1.10

VIC068A Block Transfer Functions

The ability to transfer large blocks of data at a high-sustained transfer rate is paramount in today's VMEbus market. When implemented properly, transfer rates exceeding 30 Mbyte/sec can be obtained using a high-speed processor, high-speed memory and high-speed VMEbus interfaces such as the VIC068A.

1.10.1 VIC068A Master Block Transfer

The VIC068A can be configured for block transfers while the local processor is bus master. This is referred to as a *MOVEM* block transfer. The term MOVEM is derived from the Motorola 68K MOVEM (move multiple registers) instruction. The VIC068A can also become the local bus master and implement block transfers using DMA on the local side. This is referred to as *block transfers with local DMA*. For master VMEbus block transfers with local DMA, the VIC068A contains two 8-bit address counters that can automatically increment the address for both the local or CPU side and the VMEbus side.

The VMEbus specification prohibits the crossing of 256-byte boundaries during block transfers without giving up the VMEbus or toggling the VMEbus AS*. The VIC068A allows, with external logic (such as CY7C964s), implementing block transfers that exceed the 256-byte limit. The VIC068A is able to give up the bus at the 256-byte limit (or any limit), then re-arbitrate for the bus at a programmed time later. The time between sub-block transfers is called the *interleave period*. The number of VMEbus cycles between interleave periods is called the *burst length*. The total number of bytes to be transferred is called the *block transfer length*.

The VIC068A also allows for single-cycle VMEbus cycles to be performed during the interleave period. This feature is called the *dual-path* feature and requires external logic such as either the VAC068A or the CY7C964. As the name implies, the dual-path feature requires that a dual address path exist so that the block transfer address is not lost if a local interleave cycle is performed. Slave cycles can be interleaved between master block transfer bursts without external logic or programming.

The VIC068A only supports D32 and D16 block transfers. D8 block transfers are not supported.

VIC068A registers relevant to master block transfers are as follows:

- Block Transfer Control Register (BTCR)
- Block Transfer Definition Register (BTDR)
- Release Control Register (RCR)
- Block Transfer Length Registers (BTLRs)
- Local Bus Timing Register (LBTR)
- Slave Select 0 Control Register 1 (SS0CR1)
- DMA Status Register (DMASR)
- DMA Status Interrupt Control Register (DMASICR)

It is important to note that the BTLRs contain the number of bytes to be transferred, and the burst length in the RCR contains the number of VMEbus cycles, independent of the number of bytes per cycle.

1.10.1.1 Block Transfers with Local DMA

The block transfer with local DMA is a block transfer in which the VIC068A becomes not only the VMEbus master but the local bus master as well. Upon becoming the local bus master, the VIC068A accesses memory in a DMA-like fashion. As in any DMA operation, this increases the throughput of a system by making memory accesses memory-speed dependent, as opposed to processor-speed dependent. As in any block transfer, the VMEbus slave needs to be enabled for block transfers.

After the VIC068A registers are initialized, a VMEbus write must be performed to the VMEbus destination address with the local starting address as the data. This is referred to as a pseudo write cycle (since an actual VMEbus write is not performed) or the BLT initialization cycle. This pseudo cycle must be a VMEbus write. A read cycle would not place the correct data on the local data bus. *Figure 1–9* shows the BLT initiation cycle (pseudo cycle). This pseudo cycle only starts the block transfer mechanisms and loads the addresses. Since any assertion of MWB* after BTCR[6] is set is interpreted as the pseudo cycle, it is important that no normal VMEbus read/writes be performed until after the completion of the block transfer, or during the next interleave period if the dual-path feature (described later) is enabled. It also may be necessary to disable interrupts or have an interrupt service routine that saves the block transfer registers.

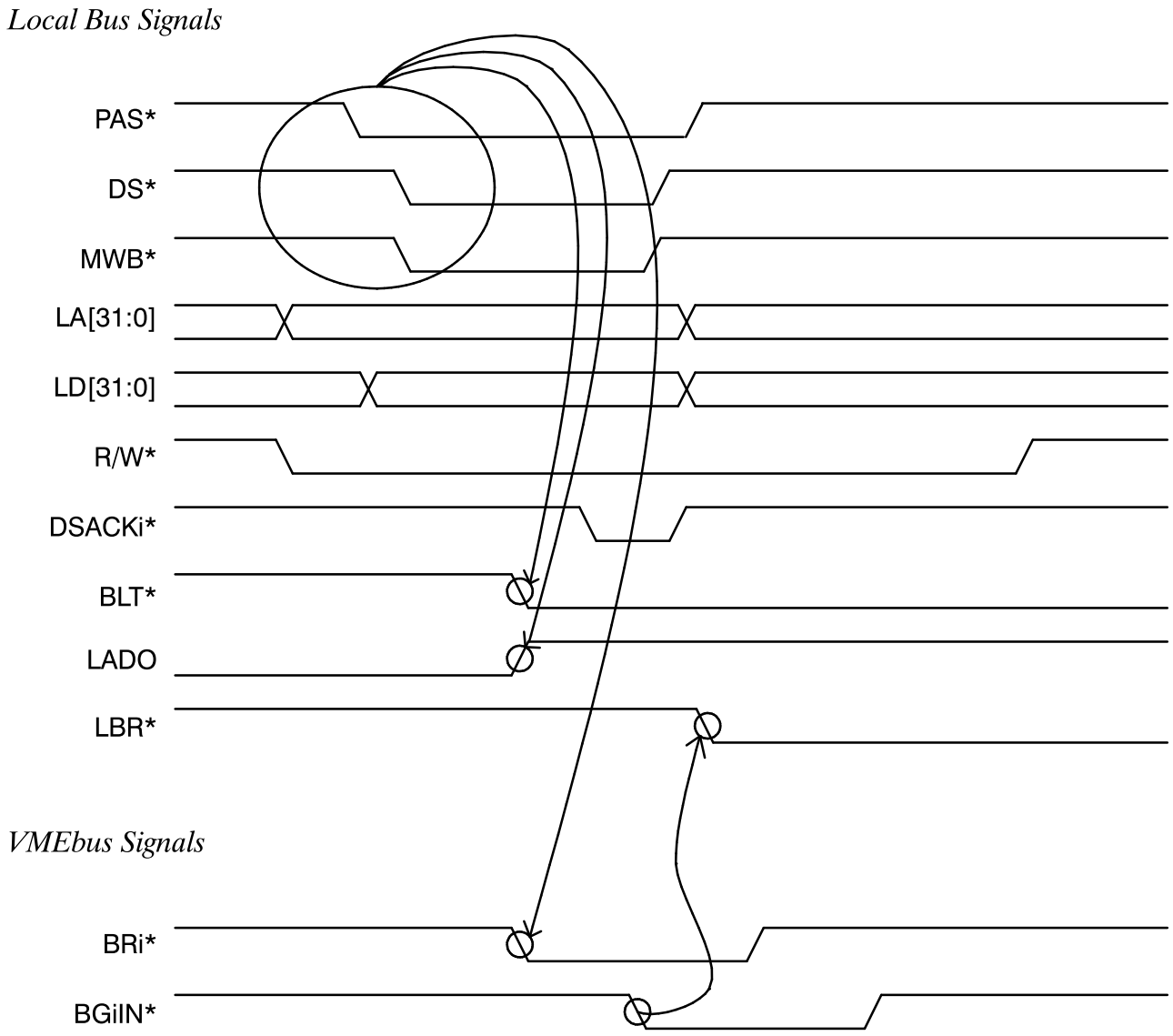

Figure 1−9. Master Block Transfer with Local DMA Initiation Cycle (pseudo cycle)

During the pseudo write, the VIC068A loads the values on the LA[7:0] into its internal VMEbus address counter and asserts LADO to signal external latches to load the remainder of the LA[+:8] as the VMEbus address. The VIC068A at this time also loads the values of LD[7:0] into internal local address counters and asserts BLT* to latch the remainder of the LD[+:8] as the local address. The VIC068A then asserts the DSACKi* signals to terminate the local cycle, and requests the VMEbus. After the VMEbus is granted, the local bus is requested by asserting LBR*. After the local bus is granted, the VIC068A drives the local DMA address onto the local address bus. CY7C964s, or external logic, should decode BLT*, LBG*, PAS*, LAEN, and the FCi signals to drive the upper portion of the address

lines. See section 1.10.1.1.8.1.3 for more details. The VIC068A then accesses the local data by asserting the local address and data strobes. The local resource then acknowledges the VIC068A that local data has been read or written by asserting the DSACKi* signals. Local data is held in the interface while the local address is incremented.

1.10.1.1.1 DMA Burst Length

Recall that the VIC068A is able to divide block transfers into a programmable number of bursts. The length of a burst is programmable from 1 to 64 cycles through writing RCR[5:0]. Clearing these bits (the default value) implies a burst length of 64, not 0. The burst count is independent of the number of bytes transferred per cycle.

1.10.1.1.2 Block Transfer Length

The total length of a block transfer is programmed separately from the burst length. The length is programmed by writing the BTLRs. The LSB of the number is programmed into register BTLR1. Since the VIC068A does not support D8 block transfers, if BTLR0[0] is set, the BTLRs are ignored and only a single burst, as defined in RCR[5:0], will be performed.

1.10.1.1.3 256-Byte Boundary Crossing on the VMEbus

The VMEbus specification prohibits the crossing of 256-byte address boundaries. It is possible, when transferring large amounts of data, to simply deassert the AS* at the boundary crossing and reassert it after the address has incremented without releasing the VMEbus. In this case, BBSY* is held Low so that VMEbus mastership is not lost during the toggle of the AS*. Once the boundary has been crossed, the VIC068A releases BBSY* (if configured for RWD) after AS* is reasserted. The VIC068A, when enabled for VMEbus boundary crossing in the BTDR, does this without any user intervention. External circuitry is required to increment any address bits other than the lower 7 bits driven by the VIC068A. LADO, ABEN*, and the FCi function codes may be used to control the loading, holding, and incrementing of external latches and counters. During the VMEbus cycle before the boundary crossing, LADO will toggle. External counters should increment after two edge transitions of LADO. This is because LADO pulses Low if dual-path is not enabled and pulses High if dual-path is enabled. If the VAC068A or CY7C964 is used in conjunction with the VIC068A, no external hardware for boundary crossing is required.

1.10.1.1.4 256-Byte Boundary Crossing on the Local Bus

Just as the VIC068A allows for 256-byte boundary crossing on the VMEbus, similar provisions are made for the local address bus. Local boundary crossing is enabled in the BTDR.

External circuitry is again required to increment any address bits other than the lower 8 bits driven by the VIC068A. BLT* and the FCi function codes may be used to control the loading and incrementing of these upper address bits. During the local cycle before the boundary crossing, the BLT* will toggle. External counters should increment on the falling edge of BLT*. If the VAC068A or CY7C964 is used in conjunction with the VIC068A, no external hardware for boundary crossing is required.

1.10.1.1.5 Interleave Period

The time between bursts is known as the interleave period. The length of the interleave period is configurable in BTCR[3:0]. During the interleave period, slave cycles can be performed. Master cycles are allowed if the dual-path feature is enabled.

1.10.1.1.6 Dual Path

Normally, during the interleave period no VMEbus master cycles are allowed by the VIC068A. All requests for VMEbus master cycles are DEDLKed by the VIC068A. If, however, the VIC068A is configured for dual-path operation, the VIC068A will allow master cycles. If the dual-path option is to be used, external logic such as the VAC068A or CY7C964 is required to maintain the local and VMEbus addresses at the completion of the last burst. If the VIC068A is enabled for local and VMEbus address boundary crossing, the external counters (assuming all 32 bits are handled by counters) may be used. If the VAC068A is used in conjunction with the VIC068A, no external hardware is required for the dual-path option.

All VMEbus interrupt acknowledge cycles are DEDLKed by the VIC068A whether dual-path is enabled or not.

1.10.1.1.7 The Block Transfer with Local DMA Enable Bit

When the BLT enable bit (BTCR[6]) is set, any assertion of MWB* (qualified by PAS* and DS*) will begin the block transfer. Special care must be taken to insure that this bit is set and cleared as close as possible to the actual block transfer. This means that once the block transfer begins the BLT enable bit must be cleared. It is not necessary to wait for the completion of the block transfer before clearing BTCR[6], this bit can be cleared during an interleave period. This is important if retry logic is employed for deadlocked VMEbus transfers. For example, if an attempted VMEbus cycle is DEDLKed during an interleave period when dual-path is not enabled, the processor, such as a 68K processor, may continually retry the cycle waiting for the VIC068A to complete the block transfer. At the completion of the

block transfer, this retry could initiate another block transfer since the BLT enable bit has not been cleared. For this reason, retry logic may have to be disabled during block transfers with local DMA.

1.10.1.1.8 Example Configurations

The following paragraphs further explain some application details regarding the different modes of block transfer operation.

1.10.1.1.8.1 Boundary Crossing Disabled

This is the simplest form of the block transfer with local DMA. In this mode, block transfers are restricted to those that do not cross a 256-byte boundary on either the local bus or the VMEbus. Enabling dual path has no effect since there will not be an interleave. This cycle is initiated by writing the following registers on the master module:

- BTCR, to set up block transfer
- BTDR, to set up block transfer
- BTLRs, to configure length of block transfer
- LBTR, to configure local timing
- SS0CR1, to configure programmable delays
- DMASR, for DMA status
- DMASICR, to enable DMA complete interrupt

To enable the slave module for block transfers, configure the SS0CR0 or SS1CR0 register in the slave's VIC068A as appropriate.

1.10.1.1.8.1.1 Sample 68K Code (Write)

If we assume that the VIC068A registers start at a base location of *vic_reg*, a sample of 68K code that would configure the VIC068A for a block transfer with local DMA could be as follows:

```
; Set up VIC068A to block transfer 128
; bytes in the BTLRs.
      move.b #$00, (vic_reg, $db)
      move.b #$80, (vic_reg, $df)
;
; Disable boundary crossing and dual
; path in the BTDR. This is the default configuration,
```

```
; but it is good to confirm the value.
      move.b #$00, (vic_reg, $ab)
;
; Configure the local bus timing for:
; PAS* asserted time = 90ns
; PAS* deasserted time = 45ns
; DS* deasserted time = 15ns
; in the LBTR.
; (These were probably set up at
; power-up configuration)
      move.b #$44, (vic_reg, $a7)
; Configure for:
; no interleave period
; write to VMEbus
; and enable block transfer w/ local
; DMA in the BTCR.
      move.b #$40, (vic_reg, $d7)
```

After the block transfer bit is set, the local processor performs a 32-bit write to the VMEbus location. This causes MWB* to be asserted to the VIC068A, which then requests the VMEbus. Any VMEbus access through the VIC068A then causes the VIC068A to enter the block transfer mode. It is important that this first transfer contain the starting address of local memory, not the data to be transferred.

If the 68K A0 register contains the local starting address and VME contains the VMEbus starting address, then the following instruction can be used to start the block transfer.

```
; Start the block transfer
      move.l vme, A1
      move.l local, A0
      move.l A0, (A1)
```

The local processor can check for the end of the block transfer by checking the DMASR. The following code does this:

```
; Test the DMA bit of the DMASR.
      LOOP:
      btst #0, (vic_reg, $BF)
      bne LOOP
```

Also, the VIC068A can be programmed to issue an interrupt to the local processor to indicate the end of the transfer. Use the DMSICR to enable this operation.

1.10.1.1.8.1.2 Sample 68K Code (Read)

For this example, the set-up is the same as in the previous example with the exception that the BTCR would be set as follows:

```
; Configure for:
; no interleave period
; read from VMEbus
; and enable block transfer w/
; local DMA in the BTCR.
      move.b #$50, (vic_reg,$d7)
```

The transfer could be started with the same code fragment used for the write example. As before, the completion of the DMA could be indicated by testing DMASR[0], or by an interrupt.

1.10.1.1.8.1.3 External Hardware Implementation

Figure 1–10 shows the hardware required for implementing the block transfers described above. The additional logic required is for latching the upper address bytes, LA[31:8]. Notice that for normal transfers, the ’543 drives the address bus. For block transfers, the address bus is driven by the ’373. The ’373 is loaded from LD[31:8] at the assertion of BLT*.

The simplest solution is to use CY7C964s in place of using discrete components (’373s and ’543s).

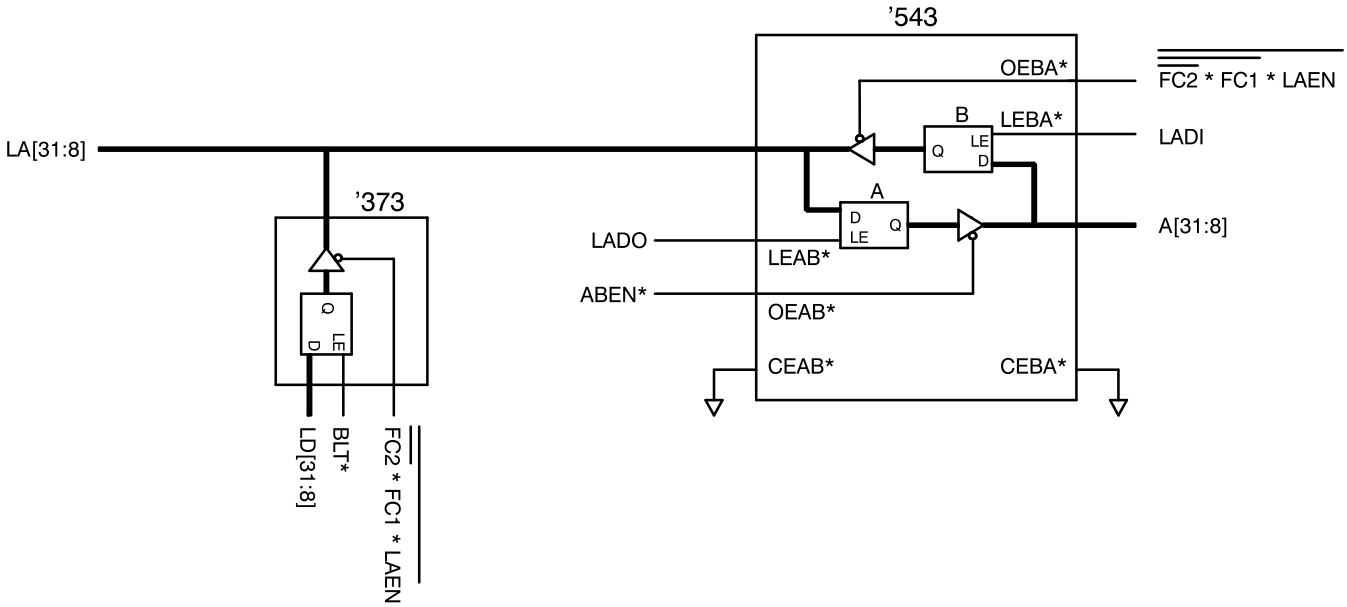

Figure 1–10. Minimum BLT Logic

1.10.1.1.8.2 Boundary Crossing Enabled, Dual-Path Feature Disabled

This mode allows implementing block transfers that are larger than the 256 bytes restricted by the VMEbus specification. The VIC068A does this by giving up the VMEbus after a 256-byte burst (or any size burst) and re-arbitrating for it at a later time (specified in the BTCR). The VIC068A system keeps track of all information required during the transfer so no additional software overhead is required. Some of the additional logic required for this operation consists of external counters that increment the LA[+:8] and A[+:8] counters as required. The VIC068A counts the lower 8 address bits only.

This cycle is initiated by writing the following registers on the master module:

- BTCR, to set up block transfer
- BTDR, to set up block transfer
- BTLRs, to configure length of block transfer
- LBTR, to configure local timing
- RCR, to specify burst length
- SS0CR1, to configure programmable delays
- DMASR, for DMA status
- DMASICR, to enable DMA-complete interrupt

Notice that the RCR must now be configured in order to specify the burst length.

The slave module must always be enabled for block transfers.

The software set-up for this operation is very similar to that of any block transfer with local DMA. The RCR must be configured to specify the burst length.

1.10.1.1.8.2.1 Sample 68K Code (write)

If we assume that the VIC068A registers are at *vic_reg*, as in the previous example, a sample of 68K code that would configure the VIC068A for a block transfer with local DMA and boundary crossing could be as follows:

```
; Set up VIC068A to block transfer 512
; bytes in the BTLRs.
      move.b #$02, (vic_reg, $db)
      move.b #$00, (vic_reg, $df)
;
; Specify a burst length of 64 cycles
```

```
; in the RCR
; ($00 specifies 64 cycles)
      move.b #$00, (vic_reg, $d3)
;
; Enable boundary crossing for both
; the local bus and the VMEbus.
; Disable dual path. Use the BTDR
      move.b #$0c, (vic_reg, $ab)
;
; Configure the local bus timing for:
; PAS* asserted time = 90ns
; PAS* deasserted time = 45ns
; DS* deasserted time = 15ns
; in the LBTR.
      move.b #$44, (vic_reg, $a7)
;
; Configure for:
; interleave period = 0
; write to VMEbus
; and enable block transfer w/ local
; DMA using the BTCR.
      move.b #$40, (vic_reg, $d7)
```

As in any block transfer with local DMA, the VMEbus is requested at the assertion of MWB*. In the above example, two bursts of 256 bytes (64 cycles of longwords) are performed. Between these bursts, an interleave time of “0” was specified. In this case, the VMEbus is immediately requested after it is given up after the first burst and the local bus will not be released. If a VMEbus master cycle is attempted during the interleave period, it will be DEDLKed (dual-path is not enabled).

The block transfer could be started as in any of the previous examples. Program the VIC068A to issue an interrupt or check for the completion of the transfer with the DMASR.

Notice that no additional software overhead is required (with the exception of writing the RCR) to perform a boundary crossing block transfer.

1.10.1.1.8.3 Boundary Crossing Enabled, Dual-Path Enabled

This mode of operation is similar to that of the previous example except that master cycles are allowed during the interleave period. If a master cycle is desired (MWB* asserted), the

VMEbus is requested and the transfer takes place like any other master cycle. A request for a master cycle is DEDLKed if dual-path is not enabled.

The same registers that were used in the previous example are used in this example. Namely:

- BTCR, to set up block transfer
- BTDR, to set up block transfer
- BTLRs, to configure length of block transfer
- LBTR, to configure local timing
- RCR, to specify burst length
- SS0CR1, to configure programmable delays
- DMASR, for DMA status
- DMASICR, to enable DMA-complete interrupt

The required software set-up is no different from that of any block transfer with local DMA except that the dual-path bit (BTDR[0]) must be enabled.

1.10.1.1.8.3.1 Sample 68K Code (Write)

A sample of 68K code that would configure the VIC068A for a block transfer with local DMA with boundary crossing and dual-path could be as follows:

```
; Set up VIC068A to block transfer 512
; bytes in the BTLRs.
      move.b #$02, (vic_reg, $db)
      move.b #$00, (vic_reg, $df)
;                  ;
; Specify a burst length of 64 cycles
; in the RCR.
; ($00 specifies 64 cycles)
      move.b #$00, (vic_reg, $d3)
;
; Enable boundary crossing for both
; the local bus and the VMEbus.
; Disable dual path. Use the BTDR.
      move.b #$0d, (vic_reg, $ab)
;
; Configure the local bus timing for:
; PAS* asserted time = 90ns
```

```
; PAS* deasserted time = 45ns
; DS* deasserted time = 15ns
; in the LBTR.
      move.b #$44, (vic_reg, $a7)
;
; Configure for:
; interleave period = 1000ns
; write to VMEbus
; and enable block transfer w/ local
; DMA using the BTCR.
      move.b #$44, (vic_reg, $d7)
```

The VMEbus is requested at the assertion of MWB*.

In this scenario, a block transfer of 512 bytes is performed in two blocks of 256 bytes. During the interleave period (1000 ns), master cycles can be performed.

1.10.1.1.9 D16 Block Transfers

The VIC068A is capable of performing D16 block transfers. If the WORD* signal is asserted during the initiation cycle, the VIC068A performs the block transfer with D16 protocol on the VMEbus.

The SWDEN* and ISOBE* signals can be configured to alternately toggle between the lower and the upper word banks, or swap all data to the upper word bank. If SS0CR0[4] (D32 enable) is set, SWDEN* and ISOBE* will toggle for 32-bit memory. If it is cleared, the data will not switch between the upper and lower banks; only SWDEN* is asserted.

1.10.1.1.10 Data Acquisition Delays

The programmable delays in SS0CR1 take on different meanings depending on whether the system is reading, writing, or even whether it is the first transfer or subsequent transfers. During block transfer writes, the MBAT timing reflects the minimum delays. The actual delays will depend on the speed in which DTACK* for the previous cycle is asserted. Notice that the VIC068A asserts the DS* signals before the DTACK* signal has been asserted in an attempt to read-ahead the next data. During the read-ahead cycle, the VIC068A cannot deassert the LEDO and DS* signals before DTACK* is asserted from the previous cycle.

Master Write Block Transfer (MBAT1/0)

DSACKi*(L) & DS*(L) to DS*(H)
DSACKi*(L) & DS*(L) to LEDO(H)

DSACKi*(L) & DS*(L) to DSi*(L)
DSACKi*(L) & DS*(L) to LA(7:0)

Master Read Block Transfer (MBAT1/0)

DSACKi*(L) & DS*(L) to DS*(H)
DSACKi*(L) & DS*(L) to LEDI(H)
DSACKi*(L) & DS*(L) to DSi*(L)
DSACKi*(L) & DS*(L) to LA(7:0)

The terms MBAT refer to the following bit fields:

MBAT0 = SSiCR1[3:0]
MBAT1 = SSiCR1[7:4]

Figures 1−11 through *1−14* demonstrate these delays. See Chapter 1.13 for the actual AC performance specifications.

1.10.1.2 MOVEM Block Transfers

The MOVEM block transfer is one in which the local CPU continues to be the local bus master. The VMEbus block transfer protocol is the same as it was for block transfers with local DMA. This cycle is initiated by writing the registers:

- RCR, to configure burst length
- BTCR, to configure block transfer

Once BTCR[5] is set, the first assertion of MWB* qualified with the assertion of PAS*, causes the VIC068A to start the MOVEM transfer. The MOVEM block transfer will terminate if any of the following occur:

- BTCR[5] is cleared
- BERR* is asserted
- any local bus cycle occurs in which MWB* is not asserted

BTLR1/0 are not used for MOVEM block transfers. During MOVEM block transfers, there is no latching of addresses. The VIC068A passes the local address onto the VMEbus address lines directly.

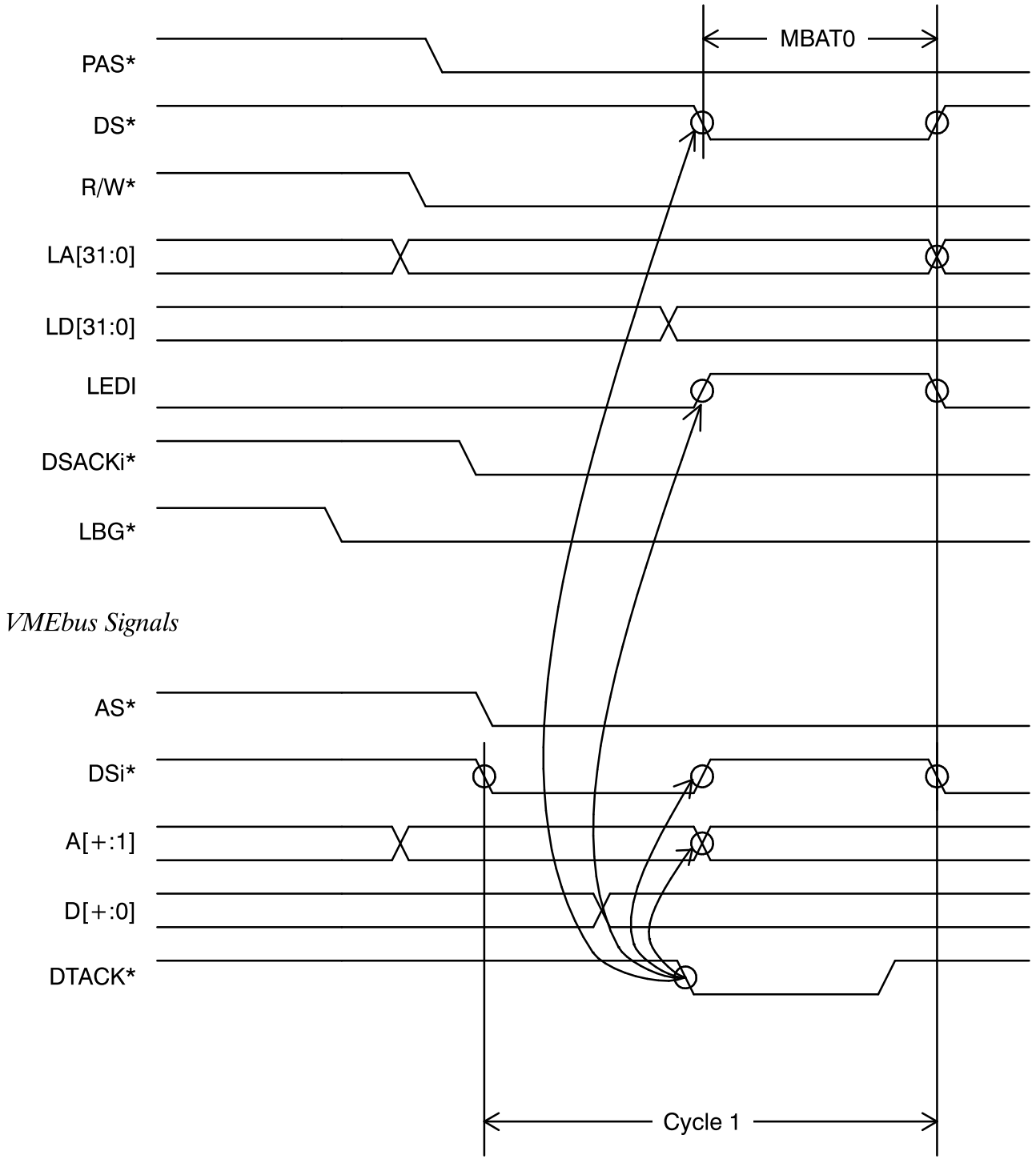

Figure 1−11. Master Block Transfer—Read, First Cycle

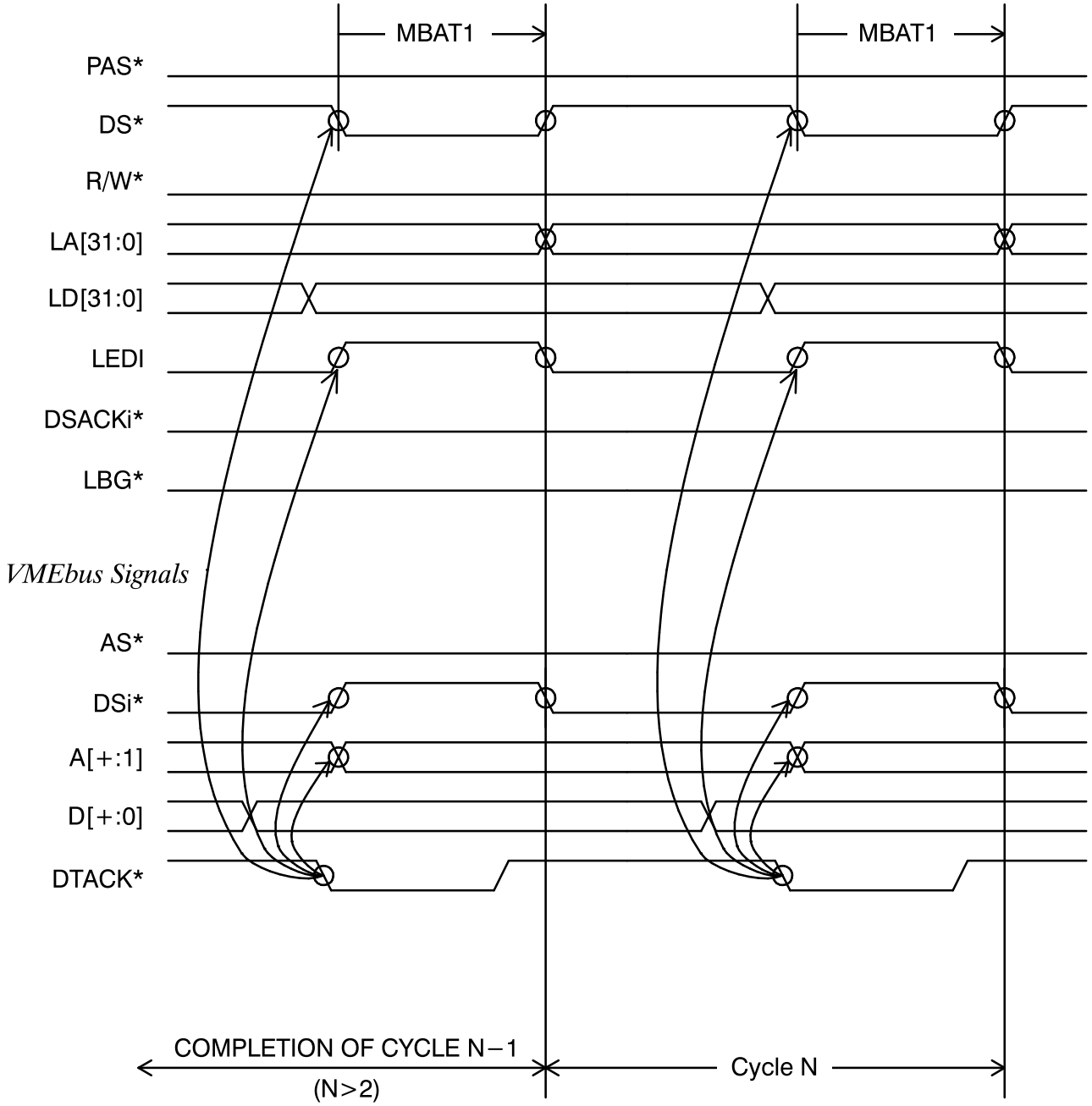

Figure 1−12. Master Block Transfer—Read, Second and Subsequent Cycles

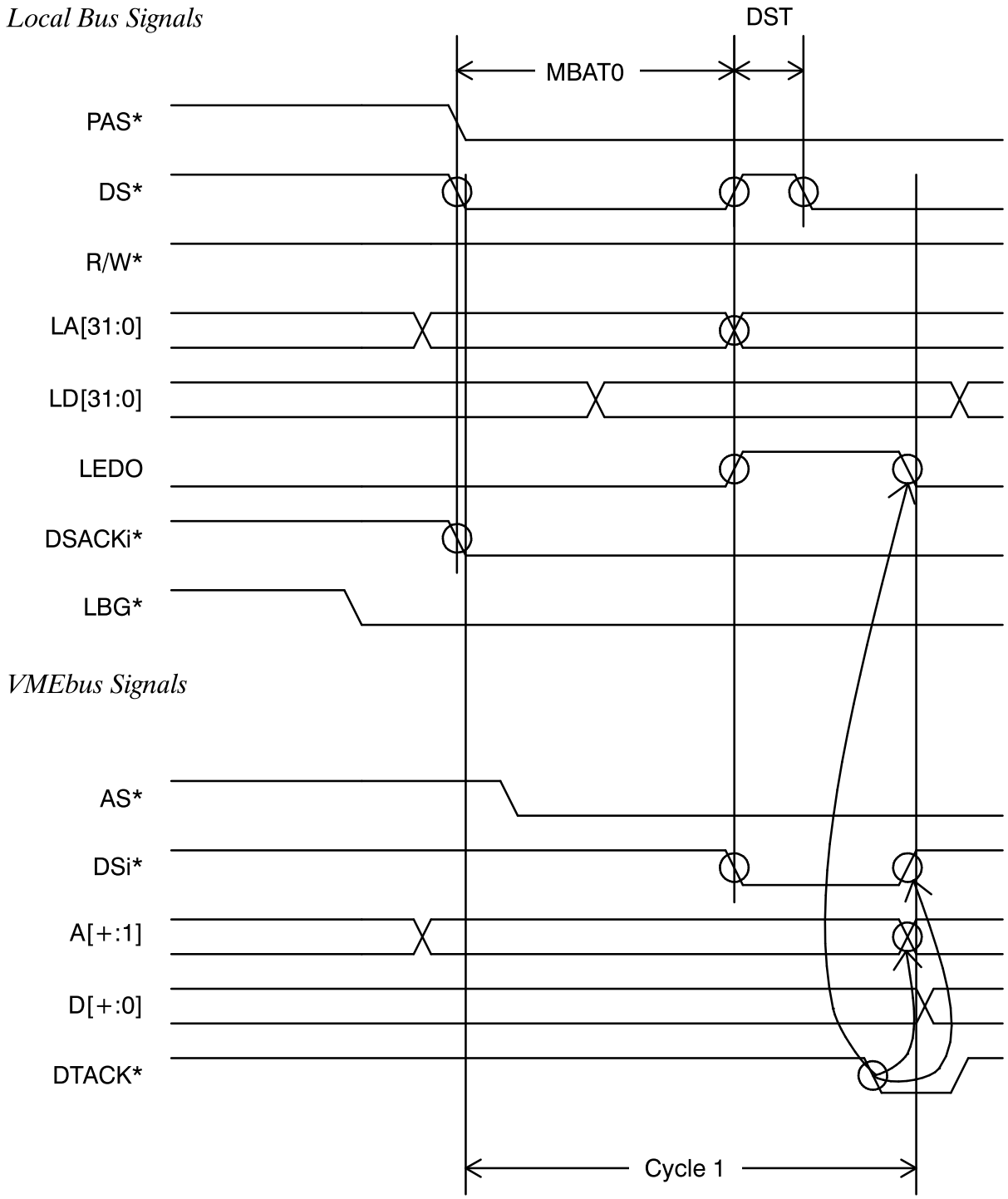

Figure 1−13. Master Block Transfer—Write, First Cycle

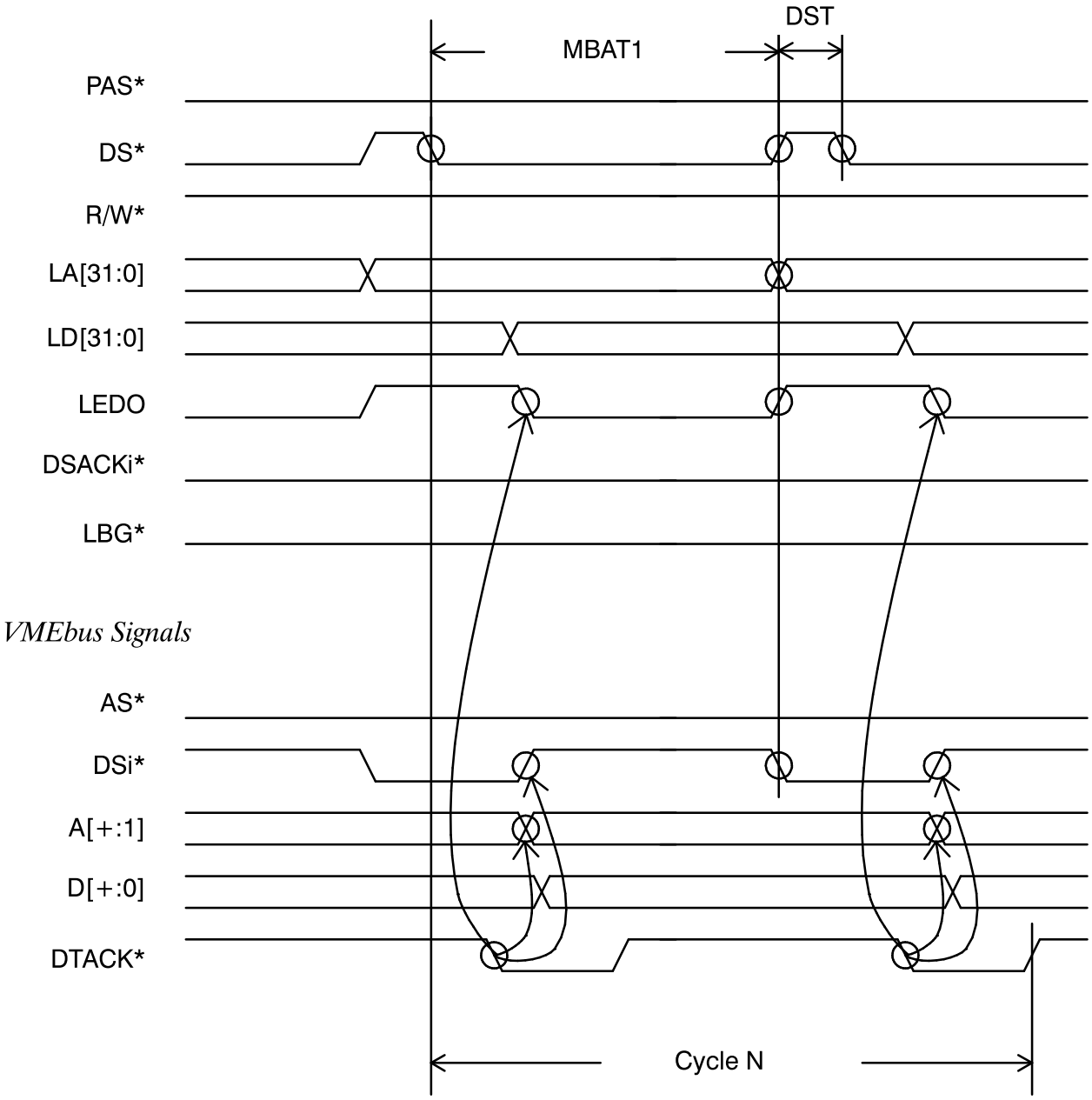

Figure 1−14. Master Block Transfer—Write, Second and Subsequent Cycles (Fast Slave)

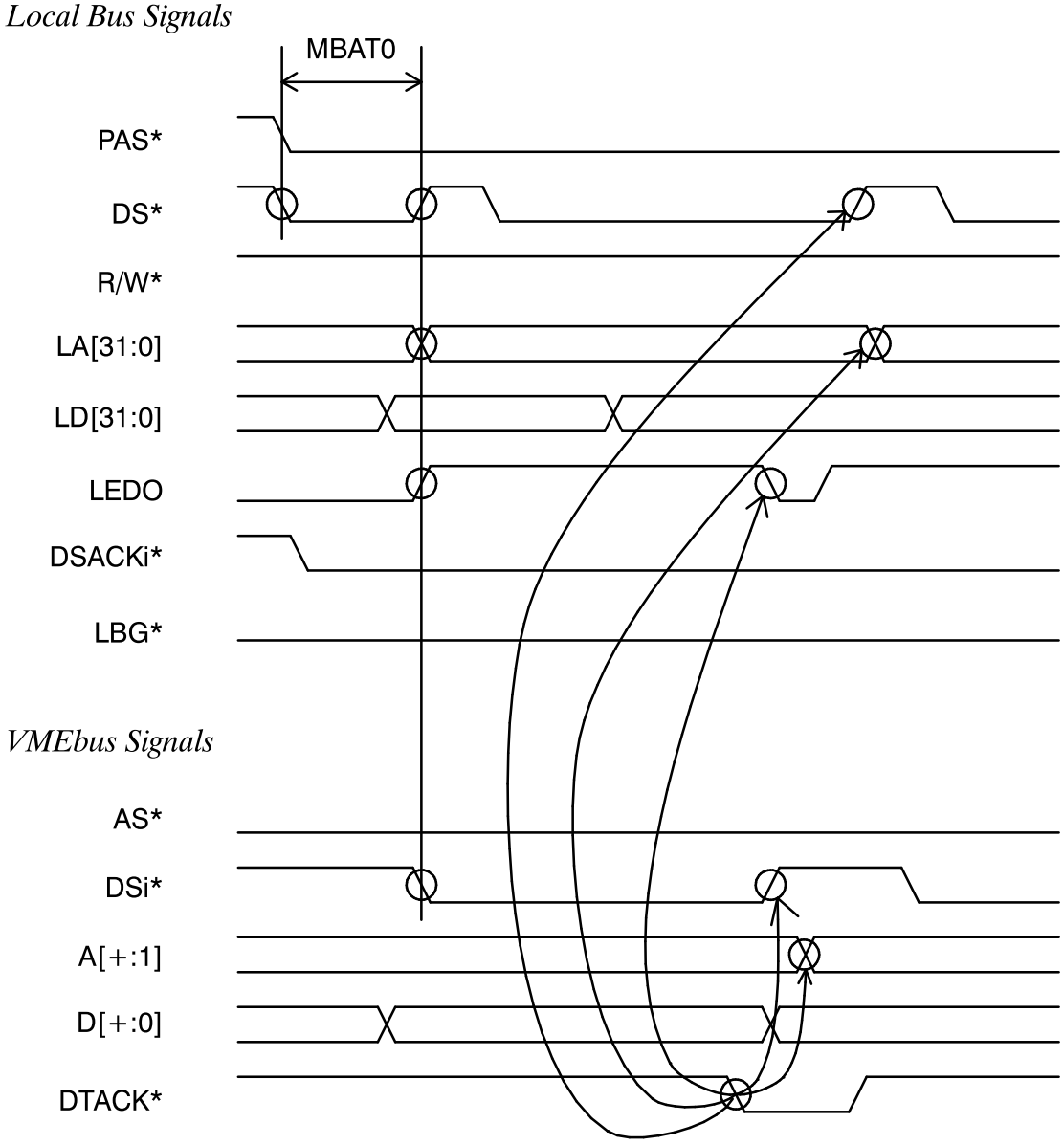

Figure 1−15. Master Block Transfer—Write (Slow Slave)

1.10.1.2.1 Sample MOVEM 68K Code

If, for example, we assume the VIC068A registers start at the base address *vic_reg*, a sample of 68K code that would configure a MOVEM block transfer and dump the word contents of the 68K d0 through d7 registers to memory could be as follows:

```
; Write burst length in to the RCR.
      move.b #$0f, (vic_reg, $d3)
;
; Set the MOVEM bit in the BTCR.
      move.b #$20, (vic_reg, $d7)
;
; Perform the MOVEM instruction.
; (a0 contains the destination
; address)
      movem.l 0xffff, (a0)
;
; Clear the MOVEM bit.
      move.b #$00, (vic_reg, $d7)
```

1.10.1.3 Buffer Control Signals During Master Block Transfers

The buffer control signals provide latching and bus-driving control for the address and data lines on both local and VMEbus sides. For MOVEM block transfers, the buffer control signals are identical in behavior to that of normal master transfers. For block transfers with local DMA, the behavior of the buffer control signals may act slightly differently. When using block transfers with local DMA, the loading, holding, and incrementing of the address counters and latches is controlled by LADO, ABEN* and the FC2/1 function codes.

Initialization Cycle

LADO is asserted as a result of PAS*, DS*, and MWB* being asserted for the initiation cycle.

VMEbus Read (Local Write)

LEDI is asserted as a result of the VMEbus DTACK* being asserted by the slave, indicating that valid data is available on the VMEbus. LEDI is deasserted as a result of MBAT1/0 expiring.

VMEbus Write (Local Read)

During block transfer writes, the MBAT timing reflects the minimum delays. The actual delays will depend on the speed in which DTACK* for the previous cycle is asserted. The

VIC068A asserts the DS* signal before the DTACK* signal has been asserted in an attempt to read-ahead the next data. During the read-ahead cycle, the VIC068A cannot deassert the LEDO and DS* signals before DTACK* is asserted from the previous cycle.

1.10.1.4 Performing Block Transfers to VMEbus Slaves Not Supporting Block Transfers

The VIC068A may be used to perform block transfers with local DMA to cards that do not accept block transfers. This is accomplished by performing the block transfer in bursts of 1 and using single-cycle AM codes (set BTDR[1]). This makes the VMEbus data appear to be single-cycle data, but data is transferred on the local bus by DMA. Because the VIC068A is in a block transfer mode, the LADO signal operates by deasserting after DS* is asserted (recall that the address for single-cycle transfers must be held for the duration of the cycle, but for block transfers the slave is responsible for latching the address at the beginning of the cycle). When the VIC068A is configured for bursts of 1, LADO may be configured to behave differently depending on certain register configurations. If BTDR[3] is set (VMEbus boundary crossing enabled), LADO is deasserted when DSi* is asserted for the final transfer. If BTDR[3] is clear, LADO will hold until the final DTACK* of the final transfer. Therefore, it is recommended that boundary crossing be disabled when performing these burst-of-1 transfers to non-block slaves.

1.10.2 VIC068A Slave Block Transfers

The VIC068A as a slave may be configured, in SSiCR0, to perform in one of three modes for block transfers:

- does not support block transfers
- supports block transfers, but emulates single-cycle transactions on the local slave side (toggle PAS* and DSACKi* for each transfer)
- supports block transfers in a DMA-type mode where PAS* and DSACKi* are asserted throughout the cycle and not toggled (accelerated transfers)

The VIC068A contains a slave block-transfer address counter separate from either of the address counters used for master block transfers. This counter, initialized by the VMEbus address, drives the local bus during the slave block transfer, and is incremented by the amount of data transferred with each local acknowledgement.

The timing for the slave's response to a block transfer is the same for that of a master block transfer with local DMA. VIC068A registers relevant to slave block transfers are as follows:

- Slave Select Control Registers (SS0CRi, SS1CRi)
- Local Bus Timing Register (LBTR)

The DMASR, DMASIR, RCR, BTCR, and the BTLRs are not used for slave block transfers. Slave block transfers that cross 256-byte boundaries are not supported in accordance with the VMEbus specification.

1.10.3 Buffer Control Signals During Slave Block Transfers

When the VIC068A is selected as a valid slave, LADI is asserted to latch the first address. From that point on, the VIC068A increments the lower 8 local address bits for each transfer. The data latches work as follows:

VMEbus Write (Local Write)

LEDI is asserted as a result of the VMEbus DSi* being asserted. LEDI is deasserted as a result of SBAT1/0 expiring.

VMEbus Read (Local Read)

Similar to master block transfer writes, the VIC068A attempts to read ahead the next data while the cycle is completing on the VMEbus. If DSi* from the previous cycle is not deasserted before SBAT1 expires for the read-ahead cycle, LEDO and DS* are deasserted at the deassertion of DSi*. If DSi* is deasserted before SBAT1 expires, LEDO and DS* are deasserted at that point.

1.10.4 Using the CY7C964 for Additional Block Transfer Support

The CY7C964s are perfect companion chips for designs for which all of the VAC068A's functionality is not required, or for designs that cannot incorporate the VAC068A's address mapping.

The CY7C964 provides all logic necessary for

- local boundary crossing
- VMEbus boundary crossing
- dual-path functionality

See Section 4, The CY7C964 Bus Interface Interface Logic Circuit, for details on a VIC068A/CY7C964 interface.

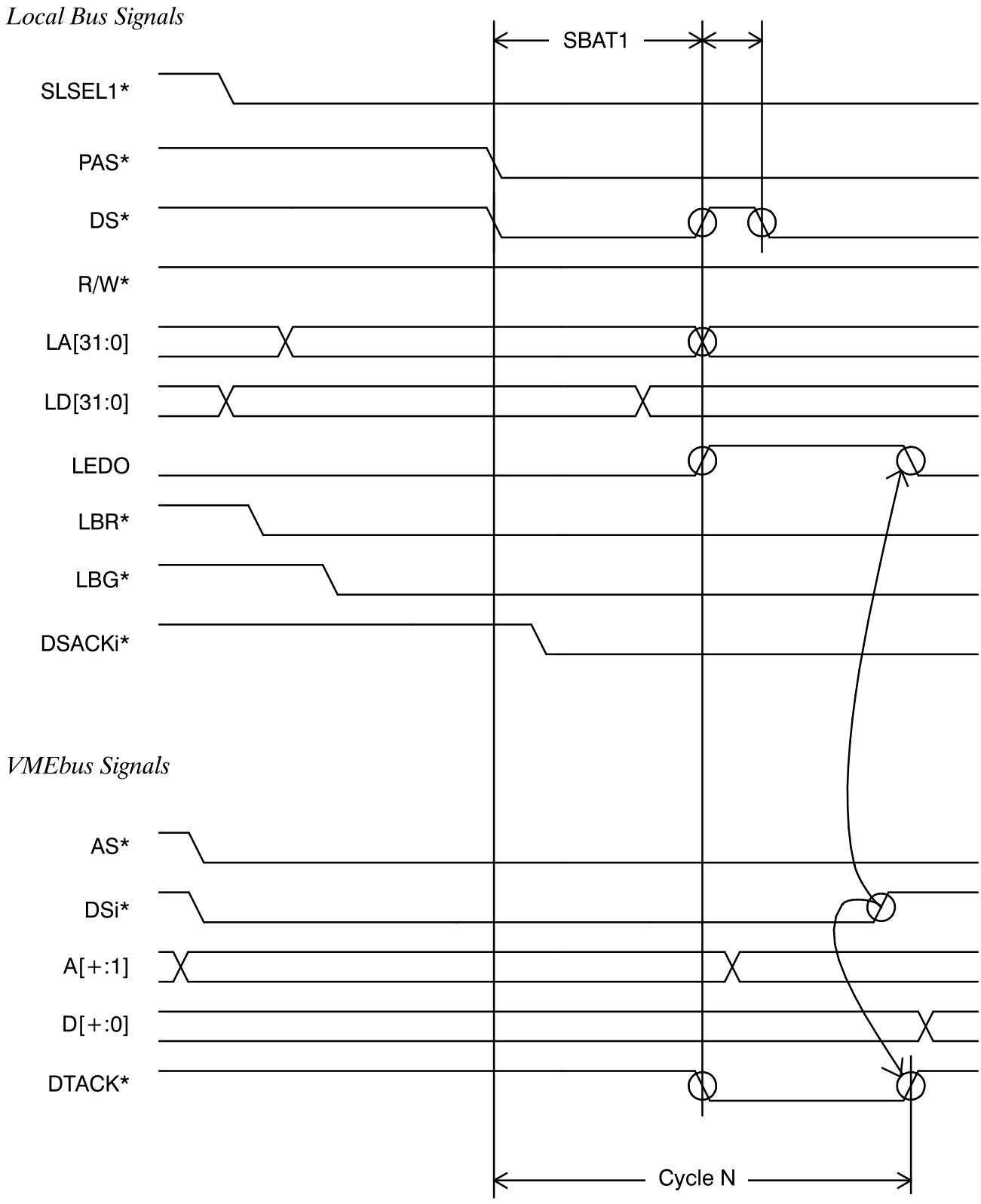

Figure 1−16. Slave Block Transfer—Read, First Cycle

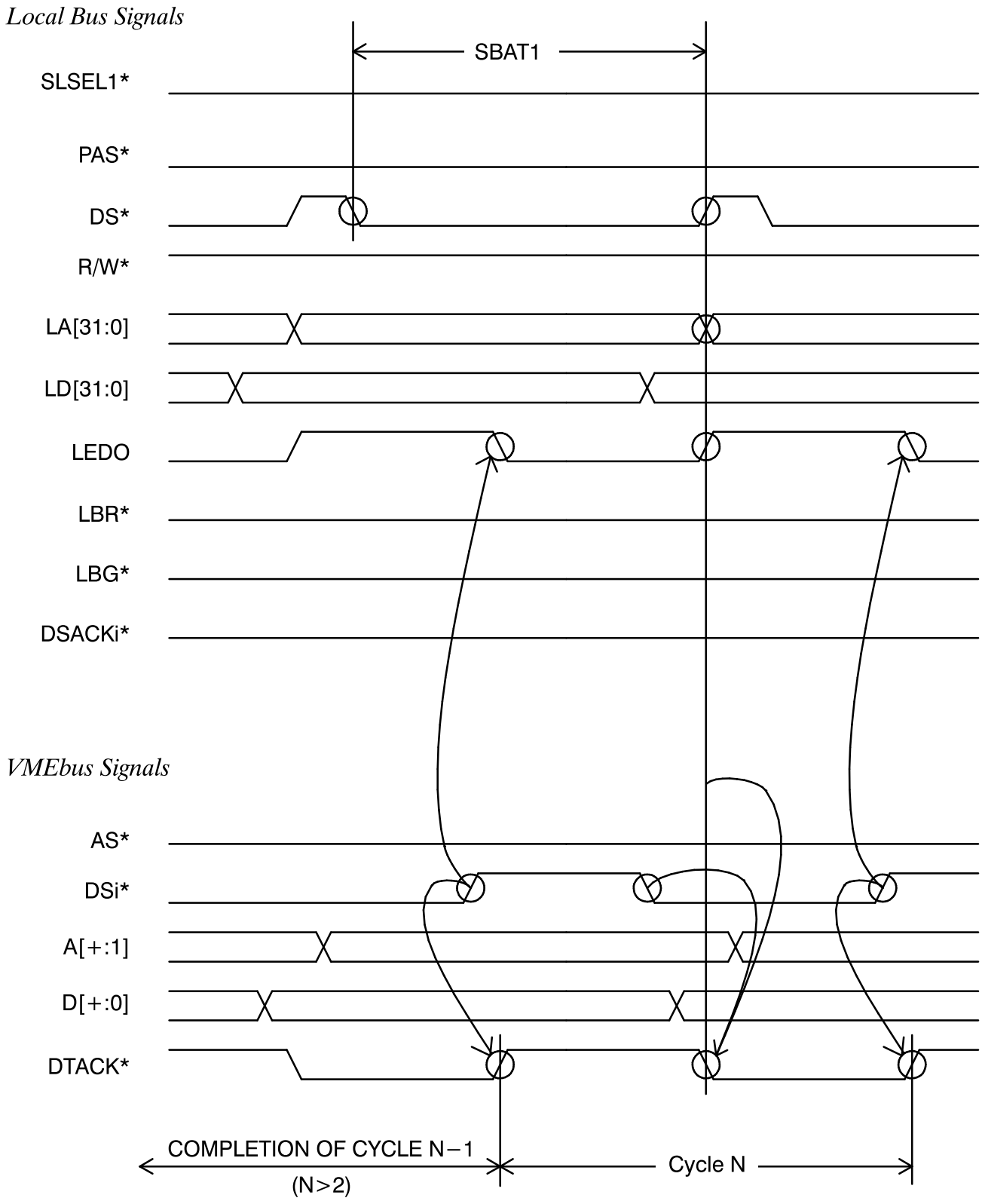

Figure 1−17. Slave Block Transfer—Read, Second and Subsequent Cycles

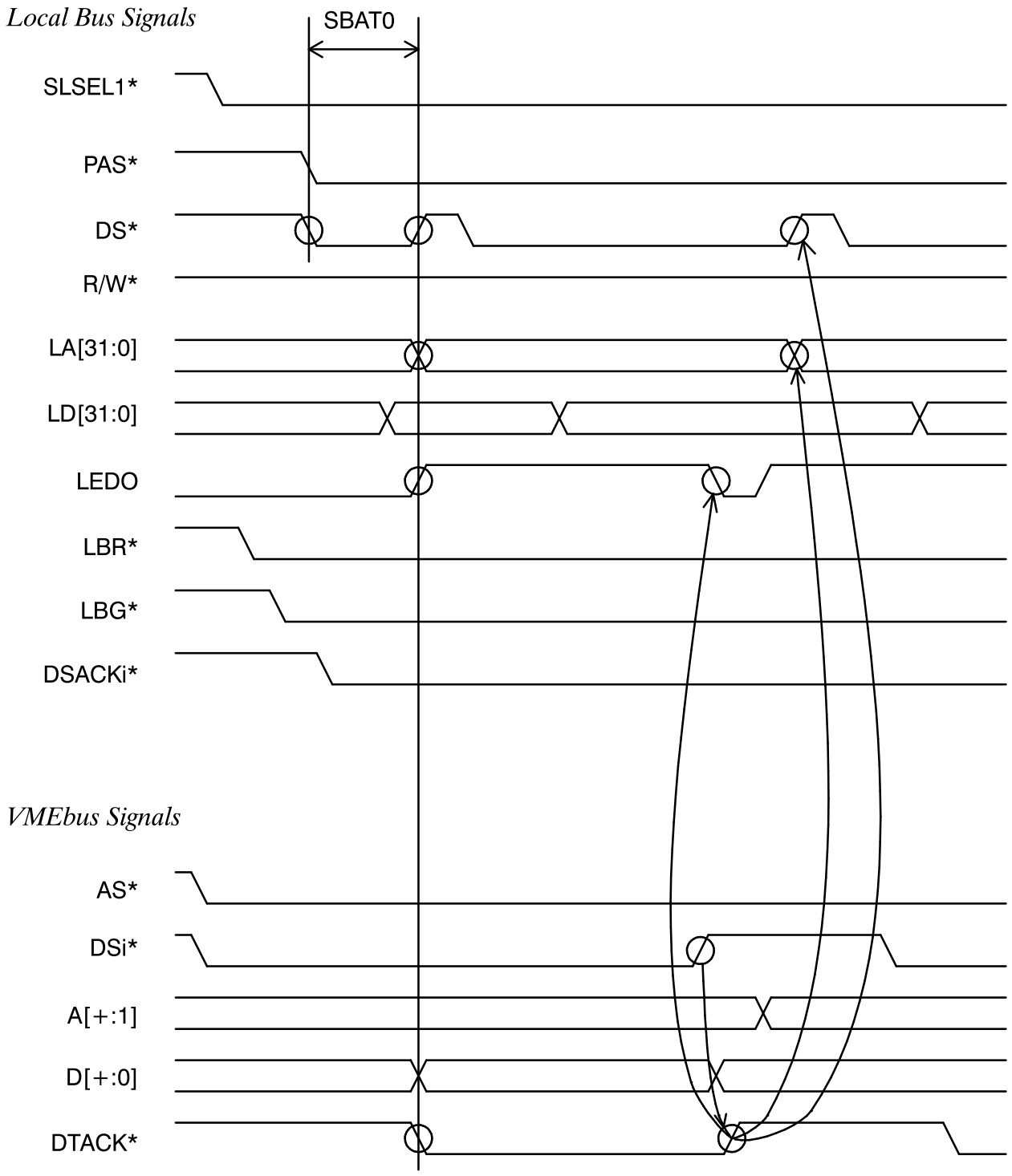

Figure 1−18. Slave Block Transfer—Read, Slow Master

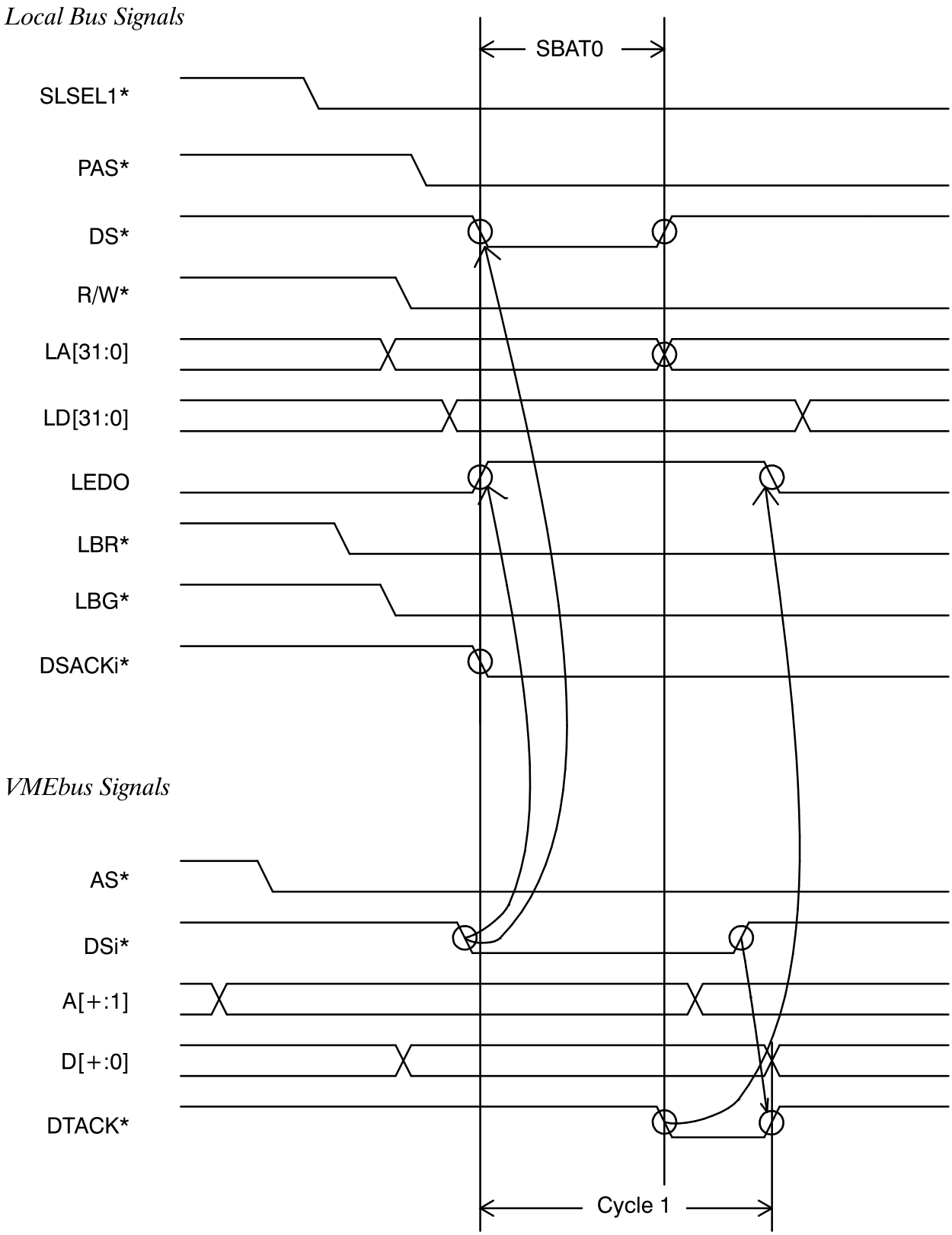

Figure 1−19. Slave Block Transfer—Write, First Cycle

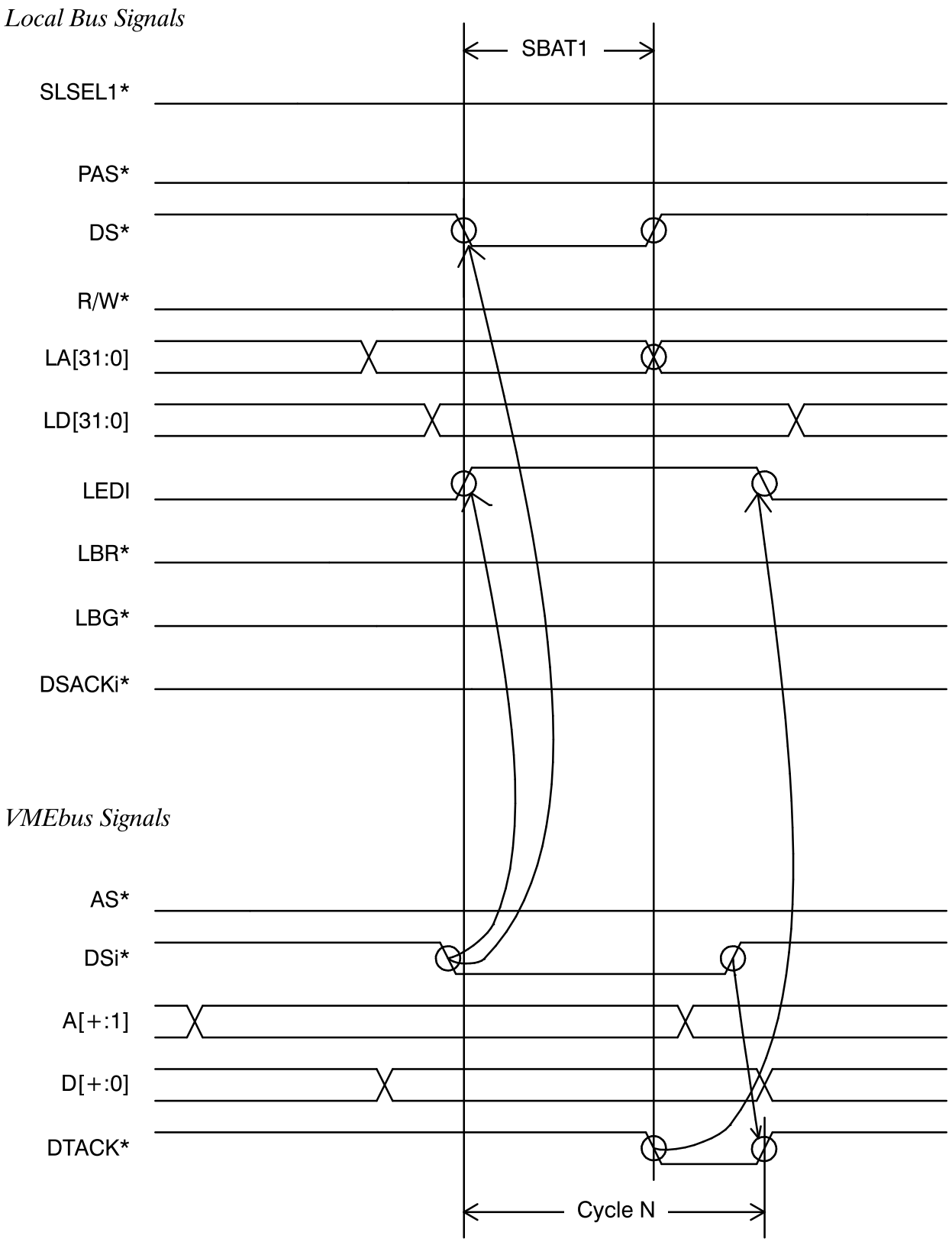

Figure 1−20. Slave Block Transfer—Write, Second and Subsequent Cycles