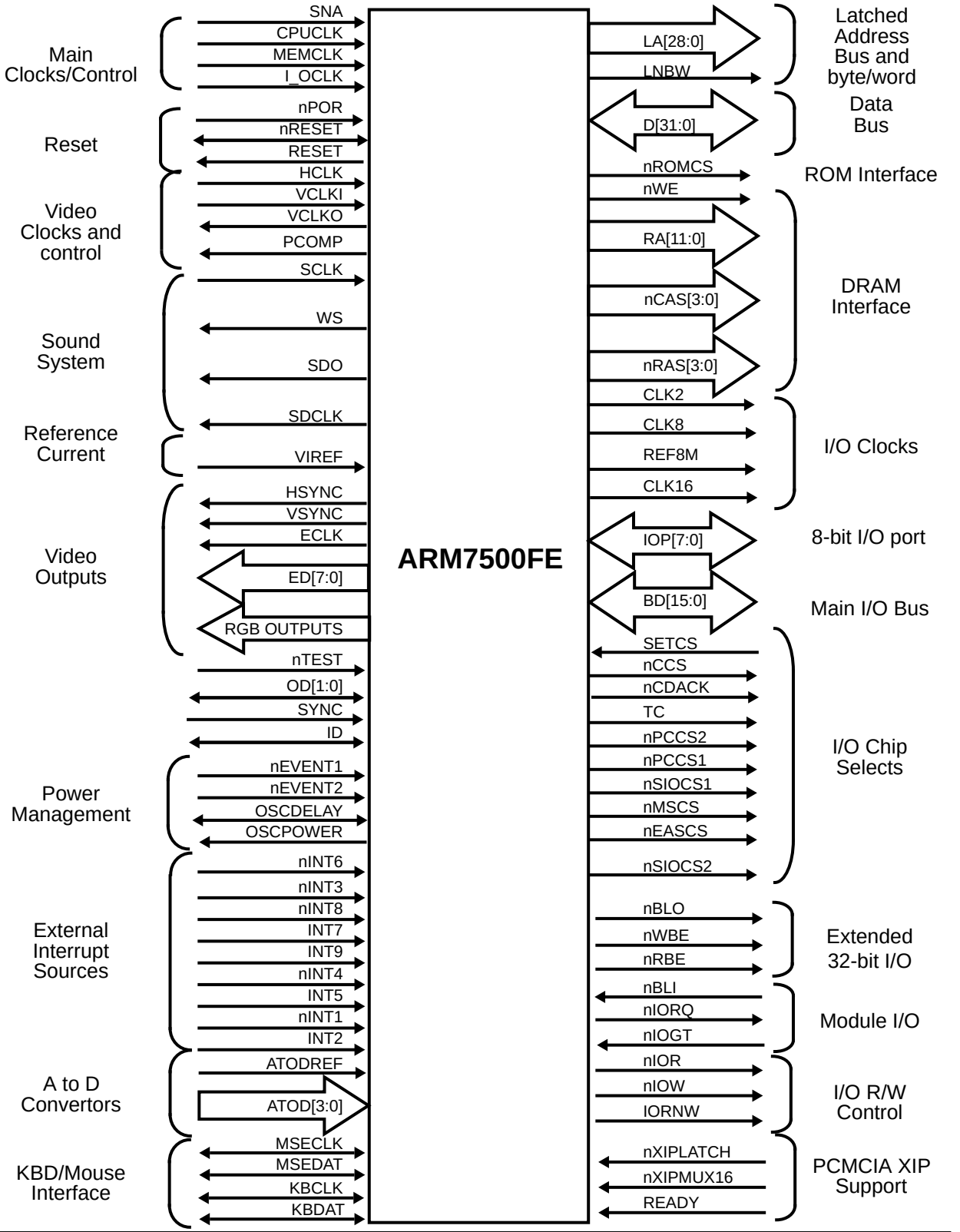
2 Signal Description

This chapter gives the name, type, and relevant details of each of the ARM7500FE signals.

Signal Description



2.1 Signal Description for ARM7500FE

Note: *When output signals are placed in the high impedance state for long periods, care must be taken to ensure that they do not float to an undefined logic level.*

Key to signal types:

IC	Input, CMOS threshold
OCZ	Output, CMOS levels, tri-stateable
IT	Input, TTL threshold
ICS	Input, CMOS Schmitt
IA	Input, analog
OA	Output, analog
BTZ	Bidirectional, CMOS output, TTL threshold input level
TOD	Open drain, TTL input
CSOD	Open drain, CMOS schmitt input
IAOD	Input, analog with programmable internal pull-down transistor

For outputs and bidirectionals, drive strength is classified 1,2 or 3. See *Chapter 22: DC and AC Parameters* for DC and AC characteristics.

Pin allocation is described in *Chapter 24: Pinout* .

Name	Type	Description
LA[28:0]	OCZ2	Latched address bus. This bus is the latched version of the ARM address for memory accesses, changing on the falling edge of the internal MCLK signal.
LNBW	OCZ2	Latched Not Byte word signal. This is a latched version of the internal NBW signal from the ARM processor, changing on the falling edge of the internal MCLK signal.
D[31:0]	BTZ2	The main data bus for the ARM7500FE. All external data transfers happen via this bus. When the ARM7500FE is configured for operation in 16-bit mode, only the lower 16 bits are used.
SnA	IC	Synchronous/not Asynchronous. This pin is set according to the relationship required between the internal clock signals MCLK and FCLK for the ARM. If this pin is set HIGH, both the memory system and the CPU are driven from the **MEMCLK** pin, and the required synchronous timing relationship between the ARM processor clocks is generated automatically on-chip. If different clocks are to be used, for the **MEMCLK** and **CPUCLK** inputs, the **SnA** pin must be set LOW.
BOUT	AO	Blue Analog Output. The video signal analog outputs are designed to drive doubly-terminated 75 lines.
ECLK	OCZ3	External Clock. When enabled, this clock validates the data on ED[7:0]. In normal video mode, it runs at the pixel rate, but when LCD data is being produced, it runs at a quarter of the pixel rate.

Table 2-1: ARM7500FE signal description

Signal Description

Name	Type	Description
ED[7:0]	OCZ2	External Data. This is the digital video output port of the ARM7500FE. From this, the digital equivalent of the analog output may be produced in any color, or data from the external palette may be produced. This may be used for a variety of purposes such as fading or supremacy. Also, data for driving LCD panels is output from this port. Data produced is validated by **ECLK**.
GOUT	AO	Green Analog Output. The video signal analog outputs are designed to drive doubly-terminated 75Ω lines.
HCLK	IT	High speed Clock for use with video subsystem.
HSYNC	OCZ3	Horizontal Synchronization. There are two synchronization outputs on ARM7500FE, **HSYNC** and **VSYNC**. Dependent on the state of bits 17 and 16 in the video External register, either a horizontal or a composite (NOR) sync may be output on this pin, in either polarity. The width of the **HSYNC** pulse is definable in units of 2 pixels.
PCOMP	OCZ1	Phase Comparator Output for use with VCLK pins.
ROUT	AO	Red Analog Output. The video signal analog outputs are designed to drive doubly-terminated 75Ω lines.
SCLK	IT	Sound Clock. This signal can be used to clock the sound system, when a clock asynchronous to the internal video reference clock is required.
SDCLK	OCZ2	Serial Data Clock. This clock validates serial sound data on its rising edge.
SDO	OCZ2	Serial Data Out. Serial sound data is output from this pin.
SYNC	IT	External SYNC. This signal is used to synchronize ARM7500FE with another video system.
VCLKI	IC	Phase Comparator Clock In (for video subsystem).
VCLKO	OCZ2	Phase Comparator Clock Out (for video subsystem).
VDD_Analog		Positive (+5V) supply for analog video system.
VIREF	IA	Video Reference Current. The video DACs need a reference current in order to calibrate them. A constant current source is recommended, although a resistor up to **VDD** is sufficient for many applications. This current also generates the constant source for the A to D comparators.
VSS_Analog		Supply ground for analog video system.
VSYNC	OCZ3	Vertical Synchronization. Dependent on the state of bits 19 and 18 in the external register, either a vertical or a composite (XNOR) sync may be output on this pin, in either polarity. The width of the **VSYNC** pulse may be defined in units of a raster.
WS	OCZ2	Word Select. This signal denotes whether the output serial data is for the left hand stereo channel or the right hand channel.

Table 2-1: ARM7500FE signal description (Continued)

Name	Type	Description
nTEST	IT	Test mode input. This pin should be held permanently HIGH. It is only intended to be used during production test of the ARM7500FE. An on-chip pull-up is included, but it is advisable to fit an external pull-up resistor to this pin.
nWE	OCZ3	Write enable. Active low.
RA[11:0]	OCZ2	DRAM row/column multiplexed address bus. Addresses for this bus are decoded from the ARM processor address for normal memory accesses, and are generated by the DMA controller for DMA.
nRAS[3:0]	OCZ3	DRAM row address strobes. Each of these selects one of the four banks of DRAM available.
nCAS[3:0]	OCZ2	DRAM column address strobes. These select the byte within the word for DRAM accesses.
VDD_ATOD	power	Positive 5V supply for the A to D converter comparators
VSS_ATOD	power	Analog ground for the A to D converter comparators
ATOD[3:0]	IAOD	Four A to D channel input voltages.
ATODREF	IA	Reference voltage for the A to D converter comparators.
OSCPOWER	OCZ1	Enable signal for the system oscillator(s). When LOW, this signal can be used to disable the external oscillator(s).
OSCDELAY	CSOD1	Requires an RC network to generate a fixed delay when restarting the system oscillator(s) on exit from STOP mode.
RESET	OCZ1	Reset output, synchronized version of internal system reset signal.
nRESET	CSOD2	Open drain output and ‘soft’ reset input. This pin is sampled every 1µs for reset events, so to guarantee a successful reset, a reset pulse applied to this pin must be longer than 1µs. (Note-1µs, assuming the internal I/O clock is 32MHz)
nROMCS	OCZ1	ROM Chip select. Goes LOW to indicate a ROM access.
I_OCLK	IC	I/O system clock. This clock input should always be 32MHz when in divide by 1 mode, and 64MHz in divide by 2 mode.
MEMCLK	IC	Memory system clock. In synchronous mode, ARM processor FCLK is also driven from this clock.
CPUCLK	IC	Clock used to create FCLK for the ARM CPU in asynchronous mode. When SnA is HIGH this should be tied HIGH or LOW permanently.
BD[15:0]	BTZ2	The main external 16-bit I/O bus.
MSCLK	TOD2	Mouse clock. An open drain pin for the mouse PS/2 interface.
MSDATA	TOD2	Mouse data. An open drain pin for the mouse PS/2 interface.
KBCLK	TOD2	Keyboard clock. An open drain pin for the keyboard PS/2 interface.

Table 2-1: ARM7500FE signal description (Continued)

Name	Type	Description
KBDATA	TOD2	Keyboard data. An open drain pin for the keyboard PS/2 interface.
nPOR	ICS	Power on reset. Any LOW transitions on this pin are detected and stretched to ensure full reset.
IOP[7:0]	TOD1	8 bit wide I/O port. Each bit is directly controllable via an ARM7500FE register, and can be used as an interrupt source if required.
ID	TOD1	The ID pin can be used to activate a system ID chip. It is forced LOW during the power on reset sequence.
OD[1:0]	TOD1	Two open drain pins which (unlike the **IOP[7:0]** bus) cannot be used to generate interrupts, but can be used as general purpose I/O pins, for example to communicate with a real time clock chip.
SETCS	IC	**SETCS** selects between two address decoding options for the three main I/O chip selects. It affects the outputs **nEASCS**, **nMSCS** and **nSIOCS2**.
nINT1	IT	Falling edge triggered interrupt pin. This pin also has the feature that its value can be read directly in the IOCR I/O control register.
INT2	IT	Rising edge triggered interrupt pin. Can generate an IRQ interrupt.
nINT3	IT	Active LOW interrupt pin. Can generate an IRQ interrupt.
nINT4	IT	Active LOW interrupt pin. Can generate an IRQ interrupt.
INT5	IT	Active HIGH interrupt pin. Can be used to generate either an IRQ or a FIQ interrupt, depending on the status of the relevant mask register bits.
nINT6	IT	Active LOW interrupt pin. Can generate either an IRQ or a FIQ depending on the programming of the mask registers.
INT7	IT	Active HIGH interrupt pin. Can generate an IRQ interrupt.
nINT8	IT	Active LOW interrupt pin. Can be used to generate either a FIQ or an IRQ interrupt.
INT9	IT	Active HIGH interrupt pin, which can only be used to generate a FIQ (highest priority) interrupt.
nEVENT1	IC	Active LOW asynchronous event pin 1. A falling edge is used to terminate STOP or SUSPEND power saving modes.
nEVENT2	IT	Active LOW asynchronous event pin 2. A falling edge is used to terminate STOP or SUSPEND power saving modes.
READY	IT	Can be used to stretch I/O accesses when set LOW during a 16MHz PC-style I/O cycle.
nIORQ	OCZ2	I/O request signal used for Module type I/O for handshaking, together with **nIOGT**.
nIOGT	IT	I/O grant signal used for Module type I/O for handshaking, together with **nIORQ**.
nBLI	IT	Input used during Module-style I/O reads to cause the latching of data from the BD port.

Table 2-1: ARM7500FE signal description (Continued)

Name	Type	Description
nBLO	OCZ1	Latching signal for use with external latches on the upper 16 bits of the external datapath to create a 32-bit wide I/O bus.
nRBE	OCZ1	Active LOW Read enable for an external transceiver attached to the upper 16 bits of the I/O bus, to create a 32-bit wide I/O bus.
nWBE	OCZ1	Active LOW Write enable for an external transceiver attached to the upper 16 bits of the I/O bus, to create a 32-bit wide I/O bus.
nXIPMUX16	IT	For Execute in place (XIP) support. This signal multiplexes 16 bits of data from the upper or lower halfword of the ARM7500FE internal data bus to the 16-bit I/O bus, depending on its state during writes.
nXIPLATCH	IC	For XIP support. Latches the upper 16 bits of data from the I/O bus while the lower 16 bits are being read. Used in conjunction with **nXIPMUX16** to enable XIP from, for example, a 16-bit PCMCIA card.
nSIOCS1	OCZ1	Active LOW chip select for simple I/O.
nSIOCS2	OCZ1	Active LOW chip select for simple I/O, with address decode modified according to the state of **SETCS**.
nMSCS	OCZ1	Active LOW chip select for module type I/O, with address decode modified according to the state of **SETCS**.
nEASCS	OCZ1	Active LOW chip select for extended 16Mhz PC-style I/O, with address decode modified according to the state of **SETCS**.
nCCS	OCZ1	Not Combo Chip Select. Chip select signal for a PC Combo chip.
nCDACK	OCZ1	Not Combo Dack. Chip select and Dack signal for PC Combo chip.
TC	OCZ1	Active HIGH terminal count. Used in conjunction with the **nCDACK** signal for pseudo DMA to a Combo chip.
nPCCS1	OCZ1	Active LOW chip select for an area of 16Mhz PC-style I/O space.
nPCCS2	OCZ1	Active LOW chip select for an area of 16Mhz PC-style I/O space.
IORNW	OCZ2	I/O read/not write, HIGH during an I/O read, and LOW during an I/O write.
nIOR	OCZ2	Not I/O read. This has two functions: • It is LOW during simple and PC-style I/O reads. Not used for Module type I/O. • It is also asserted LOW during ROM read cycles to act as an Output Enable.
nIOW	OCZ2	Not I/O write.This has two functions: • It is LOW during simple and PC-style I/O reads. Not used for Module type I/O. • It is also asserted LOW during writes to ROM space, to act as a Write Enable, if writes are enabled in the ROMCR register.
CLK2	OCZ2	2MHz I/O clock output.

Table 2-1: ARM7500FE signal description (Continued)

Name	Type	Description
CLK8	OCZ2	8MHz I/O clock output, the inverted version of REF8M.
REF8M	OCZ2	8MHz I/O clock output.
CLK16	OCZ2	16MHz I/O clock output, for PC-style I/O.

Table 2-1: ARM7500FE signal description (Continued)

3 The ARM Processor Macrocell

This chapter introduces the ARM processor 32-bit microprocessor macrocell.

The ARM Processor Macrocell

3.1 Introduction

The ARM7500FE contains a 32-bit RISC ARM processor, similar to the ARM710C macrocell. It has a 4Kbyte cache, write buffer, and a Memory Management Unit (MMU). The ARM processor macrocell offers high-level RISC performance, yet its fully static design ensures minimal power consumption. This makes it ideal for incorporation into the ARM7500FE. The ARM7500FE aims to make maximum use of the performance and flexibility offered by the ARM processor.

This part of the datasheet describes the features of the ARM processor macrocell which are available to the user in its embedded state within the ARM7500FE single-chip computer. It is not intended that this should be used as a stand-alone datasheet for a separate ARM processor macrocell.

3.1.1 Architecture

The ARM processor architecture is based on 'Reduced Instruction Set Computer' (RISC) principles, and the instruction set and related decode mechanism are greatly simplified compared with microprogrammed 'Complex Instruction Set Computers' (CISC).

The mixed data and instruction cache together with the write buffer substantially raise the average execution speed and reduce the average amount of memory bandwidth required by the processor. This allows the ARM7500FE bus structure to support Direct Memory Access (DMA) channels with minimal performance loss.

The MMU supports a conventional two-level page-table structure and a number of extensions which make it ideal for embedded control, UNIX and Object Oriented systems.

3.2 Instruction Set

The instruction set comprises ten basic instruction types:

- two of these make use of the on-chip arithmetic logic unit, barrel shifter and multiplier to perform high-speed operations on the data in a bank of 31 registers, each 32 bits wide
- three classes of instruction control data transfer between memory and the registers, one optimized for flexibility of addressing, another for rapid context switching and the third for swapping data
- two instructions control the flow and privilege level of execution
- three types are dedicated to the control of coprocessors which allow the functionality of the instruction set to be extended in an open and uniform way; the on-chip FPA is one such processor. However, as for the ARM710, the facility to add external coprocessors to the ARM7500FE is not available, and software emulation of coprocessor activity will be required if instructions other than those for the on-chip FPA or control coprocessor #15, are to perform a defined function.

The ARM instruction set is a good target for compilers of many different high-level languages. Where required for critical code segments, assembly code programming is also straightforward, unlike some RISC processors which depend on sophisticated compiler technology to manage complicated instruction interdependencies.

3.3 Memory Interface

The memory interface has been designed to allow the performance potential to be realized without incurring high costs in the memory system. Speed-critical control signals are pipelined to allow system control functions to be implemented in standard low-power logic, and these control signals permit the ARM7500FE to exploit the paged mode access offered by industry-standard DRAMs.

3.4 Clocks and Synchronous/Asynchronous Modes

The ARM processor uses two independent clock sources, MCLK and FCLK. Both are generated internally to ARM7500FE from **MEMCLK** and **CPUCLK**. The ARM7 core CPU switches between MCLK and FCLK according to the operation being carried out. For example, if the ARM7 core CPU is reading data from the cache it will be clocked by FCLK, whereas if the core CPU is reading data from uncached memory then it will be clocked by MCLK. The ARM processor's control logic ensures that the correct clock is used internally and switches between the two clocks automatically.

When SnA is tied high **MEMCLK** creates both FCLK and MCLK, with MCLK having half the frequency of FCLK. This synchronous mode ensures that there are no synchronization penalties whenever the ARM 7 core is switched between FCLK and MCLK.

When SnA is tied low, **MEMCLK** creates MCLK and **CPUCLK** must be driven to supply FCLK. **MEMCLK** and **CPUCLK** can be of unrelated frequency. There is a synchronization penalty whenever the ARM7 core clock switches between MCLK and FCLK. This penalty is symmetric, and varies between nothing and a whole period of the clock to which the core is resynchronizing. Thus when changing there is an average resynchronization penalty of half a clock period, MCLK or FCLK as appropriate.

3.5 ARM Processor Block Diagram

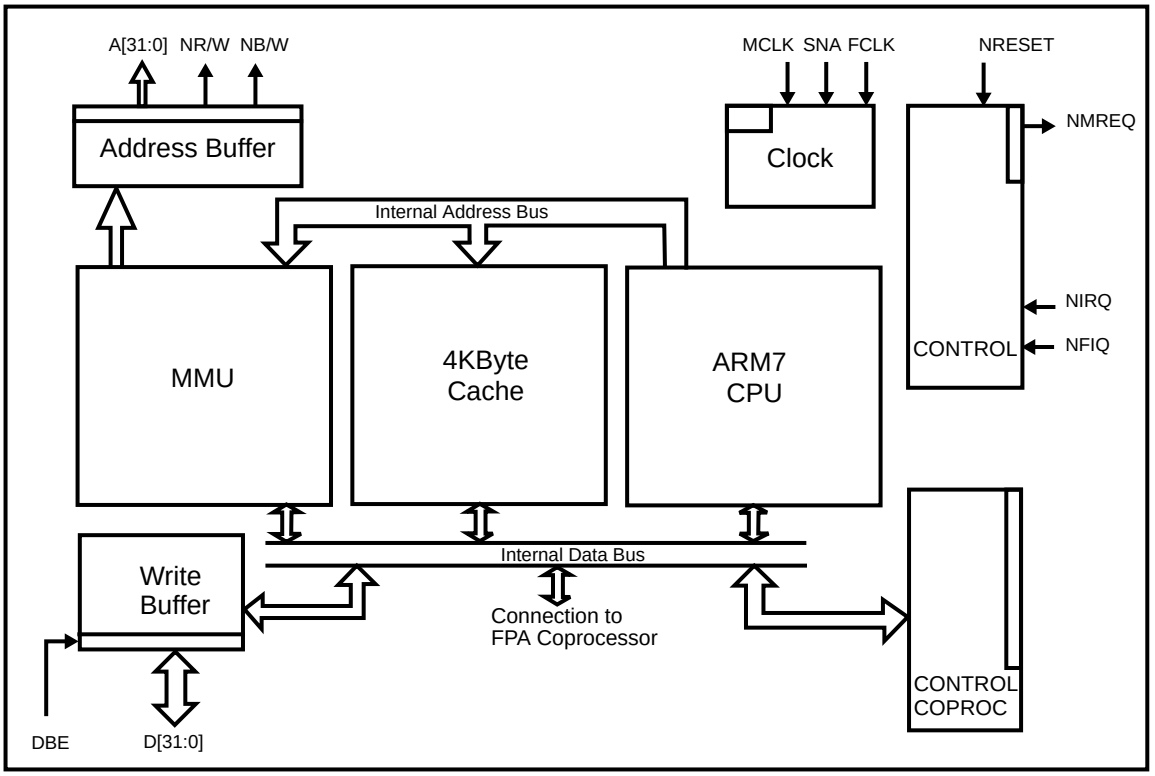


Figure 3-1: ARM processor block diagram

4 The ARM Processor Programmers' Model

This chapter details the ARM processor's programmable registers.

The ARM Processor Programmers' Model

4.1 Introduction

The ARM processor supports a variety of operating configurations.
Some are controlled by register bits and are known as the *configurations*.
Others may be controlled by software and are known as *operating modes*.

4.2 Register Configuration

The ARM processor provides 3 register configuration settings which may be changed while the processor is running. These are discussed below.

4.2.1 Big- and little-endian (the bigend bit)

The bigend bit, in the Control Register, sets whether the ARM7500FE treats words in memory as being stored in big-endian or little-endian format. Memory is viewed as a linear collection of bytes numbered upwards from zero. Bytes 0 to 3 hold the first stored word, bytes 4 to 7 the second, and so on.

Little-endian

In the little-endian scheme, the lowest-numbered byte in a word is considered to be the least-significant byte of the word, and the highest-numbered byte is the most-significant byte.

Byte 0 of the memory system should be connected to data lines 7 through 0 (**D[7:0]**) in this scheme.

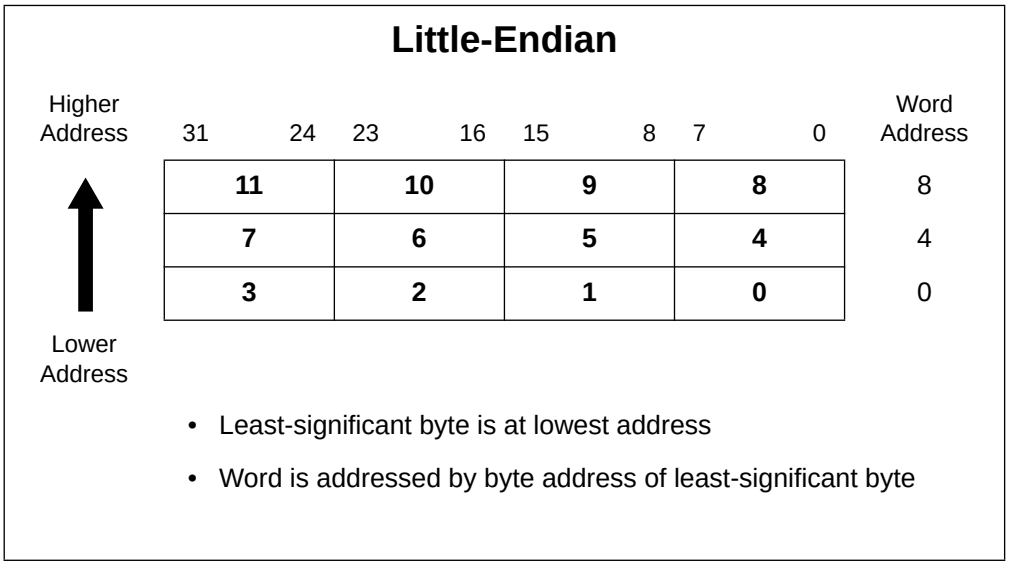


Figure 4-1: Little-endian addresses of bytes within words

Big-endian

In the big-endian scheme, the most-significant byte of a word is stored at the lowest-numbered byte, and the least-significant byte is stored at the highest-numbered byte.

Byte 0 of the memory system should therefore be connected to data lines 31 through 24 (**D[31:24]**).

Load and store are the only instructions affected by the endiannism.

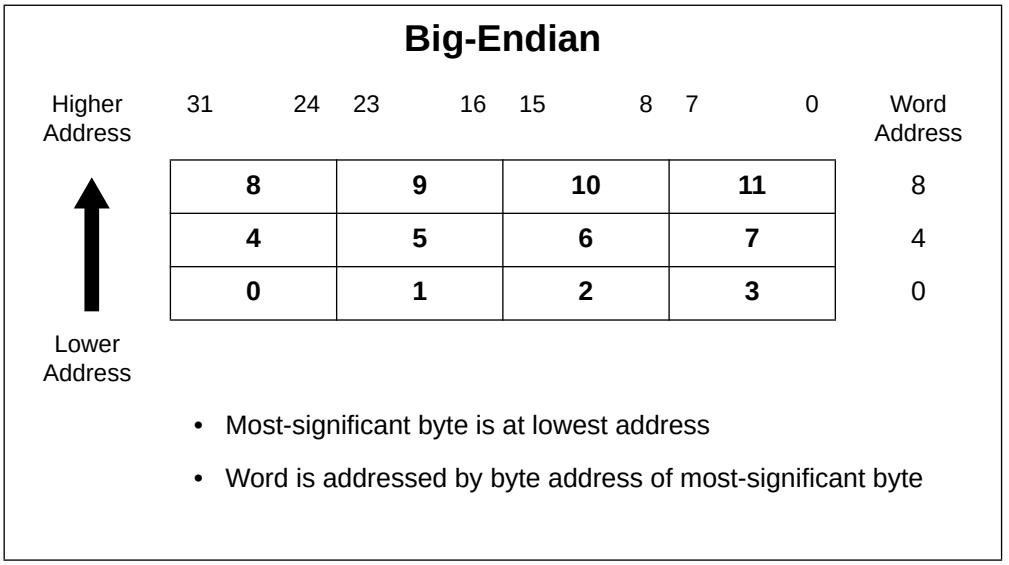


Figure 4-2: Big-endian addresses of bytes within words

4.2.2 Configuration bits for backward compatibility

Two register bits, PROG32 and DATA32, select one of three processor configurations:

1 **26-bit program and data space**

(PROG32 LOW, DATA32 LOW).
This configuration forces ARM processor to operate like the earlier ARM processors with 26-bit address space. The programmer's model for these processors applies, but the new instructions to access the CPSR and SPSR registers operate as detailed in *5.5 PSR Transfer (MRS, MSR)* on page 5-13. In this configuration it is impossible to select a 32-bit operating mode, and all exceptions (including address exceptions) enter the exception handler in the appropriate 26-bit mode.

2 **26-bit program space and 32-bit data space**

(PROG32 LOW, DATA32 HIGH).
This is the same as the 26-bit program and data space configuration, but with address exceptions disabled to allow data transfer operations to access the full 32-bit address space.

3 **32-bit program and data space**

(PROG32 HIGH, DATA32 HIGH).
This configuration extends the address space to 32 bits, introduces major changes in the programmer's model and provides support for running existing 26-bit programs in the 32-bit environment.

(The fourth processor configuration (26-bit data space and 32-bit program space) should not be selected.)

26-bit program space

When configured for 26-bit program space, ARM7500FE is limited to operating in one of four modes known as the 26-bit modes. These modes correspond to the modes of the earlier ARM processors and are known as:

- User26
- FIQ26
- IRQ26
- Supervisor26

Note: The PROG32 and DATA32 bits are used only for backward compatibility with earlier ARM processors and should normally be set to 1. The 32-bit mode is recommended for compatibility with future ARM processors and all new code should be written to use only the 32-bit operating modes.

Because the original ARM instruction set has been modified to accommodate 32-bit operation there are certain additional restrictions which programmers must note. Refer to the *ARM Application Notes "Rules for ARM Code Writers"* and *"Notes for ARM Code Writers"* available from your supplier.

4.3 Operating Mode Selection

The ARM processor has a 32-bit data bus and a 32-bit address bus. However, only 29 of the address bits are available at the ARM7500FE pins. The data types which the processor supports are:

- Bytes (8-bits)
- Words (32-bits), which must be aligned to four-byte boundaries.

Instructions are exactly one word, and data operations (e.g. ADD) are only performed on word quantities. Load and store operations can transfer either bytes or words.

ARM processor supports six modes of operation:

User mode	(usr)	The normal program execution state.
FIQ mode	(fiq)	Designed to support a data transfer or channel process.
IRQ mode	(irq)	Used for general purpose interrupt handling.
Supervisor mode	(svc)	A protected mode for the operating system.
Abort mode	(abt)	Entered after a data or instruction prefetch abort.
Undefined mode	(und)	Entered when an undefined instruction is executed.

Mode changes may be made under software control or may be brought about by external interrupts or exception processing. Most application programs execute in User mode. The other modes, known as *privileged modes*, are entered to service interrupts or exceptions, or to access protected resources.

4.4 Registers

The processor macrocell has a total of 37 registers made up of:

- 31 general 32-bit registers
- 6 status registers

At any one time 16 general registers (R0 to R15) and one or two status registers are visible to the programmer. The visible registers depend on the processor mode, and the other registers (the *banked registers*) are switched in to support IRQ, FIQ, Supervisor, Abort and Undefined mode processing.

The register bank organization is shown in *Figure 4-3: Register organization*. The banked registers are shaded in the diagram.

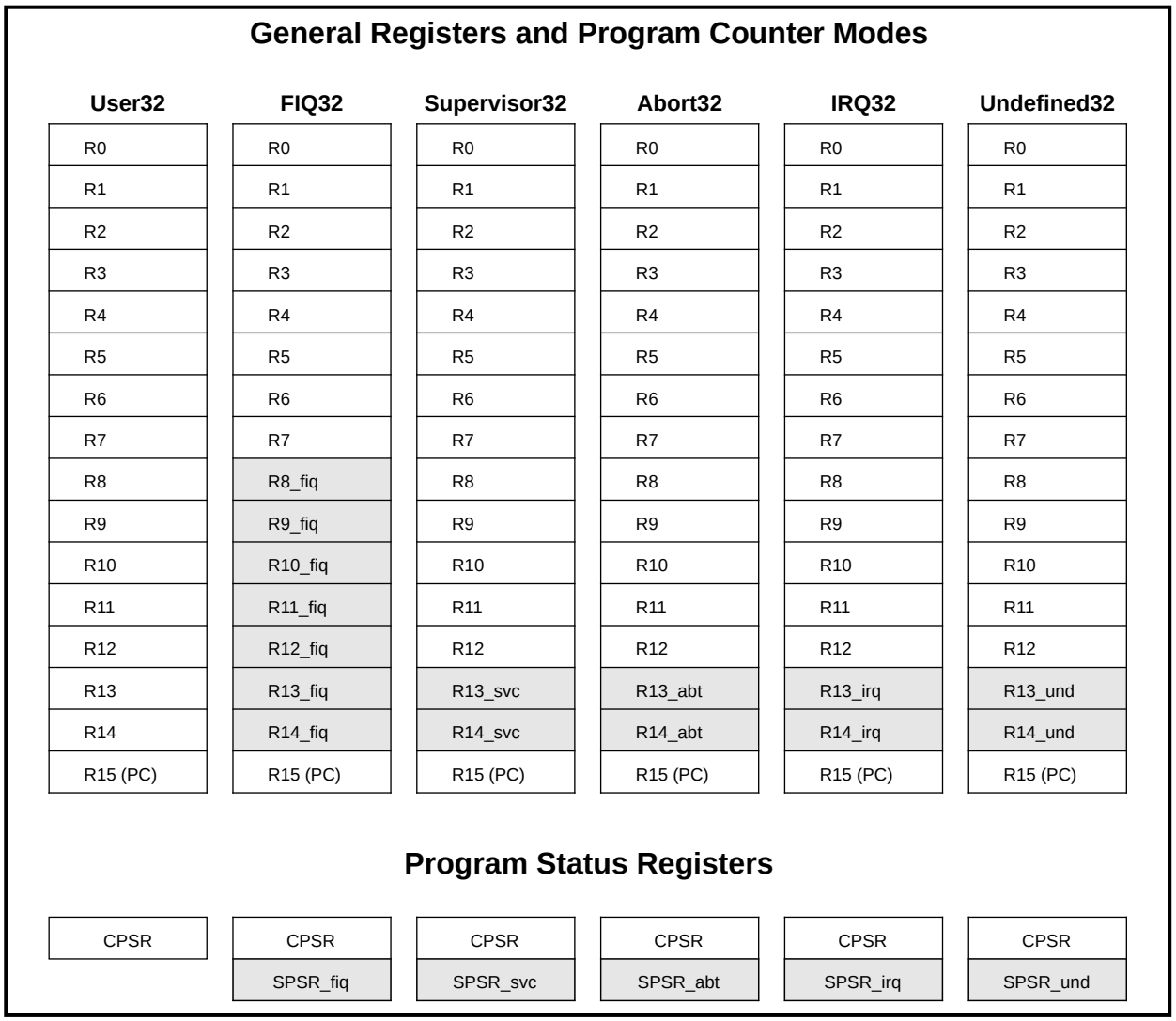


Figure 4-3: Register organization

In all modes, 16 registers (R0 to R15) are directly accessible. All registers except R15 are general-purpose and may be used to hold data or address values. Register R15 holds the Program Counter (PC). When R15 is read, bits [1:0] are zero and bits [31:2] contain the PC. A seventeenth register (the CPSR - Current Program Status Register) is also accessible. It contains condition code flags and the current mode bits and may be thought of as an extension to the PC.

R14 is used as the subroutine link register and receives a copy of R15 when a Branch and Link instruction is executed. It may be treated as a general purpose register at all other times. R14_svc, R14_irq, R14_fiq, R14_abt and R14_und are used similarly to hold the return values of R15 when interrupts and exceptions arise, or when Branch and Link instructions are executed within interrupt or exception routines.

FIQ mode has seven banked registers mapped to R8-14 (R8_fiq-R14_fiq). Many FIQ programs will not need to save any registers.

User mode, IRQ mode, Supervisor mode, Abort mode and Undefined mode each have two banked registers mapped to R13 and R14. The two banked registers allow these modes to each have a private stack pointer and link register.

Supervisor, IRQ, Abort and Undefined mode programs which require more than these two banked registers are expected to save some or all of the caller's registers (R0 to R12) on their respective stacks. They are then free to use these registers which they will restore before returning to the caller.

In addition, there are also five SPSRs (Saved Program Status Registers) which are loaded with the CPSR when an exception occurs. There is one SPSR for each privileged mode.

4.4.1 Program status registers

The format of the Program Status Registers is shown in *Figure 4-4: Format of the Program Status Registers (PSRs)*.

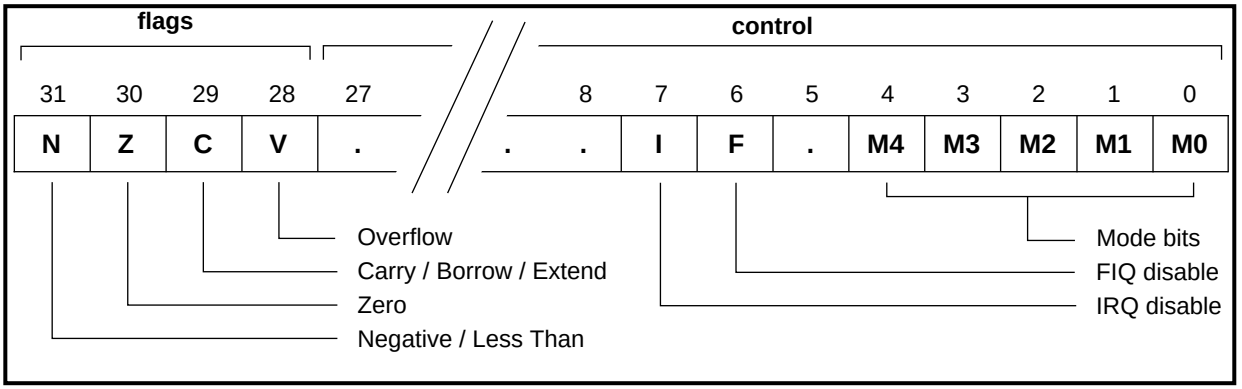


Figure 4-4: Format of the Program Status Registers (PSRs)

Condition code flags

The N, Z, C and V bits are the *condition code flags*. The condition code flags in the CPSR may be changed as a result of arithmetic and logical operations in the processor and may be tested by all instructions to determine if the instruction is to be executed.

Interrupt disable bits

The I and F bits are the *interrupt disable bits.* The I bit disables IRQ interrupts when it is set and the F bit disables FIQ interrupts when it is set.

Mode bits

The M0, M1, M2, M3 and M4 bits (M[4:0]) are the *mode bits*, and these determine the mode in which the processor operates. The interpretation of the mode bits is shown in *Table 4-1: The mode bits*. Not all combinations of the mode bits define a valid processor mode. Only those explicitly described shall be used.

M[4:0]	Mode	Accessible register set	
10000	User	PC, R14..R0	CPSR
10001	FIQ	PC, R14_fiq..R8_fiq, R7..R0	CPSR, SPSR_fiq
10010	IRQ	PC, R14_irq..R13_irq, R12..R0	CPSR, SPSR_irq
10011	Supervisor	PC, R14_svc..R13_svc, R12..R0	CPSR, SPSR_svc
10111	Abort	PC, R14_abt..R13_abt, R12..R0	CPSR, SPSR_abt
11011	Undefined	PC, R14_und..R13_und, R12..R0	CPSR, SPSR_und

Table 4-1: The mode bits

Control bits

The bottom 28 bits of a PSR (incorporating I, F and M[4:0]) are known collectively as the *control bits.* The control bits change when an exception arises and in addition can be manipulated by software when the processor is in a privileged mode. Unused bits in the PSRs are reserved and their state must be preserved when changing the flag or control bits. Programs must not rely on specific values from the reserved bits when checking the PSR status, since they may read as one or zero in future processors.

4.5 Exceptions

Exceptions arise whenever there is a need to break the normal flow of program execution. For example, the processor can be diverted to handle an interrupt from a peripheral. The processor state just prior to handling the exception must be preserved so that the original program can be resumed when the exception routine has completed. Many exceptions may arise at the same time.

The ARM processor handles exceptions by making use of the banked registers to save state. The old PC and CPSR contents are copied into the appropriate R14 and SPSR, and the PC and mode bits in the CPSR bits are forced to a value which depends on the exception. Interrupt disable flags are set where required to prevent otherwise unmanageable nestings of exceptions. In the case of a re-entrant interrupt handler, R14 and the SPSR should be saved onto a stack in main memory before re-enabling the interrupt.

Note: When transferring the SPSR register to and from a stack, it is important to transfer the whole 32-bit value, and not just the flag or control fields.

When multiple exceptions arise simultaneously, a fixed priority determines the order in which they are handled. The priorities are listed in *4.5.7 Exception priorities* on page 4-12.

4.5.1 FIQ

The FIQ (Fast Interrupt reQuest) exception is generated by the interrupt handler within the ARM7500FE. This input is delayed by one clock cycle for synchronization before it can affect the processor execution flow. It is designed to support a data transfer or channel process, and has sufficient private registers to remove the need for register saving in such applications (thus minimizing the overhead of context switching).

Note: The FIQ exception may be disabled by setting the F flag in the CPSR (but note that this is not possible from User mode).

If the F flag is clear, the ARM processor checks for a LOW level on the output of the FIQ synchronizer at the end of each instruction. When a FIQ is detected, the ARM processor performs the following:

1 Saves the address of the next instruction to be executed plus 4 in R14_fiq; saves CPSR in SPSR_fiq.

2 Forces M[4:0]=10001 (FIQ mode) and sets the F and I bits in the CPSR.

3 Forces the PC to fetch the next instruction from address 0x1C.

Returning from FIQ

To return normally from FIQ, use SUBS PC, R14_fiq,#4, which will restore both the PC (from R14) and the CPSR (from SPSR_fiq) and resume execution of the interrupted code.

4.5.2 IRQ

The IRQ (Interrupt ReQuest) exception is a normal interrupt caused by the interrupt handler within the ARM7500FE. It has a lower priority than FIQ, and is masked out when a FIQ sequence is entered. Its effect may be masked out at any time by setting the I bit in the CPSR (but note that this is not possible from User mode).

If the I flag is clear, the ARM processor checks for a LOW level on the output of the IRQ synchronizer at the end of each instruction. When an IRQ is detected, the ARM processor performs the following:

1 Saves the address of the next instruction to be executed plus 4 in R14_irq; saves CPSR in SPSR_irq.

2 Forces M[4:0]=10010 (IRQ mode) and sets the I bit in the CPSR.

3 Forces the PC to fetch the next instruction from address 0x18.

Returning from IRQ

To return normally from IRQ, use SUBS PC,R14_irq,#4, which will restore both the PC and the CPSR and resume execution of the interrupted code.

4.5.3 Abort

An ABORT is signalled by the internal Memory Management Unit, and indicates that the current memory access cannot be completed. For instance, in a virtual memory system the data corresponding to the current address may have been moved out of memory onto a disc, and considerable processor activity may be required to recover the data before the access can be performed successfully.

The abort mechanism allows a *demand paged virtual memory system* to be implemented when suitable memory management software is available.
The processor is allowed to generate arbitrary addresses, and when the data at an address is unavailable, the MMU signals an abort. The processor traps into system software which must work out the cause of the abort, make the requested data available, and retry the aborted instruction. The application program needs no knowledge of the amount of memory available to it, nor is its state in any way affected by the abort.

The ARM processor checks for ABORT during memory access cycles.
When successfully aborted ARM processor responds in one of two ways:

- prefetch abort
- data abort

Prefetch abort

If the abort occurred during an instruction prefetch (a *prefetch abort*), the prefetched instruction is marked as invalid but the abort exception does not occur immediately. If the instruction is not executed, for example as a result of a branch being taken while it is in the pipeline, no abort will occur. An abort will take place if the instruction reaches the head of the pipeline and is about to be executed.

Data abort

If the abort occurred during a data access (a *data abort*), the action depends on the instruction type:

- single data transfer instructions (LDR, STR) write back modified base registers and the Abort handler must be aware of this
- the swap instruction (SWP) is aborted as though it had not executed, though externally the read access may take place
- block data transfer instructions (LDM, STM) complete, and if write-back is set, the base is updated. If the instruction would normally have overwritten the base with data (i.e. LDM with the base in the transfer list), this overwriting is prevented. All register overwriting is prevented after the Abort is indicated, which means in particular that R15 (which is always last to be transferred) is preserved in an aborted LDM instruction.

Abort sequence

When either a prefetch or data abort occurs, ARM processor performs the following:

1. Saves the address of the aborted instruction plus 4 (for prefetch aborts) or 8 (for data aborts) in R14_abt; saves CPSR in SPSR_abt.
2. Forces M[4:0]=10111 (Abort mode) and sets the I bit in the CPSR.
3. Forces the PC to fetch the next instruction from either:
   - address 0x0C (prefetch abort) or
   - address 0x10 (data abort)

Returning from an abort

To return after fixing the reason for the abort, use SUBS PC,R14_abt,#4 (for a prefetch abort) or SUBS PC,R14_abt,#8 (for a data abort). This will restore both the PC and the CPSR and retry the aborted instruction.

4.5.4 Software interrupt

The software interrupt instruction (SWI) is used for getting into Supervisor mode, usually to request a particular supervisor function. When a SWI is executed, ARM processor performs the following:

1. Saves the address of the SWI instruction plus 4 in R14_svc; saves CPSR in SPSR_svc.
2. Forces M[4:0]=10011 (Supervisor mode) and sets the I bit in the CPSR.
3. Forces the PC to fetch the next instruction from address 0x08.

Returning from a SWI

To return from a SWI, use MOVS PC,R14_svc. This will restore the PC and CPSR and return to the instruction following the SWI.

4.5.5 Undefined instruction trap

When the ARM processor comes across an instruction which it cannot handle, it takes the undefined instruction trap. This includes all coprocessor instructions, except MCR and MRC operations which access the internal control coprocessor.

The trap may be used for software emulation of a coprocessor in a system which does not have the coprocessor hardware, or for general-purpose instruction set extension by software emulation.

When the ARM processor takes the undefined instruction trap, it performs the following:

1 Saves the address of the Undefined or coprocessor instruction plus 4 in R14_und; saves CPSR in SPSR_und.

2 Forces M[4:0]=11011 (Undefined mode) and sets the I bit in the CPSR.

3 Forces the PC to fetch the next instruction from address 0x04.

Returning from an undefined instruction trap

To return from this trap after emulating the failed instruction, use MOVS PC,R14_und. This will restore the CPSR and return to the instruction following the undefined instruction.

4.5.6 Vector summary

These are byte addresses, and will normally contain a branch instruction pointing to the relevant routine.

The FIQ routine might reside at 0x1C onwards, and thereby avoid the need for (and execution time of) a branch instruction.

Address	Exception	Mode on entry
0x00000000	Reset	Supervisor
0x00000004	Undefined instruction	Undefined
0x00000008	Software interrupt	Supervisor
0x0000000C	Abort (prefetch)	Abort
0x00000010	Abort (data)	Abort
0x00000014	-- reserved --	--
0x00000018	IRQ	IRQ
0x0000001C	FIQ	FIQ

Table 4-2: Vector summary

4.5.7 Exception priorities

When multiple exceptions arise at the same time, a fixed priority system determines the order in which they will be handled:

1 Reset (highest priority)

2 Data abort

3 FIQ

4 IRQ

5 Prefetch abort

6 Undefined Instruction, software interrupt (lowest priority)

Note: Not all exceptions can occur at once. Undefined instruction and software interrupt are mutually exclusive since they each correspond to particular (non-overlapping) decodings of the current instruction.

If a data abort occurs at the same time as a FIQ, and FIQs are enabled (i.e. the F flag in the CPSR is clear), the ARM processor will enter the data abort handler and then immediately proceed to the FIQ vector. A normal return from FIQ will cause the data abort handler to resume execution. Placing data abort at a higher priority than FIQ is necessary to ensure that the transfer error does not escape detection; the time for this exception entry should be added to worst-case FIQ latency calculations.

4.5.8 Interrupt latencies

Calculating the worst-case interrupt latency for the ARM processor is quite complex due to the cache, MMU and write buffer and is dependent on the configuration of the whole system.

4.5.9 Reset

When the ARM7500FE is reset, the ARM processor abandons the executing instruction and then performs idle cycles from incrementing word addresses.

When the ARM7500FE comes out of reset, the ARM processor does the following:

1 Overwrites R14_svc and SPSR_svc by copying the current values of the PC and CPSR into them. The value of the saved PC and CPSR is not defined.

2 Forces M[4:0]=10011 (Supervisor mode); sets the I and F bits in the CPSR.

3 Forces the PC to fetch the next instruction from address 0x00.

End of reset sequence

At the end of the reset sequence:

- the MMU is disabled and the TLB is flushed, so forces "flat" translation (i.e. the physical address is the virtual address, and there is no permission checking)
- alignment faults are also disabled
- the cache is disabled and flushed
- the write buffer is disabled and flushed
- the ARM7 CPU core is put into 26-bit data and address mode, little-endian mode

To make the ARM7 enter normal 32-bit operation, execute the following instructions at the start of the reset code to which the reset vector branches:

```
MOV R0, #0x70
MCR P15, 0, R0, C1, C0  ;Set 32-bit program and data
                        ;configuration
MOV R0, #0xD3           ;And enter Supervisor-32 mode with
MSR CPSR_c, R0          ;interrupts disabled
```

Also, make certain that this reset code lies within the first 32MB of memory to ensure that the instruction at the reset vector branches to the expected place even though the processor is operating in a 26-bit mode at the time.

4.6 Configuration Control Registers

The operation and configuration of the ARM processor is controlled both directly via coprocessor instructions and indirectly via the Memory Management Page tables.

The coprocessor instructions manipulate a number of on-chip registers which control the configuration of the Cache, write buffer, MMU and a number of other configuration options.

Backwards compatibility

To ensure backwards compatibility of future CPUs:

- all reserved or unused bits in registers and coprocessor instructions should be programmed to '0'.
- invalid registers must not be read/written.
- the following bits must be programmed to '0':
  Register 1 bits[31:11]
  Register 2 bits[13:0]
  Register 5 bits[31:0]
  Register 6 bits[11:0]
  Register 7 bits[31:0]

Note: *The areas marked "Reserved" in the register and translation diagrams should be programmed 0 for future compatibility.*

4.6.1 Internal coprocessor instructions

The on-chip registers may be read using MRC instructions and written using MCR instructions. These operations are only allowed in non-user modes and the undefined instruction trap will be taken if accesses are attempted in user mode.
Refer to *5.14 Coprocessor Register Transfers (MRC, MCR)* on page 5-41.

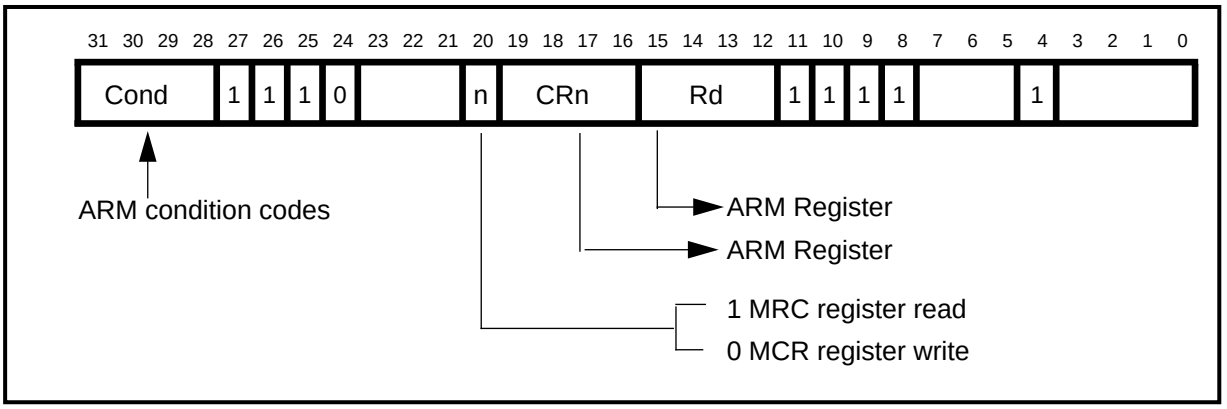


Figure 4-5: Format of Internal Coprocessor Instructions MRC and MCR

4.6.2 Registers

The ARM processor contains registers which control the cache and MMU operation. These registers are accessed using CPRT instructions to Coprocessor #15 with the processor in a privileged mode.

Only some of registers 0-7 are valid:

- an access to an invalid register will cause neither the access nor an undefined instruction trap, and therefore should never be carried out
- an access to any of the registers 8-15 will cause the undefined instruction trap to be taken.

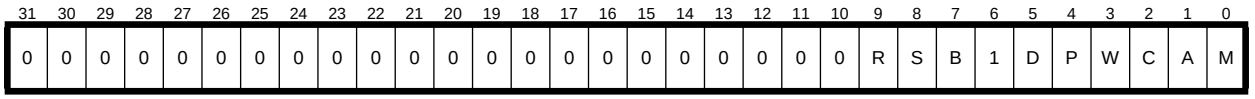
Register	Register reads	Register writes
0	CPU ID	Reserved
1	Reserved	Control
2	Reserved	Translation Table Base
3	Reserved	Domain Access Control
4	Reserved	Reserved

Table 4-3: Cache and MMU control registers

Register	Register reads	Register writes
5	Fault Status	Flush TLB
6	Fault Address	Purge TLB
7	Reserved	Flush IDC
8-15	Reserved	Reserved

Table 4-3: Cache and MMU control registers

Register 1: Control



Register 1 is write-only and contains control bits. All bits in this register are forced LOW by reset.

M Bit 0 Enable/disable

0 on-chip Memory Management Unit turned off
1 on-chip Memory Management Unit turned on.

A Bit 1 Address Fault Enable/Disable

0 alignment fault disabled
1 alignment fault enabled

C Bit 2 Cache Enable/Disable

0 Instruction / data cache turned off
1 Instruction / data cache turned on

W Bit 3 Write buffer Enable/Disable

0 Write buffer turned off
1 Write buffer turned on

P Bit 4 ARM 32/26-bit Program Space

0 26-bit Program Space selected
1 32-bit Program Space selected

D Bit 5 ARM 32/26-bit Data Space

0 26-bit Data Space selected
1 32-bit Data Space selected

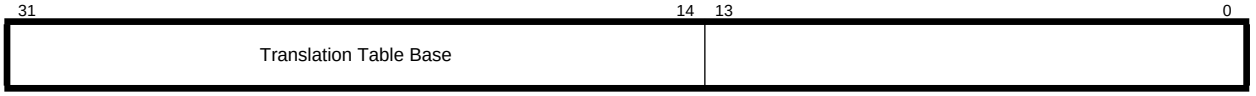
B Bit 7 Big/Little-Endian

0 Little-endian operation
1 Big-endian operation

S Bit 8 System bit, which controls the ARM processor permission system.

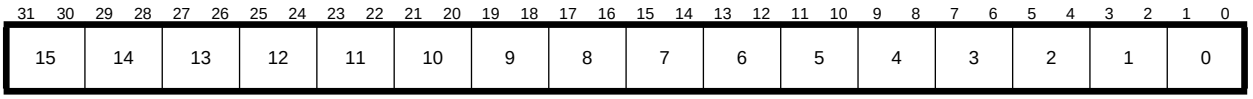
R Bit 9 ROM bit, which controls the ARM processor permission system

Register 2: Translation Table Base



Register 2 is a write-only register which holds the base of the currently active Level One page table.

Register 3: Domain Access Control

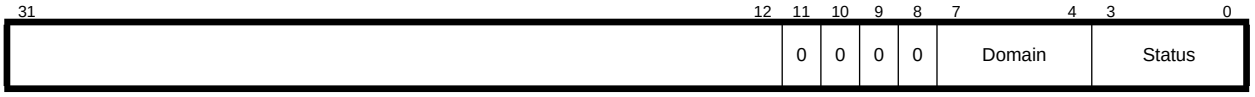


Register 3 is a write-only register which holds the current access control for domains 0 to 15. See *7.10 Domain Access Control* on page 7-13 for the access permission definitions and other details.

Register 4: Reserved

Register 4 is Reserved.
Accessing this register has no effect, but should never be attempted.

Register 5: Fault Status/Translation Lookaside Buffer Flush



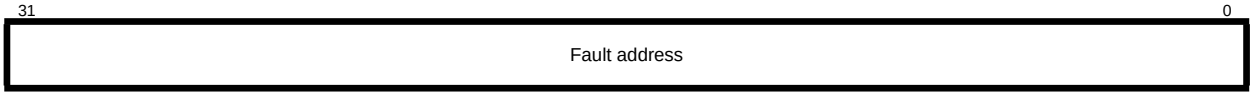
Read: Fault Status

Reading register 5 returns the status of the last data fault. It is not updated for a prefetch fault. See *Chapter 7: ARM Processor MMU* for more details. Note that only the bottom 12 bits are returned. The upper 20 bits will be the last value on the internal data bus, and therefore will have no meaning. Bits 11:8 are always returned as zero.

Write: Translation Lookaside Buffer Flush

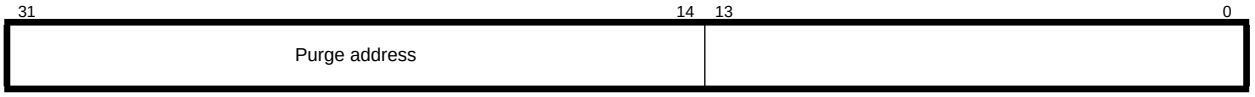
Writing Register 5 flushes the TLB. (The data written is discarded).

Register 6: Fault Address/ TLB Purge



Read: Fault Address

Reading register 6 returns the virtual address of the last data fault.



Write: TLB Purge

Writing Register 6 purges the TLB; the data is treated as an address and the TLB is searched for a corresponding page table descriptor. If a match is found, the corresponding entry is marked as invalid. This allows the page table descriptors in main memory to be updated and invalid entries in the on-chip TLB to be purged without requiring the entire TLB to be flushed.

Register 7: **IDC Flush**

Register 7 is a write-only register. The data written to this register is discarded and the IDC is flushed.

Registers 8 -15: **Reserved**

Accessing any of these registers will cause the undefined instruction trap to be taken.