18 I/O Subsystems

This chapter describes the ARM7500FE I/O subsystems.

I/O Subsystems

18.1 Introduction

ARM7500FE has a 16-bit wide general I/O port, BD[15:0]. This allows slow I/O access to continue independently of DMA activity on the ARM7500FE data bus. There are three types of I/O access supported over the I/O bus:

- 16MHz PC-style I/O
- 8MHz request/grant-based I/O
- simple 8MHz-based fixed timing I/O

ARM7500FE also has a separate 8-bit wide general purpose open drain I/O port, each bit of which can be configured as an interrupt source. There are four analog comparators, each with a 16 bit 2MHz timer which can be used as a four channel analog joystick interface. Two identical PS/2 serial mouse/keyboard ports are included. There are two general-purpose 2MHz 16-bit counter timers, which can be programmed to produce interrupts at timed intervals.

ARM7500FE includes an interrupt handler, with enable and mask bits for each interrupt source, which can process potential interrupts from a number of internal and external sources.

The 16MHz PC style I/O provides all the signals required to interface with a standard PC Combo chip, enabling an industry standard part to be used to complete the I/O interfaces to devices such as a floppy disc.

The facility is available to expand the width of the I/O bus externally by adding latches and buffers to the upper 16 bits of the main external data bus and control signals for these devices are provided from ARM7500FE.

Support is provided for Execute-in-place (XIP) from a 16-bit wide PCMCIA card attached to the I/O bus, using an external PCMCIA controller.

Because the I/O clocks can be completely asynchronous to the memory system clock (which is controlling the main bus arbitration state machine), there will be additional synchronization penalties at the start and end of the I/O cycle. The exact additional delay will depend on the actual phase of the clocks at the point in question, and the timing diagrams do not attempt to show this in detail. However, the worst case synchronization delays are indicated.

18.2 I/O Address Space Usage

The main I/O address space is defined as being from address 0x03000000 to 0x03FFFFFF, as shown in *Table 18-1: I/O address space usage* on page 18-3.

In addition, there is an extended I/O address space for 16MHz PC style I/O from address 0x08000000 up to 0x0FFFFFFF, divided into eight 16MB areas. The chip select generated throughout this area is **nEASCS**.

I/O address	Contents
0x03000000	Module space - asserts **nMSCS**
0x03010000	16MHz I/O - asserts **nCCS** (Combo chip select)
0x03012000	16MHz I/O - asserts **nCDACK** (Combo DACK)
0x0302A000	16MHz I/O - asserts **nCDACK** and **TC** (Combo DACK and TC)
0x0302B000	16MHz I/O - asserts **nPCCS2**
0x0302B800	16MHz I/O - asserts **nPCCS1**
0x0302C000	Reserved
0x03030000	Module space - asserts **nMSCS**
0x03040000	Reserved
0x03200000	ARM7500FE internal I/O and memory control registers
0x03210000	Simple I/O space - asserts **nSIOCS1/2**
0x03400000	ARM7500FE internal video and sound control registers
0x03500000	Reserved

Table 18-1: I/O address space usage

18.3 Additional I/O Chip Select Decode Logic

The **SETCS** input selects additional decode logic for some of the chip select outputs.

- When **SETCS** is HIGH:

  nMSCS is asserted only in the following ranges of Module I/O space:

  0x03000000 -> 0x03003FFF
  0x03030000 -> 0x03033FFF

  nEASCS is asserted only in the following range of Extended I/O space:

  0x08000000 -> 0x08FFFFFF

  nSIOCS2 is asserted only in the following ranges of Simple I/O space:

  0x03240000 -> 0x03243FFF
  0x032C0000 -> 0x032C3FFF
  0x03340000 -> 0x03343FFF
  0x033C0000 -> 0x033C3FFF

- When **SETCS** is LOW:

  nMSCS is asserted over the whole of Module space

  nEASCS is asserted over the whole of Extended I/O address space

  nSIOCS2 is asserted only in the following ranges of simple I/O space:

  0x03240000 -> 0x0324FFFF
  0x032C0000 -> 0x032CFFFF
  0x03340000 -> 0x0334FFFF
  0x033C0000 -> 0x033CFFFF

18.4 Simple 8MHz I/O

The Simple I/O type of access is 16-bit only and has a selection of 4 different cycle speeds selectable by bits 20 and 19 of the address. This type of I/O will be selected for addresses in the range 0x3210000 to 0x32FFFFFF. When writing, the upper halfword of the ARM7500FE data bus is written out on the I/O bus. When reading, the I/O bus data is read back onto the lower half-word of the ARM7500FE data bus. This type of I/O cycle is not affected by the **READY** signal.

During these accesses, the signal **nSIOCS1** is always asserted with a read or write strobe as appropriate based on the CLK8 8MHz clock. **nSIOCS2** is asserted according to the decoding in the section above. The read and write strobes are the **nIOR** and **nIOW** output pins respectively. The four timings of the Simple 8MHz I/O accesses are shown below:

Address [20:19]	Name	Minimum CLK8 cycles
0 0	slow	7
0 1	medium	6
1 0	fast	5
1 1	sync	5

Table 18-2: Timings of the Simple 8MHz I/O accesses

The “sync” timing is referenced to the 2MHz CLK2 output, and there will thus be an additional possible synchronization penalty of up to 3 CLK8 cycles depending on the phase of CLK2 and CLK8 at the commencement of the I/O cycle. This is in addition to synchronization between the I/O and memory subsystem signals.

The diagrams below show the timing of the four different types of simple I/O cycles.

Note: *All diagrams assume **I_OCLK** is running at 32MHz using divide-by-1 mode.*

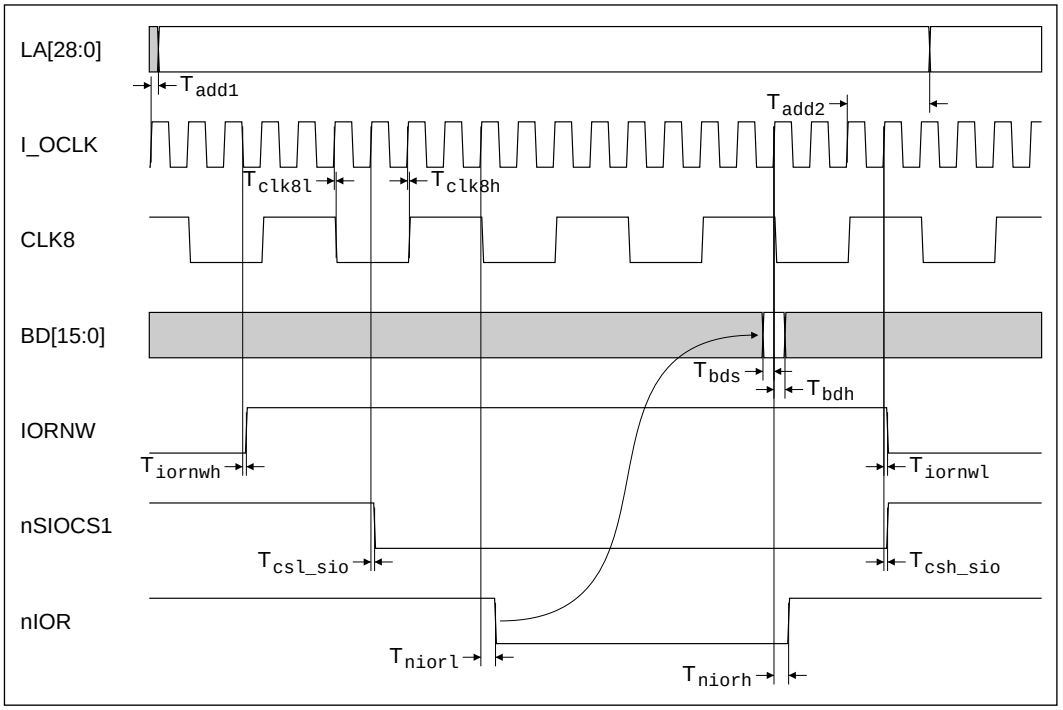


Figure 18-1: ‘Fast’ 8MHz Simple I/O read cycle timing

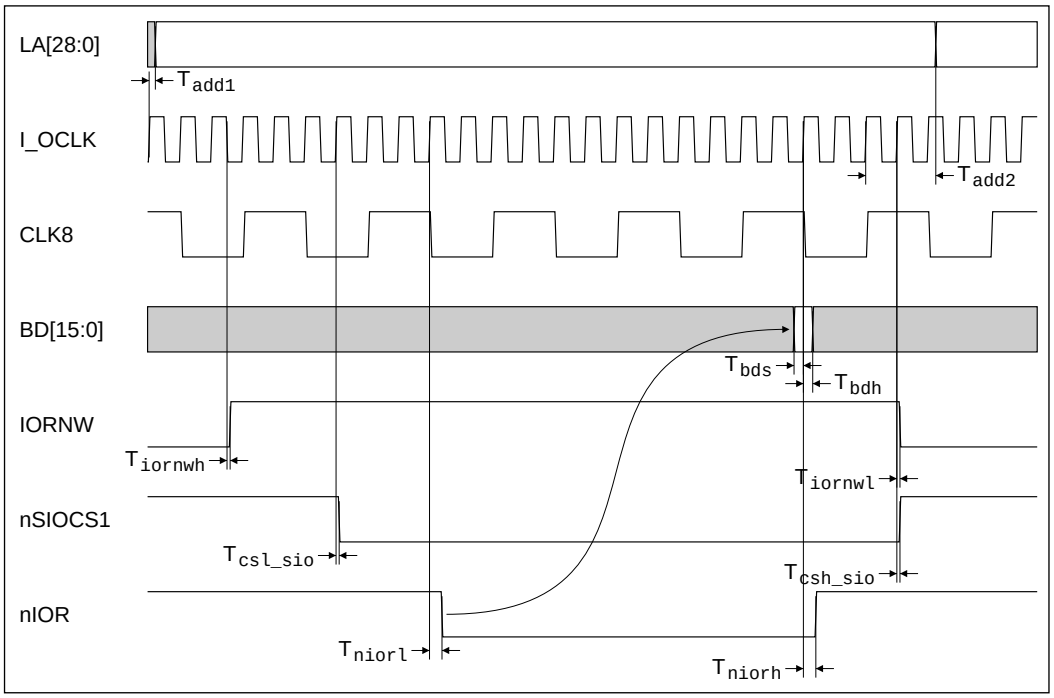


Figure 18-2: 'Medium' 8MHz Simple I/O read cycle timing

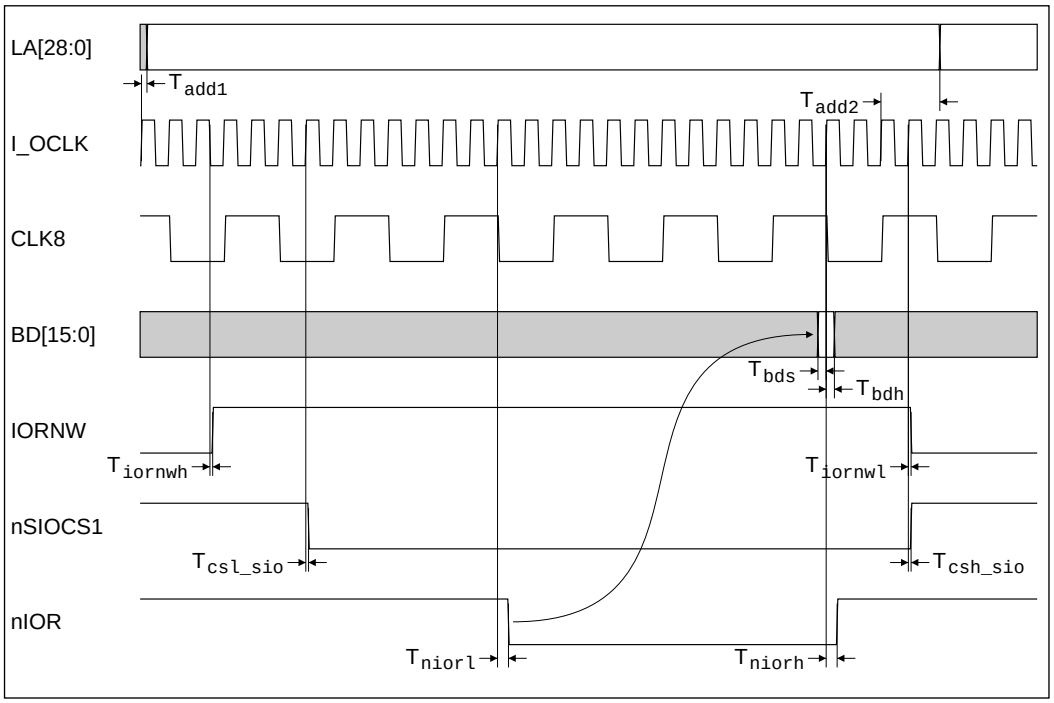


Figure 18-3: 'Slow' 8MHz Simple I/O read cycle timing

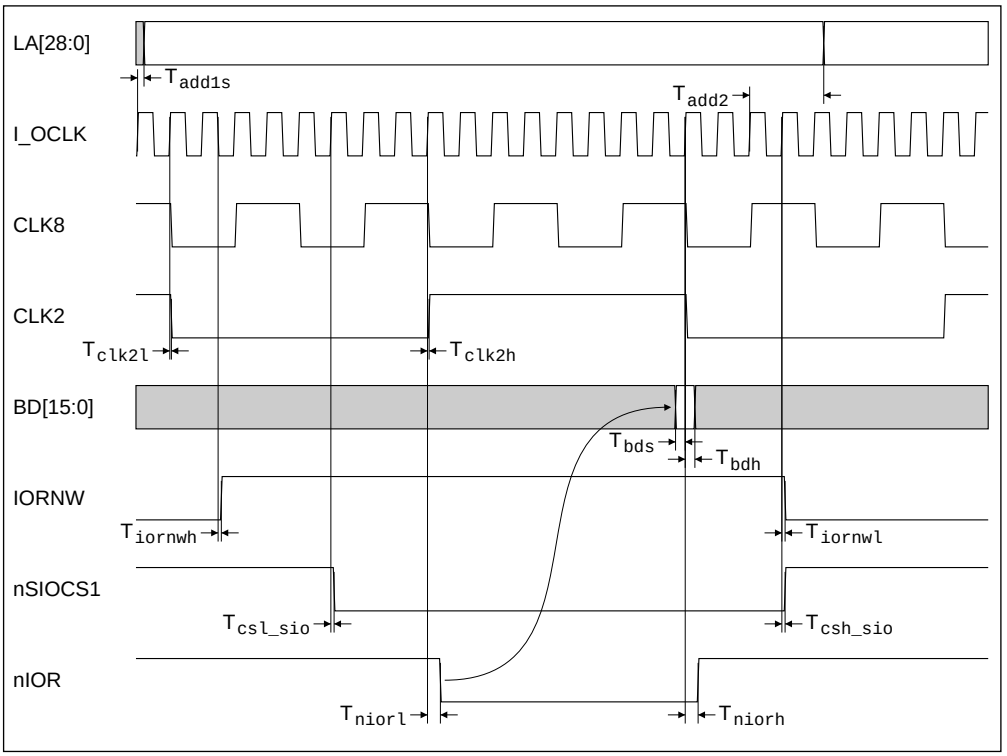


Figure 18-4: 'Sync' 8MHz I/O read cycle timing

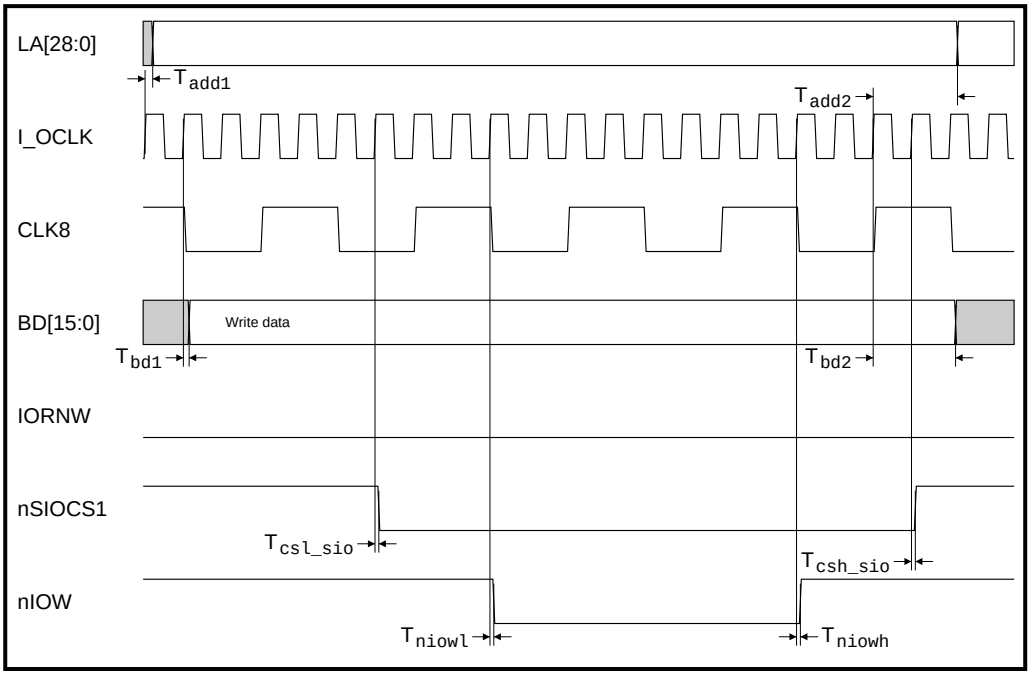


Figure 18-5: 'Fast' 8MHz Simple I/O write cycle timing

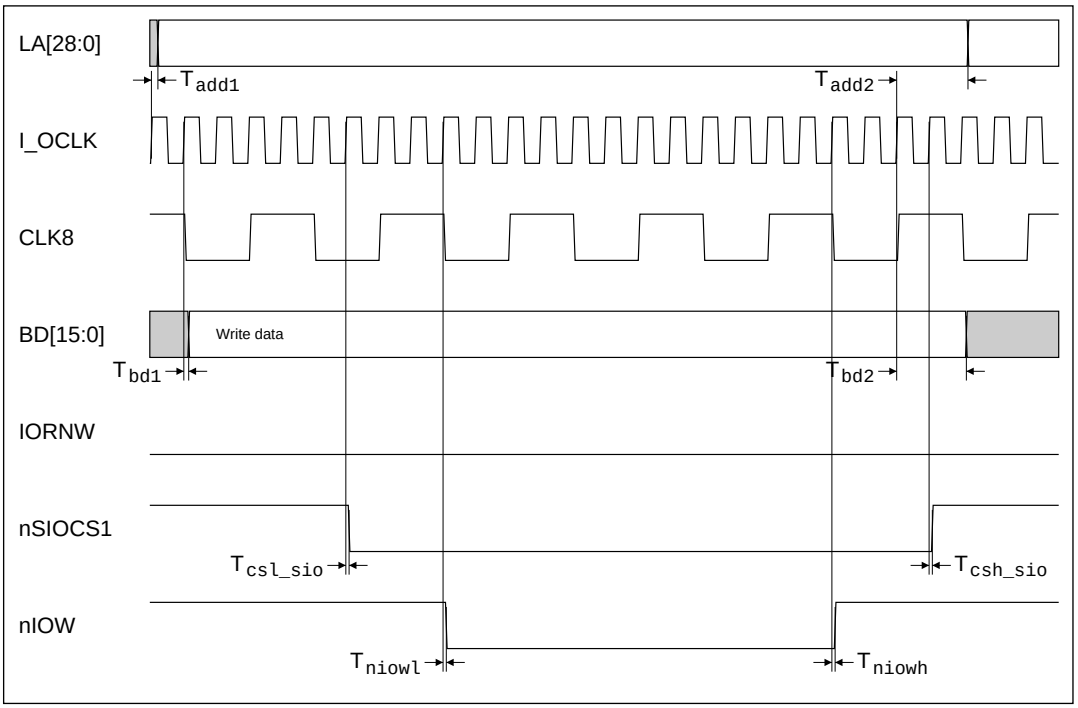


Figure 18-6: 'Medium' 8MHz Simple I/O write cycle timing

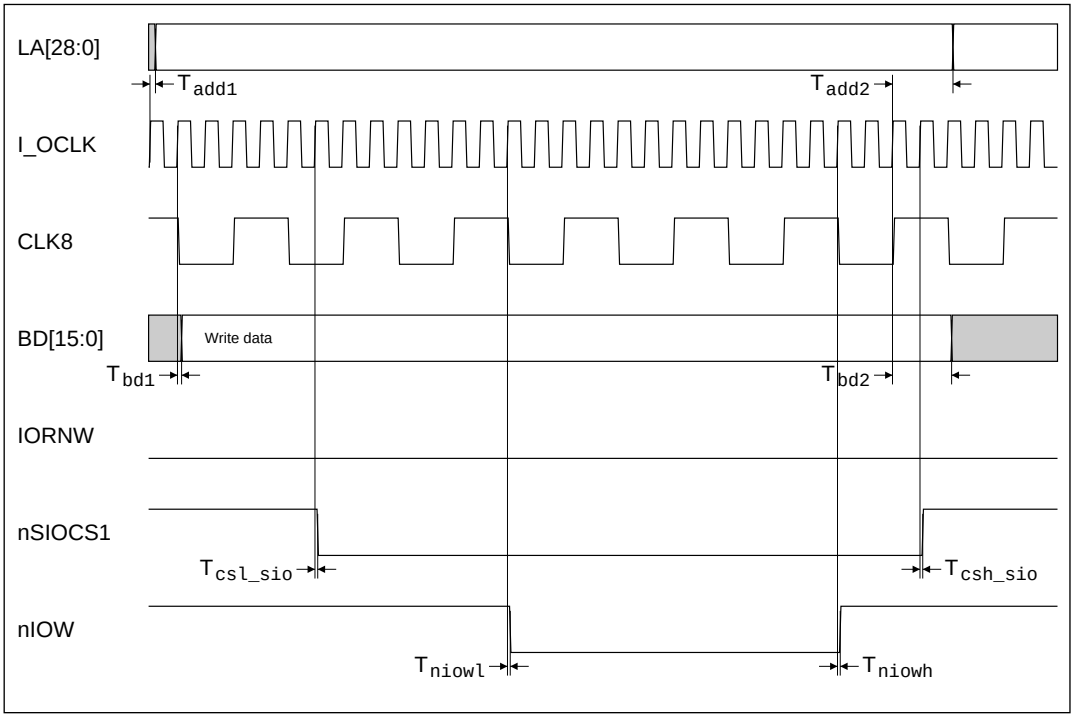


Figure 18-7: 'Slow' 8MHz Simple I/O write cycle timing

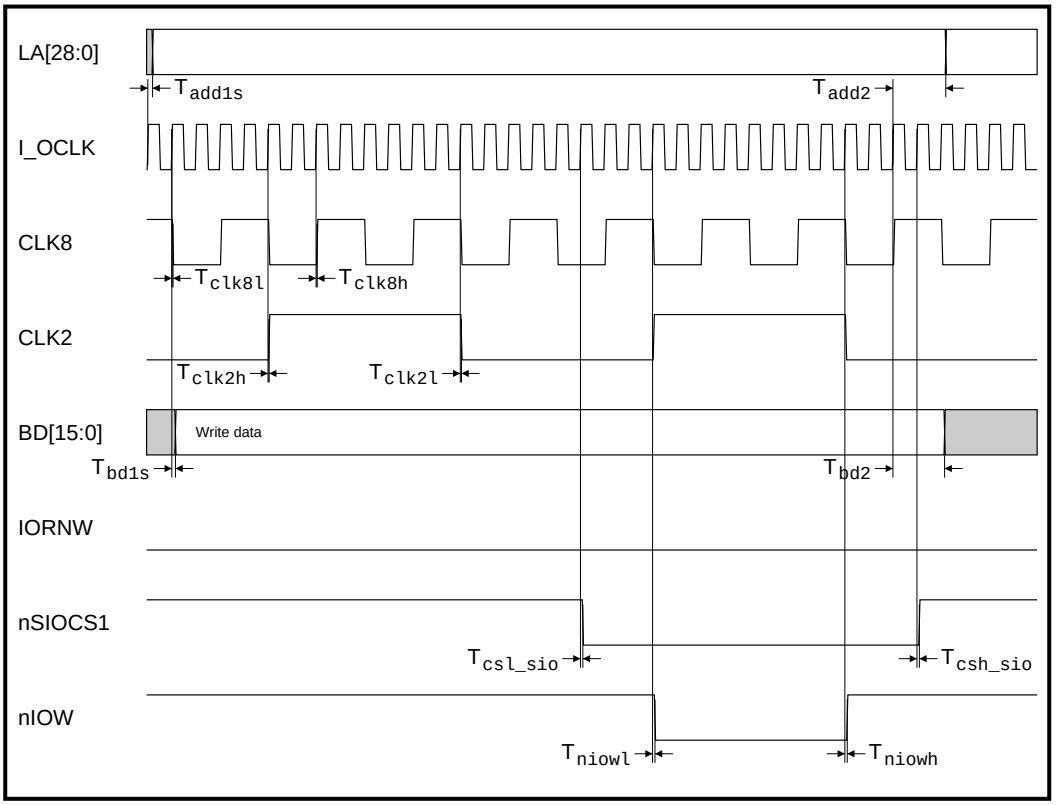


Figure 18-8: 'Sync' 8MHz Simple I/O write cycle timing

Symbol	Parameters	Min	Max	Units	Notes
Tclk8l	**I_OCLK** rising to **CLK8** falling		13	ns	
Tclk8h	**I_OCLK** rising to **CLK8** rising		12	ns	
Tclk2l	**I_OCLK** rising to **CLK2** falling		16	ns	
Tclk2h	**I_OCLK** rising to **CLK2** rising		16	ns	
Tcsl_sio	**I_OCLK** rising to **nSIOCS1/nSIOCS2** falling		16	ns	
Tcsh_sio	**I_OCLK** rising to **nSIOCS1/nSIOCS2** rising		16	ns	
Tbd1	**I_OCLK** rising to **BD** write data valid	0	102	ns	1
Tbd1s	**I_OCLK** rising to **BD** write data valid (**SYNC** cycles)	0	476	ns	2
Tbd2	**I_OCLK** rising to **BD** write data valid	133	152	ns	3,7
Tbd2	**I_OCLK** rising to **BD** write data valid	149	168	ns	3,8
Tbdh	DATA hold from **I_OCLK** rising	10		ns	
Tbds	DATA setup to **I_OCLK** rising	0		ns	
Tiornwh	**I_OCLK** falling to **IORNW** rising		13	ns	
Tiornwl	**I_OCLK** rising to **IORNW** falling		16	ns	
Tniorl	**I_OCLK** rising to **nIOR** falling		16	ns	
Tniorh	**I_OCLK** rising to **nIOR** rising		16	ns	
Tniowl	**I_OCLK** rising to **nIOW** falling		17	ns	
Tniowh	**I_OCLK** rising to **nIOW** rising		16	ns	
Tadd1	**LA[]** changing after **I_OCLK** rising before start	0	143	ns	4
Tadd1s	**LA[]** changing after **I_OCLK** rising before start (**SYNC** cycles)	0	518	ns	5
Tadd2	**LA[ ]** changing after **I_OCLK** rising after end	74	89	ns	6,7
Tadd2	**LA[ ]** changing after **I_OCLK** rising after end	90	105	ns	6,8

Table 18-3: Simple 8MHz I/O timing

Note 1: Synchronization penalty is between 0 and 3 **I_OCLK** cycles
Note 2: Synchronization penalty is between 0 and 15 **I_OCLK** cycles
Note 3: Delay includes 4 **MEMCLK** cycles
Note 4: Synchronization penalty is between 1 and 4 **I_OCLK** cycles
Note 5: Synchronization penalty is between 1 and 16 **I_OCLK** cycles
Note 6: Delay includes 2 **MEMCLK** cycles

Note 7: Timings refer to the case where ASTCR bit=0.
See *Appendix C: Using ASTCR at High MEMCLK Frequencies*.

Note 8: Timings refer to the case where ASTCR bit = 1.

Note: *The output delays above only include the intrinsic delay of the output pad driver. See section 22.5 De-rating* on page 22-6 *to calculate the final delay dependent upon the expected output load.*

18.5 Module I/O

The Module I/O type of access is 16-bit only and its speed is controlled by a handshake mechanism with the external hardware. The signals **nIORQ** (output) and **nIOGT** (input) are used for this handshaking. When writing, the upper half-word of the ARM7500FE data bus is written out on the I/O bus. When reading, the I/O bus data is read back onto the lower half-word of the ARM7500FE data bus. The module type of I/O will be initiated for addresses in the ranges 0x03000000 to 0x0300FFFF and 0x03030000 to 0x0303FFFF.

During these accesses, the signal **nMSCS** is asserted but read and write strobes are not used, although the **IORNW** signal is active. **READY** does not affect this type of access.

The **nBLI** is driven by the external hardware to indicate when the read or write data should be latched from the BD I/O bus.

The I/O cycle will terminate when both **nIORQ** and **nIOGT** are LOW at the rising edge of REF8M.

The following timing diagrams show the signal relationship for the **nIORQ**/**nIOGT** module I/O type of access.

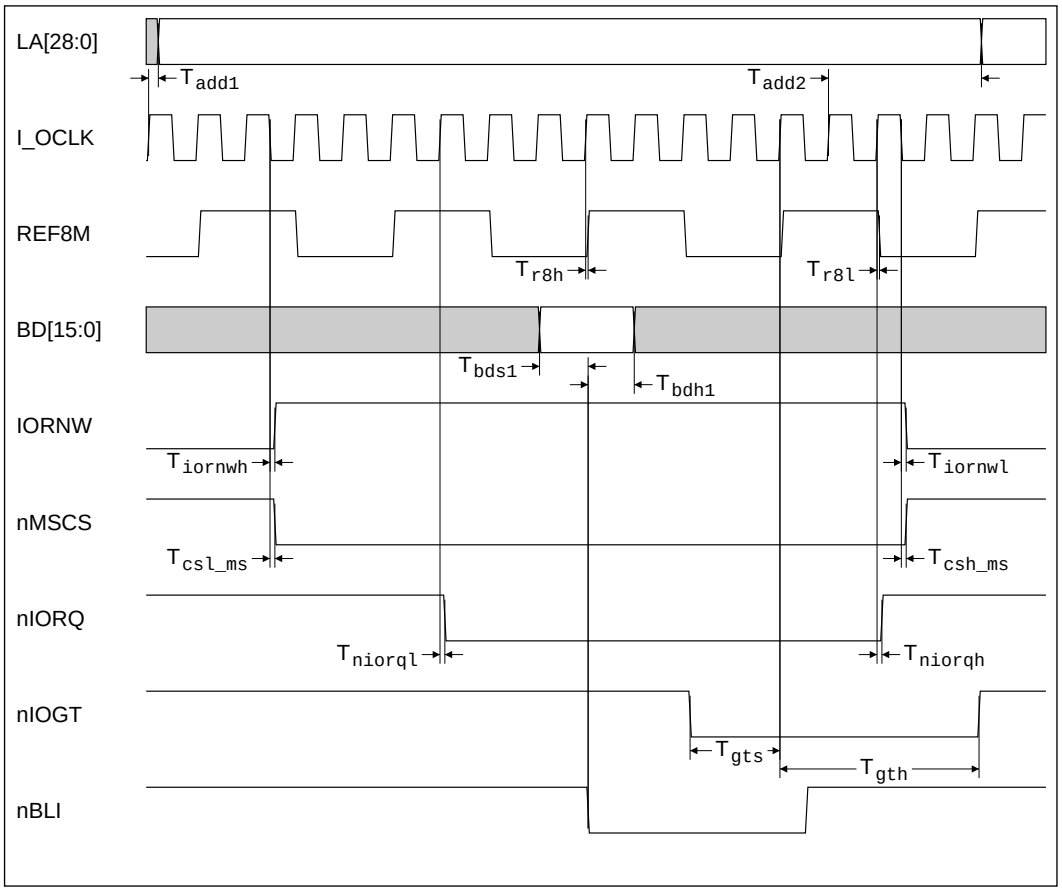


Figure 18-9: 8 MHz Module read I/O cycle

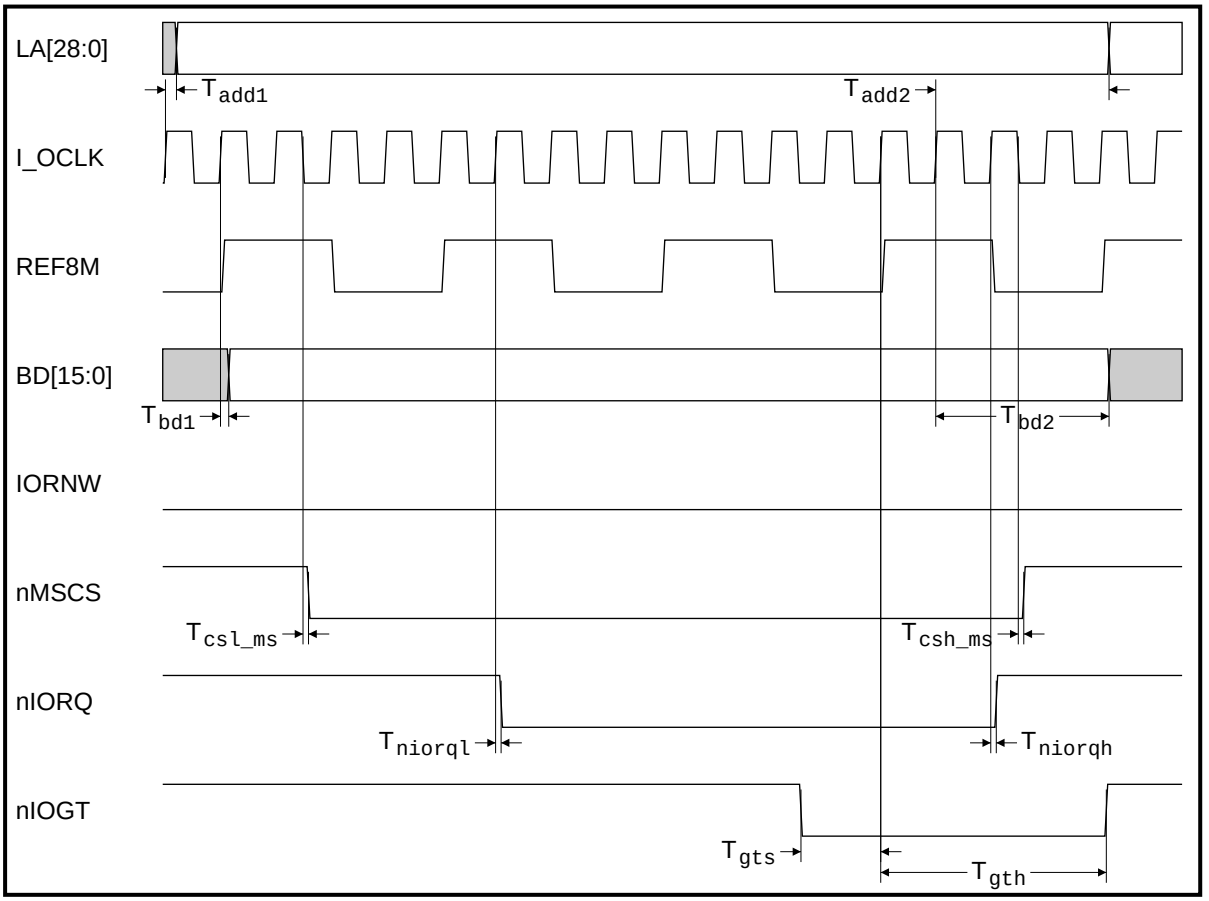


Figure 18-10: 8 MHz module write I/O cycle

Symbol	Parameters	Min	Max	Units	Notes
Tbds1	Data setup up to **nBLI** falling	0		ns	
Tbdh1	Data hold from **nBLI** falling	2		ns	
Tcsl_ms	**I_OCLK** falling to **nMSCS** falling		15	ns	
Tcsh_ms	**I_OCLK** falling to **nMSCS** rising		18	ns	
Tiornwh	**I_OCLK** falling to **IORNW** rising		13	ns	
Tiornwl	**I_OCLK** falling to **IORNW** falling		14	ns	
Tbd1	**I_OCLK** rising to **BD** write data valid	0	102	ns	1
Tbd2	**I_OCLK** rising to **BD** write data valid	133	150	ns	2,5
Tbd2	**I_OCLK** rising to **BD** write data valid	164	181	ns	2,6
Tniorql	**I_OCLK** rising to **nIORQ** falling		15	ns	
Tniorqh	**I_OCLK** rising to **nIORQ** rising		15	ns	
Tr8ml	**I_OCLK** rising to **REF8M** falling		13	ns	
Tr8mh	**I_OCLK** rising to **REF8M** rising		12	ns	
Tgts	setup of **nIOGT** to **I_OCLK** rising	0		ns	
Tgth	hold of **nIOGT** from **I_OCLK** rising	5		ns	
Tadd1	**LA[ ]** changing after I_OCLK rising before start	0	143	ns	3
Tadd2	**LA[ ]** changing after **I_OCLK** rising at end	74	89	ns	4,5
Tadd2	**LA[ ]** changing after **I_OCLK** rising at end	105	120	ns	4,6

Table 18-4: 8 MHz Module read and write I/O cycles

In *Table 18-4: 8 MHz Module read and write I/O cycles* on page 18-14:

Note 1: Synchronization penalty is between 0 and 3 **I_OCLK** cycles

Note 2: Delay includes 4 **MEMCLK** cycles

Note 3: Synchronization penalty is between 1 and 4 **I_OCLK** cycles

Note 4: Delay includes 2 **MEMCLK** cycles

Note 5: Timings refer to the case where ASTCR bit=0.
See *Appendix C: Using ASTCR at High MEMCLK Frequencies*.

Note 6: Timings refer to the case where ASTCR bit = 1.

Note: *The output delays above only include the intrinsic delay of the output pad driver. See section 22.5 De-rating* on page 22-6 *to calculate the final delay dependent upon the expected output load.*

18.6 PC Bus-style I/O

This type of I/O is designed to function in conjunction with a standard PC Combo chip, and cycles are generated from a 16MHz clock.

The PC bus-style I/O type of access routes the lower halfword of the ARM7500FE bus through the device providing a direct 16 bit interface. Additionally, signals are generated to support the addition of external latches/drivers to extend the I/O data by 16 bits. The upper half-word of the ARM7500FE data bus is routed through these external devices if present. This type of I/O access is used for the address space from 03010000 to 0302CFFF (five sections), and in the larger extended address space from 0x08000000 to 0x0FFFFFFF (eight sections). There are 4 fixed cycle types based on the 16MHz clock, although the larger extended address area only supports two of these cycle types. Any access may be held up by external circuitry removing the **READY** signal before the end of the cycle.

The signals used to control the external buffers and latches required to implement 32-bit wide I/O are:

- **nWBE**
- **nRBE**
- **nBLO**

The timing diagrams in this section (*Figure 18-12: 16 MHz Type D read I/O cycle* and *Figure 18-11: 16 MHz Type D write I/O cycle*) show the timing of these signals relative to the external data bus.

For full details of the external circuitry and connections required to implement a 32-bit wide I/O system using the ARM7500FE, refer to *Appendix D: Expanding PC-Style I/O to 32 Bit*.

Two additional inputs are provided to allow external circuitry to route a full 32-bit data word through the 16-bit I/O bus using multiplexing:

- **nXIPLATCH**
- **nXIPMUX16**

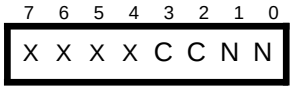
This would allow, for example, the execution of ARM code from a 16-bit-wide PCMCIA card with a suitable external controller. The **nXIPMUX16** signal directly controls an internal multiplexer which maps either the upper or lower 16 bits of the internal data bus through to the 16 bit wide I/O bus, for writes to an I/O peripheral.

When **nXIPMUX16** is LOW, the upper 16 bits of the data bus are passed to **BD[15:0]**, and when **nXIPMUX16** is HIGH, the lower 16 bits of the data bus are passed to **BD[15:0]**.

For reads from an I/O peripheral, the falling edge of the **nXIPLATCH** signal causes the first 16 bits provided on the **BD[15:0]** bus to be latched as the upper halfword for the main internal data bus, after which the lower 16 bits can be output from the peripheral and the I/O cycle can be allowed to complete normally. If **nXIPLATCH** has been driven low, the upper halfword of data is driven to the ARM processor internally and not from the external transceivers if present.

Figure 18-19: 16 MHz Type B read I/O cycle with PCMCIA and *Figure 18-20: 16 MHz Type B write I/O cycle with PCMCIA* show the relevant timing details. Depending on the cycle timing, it will usually be necessary for the external controller to use the **READY** signal to stretch the I/O access to give sufficient time for both half words to be read or written as appropriate. If an I/O access is to be stretched, the **READY** signal must be set LOW before the end of the cycle as shown in the timing diagrams. This will cause the **nIOR** or **nIOW** strobe and the chip select to be held LOW until **READY** is set back to HIGH again, when the I/O cycle will complete as normal. **READY** is sampled on the rising edge of the first 16MHz cycle before the I/O cycle is due to complete.

The four address areas for 16MHz I/O within the main I/O address space can support any of the four available cycle types A to D. The IOTCR register can be programmed (at address 0x032000C4) to determine which type of cycle will be used for each group of addresses. The addresses are grouped such that the **nCCS** and pseudo DMA address spaces form one group, and the **nPCCS1** and **nPCCS2** address area forms another group.



C	**nCCS** + pseudo DMA access speed
N	**nPCCS1** and **nPCCS2** area access speed
Write	bits[7:6] unused
	bits[5:4] unused
	bits[3:2]
	00 Type A (slowest)
	01 Type B
	10 Type C
	11 Type D (fastest).
	bits[1:0]
	00 Type A (slowest)
	01 Type B
	10 Type C
	11 Type D (fastest).
Read	read back above values
Reset	set to zero (slowest)

The extended address space from address 0x08000000 onwards for 16MHz I/O accesses supports only cycle types A and C, and the ECTCR register should be programmed to specify which cycle type is required for each of the eight 16MB areas within the extended address space. The details of this register, at address 0x032000C8, are shown below:

E = expansion card area access speed

Write bit[7] (0F00 0000 -> 0FFF FFFF)

0 Type A

1 Type C

bit[0] (0800 0000 -> 08FF FFFF)

0 Type A

1 Type C

Read read back above values

Reset set to zero (slowest)

This type of I/O asserts a single chip select according to the area, except in Combo DACK + TC space, where both the **nCDACK** and **TC** outputs are asserted to signal to the PC Combo chip that the end of a pseudo DMA sequence has been reached. In the extended address space the **nEASCS** chip select is asserted.

The timing diagrams in the figures below show the four types of 16 MHz I/O cycle.

Note: *All diagrams assume divide by 1 mode for both **MEMCLK** and **I_OCLK**.*

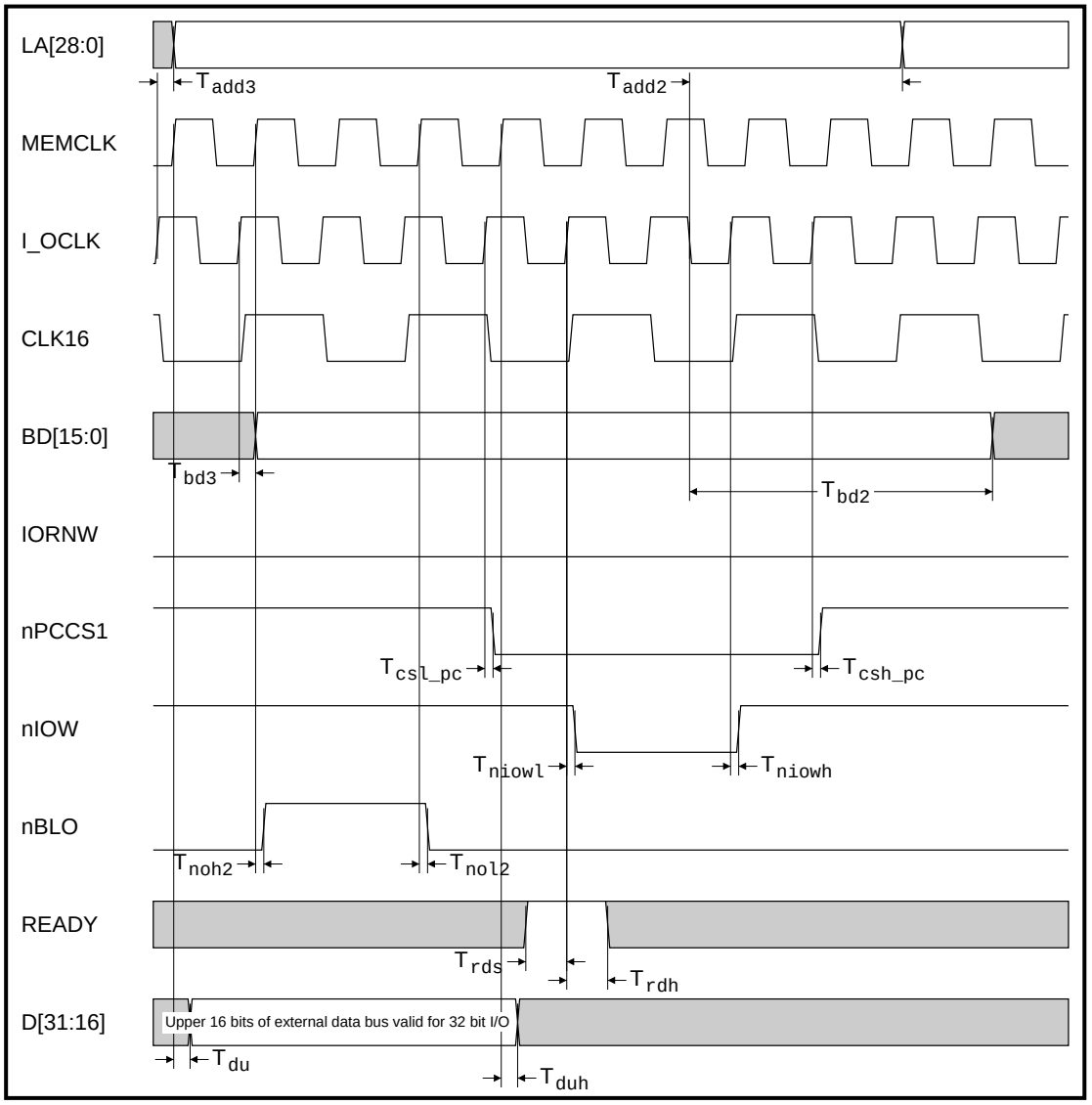


Figure 18-11: 16 MHz Type D write I/O cycle

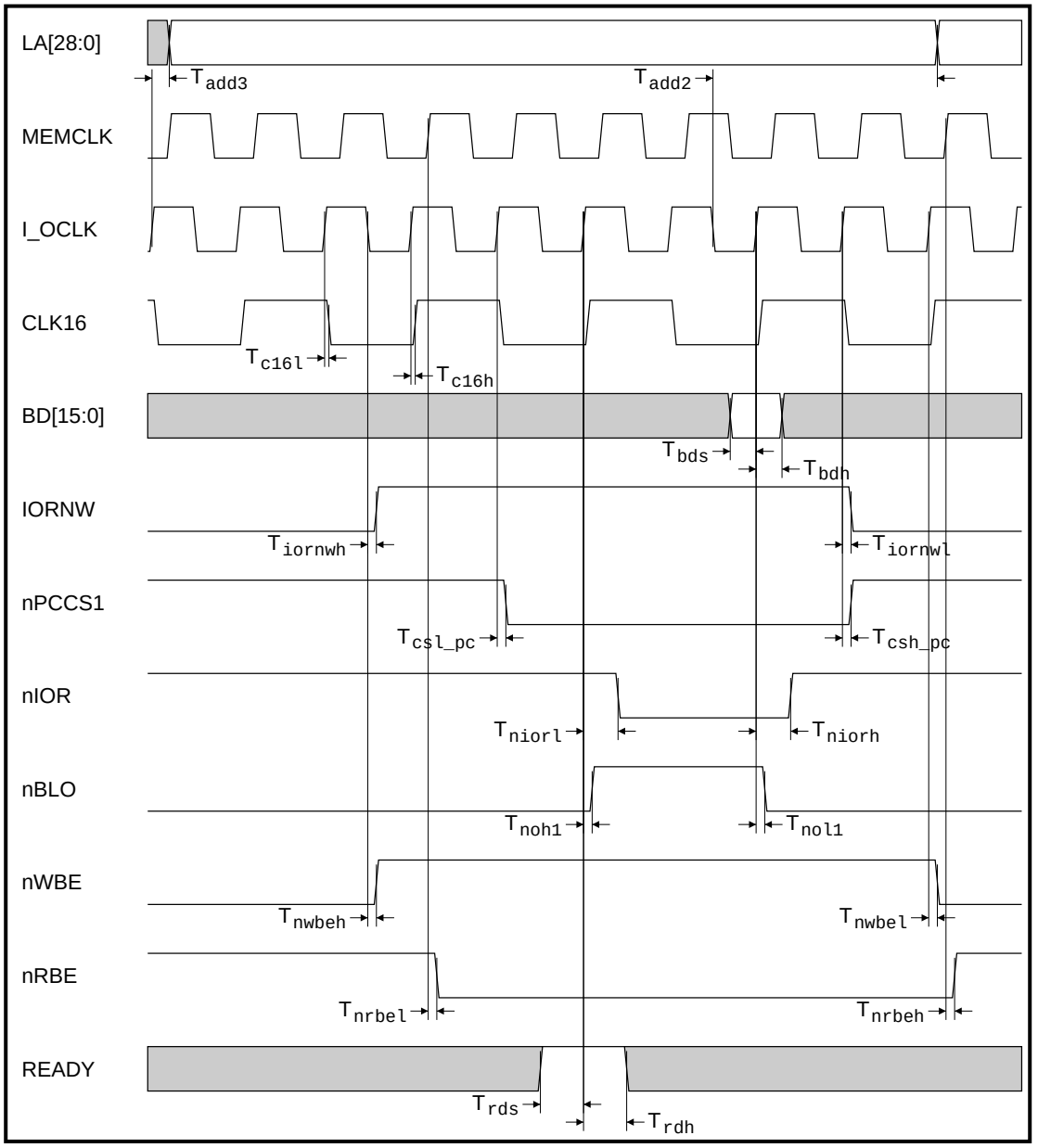


Figure 18-12: 16 MHz Type D read I/O cycle

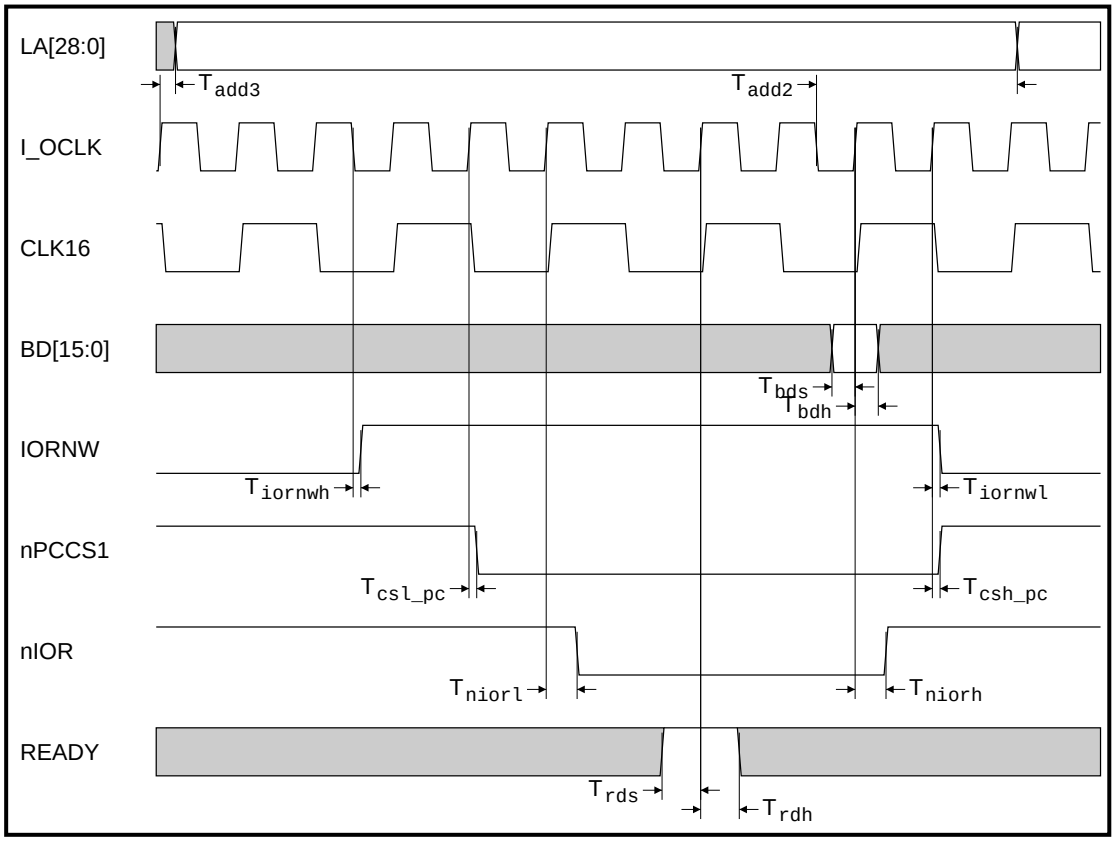


Figure 18-13: 16 MHz Type C read I/O cycle

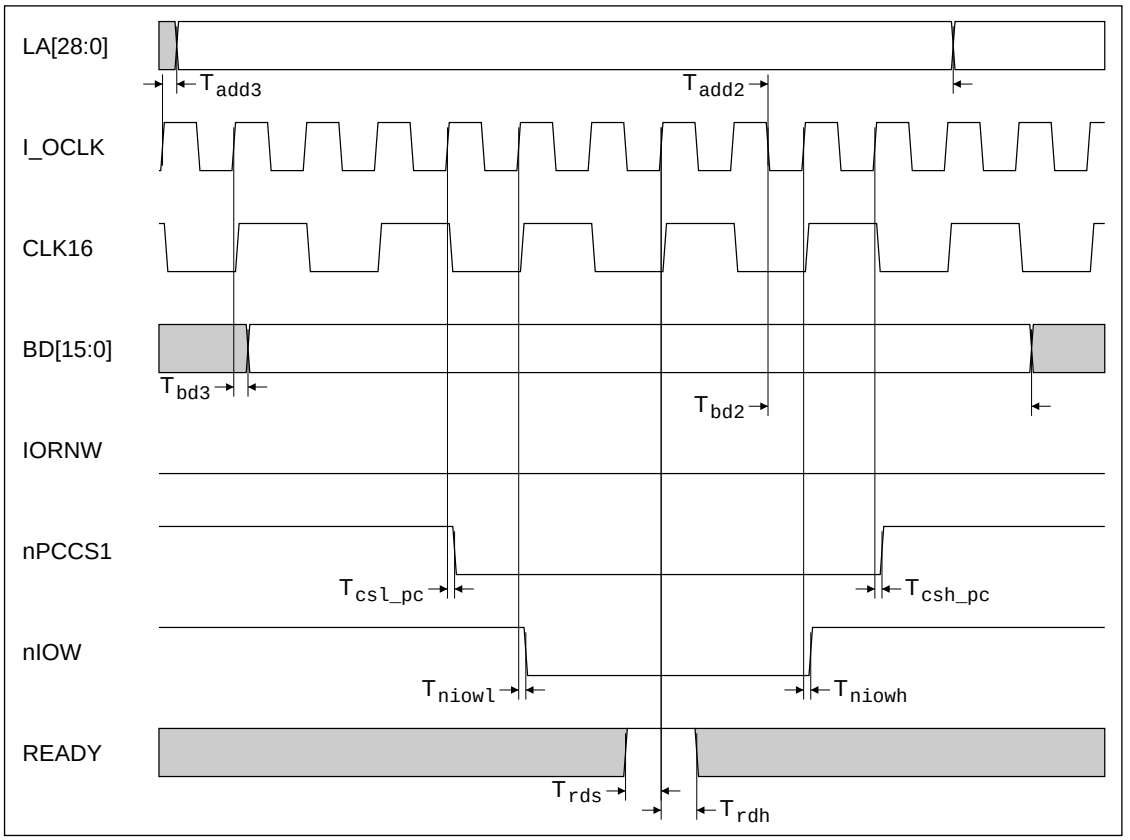


Figure 18-14: 16 MHz Type C write I/O cycle

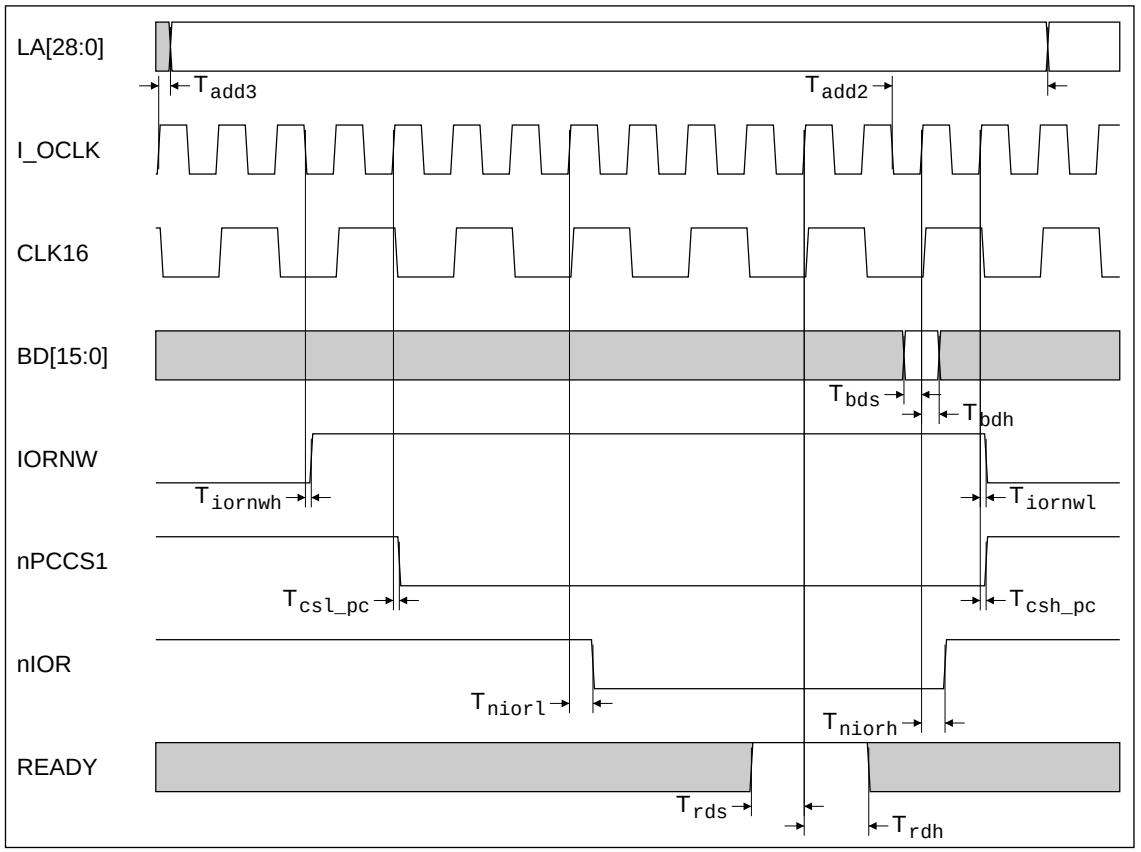


Figure 18-15: 16 MHz Type B read I/O cycle

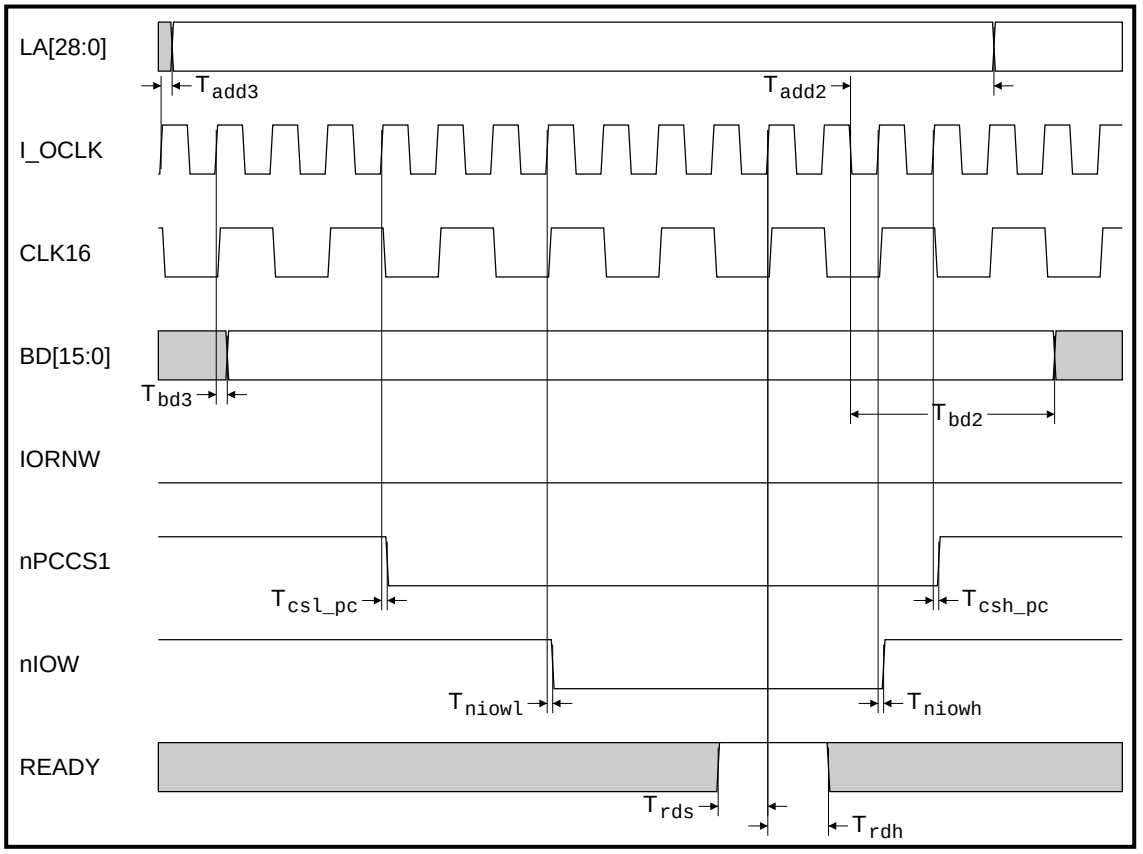


Figure 18-16: 16 MHz Type B write I/O cycle

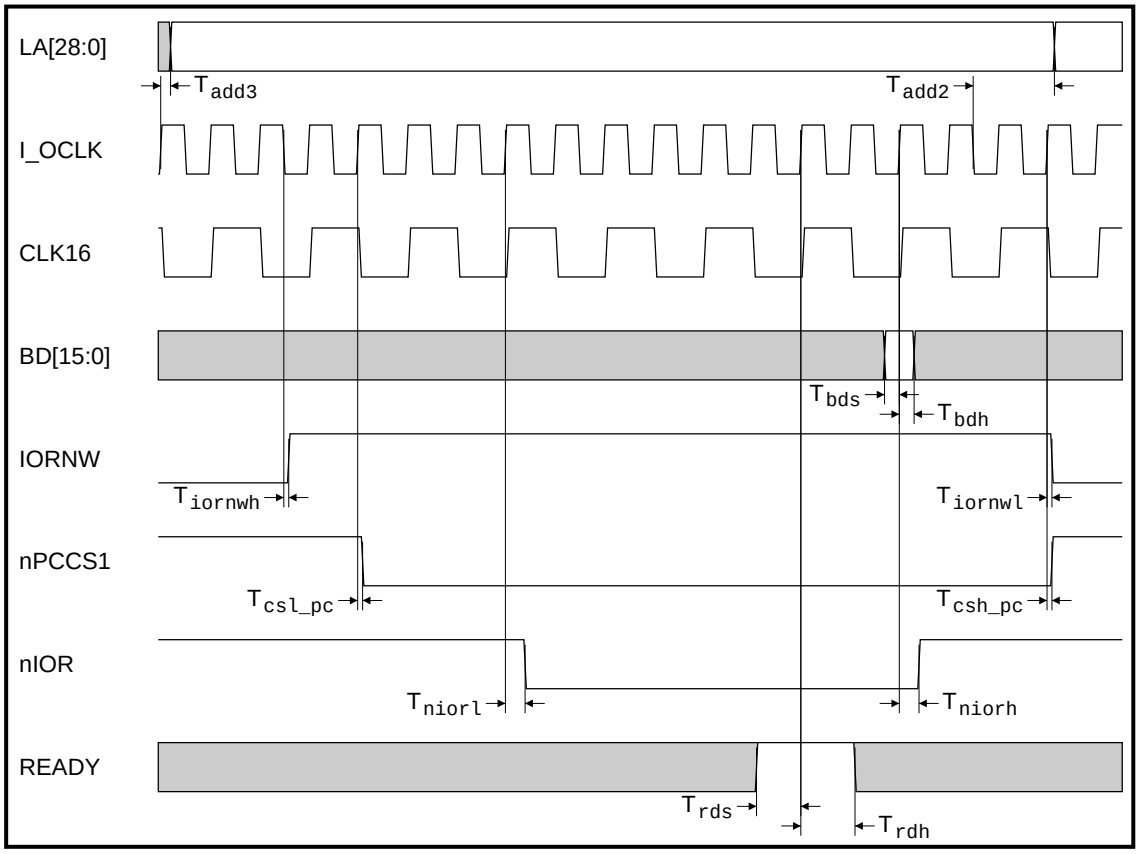


Figure 18-17: 16 MHz Type A read I/O cycle

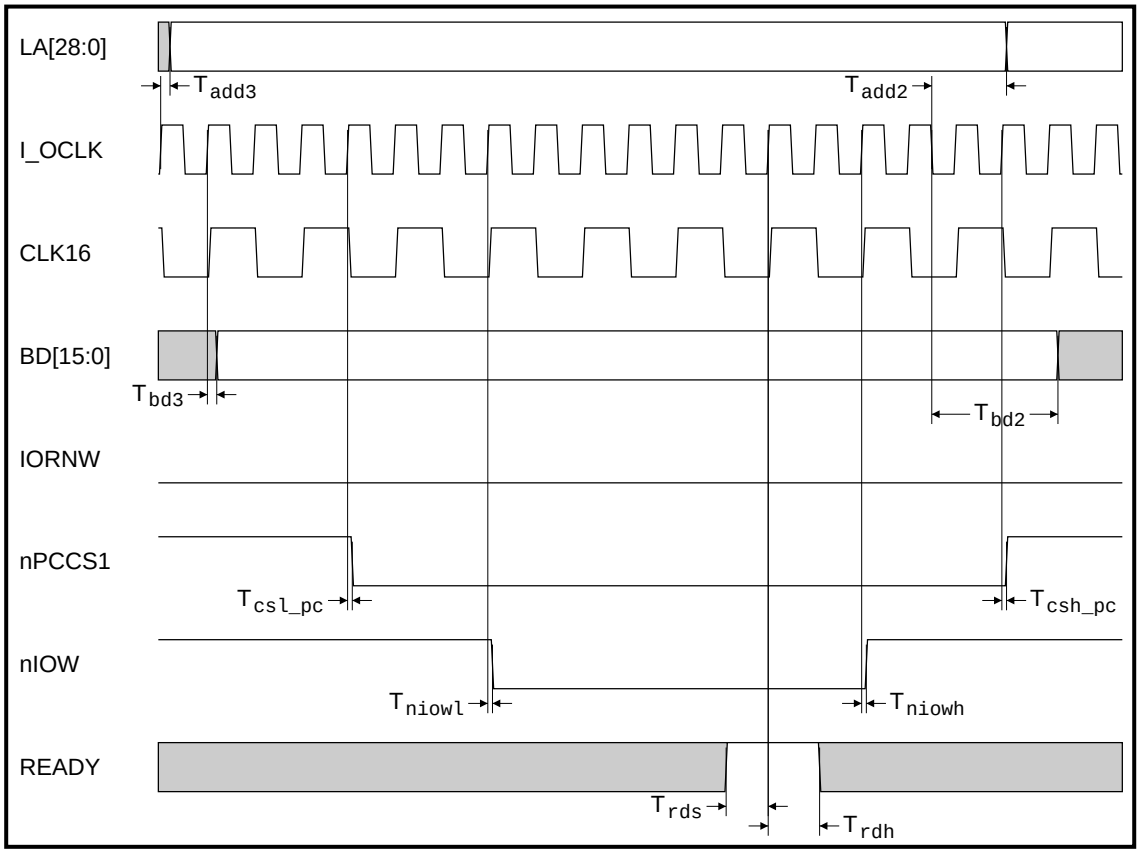


Figure 18-18: 16 MHz Type A write I/O cycle

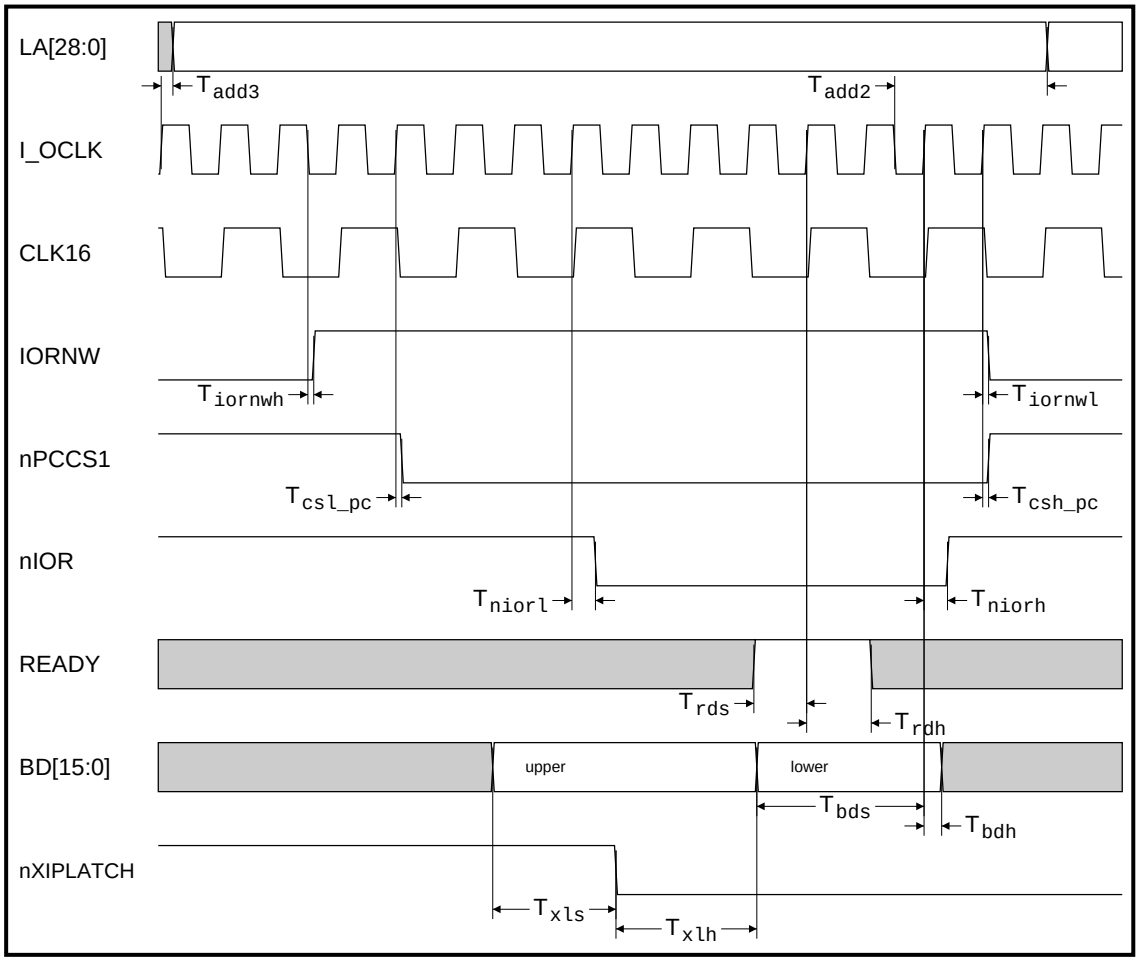


Figure 18-19: 16 MHz Type B read I/O cycle with PCMCIA

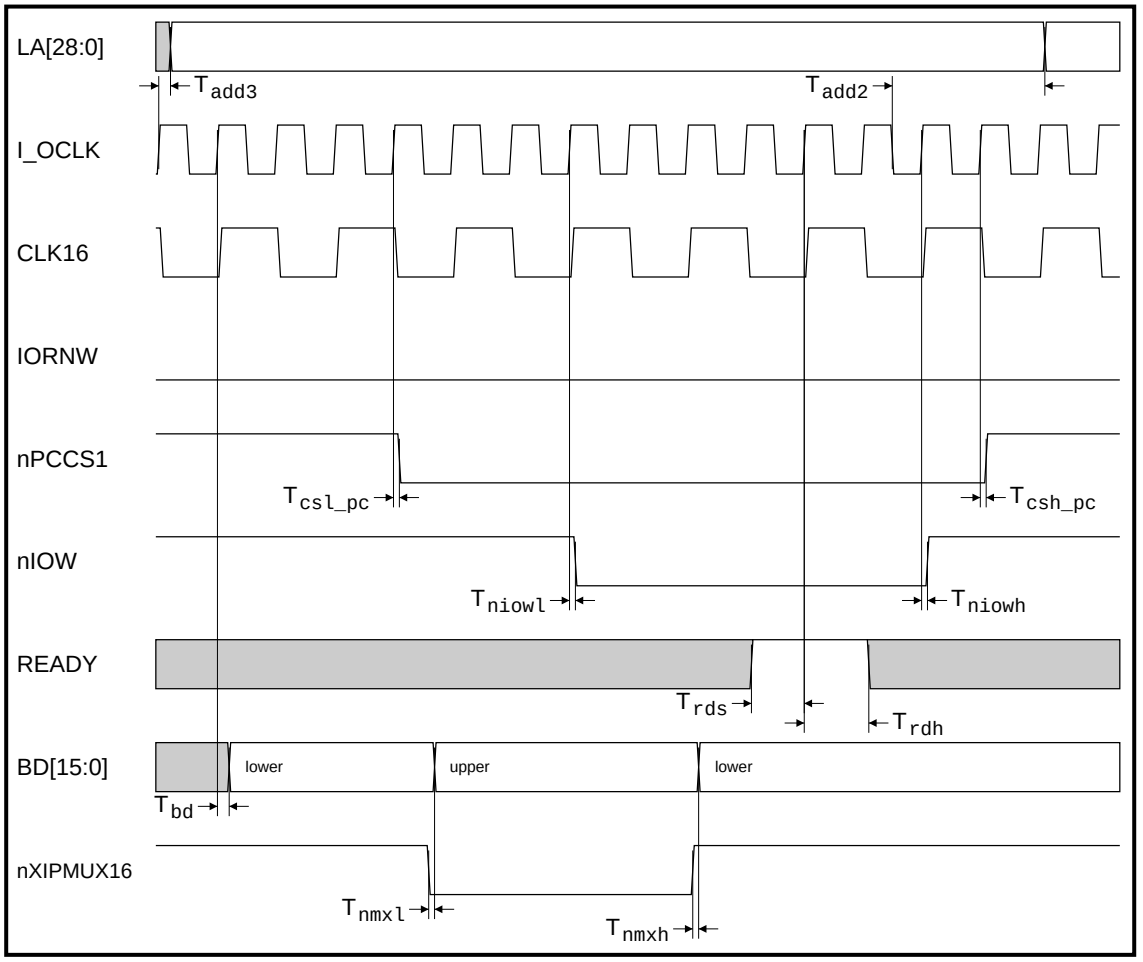


Figure 18-20: 16 MHz Type B write I/O cycle with PCMCIA

Symbol	Parameters	Min	Max	Units	Notes
Tnmxl	**nXIPMUX16** falling to upper data output on **BD[15:0]**		6	ns	
Tnmxh	**nXIPMUX16** rising to lower data output on **BD[15:0]**		5	ns	
Txls	DATA setup to **nXIPLATCH** falling	1		ns	
Txlh	DATA hold from **nXIPLATCH** falling	2		ns	
Tc16l	**I_OCLK** rising to **CLK16** falling		12	ns	
Tc16h	**I_OCLK** rising to **CLK16** rising		12	ns	
Tbdh	Data hold from **I_OCLK** rising	10		ns	
Tbds	Data setup to **I_OCLK** rising	0		ns	
Tiornwh	**I_OCLK** falling to **IONRW** rising		13	ns	
Tiornwl	**I_OCLK** rising to **IONRW** falling		16	ns	
Tcsl_pc	**I_OCLK** rising to PC I/O chip select falling		17	ns	1
Tcsh_pc	**I_OCLK** rising to PC I/O chip select rising		17	ns	1
Trds	**READY** setup to **I_OCLK** rising	0		ns	
Trdh	**READY** hold from **I_OCLK** rising	8		ns	
Tbd2	**I_OCLK** rising to **BD** write data valid	133	150	ns	2,6
Tbd2	**I_OCLK** rising to **BD** write data valid	164	181	ns	2,7
Tbd3	**I_OCLK** rising to **BD** write data valid	0	40	ns	3
Tniorl	**I_OCLK** rising to **nIOR** falling		16	ns	
Tniorh	**I_OCLK** rising to **nIOR** rising		16	ns	
Tnoh1	**I_OCLK** rising to **nBLO** rising, read		18	ns	
Tnol1	**I_OCLK** rising to **nBLO** falling, read		18	ns	
Tnoh2	**MEMCLK** rising to **nBLO** rising, write		18	ns	
Tnol2	**MEMCLK** rising to **nBLO** falling, write		16	ns	
Tnwbeh	**I_OCLK** falling to **nWBE** rising		17	ns	
Tnwbel	**I_OCLK** rising to **nWBE** falling		13	ns	
Trbel	**MEMCLK** rising to **nRBE** falling		16	ns	
Trbeh	**MEMCLK** rising to **nRBE** rising		16	ns	
Tniowl	**I_OCLK** rising to **nIOW** falling		17	ns	

Table 18-5: 16 MHz I/O cycles

Symbol	Parameters	Min	Max	Units	Notes
Tniowh	**I_OCLK** rising to **nIOW** rising		16	ns	
Tdu	**MEMCLK** rising to **D[31:16]** valid		35	ns	
Tadd3	**LA[]** changing after **I_OCLK** rising before start	0	82	ns	4
Tduh	**MEMCLK** rising to **D[31:16]** invalid	10		ns	
Tadd2	**LA[ ]** changing after **I_OCLK** rising at end	74	89	ns	5,6
Tadd2	**LA[ ]** changing after **I_OCLK** rising at end	105	120	ns	5,7

Table 18-5: 16 MHz I/O cycles (Continued)

In *Table 18-5: 16 MHz I/O cycles* on page 18-28:

Note 1: Timing is for all PC style I/O chip selects: **nCCS, nCDACK, nPCCS1, nPCCS2, nEASCS, TC**

Note 2: Delay includes 4 **MEMCLK** cycles

Note 3: Synchronization penalty is 0 or 1 **I_OCLK** cycles

Note 4: Synchronization penalty is 1 or 2 **I_OCLK** cycles

Note 5: Delay includes 2 **MEMCLK** cycles

Note 6: Timings refer to the case where ASTCR bit=0.
See *Appendix C: Using ASTCR at High MEMCLK Frequencies*

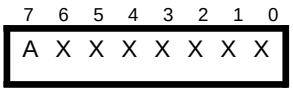
Note 6: Timings refer to the case where ASTCR bit=1.

Note: *The output delays above only include the intrinsic delay of the output pad driver. See section 22.5 De-rating* on page 22-6 *to calculate the final delay dependent upon the expected output load.*

18.7 DMA During I/O Cycles

DMA to the Video and Sound Macrocell can continue during I/O cycles. Write data from the ARM Processor is latched early, so that the data bus can be used freely for DMA data. Thus, only the start of an I/O cycle needs to be added to any DMA latency calculations.

18.8 Clock Synchronization Conditions



In a system using a **MEMCLK** frequency greater than **I_OCLK**, it may be necessary to insert an extra I/O clock cycle to allow sufficient address hold time before the chip select is taken away. The problem arises because the chip select is generated from the fixed frequency I/O world clock, whereas the address changes according to the memory system clock. When a faster **MEMCLK** is used, it is possible for the synchronization to the memory clock to occur rapidly at the end of the cycle, and thus for the I/O address to change before the chip select has been removed. This may be a problem for some peripherals.

To avoid this, there is a register bit in the ASTCR register, at address 0x032000CC, which is normally set to zero, but can be programmed to one to add an extra I/O clock period to ensure that the address will not change before the chip select has been de-asserted.

A asynchronous timing control
- 0 minimal delay
- 1 wait states to ensure address hold time

See *Appendix C: Using ASTCR at High MEMCLK Frequencies*.

18.9 Keyboard/mouse Interface

The keyboard and mouse interfaces are identical, differing only in the names of the external pins. The interfaces are designed to communicate with a standard PS/2 keyboard or mouse, via a 2 pin serial link.

The keyboard interface uses the pins KBDATA, KBCLK, and the mouse interface uses the pins MSDATA and MSCLK, all of which are open drain.

There is an 8-bit control register for each interface, which provides direct access to the CLK and DATA outputs, an enable bit to enable the interface, and five status flags. The KBDCR is programmed at address 0x03200008, and the MSECR (mouse control register) at address 0x032000AC.

T transmit status

R receive status

E enable

P received parity

D data pin status

C clock pin status

Write bits[7:4,2] ignored

bit[3] enable
- 0 state machine cleared
- 1 state machine enabled

bit[1] force **KBDATA**/**MSDATA** pin LOW
- 0 don't force LOW
- 1 force LOW

bit[0] force **KBCLK**/**MSCLK** pin LOW
- 0 don't force LOW
- 1 force LOW

Read bit[7] TXE, shift register empty
- 0 not ready
- 1 enabled and ready to transmit

bit[6] TXB, transmitter busy
- 0 not busy
- 1 currently sending data

bit[5] RXF, receive shift register full
- 0 not full
- 1 ready to read

bit[4] RXB, receiver busy
- 0 not busy
- 1 currently receiving data

bit[3] ENA, state machine enable
- 0 disabled
- 1 enabled

bit[2] RXP, receive parity bit, odd parity bit for last received data

bit[1] **KBDATA**/**MSDATA** pin value after synchronization

bit[0] **KBCLK**/**MSCLK** pin value after synchronization

There is also a data register (KBDAT) which is used both to write bytes to be transmitted across the serial link and to read bytes received. The KBDAT register is programmed at address 0x03200004, and the MSEDAT (Mouse data register) is programmed at address 0x032000A8.

The interfaces generate two interrupts each, one to indicate that the transmit buffer is empty and thus that another byte can be transmitted, and one to indicate that a byte has been received by the interface. These interrupt bits are processed by the IRQB register set (for Keyboard) and the IRQD register set (for Mouse).

The keyboard interface is held in reset until the enable bit in the control register is set. The interface can be controlled on the basis of the interrupts generated, or by polling the status flags in the control register. The Tx interrupt is generated when the transmit buffer has been emptied and the interface is ready to be programmed with another character for transmission. The Rx interrupt is set when a complete character has been received in the receive buffer, and the byte is ready to be read from the register. The received data parity bit, RXP, is available in the control register at bit 2. Odd parity is used. The keyboard and mouse interface state machines are clocked by the 8MHz I/O system clock.

The **KCLK**/**MSCLK** signal is always driven by the keyboard/mouse, unless ARM7500FE wishes to prevent the peripheral from transmitting (because it is about to transmit some data itself). When data is received from the peripheral, the **KDATA**/**MSDATA** line is pulled low as a start bit. Each data bit is set up to the falling edge of the clock. Eight data bits are transmitted from the keyboard/mouse, followed by a parity bit (odd parity) and a HIGH stop bit. The diagram below shows the protocol of this transfer.

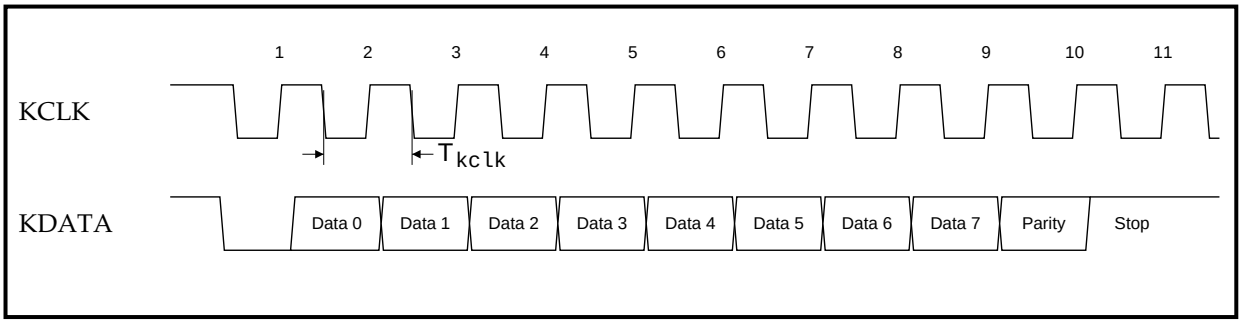


Figure 18-21: ARM7500FE Keyboard/mouse controller receive protocol

When ARM7500FE transmits a byte to the peripheral, the **KCLK**/**MSCLK** line is pulled LOW, then allowed to float and the **KDATA**/**MSDATA** line is pulled LOW, as a request to send. The keyboard/mouse then drives the clock, causing ARM7500FE to put eight bits of serial data out onto the **KDATA**/**MSDATA** line. A parity bit is driven out, followed by a stop bit, and the stop bit may be acknowledged by the peripheral (the ARM7500FE does not check on the acknowledge). The timing requirements of the interface are shown in *Figure 18-22: Keyboard/mouse interface timing*:

.

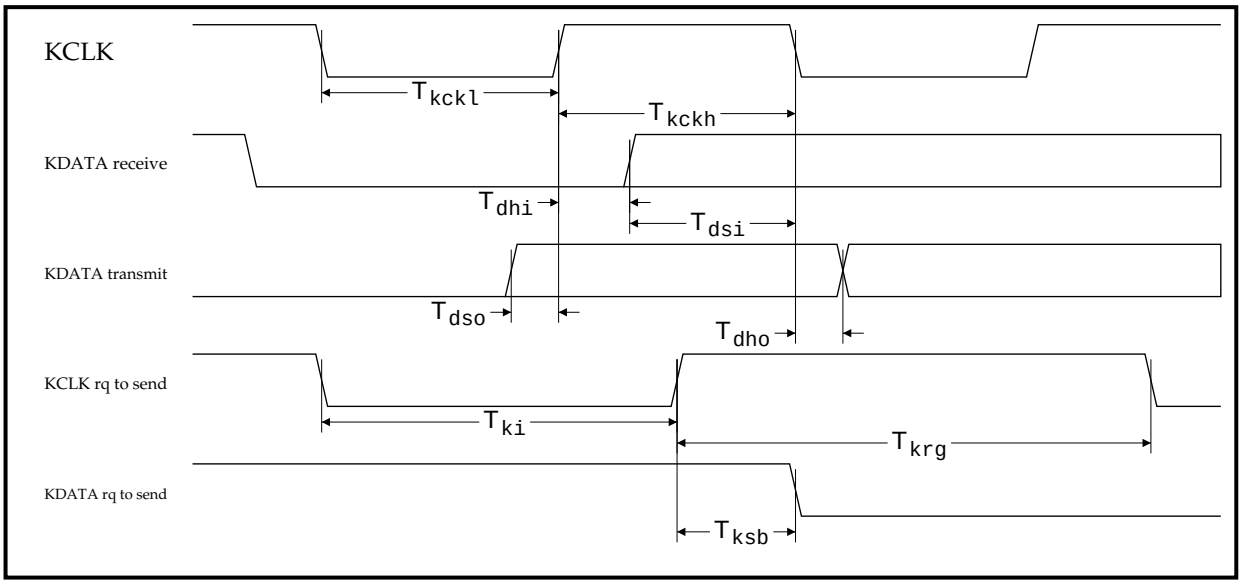


Figure 18-22: Keyboard/mouse interface timing

Symbol	Parameters	Min	Typ	Max	Units	Notes
Tkclk	keyboard clock period	1		100	µs	
Tkckl	keyboard clock low time	0.5		50	µs	
Tkckh	keyboard clock high time	0.5		50	µs	
Tdhi	hold on DATA from CLK rising for Receive	1		Tkckh - 1µs	µs	
Tdsi	setup on DATA to CLK falling for Receive	1		Tkckh - 1µs		
Tdso	setup on DATA to CLK rising for Transmit	Tkckl - 1µs		Tkckl		
Tdho	hold on DATA from CLK falling for Transmit	0ns		1µs		
Tki	time for which CLK is held low to request a send	63.5	64	64.5	µs	
Tkrg	clock low from ARM7500FE to clock low from keyboard for request to send	1			µs	
Tksb	clock low to data low hold time for request to send	1			µs	

Table 18-6: Keyboard/mouse cycles

18.10 Analog to Digital Converter Interface

ARM7500FE contains four analog comparators with 16-bit timers, which are designed primarily for the implementation of an analog joystick interface. Each converter is of the slope integration type, using an external RC network attached to the appropriate **ATOD[3:0]** pin to generate a variable ramp delay.

The time taken for the voltage at the input to the comparator to reach the comparator's threshold is measured by a 16-bit counter which is stopped when the threshold of the comparator is reached. At this point an internal "stop" flag for that channel is set. The value is held in the counter until it has been read and the channel is then reset.

Discharge transistors on the analog inputs are used to discharge the external capacitor and to initiate a new integration cycle.

18.10.1 Counters

Each of the four counters can be reset by programming one of four bits in the ATODCR register. The four counters cannot be written to but can be read at addresses as follows:

CNT1 (0x032000EC)	counter 1
CNT2 (0x032000F0)	counter 2
CNT3 (0x032000F4)	counter 3
CNT4 (0x032000F8)	counter 4

The four counters have been implemented as simple asynchronous ripple counters, and it is therefore important that they should not be read until the 'stop' flag for that particular channel has been set, as seen in the status register, to indicate that the counter has been stopped and the read back value will be stable.

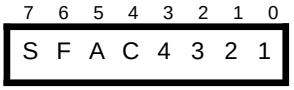
18.10.2 Interrupt control

There is a single bit in the main ARM7500FE interrupt handling registers (bit 2 of the IRQD set) which can accept an interrupt from the A to D converters. Thus, some interrupt pre-processing is done to determine how this main interrupt is to be generated. An interrupt control register is provided so that various combinations of channels can generate the final interrupt.

There are four possible interrupt sources, one for each channel, and each channel attempts to generate an interrupt when the comparator threshold is reached and the 'stop' flag is set internally.

Each of these interrupt sources can be individually enabled using the lower four bits of the Interrupt Control register, and the upper four bits determine which combination of bits will create the main interrupt which is passed to the IRQD registers.

Address 0x032000E0 - Interrupt Control



1	channel 1 interrupt enable
2	channel 2 interrupt enable
3	channel 3 interrupt enable
4	channel 4 interrupt enable
C	any combination of channels generates nIRQ
A	only all channels enabled generates nIRQ
F	first pair enabled generates nIRQ
S	second pair enabled generates nIRQ
Write	bit[7:0] 0: disabled, 1: enabled
Read	return above values
Reset	reset to 0x0F

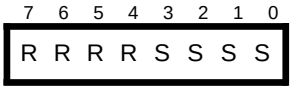
Note: *The OR of bit[3:0] is used to power-up all the comparators. Thus they reset to the powered-up state.*

18.10.3 Status of interface

The status of the 'stop' flag for each channel can be read directly from bits 0 to 3 of the status register, as can the interrupt status, which is simply the logical AND of the 'stop' flag values and the corresponding channel enables from the interrupt control register.

This register should be read by the system software in a polled system to check whether a channel has reached its final count value and is thus waiting to be read before another conversion cycle can be initiated.

Address 0x032000E4 - Status



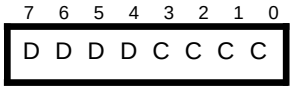
R[3:0]	interrupt request state for channels 4 to 1
S[3:0]	stop flag for channels 4 to 1
Write	ignored
Read	bit[7:4]
	0 not requesting
	1 requesting
Reset	set all zero (not requesting)

18.10.4 Control

The converter control register allows the discharge transistors and counters for each channel to be enabled and disabled, to give full control over the resetting of the counter and the timing of the start of a conversion cycle. Before a conversion can be started, the discharge bit and the counter clear bit for the channel in question should be forced one and zero respectively, and then the bits should be returned to zero and one respectively to actually initiate a conversion cycle. This will cause the analog voltage across the external capacitor to begin to ramp up, and simultaneously the 2MHz clock to the counters will be enabled, thus starting the count.

Synchronization between the memory system clock which is used to program the registers, and the 2MHz I/O world clock results in a small extra delay before the counter is really enabled, but this is negligible against the 0.5µs period of the 2MHz clock.

Address 0x032000E8 - Converter control



D[3:0]	discharge transistor control for channels 4 to 1
C[3:0]	clear counter for channels 4 to 1
Write	bit[7:4]
	0 transistor off
	1 transistor on (discharge)
	bit[3:0]
	0 clear counter
	1 enable counter
Read	return above values
Reset	set all zero (clear counters and don't discharge)

18.10.5 Comparators

The comparators are accurate to 2.5mV resolution and require a stable reference voltage of less than 2.5V to function correctly. The reference voltage is applied at the **ATODREF** pin. The same reference voltage is routed to all four comparators.

In order for the comparators to function correctly, it is essential that the reference current to the Video DACs on the VIREF pin is present, as this current is used to generate the operating current used by the gain stages in the comparator. The comparator reference currents are disabled to save power if all the interrupt enables (bits 0 to 3 of the interrupt control register) are set to zero. So, at least one channel must be enabled for any of the channels to function correctly.

18.10.6 Converter operation

The values of the capacitance and variable resistance used in the external RC circuit determine the range of time delays which will be seen from the moment the capacitor begins to charge to the moment that the comparator threshold is crossed.

The 16-bit counters are clocked by the 2MHz internal clock (derived from the 32MHz **I_OCLK**), and thus the counter will count for 65536 values over 32.7ms before returning to zero. In order to provide a meaningful reading from the converter, it is important that the capacitor and variable resistor values are such that this time will not be exceeded under the worst case conditions. The A to D converter is effectively providing a digital count directly related to the value of the resistance in the RC circuit.

18.11 Timers

The ARM7500FE includes two general-purpose timers which can be used as interrupt sources. Each timer is implemented as a 16-bit down counter, and has an input latch and an output latch associated with it. The counter decrements continuously, clocked at 2MHz. When it reaches zero, it is reloaded from the input latch and the downcount restarts.

There are four 8-bit-wide registers associated with the two timers. Each timer has

- two eight bit registers corresponding to the 16-bits of the timer
- two further write-only registers which cause the GO and LATCH commands to be issued to the appropriate timer when written to

The diagram below shows the timer configuration.

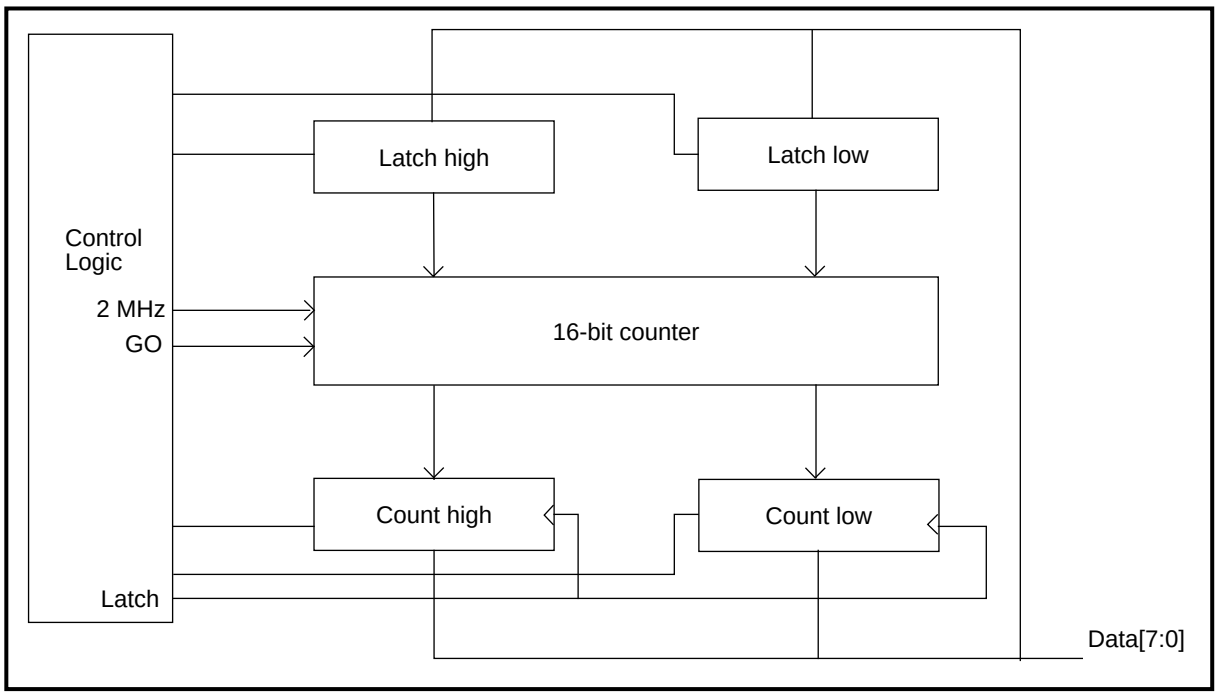


Figure 18-23: Timer configuration

18.11.1 Programming the timers

The locations of the registers can be found in *Chapter 16: Memory and I/O Programmers' Model* .

Writing to the following registers updates the values as described below:

T0LOW register	updates the value in the lower half of the timer 0 input latch
T0HIGH register	updates the value in the upper half of the timer 0 input latch with the written value.
T0GO register	loads the counters immediately with the value programmed into the input latch. If the counter is loaded with zero it will continuously reload.
T0LATCH register	places the current count value in the output latch.

Reading the following registers updates the values as described below:

T0HIGH register	returns the upper 8 bits of the count value
T0LOW register	returns the lower 8 bits of the count value.

18.11.2 Timer interrupts

Each timer will generate an interrupt when it reaches zero and is reloaded. These interrupts are handled by the IRQA set of interrupt processing registers (bits 5 and 6).

The timers can be used to generate timed interrupts at regular intervals T, where:

$T = (T0LOW + (256 * T0HIGH)) * 0.5\,\mu s.$

18.12 General-purpose, 8-bit-wide, I/O Port

A general-purpose 8-bit-wide I/O port is included in the ARM7500FE. The eight open drain output pins **IOP[7:0]** can be driven LOW or monitored as inputs by using the IOLINES register at address 0x0320000C.

When read, this register will return the current value seen at the **IOP[7:0]** pins. When written to, each bit will control the status of the corresponding **IOP** pin. When a one is written to a bit, that pin's output enable is switched off and it can be driven as an input. When a zero is written to a bit, the corresponding output pin is forced LOW.

There is a complete set of three interrupt control and status registers (IRQD) for the **IOP** pins, which allow any bit to generate a unique interrupt. The interrupt is generated when the corresponding **IOP** bit is LOW.

18.13 ID and OD Open Drain I/O Pins

There are three further open drain I/O pins:

ID	is intended to be used with an ID chip, which outputs a unique system ID when the ID pin is forced LOW. During Power On Reset the ID output is forced LOW, and it then becomes tri-state on leaving reset.
OD[1:0]	could be used to implement a simple serial link.

These are written to via the IOCR register, and are not capable of generating interrupts. Each pin is forced LOW by programming a zero to the appropriate bit in the IOCR register. Programming a one to any bit causes the corresponding pin to be tri-stated, and the value of the input level applied to the pin can then be read back from the same bit of the IOCR register.

Note: *These three pins do not have pull ups on-chip, and so it is advisable to fit them externally if they are not connected to another device.*

18.14 Version and ID Registers

The ID register is composed of two 8-bit hardwired registers which are read only. The lower byte is accessed at location 0x03200094, and the upper byte at location 0x03200098. Together they should return the value 0xAA7C.

The Version register is accessed at location 0x0320009C, and this will read back the version number of the device.

Note: *Under no condition should either of these registers be written to, as this may cause the chip to enter a test mode.*

18.15 Interrupt Control

The ARM7500FE interrupt handler takes interrupts from a variety of sources and generates the IRQ or FIQ interrupt signals required by the ARM processor, depending on the settings of the control and enable bits in the five sets of interrupt registers. The five sets are:

- FIQ
- IRQA
- IRQB
- IRQC
- IRQD

Each of these has a status, mask and request register associated with it, giving a total of 15 registers.

Table 18-7: Interrupt table on page 18-40 shows the interrupt sources featuring in each set of registers. The polarity entry refers to the level required at the external pin to set the interrupt. ‘Internal’ means that the interrupt is generated as a result of an internal state change, as opposed to change on an external pin.

When an interrupt signal is received from one of the interrupt sources, it causes the corresponding bit in the status register to go HIGH. This bit is then logically ANDed with the appropriate bit in the mask register, to create a value in the appropriate bit of the request register. If any of the bits in any of the IRQ request registers are HIGH, then the ARM7500FE will generate an internal IRQ interrupt to the ARM processor macrocell, causing the IRQ exception to be taken. If any of the bits in the FIQ request register are HIGH, the ARM7500FE will generate an internal FIQ interrupt to the ARM processor, causing the FIQ exception to be taken.

The system software can then read the request registers to determine which sources were requesting an interrupt. Reading the status registers will show which sources were requesting interrupts, even if they were masked.

The IRQA request register is slightly different in that some of the interrupt flags are edge triggered and thus need to be cleared after they have been read. All other request registers are read only, but the IRQRQA register can be written to clear triggered interrupts. Writing a one to a bit clears that interrupt. Writing a zero causes no action to be taken.

Register	Bit	Polarity/Type	Name/Function
FIQ	7		Always active for software generated **FIQ**.
	6	LOW	**nINT8** interrupt pin
	5		
	4	LOW	**nINT6** interrupt pin
	3		
	2		
	1	HIGH	**INT5** interrupt pin
	0	HIGH	**INT9** interrupt pin
IRQA	7		Always active for software generated **IRQ**.
	6	internal	2MHz timer 1
	5	internal	2MHz timer 0
	4	falling edge	**nPOR** power on reset
	3	internal	Flyback from video subsystem
	2	falling edge	**nINT1** interrupt pin
	1		
	0	rising edge	**INT2** interrupt pin
IRQB	7	internal	Keyboard Rx buffer full
	6	internal	Keyboard Tx buffer empty
	5	LOW	**nINT3** interrupt pin
	4	LOW	**nINT4** interrupt pin
	3	HIGH	**INT5** interrupt pin
	2	LOW	**nINT6** interrupt pin
	1	HIGH	**INT7** interrupt pin
	0	LOW	**nINT8** interrupt pin
IRQC	7	LOW	**IOP[7]** interrupt pin

Table 18-7: Interrupt table

Register	Bit	Polarity/Type	Name/Function
	6	LOW	**IOP[6]** interrupt pin
	5	LOW	**IOP[5]** interrupt pin
	4	LOW	**IOP[4]** interrupt pin
	3	LOW	**IOP[3]** interrupt pin
	2	LOW	**IOP[2]** interrupt pin
	1	LOW	**IOP[1]** interrupt pin
	0	LOW	**IOP[0]** interrupt pin
IRQD	7		
	6		
	5		
	4	LOW	**nEVENT2** wake-up event
	3	LOW	**nEVENT1** wake-up event
	2	internal	A to D convertor interrupt
	1	internal	Mouse Tx buffer empty
	0	internal	Mouse Rx buffer full

Table 18-7: Interrupt table (Continued)

19 Clocks, Power Saving, and Reset

This chapter describes clock control, power management, and reset.

Clocks, Power Saving, and Reset

19.1 Clock Control

ARM7500FE has a clocking scheme designed to allow maximum flexibility for the system designer. There are three main clock inputs:

CPUCLK CPU clock, used to generate the ARM processor's FCLK

MEMCLK Memory subsystem clock, used to generate the memory system clock, and the ARM processor's MCLK

I_OCLK I/O system clock, which should be fixed at 32MHz (in divide by 1 mode) or 64MHz (in divide by 2 mode), and is used to generate all the fixed frequency I/O clocks and refresh rates.

19.1.1 Video and sound subsystem clocks

The video sub-system has two separate external clock inputs and includes a phase locked loop to enable the control of an external VCO.

The pixel clock source can be selected to be **VCLKI** (using an external VCO), **HCLK**, which is driven directly in from the **HCLK** pin, or IOCK32 (also referred to as RCLK), which is the internal I/O subsystem clock and is generated directly from the main I_OCLK input pin as described below. The sound subsystem can be clocked either from IOCK32 generated internally from **I_OCLK**, or by using an externally generated clock connected to the **SCLK** pin.

Selection between these various clock sources is described in the video and sound sub-systems section of this data sheet.

19.1.2 I/O clock outputs

Four fixed frequency I/O clocks are output by the ARM7500FE, all divided down from the fixed frequency input **I_OCLK** which should be set to 32MHz in divide-by-1 mode. These are:

CLK16 (16MHz)

REF8M (8MHz)

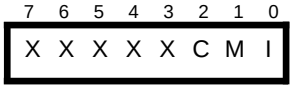
CLK8 (An inverted version of **REF8M**)

CLK2 (2MHz)

19.1.3 Synchronous/asynchronous mode for the ARM processor

The ARM processor macrocell can be configured to work in synchronous or asynchronous mode, under the control of the **SnA** pin. Synchronous mode can only be used within the ARM7500FE if the correct relationship is maintained between the internal ARM processor clocks, FCLK and MCLK and in fact when **SnA** is set HIGH, both FCLK and MCLK are derived from **MEMCLK**, with a suitable delay to ensure the required phase relationship between FCLK and MCLK is held correctly, ie. **CPUCLK** is ignored when **SnA** = 1. In particular, FCLK will be equal to MEMRFCK (see section *19.1.4 Clock prescalers* on page 19-3) and MCLK will be equal to half MEMRFCK. If the FCLK frequency is required to be different from the MEMRFCK frequency, the **SnA** pin must be held LOW, and a suitable frequency applied to **CPUCLK**.

19.1.4 Clock prescalers



Each of the three main clock inputs **CPUCLK**, **I_OCLK** and **MEMCLK** has a selectable divide by 2 prescaler available within ARM7500FE to enable a guaranteed 50:50 mark-space ratio internal clock to be produced using a higher frequency external oscillator. The internal clocks, which will be referred to elsewhere in this data sheet, are called FCLK, IOCK32 and MEMRFCK respectively.

On Power On Reset, all the prescalers will be set to divide by 2. The prescaling is controlled by the CLKCTL register at address 0x0320003C, and there is one bit to enable or disable each divide by 2 prescaler as required:

C **CPUCLK** divide control

M **MEMCLK** divide control

I **I_OCLK** divide control

Write bit[2]

- 0 FCLK x 2 = **CPUCLK**
- 1 FCLK = **CPUCLK**

bit[1]

- 0 MEMRFCK x 2 = **MEMCLK**
- 1 MEMRFCK = **MEMCLK**

bit[0]

- 0 IOCK32 x 2 = **I_OCLK**
- 1 IOCK32 = **I_OCLK**

Read return above value

Power On Reset

set all to zero, ie. divide by 2 clocks

19.1.5 Clocking schemes

The simplest mode of operation of the ARM7500FE has all three of the main clocks driven by a single 32MHz oscillator, with the prescalers set to divide-by-1 mode. However, it is possible to increase the speed of the memory and CPU clocks, noting that if this requires FCLK and MEMRFCK frequencies to be different, the **SnA** input must be set LOW for asynchronous operation and a suitable clock applied to **CPUCLK**. The **I_OCLK** frequency must remain at 32MHz (or 64MHz if the divide by 2 prescalers are enabled).

Note: *Nearly all timings in this datasheet assume that both **I_OCLK** and **MEMCLK** are running at 32MHz (or 64MHz with the divide by 2 prescalers on).*

Increasing the memory clock frequency allows the system designer to take advantage of faster DRAM memory. The ARM7500FE includes full synchronization at the interface between the memory and I/O sub-systems to ensure safe operation under asynchronous conditions.

19.2 Power Management

The ARM7500FE includes power management circuitry which greatly enhances its suitability for battery powered portable applications where power consumption is of paramount importance. There are three power management modes:

NORMAL	the default operating condition in which all clocks are running and the chip is functioning normally.
SUSPEND	the clocks to the CPU (FCLK and MCLK) are stopped, but all other parts of the chip remain active so DMA can continue and the display can continue to be refreshed. It is also possible to stop some of the external I/O clock outputs to save more power if this can be done safely without causing problems for I/O peripherals connected to these clocks.
STOP	allows all the clocks to the ARM7500FE to be stopped, and the whole chip will then draw only leakage currents provided all required registers have been appropriately programmed. Outputs are provided from the ARM7500FE to enable the oscillator(s) to be powered down, and circuitry to allow the oscillator(s) to cleanly restart using an external RC delay before the clocks inside the ARM7500FE are re-enabled. Before STOP mode is entered, a number of registers need to be programmed appropriately in the video sub-system, and further details of the full sequence of events required to make most effective use of the power management features can be found in *19.2.2 STOP mode*.

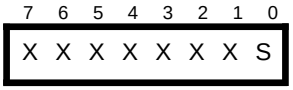
19.2.1 SUSPEND mode

Entry into SUSPEND mode is achieved by writing to the register location 0x0320001C. Any value can be used, but the value written to bit 0 will determine whether the external I/O output clocks **CLK16**, **CLK8**, **REF8M** and **CLK2** are stopped. DMA may continue unaffected, allowing the display and DRAM data to remain refreshed.

Exit from SUSPEND mode is achieved by a falling edge on either of the asynchronous input event pins, **nEVENT1** and **nEVENT2**, or by any enabled interrupt source generating a FIQ or IRQ interrupt for the ARM processor. The assertion of **nRESET** will also cause exit from SUSPEND mode. It is important that the interrupt mask and enable registers are programmed appropriately before SUSPEND mode is entered if it is intended that an interrupt source be used to terminate the power saving mode.

The CPU will merely see SUSPEND mode as a write to a location in the memory and I/O register area. It will be unaware of the duration of this write, as both **MCLK** and **FCLK** are frozen, and it is a fully static device. The careful use of SUSPEND mode when no CPU operations are required will have a significant effect on the device‘s average power consumption. It could be used, for example, between key presses while waiting for more user input. The keyboard controller is still clocked during SUSPEND mode and so will be able to generate interrupts which will cause the termination of the write cycle and then cause the CPU to take the interrupt exception.

Details of the SUSMODE register (address 0x0320001C) are shown below:



S	SUSPEND mode control of external I/O clocks
Write	turn off external I/O clocks when in this mode
	0 turn off
	1 don't turn off
	Enter Suspend mode with MCLK,FCLK,I/O clocks and some internal clocks stopped. DMA continues and instruction completes on either wake-up event, nIRQ or nFIQ.
Read	return above value
Reset	set to zero

19.2.2 STOP mode

Entry into STOP mode is achieved by writing to the register location 0x0320002C. Any value can be written to the register to enter STOP mode, but the value written will appear on the external data bus of the ARM7500FE while the chip is in STOP mode. It is therefore recommended that the value 0xFFFFFFFF be written to this register as this will mean that both **D[31:0]** and **LA[28:0**] are driven HIGH during STOP mode.

It is very important that all DMA activity is stopped, that all I/O activity is completed, and that the video subsystem is powered down correctly before the STOP mode register is written to. The **OSCPOWER** output is controlled by the power management circuitry, and will be forced LOW a short time after the write cycle begins. This output may be used to disable the external oscillator(s).

Exit from STOP mode can only be achieved by the use of the asynchronous wake-up event pins **nEVENT1** and **nEVENT2**. When either of these is forced LOW, a sequence of events will be triggered which will cause the oscillator(s) to be restarted cleanly.

During STOP mode, a zero is driven out from the **OSCDELAY** pin, which ensures that an external capacitor forming part of an RC network attached to the **OSCDELAY** pin remains discharged. As soon as a wake up event occurs the **OSCPOWER** pin is set HIGH again, and the open drain **OSCDELAY** pin is allowed to float and becomes an input.

At this point, the external capacitor starts to charge, until the schmitt threshold of the **OSCDELAY** input is exceeded. From this point, a further two rising edges must be seen on the input clock from the oscillator before the clock is allowed through to the internal ARM7500FE circuitry. The component values used in the RC circuit should be chosen to ensure that the oscillator has sufficient time to stabilize before the **OSCDELAY** input is triggered.

As the video subsystem is inherently dynamic for performance reasons, it is necessary to set it into a special Powerdown mode before STOP mode is entered. To do this, the video Ext register should be programmed with the data 0xC0000000, the Video Control register should be programmed with the data 0xE00040xx (the last byte will depend on the clock source and configuration), and the Sound Control register should be programmed with the data 0xB1000000 (if the sound system is configured for use

with the **SCLK** pin as the clock source). If the sound system is being clocked from the ARM7500FE's internal 32MHz I/O clock, then the register should be programmed with the value 0xB1000001. These actions will disable the video datapath and ensure the entire macrocell is forced into a static state. To ensure that the comparators in the A to D converters do not consume current, they should be shut down by programming the value 0x00 into the ATODICR register at location 0x032000E0.

ARM7500FE includes support for self refresh DRAM, and it is intended that this feature should be used during STOP mode to ensure that DRAM contents are preserved. This DRAM mode is activated by allowing direct software control of the **nCAS** and **nRAS** output pins. The SELFREF register (0x032000D4) can be used to directly force the **nRAS** and **nCAS** output pins according to the protocol required for a particular DRAM, in order to enter self-refresh mode. This programming must be performed by code executing from ROM.

In STOP mode ARM7500FE will consume leakage currents only, and can be held indefinitely without corruption of the internal registers, CPU cache, etc.

19.3 Reset

The ARM7500FE has three pins associated with reset. The **nPOR** pin is intended for use with an external RC delay to generate a power-on-reset pulse when the chip is switched on. The **nRESET** pin is an open drain I/O pin, which is intended to be used to generate a "soft" reset. Both **nPOR** and **nRESET** are active LOW schmitt inputs. The active HIGH **RESET** pin is a clean reset output, which is created from the synchronized version of the **nRESET** input, and is also forced HIGH during **nPOR**.

A LOW state on the **nPOR** input sets the POR bit in the IRQA status register. This bit can later be examined to show that the reset which occurred was an **nPOR** type rather than **nRESET**. The POR bit in the IRQA status register is not reset until the POR clear bit in the IRQA request register is written to. **nPOR** also causes the prescalers on the clock inputs to be set to divide by 2. The **nPOR** input is passed through a pulse stretcher which ensures that even a short pulse on the input will guarantee a full reset of the whole of ARM7500FE. See *Figure 19-1: nPOR timing diagram*. During **nPOR** reset, **nCAS** is forced low throughout and the **nRAS** outputs are changed according to the sequence in *Figure 17-14: Refresh cycle timing* on page 17-19. While **nPOR** is LOW, **nRESET** and **ID** (which are both open drain pins) are held LOW, and an incrementing address value will be output on the **LA** address bus.

A LOW state on the **nRESET** input is used to generate a 'soft' reset. This does not set any interrupt flags, and the **nRESET** LOW state must exist for longer than 2us to guarantee that it is seen, as it is passed through a synchronizer before being used by the internal circuitry. *Figure 19-2: nRESET timing diagram* below shows the required timing of **nRESET** to ensure correct operation. At the start of the **nRESET** active period, the whole ARM7500FE (including the DRAM refresh state machine and counter) is reset for 1us, and for the remaining duration of the **nRESET** pulse, DRAM refresh takes place at the highest selectable rate. During **nRESET**, the ARM processor outputs an incrementing address on the **LA** bus.

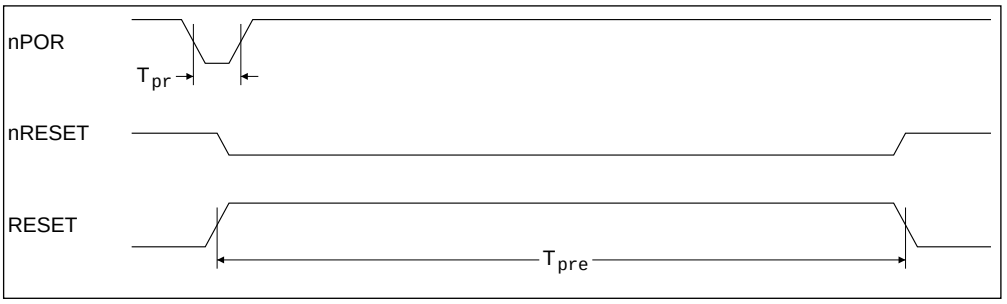


Figure 19-1: nPOR timing diagram

Symbol	Parameters	Min	Typ	Max	Units	Notes
Tpr	time for which nPOR must be held low to guarantee a reset	20			ns	
Tpre	length of internal reset	2		4	µs	1

Table 19-1: nPOR and nRESET timing

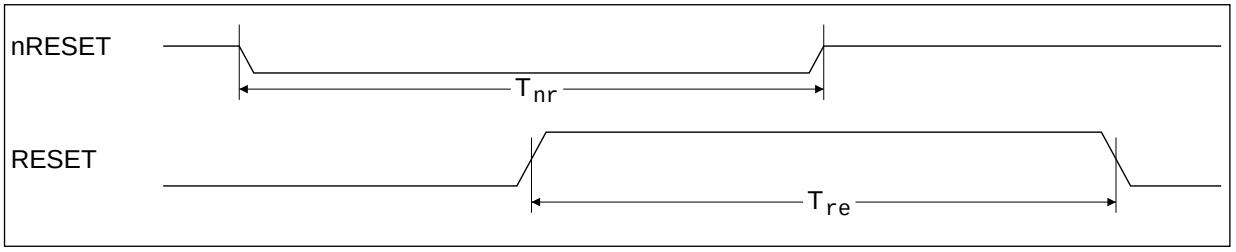


Figure 19-2: nRESET timing diagram

Symbol	Parameters	Min	Typ	Max	Units	Notes
Tnr	time for which nRESET must be held low to guarantee reset	2			µs	2, 3
Tre	length of internal reset	2			µs	3

Table 19-2: nRESET timing

1 Tpre = 2µs if I_OCLK is 64MHz. Tpre is 4µs if I_OCLK is 32 MHz as this reset forces divide by 2 mode on the clock inputs.

2 DMA or writes from the ARM Processor prevent **nRESET** having any effect for their duration. Thus the “soft” reset cannot break write cycles or cause partial DRAM refresh.

3 Assuming IOCK32 is 32MHz.

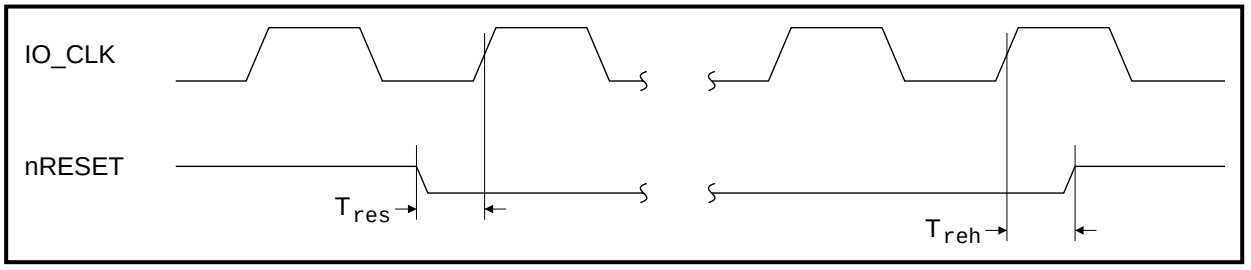


Figure 19-3: nRESET timing

Symbol	Parameters	Min	Typ	Max	Units	Notes
Tres	nRESET setup to I_OCLK rising	0			ns	
Treh	nRESET hold from I_OCLK rising	30			ns	

Table 19-3: nRESET timing

When in STOP mode, **nRESET** will force the power management control circuitry to revert to normal mode, without necessarily causing a reset sequence to occur.