Features

- **Compatible with an Embedded ARM7TDMI® Processor**
- **Generates Transfers to/from Serial Peripheral Such as UART, USART, SSC and SPI**
- **Supports Up to 12 Peripherals – Parameterizable on Request**
- **One ARM® Cycle Needed for a Transfer from Memory to Peripheral**
- **Two ARM Cycles Needed for a Transfer from Peripheral to Memory**
- **Fully Scan Testable up to 98% Fault Coverage**
- **Can be Directly Connected to the Atmel Implementation of the AMBA™ Bridge**
- **Not Fully Compatible with AMBA: Retract Response not Supported**

Description

The Peripheral Data Controller 2 (PDC2) transfers data between on-chip peripherals such as the UART, USART, SSC and SPI and the on- and off-chip memories. This transfer is achieved via the AMBA Bridge using a simple arbitration mechanism between the AMBA System Bus (ASB) and the PDC2 to control Bridge access. This avoids processor intervention and removes the processor interrupt handling overhead. This significantly reduces the number of clock cycles required for a data transfer and, as a result, improves the performance of the microcontroller and makes it more power-efficient.

The PDC2 channels are implemented in pairs, each pair being dedicated to a particular peripheral. One PDC2 channel in the pair is dedicated to the receiving channel and one to the transmitting channel of each UART, USART, SSC and SPI.

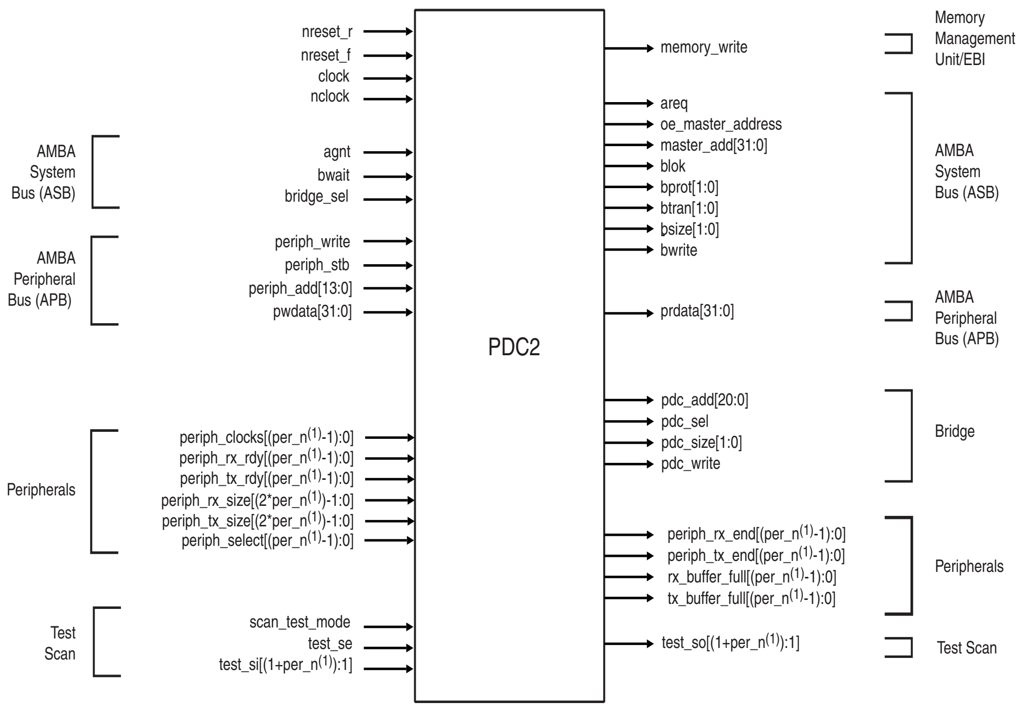
The user interface of a PDC2 channel is integrated in the memory space of each peripheral. It contains a 32-bit memory pointer register and a 16-bit transfer count register plus a 32-bit register for next memory pointer and a 16-bit register for next transfer count. The peripheral triggers PDC2 transfers using transmit and receive signals. When the programmed data is transferred, an end of transfer interrupt is generated by the corresponding peripheral.

32-bit Embedded Core Peripheral

Peripheral Data Controller 2 (PDC2)

Rev. 1734A–03/01

Figure 1. PDC2 Symbol



Note: 1. per_n = Number of Peripherals

Table 1. PDC2 Pin Description

Name	Definition	Type	Active Level	Comments
Chip-wide				
nreset_r	System Reset	Input	Low	Resets all counters and signals – clocked on rising edge of clock
nreset_f	System Reset	Input	Low	Resets all counters and signals – clocked on falling edge of clock
clock	System Clock	Input	–	System clock
nclock	System Clock	Input	–	Inverted system clock
AMBA System Bus (ASB)				
agnt	Grant Signal	Input	High	Arbiter grants the bus to the PDC2 when this input is set to 1
bwait	Bus Wait	Input	High	1 cycle wait is required
bridge_sel	Bridge Select	Input	High	From address decoder of system bus
areq	Request Signal	Output	High	Bus request sent to the Arbiter
oe_master_address	Output Enable	Output	High	Output address enable – this signal indicates that master_add[31:0], blok, bprot[1:0], bsize[1:0] and bwrite signals are currently valid with PDC2 granted on the bus
master_add[31:0]	Address System Bus	Output	–	Address bus generated by master
blok	Bus Locked	Output	High	Indicates that the ongoing instruction must not be interrupted
bprot[1:0]	Bus Protection	Output	–	Protection information
bsize[1:0]	Size of Transfer	Output	–	Bus size
btran[1:0]	Type of Transfer	Output	–	Bus transfer
bwrite	Bus Write	Output	High	The PDC2 transfers data from the peripheral to internal memory
AMBA Peripheral Bus (APB)				
periph_write	Peripheral Write Enable	Input	High	From host (Bridge)
periph_stb	Peripheral Strobe	Input	High	From host (Bridge)
periph_add[13:0]	Peripheral Address Bus	Input	–	From host (Bridge)
pwdata[31:0]	Peripheral Data Bus	Input	–	From host (Bridge) – user interface data bus
prdata[31:0]	Peripheral Data Bus output	Output	–	User interface data bus
Peripherals				
periph_clocks [per_n-1:0]	Peripheral System Clocks (UART/ USART/SSC/SPI)	Input	–	Per_n values range from 1 to 12. The number of each type of peripheral connected to PDC2 is free. For example, the user can have 8 UARTS, 0 USARTS and 3 SPIs. LSBs are reserved for USARTs. Remaining upper bits are reserved for SPIs.

Table 1. PDC2 Pin Description (Continued)

Name	Definition	Type	Active Level	Comments
periph_rx_rdy [(per_n-1):0]	Peripheral Receiver Ready	Input	High	Once a character has been received by peripheral, one of these bits is set to 1. LSBs are reserved for USARTs. Remaining upper bits are reserved for SPIs
periph_tx_rdy [(per_n-1):0]	Peripheral Transmitter Ready	Input	High	Once the holding transmit register is available, one of these bits is set to 1
periph_rx_size [(2*per_n)-1:0]	Peripheral Transfer Sizes for Reception Side	Input	–	The per_n is the number of peripherals connected to the PDC2. This value changes the memory pointer. Two bits are reserved for each peripheral, for example, with two USARTs and one SPI, the size of transfer on the receiver side for: USART0 = periph_rx_size[1:0], USART1 = periph_rx_size[3:2] and SPI0 = periph_rx_size[5:4]
periph_tx_size [(2*per_n)-1:0]	Peripheral Transfer Sizes for Transmission Side	Input	–	The per_n is the number of peripherals connected to the PDC2. This value changes the memory pointer. Two bits are reserved for each peripheral, for example, with two USARTs and one SPI, the size of transfer on the transmit side for: USART0 = periph_tx_size[1:0], USART1 = periph_tx_size[3:2] and SPI0 = periph_tx_size[5:4]
periph_select [(per_n-1):0]	Peripheral selects	Input	High	From host (Bridge) – also input of each peripheral connected
periph_rx_end [(per_n-1):0]	Peripheral receive end	Output	High	End of receive transfer (each bit corresponds to a peripheral) – the associated buffer for the channel is full
periph_tx_end [(per_n-1):0]	Peripheral Transmit End	Output	High	End of transmit transfer (each bit corresponds to a peripheral) – the associated buffer for the channel is empty
rx_buffer_full [(per_n-1):0]	Peripheral Receive Buffer Full	Output	High	End of receive transfer (each bit corresponds to a peripheral) – the associated buffers for the channel are full
tx_buffer_empty [(per_n-1):0]	Peripheral Transmit Buffer Empty	Output	High	End of transmit transfer (each bit corresponds to a peripheral) – the associated buffers for the channel are empty
		Bridge Interface		
pdc_add[20:0]	PDC2 Address Bus	Output	–	Used by the Bridge to access the peripherals
pdc_sel	PDC2 Select	Output	High	Used by the Bridge to access the peripherals
pdc_size[1:0]	PDC2 Size of Transfer	Output	–	Multiplex the spi_size inputs – used by the Bridge to determine the size of the transfer between memories and the SPI
pdc_write	PDC2 Write	Output	High	Used by the Bridge to access the peripherals

Table 1. PDC2 Pin Description (Continued)

Name	Definition	Type	Active Level	Comments
Memory Interface				
memory_write	Memory Write from Peripheral	Output	High	Used by Memory Management Unit or EBI to select data coming from masters or peripherals (Bridge)
Test Scan				
scan_test_mode	Clock Selection for Test Purposes	Input	High	Tied to 1 during scan test – tied to 0 when in function mode
test_se	Scan Test Enable	Input	High /Low	Scan shift/scan capture
test_si [(1+per_n):1]	Scan Test Input	Input	–	Entry of scan chain
test_so [(1+per_n):1]	Scan Test Output	Output	–	Ouput of scan chain

Scan Test Configuration

The fault coverage is maximum if all non-scan inputs can be controlled and all non-scan outputs can be observed. In order to achieve this, the ATPG vectors must be generated on the entire circuit (top-level) which includes the PDC2, or all PDC2 I/Os must have a top level access and ATPG vectors must be applied to these pins.

Configuration

The PDC2 has a standard Atmel Bridge interface that enables the user to configure and control the data transfers for each channel. The user interface of a PDC2 channel is integrated into the user interface of the peripheral which it is related to. Per peripheral, it contains four 32-bit Pointer Registers (RPR, RNPR, TPR, TNPR) and four 16-bit Counter Registers (RCR, RNCR, TCR, TNCR).

The size of the transfer (number of transfers) is configured in an internal 16-bit transfer counter register, and it is possible, at any moment, to read the number of transfers left for each channel.

The base memory address is configured in a 32-bit memory pointer, by defining the location of the first access point in the memory. It is possible, at any moment, to read the location in memory of the next transfer.

The PDC2 has dedicated status registers which indicate if transfer is enabled or disabled for each channel — the remaining status for each channel is located in the peripheral. Transfers can be enabled and/or disabled by setting TXTEN/TXTDIS and RXTEN/RXTDIS in PDC2 Transfer Control Registers. The PDC2 sends status flags (periph_rx_end, periph_tx_end, rx_buffer_full, tx_buffer_empty) to the peripheral, which can latch the flags in its status register.

System Bus Interface

The PDC2 interfaces with the AMBA System Bus (ASB) and generates all the control signals for interfacing with a Memory Management Unit or EBI for memory read and write.

Memory Pointers

Each peripheral is connected to the PDC2 by a receive data channel and a transmit data channel. Each channel has an internal 32-bit memory pointer. Each memory pointer points to a location in the system bus memory space (on-chip memory or external bus interface memory).

Depending on the type of transfer (byte, half-word or word), the memory pointer is incremented by 1, 2 or 4, respectively for peripheral transfers.

If a memory pointer is reprogrammed while the PDC2 is in operation, the transfer addresses are changed, and the PDC2 performs transfers using the new address.

Transfer Counters

There is one internal 16-bit transfer counter for each channel. Each counter is used to count the size of the block already transferred by its associated channel. These counters are decremented after each data transfer. When the counter reaches zero, the transfer is complete and the PDC2 stops transferring data and disables the trigger while activating the related periph_end flag if the Next Counter Register is equal to zero.

If the counter is reprogrammed while the PDC2 is operating then the number of transfers is changed and the PDC2 counts transfers from the new value.

When the Next Counter Register is not equal to zero, for example, the values have been programmed into Next Pointer/Counter Registers, the behavior is the same, except that, after activating the flag periph_end when the transfer counter reaches zero, the values of the Next Pointer/Counter Registers are loaded into the Pointer/Counter Registers in order to re-enable triggers. The flag periph_end is automatically cleared when one of the counter registers (Counter or Next Counter Register) is written.

Note: When the Next Counter Register is loaded into the Counter Register, it is set to zero.

Data Transfers

The peripheral triggers PDC2 transfers using transmit (periph_tx_rdy) and receive (periph_rx_rdy) signals.

When the peripheral receives an external character, it sends a Receive Ready signal to the PDC2, which then requests access to the system bus (ASB) from the Bus Arbiter.

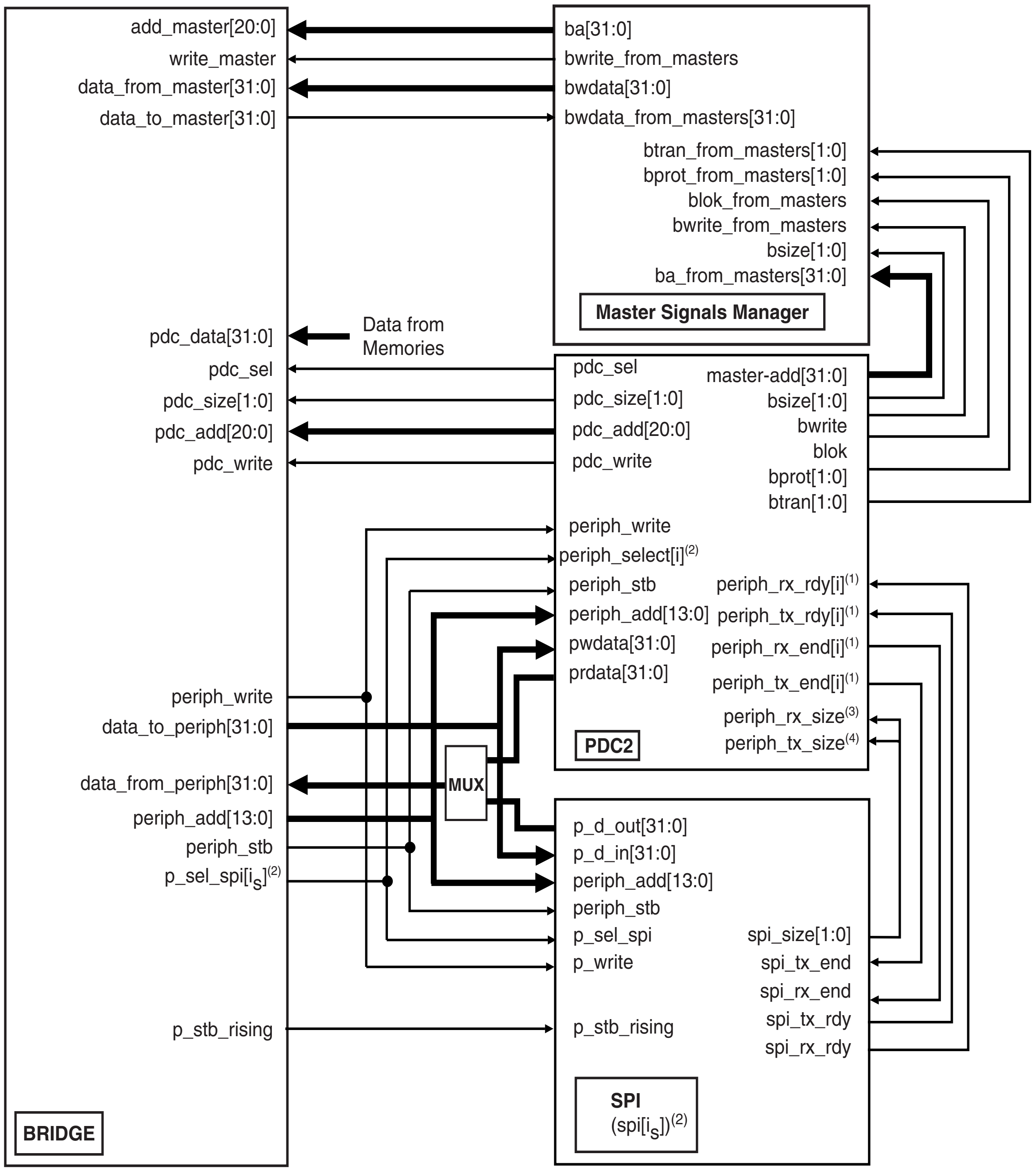
When access is granted, the PDC2 starts a read of the peripheral Receive Holding Register, via the dedicated pdc_add, pdc_sel, pdc_write and pdc_size signals to the Bridge.

Next, the PDC2 triggers a write in the memory by setting the ASB control signals and, at the same time, the Bridge provides the data that is to be written to the memory.

After each transfer, the relevant PDC2 memory pointer is incremented and the numbers of transfers left is decremented. When the memory block size is reached, a signal is sent to the peripheral and the transfer stops.

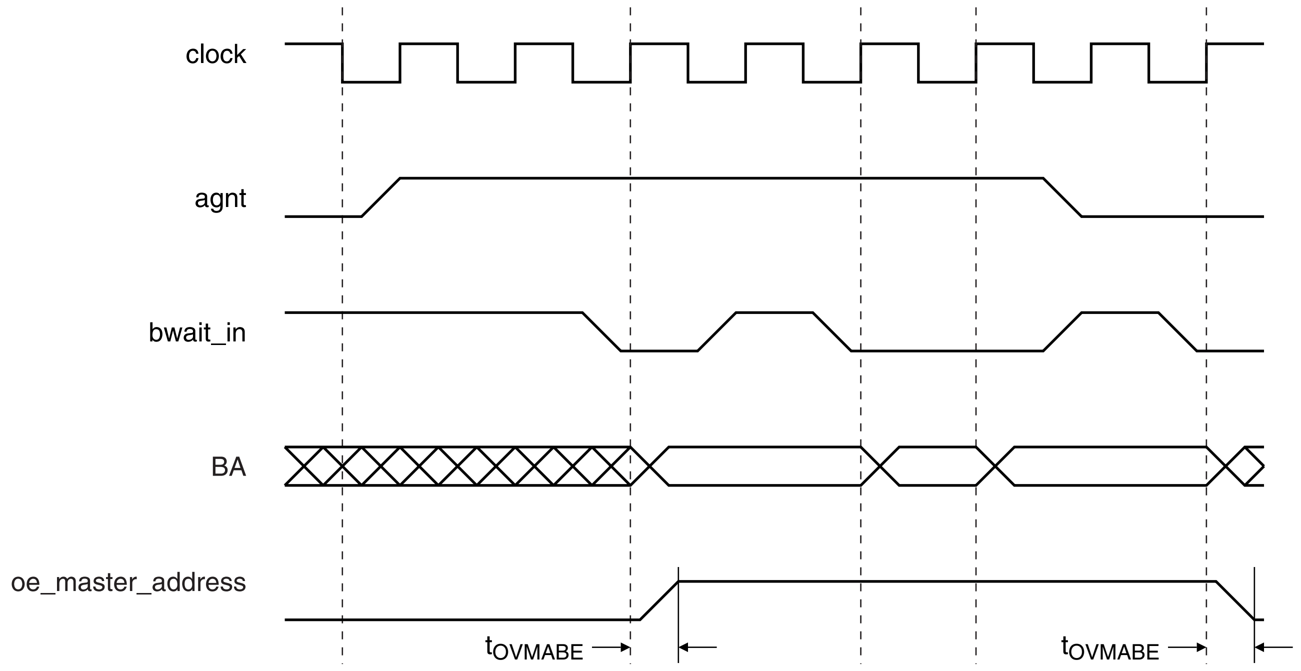
The same procedure is followed, in reverse, for transmit transfers. These timing exchanges are shown in the following figures.

Figure 2. Example of PDC2 Connection with Bridge and SPI



Notes:
1. i = index of peripheral, ranges from 0 to 11, if the SPI is the third peripheral, i = 2.
2. N = the total number of peripherals connected to PDC2. N_s= the total number of peripherals connected to PDC2 (N) – the number of SPI peripherals. i_S = index of SPI peripheral, ranges from 0 to (N - N_S - 1).
3. periph_rx_size = periph_rx_size [(2*i_s) - 1 + (2N_s):2*(i_s-1) + 2*N_s]
4. periph_tx_size = periph_tx_size [(2*i_s) - 1 + (2N_s):2*(i_s-1) + 2*N_s]

Figure 3. oe_master_address Signal for Atmel AMBA Bus



This output is generated to simplify the multiplexing of the control signals generated by the PDC2. It indicates that the PDC2 is “really granted” on the bus (ASB) and that its control signals must be sent to the slaves.

Thus, oe_master_address is asserted when the PDC2 is granted via agnt and there is no transfer being done by another master, i.e. bwait_in is inactive. oe_master_address is de-asserted when the core has finished its last transfer, i.e. bwait_in is inactive.

Figure 4. ASB to APB Transfer with Zero Wait States Memory Followed by an APB Access Made by Another Master

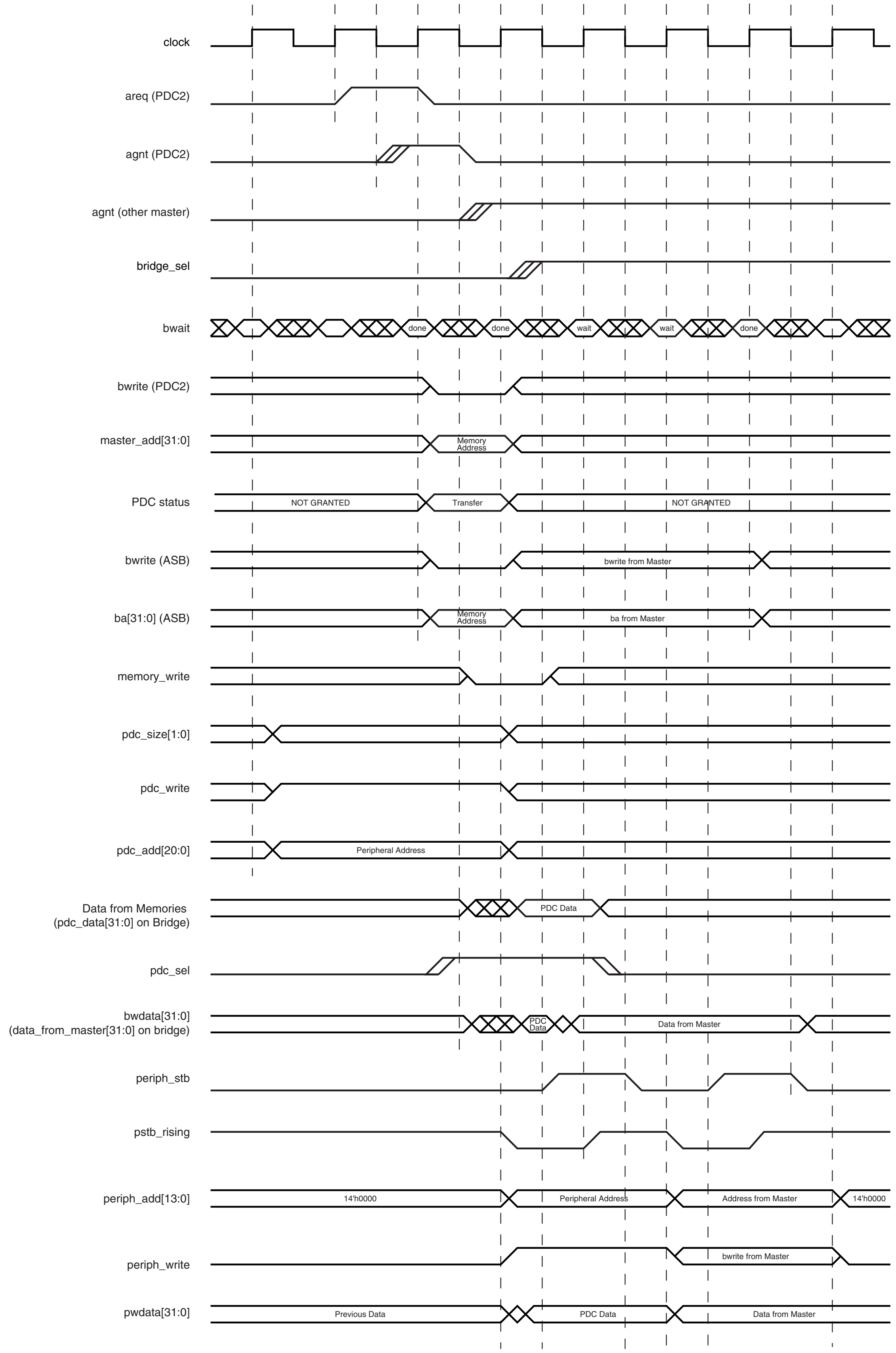


Figure 5. APB to ASB Transfer with Zero Wait States Memory Followed by an APB Access Made by Another Master

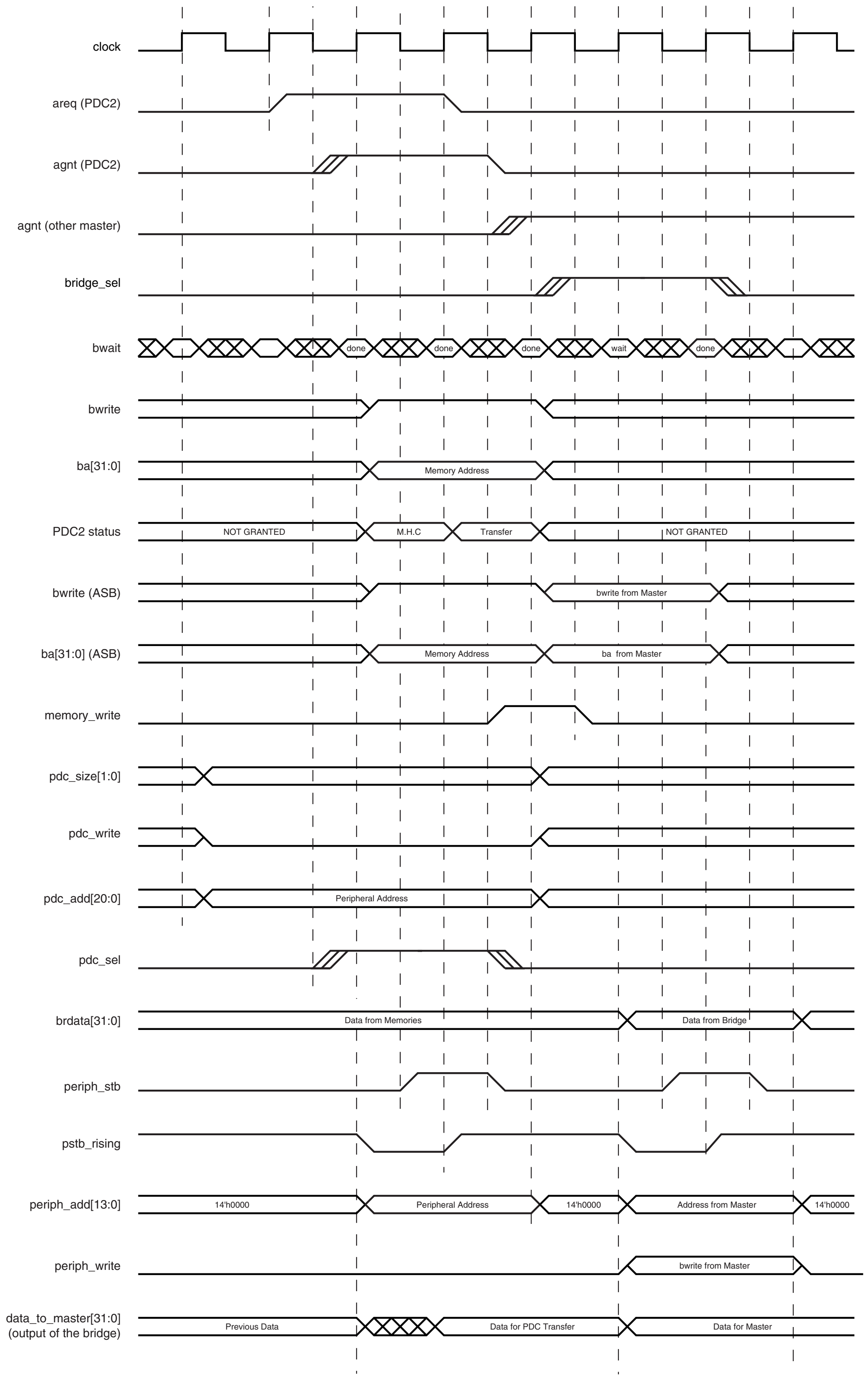


Figure 6. APB to ASB Transfer with Zero Wait States Following:

1. Series of APB Accesses Made by Another Master
2. Memory With One Wait State Made by Another Master

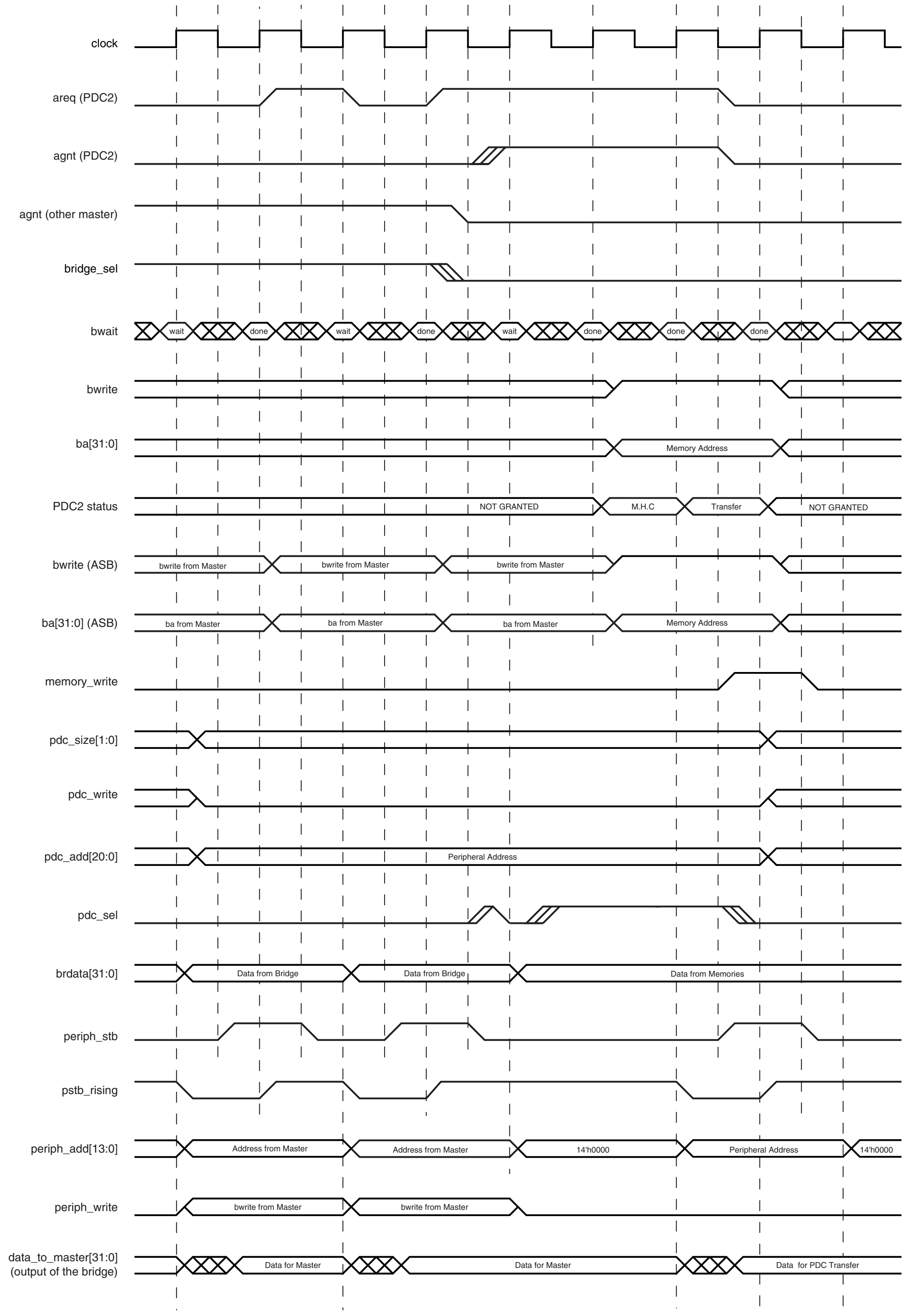


Figure 7. ASB to APB Transfer with Three Wait States Memory

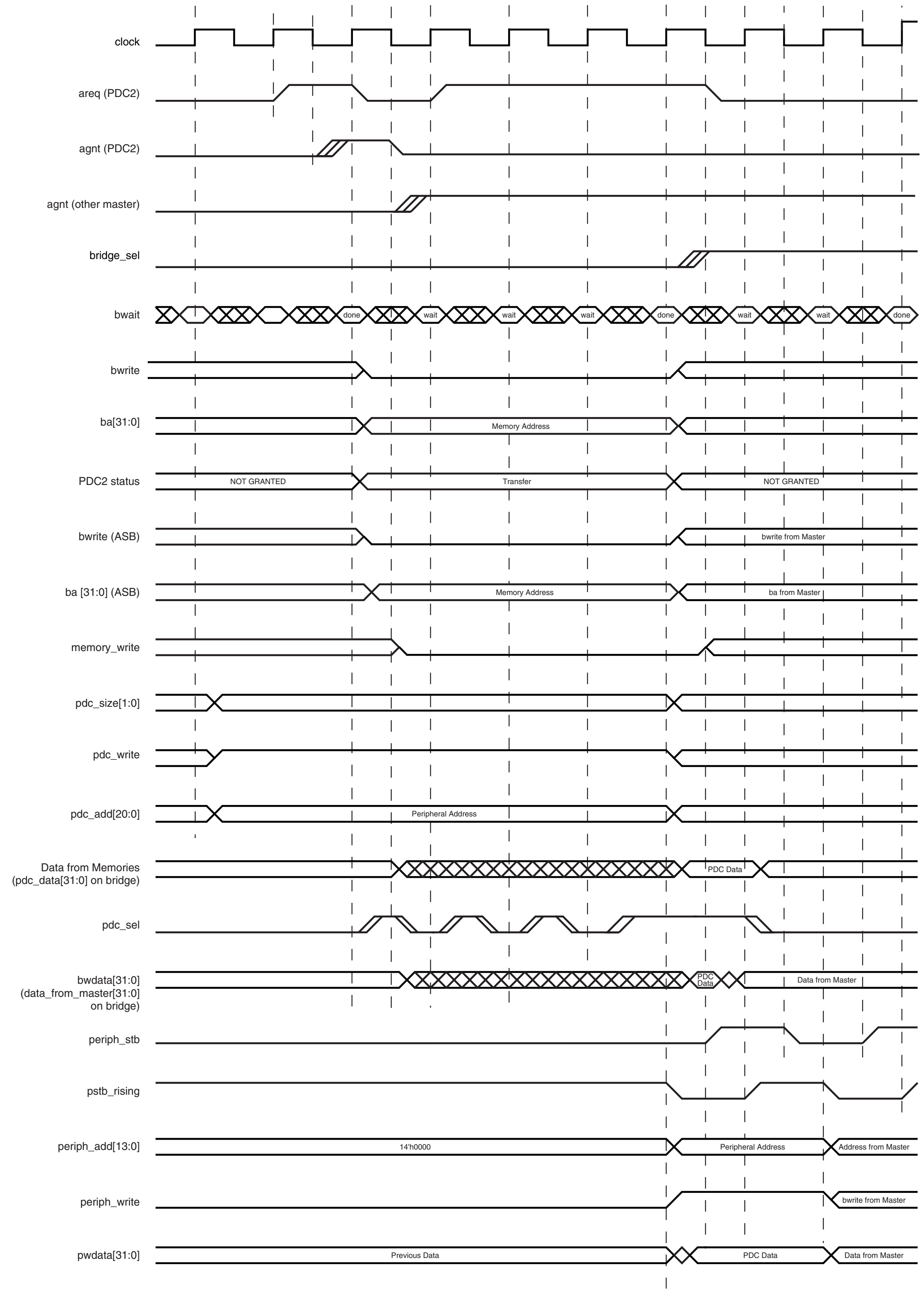
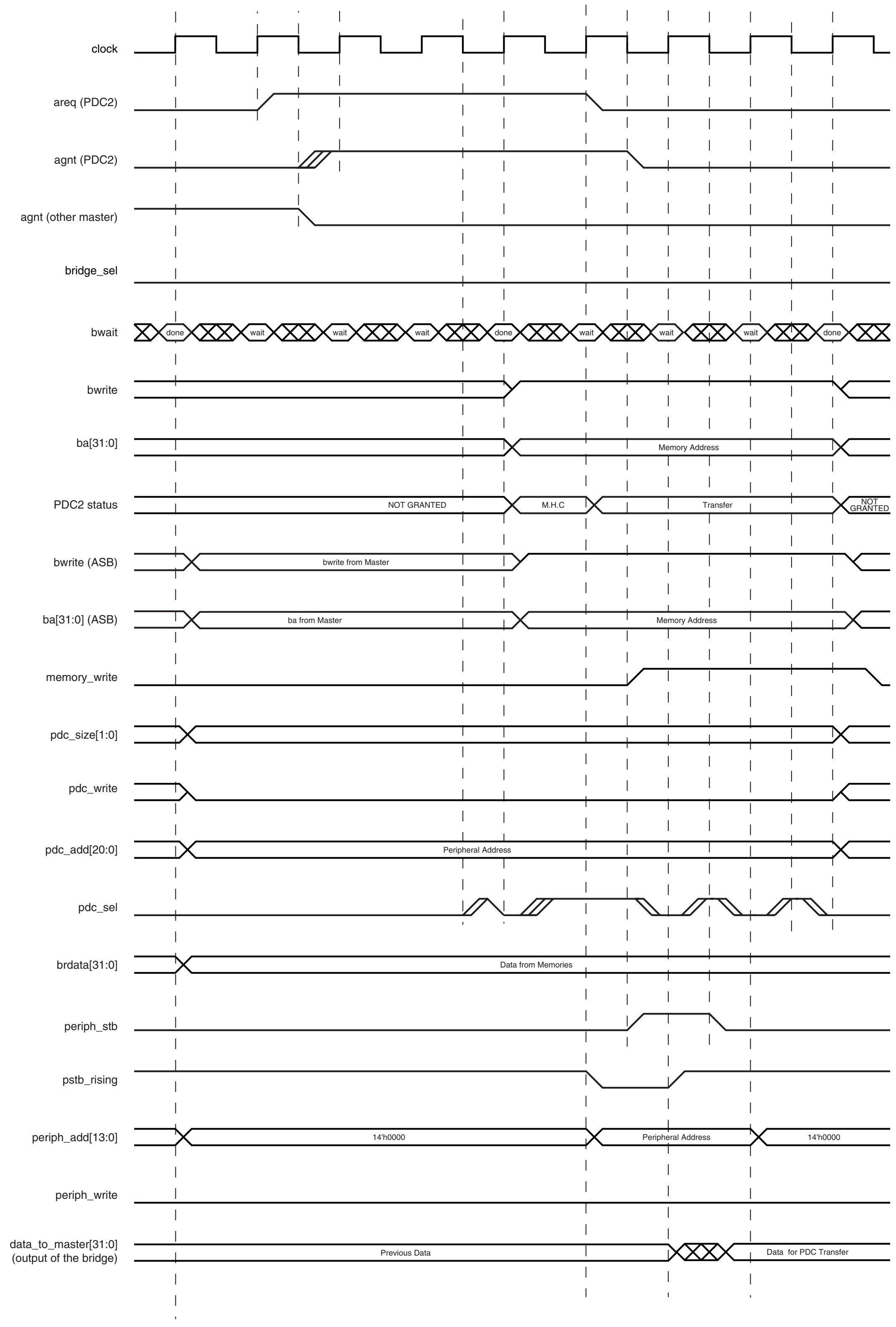


Figure 8. APB to ASB Transfer with Three Wait States Memory



Software Interface

Ten registers make up the peripheral memory map for each of the peripherals. Depending on the peripheral (UART/ USART/SSC/SPI), the offset of these registers is always the same as shown below.

Peripheral User Interface

Table 2. Peripheral Memory Map

Offset	Register	Name	Access	Reset State
0x100	Receive Pointer Register	PERIPH_RPR	Read/Write	0
0x104	Receive Counter Register	PERIPH_RCR	Read/Write	0
0x108	Transmit Pointer Register	PERIPH_TPR	Read/Write	0
0x10C	Transmit Counter Register	PERIPH_TCR	Read/Write	0
0x110	Receive Next Pointer Register	PERIPH_RNPR	Read/Write	0
0x114	Receive Next Counter Register	PERIPH_RNCR	Read/Write	0
0x118	Transmit Next Pointer Register	PERIPH_TNPR	Read/Write	0
0x11C	Transmit Next Counter Register	PERIPH_TNCR	Read/Write	0
0x120	PDC2 Transfer Control Register	PERIPH_PTCR	Write	0
0x124	PDC2 Transfer Status Register	PERIPH_PTSR	Read	0

UART/USART/SSC/SPI Receive Pointer Register

Register Name: UART_RPR, USART_RPR, SSC_RPR, SPI_RPR
Access Type: Read/Write

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

- **RXPTR: Receive Pointer Register**
  RXPTR must be loaded with the address of the receive buffer.

UART/USART/SSC/SPI Receive Counter Register

Register Name: UART_RCR, USART_RCR, SSC_RCR, SPI_RCR
Access Type: Read/Write

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

- **RXCTR: Receive Counter Register**
  RXCTR must be loaded with the size of the receive buffer.
  0 = Stop peripheral data transfer to the receiver
  1 - 65535 = Start peripheral data transfer if corresponding periph_px_rdy is active

UART/USART/SSC/SPI Transmit Pointer Register

Register Name: UART_TPR, USART_TPR, SSC_TPR, SPI_TPR

Access Type: Read/Write

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

- **TXPTR: Transmit Counter Register**

  TXPTR must be loaded with the address of the transmit buffer.

UART/USART/SSC/SPI Transmit Counter Register

Register Name: UART_TCR, USART_TCR, SSC_TCR, SPI_TCR

Access Type: Read/Write

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

- **TXCTR: Transmit Counter Register**

  TXCTR must be loaded with the size of the transmit buffer.

  0 = Stop peripheral data transfer to the transmitter

  1- 65535 = Start peripheral data transfer if corresponding periph_tx_rdy is active

UART/USART/SSC/SPI Receive Next Pointer Register

Register Name: UART_RNPR, USART_RNPR, SSC_RNPR, SPI_RNPR
Access Type: Read/Write

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

- **RXNPTR: Receive Next Pointer**
  RXNPTR contains the address of the next buffer to fill with received data when the current one is completed.

UART/USART/SSC/SPI Receive Next Counter Register

Register Name: UART_RNCR, USART_RNCR, SSC_RNCR, SPI_RNCR
Access Type: Read/Write

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

- **RXNCTR: Receive Next Counter**
  RXNCTR contains the next buffer maximum size.

UART/USART/SSC/SPI Transmit Next Pointer Register

Register Name: UART_TNPR, USART_TNPR, SSC_TNPR, SPI_TNPR
Access Type: Read/Write

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

- **TXNPTR: Transmit Next Pointer**
  TXNPTR contains the address of the next buffer from where to read data when the current one is complete.

UART/USART/SSC/SPI Transmit Next Counter Register

Register Name: UART_TNCR, USART_TNCR, SSC_TNCR, SPI_TNCR
Access Type: Read/Write

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

- **TXNCTR: Transmit Counter Next**
  TXNCTR contains the next transmit buffer size.

UART/USART/SSC/SPI PDC2 Transfer Control Register

Register Name: UART_PTCR, USART_PTCR, SSC_PTCR, SPI_PTCR
Access Type: Write

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

- **RXTEN: Receiver Transfer Enable**
  0 = No effect.
  1 = Enables the receiver PDC2 transfer requests if RXTDIS is not set.
- **RXTDIS: Receiver Transfer Disable**
  0 = No effect.
  1 = Disables the receiver PDC2 transfer requests.
- **TXTEN: Transmitter Transfer Enable**
  0 = No effect.
  1 = Enables the transmitter PDC2 transfer requests.
- **TXTDIS: Transmitter Transfer Disable**
  0 = No effect.
  1 = Disables the transmitter PDC2 transfer requests.

UART/USART/SSC/SPI PDC2 Transfer Status Register

Register Name: UART_PTSR, USART_PTSR, SSC_PTSR, SPI_PTSR
Access Type: Read

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

- **RXTEN: Receiver Transfer Enable**
  0 = Receiver PDC2 transfer requests are disabled.
  1 = Receiver PDC2 transfer requests are enabled.
- **TXTEN: Transmitter Transfer Enable**
  0 = Transmitter PDC2 transfer requests are disabled.
  1 = Transmitter PDC2 transfer requests are enabled.

Timing Diagrams

Figure 9. Peripheral Bus Interface

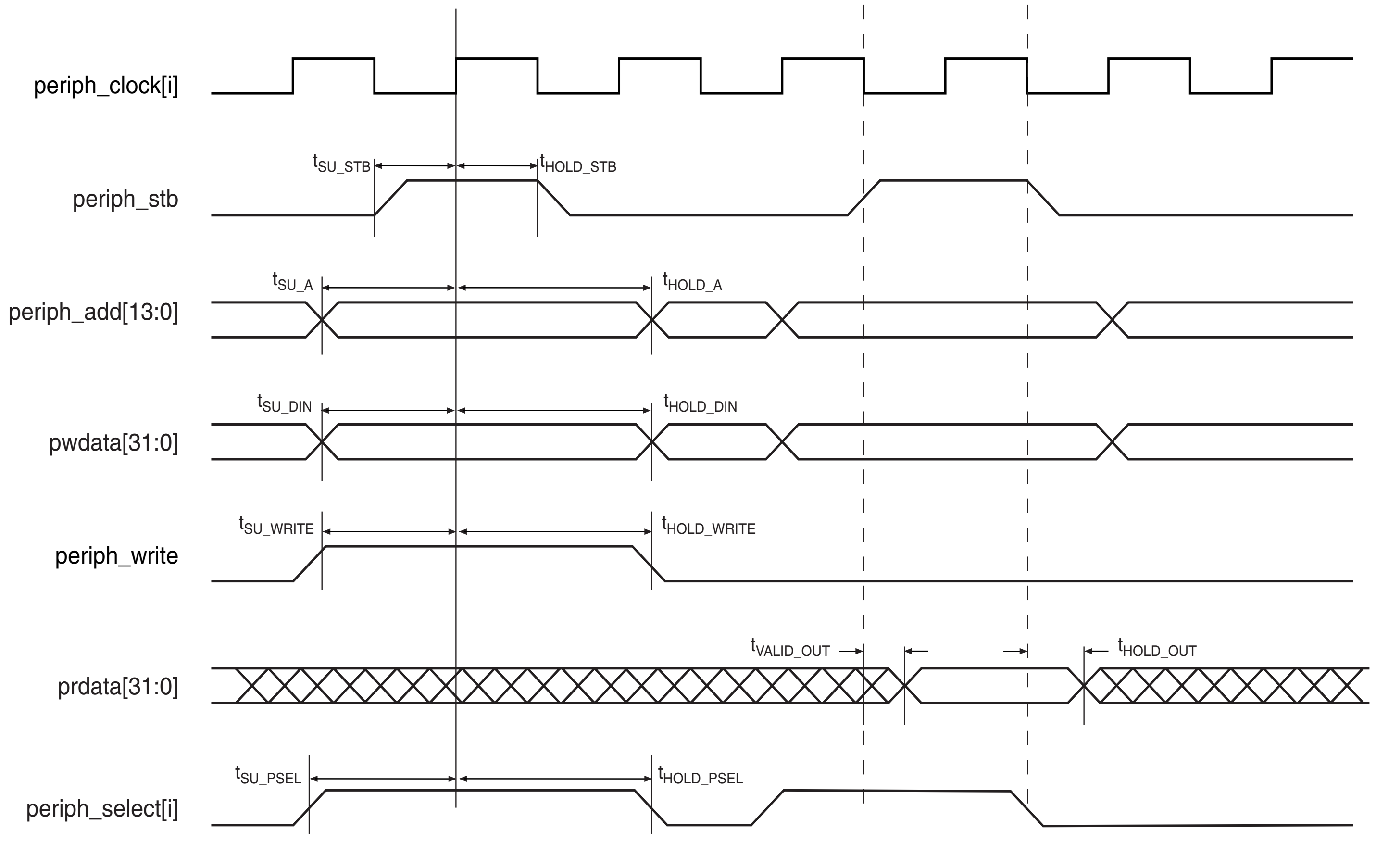


Table 3. Peripheral Bus Interface Parameters

Parameter	Description
t_{SU_STB}	periph_stb setup to rising periph_clocks [i]
t_{HOLD_STB}	periph_stb hold after rising periph_clocks [i]
t_{SU_A}	periph_add setup to rising periph_clocks [i]
t_{HOLD_A}	periph_add hold after rising periph_clocks [i]
t_{SU_DIN}	pwdata setup to rising periph_clocks [i]
t_{HOLD_DIN}	pwdata hold after rising periph_clocks [i]
t_{SU_WRITE}	periph_write setup to rising periph_clocks [i]
t_{HOLD_WRITE}	periph_write hold after rising periph_clocks [i]
t_{SU_PSEL}	periph_select setup to rising periph_clocks [i]
t_{HOLD_PSEL}	periph_select hold after rising periph_clocks [i]
t_{VALID_OUT}	prdata valid after falling periph_clocks[i]
t_{HOLD_OUT}	prdata hold after falling periph_select[i]

Figure 10. Advanced System Bus Dedicated Signals

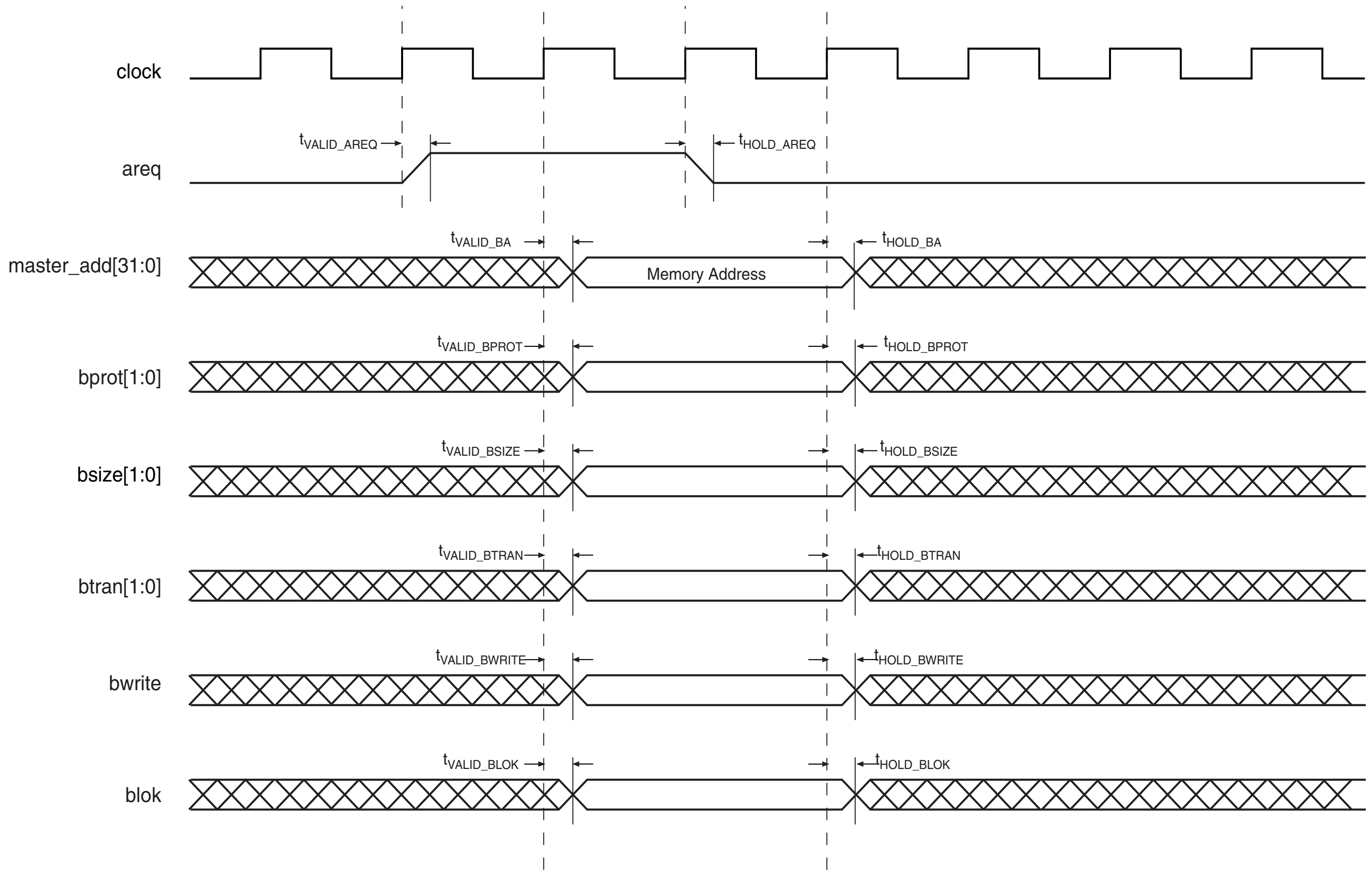
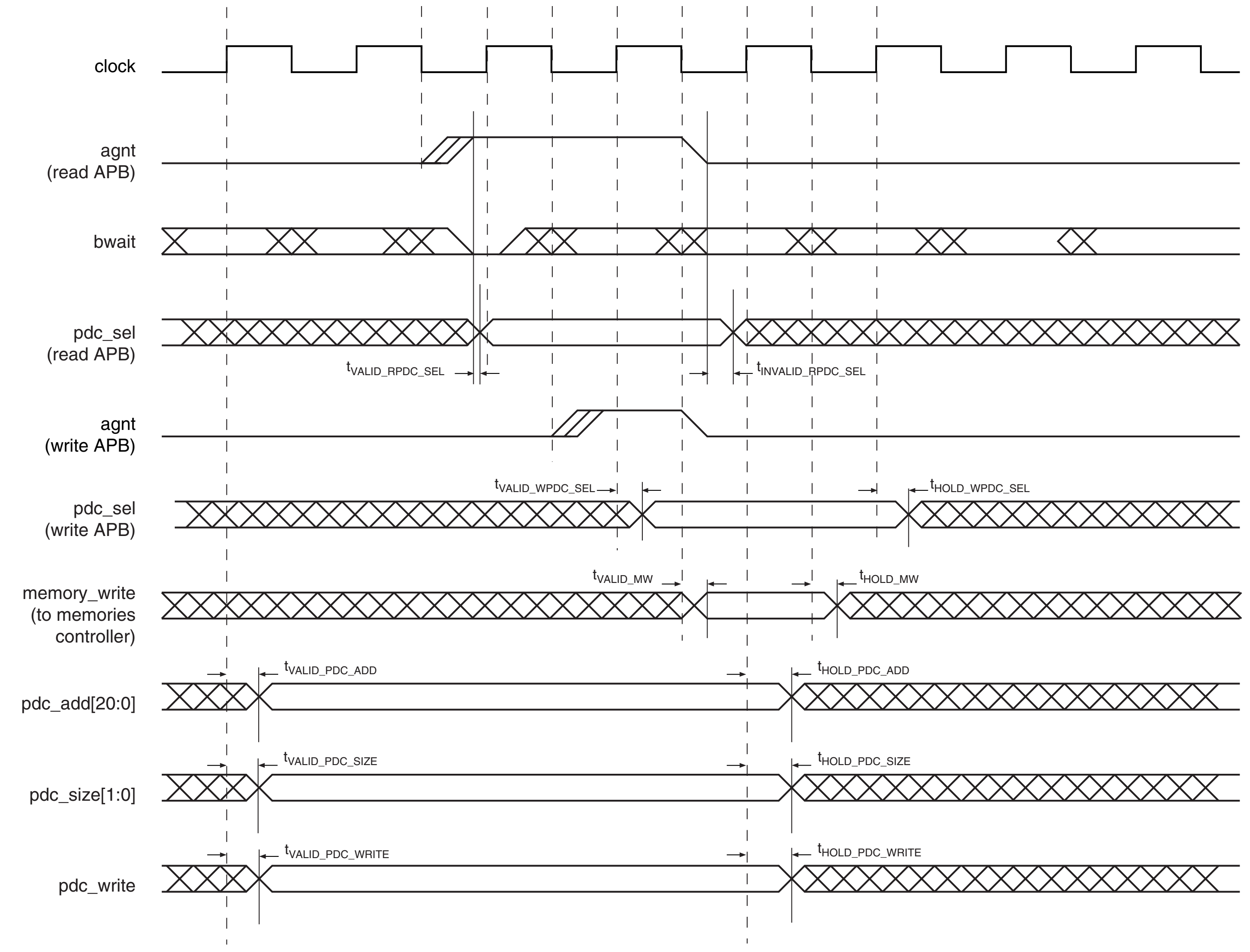


Figure 11. Specific Signals Interfacing with Bridge and EBI



Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex
France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

Rev. 1734A–03/01/0M