

# MPC8260 UPM Timing Diagram

The three user-programmable machine (UPMs) of the MPC8260 PowerQUICC™ II integrated communications processor are flexible interfaces that connect to a wide range of memory devices. At the heart of each UPM is an internal-memory RAM array that specifies the logical value driven on the external memory controller pins for a given clock cycle.

This application note presents a series of timing diagrams for several UPM usage scenarios. All the timing diagrams are based on simulations and are for the 60x bus. Timing for the local bus is basically the same as for the 60x bus, except that there are no  $\overline{\text{PSDVAL}}$  and  $\overline{\text{TA}}$  signals to indicate the termination of the memory cycle. Also, the local bus has its own signals:

- Local address bus
- Local data bus
- $\overline{\text{LBS}}$
- $\overline{\text{LGPLx}}$

Local address pins are multiplexed with PCI signals. To select the local bus function of these pins, configure the HRCW[L2CPC] bits to 00 during configuration or program them to 00 after configuration. Because of the similarity of the timing for the 60x and local buses, refer to the 60x bus timing diagrams for local bus timings.

## Contents

1 UPM Programming .....	2
2 UPM Timings.....	6
3 UPM $\overline{\text{ARTRY}}$ Cycle .....	12
4 UPM Read-Modify-Write Cycle .....	14

# 1 UPM Programming

The basic steps to program the UPMs are as follows:

1. Set up BRx and ORx.
2. Configure MxMR[OP] = 01 for writing to a RAM array.
3. Write patterns to the RAM array by accessing the UPM with a single-byte transaction.
4. Program MPTPR and L/PSRT if refresh is required.
5. Configure MxMR[OP] = 00 for normal operation.

[Example 1](#) shows the detailed assembly code for completing these general steps.

---

## Example 1. Assembly Language

---

```
# Set up OR1
addis r2,r0,0xffff
ori   r2,r2,0x0820
addis r1,r0,0x0f01
ori   r1,r1,0x010c
stw   r2, 0x0000(r1)

# Set up BR1
addis r2,r0,0x0100
ori   r2,r2,0x1881
addis r1,r0,0x0f01
ori   r1,r1,0x0108
stw   r2, 0x0000(r1)

# MAMR OP = 01 for write RAM code starting at address 00 (READ Routine)
addis r2,r0,0x1000
ori   r2,r2,0x8000
addis r1,r0,0x0f01
ori   r1,r1,0x0170
stw   r2, 0x0000(r1)

# MDR
addis r2,r0,0x08ea
ori   r2,r2,0xa800
addis r1,r0,0x0f01
```

```
ori    r1,r1,0x0188
stw    r2, 0x0000(r1)
# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 00
addis  r2,r0,0x0100
ori    r2,r2,0x0000
stb    r1, 0x0008(r2)

# MDR
addis  r2,r0,0x00a0
ori    r2,r2,0xa800
addis  r1,r0,0x0f01
ori    r1,r1,0x0188
stw    r2, 0x0000(r1)

# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 01
addis  r2,r0,0x0100
ori    r2,r2,0x0000
stb    r1, 0x0008(r2)

# MDR
addis  r2,r0,0x00a0
ori    r2,r2,0xa800
addis  r1,r0,0x0f01
ori    r1,r1,0x0188
stw    r2, 0x0000(r1)

# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 02
addis  r2,r0,0x0100
ori    r2,r2,0x0000
stb    r1, 0x0008(r2)

# MDR
addis  r2,r0,0x00a0
ori    r2,r2,0xa800
addis  r1,r0,0x0f01
ori    r1,r1,0x0188
```

## UPM Programming

```
stw    r2, 0x0000(r1)
# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 03
addis  r2,r0,0x0100
ori    r2,r2,0x0000
stb    r1, 0x0008(r2)

# MDR
addis  r2,r0,0x00a0
ori    r2,r2,0xa800
addis  r1,r0,0x0f01
ori    r1,r1,0x0188
stw    r2, 0x0000(r1)

# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 04
addis  r2,r0,0x0100
ori    r2,r2,0x0000
stb    r1, 0x0008(r2)

# MDR
addis  r2,r0,0x01b5
ori    r2,r2,0x4405
addis  r1,r0,0x0f01
ori    r1,r1,0x0188
stw    r2, 0x0000(r1)

# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 05
addis  r2,r0,0x0100
ori    r2,r2,0x0000
stb    r1, 0x0008(r2)

# MAMR OP = 01 for write RAM code starting at address 18 (WRITE Routine)
addis  r2,r0,0x1000
ori    r2,r2,0x8018
addis  r1,r0,0x0f01
ori    r1,r1,0x0170
stw    r2, 0x0000(r1)
```

```
# MDR
addis r2,r0,0x8000
ori   r2,r2,0xa800
addis r1,r0,0x0f01
ori   r1,r1,0x0188
stw   r2, 0x0000(r1)

# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 18
addis r2,r0,0x0100
ori   r2,r2,0x0000
stb   r1, 0x0008(r2)

# MDR
addis r2,r0,0x1000
ori   r2,r2,0x0005
addis r1,r0,0x0f01
ori   r1,r1,0x0188
stw   r2, 0x0000(r1)

# Single byte hit (stb) of 0x0100_0008 to write MDR to RAM array 19
addis r2,r0,0x0100
ori   r2,r2,0x0000
stb   r1, 0x0008(r2)

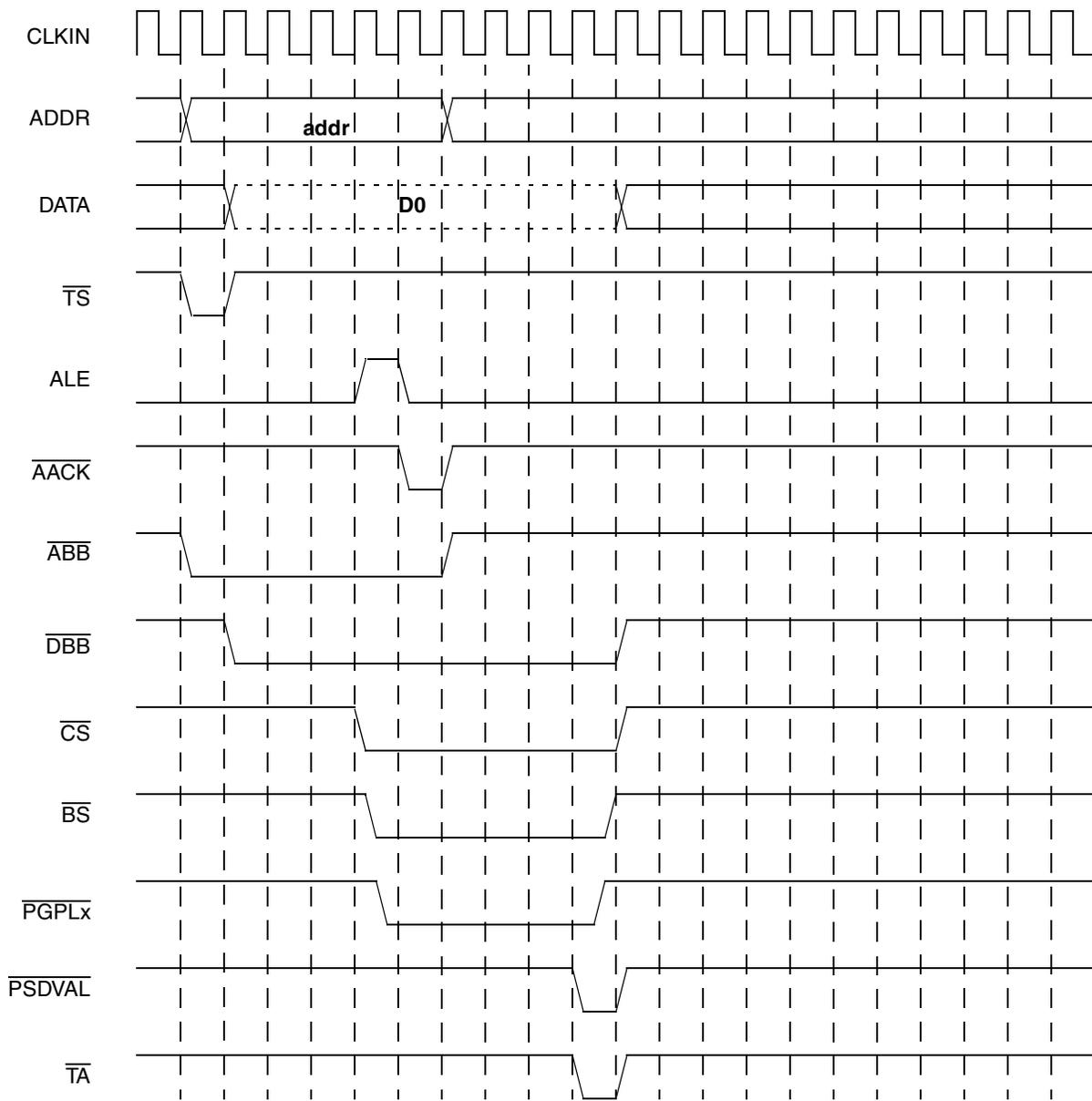
# MAMR OP = 00 for normal operation
addis r2,r0,0x0000
ori   r2,r2,0x0000
addis r1,r0,0x0f01
ori   r1,r1,0x0170
stw   r2, 0x0000(r1)

#Access to memory controlled by UPM
addis r2,r0,0x0100
ori   r2,r2,0x0000

lwz   r3, 0x0018(r2)
stw   r1, 0x0018(r2)
```

## 2 UPM Timings

Figure 1 shows a 32-bit single beat read/write on a 32-bit port.



Notes:

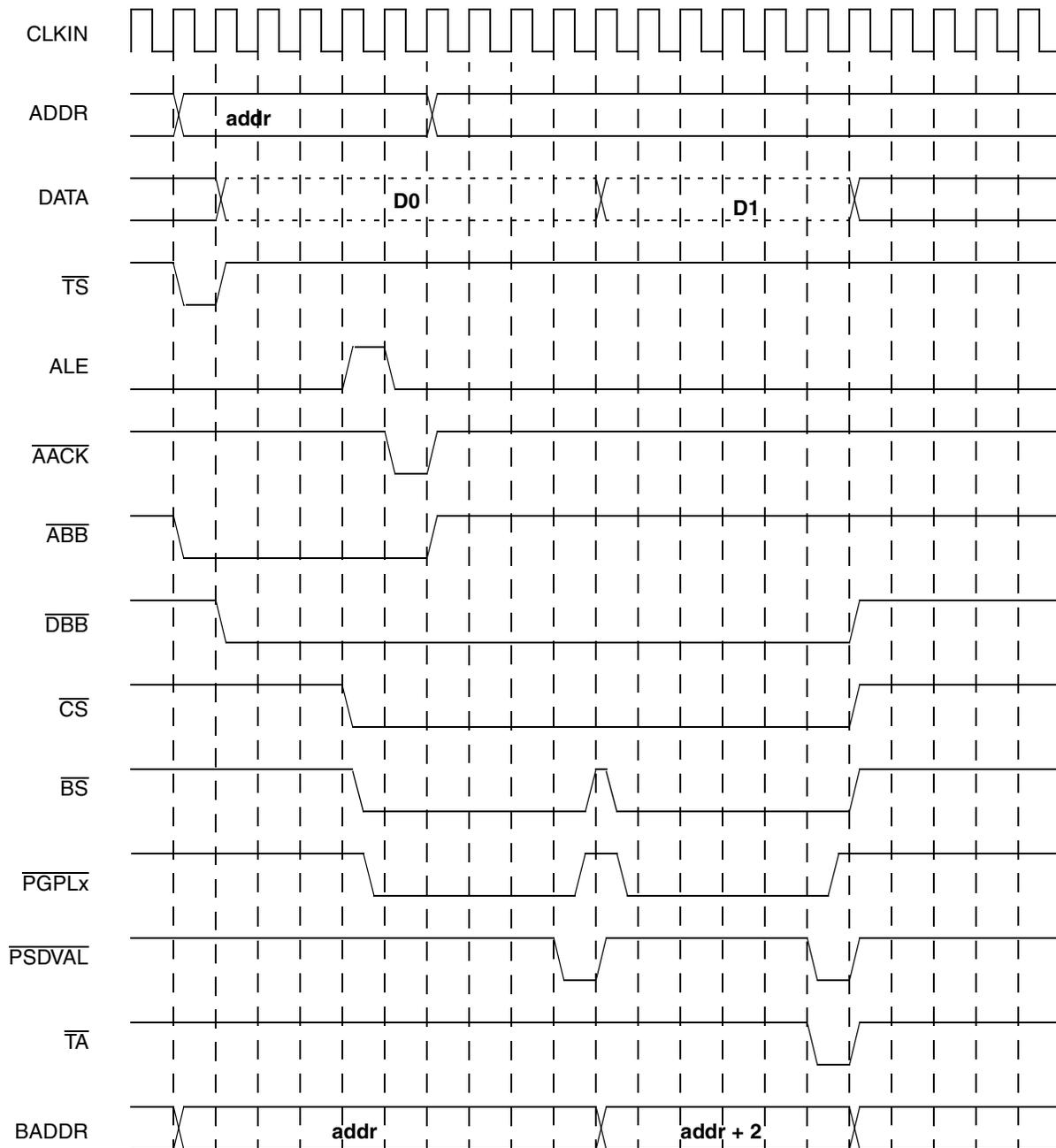
<sup>1</sup> RAM word for a single-beat read (a single-beat write starts at RAM address 0x18)

00: 08ea a800	03: 00a0 0000
01: 00a0 0000	04: 00a0 0000
02: 00a0 0000	05: 01b5 5405

<sup>2</sup> Data bus end at PSDVAL for both a single-beat read and write. For a write, the data bus starts when  $\overline{DBB}$  is asserted. For a read, the starting-point depends on the RAM pattern and the memory UPM controls.

**Figure 1. 32-Bit Single Beat Read/Write on 32-Bit Port**

Figure 2 shows a 32-bit single beat read/write on a 16-bit port.

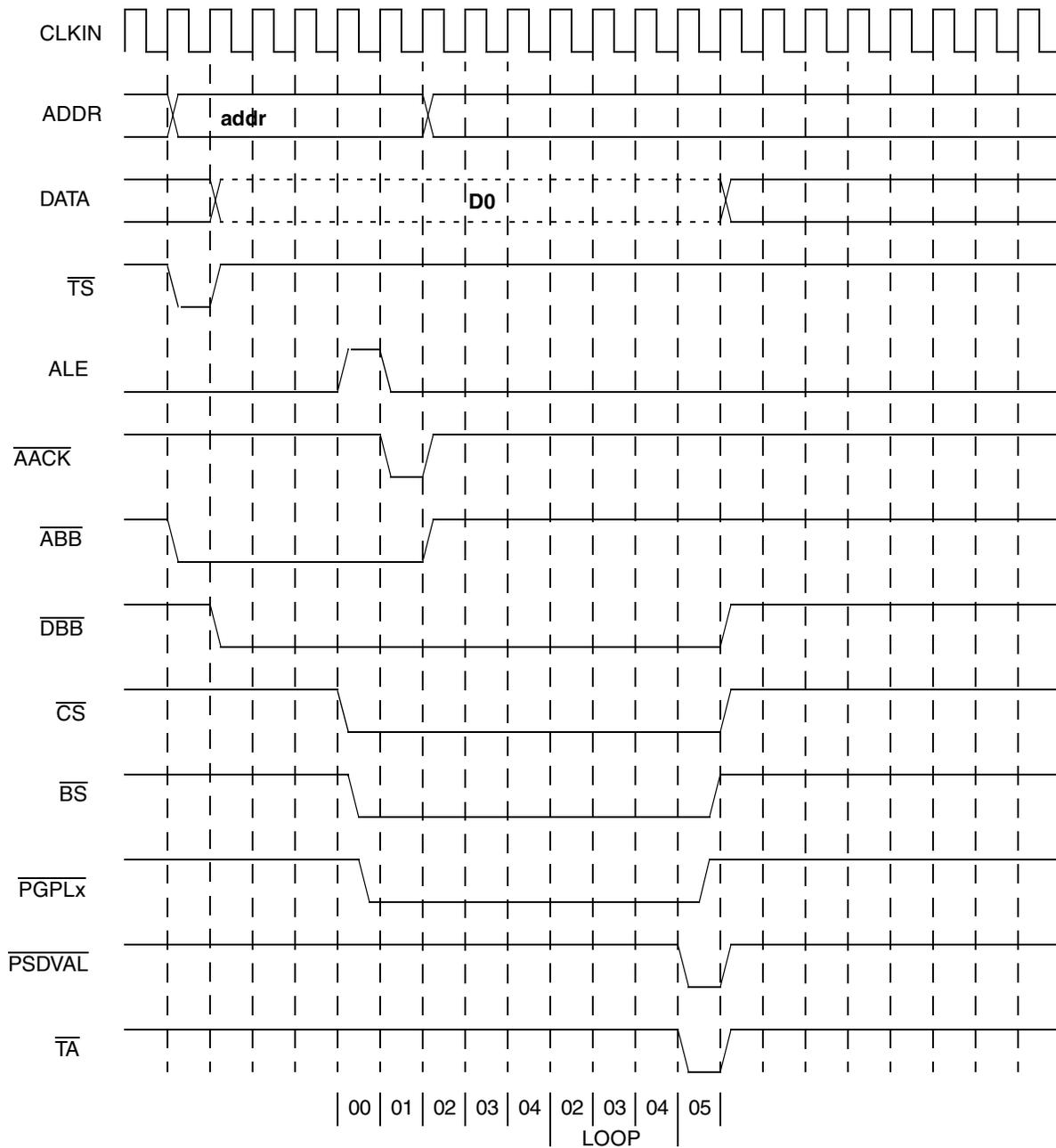


Notes:

- 1 RAM code is the same as for Figure 1.
- 2 The single-beat 32-bit access is split to two 16-bit back-to-back accesses due to port size limitations.

**Figure 2. 32-Bit Single Beat Read/Write on 16-Bit Port**

Figure 3 shows a 32-bit single-beat read/write on a 32-bit port.



Note:

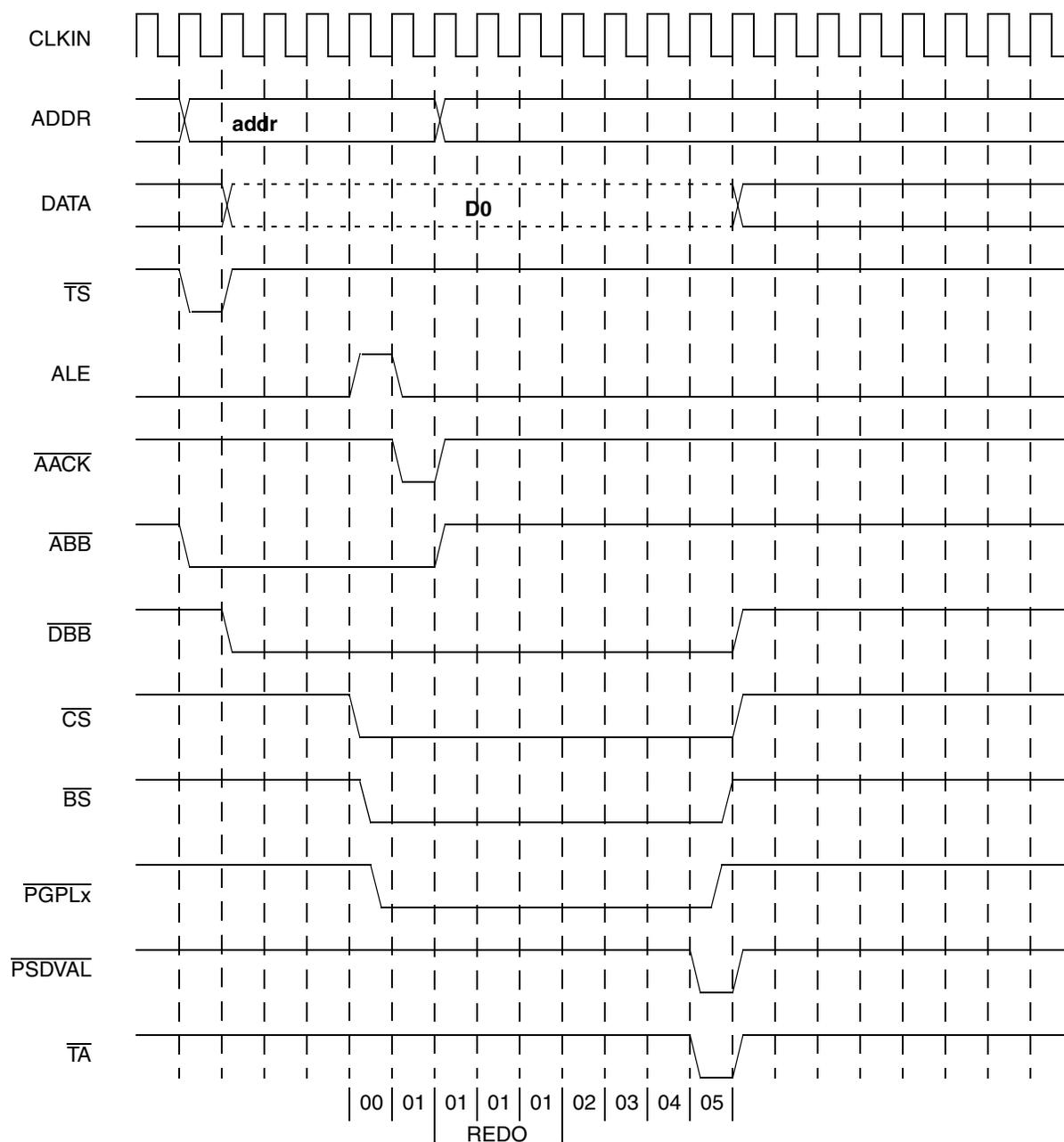
<sup>1</sup> RAM word for single beat read with LOOP:

00: 08ea a800	03: 00a0 0000
01: 00a0 0000	04: 00a0 0080 //LOOP=1
02: 00a0 0080 //LOOP=1	05: 01b5 5405

<sup>2</sup> Loop number is set at MxMR. RLFx for read, WLFx for write. Loop number is one for the above figure.

**Figure 3. UPM with LOOP, 32-Bit Single Beat Read/Write on 32-Bit Port**

Figure 4 shows the UPM with REDO.



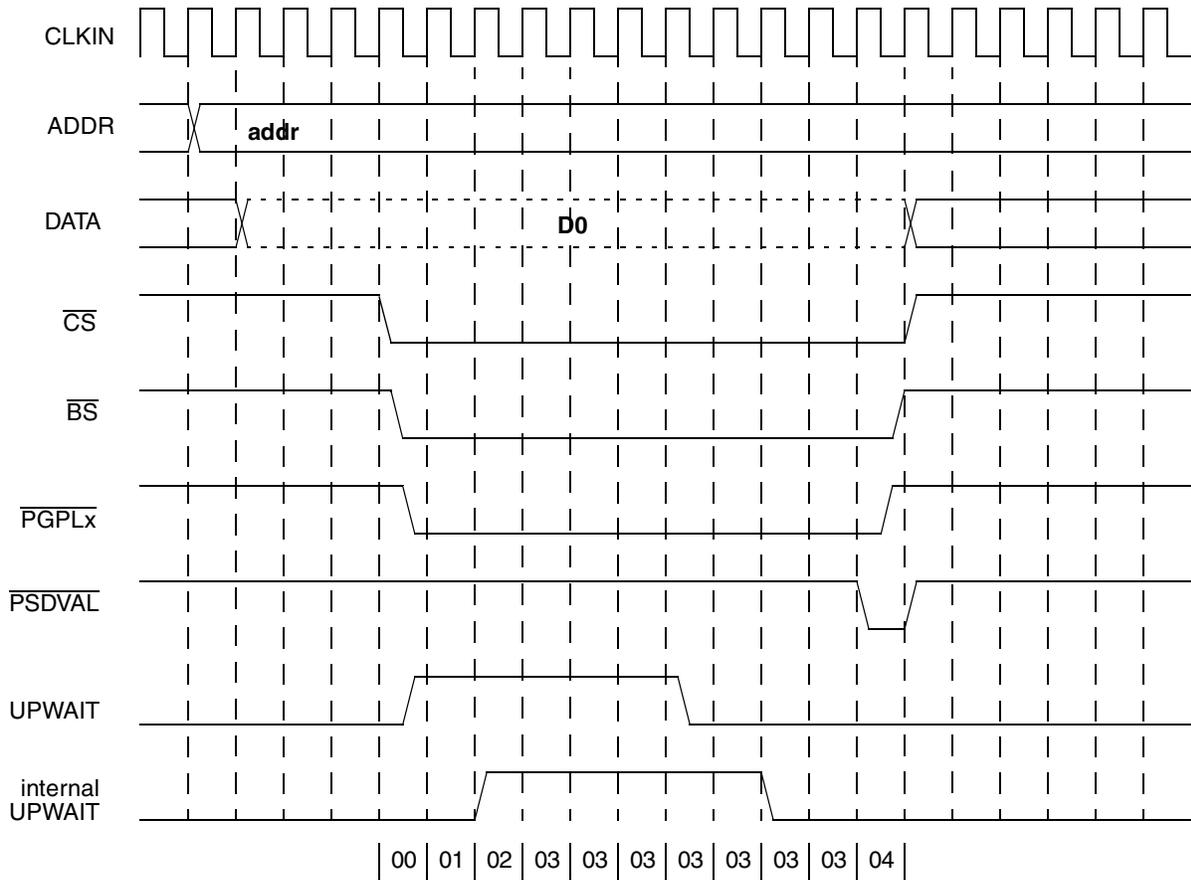
Note:

<sup>1</sup> The RAM word for a single-beat read with REDO:

00: 08ea a800	03: 00a0 0000
01: 00a0 0300 // REDO=3	04: 00a0 0000
02: 00a0 0000	05: 01b5 5405

Figure 4. UPM with REDO

Figure 5 shows the UPM with two consecutive WAEN.



Note:

<sup>1</sup> Minimum RAM words for UPWAIT to take effect:

00: 08ea a800

01: 00a0 0000

02: 00a0 1000 // WAEN=1

03: 00a0 1000 // WAEN=1

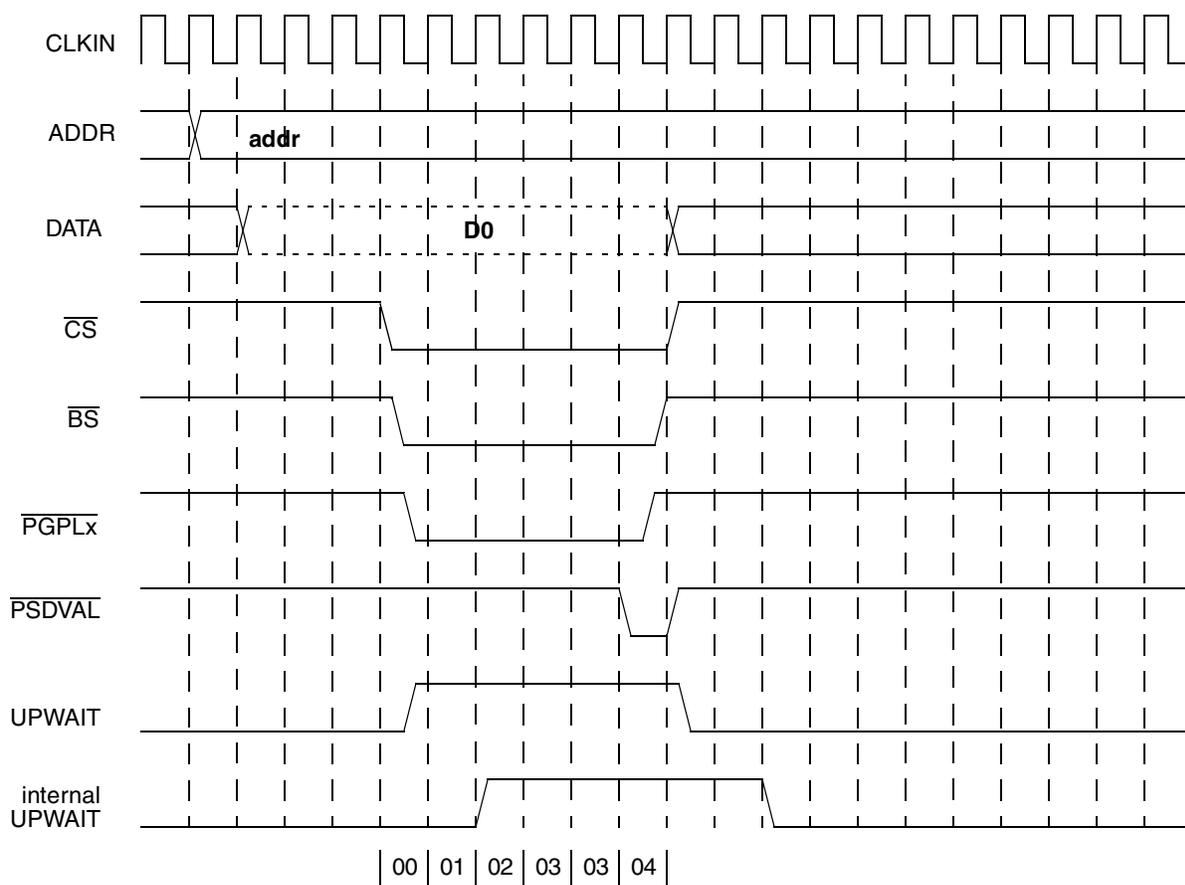
04: 01b5 4405

<sup>2</sup> WAEN needs to be set in two consecutive RAM words. The UPM waits at the second RAM word. Figure 6 shows what happens if only one WAEN is set.

<sup>3</sup> Due to synchronization of UPWAIT, the internal UPWAIT is two cycles later than external UPWAIT. If the external UPWAIT goes active half a cycle after CS is active, the first WAEN that has an effect is at RAM address 02 for a read. Even if WAEN values of 00 and 01 are configured, the UPM does not wait because UPWAIT has not arrived yet due to delay.

Figure 5. UPM with Two Consecutive WAEN

Figure 6 shows the UPM with UPWAIT and one WAEN.



Note:

<sup>1</sup> RAM words with one WAEN

00: 08ea a800

01: 00a0 0000

02: 00a0 1000 //WAEN=1

03: 00a0 0000

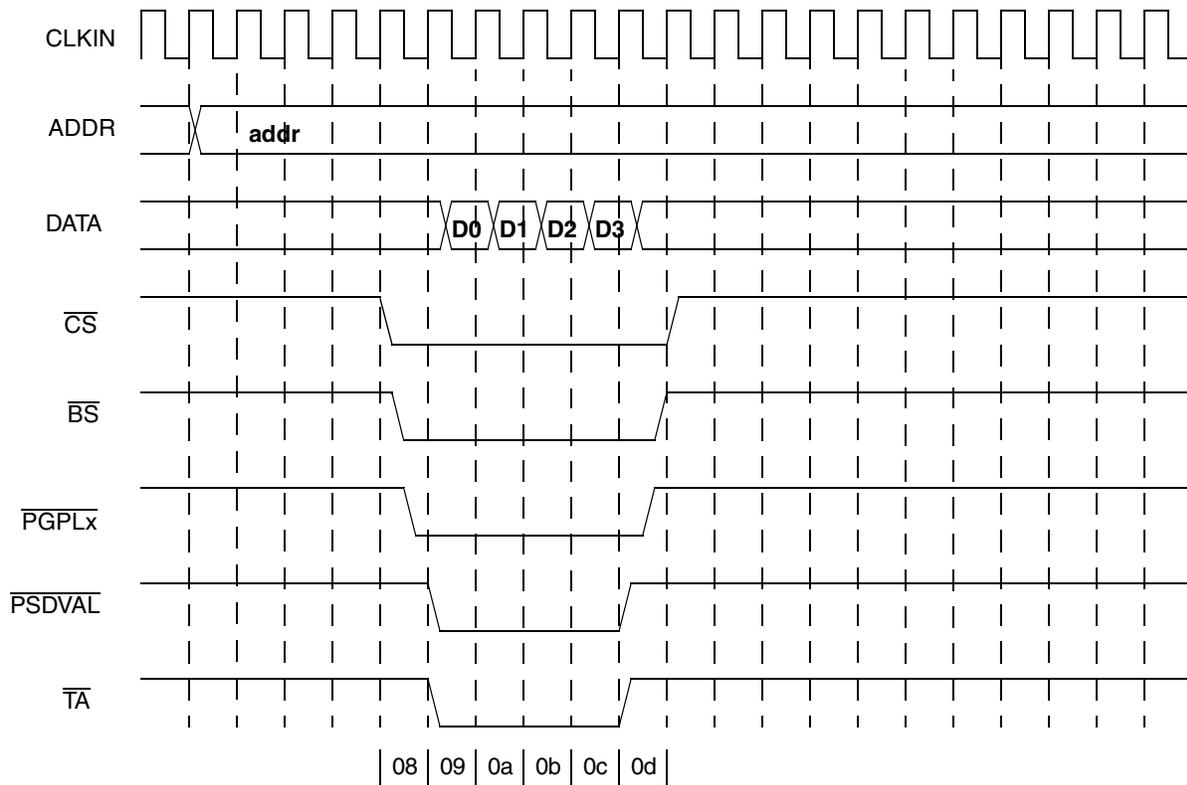
04: 01b5 4405

<sup>2</sup> Setting WAEN at RAM address 02 causes a single-cycle wait at pattern 03.

If UPWAIT is negated, of course the UPM does not wait. However, even if UPWAIT is asserted, a single WAEN causes a wait of only one cycle for next the RAM pattern. Then it proceeds while ignoring the UPWAIT. That is why two consecutive WAENs are needed for a continuous UPWAIT.

Figure 6. UPM with UPWAIT, One WAEN

Figure 7 shows the UPM burst read/write on a 64-bit port.



Note:

<sup>1</sup> Example RAM words for burst read:

08: 08ea a800

09: 00a0 0004 // UTA=1

0a: 00a0 0004 // UTA=1

0b: 00a0 0004 // UTA=1

0c: 00a0 0004 // UTA=1

0d: 01b5 5401

<sup>2</sup> A burst write starts at RAM address 0x20

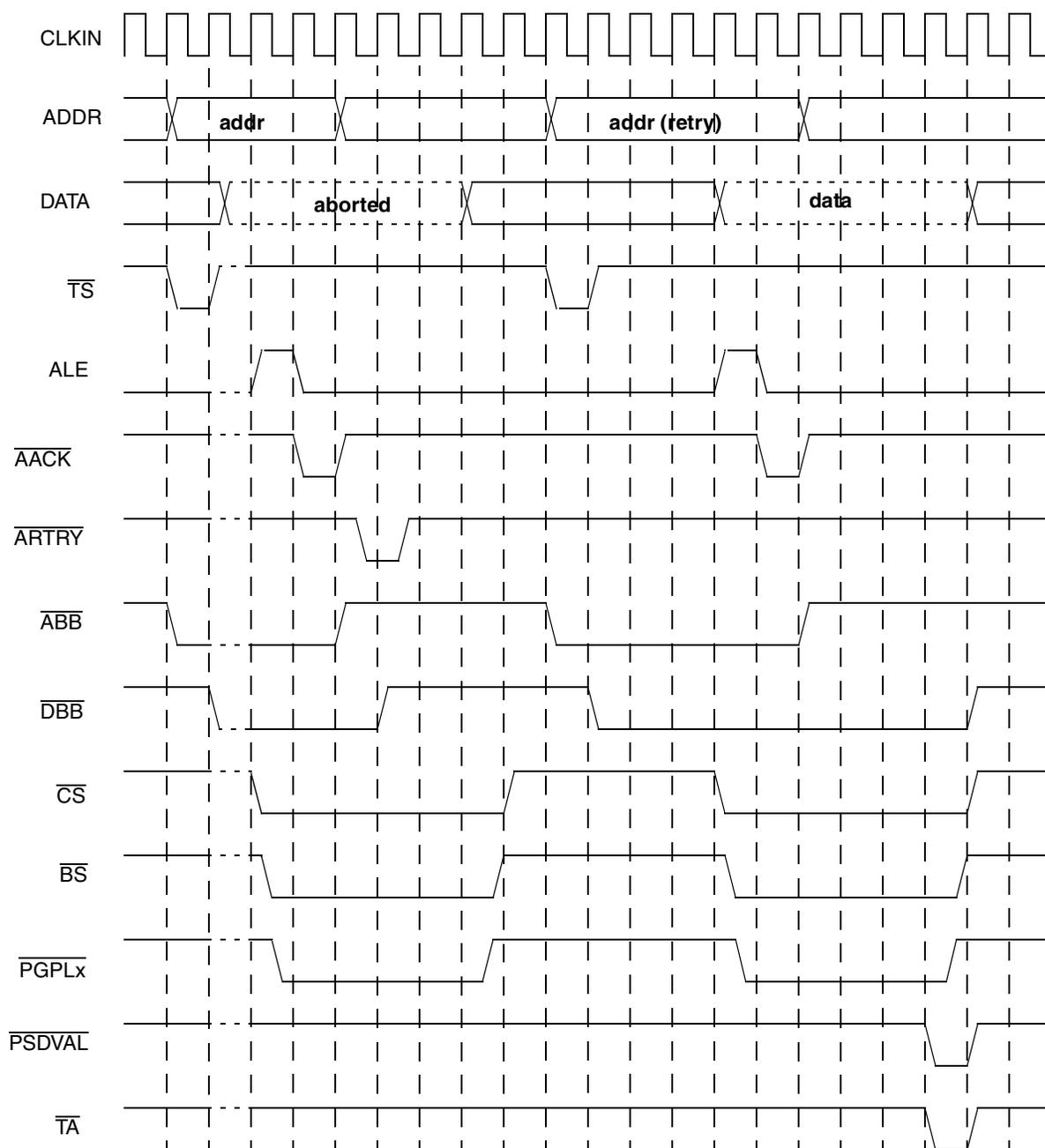
<sup>3</sup> Care should be taken to program the correct number of UTAs in the burst code so that the UPM generates the required number of PSDAL signals for a burst.

Figure 7. UPM Burst Read/Write on 64-Bit Port

### 3 UPM ARTRY Cycle

In 60x-compatible mode, the address transfer can be terminated with the requirement to retry if  $\overline{\text{ARTRY}}$  is asserted during the address tenure and through the cycle following  $\overline{\text{AACK}}$ . The assertion causes the entire transaction (address and data tenure) to be rerun.

Figure 8 shows the UPM  $\overline{\text{ARTRY}}$  cycle.



Note:

<sup>1</sup> RAM word for  $\overline{\text{ARTRY}}$  cycle:

00: 08ea a800	03: 00a0 0000
01: 00a0 0000	04: 00a0 0000
02: 00a0 0000	05: 01b5 5405

<sup>2</sup> When a valid  $\overline{\text{ARTRY}}$  is recognized, the UPM finishes running the current RAM pattern without asserting  $\overline{\text{PSDVAL}}$  and  $\overline{\text{TA}}$ . After that, a retry phase starts.

Figure 8. UPM  $\overline{\text{ARTRY}}$  Cycle

## 4 UPM Read-Modify-Write Cycle

If the UPM is programmed for read-modify-write parity checking or ECC correction and checking, every write access to memory that is less than the port size automatically causes a read-modify-write cycle.

The following is an example of RAM code:

Read Pattern:

00: 08ea a800

01: 00a0 0000

02: 00a0 0000

03: 00a0 0000

04: 00a0 0000

05: 01b5 5405

Write Pattern:

18: 0800 0000

19: 0000 0005

Figure 9 shows a 32-bit write to a 64-bit port to trigger the read-modify-write cycle.

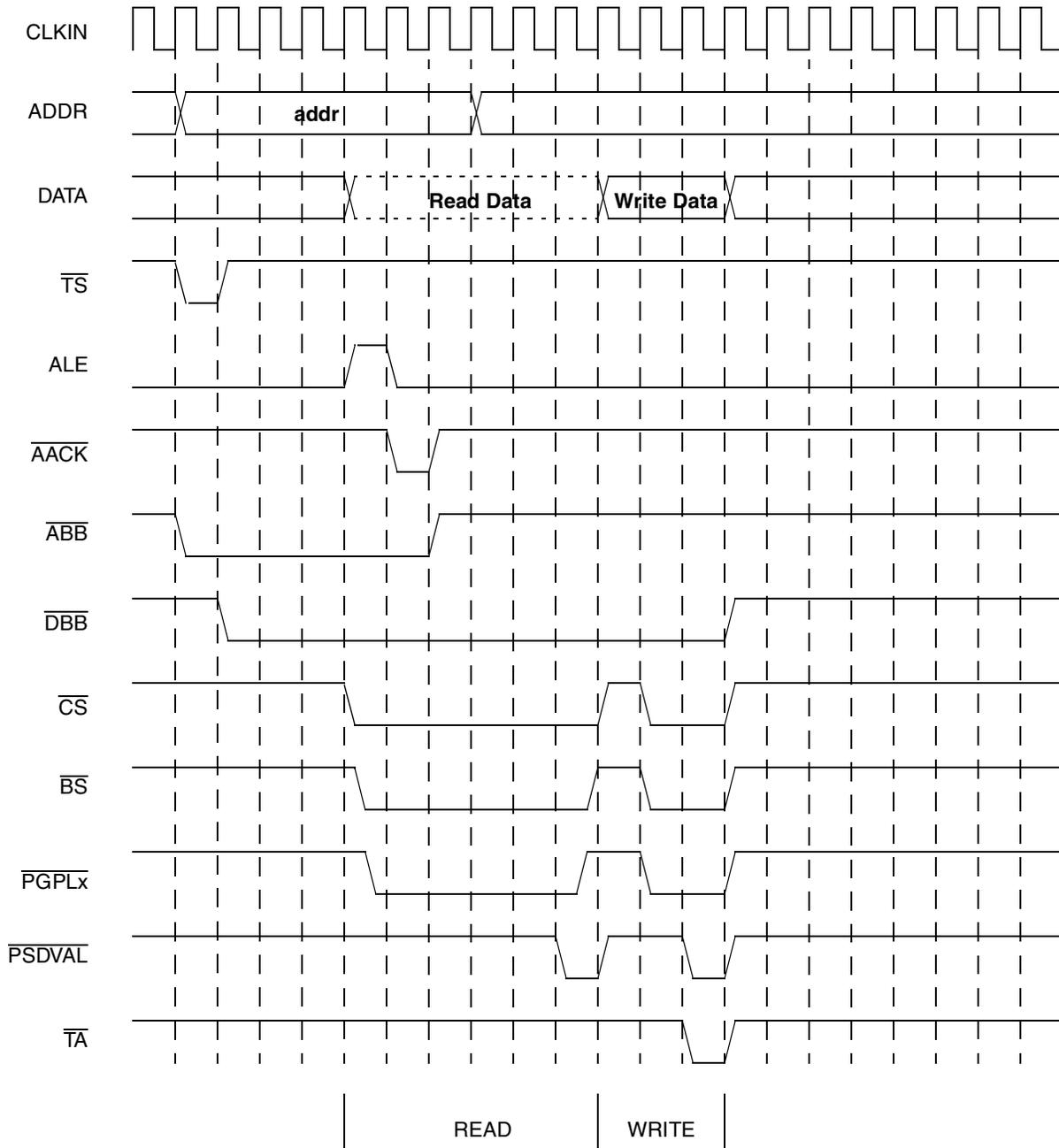


Figure 9. UPM with Read-Modify-Write Cycle

## **How to Reach Us:**

### **Home Page:**

www.freescale.com

### **email:**

support@freescale.com

### **USA/Europe or Locations Not Listed:**

Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
1-800-521-6274  
480-768-2130  
support@freescale.com

### **Europe, Middle East, and Africa:**

Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
support@freescale.com

### **Japan:**

Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku  
Tokyo 153-0064, Japan  
0120 191014  
+81 3 5437 9125  
support.japan@freescale.com

### **Asia/Pacific:**

Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate,  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
support.asia@freescale.com

### **For Literature Requests Only:**

Freescale Semiconductor  
Literature Distribution Center  
P.O. Box 5405  
Denver, Colorado 80217  
1-800-441-2447  
303-675-2140  
Fax: 303-675-2150  
LDCForFreescaleSemiconductor  
@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. The PowerPC name is a trademark of IBM Corp. and is used under license. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc., 1999, 2006.

Document Number: AN2179

Rev. 2

07/2006

