VSYNC** pulse to the end of the video display. (i.e. the first raster on which the display is *not* present).

If N rasters are required in this time, then value (N-1) should be programmed into the VDER.

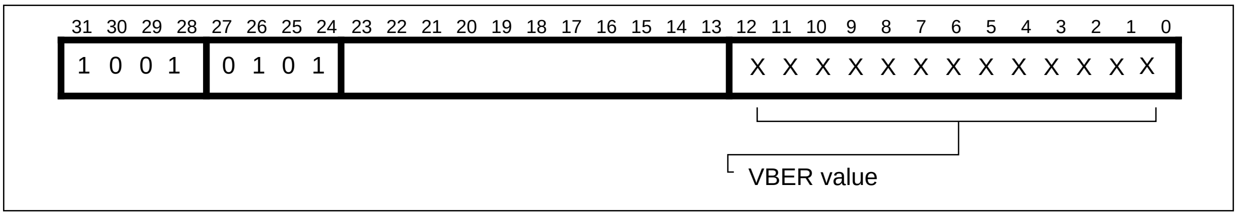


9.21 Vertical border end register (VBER): Address 0x95

This 13-bit register defines the time, in units of a raster, from the start of the **VSYNC** pulse to the end of the border display. (i.e. the first raster on which the border is not present).

If N rasters are required in this time, then value (N-1) should be programmed into the VBER.

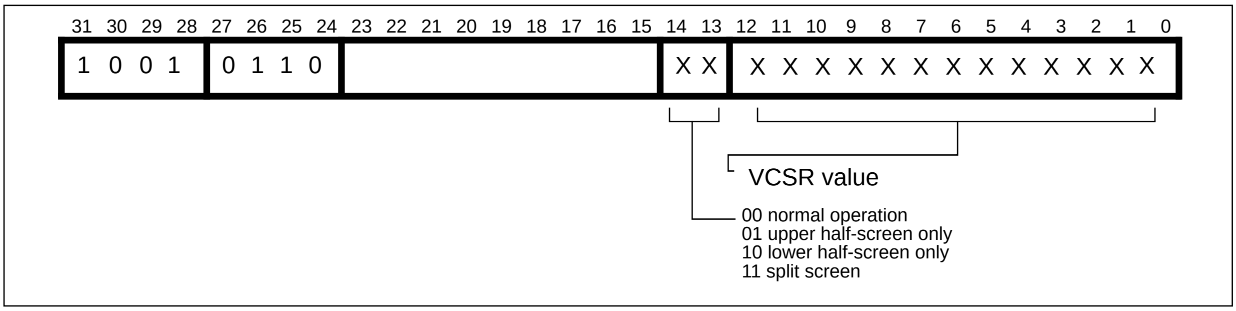
If no border is required, then this register must be programmed to the same value as the VDER.



9.22 Vertical cursor start register (VCSR): Address 0x96

This is a 15-bit register. The lower 13 bits define the time, in units of a raster, from the start of the **VSYNC** pulse to the start of the cursor display. If N rasters are required in this time, then value (N-1) should be programmed into the VCSR. The upper 2 bits are used to control the display of the cursor in duplex LCD mode. They should be programmed to zero in all other modes.

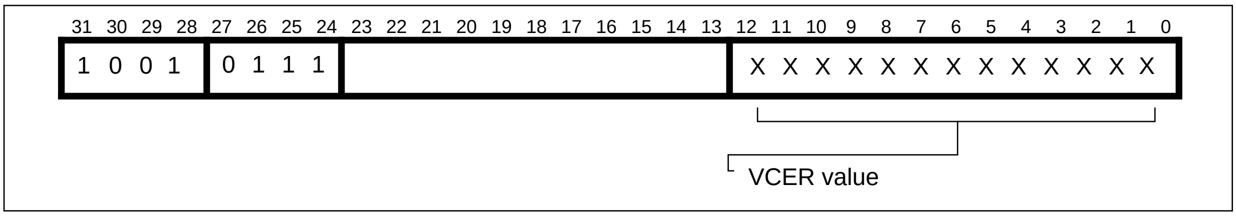
When the upper 2 bits are programmed to be 11 (split screen) the meaning of VCSR and VCER are altered as follows. The cursor is displayed in the lower half-screen only from the value of VDSR to the value of VCSR, and again in the upper half screen only from the value of VCER to the value of VDER. This allows a cursor to be positioned across the boundary of the upper and lower half screens of an LCD.



9.23 Vertical cursor end register (VCER): Address 0x97

This 13-bit register defines the time, in units of a raster, from the start of the **VSYNC** pulse to the end of the cursor display. (i.e. the first raster on which the cursor is not present).

If N rasters are required in this time, then value (N-1) should be programmed into the VCER.

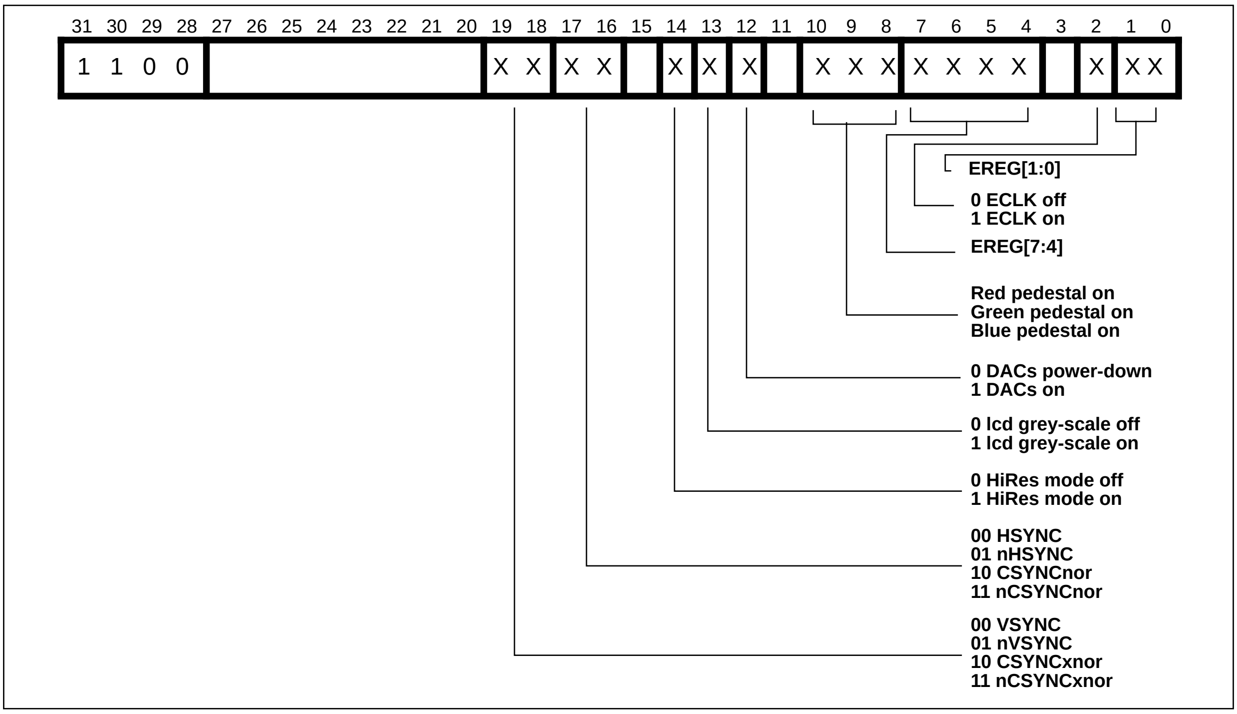


9.24 Vertical test registers: Addresses 0x98, 0x9A & 0x9C

Three registers are provided for testing the chip in production. None of these registers are intended to be used during normal operation of the device.

9.25 External register (ereg): Address 0xC

This register contains the control bits for the external functions of video and sound macrocell. In particular it controls the DACs, the configuration of the External Port **ED[7:0]**, and the configuration of the sync lines.



EREG[1:0] are internally mapped to drive esel[1:0] by ARM7500.

EREG[7:4] are exported from the chip on **ED[7:4]** if EREG[1:0]**=**3. Refer to ➲*11.6 External support* on page 11-9.

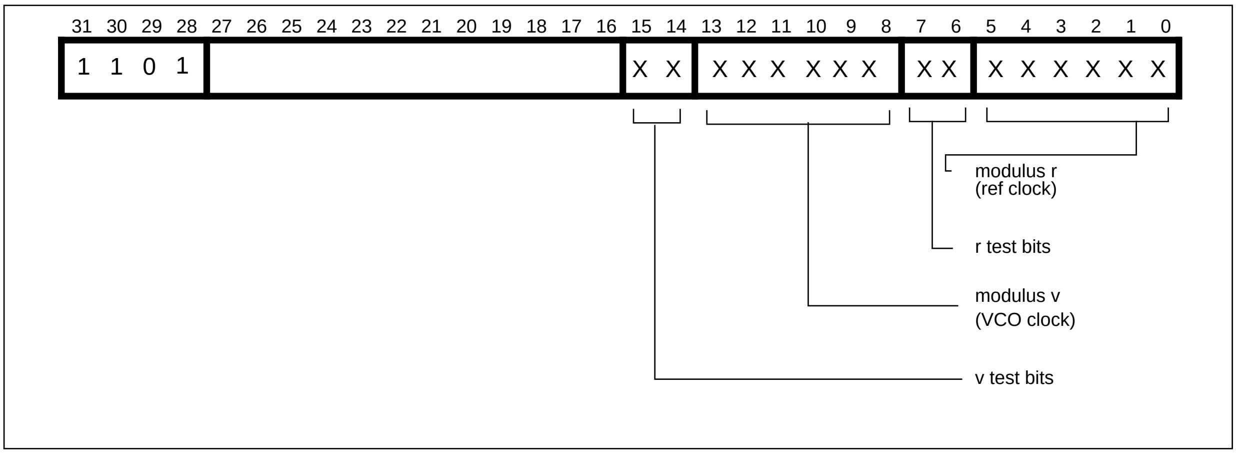
The use of pedon[2:0] and DAC is defined in ➲*11.7 Analog outputs* on page 11-11.

The uses of lcd and hrm are defined in ➲*11.6 External support* on page 11-9.

ARM7500 can export a variety of sync configurations on the pins **HSYNC** and **VSYNC**, as specified by the bits 16-17 and 18-19 respectively. For further explanation see ➲*11.6.3 Vertical and horizontal synchronisation* on page 11-11.

9.26 Frequency synthesizer register (fsynreg): Address 0xD

ARM7500 is able to drive a VCO to provide a suitable input frequency for the pixel clock derived from a reference clock. This is achieved by dividing the reference clock by modulus r, and the VCO clock by modulus v, and comparing the resulting frequencies. Refer to ➲*11.1 Pixel clock* on page 11-2 for a more detailed explanation. The two moduli, r and v are each 6-bit values, and are programmed in this register.



Associated with each counter are 2 test bits which should normally be programmed to 0.

Setting bit[6] forces the phase comparator HIGH, which drives **PCOMP** HIGH.

Setting bit[7] clears the r-modulus counter.

Setting bit[14] forces the phase comparator LOW, which drives **PCOMP** LOW.

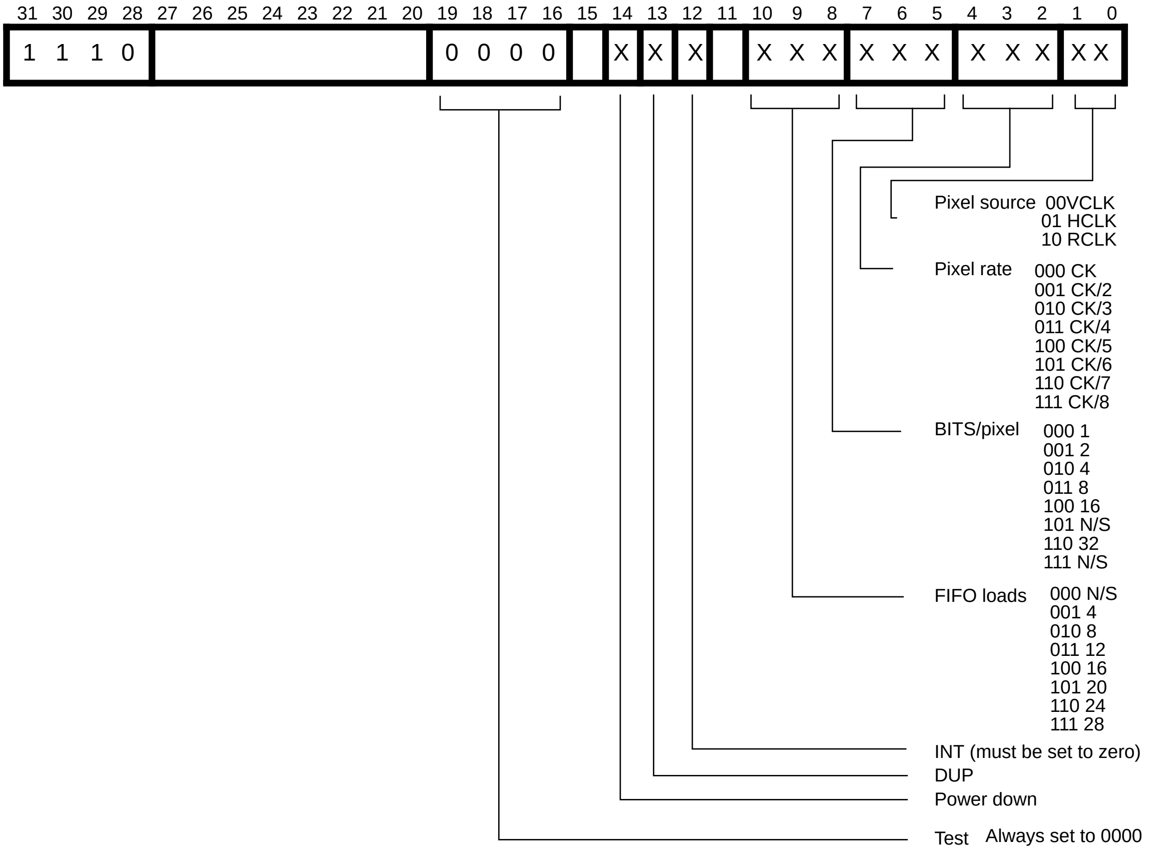
Setting bit[15] clears the v-modulus counter.

To reduce power consumption, this register should be programmed with large values when the frequency synthesizer is not in use. In particular, bits [6] and [14] should not be set at the same time.

To get a modulus of r, then value (r-1) should be programmed into the fsynreg. Likewise for the v-modulus.

9.27 Control register (conreg): Address 0xE

The main control register determines the basic operation of the chip. In particular the pixel clock source, the pixel rate, the number of bits/pixel, the control of the video FIFO, and the data format are programmed here. In addition there is a 4-bit test register which must be programmed to zero for normal operation.



Note *The INT bit should always be set to zero.*

The pixel clock (pixclk) is selected from one of 3 sources, corresponding to the respective input pins, and the selected clock is then fed through a prescaler as defined by the 3 bits conreg[4:2]. The output of this prescaler is the actual pixel clock. See *Chapter 11: Video Features* for more detail.

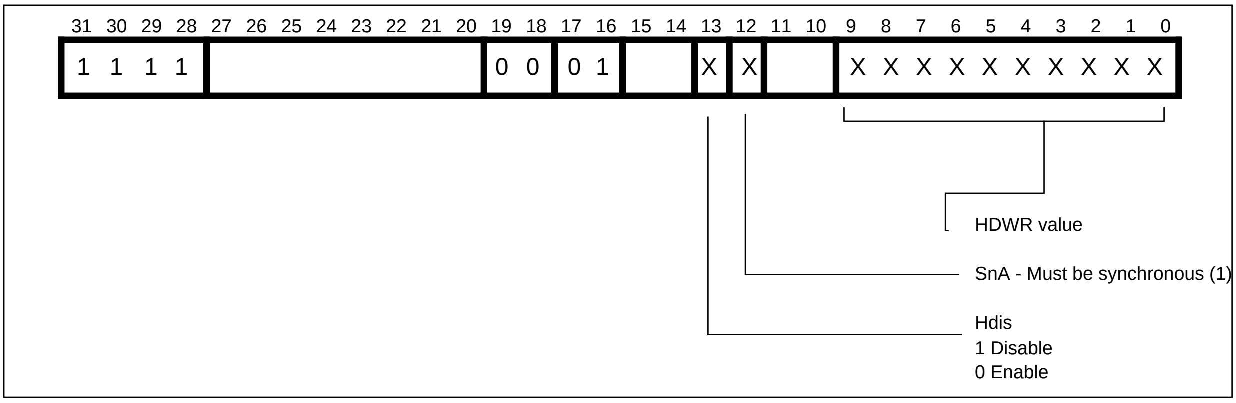
The Video FIFO can be programmed to have any number of quad words loaded into it at the start of display. The value chosen should take into account the bandwidth of the display as well as the latency of the DMA subsystem. Refer to *Chapter 10: Video Macrocell Interface* before programming these values.

Setting the dup bit configures the display for dual-panel LCDs. This is described further in *Chapter 11: Video Features* .

Note that after a reset the Control Register should be the first register programmed. The Powerdown bit (14) must immediately be programmed LOW. The test registers bits (16 to 19) also should be programmed LOW, as any other setting will inhibit normal operation.

The video macrocell uses dynamic logic structures for maximum performance. When the powerdown bit is set HIGH, the main video data path will be set into a state where it will not consume static current. This must be done before the ARM7500 is set into STOP mode.

9.28 Data control register (DCTL): Address 0xF

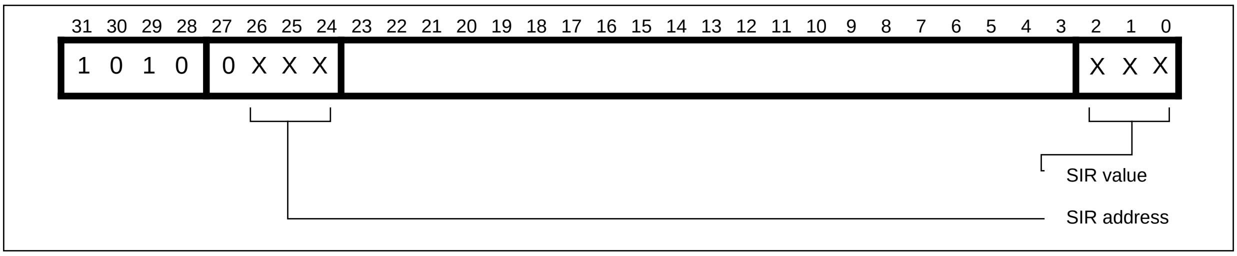


The horizontal display width is also defined in this register, and should be programmed to be the number of words of data in a displayed raster. It must be programmed in most configurations of the device, as it inhibits a DMA request near the end of a raster, when there are enough words in the video FIFO for that raster. The request is uninhibited after the **HSYNC** at the start of the next raster. When driving a dual panel LCD screen, this register must be programmed with twice the number of words in a displayed raster. Hdis should normally be programmed to zero. If Hdis is programmed to one, the inhibition of DMA requests is disabled.

Note *Bits 19:16 MUST be set to 0001 (binary).*

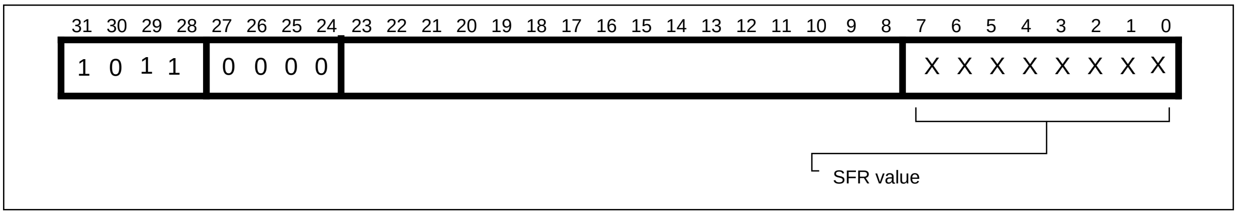
9.29 Stereo image register 0-7: Addresses 0xA0-0xA7

These are eight, 3-bit registers which define the stereo position for the eight possible channels, as defined in ➲*Chapter 12: Sound Features* .



9.30 Sound frequency register: Address 0xB0

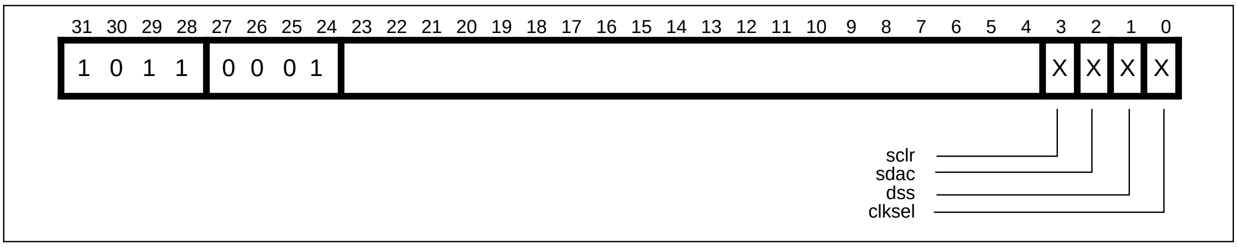
This 8-bit register specifies the byte sample rate of the sound data. It is defined in units of 1µS. See ➲*Chapter 12: Sound Features* for more detail.



If a sample rate of N µs is required, then N-2 should be programmed into the SFR. N may take any value between 3 and 256.

9.31 Sound control register: Address 0xB1

This is a 4-bit register which defines various control bits for the sound system.



Bit 3: SCLR — This bit should always be programmed LOW.

Bit 2: SDAC — When HIGH, the sound DACs are enabled. Two digital signals are also output from the chip in this mode. The first, **WS_LnR**, denotes whether the sound is for the left or right stereo channel. The other, **SD0_MUTE**, goes HIGH between samples, when the sound DACs are being muted to allow for settling. These two signals are

	intended to ease the connection of external audio processing systems, but for basic operation can be ignored. Note that these two signals have different functions when the serial sound interface is in use.
Bit 1: serial sound	This bit is used to select serial sound mode.
Bit 0: CLKSEL	This bit is used to select which clock is used in the sound system. When HIGH, the ARM7500’s internal 32MHz I/O reference clock is used, when LOW the optional sound clock is used.