7 ARM Processor MMU

This chapter describes the ARM Processor Memory Management Unit.

7.1	Introduction	7-2
7.2	MMU Program Accessible Registers	7-3
7.3	Address Translation	7-4
7.4	Translation Process	7-5
7.5	Translating Section References	7-8
7.6	Translating Small Page References	7-10
7.7	Translating Large Page References	7-11
7.8	MMU Faults and CPU Aborts	7-12
7.9	Fault Address and Fault Status Registers (FAR and FSR)	7-13
7.10	Domain Access Control	7-14
7.11	Fault Checking Sequence	7-15
7.12	Interaction of the MMU, IDC and Write Buffer	7-18
7.13	Effect of Reset	7-19

7.1 Introduction

The Memory Management 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 is shown in *Figure 7-1: MMU register summary* on page 7-3. A brief description of the registers is provided below. Each register will be discussed in more detail within the section that describes its use.

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

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.

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).

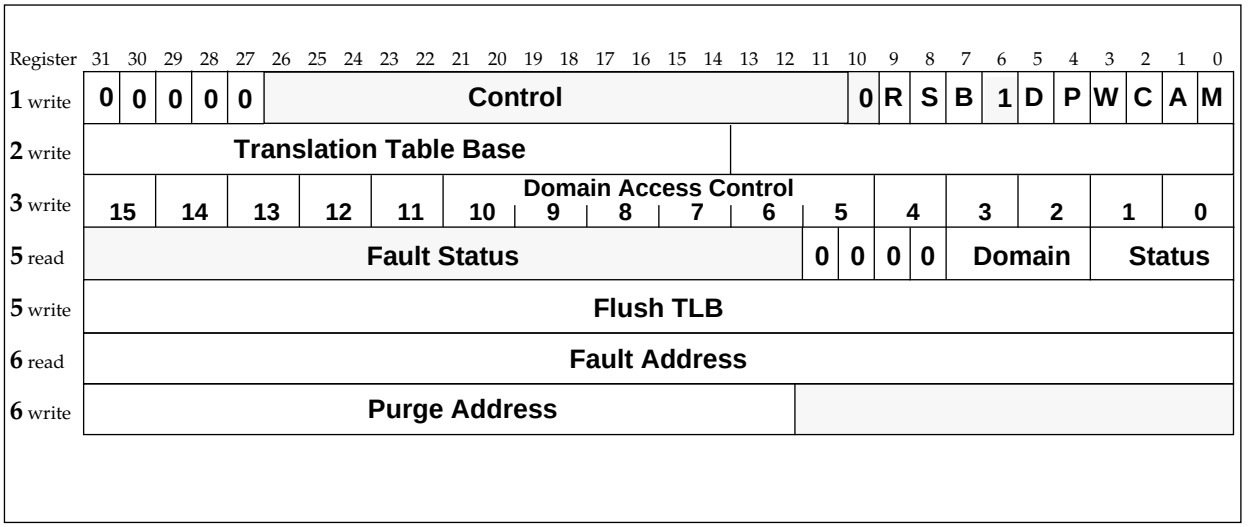


Figure 7-1: MMU register summary

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

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 7-13. A write to this register flushes the TLB.

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.

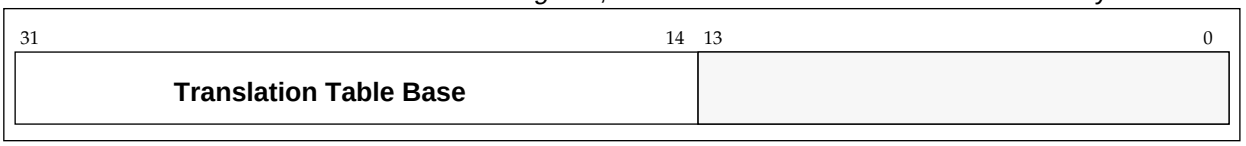


Figure 7-2: Translation table base register

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*. 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)

.

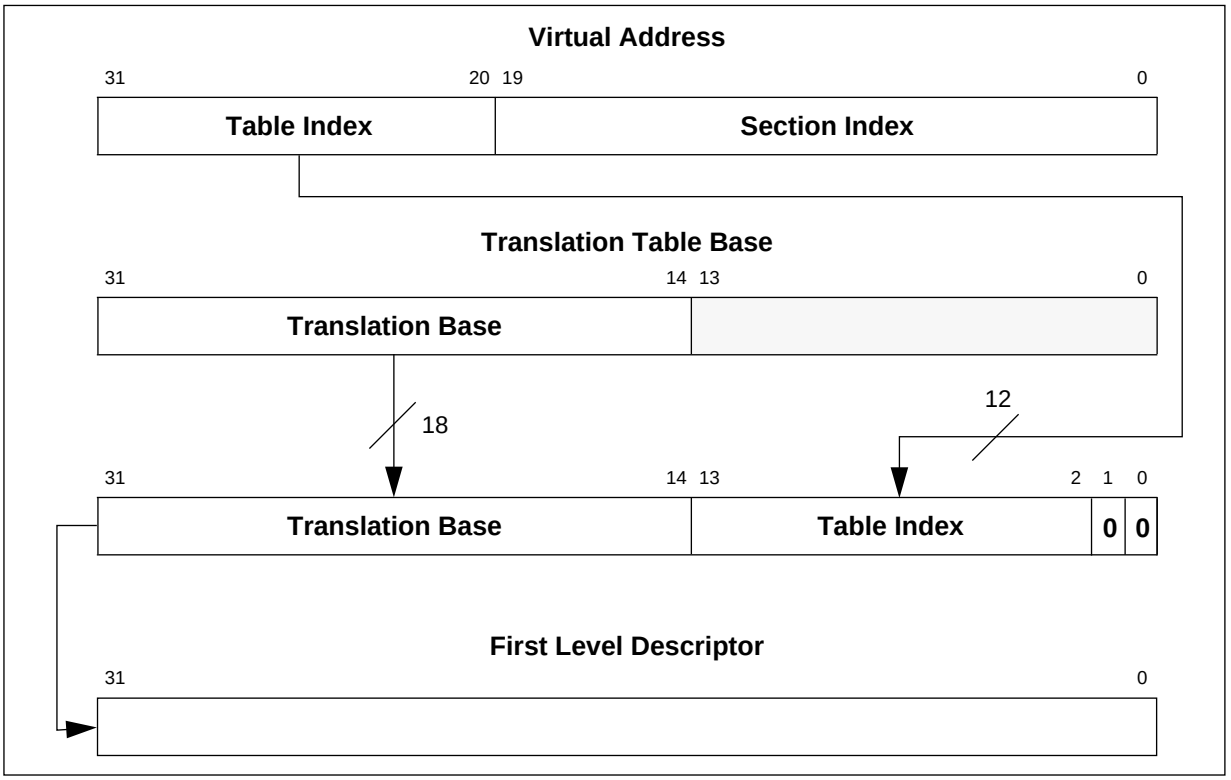


Figure 7-3: Accessing the translation table first level descriptors

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.

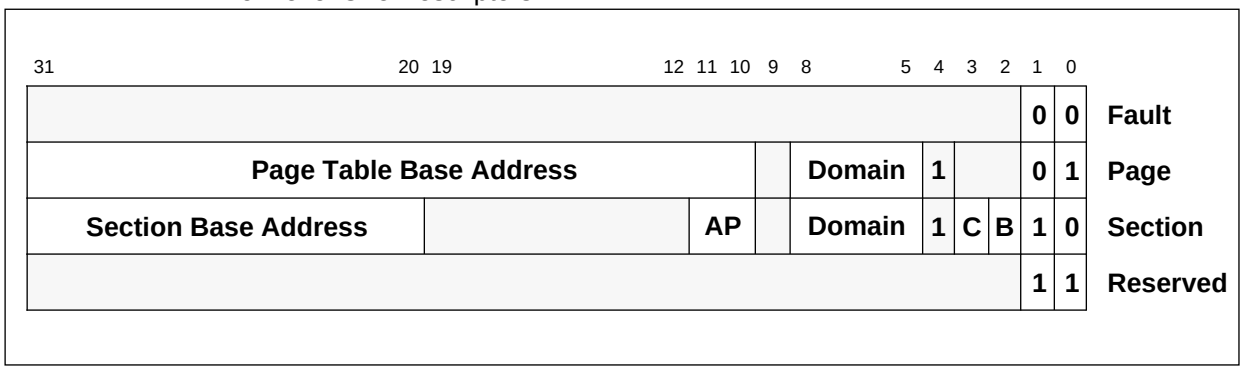


Figure 7-4: Level one descriptors

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, and 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	Permissions Supervisor	User	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.

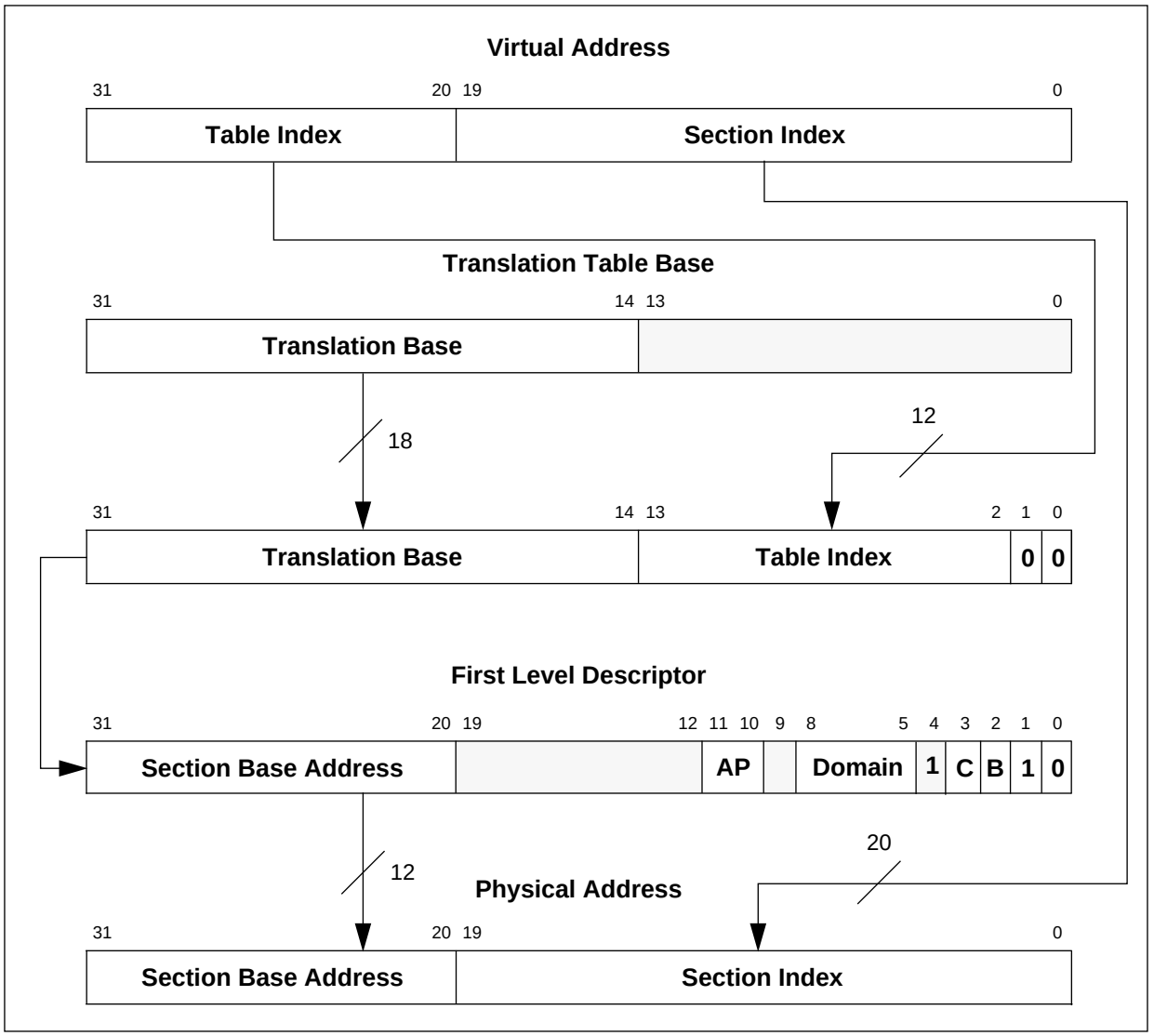


Figure 7-5: Section translation

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* on page 7-10, 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. The figure below shows the format of Level Two Descriptors

.

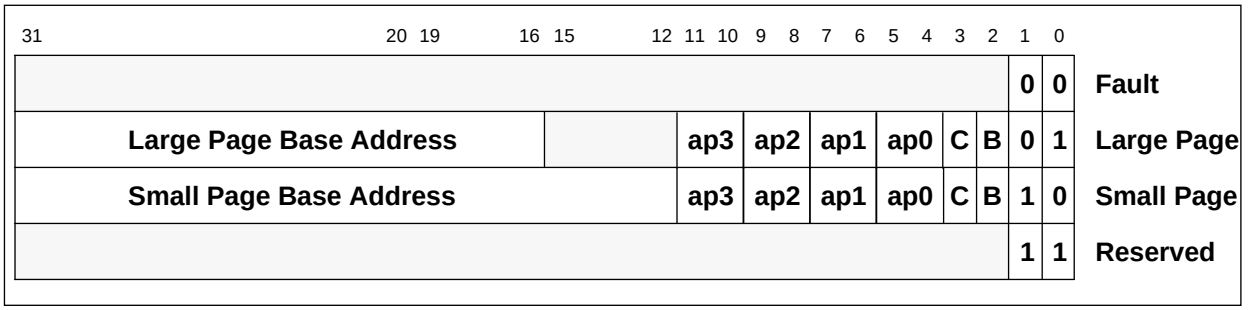


Figure 7-6: Page table entry (Level Two descriptor)

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).

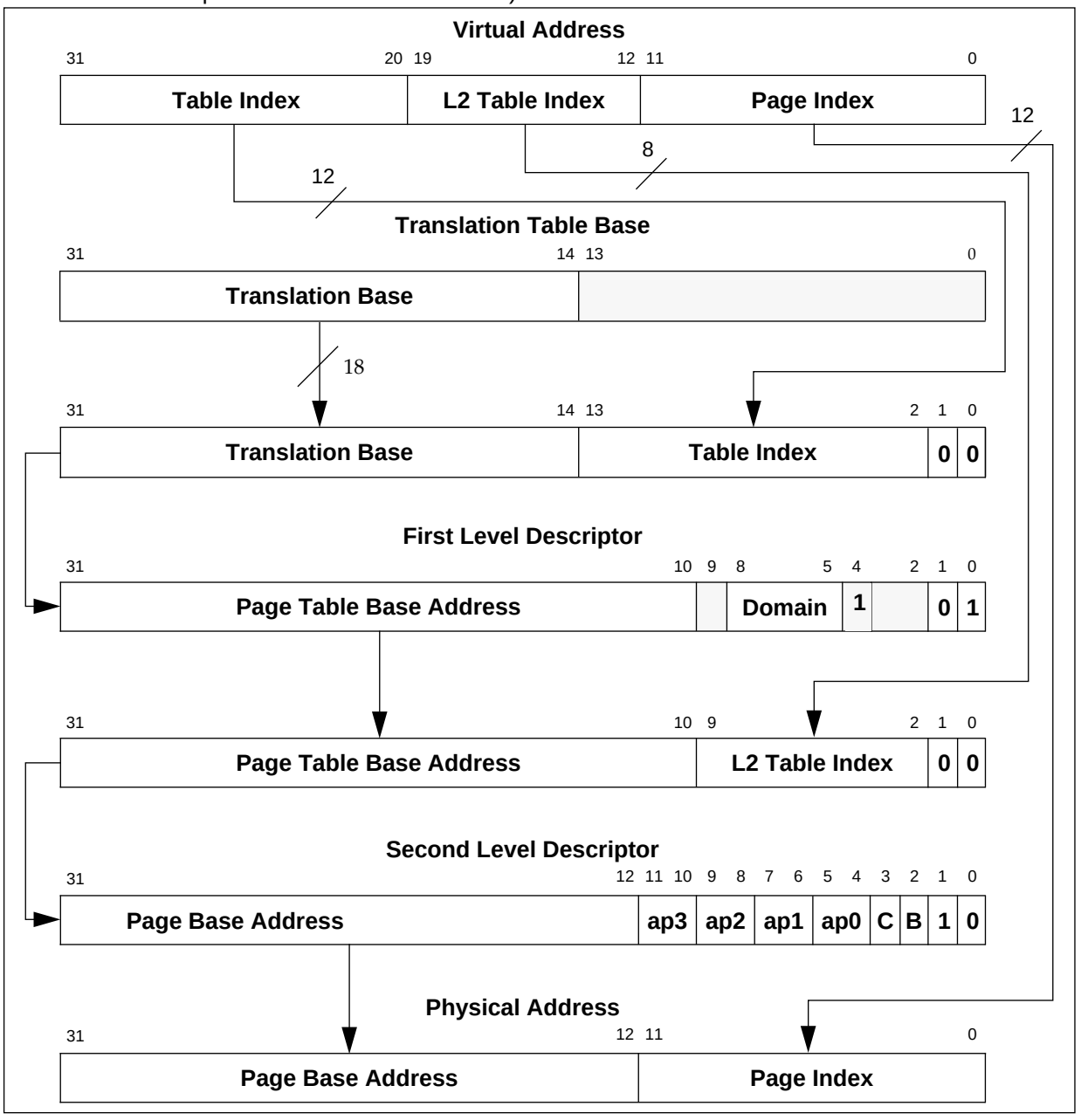


Figure 7-7: Small page translation

7.7 Translating Large Page References

Figure 7-8: Large page translation illustrates the complete translation sequence for a 64 kB 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.

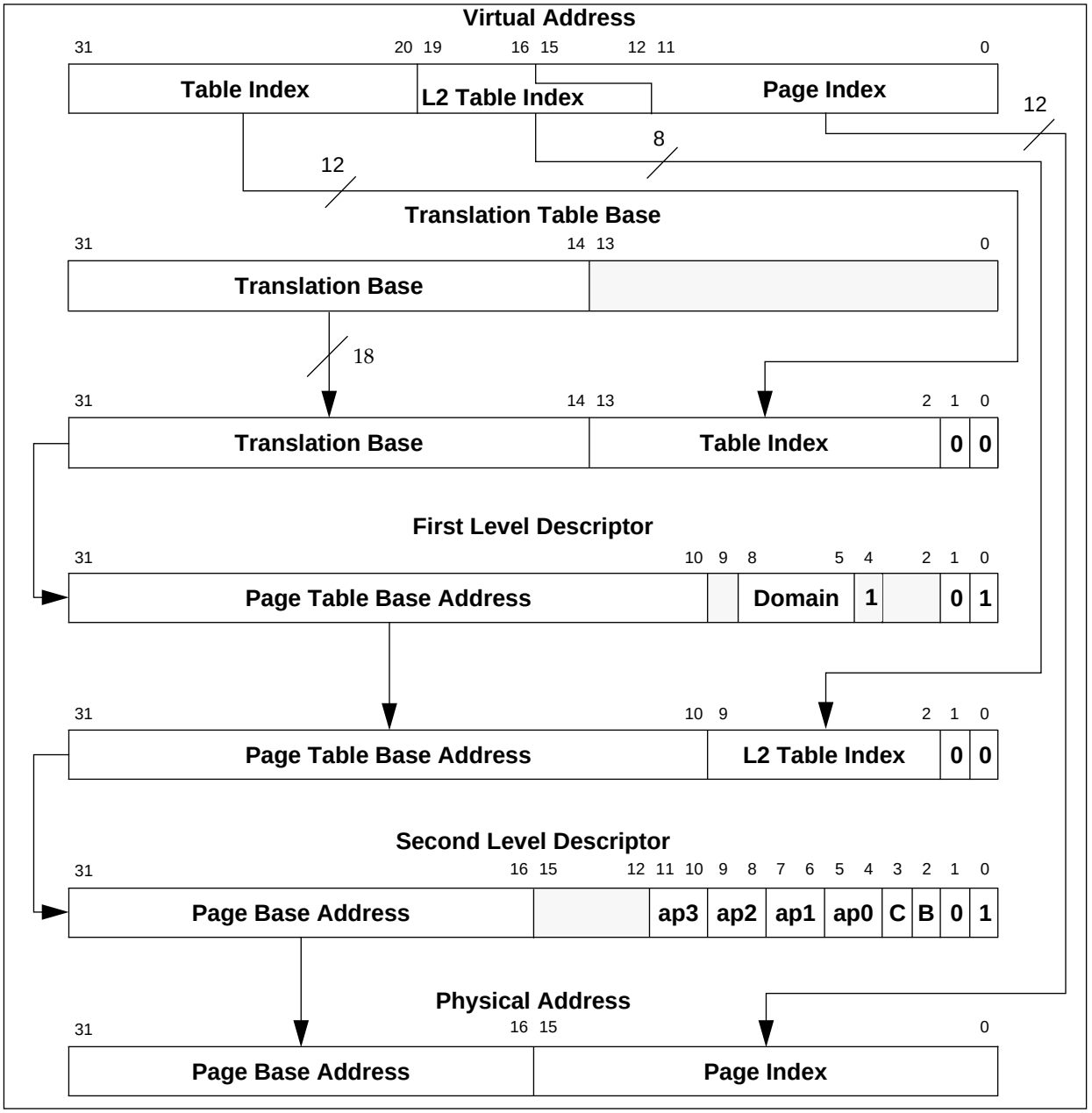


Figure 7-8: Large page translation

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.

If the ARM Processor is operating in fastbus mode an internally aborting access may cause the address on the external address bus to change, even though the external bus cycle has been cancelled. The address that is placed on the bus will be the translation of the address that caused the abort, though in the case of the a Translation Fault the value of this address will be undefined. No memory access will be performed to this address.

7.9 Fault Address and Fault Status Registers (FAR and 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 7-13.

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 (ie. 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.

	Source		FS[3210]	Domain[3:0]	FAR
Highest	Alignment		00x1	x	valid
	Bus Error (translation)	level1 level2	1100 1110	x valid	valid valid
	Translation	Section Page	0101 0111	Note 2 valid	valid valid
	Domain	Section Page	1001 1011	valid valid	valid valid
	Permission	Section Page	1101 1111	valid valid	valid valid
	Bus Error (linefetch)	Section Page	0100 0110	valid valid	valid valid
Lowest	Bus Error (other)	Section Page	1000 1010	valid valid	valid valid

Table 7-4: Priority encoding of fault status

x is undefined, and may read as 0 or 1

Notes

1 Any abort masked by the priority encoding may be regenerated by fixing the primary abort and restarting the instruction.

2 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* on page 7-14 illustrates how the 32 bits of the register are allocated to define the sixteen 2-bit domains.

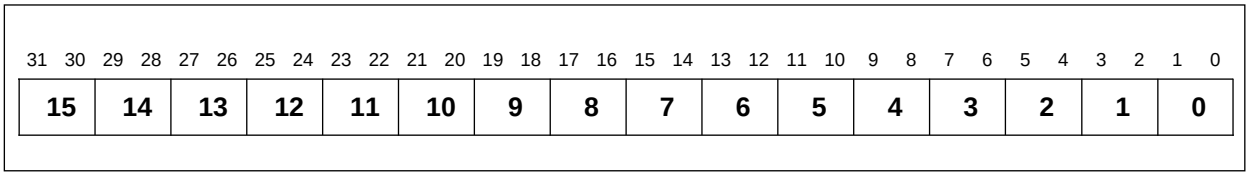


Figure 7-9: Domain access control register format

◆*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.

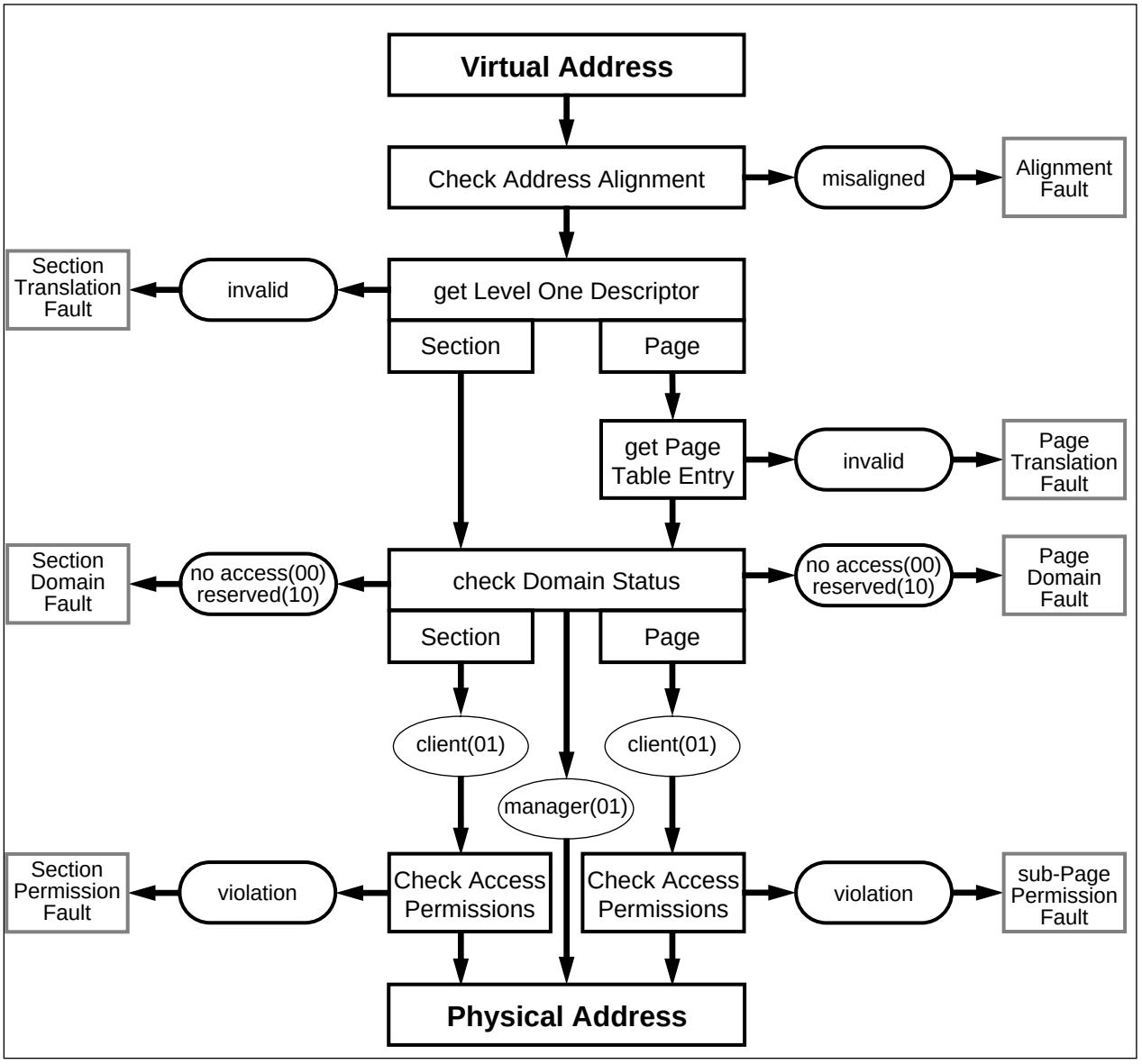


Figure 7-10: Sequence for checking faults

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 Interaction of the MMU, IDC and Write Buffer

The MMU, IDC and WB may be enabled/disabled independently. However, in order for the write buffer or the cache to be enabled the MMU must also be enabled. 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 write buffer 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 *4.7 Reset* on page 4-16.

8 ARM7100 Programmer's Model

This chapter details the programmable registers for ARM7100.

8.1	Introduction	8-2
8.2	Summary of Registers	8-3
8.3	Register Descriptions	8-5

Preliminary

8.1 Introduction

ARM7100 contains internal programmable registers in addition to those in the ARM processor.

The registers internal to ARM7100 are all programmed by writing to memory locations 8000.0000 to 8000.FFFF. Accessing memory in this range will not cause any external bus activity unless broadcast mode is enabled. Any access to the undefined range from 8000.1000 to C000.0000 will have no effect.

Writes to bits that are not explicitly defined in the internal area are legal and will have no effect. Reads from bits not explicitly defined in the internal area are legal but will read undefined values.

It is only possible to access internal addresses as 32-bit words and they are always on a word boundary, except for the PIO port registers which can be accessed as bytes. Each internal register is valid for 256 bytes, since address bits in the range A[0:5] are not decoded, for example, the SYSFLG register appears at locations 8000.0140 to 8000.017C. The PIO port registers are byte wide but can be accessed as a word. These registers additionally decode A0 and A1.

8.2 Summary of Registers

Key

✓ can write/read

✗ do not write/read

Name	Address	Size	Read	Write	Function
PADR	8000.0000	8	✓	✓	Port A data register.
PBDR	8000.0001	8	✓	✓	Port B data register.
PCDR	8000.0002	8	✓	✓	Port C data register.
PDDR	8000.0003	8	✓	✓	Port D data register.
PADDR	8000.0040	8	✓	✓	Port A data direction register.
PBDDR	8000.0041	8	✓	✓	Port B data direction register.
PCDDR	8000.0042	8	✓	✓	Port C data direction register.
PDDDR	8000.0043	8	✓	✓	Port D data direction register.
PEDR	8000.0080	4	✓	✓	Port E data register.
PEDDR	8000.00C0	4	✓	✓	Port E data direction register.
SYSCON	8000.0100	32	✓	✓	System control register
SYSFLG	8000.0140	32	✓	✗	System status flags.
MEMCFG1	8000.0180	32	✓	✓	Expansion and ROM memory configuration register 1.
MEMCFG2	8000.01C0	32	✓	✓	Expansion and ROM memory configuration register 2.
DRFPR	8000.0200	8	✓	✓	DRAM refresh period register.
INTSR	8000.0240	16	✓	✗	Interrupt status register.
INTMR	8000.0280	16	✓	✓	Interrupt mask register.
LCDCON	8000.02C0	32	✓	✓	LCD control register.
TC1D	8000.0300	16	✓	✓	Read-write data to TC1.
TC2D	8000.0340	16	✓	✓	Read-write data to TC2.
RTCDR	8000.0380	32	✓	✓	Real time clock data register.
RTCMR	8000.03C0	32	✓	✓	Real time clock match register.
PMPCON	8000.0400	12	✓	✓	DC to DC pump control register.

Table 8-1: ARM7100 registers

Name	Address	Size	Read	Write	Function
CODR	8000.0440	8	✓	✓	CODEC data I/O register.
UARTDR	8000.0480	8	✓	✓	UART FIFO data register.
UBLCR	8000.04C0	32	✓	✓	UART bit rate and line control register.
SYNCIO	8000.0500	16	✓	✓	Synchronous serial I/O data register
PALLSW	8000.0540	32	✓	✓	Least significant 32-bit word of LCD palette register
PALMSW	8000.0580	32	✓	✓	Most significant 32-bit word of LCD palette register
STFCLR	8000.05C0	-	✗	✓	Write to clear all start up reason flags.
BLEOI	8000.0600	-	✗	✓	Write to clear battery low interrupt.
MCEOI	8000.0640	-	✗	✓	Write to clear **MEDCHG** interrupt.
TEOI	8000.0680	-	✗	✓	Write to clear tick and watchdog interrupt.
TC1EOI	8000.06C0	-	✗	✓	Write to clear TC1 interrupt.
TC2EOI	8000.0700	-	✗	✓	Write to clear TC2 interrupt.
RTCEOI	8000.0740	-	✗	✓	Write to clear RTC match interrupt.
UMSEOI	8000.0780	-	✗	✓	Write to clear UART modem status changed interrupt.
COEOI	8000.07C0	-	✗	✓	Write to clear CODEC sound interrupt
HALT	8000.0800	-	✗	✓	Write to enter idle state
STDBY	8000.0840	-	✗	✓	Write to enter standby state
Reserved	8000.0880 - 8000.0FFF	-			Write will have no effect, read is undefined

Table 8-1: ARM7100 registers (Continued)

8.3 Register Descriptions

All internal registers in ARM7100 are reset (cleared to zero) by a system reset (**nPOR**, **nRESET** or **nPWRFL** signals becoming active), except for the DRAM refresh period register (DRFPR) which is only reset by **nPOR** becoming active. This ensures that the contents of DRAM are preserved though a user reset or power fail condition. Additionally, the real time clock registers are only cleared by **nPOR**.

8.3.1 Port A data register (PADR)

Values written to this 8-bit read-write register are output on port A pins if the corresponding data direction bits are set HIGH (port output). Values read from this register reflect the external state of port A, not necessarily the value written to it. All bits are cleared by a system reset.

8.3.2 Port B data register (PBDR)

Values written to this 8-bit read-write register are output on port B pins if the corresponding data direction bits are set HIGH (port output). Values read from this register reflect the external state of port B, not necessarily the value written to it. All bits are cleared by a system reset.

8.3.3 Port C data register (PCDR)

Values written to this 8-bit read-write register are output on port C pins if the corresponding data direction bits are set LOW (port output). Values read from this register reflect the external state of port C, not necessarily the value written to it. All bits are cleared by a system reset.

8.3.4 Port D data register (PDDR)

Values written to this 8-bit read-write register are output on port D pins if the corresponding data direction bits are set LOW (port output). Values read from this register reflect the external state of port C, not necessarily the value written to it. All bits are cleared by a system reset.

8.3.5 Port A data direction register (PADDR)

Bits set in this 8-bit read-write register select the corresponding pin in port A to become an output. Clearing a bit sets the pin to input. All bits are cleared by a system reset.

8.3.6 Port B data direction register (PBDDR)

Bits set in this 8-bit read-write register select the corresponding pin in port B to become an output. Clearing a bit sets the pin to input. All bits are cleared by a system reset.

8.3.7 Port C data direction register (PCDDR)

Bits cleared in this 8-bit read-write register select the corresponding pin in port C to become an output. Setting a bit sets the pin to input. All bits are cleared by a system reset so that port C is output by default.

8.3.8 Port D data direction register (PDDDR)

Bits cleared in this 8-bit read-write register select the corresponding pin in port D to become an output, setting a bit sets the pin to input. All bits are cleared by a system reset so that port D is output by default.

8.3.9 Port E data register (PEDR)

Values written to this 4-bit read-write register will be output on port E pins if the corresponding data direction bits are set HIGH (port output). Values read from this register reflect the external state of port E, not necessarily the value written to it. All bits are cleared by a system reset.

8.3.10 Port E data direction register (PDDDR)

Bits set in this 4-bit read-write register will select the corresponding pin in port E to become an output, clearing bit sets the pin to input. All bits are cleared by a system reset so that port E is input by default.

8.3.11 The system control register (SYSCON)

The system control register is a 21-bit read /write register controlling all the general configuration of ARM7100 as well as operating modes for peripheral devices. All bits in this register are cleared by a system reset.

The bits in the system control register SYSCON are shown in *Figure 8-1: The system control register*.

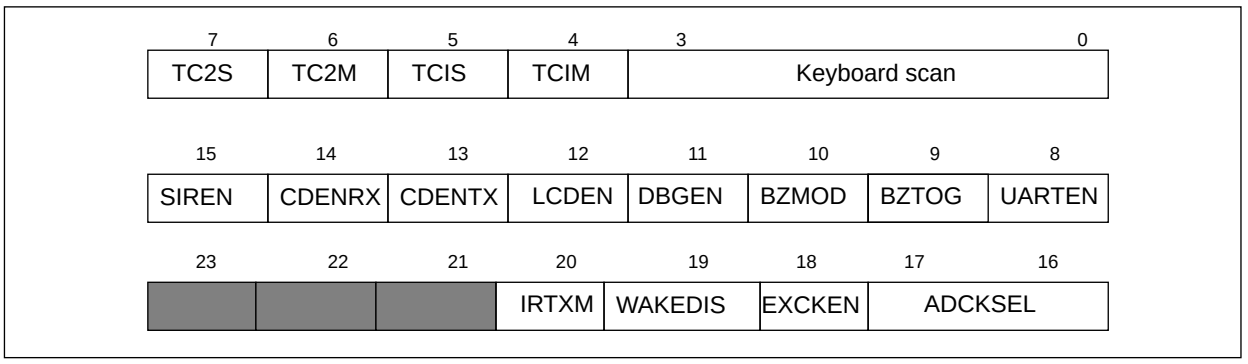


Figure 8-1: The system control register

Keyboard scan — This 4-bit field defines the state of the keyboard column drives. *Table 8-2: Keyboard scan field* gives definitions of these states.

Keyboard Scan	Column
0	All driven HIGH
1	All driven LOW
2 - 7	All Tristate
8	Column 0 only driven HIGH
9	Column 1 only driven HIGH
10	Column 2 only driven HIGH
11	Column 3 only driven HIGH
12	Column 4 only driven HIGH
13	Column 5 only driven HIGH
14	Column 6 only driven HIGH
15	Column 7 only driven HIGH

Table 8-2: Keyboard scan field

TC1M — Timer counter 1 mode. Setting this bit sets TC1 to prescale mode. Clearing it sets free running mode.

TC1S — Timer counter 1 clock source. Setting this bit sets the TC1 clock source to 512 KHz. Clearing it sets the clock source to 2KHz.

TC2M — Timer counter 2 mode. Setting this bit sets TC2 to prescale mode. Clearing it sets free running mode.

TC2S — Timer counter 2 clock source. Setting this bit sets the TC2 clock source to 512 KHz. Clearing it sets the clock source to 2KHz.

UARTEN — Internal UART enable bit. Setting this bit enables the internal UART.

BZTOG — This bit is used to drive the **BUZ** output directly.

BZMOD — This bit sets the **BUZ** output mode:

0 **BUZ** is connected directly to the BZTOG bit

1 **BUZ** is connected to the TC1 under flow bit

ADCKSEL — Microwire / SPI peripheral clock speed select. This 2-bit field selects the frequency of the ADC sample clock. This is twice the frequency of the synchronous serial ADC interface clock. ➲*Table 8-3: ADCCLK frequencies* shows the available frequencies.

Preliminary

ADCKSEL	ADC Sample Frequency (kHz)	ADC Interface Frequency (kHz)
00	8	4
01	32	16
10	128	64
11	256	128

Table 8-3: ADCCLK frequencies

DBGEN — Setting this bit enables debug/broadcast mode. In this mode, all internal accesses are output as if they were reads or writes to expansion memory addressed by CS6. CS6 will still be active in its standard address range. In addition the internal interrupt request and the fast interrupt request signals to the processor are output on port E bits 1 and 2 ie. in debug mode:

CS6 = CS6 / internal strobe
PE1 = nIRQ
PE2 = nFIQ

LCDEN — Setting this bit enables the LCD controller.

CDENTX — CODEC interface enable Tx bit. Setting this bit enables the CODEC interface for data transmission to an external CODEC device.

CDENRX — CODEC interface enable Rx bit. Setting this bit enables the CODEC interface for data reception from an external CODEC device.

SIREN — SIR protocol encoding enable bit. This has no effect if the UART is not enabled.

EXCLKEN — External expansion clock enable. If this bit is set the **EXPCLK** is enabled continuously with the same speed and phase as the CPU clock and will free run all the time the main oscillator is running. This bit should not be left set all the time for power consumption reasons. If the system enters the standby state the **EXPCLK** will become undefined. If this bit is clear **EXPCLK** will be active during memory cycles to expansion slots that have external wait state generation enabled only.

WAKEDIS — Switch on via the wakeup input is disabled if this bit is set.

IRTXM — IrDA Tx mode bit. This bit controls the IrDA encoding strategy. Clearing it means each zero bit transmitted is represented as a pulse of width 3/16th of the bit rate period. Setting this bit means each zero bit is represented as a pulse of width 3/16th of the period of 115,000 bit rate clock ie. 1.6µSec regardless of the selected bit rate. Setting this bit will use less power but probably reduce transmission distances.

BITS 21-31 — Reserved. Write will have no effect, will always read zero.

8.3.12 The System Status Flags Register (SYSFLG)

The system status flags register (SYSFLG) is a 32-bit read only register which indicates various system information. The bits in SYSFLG are defined in ➲*Figure 8-2: The System Status Flag Register* and are described below.

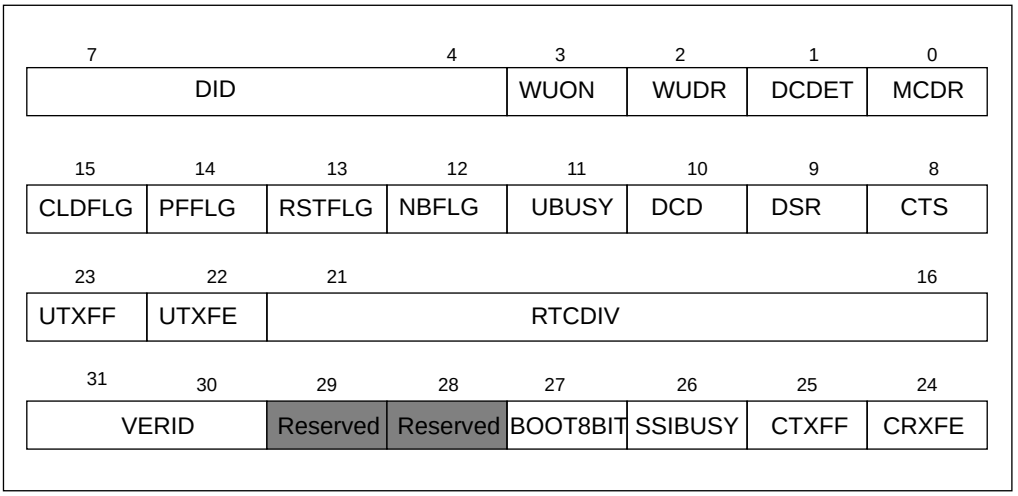


Figure 8-2: The System Status Flag Register

MCDR — This bit reflects the non-latched status of the media changed input.

DCDET — This bit reflects the inverted state of the **nEXTPWR** input pin.

WUDR — Wake up direct read. This bit reflects the non-latched state of the **WAKEUP** signal.

WUON — This bit is set if the system has been brought out of standby by a rising edge on the **WAKEUP** signal. It is only cleared by a system reset or by writing to the HALT or STDBY locations.

DID — Display ID nibble. This 4-bit nibble reflects the latched state of the 4 LCD data lines **DO[3:0]**. The state of the 4 LCD data lines is latched by the LCDEN bit and so always reflects the last state of these lines before the LCD controller was enabled. These bits identify the LCD display panel fitted.

CTS — This bit reflects the current status of the clear to send (CTS) modem control input to the built in UART.

DSR — This bit reflects the current status of the data set ready (DSR) modem control input to the built in UART.

DCD — This bit reflects the current status of the data carrier detect (DCD) modem control input to the built in UART.

UBUSY	UART transmitter busy. Set while the internal UART is busy transmitting data, it is guaranteed to remain set until the complete byte has been sent, including all stop bits.
NBFLG	The new battery flag bit is set if a LOW to HIGH transition has occurred on the **nBATCHG** input. It is cleared by writing to the STFCLR location.
RSTFLG	The reset flag is set if the **nURESET** input has been forced LOW. It is cleared by writing to the STFCLR location.
PFFLG	The Power Fail Flag is set if the system has been reset by the **PWRFL** input pin. It is cleared by writing to the STFCLR location.
CLDFLG	The cold start flag is set if ARM7100 has been reset with a power on reset. It is cleared by writing to the STFCLR location.
RTCDIV	This 6-bit field reflects the number of 64Hz ticks that have passed since the last increment of the RTC. It is the output of the divide by 64 chain that divides the 64Hz tick clock down to 1Hz for the RTC. The MSB is the 32Hz output, the LSB is the 1Hz output.
URXFE	UART receiver FIFO empty. The meaning of this bit depends on the state of the UFIFOEN bit in the UART bit rate and line control register. If the FIFO is disabled, this bit is set when the Rx holding register is empty. If the FIFO is enabled the URXFE bit will be set when the Rx FIFO is empty.
UTXFF	UART transmit FIFO full. The meaning of this bit depends on the state of the UFIFOEN bit in the UART bit rate and line control register. If the FIFO is disabled, this bit is set when the Tx holding register is full. If the FIFO is enabled the UTXFF bit will be set when the Tx FIFO is full.
CRXFE	The CODEC Rx FIFO empty bit is set if the 16 byte CODEC Rx FIFO is empty.
CTXFF	The CODEC Tx FIFO full bit is set if the 16 byte CODEC Tx FIFO is full.
SSIBUSY	The synchronous serial interface busy bit is set while data is being shifted in or out of the synchronous serial interface. When clear, data is valid to read.
Reserved	This will always read zero.
VERID	Version ID bits. These 2 bits determine the revision id for ARM7100. It will read 0 for the first revision.
BOOT8BIT	This bit indicates the default (power on reset) bus width of the ROM interface. If set, the initial bus width will be 8 bits. If clear, it will be 32 bits. See *8.3.13 Memory configuration register 1 (MEMCFG1)* on page 8-11 and *8.3.14 Memory configuration register 2 (MEMCFG2)* on page 8-11 for more

details on the ROM interface bus width. The state of this bit is determined by the state of Port E bit 0 during power on reset. LOW during power on reset will clear the BOOT8BIT bit and the system will boot from a 32-bit ROM. HIGH during power on reset will set the BOOT8BIT bit and the system will boot from an 8-bit ROM.

8.3.13 Memory configuration register 1 (MEMCFG1)

The memory configuration register 1 is a 32-bit read-write register which sets the configuration of the four expansion and ROM selects **nCS[0:3]**. Each select is configured with a one byte field, starting with expansion select 0.

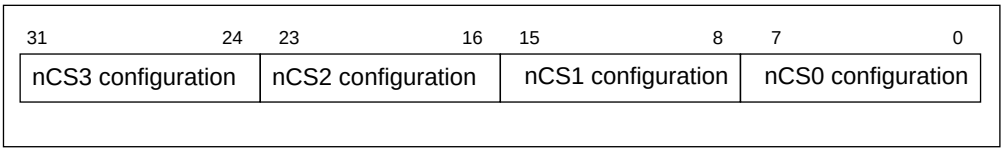


Figure 8-3: Memory configuration register 1

8.3.14 Memory configuration register 2 (MEMCFG2)

The memory configuration register 2 is a 32-bit read-write register which sets the configuration of the four expansion and ROM selects **CS[4:7**]. Each select is configured with a one byte field, starting with expansion select 4.

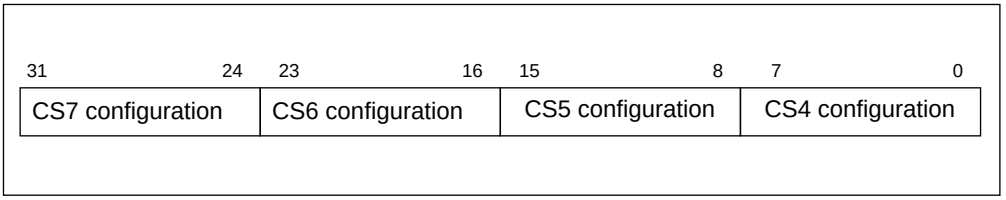


Figure 8-4: Memory configuration register 2

Each of the 8-bit fields in the memory configuration registers are identical and define the number of wait states, define the bus width, enable the **EXPCLK** output during accesses and enable sequential mode access. This is shown in *Figure 8-5: Byte fields in the memory configuration register* below.

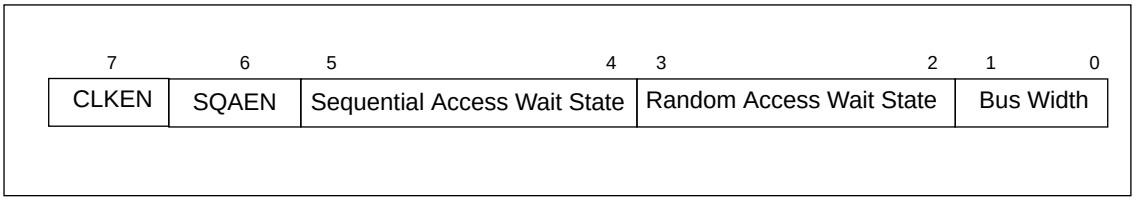


Figure 8-5: Byte fields in the memory configuration register

Table 8-4: Values of the bus width field defines the bus width field. The effect of this field is dependent on the BOOT8BIT bit which can be read in the SYSFLG register. All bits in the memory configuration register are cleared by a system reset and the state of the BOOT8BIT bit is determined by Port E bit 0 pin on ARM7100 during power on reset. In this way, pulling Port E bit 0 either LOW or HIGH during power on reset allows ARM7100 to boot from either 32-bit wide or 8-bit wide ROMs.

Bus Width Field	BOOT8BIT	Expansion Transfer Mode	Port E bit 0 during power on reset
00	0	32-bit wide bus access	LOW
01	0	16-bit wide bus access	LOW
10	0	8-bit wide bus access	LOW
11	0	PCMCIA mode	LOW
00	1	8-bit wide bus access	HIGH
01	1	PCMCIA mode	HIGH
10	1	32-bit wide bus access	HIGH
11	1	16-bit wide bus access	HIGH

Table 8-4: Values of the bus width field

When the bus width field is programmed to PCMCIA mode the bus width and bus conversion is defined by the state of **A[27**] and **A[26]**. *Table 8-5: PCMCIA mode bus widths* on page 8-12 defines the bus width and bus conversion for values of **A[27]** and **A[26]**. Word bus conversion converts an ARM 32-bit word access into a series of byte or 16-bit accesses. A special case is 16-bit I/O accesses (**A[26]** and **A[27]** HIGH). In this case, 32-bit ARM word accesses are not converted into two 16-bit accesses. This is to allow individual 16-bit register access. In this mode **D[16:31]** will be invalid and the output expansion address bit 1 is selected by the value of **A[25]**. ARM7100 will always output 0 on expansion address bit 25, ie. in 16-bit I/O mode processor address bit 25 becomes **PCMIA** address bit 1, and **PCMIA** address bit 25 is 0 limiting the 16-bit I/O address space to 32 Mb.

Note *16-bit I/O accesses are not converted to 32-bit ARM word accesses. This means that D[16:31] will be invalid during ARM word accesses to this memory area.*

A26	A27	Bus Width	Word Bus Conversion	PCMCIA Memory Area
0	0	8 Bits	Yes	8-bit attribute memory access
1	0	16 Bits	Yes	16-bit common memory access
0	1	8 Bits	Yes	8-bit I/O access
1	1	16 Bits	No	16-bit I/O access

Table 8-5: PCMCIA mode bus widths

Table 8-7: Values of the page mode access wait state field defines the values of the Random access wait state field.

Value	No. Wait States	Required Random Access Speed (nSEC)
00	4	250
01	3	200
10	2	150
11	1	100

Table 8-6: Values of the random access wait state field

Table 8-7: Values of the page mode access wait state field defines the values of the page mode access wait state field.

Value	No. Wait States	Required Random Access Speed (nSEC)
00	3	150
01	2	120
10	1	80
11	0	40

Table 8-7: Values of the page mode access wait state field

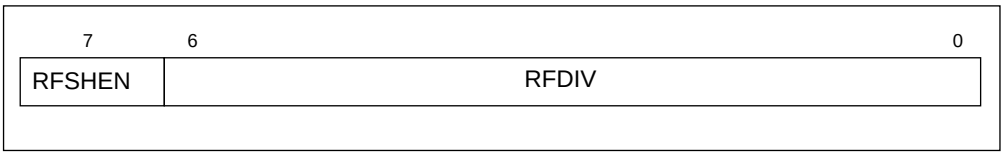
SQAEN — Sequential access enable. Setting this bit enables sequential accesses that are on a quad word boundary to take advantage of faster access times from devices that support page mode. The sequential access will be faulted after four words, (to allow video refresh cycles to occur) even if the access is part of a longer sequential access.

CLKEN — Expansion clock enable. Setting this bit enables the **EXPCLK** to be active during accesses to the selected expansion device. This provides a timing reference for devices that need to extend bus cycles using the **EXPRDY** input. Back to back (but not necessarily page mode) accesses result in a continuous clock.

For more details on bus timing, refer to *Chapter 20, DC and AC Parameters*.

8.3.15 DRAM refresh period register (DRFPR)

The DRAM refresh period register is an 8-bit read-write register which enables refresh and selects the refresh period used by the DRAM controller for its periodic CAS before RAS refresh. The value in the DRAM refresh period register is only cleared by a *power on reset*. Its state is maintained during a power fail or user reset.



RFSHEN — DRAM refresh enable. Setting this bit enables periodic refresh cycles to be generated by ARM7100 at a rate set by the RFDIV field. Setting this bit also enables self refresh mode when ARM7100 is in the standby state.

RFDIV — This 7-bit field sets the DRAM refresh rate. The refresh period is derived from a 128 KHz clock and is given by the following formula:

Frequency (KHz) = 128/(RFDIV + 1)

or

RFDIV = (128/Refresh frequency (KHz)) - 1

The maximum refresh frequency is 64 KHz, the minimum is 1KHz. The RFDIV field should not be programmed with zero as this results in no refresh cycles being initiated.

8.3.16 Interrupt status register (INTSR)

The interrupt status register is a 16-bit read only register. It reflects the current state of the 16 interrupt sources within ARM7100. Each bit is set if the appropriate interrupt is active. The interrupt assignment is given in ❍*Figure 8-6: Interrupt Assignment*.

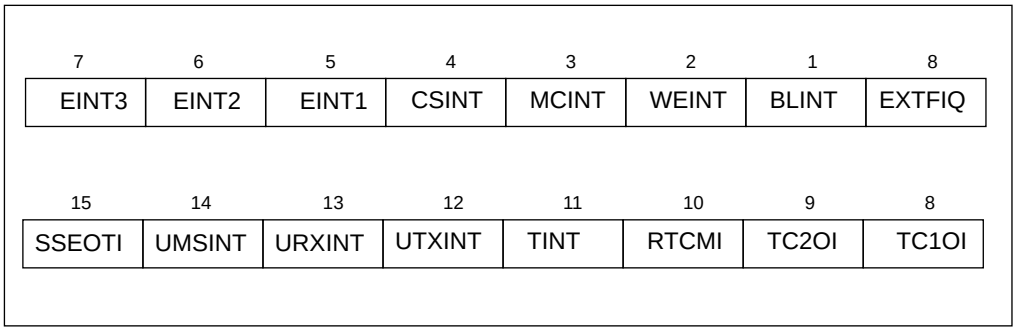


Figure 8-6: Interrupt Assignment

EXTFIQ — The external fast interrupt is active if the **nEXTFIQ** input pin is forced LOW and is mapped to the **nFIQ** input on the ARM7 processor.

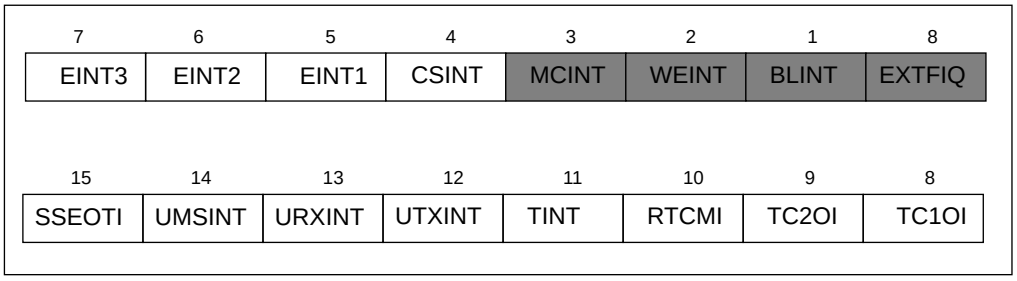
BLINT — Battery low interrupt is active if no external supply is present (**nEXTPWR** is HIGH) and the battery OK input pin **BATOK** is forced LOW. This interrupt is de-glitched with a 16 KHz clock so only generates an interrupt if it is active for longer than 62.5 mSec. It is mapped to the **nFIQ** input on the ARM7 processor and is cleared by writing to the BLEOI location.

WEINT	Watch dog expired interrupt becomes active on a rising edge of the periodic 64 Hz tick interrupt clock if the tick interrupt is still active, ie. if a tick interrupt has not been serviced for a complete tick period. It is cleared by writing to the TEOI location.
MCINT	This interrupt is active after a rising edge on the **MEDCHG** input pin has been detected, This input is de-glitched with a 16 KHz clock so only generates an interrupt if it is active for longer than 62.5 mSec. It is mapped to the **FIQ** input on the ARM7 processor and is cleared by writing to the MCEOI location.
CSINT	The CODEC sound interrupt is active if the CODEC interface is enabled and the CODEC data FIFO has reached half full or empty (depending on the interface direction). It is cleared by writing to the COEOI location.
EINT1	The external interrupt input 1 is active if the **nEINT1** input is active (LOW). It is cleared by returning **nEINT1** to the passive (HIGH) state.
EINT2	The external interrupt input 2 is active if the **NEINT2** input is active (LOW). It is cleared by returning **nEINT2** to the passive (HIGH) state.
EINT3	External interrupt input 3. This input will be active if the **EINT3** input is active (HIGH). It is cleared by returning **EINT3** to the passive (LOW) state.
TC1OI	The TC1 under flow interrupt becomes active on the next rising edge of the timer counter 1 clock after the timer counter has under flowed (reached zero). It is cleared by writing to the TC1EOI location.
TC2OI	The TC2 under flow interrupt becomes active on the next rising edge of the timer counter 2 clock after the timer counter has under flowed (reached zero). It is cleared by writing to the TC2EOI location.
RTCMI	The RTC compare match interrupt becomes active on the next rising edge of the 1Hz real time clock (one second later) after the 32-bit time written to the real time clock match register exactly matches the current time in the RTC. It is cleared by writing to the RTCEOI location.
TINT	64 Hz tick interrupt. This interrupt becomes active on every rising edge of the internal 64Hz clock signal. This 64 Hz clock is derived from the 15 stage ripple counter that divides the 32.768 KHz oscillator input down to 1Hz for the real time clock. This interrupt is cleared by writing to the TEOI location.

UTXINT	Internal UART transmit FIFO empty interrupt. The function of this interrupt source depends on whether the UART FIFO is enabled. If the FIFO is disabled (FIFOEN bit is clear in the UART bit rate and line control register), this interrupt is active when there is no data in the UART Tx data holding register. It is cleared by writing to the UART data register. If the FIFO is enabled, this interrupt is active when the UART Tx FIFO is half or more empty, and is cleared by filling the FIFO to at least half full.
URXINT	Internal UART receive FIFO full interrupt. The function of this interrupt source depends on whether the UART FIFO is enabled. If the FIFO is disabled this interrupt is active when there is valid Rx data in the UART Rx data holding register. It is cleared by reading this data. If the FIFO is enabled this interrupt is active when the UART Rx FIFO is half or more full or if the FIFO is non empty and no more characters have been received for a three character time out period. It is cleared by reading all the data from the Rx FIFO.
UMSINT	Internal UART modem status changed interrupt. This interrupt will be active if either of the two modem status lines (**CTS** or **DSR**) change state. It is cleared by writing to the UMSEOI location.
SSEOTI	Synchronous serial interface end of transfer interrupt. This interrupt is active after a complete data transfer to and from the external ADC has completed. It is cleared by reading the ADC data from the SYNCIO register.

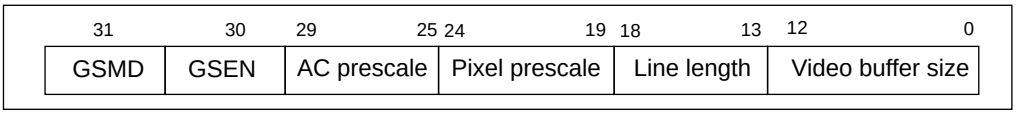
8.3.17 Interrupt mask register (INTMR)

The interrupt mask register is a 16-bit read-write register which is used to enable any of the 16 interrupt sources selectively within ARM7100. The four shaded interrupts all generate a fast interrupt request to the ARM7 processor. This causes a jump to processor virtual address 0000.0001C. All other interrupts generate a standard interrupt request causing a jump to processor virtual address 0000.00018. See *Table 9-1: Interrupt allocation* on page 9-3 for the interrupt allocation. Setting the appropriate bit in this register enables the corresponding interrupt. All bits are cleared by a *system reset.*



8.3.18 The LCD control register (LCDCON)

The LCD control register is a 32-bit read-write register which controls the size of the LCD screen and the mode in which the LCD controller operates. Refer to the system description of the LCD controller for more information on video buffer mapping.



Video buffer size — The video buffer size field is a 13-bit field that sets the total number of bytes (*128 quad words) in the video display buffer. This is calculated from the following formula:

```
Video buffer size = (Total bytes in video buffer/128) - 1
```

For example, for a 640 x 240 LCD and 4 bits per pixel the size of the video buffer = 640 x 240 x 4 = 614400 bits.
video buffer size field = (614400/128)-1
= 4799 or 0x12BF Hex

Line length — The line length field is a 6-bit field that sets the number of pixels in one complete line. This field is calculated from the formula:

```
Line length  = ( No. pixels in line/16 ) - 1
```

For example:
640 x 240 LCD line length = (640/16)-1
= 39 or 0x27 Hex

Pixel prescale — The pixel prescale field is a 6-bit number that sets the pixel rate prescale. The pixel rate is derived from a 36.864 MHz clock and is calculated from the following formula:

```
Pixel rate (MHz) = 36.864/ (Pixel prescale + 1)
```

The pixel rate should be chosen to give a complete screen refresh frequency of approximately 70 Hz to avoid flicker. Frequencies above 70 Hz should be avoided as this consumes additional power. The pixel prescale value can be expressed in terms of the LCD size by the following formula:

```
Pixel prescale = (526628/Total pixels in display)-1
```

The value should be rounded down to the nearest whole number, and zero is illegal and results in no pixel clock. For example:

640 x 240 LCD pixel prescale = 526628/(640x240)-1
= 2.428(2)

Actual pixel rate = 36.864E6/2+1
=12.288MHz

Actual refresh frequency = 12.288E6/(640x240)
= 80Hz

As the CL2 low pulse time is doubled after every CL1 high pulse, this refresh frequency is only an approximation, the accurate formula is:

12.288E6/((640x240)+120) =79.937Hz.

AC prescale — The AC prescale field is a 5-bit number that sets LCD AC bias frequency. This frequency is the required AC bias frequency for a given manufacturer's LCD plate. It is derived from the frequency of the line clock (CL1).

The **M** signal toggles after n + 1 counts of the line clock (CL1) where n is the number programmed into the AC prescale field. This number must be chosen to match the manufacturers recommendation. This is normally 13 but must not be exactly divisible by the number of lines in the display.

GSEN — Grey scale enable bit. Setting this bit enables grey scale output to the LCD. When it is cleared, each bit in the video map directly corresponds to a pixel in the display.

GSMD — Grey scale mode bit. Clearing this bit sets the controller to 2 bits per pixel (4 grey scales). Setting it sets it to 4 bits per pixel (15 grey scales).

8.3.19 Timer counter 1 data register (TC1D)

The timer counter 1 data register is a 16-bit read-write register which sets and reads data to TC1. Any value written will be decremented on the next rising edge of the clock.

8.3.20 Timer counter 2 data register (TC2D)

The timer counter 2 data register is a 16-bit read-write register which sets and reads data to TC2. Any value written will be decremented on the next rising edge of the clock.

8.3.21 Real time clock data register (RTCDR)

The real time clock data register is a 32-bit read-write register which sets and reads the binary time in the RTC. Any value written will be incremented on the next rising edge of the 1 Hz clock.

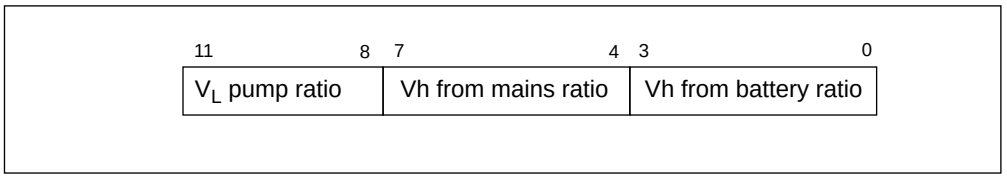
All bits in the real time clock data register are only cleared by an active **nPOR**.

8.3.22 Real time clock match register (RTCMR)

The real time clock match register is a 32-bit read-write register which sets and reads the binary match time to RTC. Any value written will be compared to the current binary time in the RTC, if they match it will assert the RTCMI interrupt source.

8.3.23 Pump control register (PMPCON)

The DC to DC converter pump control register is a 12-bit read-write only register which sets and controls the variable mark space ratio drives for two DC to DC converters. All bits in this register are cleared by a *system reset.*



Vh from battery — This 4-bit field controls the on time for the DC to DC pump for a Vh rail while the **nEXTPWR** input is HIGH. Setting these bits to 0 disables this pump. Setting them to 1 allows the pump to be driven in a 1:16 duty ratio, 2 in a 2:16 duty ratio etc. up to a 15:16 duty ratio. An 8:16 duty ratio results in a square wave of 96 KHz.

Vh from mains — This 4-bit field controls the on time for the DC to DC pump for a Vh rail while the **nEXTPWR** input is LOW. Setting these bits to 0 disables this pump. Setting them to 1 allows the pump to be driven in a 1:16 duty ratio, 2 in a 2:16 duty ratio etc. up to a 15:16 duty ratio. An 8:16 duty ratio results in a square wave of 96 KHz.

V_L pump ratio — This 4-bit field controls the on time for the DC to DC pump for the V_L voltage rail. Setting these bits to 0 disables this pump. Setting them to 1 allows the pump to be driven in a 1:16 duty ratio, 2 in a 2:16 duty ratio etc. up to a 15:16 duty ratio. An 8:16 duty ratio results in a square wave of 96 KHz. The state of the output drive pin (drive 1) is latched during power on reset, this latched value is used to determine the polarity of the bias voltage. The sense of the DC to DC converter control lines is summarised in *Table 8-8: Sense of DC to DC Converter Control Lines*.

Initial State of Drive n during POR	Sense of Drive n	Polarity of Bias Voltage
LOW	Active HIGH	+ve
HIGH	Active LOW	-ve

Table 8-8: Sense of DC to DC Converter Control Lines

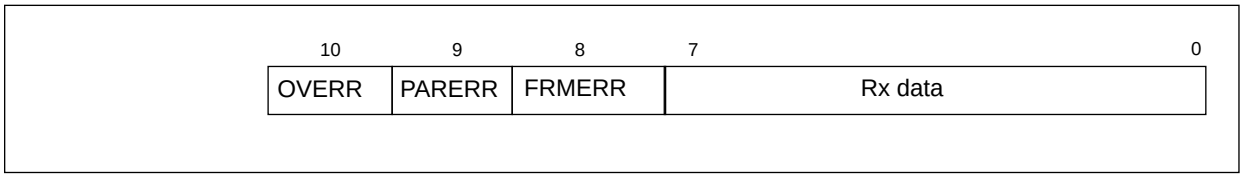
8.3.24 The CODEC interface data register (CODR)

The CODR register is an 8-bit read-write register. Data written to or read from this register is pushed or popped onto a 16 byte FIFO buffer. Data from this buffer is then serialised and sent to or received from the CODEC sound device. The CODEC interrupt CSINT is generated repetitively at 1/8th of the byte transfer rate and the state of the FIFOs can be read in the system flags register. The net data transfer rate to or from the CODEC device is 8 Kb per second giving an interrupt rate of 1 KHz.

8.3.25 UART data register (UARTDR)

The UARTDR register is an 11-bit read and an 8-bit write register for all data transfers to or from the internal UART. Data written to this register is pushed onto the 16-byte data Tx holding FIFO if the FIFO is enabled, or stored in a one byte holding register. This write initiates transmission from the UART. The UART data read register is made up of the 8-bit data byte received from the UART together with three bits of error status. Data read from this register is popped from the 16 byte data Rx FIFO if the FIFO is enabled, or read from a one byte buffer register containing the last byte received and error status if not enabled. Data received by the UART is automatically pushed onto the Rx FIFO if it is enabled.

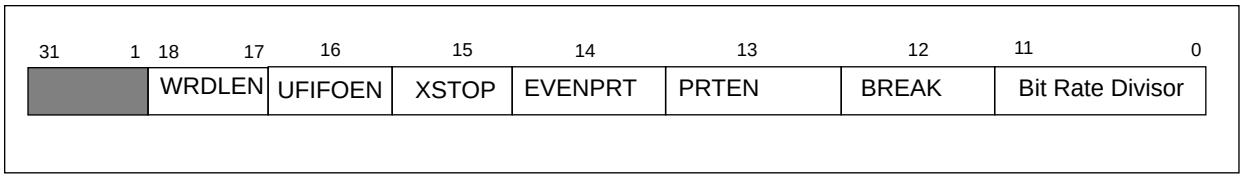
The Rx FIFO is 10 bits wide by 16 deep.



FRMERR	UART framing error. This bit is set if the UART detected an overrun or framing error while receiving the Rx data byte.
PARERR	UART parity error. This bit is set if the UART detected a parity error while receiving the Rx data byte.
OVERR	UART overrun error. This bit is set if more data is received by the UART and the FIFO is full. The overrun error bit is not associated with any single character and so is not stored in the FIFO. If this bit is set, the entire contents of the FIFO is invalid and should be cleared. This error bit is cleared by reading the UARTDR register.

8.3.26 UART bit rate and line control register (UBRLCR)

The bit rate divisor and line control register is a 19-bit read-write register. Writing to this register sets the bit rate and mode of operation for the internal UART.



Bit rate divisor	This 12-bit field sets the bit rate. The bit rate divider is fed by a clock frequency of 3.6864 MHz. It is then further divided internally by 16 to give the bit rate. The following formula gives the divisor value for any bit rate: `Divisor = (230400/bit rate)-1` A value of zero in this field is illegal.

Table 8-9: Internal UART Bit Rates shows some example bit rates with the corresponding divisor value.

Divisor Value	Bit Rate
1	115200
2	76800
3	57600
5	38400
11	19200
15	14400
23	9600
95	2400
191	1200
2094	110

Table 8-9: Internal UART Bit Rates

BREAK — Setting this bit drives the Tx output active (HIGH) to generate a break.

PRTEN — Parity enable bit. Setting this bit enables parity detection and generation.

EVENPRT — Even parity bit. Setting this bit sets parity generation and checking to even parity, clearing it sets odd parity. This bit has no effect if the PRTEN bit is clear.

XSTOP — Extra stop bit. Setting this bit will cause the UART to transmit two stop bits. Clearing it sets one stop bit after each data byte.

FIFOEN — Set to enable FIFO buffering of Rx and Tx data. Clear to disable the FIFO, ie. set its depth to one byte.

WRDLEN — This two bit field selects the word length according to *Table 8-10: UART word length* on page 8-21.

WRDLEN	Word length
00	5 bits
01	6 bits
10	7 bits
11	8 bits

Table 8-10: UART word length

8.3.27 Least significant word - LCD palette register (PALLSW)

The least and most significant word LCD palette registers make up a 64-bit read-write register which maps the logical pixel value to a physical grey scale level. The 64-bit register is made up of 16 4-bit nibbles, each nibble defining the grey scale level associated with the appropriate pixel value. If the LCD controller is operating in two bits per pixel, only the lower 4 nibbles are valid (D[15:0] in the least significant word). Similarly one bit per pixel means only the lower 2 nibbles are valid (D[7:0] in the least significant word). The pixel to grey scale level assignments are shown in ◗*Figure 8-7: Least significant word palette assignments* and ◗*Figure 8-8: Most significant word palette assignments*.

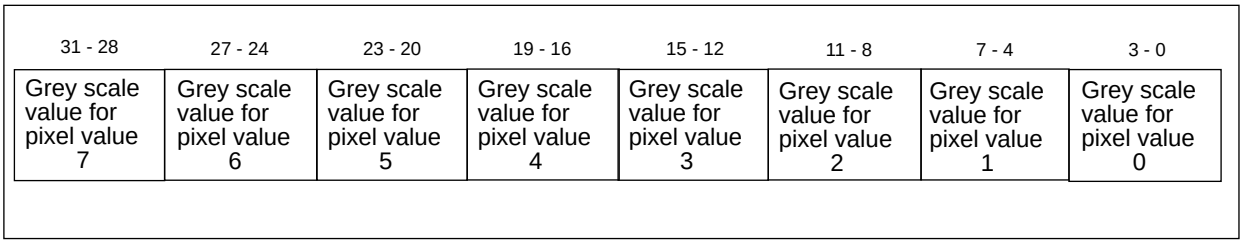


Figure 8-7: Least significant word palette assignments

8.3.28 Most significant word - LCD palette register (PALMSW)

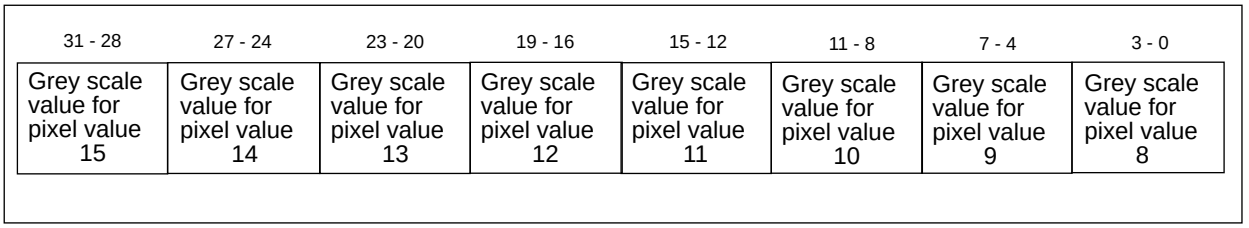


Figure 8-8: Most significant word palette assignments

The actual physical colour and pixel duty ratio for the grey scale values is shown in ◗*Table 8-11: Grey scale value to colour mapping*. Note that colours 8-15 are the inverse of colours 7-0 respectively. This means that colours 7 and 8 are identical. The steps in the grey scale are non linear but have been chosen to give a close approximation to perceived linear grey scales. This is due to the eye being more sensitive to changes in grey level close to 50% grey.

Grey scale value	Duty cycle	% pixels lit
0	0	0
1	1/9	11.1
2	1/5	20
3	4/15	26.7
4	3/9	33.3
5	2/5	40
6	4/9	44.4
7	1/2	50
8	1/2	50
9	5/9	55.6
10	3/5	60
11	6/9	66.7
12	11/15	73.3
13	4/5	80
14	8/9	88.9
15	1	100

Table 8-11: Grey scale value to colour mapping

8.3.29 Synchronous serial interface data register (SYNCIO)

SYNCIO is a 16-bit read-write register. The byte written to the SYNCIO register will be serialised and transmitted out of the synchronous serial interface. The clock will automatically be started at the programmed frequency and a synchronisation pulse will be issued. The **ADCIN** pin is sampled on every clock edge and the result is shifted in the SYNCIO read register.

During data transfer the SSIBUSY bit is set HIGH, at the end of a transfer the SSEOTI interrupt will be asserted. This interrupt is cleared by reading the SYNCIO register.

The data read from the SYNCIO register will be the last 16 bits shifted out of the ADC. The length of the data frame can be programmed by writing to the SYNCIO register allowing many different ADCs to be accommodated. ◔*Figure 8-9: Bits in SYNCIO write register* defines the bits in the SYNCIO register.

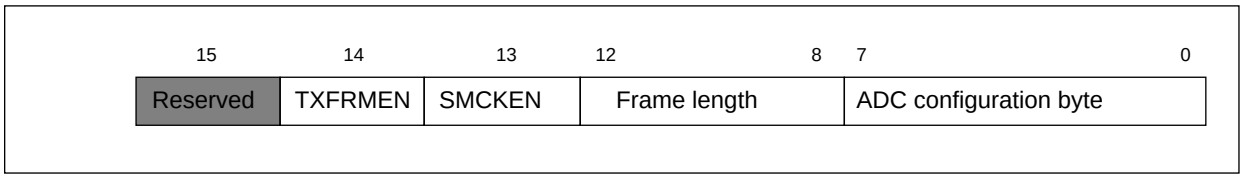


Figure 8-9: Bits in SYNCIO write register

ADC configuration	8-bit configuration data to be sent to the ADC.
Frame length	The 5-bit frame length field is the total number of shift clocks required to complete a data transfer. For many ADCs this is 25, 8 for configuration byte + 1 null bit + 16-bit result.
SMCKEN	Setting this bit will enable a free running sample clock at the programmed ADC clock frequency to be output on the **SMPLCK** pin.
TXFRMEN	Setting this bit causes an ADC data transfer to be initiated. The value in the ADC field will be shifted out to the ADC and depending on the frame length programmed, a number of bits will be captured from the ADC. If the SYNCIO register is written to with the TXFRMEN bit LOW, no ADC transfer will take place but the frame length and SMCKEN bits will be affected.

8.3.30 Clear all start up reason flags location (STFCLR)

A write to this location clears all the start up reason flags in the system flags status register SYSFLG.

8.3.31 Battery low end of interrupt (BLEOI)

A write to this location clears the interrupt generated by a low battery (falling **BATOK** with **nEXTPWR** HIGH).

8.3.32 Media changed end of interrupt (MCEOI)

A write to this location clears the interrupt generated by a rising edge of the **MEDCHG** input pin.

8.3.33 Tick end of interrupt location (TEOI)

A write to this location clears the current pending tick interrupt and watchdog interrupt.

8.3.34 TC1 end of interrupt location (TC1EOI)

A write to this location clears the under flow interrupt generated by TC1.

8.3.35 TC2 end of interrupt location (TC2EOI)

A write to this location clears the under flow interrupt generated by TC2.

8.3.36 RTC match end of interrupt (RTCEOI)

A write to this location clears the RTC match interrupt.

8.3.37 UART modem status changed end of interrupt (UMSEOI)

A write to this location clears the modem status changed interrupt.

8.3.38 CODEC end of interrupt location (COEOI)

A write to this location clears the sound interrupt (**CSINT**).

8.3.39 Enter idle state location (HALT)

A write to this location puts the system into the *idle* state by halting the clock to the processor until an interrupt is generated.

Note If the idle state is entered with no interrupts enabled, there is no mechanism for exiting the idle state except for a system reset.

8.3.40 Enter standby state location (STDBY)

A write to this location puts the system into the *standby* state by halting the main oscillator. It will automatically switch the DRAM's to self refresh if the RFSHEN bit is set in the DRAM refresh period register. All transitions to the *standby* state are synchronised with the DRAM cycles.

9 Interrupt Controller

This chapter describes the interrupt controller.

9.1 Interrupt Controller 9-2

Preliminary

9.1 Interrupt Controller

The ARM 710a has two interrupt types:

- interrupt request (IRQ)
- fast interrupt request (FIQ)

The interrupt controller in ARM7100 controls interrupts from 16 different sources. Twelve interrupt sources are mapped to the **IRQ** input and four sources to the **FIQ** input. FIQs have a higher priority than IRQs and if two interrupts at the same priority are active, the priority they are serviced in must be resolved in software.

All interrupts are *level sensitive*, ie. they must conform to the following sequence:

1. The device asserts the appropriate interrupt request line.
2. If the appropriate bit is set in the interrupt mask register, either FIQ or IRQ is asserted by the interrupt controller.
3. If interrupts are enabled, the processor jumps to the appropriate vector.
4. Interrupt despatch software reads the interrupt control and status register to establish the source(s) of the interrupt and calls the appropriate interrupt service routine(s).
5. Software in the interrupt service routine clears the interrupt source by an action specific to the device requesting the interrupt, eg. reading the UART Rx register.
6. The interrupt service routine may then re-enable interrupts and any other pending interrupts will be serviced in a similar way, or return to the interrupt dispatch code which can check for any more pending interrupts and dispatch them accordingly.

See ➲*Chapter 8, ARM7100 Programmer's Model* for details of interrupt registers.

Table 9-1: Interrupt allocation shows the names and allocation of interrupts in ARM7100.

Interrupt	Bit in Mask and ISR	Name	Comment
FIQ	0	EXTFIQ	External fast interrupt input
FIQ	1	BLINT	Battery low interrupt
FIQ	2	WEINT	Watch dog expired interrupt
FIQ	3	MCINT	MEDCHG interrupt
IRQ	4	CSINT	CODEC sound interrupt
IRQ	5	EINT1	External interrupt input 1
IRQ	6	EINT2	External interrupt input 2
IRQ	7	EINT3	External interrupt input 3
IRQ	8	TC1OI	TC1 under flow interrupt
IRQ	9	TC2OI	TC2 under flow interrupt
IRQ	10	RTCMI	RTC compare match interrupt
IRQ	11	TINT	64 Hz tick interrupt
IRQ	12	UTXINT	Internal UART transmit FIFO empty interrupt
IRQ	13	URXINT	Internal UART receive FIFO full interrupt
IRQ	14	UMSINT	Internal UART modem status changed interrupt
IRQ	15	SSEOTI	Synchronous serial interface end of transfer interrupt

Table 9-1: Interrupt allocation

10 The Expansion and ROM Interface

This chapter describes the ROM Interface.

10.1 The Expansion and ROM Interface 10-2

Preliminary

10.1 The Expansion and ROM Interface

Eight separate linear memory or expansion segments are decoded by the ARM7100. Each segment is 256 Mb and can be interfaced to a conventional SRAM-like interface.

Each segment can be programmed individually to:

- be 8, 16 or 32 bits wide
- support page mode access
- execute from 0 to 4 wait states.

In addition, bus cycles can be extended using the **EXPRDY** input signal.

Page mode access is accomplished by running up to four accesses together. This can significantly improve bus bandwidth to devices such as ROMs. Sequential burst mode access is always faulted (the bus returned to idle) after four *accesses* regardless of bus width to allow DMA and refresh cycles.

See ➲*Chapter 8, ARM7100 Programmer's Model* for details of the expansion and ROM interface registers.

11 DRAM controller

This chapter describes the DRAM controller.

11.1 DRAM Controller 11-2

Preliminary

11.1 DRAM Controller

The DRAM controller in ARM7100 provides connections allowing a direct interface to up to four banks of DRAM. Each bank is 32 bits wide and up to 256 Mb in size. Four RAS lines are provided (one per bank) and four CAS lines (one per byte line). The DRAM device size is not programmable if devices smaller than the largest size supported (1 Gbit) are used. This leads to a segmented memory map with each bank separated by 256 MBytes. Segments that are smaller than the bank size will repeat within the bank.

Table 11-1: Physical to DRAM address mapping shows the mapping of physical address to DRAM row and column address. This mapping has been organised to support any DRAM device size from 4 Mbit to 1 Gbit with a square row and column configuration, ie. the number of column addresses is equal to the number of row addresses. If a non-square DRAM is used, further fragmentation of the memory map will occur. However the smallest contiguous segment will always be 1 Mb.

Memory Address	DRAM Column	DRAM Row	Pin Name
0	A2	A10	A[27]/DRA[0]
1	A3	A11	A[26]/DRA[1]
2	A4	A12	A[25]/DRA[2]
3	A5	A13	A[24]/DRA[3]
4	A6	A14	A[23]/DRA[4]
5	A7	A15	A[22]/DRA[5]
6	A8	A16	A[21]/DRA[6]
7	A9	A17	A[20]/DRA[7]
8	A19	A18	A[19]/DRA[8]
9	A21	A20	A[18]/DRA[9]
10	A23	A22	A[17]/DRA[10]
11	A25	A24	A[16]/DRA[11]
12	A27	A26	A[15]/DRA[12]

Table 11-1: Physical to DRAM address mapping

Table 11-2: DRAM address mapping shows the address mapping for various DRAMs with square and non-square row and address inputs, assuming two x16 devices are connected to each RAS line. This mapping is repeated every 256 Mb for each DRAM bank. n is given by n = 0xC + bank number, eg. 0 for bank 0.

Device Size	Address Configuration	Total Size of Bank	Address Range of Segment(s)	Size of Segment(s)
4 Mbit	9 Row x 9 Column	1 Mbyte	n000.0000 - n00F.FFFF	1 Mbyte
16 Mbit	10 Row x 10 Column	4 MBytes	n000.0000 - n03F.FFFF	4 MBytes
16 Mbit	12 Row x 8 Column	4 MBytes	n000.0000 - n003.FFFF n010.0000 - n013.FFFF n040.0000 - n043.FFFF n050.0000 - n053.FFFF n100.0000 - n103.FFFF n110.0000 - n113.FFFF n140.0000 - n143.FFFF n150.0000 - n153.FFFF n400.0000 - n403.FFFF n410.0000 - n413.FFFF n440.0000 - n443.FFFF n450.0000 - n453.FFFF n500.0000 - n503.FFFF n510.0000 - n513.FFFF n540.0000 - n543.FFFF n550.0000 - n553.FFFF	256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes 256 KBytes
64 Mbit	11 Row x 11 Column	16 MBytes	n000.0000 - n0FF.FFFF	16 MBytes
64 Mbit	13 Row x 9 Column	16 MBytes	n000.0000 - n00F.FFFF n020.0000 - n02F.FFFF n080.0000 - n08F.FFFF n0A0.0000 - n0AF.FFFF n200.0000 - n20F.FFFF n220.0000 - n22F.FFFF n280.0000 - n28F.FFFF n2A0.0000 - n2AF.FFFF n800.0000 - n80F.FFFF n820.0000 - n82F.FFFF n880.0000 - n88F.FFFF n8A0.0000 - n8AF.FFFF nA00.0000 - nA0F.FFFF nA20.0000 - nA2F.FFFF nA80.0000 - nA8F.FFFF nAA0.0000 - nAAF.FFFF	1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte 1 MByte
256 Mbit	12 Row x 12 Column	64 MBytes	n000.0000 - n3FF.FFFF	64 MBytes
1 Gbit	13 Row x 13 Column	256 MBytes	n000.0000 - nFFF.FFFF	256 MBytes

Table 11-2: DRAM address mapping

The DRAM controller contains a programmable refresh counter. The refresh rate is controlled using the DRAM refresh period register (DRFPR).