20 Bus Interface

This chapter describes the ARM7500FE bus interface.

20.1 Bus Arbitration 20-2
20.2 Bus Cycle Types 20-2
20.3 Video DMA Bandwidth 20-3
20.4 Video DMA Latency 20-4

Bus Interface

20.1 Bus Arbitration

Arbitration for the main ARM7500FE data bus is carried out with the priorities shown below:

1 Video/cursor DMA

2 Sound DMA

3 DRAM refresh

4 ARM processor memory cycles

As the ARM7500FE contains a cached processor, ARM internal cycles can continue while DMA is in progress, but the CPU will stall when it suffers a cache miss and wishes to fill a cache line from memory.

Once an external memory cycle has started, DMA has to wait until it is completed. The exception is for I/O reads or writes and SUSPEND mode, where the write data is latched internally at the start of the cycle, after which DMA requests can be serviced even though the I/O access or SUSPEND mode is under way. The end of an I/O access is held up until the current DMA access is completed. I/O read data is latched internally when available, and is not enabled onto the ARM7500FE data bus until any DMA transfers have completed.

20.2 Bus Cycle Types

There are a large number of different types of cycle which make use of the ARM7500FE data bus. Except for DMA accesses, the cycle type is decoded according to the address put out by the ARM processor macrocell, and the detailed timing is controlled by the relevant section of the I/O or memory controller subsystem.

The ARM processor supports two basic types of external cycle:

non-sequential	consists of an Idle cycle followed by a memory cycle
sequential	consists simply of a memory cycle

The idle cycle allows the memory and I/O controller subsystems time to prepare for a new cycle type. These two cycles are used as the basic building block for the more complex I/O and memory access cycle timings generated by the ARM7500FE. ARM processor external cycles are clocked by the internal Mclk signal which is generated by the ARM7500FE's memory controller according to the type of cycle.

Only the latched version of the ARM processor's address is exported from the ARM processor, and this can only change immediately after the falling edge of the internal Mclk signal which clocks the ARM for external accesses. The timing diagrams in this datasheet may include Mclk as a reference as it indicates the end of a particular cycle. The ARM7500FE internal data bus is not always exported during internal register programming, to save power.

When the ARM processor requests an external memory access, it will do so for one of a number of reasons:

- A cache linefetch will always consist of memory reads from four sequential addresses.
- A level 1 translation fetch will consist of a read from memory followed by the address translation such that the next address put out by the ARM will be the translated physical address as generated from the read back section descriptor.
- A level 2 translation fetch is always preceded by a level 1 fetch, and returns the page table entry, which is then used to create the physical address for the next cycle.

External buffered and unbuffered write cycles take place with indistinguishable bus timing. When the ARM wishes to read from a location and the data is not in the cache or is uncacheable (eg. for I/O), then an external read access is performed.

20.3 Video DMA Bandwidth

The maximum video DMA bandwidth depends on the **MEMCLK** frequency and the DRAM width (16 or 32-bit), but can be calculated as follows.

The length of the non-sequential cycle at the start of a DRAM read will vary.

Assuming bit 5 of the DRAMCTL register is LOW:

- in Page Mode, each non-sequential cycle will take 5 cycles
- in EDO mode, each non-sequential cycle will take 6 cycles

This will be increased by 1 if Bit 5 is HIGH, and by a further 1 or 2 to preserve RAS precharge times, depending on whether the access just finished was to the same bank as the current one, and whether bit 6 of DRAMCTL is also set.

Assuming Fast Page Mode without further non-sequential delays, each quadword DMA requires 5+2+2+2 = 11 **MEMCLK** cycles to complete. It is possible for DMA requests for the video to be serviced sequentially such that the second and subsequent quadword DMA bursts take only 2+2+2+2=8 **MEMCLK** cycles each. However, all accesses will be broken up at page boundaries (every 256 words). So every 64 DMA bursts, there will be three extra **MEMCLK** periods required.

Therefore, at 32MHz **MEMCLK**, with 32-bit wide DRAM, 64 quadwords would be transferred approximately every 16us. The maximum theoretical DMA bandwidth is thus 63.6MBytes/second. If a greater video DMA bandwidth than this is required, a higher **MEMCLK** frequency will need to be used. In a real system, the average bandwidth will not achieve this theoretical maximum.

20.4 Video DMA Latency

DMA latency is defined to be the time from the generation of the internal request for more data from the video FIFO in the video macrocell, to the time at which the first word of DMA data is clocked into the video macrocell.

There are several possible limiting factors which may determine the worst case DMA latency, depending on the type of memory system with which ARM7500FE is configured to be used. There are three possible limiting cases:

1 Internal register programming cycles

2 Burst mode ROM accesses, or very long non sequential ROM accesses

3 DRAM accesses in 16-bit mode

The following assumes that the internal **MEMRFCK** frequency is equal to the **MEMCLK** frequency, ie the prescalers are set to divide-by-one. The above cases determine the maximum period before arbitration for DMA occurs in different systems. In addition to the latency resulting from these sequences, the worst case latency has a possible 5.5 **MEMCLK** cycles factor for synchronization, such that the synchronized request arrives just too late to be arbitrated for, and ARM7500FE commits to a memory cycle. The 5.5 **MEMCLK** cycles also includes the ARM processor idle cycle on which the arbitration (which was just missed) takes place.

From the clock edge at which arbitration finally takes place, to the time at which the first word of DMA data is clocked into the video macrocell, is 5.5 **MEMCLK** cycles, or 7.5 **MEMCLK** cycles if the preceding access was to DRAM in the same bank as this. These values assume bits [7:5] in DRAMCTL are all set HIGH; ie. EDO memory.

Internal register programming bursts can occur in blocks of up to four before re-arbitration takes place, and this will take 16 **MEMCLK** cycles. Burst mode ROM cycles are re-arbitrated after every four, as are sequential DRAM accesses. Successive non-sequential accesses will always allow DMA onto the bus, so it is unlikely that these will be the cause of the worst case latency. However, it would be possible to use the ROM interface in half speed mode, with the slowest ROM timing and a 16-bit-wide ROM, in which case an access would take 28 **MEMRFCK** cycles. Under these circumstances the ROM interface could be the limiting factor.

To determine the limiting factor in a system, calculate the number of cycles required for a worst case ROM access. The number of cycles for each programmed value in the ROMCR register is shown below:

For a non sequential access, programming bits 0-2:

000 - 7 cycles	
001 - 6 cycles	For all:
010 - 5 cycles	Multiply by 2 if 16-bit mode set
011 - 4 cycles	Multiply by 2 if half-speed bit set
100 - 3 cycles	
101 - 2 cycles	

If the burst bits (3-4) are programmed to a value other than 00, then the total worst case number of cycles will be one times the non-sequential number above, plus three times the burst number from below:

01 - 4 cycles	For all:
10 - 3 cycles	Multiply by 2 if 16-bit mode set
11 - 2 cycles	Multiply by 2 if half-speed bit set

Then calculate the number of cycles required for a worst case DRAM access. This can only be the limiting factor when 16-bit wide DRAM is used, and in this case the delay will be:

```
9 + (2x7) = 23 cycles
```

As described above, the worst case delay for four sequential internal register programming cycles is 16 cycles. So the worst case delay is caused by internal register access cycles, ROM or DRAM according to which of the above calculated figures is worst.

DMA can continue over the top of I/O accesses, so these do not feature in the options for worst case delay. So for a system which is limited by internal register access cycles, the worst case latency will be:

```
3.5 + 2 + 16 + 5.5 = 27 MEMCLK cycles.
```

So if **MEMCLK** is running at 32MHz, the total worst case DMA latency will be 0.84µs.

As another example, suppose that the ROM interface non sequential access time is programmed at 7 cycles, and the sequential programmed to 4, using 16-bit wide ROM. Then the total latency would be:

```
3.5 + 2 + 14 + 8 + 8 + 8 + 5.5 = 49 MEMCLK cycles.
```

At 32MHz this corresponds to 1.5µs.

21 Memory Map

This chapter gives details of the ARM7500FE memory map.

21.1 ARM7500FE Memory Map 21-2

21.1 ARM7500FE Memory Map

All addresses featured in the ARM7500FE memory map table are physical addresses. Only 29 bits of the address bus are available, which limits the total memory space to 512Mb.

Memory (Mbytes)	Address (Hex)	To (Hex)	Device
0	00000000	00FFFFFF	ROM bank 0
16	01000000	01FFFFFF	ROM bank 1
32	02000000	02FFFFFF	Reserved
48	03000000	0300FFFF	Module I/O space
	03010000	0302BFFF	16MHz PC style I/O
	0302C000	0302FFFF	Reserved
	03030000	0303FFFF	Further module I/O space
	03040000	031FFFFF	Reserved
	03200000	0320FFFF	ARM7500FE registers
	03210000	033FFFFF	Simple I/O space
	03400000	034FFFFF	Video registers
	03500000	03FFFFFF	Reserved
64	04000000	07FFFFFF	Reserved
128	08000000	0FFFFFFF	Extended I/O space
256	10000000	DRAM bank 0	
320	14000000	DRAM bank 1	
384	18000000	DRAM bank 2	
448	1C000000	DRAM bank 3	
512	20000000	ROM bank 0 (repeated)	

Table 21-1: ARM7500FE memory map table

22 DC and AC Parameters

This chapter gives the ARM7500FE DC and AC parameters.

22.1 Absolute Maximum Ratings 22-2
22.2 DC Operating Conditions 22-2
22.3 DC Characteristics 22-3
22.4 AC Parameters 22-4
22.5 De-rating 22-6

DC and AC Parameters

22.1 Absolute Maximum Ratings

Note: *These are stress ratings only. Exceeding the absolute maximum ratings may permanently damage the device. Operating the device at absolute maximum ratings for extended periods may affect device reliability.*

Symbol	Parameters	Min	Max	Units	Notes
VDD	Supply voltage	VSS-0.3	VSS+7.0	V	1
Vip	Voltage applied to any pin	VSS-0.3	VDD+0.3	V	1
Ts	Storage temperature	-40	125	deg C	1

Table 22-1: ARM7500FE DC maximum ratings

22.2 DC Operating Conditions

Symbol	Parameters	Min	Typ	Max	Units	Notes
VDD	Supply voltage	4.75	5.0	5.25	V	
Vihc	IC input HIGH voltage	0.8xVDD		VDD	V	1, 2
Vilc	IC input LOW voltage	0.0		0.2xVDD	V	1, 2
Viht	IT input HIGH voltage	2.3V		VDD	V	1, 3
Vilt	IT input LOW voltage	0.0		0.6V	V	1, 3
Vihs	IS input HIGH voltage	3.7		VDD	V	1, 5
Vils	IS input LOW voltage	0.0		1.6	V	1, 5
Vohc	OCZ output HIGH voltage	0.9xVDD		VDD	V	1, 4
Volc	OCZ output LOW voltage	0.0		0.1xVDD	V	1, 4
Ta	Ambient operating temperature	0		70	deg C	

Table 22-2: ARM7500FE DC operating conditions

Notes:

1 Voltages measured with respect to VSS.

2 IC - CMOS inputs

3 IT - TTL inputs (includes BTZ, TOD, and IT pin types)

4 OCZ - Output, CMOS levels, tri-stateable (includes OCZ, BTZ, TOD, and CSOD pin types)

5 IS - CMOS Schmitt inputs (includes ICS and CSOD pin types)

22.3 DC Characteristics

Symbol	Parameter	Min	Typ	Units	Note
IDD	Static Supply current		100	µA	1
Isc	Output short circuit current		100	mA	
Ilu	DC latch-up current		>500	mA	
Iin	IC input leakage current		1	uA	
Ioh1	x1 Output HIGH current (Vout = VDD-0.8V)		4	mA	
Iol1	x1 Output LOW current (Vout = VSS+0.4V)		-4	mA	
Ioh2	x2 Output HIGH current (Vout = VDD-0.8V)		12	mA	
Iol2	x2 Output LOW current (Vout = VSS+0.4V)		-12	mA	
Ioh3	x3 Output HIGH current (Vout = VDD-0.8V)		24	mA	
Iol3	x3 Output LOW current (Vout = VSS+0.4V)		-24	mA	
Vihst	IS input rising voltage threshold		3.58	V	2
Vilst	IS input falling voltage threshold		1.42	V	2
Cin	Input capacitance		3.0	pF	
ESD	HMB model ESD		4	KV	3

Table 22-3: ARM7500FE DC characteristics

Notes:

1 When the video subsystem is correctly powered down and ARM7500FE is in STOP mode.

2 IS - Schmitt trigger input.

3 This does not apply to the video and sound analog pins: **VIREF**, **ROUT**, **GOUT**, **BOUT.**

22.4 AC Parameters

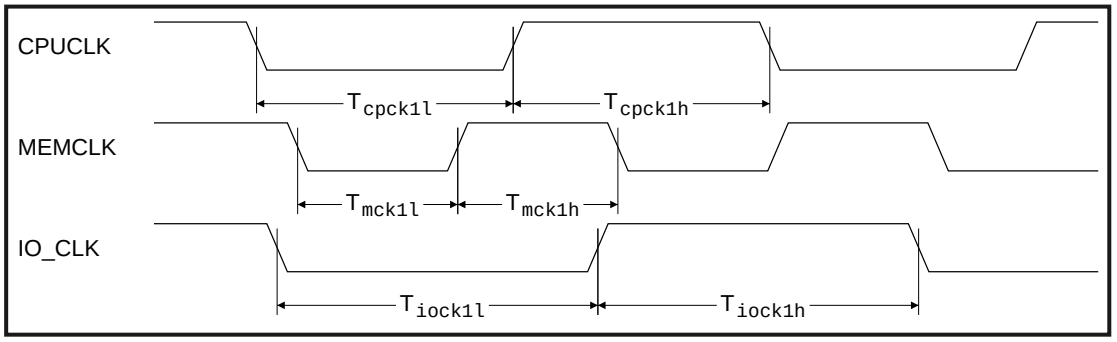


Figure 22-1: Clock timings with Divide-by-1 prescalers selected

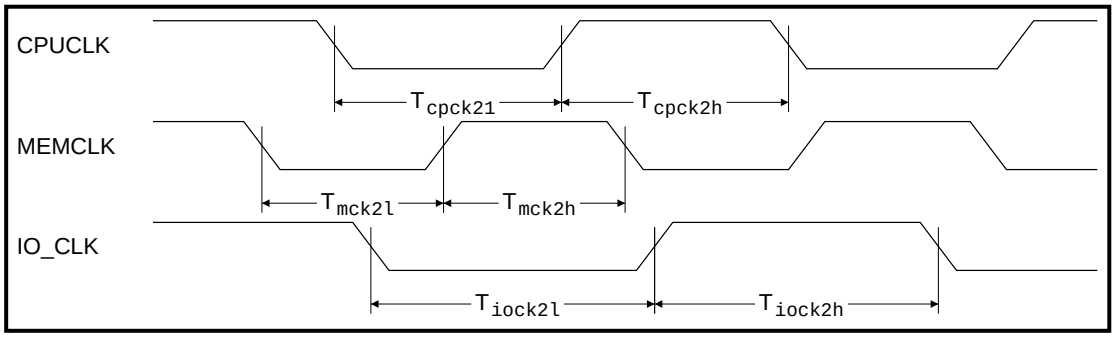


Figure 22-2: Clock timings with Divide-by-2 prescalers selected

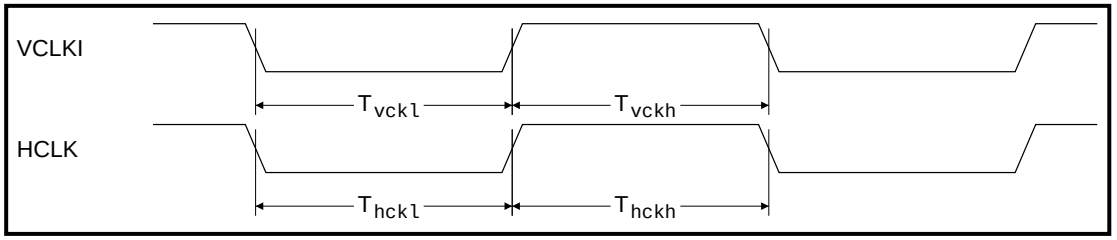


Figure 22-3: Video clock timing

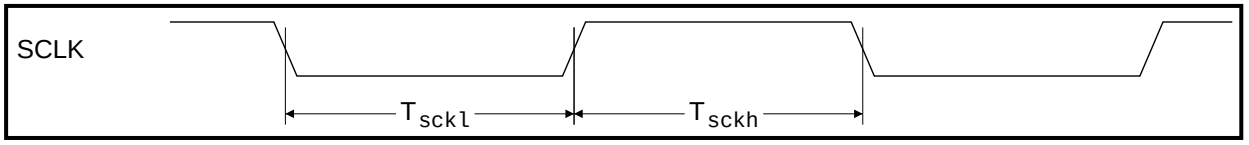


Figure 22-4: Sound clock timing

Symbol	Parameter	Min	Nominal	Units	Note
Tcpck1l	**CPUCLK** LOW time	12.5		ns	1
Tcpck1h	**CPUCLK** HIGH time	12.5		ns	1
Tmck1l	**MEMCLK** LOW time	7.8		ns	1
Tmck1h	**MEMCLK** HIGH time	7.8		ns	1
Tiock1l	**I_OCLK** LOW time		15.625	ns	1,2
Tiock1h	**I_OCLK** HIGH time		15.625	ns	1,2
Tcpck2l	**CPUCLK** LOW time	6.25		ns	3
Tcpck2h	**CPUCLK** HIGH time	6.25		ns	3
Tmck2l	**MEMCLK** LOW time	5		ns	3
Tmck2h	**MEMCLK** HIGH time	5		ns	3
Tiock2l	**I_OCLK** LOW time		7.8125	ns	3,4
Tiock2h	**I_OCLK** HIGH time		7.8125	ns	3,4
Tvckl	**VCLKI** LOW time	4		ns	
Tvckh	**VCLKI** HIGH time	4		ns	
Thckl	**HCLK** LOW time	4		ns	
Thckh	**HCLK** HIGH time	4		ns	
Tsckl	**SCLK** LOW time	TBD		ns	
Tsckh	**SCLK** HIGH time	TBD		ns	

Table 22-4: Clock timing

Notes:

1 Divide-by-1 prescaler selected.

2 **I_OCLK** = 32MHz in divide-by-1 mode.

3 Divide-by-2 prescaler selected.

4 **I_OCLK** = 64MHz in divide-by-2 mode.

All other ARM7500FE AC parameters and the associated timing diagrams have been included in the appropriate sections of the datasheet. The timing values shown are for the following conditions, as appropriate:

worst case	slow silicon, 100 deg junction temperature, VDD=4.75V
best case	fast silicon, 0 deg junction temperature, VDD=5.25V

22.5 De-rating

The AC timings included with each timing diagram in this datasheet include only the intrinsic delay through the output pads. In order to calculate actual delays when designing the ARM7500FE into a system, it is necessary to add the load-dependent element of the output pad delay.

The output pads of ARM7500FE are CMOS drivers which exhibit a propagation delay that increases linearly with the increasing capacitance. An *Output derating* figure is given for each of the three types of output pads, showing the increase in output delay with increasing load capacitance.

Details of which driver is used for which output can be found in *Chapter 2: Signal Description*.

De-rating figures are quoted for rising and falling edges.

Label	Pad type	Rising	Falling	Units
x1	Low drive capability pad	0.179	0.148	ns/pF
x2	Medium drive capability pad	0.054	0.052	ns/pF
x3	High drive capability pad	0.045	0.037	ns/pF

Table 22-5: ARM7500FE Pad de-rating

23 Packaging

This chapter describes the physical details of the ARM7500FE.

23.1 Pin Diagrams for the ARM7500FE 23-2

Packaging

23.1 Pin Diagrams for the ARM7500FE

The following two diagrams illustrate the top and side views of the ARM7500FE. All dimensions are given in millimeters.

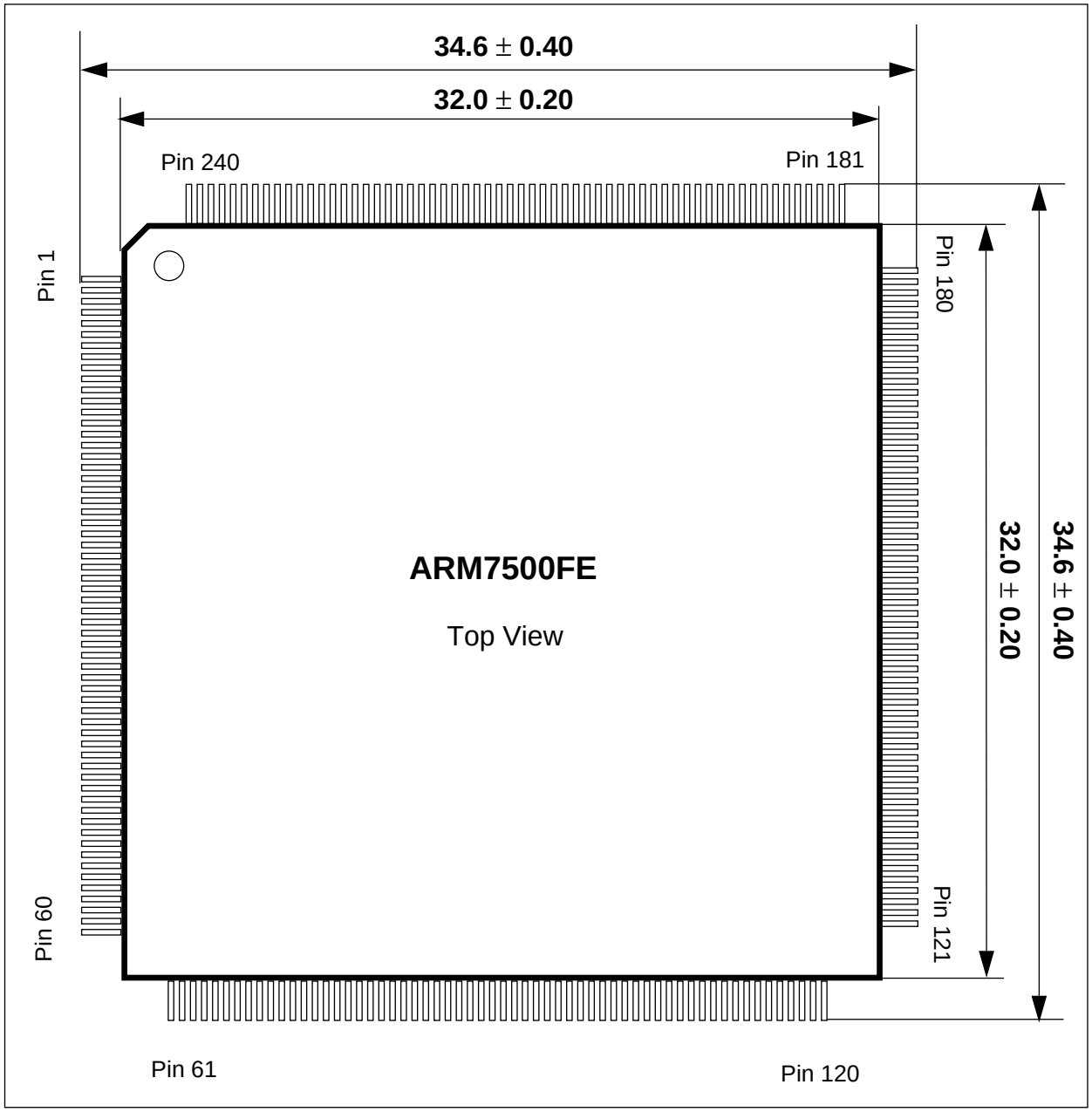


Figure 23-1: Pin diagram for the ARM7500FE

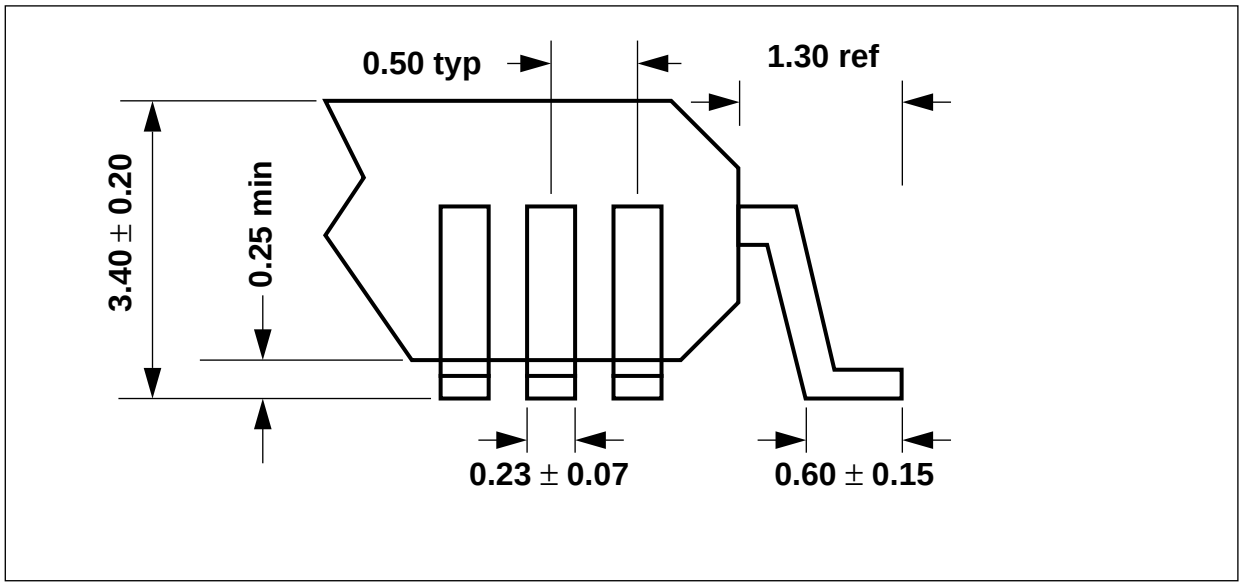


Figure 23-2: Side view of ARM7500FE chip

Packaging

24 Pinout

This chapter describes the ARM7500FE pinout.

24.1 Pin Details 24-2

Pinout

24.1 Pin Details

The following table gives the signal name for each of the 240 pins of the ARM7500FE.

Pin number	Signal name	Pin number	Signal name	Pin number	Signal name
1	LA[15]	29	D[21]	57	PCOMP
2	LA[16]	30	VSS_CORE	58	VSS
3	LA[17]	31	D[20]	59	VCLKI
4	LA[18]	32	VDD_CORE	60	VCLKO
5	LA[19]	33	D[19]	61	VDD
6	LA[20]	34	D[18]	62	VDD
7	LA[21]	35	VSS	63	VSS
8	VDD	36	D[17]	64	VSS
9	LA[22]	37	D[16]	65	VDD_CORE
10	VSS	38	D[15]	66	VSS
11	LA[23]	39	D[14]	67	VSS_CORE
12	LA[24]	40	D[13]	68	SDO
13	LA[25]	41	VDD	69	SCLK
14	LA[26]	42	D[12]	70	SDCLK
15	LA[27]	43	D[11]	71	WS
16	LA[28]	44	D[10]	72	SYNC
17	D[31]	45	D[9]	73	ECLK
18	D[30]	46	D[8]	74	VSS
19	D[29]	47	VSS	75	HCLK
20	D[28]	48	D[7]	76	ED[7]
21	VSS	49	D[6]	77	ED[6]
22	D[27]	50	D[5]	78	ED[5]
23	D[26]	51	D[4]	79	VDD
24	VDD	52	D[3]	80	ED[4]
25	D[25]	53	D[2]	81	ED[3]
26	D[24]	54	D[1]	82	ED[2]
27	D[23]	55	D[0]	83	ED[1]
28	D[22]	56	VDD	84	ED[0]

Pin number	Signal name	Pin number	Signal name	Pin number	Signal name
85	VSS	115	VSS	145	nEVENT2
86	VSYNC	116	nRAS[3]	146	BD[13]
87	VSS_CORE	117	VDD	147	BD[12]
88	HSYNC	118	nRAS[2]	148	BD[11]
89	VDD_CORE	119	nRAS[1]	149	VDD
90	VIREF	120	nRAS[0]	150	BD[10]
91	VDD_ANALOG	121	VDD_ATOD	151	VSS_CORE
92	ROUT	122	ATODREF	152	MEMCLK
93	BOUT	123	ATOD[3]	153	VDD_CORE
94	GOUT	124	ATOD[2]	154	BD[9]
95	VSS_ANALOG	125	ATOD[1]	155	BD[8]
96	nTEST	126	ATOD[0]	156	BD[7]
97	nINT8	127	VSS_ATOD	157	BD[6]
98	nINT3	128	nCAS[3]	158	BD[5]
99	nINT6	129	nCAS[2]	159	VSS
100	INT7	130	VSS	160	BD[4]
101	RA[11]	131	nCAS[1]	161	BD[3]
102	RA[10]	132	VDD	162	BD[2]
103	RA[9]	133	nCAS[0]	163	BD[1]
104	VSS	134	nWE	164	BD[0]
105	RA[8]	135	OSCPOWER	165	MSCLK
106	VDD	136	OSCDELAY	166	VDD
107	RA[7]	137	SnA	167	MSDATA
108	RA[6]	138	RESET	168	KBCLK
109	RA[5]	139	nRESET	169	KBDATA
110	RA[4]	140	nROMCS	170	VSS
111	RA[3]	141	BD[15]	171	nPOR
112	RA[2]	142	BD[14]	172	IOP[7]
113	RA[1]	143	I_OCLK	173	IOP[6]
114	RA[0]	144	VSS	174	IOP[5]

Pinout

Pin number	Signal name
175	IOP[4]
176	IOP[3]
177	IOP[2]
178	IOP[1]
179	IOP[0]
180	ID
181	OD[1]
182	OD[0]
183	SETCS
184	INT9
185	nINT4
186	INT5
187	READY
188	nIOGT
189	nBLI
190	nXIPMUX16
191	nINT1
192	INT2
193	VSS
194	nEVENT1
195	nXIPLATCH
196	TC
197	nSIOCS2
198	VDD
199	nSIOCS1
200	nEASCS
201	nMSCS
202	nBLO
203	nRBE
204	nWBE

Pin number	Signal name
205	CLK2
206	REF8M
207	CLK8
208	CLK16
209	nIORQ
210	VSS
211	nIOR
212	VSS_CORE
213	CPUCLK
214	VDD_CORE
215	nIOW
216	VDD
217	nCCS
218	nCDACK
219	IORNW
220	nPCCS2
221	nPCCS1
222	LNBW
223	LA[0]
224	LA[1]
225	LA[2]
226	VSS
227	LA[3]
228	LA[4]
229	LA[5]
230	LA[6]
231	LA[7]
232	LA[8]
233	VDD
234	LA[9]

Pin number	Signal name
235	LA[10]
236	LA[11]
237	LA[12]
238	VSS
239	LA[13]
240	LA[14]

A Initialization and Boot Sequence

This appendix describes the ARM7500FE initialization and boot sequence.

A.1 Introduction A-2

A.2 Sample Boot Sequence A-2

A.3 Other Methods A-3

Initialization and Boot Sequence

A.1 Introduction

ARM7500FE is designed to operate with 16 or 32-bit-wide memory systems. In order to avoid a hardware selection mechanism, the ARM7500FE is designed to always power-up with bit 6 of the ROMCR0 register set to 1, such that the chip expects to receive the first instructions from a 16-bit-wide ROM bank. For a system which is actually using 16-bit wide ROM, no special action is required. For a system which uses 32-bit wide ROM, a software solution is needed to enable the chip to boot successfully.

A sample method of programming the first locations of ROM in order to boot the device successfully is described in the following section. The example assumes that the reset vector is to be located at physical memory address zero.

A.2 Sample Boot Sequence

The processor will start executing code from physical address 0. As ARM7500FE is initially configured to operate with a 16-bit-wide ROM, it will fetch the lower half-word of the first instruction from the lower 16 bits of address 0, and the upper half-word of the instruction from the lower 16 bits of address 4.

If these first two locations have been programmed with instructions to load the PC with the reset and undefined instruction vectors, then the combination of the lower halfwords from the first and second location always creates an instruction with a never-true condition code, and so execution will drop through to the next instruction. This will be true for all the LDR PC instructions in the exception table. The exception table occupying the first eight locations in ROM is shown below.

This vector table resides at physical address 0.

Address	Instruction
0	LDR PC, RESET_VEC
4	LDR PC, UNDEF_VEC
8	LDR PC, SWI_VEC
C	LDR PC, PREF_VEC
10	LDR PC, DATA_VEC
14	LDR PC, RES_VEC
18	LDR PC, IRQ_VEC
1C	LDR PC, FIQ_VEC

Table A-1: Vector table

Immediately after the table, the ARM7500FE should be set into 32-bit mode. The eight locations from address 20 to 3C must be programmed with eight halfwords in the lower sixteen bits of each location, which will form the four required 32-bit instructions when read in pairs by the ARM7500FE. The upper 16 bits of each location will be ignored by the ARM7500FE while still in 16-bit mode.

The four instructions program the ROMCR0 register into 32-bit mode, and cause program execution to jump back to the reset vector at physical address zero, which will now be executed correctly. The MOV PC,#0 instruction which actually causes execution to jump back to zero will have been prefetched in 16-bit mode, even though it occurs after the ARM7500FE ROMCR0 register has been reprogrammed. *Table A-2: Instructions for programming the ROM register* shows the data required at memory locations 0x20 to 0x3C to implement this scheme.

Data	Address	Instruction	Notes
0x0000B632	20		
0x0000E3A0	24	`MOV R11, 0x03200000`	point at register base
0x00000000	28		
0x0000E3A0	2C	`MOV R0, #&0`	32b, slow, 218.75us, no burst
0x00000080	30		
0x0000E5CB	34	`STRB R0, [R11,0x80]`	Program ROMCR0 & switch mode
0x0000F000	38		
0x0000E3A0	3C	`MOV PC, #0`	Jump to 0

Table A-2: Instructions for programming the ROM register

The boot code above is a general example which will set the ROM interface to use the slowest access timing, to ensure it will work with all systems. It is advisable to program the ROM control registers early on with the fastest parameters usable by the interface, as this will drastically speed up execution. In addition, on power-up the default state of the CLKCTL register is for the CPUCLK, MEMCLK and I_OCLK external clock inputs to be divided by 2, and these should be programmed to divide-by-1 if appropriate. This will also speed up execution.

A.3 Other Methods

The above method is an example of how the ARM7500FE can be booted from a system using 32-bit-wide ROM. There are other methods of doing this which may be more appropriate for the required application. The main advantage of the method described above is that it allows the exception vector table to reside at physical address 0.

If this is not a requirement the instructions which reprogram the ROMCR0 register could reside from location 0 onwards, and the vector table can be mapped into DRAM by the operating system software.

B Dual Panel Liquid Crystal Displays

This appendix describes dual-panel LCD driving within the video and sound macrocell.

B.1 Programming the Video Subsystem B-2

B.2 Configuring DMA within ARM7500FE B-3

B.3 Cursor B-3

B.1 Programming the Video Subsystem

The external register (address 0xC00xxxxx) bit 13 (lcd) must be programmed to one, as for normal LCD operation.

Bit 13 of the control register (address 0xE00xxxxx) must be programmed to one. This is the 'dup' bit to set duplex mode operational.

Video data will be channelled simultaneously to the top and bottom halves of the screen. The first quadword received from memory will be interpreted for the first part of the first raster in the top half of the screen, and the second quadword will be interpreted for the identical part of the lower half of the screen. ARM7500FE will handle the sequencing of DMA data so that the video buffer can still be programmed as though there was only one panel.

When the cursor is moved, in addition to the programming of the Vertical Cursor start (VCSR) and end (VCER) registers and the horizontal cursor start (HCSR) register as described below.

Bits 13 and 14 of VCSR (address 0x9600xxxx) should be programmed to:

14:13

0 0	Dual Panel mode not activated
0 1	Cursor in upper half screen
1 0	Cursor in lower half screen
1 1	Cursor straddles both halves

Normally VCSR defines the number of rasters from Vsync to the start of the cursor, and VCER defines the number of rasters from Vsync to the end of the cursor display. See *Chapter 12: The Video and Sound Programmer's Model* for details of exactly how these are programmed.

Split-screen operation

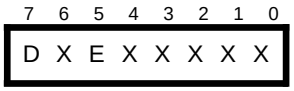
For split-screen operation, the programming of VCER and VCSR will be the same as for a single panel LCD when the cursor is completely in the top or bottom half of the screen, but when the cursor straddles the boundary, the values of these two registers will have a different meaning. The value in the VCSR register will be the number of rasters from the top of the lower panel to the end of the cursor image, and VCER will be programmed with the number of rasters from the top of the display to the start of the cursor image in the upper panel.

The cursor is displayed in the lower half screen from the value of VDSR to VCSR, and in the upper half screen from the value of VCER to VDER. So, the start register is effectively defining the “end” of the cursor in the bottom half, and the end register is defining the “start” of the cursor in the top half. This is the case because the top of the lower half of the screen will be written to before the bottom of the upper half.

B.2 Configuring DMA within ARM7500FE

The video and sound macrocell must first be programmed to drive dual panel LCDs as above. When this has been done, the macrocell will always make quad-word DMA requests in pairs. ARM7500FE is then set into dual panel mode by programming bit 7 (“dup”) of the Video Control register VIDCR (Address 0x1E0) to 1. The eight bits of the Video Control register are now allocated as follows

VIDCR (address 0x 1E0)



X = Undefined

E = Enable

D = Duplex LCD

When duplex mode is enabled, ARM7500FE will DMA two quad words from memory, offset by half the size of the video buffer, to enable two parallel data streams to be output by the video and sound macrocell to the two panels of the LCD. All DMA is quad-word only, so the auto increment of the DMA address is now always 0x10.

The VIDSTART and VIDEND registers will be programmed in the normal way, as for a single panel, with the addresses of the first and last quad-words in memory. The VIDINITA register should be programmed with the address of the first quadword to be displayed on the upper panel of the LCD, and the VIDINITB register with the address of the first quadword to be displayed on the lower panel of the LCD. The difference between the two addresses should be half the number of bytes in the video buffer. It is possible for VIDINITA to be pointing to an address in the lower half of the buffer, in which case VIDINITB should be set to point to an address in the top half of the buffer, offset by half the buffer size again.

If either of the INIT register values are equal to the END register, then bit 30 of the relevant INIT register must be set HIGH for correct operation (the “last” bit).

Note: Both “last” bits should never be programmed HIGH at the same time.

B.3 Cursor

In order to ensure smooth transition of the cursor across the dual panel boundary, it is necessary to have four images of the cursor stored in memory. This is because the ARM7500FE DMA registers must only be programmed with quadword-aligned addresses, but as the cursor is always 32 pixels wide at 2 bits per pixel, the address of data corresponding to a particular row of the cursor may be aligned with a two-word boundary.

The four images should be arranged as two pairs of contiguous images of the cursor. Only alternate rows of each cursor image will start on quad word boundaries, for reasons stated above, and so the two pairs of images are offset so that the first has all its odd rows starting on quad-word boundaries, and the second has all its even rows starting on quad word boundaries. This means that ARM7500FE can address any row of the cursor using only quadword-aligned DMA pointers.

Normally, only the first image will be used. However, when the cursor happens to be straddling the split-screen boundary, a different strategy is adopted. The VCSR and VCER registers in the video and sound macrocell are programmed differently as described above, and the cursor init register must be set to point to the location corresponding to the position of the row of the cursor which appears at the top of the lower part of the screen. In conjunction with the different meaning of the vertical cursor position registers in the video and sound macrocell, this will enable a smooth transition across the boundary.

B.3.1 Video frame buffer restrictions

In order for the dual-panel LCD to be driven correctly, it is necessary for the video frame buffer to contain an even number of quadwords, and to be aligned to a quad-word boundary. The cursor buffer must also be aligned to a quadword boundary.

C Using ASTCR at High MEMCLK Frequencies

This appendix describes the use of the ASTCR register.

C.1 Using the ASTCR Register C-2

Using ASTCR at High MEMCLK Frequencies

C.1 Using the ASTCR Register

Whenever the ARM processor performs a memory cycle, it is clocked by MCLK which is derived from **MEMCLK**. The I/O controller inside ARM7500FE is clocked by derivatives of **I_OCLK**. Thus, when the ARM processor performs a read from or a write to an area of I/O space, some synchronization must occur.

The ARM7500FE bus controller decodes the address of the ARM processor access and if it recognizes it as an I/O access, must send an I/O cycle request signal to the I/O controller. This is synchronized to the internal I/O clock, IOCK32. The I/O controller then performs the necessary cycle asserting one (or more) of the I/O chip select signals, eg. nCCS.

When the I/O controller knows the I/O cycle is about to finish, it asserts an I/O grant signal which is synchronized back to the internal memory clock, MEMRFCK.
The Bus controller will then terminate the cycle by creating a falling edge on MCLK which clocks the ARM processor.

The address from the ARM processor is latched when MCLK is LOW so that it is held stable throughout I/O cycles (as well as ROM). It is therefore important that MCLK should not fall too quickly after the end of the I/O chip select, else the address may change too quickly violating the required hold time. ARM7500FE has been designed to support **MEMCLK** running at a frequency much higher than **I_OCLK**.
In this situation, the I/O grant generated by the I/O controller will be synchronized more quickly back to MEMRFCK and so the address will change sooner after the end of the I/O chip select. Thus the I/O controller must delay the point at which it generates the I/O grant to ensure the address hold time is maintained.

A technique using the ASTCR register bit, 0x032000CC, has been employed to allow the address hold time to be maintained when **MEMCLK** frequency is greater than **I_OCLK** frequency whilst not imposing greater than necessary wait states when **MEMCLK** has the same or lower frequency than **I_OCLK**.

For a given system, the **I_OCLK** frequency should be fixed at 32MHz, while **MEMCLK** frequency will be fixed according to the speed grade of DRAMs being used.
The amount of hold time required between the end of the I/O chip select and the latched address changing should be determined and then ASTCR should be set dependent on the following details.

C.1.1 ASTCR I/O cycle type and hold times

Note: This assumes divide-by-1 mode for the clocks, **MEMCLK** and **I_OCLK**.

When ASTCR is LOW (reset value):

I/O cycle type	*Minimum Hold time*
Simple I/O	2 **MEMCLK** periods minus 1 **I_OCLK** period
Module I/O	2 **MEMCLK** periods minus 1.5 **I_OCLK** periods
PC style I/O	2 **MEMCLK** periods minus 1.5 **I_OCLK** periods

When ASTCR is HIGH:

I/O cycle type	*Minimum Hold time*
Simple I/O	2 **MEMCLK** periods minus 0.5 **I_OCLK** periods
Module I/O	2 **MEMCLK** periods minus 0.5 **I_OCLK** periods
PC style I/O	2 **MEMCLK** periods minus 0.5 **I_OCLK** periods

C.1.2 Example

For example, in a system with:

- **I_OCLK**=32MHz
- **MEMCLK**=40MHz

the minimum hold time for a PC-style access will be:

- 3.125ns if ASTCR=0
- 34.375ns if ASTCR=1

In addition there will be a small amount of extra hold time due to the delay from the internal memory clock to the latch enable signal for the address.

It should be further noted that these times refer to the signals changes at the pad on the inside of ARM7500FE. The relative capacitive loading of the latched address and I/O chip select will determine the overall timing.

D Expanding PC-Style I/O to 32 Bit

This appendix describes the extension of PC-style I/O to 32 bit.

D.1 32-bit I/O D-2

Expanding PC-Style I/O to 32 Bit

D.1 32-bit I/O

ARM7500FE provides 16-bit I/O accesses as standard using the BD[15:0] port for all I/O types. The PC-style I/O accesses, however, can be extended to allow full 32-bit accesses without any loss in access speed by the addition of external 16-bit transceivers. ARM7500FE provides all the control signals necessary to support these external devices.

During PC-style I/O write cycles, the I/O controller routes the lower 16-bit halfword from the ARM processor's data bus onto BD[15:0] and drives the upper 16-bit halfword onto D[31:16].

During read cycles, the ARM processor's data bus is driven from two sources:

- the lower halfword from the data latched from BD[15:0]
- the upper halfword from D[31:16]

If the external devices to provide the upper halfword of data are not present, or the I/O peripheral does not support more than 16-bits, the software must ignore the upper halfword read back into the ARM processor registers.

Figure D-1: 32-bit I/O interface shows an example of the system connections required to provide a full 32-bit I/O interface.

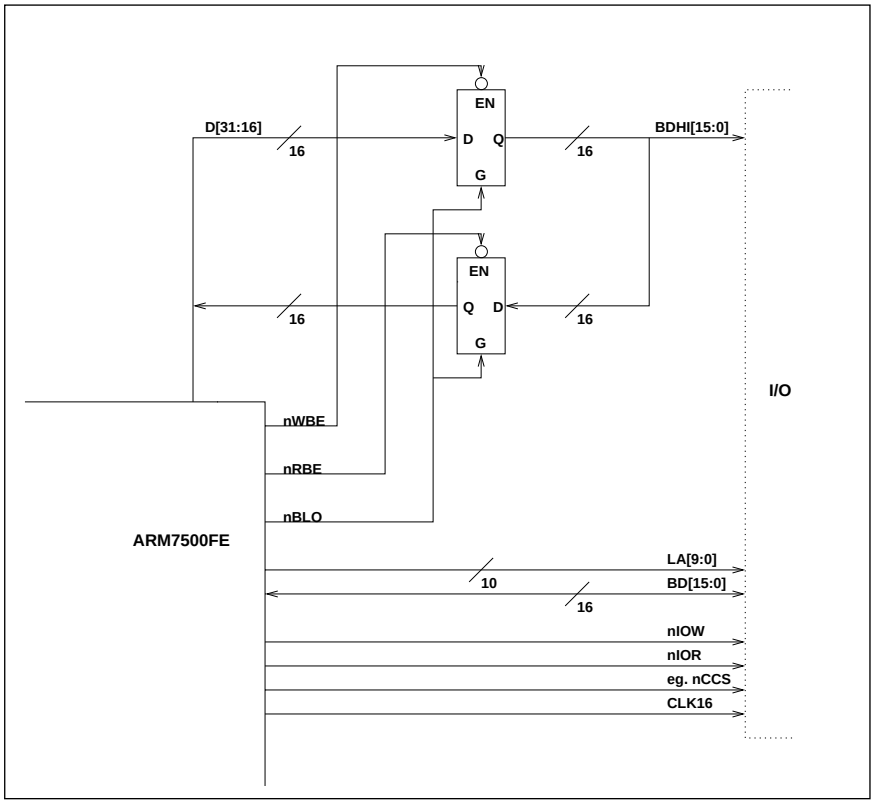


Figure D-1: 32-bit I/O interface

The write and read path should each contain a 16-bit latch with tri-state output enable control:

- The write latch should latch data from D[31:16] when **nBLO** is HIGH and drive the latched data onto the expanded I/O bus, BDHI[15:0], when **nWBE** is active LOW.
- The read latch should latch data from BDHI[15:0] when **nBLO** is HIGH and drive the latched data onto D[31:16], when **nRBE** is active LOW.

Note: *Like the BD[15:0] bus, the write enable **nWBE** remains active LOW by default. It is de-asserted only during the read cycles, thus the I/O device must not attempt to drive BD[15:0] or BDHI[15:0] except when a read cycle is taking place.*

E ARM7500FE Video Clock Sources

This appendix describes the ARM7500FE video clock sources.

E.1	Introduction	E-2
E.2	Clock Sources	E-2
E.3	Using the Phase Comparator	E-3
E.4	Phase Comparator Reset	E-6

ARM7500FE Video Clock Sources

E.1 Introduction

In order to facilitate the high-resolution screen modes that ARM7500FE is capable of producing, a suitable high-frequency clock must be applied. As screen mode is changed, the pixel rate must also change. This can be done:

- via the various clock inputs
- by the on-chip *pre-scaler*
- by using an external *voltage controlled oscillator* in conjunction with the on-chip phase comparator, to form a phase-locked-loop (PLL).

It is intended that most systems be built with a phase-locked-loop system. The required circuitry is simple, and allows a high degree of flexibility. The advantages are that all the necessary clock frequencies can be derived from the one circuit, and so the requirement for multiple on-board crystals and clock-switching circuitry is eliminated.

E.2 Clock Sources

ARM7500FE has three primary inputs for its pixel clock. These are:

- **HCLK**
- **VCLKI**
- RCLK (this is the internal 32MHz IOCK32, which is derived from **I_OCLK**)

The intention is that **VCLKI** and the internal IOCK32 signal (derived from **I_OCLK**) be used to drive the phase comparator, and that **HCLK** would only be used to provide the highest-frequency clock if this frequency is above the maximum VCO frequency.

The pixel clock source is selected by programming bits 0 and 1 of the control register. The pixel clock selected can then be passed through a pre-scaler which can divide the clock by between 1 and 8. This is done by programming bits 2 to 4 of the control register. See *12.27 Control Register (conreg): Address 0xE* on page 12-16.

SCLK

In addition to the pixel clock inputs, there is one other clock input, **SCLK**.

The sound system can be clocked from the internal 32MHz IOCK32 or a 16MHz **SCLK** (there is a divide-by-2 in the sound system). The digital sound system may run at a different frequency, (low MHz range), and this must be applied directly on **SCLK**.

Note: Any unused clock pin should be tied low.

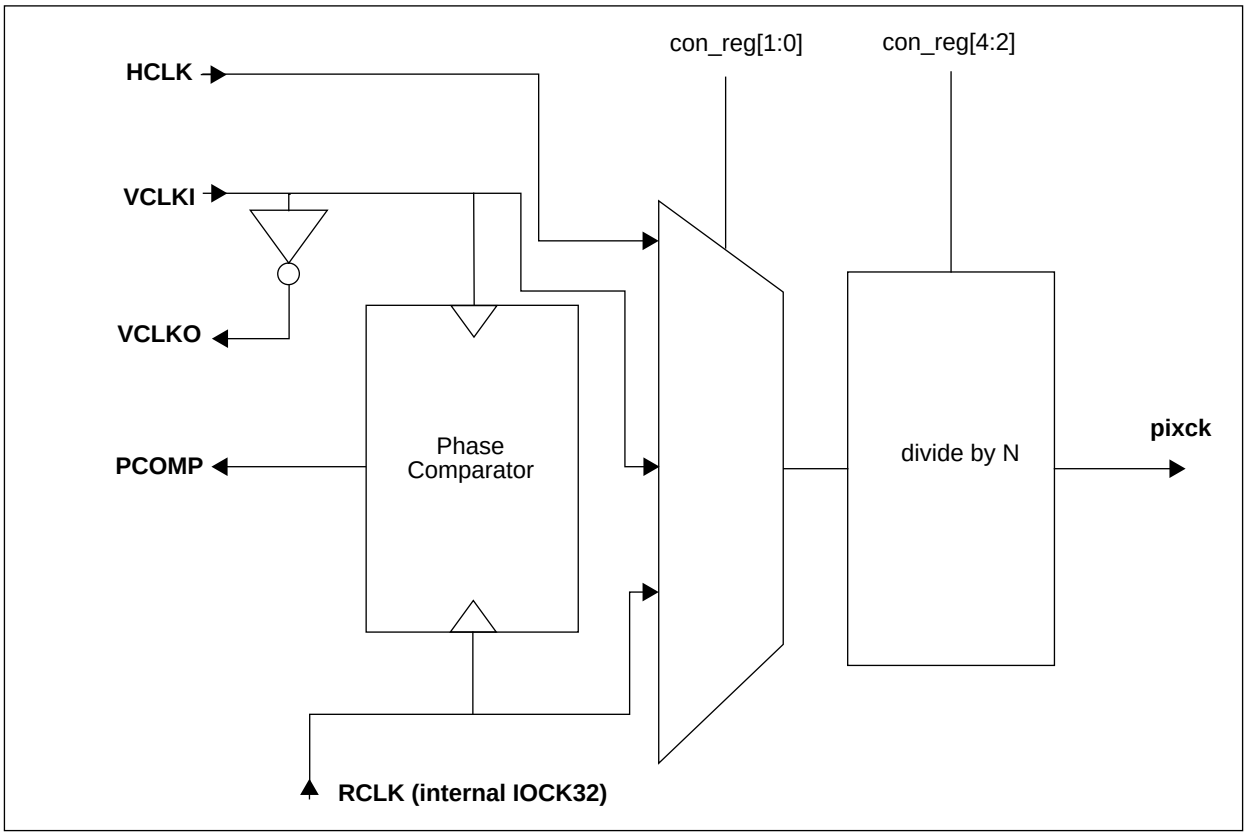


Figure E-1: Video and sound macrocell internal clock system

E.3 Using the Phase Comparator

The Video and sound macrocell contains a phase comparator which, in conjunction with an external voltage controlled oscillator (VCO), can be used to build a phase-locked-loop.

The phase comparator comprises:

- two counters
- a phase detector

The counters are pre-loadable down counters, one clocked from the internal IOCK32 signal, derived from **I_OCLK**, and the other clocked from **VCLKI**. The moduli of the counters is programmed in the Frequency Synthesizer Register.

In this register, the test bits have the following meaning:

bit [6] force **PCOMP** high and driven

bit [7] clear r-modulus counter

bit [14] force **PCOMP** low and driven

bit [15] clear v-modulus counter

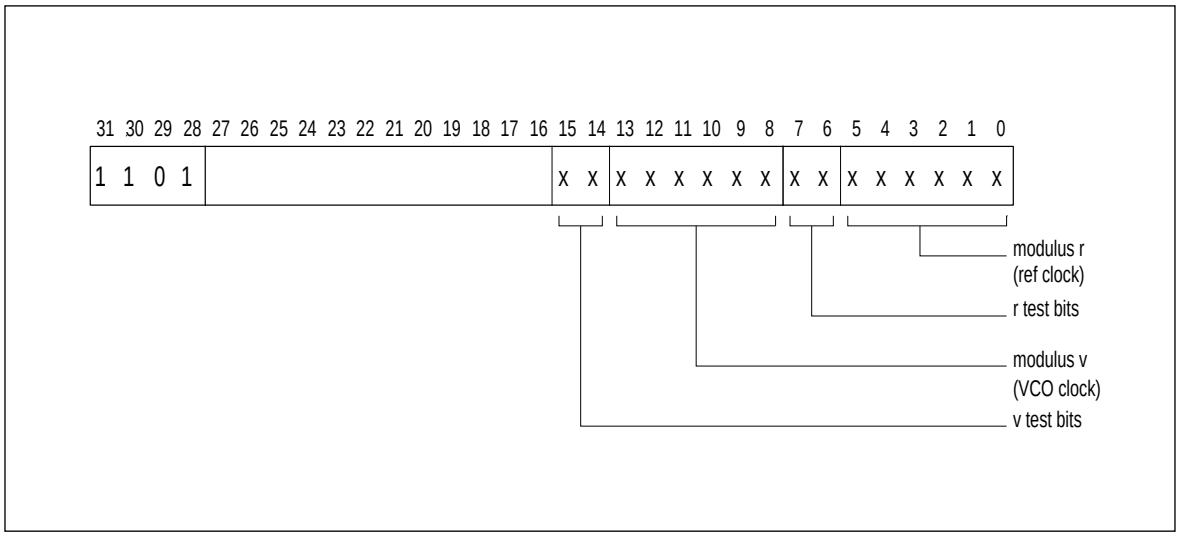


Figure E-2: Frequency Synthesizer Register

These bits are only programmed during test and at reset (see section *E.4 Phase Comparator Reset*).

The internal IOCK32 signal, derived from the **I_OCLK** input, provides a reference clock which is recommended to be 32MHz. The **VCLKI** input is driven from the output of the VCO, and it is this which is selected as the pixel clock.

The VCO is driven by the ARM7500FE's **PCOMP** output, which for most of the time is at the tri-state value.

When the VCO's frequency needs to be increased, **PCOMP** goes high, and vice-versa when the frequency needs to be decreased. The **PCOMP** output needs to be filtered before applying to the VCO.

The choice of filter and VCO are left to the user. A very simple and effective system can be built using an 74AC04 inverter pack, and a very simple LC filter. The filtered VCO output controls the operating voltage of the 74AC04 device. This system is shown in *Figure E-3: Suggested VCO/PLL circuit*, and gives an enormous range of frequencies (LF to hundreds of MHz).

Since the output of this VCO is AC coupled, **VCLKI** needs to be biased at the mid-voltage point. This is done by connecting a large resistor between **VCLKI** and **VCLKO** (**VCLKO** is the inversion of **VCLKI**).

Note: *Low-power systems may want to use more complex circuitry here to avoid DC paths during SUSPEND or STOP modes.*

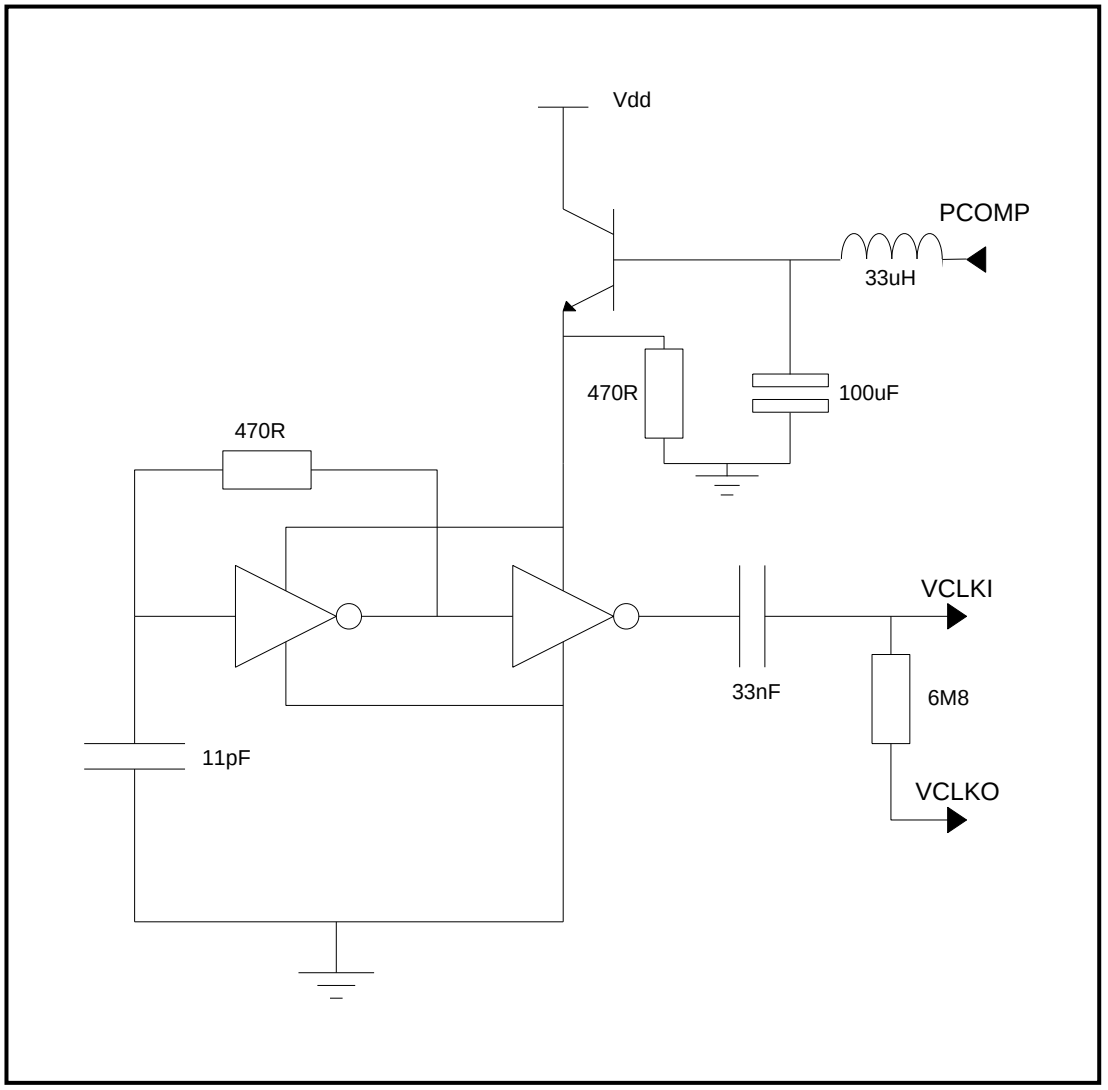


Figure E-3: Suggested VCO/PLL circuit

The actual frequency of the VCO is determined by the ratio of the v-modulus to the r-modulus as follows.

$$F_{VCO} = F_{REF} \times \frac{Vmodulus}{Rmodulus}$$

Note: For a modulus of r, r-1 is programmed, and likewise for the v modulus.

Table 24-1: Synthesized VCO frequency settings gives a list of useful frequencies with corresponding values of r and v moduli, assuming a reference frequency of 32MHz. Obviously there are many values of r and v which give the same ratio. The lower the values, the more frequently the output of the VCO will be updated and so the r and v values should be chosen to suit the response of the filter.

r-modulus	v-modulus	VCO frequency/MHz
8	2	8.0
16	6	12.0
4	2	16.0
8	6	24.0
2	2	32.0
8	9	36.0
16	35	70.0
4	15	120.0

Table 24-1: Synthesized VCO frequency settings

E.4 Phase Comparator Reset

The phase comparator and VCO form a closed-loop feedback system which has potential to become unstable. If the system powers up in the state where the **PCOMP** output is trying to drive the VCO's output higher and higher, it will very quickly reach a frequency which the phase comparator cannot resolve and thus recovery is impossible.

24.1.1 Reset procedure

To avoid this, the following reset procedure must be applied carefully.

The test bits in the Frequency Synthesizer Register can be used to force the phase comparator's output either HIGH or LOW. Thus, soon after power-up, this register must be programmed with:

- bits 15, 14 and 7 high
- bit 6 low

The r and v moduli can have anything programmed into them, but r must be greater than v. This operation forces the VCO's frequency to decrease.

When the real pixel rate is to be programmed, it should be done in two steps:

1 The values of the r and v moduli should be programmed, but the test bits left in the initialization state.

2 All the test bits should be cleared.

The VCO will then ramp up to its operating frequency. Subsequently, a change of frequency can be achieved simply by reprogramming the r and v moduli.

F

ARM7500FE Test Modes

This appendix describes the ARM7500FE test modes.

F.1 Introduction F-2

F.2 Test Modes Description F-2

ARM7500FE Test Modes

F.1 Introduction

ARM7500FE has a pin, **nTEST,** which is used in combination with the **nINT8**, **nINT3** and **nINT6** pins to set the device into various test modes. Most of these are intended only for use during production test to allow the individual macrocells within ARM7500FE to be tested directly from the external pins using a mux isolation scheme.

F.2 Test Modes Description

When the **nTEST** pin is HIGH, ARM7500FE is in normal operating mode irrespective of the states of **nINT8**, **nINT3** and **nINT6**. However, when **nTEST** is set LOW, the chip is set into one of five possible test modes dependent on the state of the three inputs **nINT8**, **nINT3** and **nINT6**. Four of these test modes are reserved for use on the tester.

However there is one test mode which, when selected, will cause all the ARM7500FE outputs to be tri-stated. This test mode is accessed by setting **nTEST**=0, **nINT8**=0, **nINT3**=1 and **nINT6**=1.

No other combinations should be selected by the user.

Index

A

A to D convertors 1-6
Abort mode 4-6
aborts 4-9, 5-22, 5-29, 5-33, 5-39
    external 7-16
AC parameters 22-4
    test conditions 22-6
address alignment 5-26, 5-39
address translation 7-4
addressing modes 5-26, 5-39
alignment faults 7-15
analogue outputs 14-12
analogue to digital convertors 18-34
ARM processor 1-2
assembler syntax 5-4, 5-12, 5-15, 5-18, 5-23, 5-30, 5-33, 5-34, 5-37, 5-40, 5-42, 5-43
asynchronous mode 19-2
auto-indexing 5-19

B

backward compatibility 4-4
    floating-point code 10-7
banked registers 4-5
base registers
    inclusion of 5-29
    restrictions 5-22
Big Endian 4-2, 5-22, 5-47
block data transfer 5-24
block diagram
    ARM704 3-4
branch 5-3
    with link 5-3
branch instructions 5-3
bufferable bit 6-4
bufferable write 6-4
bus interface 13-2, 20-2

C

cache 6-2
cacheable bit 6-2
CD offset registers 12-6
clock control 1-4, 19-2
clock prescalers 19-3
clocking schemes 19-3
comment field 5-34
comparators 18-36
compilers 3-3
condition code flags 4-7
condition field 5-2
conditional instructions
    using 5-44
configuration bits
    for backward compatibility 4-3
configuration control registers 4-13

configurations 4-2, 4-13
control 4-16
control bits 4-7
control register 12-16, 18-36
convertor operation 18-37
convertors
  analogue to digital 18-34
coprocessors 4-14, 6-5
  data operations 5-36
  data transfers 5-38
  fields 5-37, 5-38, 5-41
  instructions 5-36
    ARM704 5-36
  register transfers 5-41
counters 18-34
CPSR flags 5-6, 5-17
CPU
  aborts 7-12
  clock 19-2
cursor 14-5
  Hi-Res mode 14-5
  LCD mode 14-5
cycle times 5-11, 5-15, 5-17, 5-23, 5-29, 5-33, 5-34, 5-37, 5-39, 5-42

D

DAC control 14-12
  pedestal current 14-12
  power-save mode 14-12
data aborts 4-10, 5-22
data control register 12-17
data processing 5-4
DC
  characteristics 22-3
  operating conditions 14-11, 17-7, 17-17, 17-18, 18-10, 18-14, 18-28, 18-29, 18-33, 19-7, 19-8, 22-2
  operation 6-2
  validity 6-2
descriptors 7-5, 7-6
digital conversion 18-34
display modes 11-3
DMA 1-5
  channels 17-22
    video 17-22
domain access control 4-17, 7-13
domain access control register 7-3
domain faults 7-15
DRAM interface 17-8
  address multiplexing 17-9
  control registers 17-8
  self-refresh 17-20
  timing specification 17-11
dual panel LCDs 14-9, B-3

E

EDO DRAM 17-8
  read timing (16-bit mode) 17-16
  read timing (32-bit mode) 17-13
  timing mode selection 17-10
exceptions 4-6, 4-8
  priorities 4-12
external aborts 7-16
external register 12-14
external support 14-9

F

Fast Interrupt Request. See FIQ
faults
  address register 7-3
  addresses 7-12
  checking sequences 7-14
  status register 7-3
  status registers 7-12
FIFO
  setting preload value 13-2
FIQ 4-6, 4-8
floating-point accelerator. See FPA
FPA
  backward compatibility 10-7
  block diagram 8-5
  Control Register 9-11
  functional blocks 8-3
  IEEE conformance 10-16
  instruction cycle timing 10-17
  instruction set 10-14
    coprocessor data operations 10-7
    coprocessor data transfer 10-2
    coprocessor register transfer 10-11
    load and store floating 10-2
    load and store multiple floating 10-4
  mnemonics 10-7

number formats 9-4
double-precision 9-4
extended double precision 9-5
extended packed decimal 9-7
packed decimal 9-6
single-precision 9-4
overview 8-2
Status Register 9-8
support code 10-16
frequency synthesizer register 12-15
functional block diagram 1-2

G

genlocking 14-11

H

hardware cursor 11-2
Hi-Res mode 14-5, 14-6
horizontal
border start register 12-8
cycle register 12-8
sync width register 12-8

I

I/O
address space usage 18-3
chip select decode logic 18-4
clock outputs 19-2
control 1-5
general purpose port 18-38
ID and open drain pins 18-38
lines 1-6
Module 18-11
PC bus style 18-15
Simple 8MHz 18-4
system clock 19-2
ID register 18-39
IDC 6-2
IDC flush 6-2
immediate operand rotates 5-10
Instruction and Data Cache 6-2
instruction set 5-2
ARM704 3-2
FPA 10-2
summary of ARM704 5-2
instructions
cycle times 5-4, 5-11, 5-15, 5-17, 5-23, 5-29, 5-33, 5-34, 5-37, 5-39, 5-42
multiply 5-16
specified shift amounts 5-7
speed summary 5-47
undefined 5-43
using conditional 5-44
interface
serial sound 15-2
status of 18-35
video and sound macrocell 13-2
internal coprocessor instructions 4-14
interrupt latencies 4-12
Interrupt Request. See IRQ
interrupts 4-6, 4-10
control 18-34, 18-39
disable bits 4-7
handler 1-6
in timers 18-38
latencies 4-12
IRQ 4-9

K

keyboard interface 18-30

L

large page translation 7-11
LCD mode 14-5
LCDs 14-8
dual panel 1-2, 14-9
grey-scaling 14-8
monochrome 1-2
single panel 1-2, 14-9
LDC 5-38
LDM 5-24
LDR 5-18, 5-22
LDRB 5-21, 5-22
level one descriptor 7-5
level one fetch 7-4
link bit 5-3
Liquid Crystal Displays 14-8
Little Endian 4-2, 5-21, 5-47
loading words from an unknown alignment 5-47

M

MCR 5-41
MEMCLK C-2
Memory Management Unit 7-2
memory map 21-2
memory subsystem clock 19-2
memory system 1-5
MMU 7-2
MMU faults 7-12
mode bits 4-7
modes
    of operation 4-4
Module I/O 1-6, 18-11
monochrome output 14-12
mouse interface 18-30
MRC 5-41
multimedia 1-2
multiplication by constant
    using the barrel shifter 5-46
multiply 5-16
    instructions 5-16
multiply-accumulate 5-16

O

offsets 5-19
on-chip sound system 11-4
opcodes 5-11
operand restrictions 5-13, 5-17
operating mode selection 4-4
operating modes 4-2

P

Page Table Descriptor 7-6
Pages 7-2
palette 11-3, 14-4
    updating 14-4
PC bus style I/O 1-6, 18-15
permission faults 7-15
permissions 7-2
physical addresses 7-2
physical details 23-2
pin details 24-2
pin diagrams 23-2
pixel clock 11-3, 14-2
power consumption 19-4
power management 1-4, 11-3, 19-4
power saving 14-11
prefetch abort 4-9
program-accessible registers
    MMU 7-2
programmable registers 16-2
    interface 18-30
pseudo random binary sequence generator 5-45
PSR transfer 5-13

R

R14 4-6
R15 4-6, 5-11, 5-22, 5-28, 5-33, 5-39, 5-42
    using as an operator 5-11
    writing to 5-11
read-lock-write 6-4
register configuration 4-2
registers 4-2, 4-5, 4-14, 7-2
    configuration control 4-13
    domain access control 7-3
    fault 7-12
    fault address 7-3
    fault status 7-3
    inclusion of the base 5-29
    keyboard interface 18-30
    list of 5-24
    mouse interface 18-30
    programmable 16-2
    restrictions on the use of base registers 5-22
    shifted offsets 5-20
    specified shift amounts 5-10
    version and ID 18-39
    video and sound macrocell 12-3
reserved bits 5-13
reset 4-12, 7-17, 19-6
ROM interface 17-2
rotates 5-10

S

S bit 5-28
Sections 7-2
serial ports 1-6
serial sound interface 15-2
setting FIFO preload value 13-2
shifted register offsets 5-20
shifts 5-7

signals
  descriptions of 2-3
Simple I/O 1-5, 18-4
single data swap 5-32
single data transfer 5-18
single panel LCDs 14-9
small page translation 7-10
software IDC flush 6-2
software interrupt 4-10
software interrupts 4-10, 5-34
sound 15-2
  core 15-2
  serial interface 15-2
sound control register 12-18
sound frequency register 12-17
sound subsystem
  clock 19-2
sound system 11-4
specified shift amounts 5-7
  by registers 5-10
speed of instructions
  summary 5-47
status
  of interface 18-35
STC 5-38
STM 5-24
STOP mode 19-5
STR 5-18, 5-21, 5-22
STRB 5-21, 5-22
Supervisor mode 4-6, 4-10
SUSPEND mode 19-4
swap 5-32
SWI 4-10, 5-34
SWP 5-32
synchronization
  vertical and horizontal 14-11
synchronous mode 19-2

T

table base 7-4
test modes 1-6
timers 18-37
  interrupts 18-38
  programming 18-38
translation 7-4
translation faults 7-15
translation table base 4-16
  register 7-3

U

unbufferable writes 6-4
undefined instruction 5-43
undefined instruction trap 4-11
Undefined mode 4-6
using R15 as an operator 5-11

V

vectors 4-11
Version register 18-39
vertical registers 12-10
video and sound macrocell 1-4, 11-2
  interface 13-2
  sound features 15-2
video DAC currents 14-12
video DMA 17-22
video frame buffer restrictions B-4
video palette register 12-5
video subsystem
  clock 19-2
video system 11-2
virtual addresses 7-2

W

wb 6-3
write buffer 6-3
  disabling 6-4
  enabling 6-4
  operation 6-4
writing to R15 5-11