1.5

VIC068A VMEbus Master Operations

The transfer of data is initiated by a VMEbus master module. The master module controls the type of transfer (read, write, interrupt acknowledge, etc.) and provides the address and address modifiers for the transfer. The timing of the start of the transfer is also controlled by the master.

The following VIC068A registers are used for master operations (block transfer registers not included):

- Transfer Timeout Register (TTR), bits 1, 2–4
- Interface Configuration Register (ICR), bits 1–7
- Arbiter/Requester Configuration Register (ARCR), bits 0–3, 5, 6
- Address Modifier Source Register (AMSR)
- Bus Error Status Register (BESR), bits 0–3
- Slave Select 1 Control Register 0 (SS1CR0), bit 6
- Release Control Register (RCR), bits 6–7

See Chapter 1.12 for descriptions of these registers.

1.5.1 VMEbus Requests

There are many types of cycles in which the VIC068A requests the VMEbus. These include:

- SINGLE-cycle data transfer requests (SINGLE)
- status/ID fetches for Interrupt ACKnowledge cycles (IACK)
- Indivisible Single-Address Cycles (ISAC) such as read-modify-write cycles
- Indivisible Multiple-Address Cycles (IMAC)
- Block Transfer Requests (BLT)
- VMEBus Capture And Hold (BCAP) requests

The actual assertion of the BRi* signals are made in response to the following signals:

- assertion of MWB* qualified by PAS* for single-cycle and block-transfer accesses
- assertion of FCIACK* qualified by PAS* for VMEbus interrupt acknowledge cycles
- assertion of RMC* qualified by PAS* (when the ICR is appropriately programmed) for ISAC and IMAC cycles
- setting the BCAP bits in the ICR for BCAP and IMAC cycles

The request level is set in ARCR[6:5]. The default level is BR3*.

1.5.2 Release Modes

The VIC068A supports the four VMEbus release modes:

- Release On Request (ROR)
- Release When Done (RWD)
- Release On Clear (ROC)
- VMEbus Capture And Hold (BCAP)

In addition to these, the VIC068A also allows an extension of the above items to provide for the use of the RMC* signal. This is referred to as Release Under RMC* Control. These modes are selected by writing RCR[7:6]. The Release Under RMC* Control mode is programmed by setting ICR[5].

1.5.2.1 Release On Request (ROR)

In this release mode, the VIC068A deasserts BBSY* when a BRi* is asserted by another VMEbus module and the VIC068A has no need for the VMEbus. The VIC068A does not assert the ABEN* signal if there is no data transfer in progress and the VIC068A is currently the VMEbus master.

1.5.2.2 Release When Done (RWD)

In this mode, the VIC068A deasserts the BBSY* signal as soon as the following conditions occur:

1. BBSY* has been asserted by the VIC068A for a minimum of 90 ns
2. AS* has been deasserted by the VIC068A
3. The VIC068A has no further need for the VMEbus (the VIC068A has not asserted BRi* for the last 2T)
4. BGiIN* is not asserted to the VIC068A

1.5.2.3 Release On Clear (ROC)

In this mode, the VIC068A continues to assert BBSY* until the BCLR* signal is asserted by the system controller.

1.5.2.4 VMEbus Capture and Hold (BCAP)

In this mode, the VIC068A asserts BBSY* continuously for as long as the BCAP mode is selected. The release of BBSY* occurs by programming the release control bits to another release mode. If RWD is selected, BBSY* is released immediately. If ROR is selected, BBSY* is released at a pending VMEbus request. The VIC068A deasserts BBSY* on the assertion

of BCLR* if ROC is selected. Do not enter the BCAP mode if the VIC068A is currently the VMEbus master.

1.5.2.5 Release Under RMC* Control

In this mode, the VIC068A both requests and holds the VMEbus under control of the RMC* signal. When appropriately programmed by setting ICR[5], the assertion of RMC* and PAS* causes the VIC068A to request the VMEbus, accept the BGiIN*, and assert BBSY*. The deassertion of RMC* allows the deassertion of BBSY* based upon the release mode programmed.

1.5.3 VIC068A VMEbus Master Write Cycle

If the VIC068A is not the current VMEbus master, the VIC068A bids for access to the VMEbus when it receives the MWB* and PAS* signals asserted. When all of the following conditions occur:

1. AS* is deasserted from the previous cycle
2. DTACK* and BERR* are deasserted
3. the BGiIN* has been received
4. all appropriate metastability settling delays have elapsed

the VIC068A drives the D[7:0] data buffers onto the VMEbus and asserts DENO*, which should be used to enable the remaining data buffers. At the same time, the VIC068A enables the A[7:0] address lines onto the VMEbus in addition to asserting the ABEN* signal to drive the remaining VMEbus address lines. The VIC068A also drives AM[5:0], WRITE*, and LWORD* as required. At this time, the VIC068A initiates an internal delay to insure appropriate address set-up time before the assertion of the AS*. After AS* is asserted, the VIC068A latches the LA[7:0] and asserts the LADO signal, which should be used to latch the remaining local address lines.

After the AS* signal has been asserted, the VIC068A initiates an internal delay to assert the data strobes (DSi*). When this delay has elapsed, the VIC068A asserts the appropriate data strobes as determined by the size and alignment of the transfer. The DSi* signals remain asserted until either DTACK* or BERR* have been asserted to the VIC068A. If DTACK* is asserted, the VIC068A asserts the DSACKi* signals according to the port size. That is, if the WORD* signal was deasserted, the VIC068A acknowledges this D32 operation by asserting both the DSACK0* and DSACK1* signals. If the WORD* signal was asserted, the VIC068A acknowledges this D16 transfer by asserting only the DSACK1* signal. For example, when performing a longword transfer to a D16 device, asserting only DSACK1* would notify the processor that the additional word of data needs to be transferred. This is consistent with the Motorola 68K dynamic bus sizing capabilities using DSACKi*.

When turbo mode is enabled by setting ICR[1], the VMEbus address and data set-up times are decreased by 1T.

Tables 1-1 through *1-4* show the buffer control signals for various master cycles.

Table 1-1. Buffer Control Signals: D32 VMEbus Master Write Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL HL LL HL	L L H H	L L H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	H H H H	L L L L	H H H H	H H H H
Three-Byte	1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LL HL	L L H H	L L H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	H H H H	L L L L	H H H H	H H H H
Word	1 1 1 1	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL LL LL HL	L H H H	H L H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	L H H H	H L L L	H H H H	H H H H
Byte	1 1 1 1	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	↑ ↑ ↑ ↑	L L L L	L L L L	L L L L	H H H H	L L H H	H H L L	H H H H	H H H H

Table 1-2. Buffer Control Signals: D32 VMEbus Master Read Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL HL LL HL	L L H H	L L H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	H H L L	L L L L	L L L L	L L H H
Three-Byte	1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LL HL	L L H H	L L H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	H H L L	L L L L	L L L L	L L H H
Word	1 1 1 1	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LL LL LL LL	LL LL LL HL	L H H H	H L H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L H L L	L L L L	L L L L	H L H H
Byte	1 1 1 1	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LL LL LL LL	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	L L L L	L L L L	H H H H

Table 1-3. Buffer Control Signals: D16 VMEbus Master Write Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LH LH LH LH	LL HL LL HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H
Three-Byte	0 0 0 0	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LL HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H
Word	0 0 0 0	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LH LH LH LH	LL LL LL HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H
Byte	0 0 0 0	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	↑↑↑↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H	H H H H	H H H H

Table 1-4. Buffer Control Signals: D16 VMEbus Master Read Operation

Data Path Size	Local Bus Stimulus				VMEbus Response			Address Control			Data Control			Swap Control				
	WORD*	SIZ1/0	LA[1:0]	DSACK1/0*	DS1/0*	A01	LWORD*	ABEN*	LADI	LADO	DENO*	LEDI	LEDO	DDIR1	SWDEN*	ISOBE*	DENIN*	DENIN1*
Longword	0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LL HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H
Three-Byte	0 0 0 0	1 1 1 1 1 1 1 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LL HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H
Word	0 0 0 0	1 0 1 0 1 0 1 0	0 0 0 1 1 0 1 1	LH LH LH LH	LL HL LL HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H
Byte	0 0 0 0	0 1 0 1 0 1 0 1	0 0 0 1 1 0 1 1	LH LH LH LH	LH HL LH HL	L L H H	H H H H	L L L L	L L L L	L L L L	H H H H	↑	L L L L	L L L L	L L L L	H H H H	L L L L	H H H H

1.5.4 VIC068A VMEbus Master Read Cycle

This cycle is identical to that of the master write cycle, as described in section 1.5.3, with the following exceptions:

- The VMEbus data buffers are not driven.
- DENO* is not asserted.
- The DENIN1* and DENIN* are asserted.
- DDIR is not asserted. The address and AS* considerations are the same as well as the DSACKi* conventions.

1.5.5 Master Write Posting

The VIC068A is enabled for master write-posting by setting SS1CR0[6]. When enabled, the VIC068A captures the local address, data and control signals, requests the VMEbus, and immediately acknowledges the local processor. This frees the local processor from waiting

for VMEbus arbitration. When VMEbus mastership is obtained, the VIC068A performs the transfer according to normal VMEbus protocol. Further write-posts are disabled until a DTACK* or BERR* is asserted by the slave. If a BERR* is signaled, the VIC068A can be configured to issue a local interrupt by clearing EGICR[6].

If a slave read occurs after a write has been posted but not yet transferred, the latched write data is “toggled” to the B-to-A latch of the ’543s by asserting the LEDI signal. The slave read then occurs normally without the write data being written over by the slave read data. After the VIC068A asserts DTACK*, the DENIN1* and DENIN* signals are asserted to drive the write data to the A-to-B latch of the ’543. Then LEDO is asserted to again latch the data for the write operation. This toggling of data is referred to as a Master Write-Post Data Switchback.

1.5.6 Indivisible Cycles

Indivisible cycles can be divided into two categories:

- Indivisible Single-Address Cycles (ISACs)
- Indivisible Multiple-Address Cycles (IMACs)

The VIC068A supports both ISACs and IMACs through many different protocols. Indivisible cycles can be configured as follows:

1. Request the VMEbus on the assertion of RMC* independent of MWB* (this prevents any slave access from interrupting local indivisible cycles).
2. Stretch the VMEbus AS*.
3. Make the above behaviors dependent on the local SIZi signals.

These modes are summarized in *Table 1-5*.

Table 1-5. RMC* Control Map

ICR[7:5]	First Operation	VMEbus Requested on RMC* Assertion	AS* Stretched	VMEbus Held During RMC* Assertion
X 0 0	Any	No	No	No
0 0 1	Any	Yes	No	Yes
0 1 0	Any	No	Yes	Yes
0 1 1	Any	Yes	Yes	Yes
1 0 1	Byte	No	No	No
1 0 1	Non-byte	Yes	No	Yes
1 1 0	Byte	No	Yes	Yes
1 1 0	Non-byte	No	No	No
1 1 1	Byte	No	Yes	Yes
1 1 1	Non-byte	Yes	No	Yes

Address strobe stretching is performed for ISACs in accordance with the VMEbus RMC specification. For IMACs, the address strobe is typically not stretched in order for slave modules to latch each address.

In the 68K family of processors, ISACs and IMACs are distinguished by the fact that the first read of an IMAC is never of byte size. This allows for AS* stretching for all RMCs, no RMCs, or only RMCs in which the first transfer was of byte size.

If a processor is not capable of generating indivisible cycles or does not distinguish ISACs from IMACs, cycle indivisibility may be guaranteed by using the BCAP release mode outlined below:

1. Set the VIC068A to a BCAP release mode by setting RCR[7:6].
2. Wait for VMEbus grant (BGiIN*).
3. Perform indivisible cycles.
4. Release the VIC068A from BCAP mode in the RCR.

When the VIC068A is the VMEbus slave to a ISAC, the VIC068A maintains the local bus by keeping LBR* asserted as long as AS* is asserted.

1.5.6.1 Indivisible Single-Address Cycles (ISACs)

The most common implementation of ISACs are of the read-modify-write (RMC) category. This is the only ISAC supported by the VMEbus. The Motorola TAS (Test And Set) instruction is an example of a read-modify-write cycle. The VMEbus specification requires that, for RMC cycles, the VMEbus address strobe be held asserted between the read and the write cycles. Motorola processors prior to the 68020 performed ISACs in the same manner by asserting their address strobes for the duration of the cycle. The Motorola processors such as the 68020/30/40 provide a signal (RMC* for the 68020/30, and LOCK* for the 68040) to indicate that a RMC is being performed. The VIC068A has an RMC* signal that is typically connected to these signals to control ISACs. For RMC cycles, the AS* should be programmed to be stretched.

1.5.6.2 Indivisible Multiple-Address Cycles (IMACs)

The Motorola CAS and CAS2 instructions are examples of IMACs. The VIC068A allows for the support of IMACs without using the BCAP protocol given in section 1.5.6. In this case, the RMC* signal should be set to request and hold the VMEbus. The local cycle is not allowed to complete until the VMEbus has been obtained. In addition, AS* stretching should be disabled so that address latching may be performed by the slave.

1.5.7 Deadlock

If the VMEbus is requested in response to MWB* or FCIACK* being asserted, and at the same time a valid slave select has been signaled, a deadlock has occurred. The VIC068A may be programmed in the ICR to signal a deadlock in the following ways:

- assert DEDLK*
- assert DEDLK*, LBERR*, and HALT*
- assert DEDLK* and LBERR* (without HALT*) for RMC deadlocks

The first option is typically used for non-Motorola processors without a retry capability. In that case, DEDLK* should be used to signal the processor to vacate the local bus (deassert MWB*).

For Motorola 68K applications, the second option may be used to signal the processor to retry the current bus cycle using the Motorola BERR*/HALT* retry mechanism.

For Motorola 68K processors, the HALT*/BERR* retry mechanism is disabled for RMC cycles. In this condition, the third option should be used to signal a generic BERR* to the processor. The BERR* exception processing routines should include software code that checks the Special Status Word (SSW) of the 68K BERR* exception stack frames to indicate RMC status.

In all of the above cases, DEDLK* is not deasserted until the slave access causing the deadlock is compete.

When a cycle is DEDLKed, the VMEbus is still requested. If the VMEbus is granted (after the slave access is complete) and is still available when MWB* is reasserted, the cycle proceeds as normal. If the VMEbus is granted and MWB* is not reasserted, the VIC068A asserts BBSY* for the 90 ns required by the VMEbus specification if it is configured as RWD. If configured for ROR, the VIC068A maintains the BBSY* until requested to release it.

1.5.7.1 Undetectable Deadlocks

Poor system design can lead to deadlocks that the VIC068A cannot detect and from which it cannot recover. Consider the following example:

1. Two boards, CPUa and CPUb, contain a local processor that has dual-ported memory connected to both the VMEbus and a VSBbus.
2. CPUa is local bus master and VSBbus master and desires data from CPUb’s memory over the VSBbus.
3. CPUb is local bus master and VMEbus master and desires data from CPUa’s memory over the VMEbus.
4. Because both CPUs are their own local bus masters, neither will be able gain access to the others local bus. DEADLOCK!

The VIC068A is not able to detect this deadlock because the VIC068A is only able to monitor the status of the local bus and the VMEbus. Deadlocks due to the existence of other buses, such as VSB, will not be detected.

The only way to recover from these types of deadlocks is to use bus timeout timers.

1.5.8 Self-Access

If a slave select is signaled while it is the VMEbus master, a self-access has occurred. The VIC068A signals a self-access by asserting both the LBERR* and BERR* signals. The BESR also indicates self-access status.

Self-accesses may be used to determine the slave address map of a module.

1.5.9 VMEbus/Local Bus Data and Port Size

A distinction should be made regarding the terms *transfer size* and *port size*. Transfer size indicates the size of the data in terms of bytes, words, and longwords. The port size indicates the physical size of the bus the data will be transferred on. Port sizes for the VMEbus are usually given in terms of D8, D16, and D32 for 8-bit, 16-bit, and 32-bit-wide buses respectively.

The transfer size of the master operation is indicated to the VIC068A by the SIZ1/0 signals according to the following table:

SIZ1	*SIZ0*	*Data Size*
0	0	Longword (32 bits)
0	1	Byte (8 bits)
1	0	Word (16 bits)
1	1	3-byte

This information insures proper VMEbus protocol in terms of LWORD*, A01, and DS1/0*. In addition, the VIC068A buffer control signals will be properly asserted for the size of the transfer.

The port size of the transfer is indicated by the WORD* signal. When asserted, the master transfer is treated as a D16 transfer. For D16 operations, the LWORD* signal is not asserted, and the DS1/0* signals behave appropriately. In addition, the SWDEN* and ISOBE* signal may be asserted differently in that the D16 VMEbus data located on D[15:0] may be swapped onto the LD[31:16]. This depends on the size of the transfer, the alignment of the transfer, and whether performing a read or a write. Refer to *Tables 1-3* to *1-6* for more details on these signals for particular cases.

The WORD* signal may be changed dynamically to enable the VIC068A to deal with both D16 and D32 slaves.

When performing D16 operations, the VIC068A asserts only the DSACK1* signal to indicate to the processor the port size is 16 bits. This would indicate to a 68K processor that when

transferring a longword of data, two transfers are required. This is consistent with the Motorola 68K DSACKi* dynamic bus-sizing convention.

When using a 16-bit local processor, the WORD* signal must be asserted for all master transfers, or strapped Low at power-up to perform D16 transfers. The VIC068A does not support using a 16-bit local bus to a 32-bit VMEbus (D32).

1.5.10 Fair Request Timeout

A fair request timeout scheme may be used to prevent VMEbus starvation of any master in a bus request daisy-chain. When operating in a fair request mode, the VIC068A does not assert its BRi* signal until or unless that request level is in its deasserted state. If all boards in a system obey this fairness doctrine, VMEbus starvation will not occur.

To minimize starvation caused by unfair masters, the VIC068A is also equipped with a fair request timeout timer. If the VIC068A is unable to obtain VMEbus mastership within a programmed delay, the VIC068A stops using the fairness doctrine and asserts its BRi* without delay.

Fairness is controlled by writing the ARCR. Fairness is disabled by clearing bits ARCR[3:0]. Fairness is enabled (with no timeout) by setting bits ARCR[3:0]. The timeout timer is enabled by writing any other combination to these bits. The value of the timeout is 2 μs times the number written. A difference of 2 μs may exist between the value written and the actual delay observed.

1.5.11 Address-Only Cycles

The VIC068A will not perform address-only cycles. The VIC068A, as slave, can accept address-only cycles.

1.5.12 The Address Modifiers for Master Cycles

When the VIC068A performs master cycles, it examines the ASIZ1/0 and FC2/1 signals to determine the value of the AM[5:0] signals that will be driven. The information that the AM codes specify indicates address sizing and supervisory/user and program/data information. Under normal circumstances, the VIC068A outputs standard VMEbus AM codes. The VIC068A may also be configured to output user-defined AM codes. This is done with the ASIZ1/0 signals and the AMSR. If the ASIZ1/0 signals are both Low, the VIC068A uses the AMSR to determine the value of the AM codes.

If AMSR[7] is clear, VIC068A issues the contents of AMSR[5:0] to the AM[5:0] signals. If AMSR[7] is set, the VIC068A issues AM codes based on AMSR[5:3] and the FC2/1 inputs.

Table 1-6 summarizes the AM codes for various VIC068A operations and configurations.

Table 1-6. Master Transfer AM Code Control Map

VIC068A Master Access Inputs				VIC068A AM Code Output	
ASIZ1/0	Address Size	Block Transfer	FC2/1	Operation Type	AM[5:0]
0 1	A32 Addressing	No	0 0 0 1 1 0 1 1	User Data User Program Supervisory Data Supervisory Program	$09 $0A $0D $0E
0 1	A32 Addressing	Yes	0 X 1 X	User Block Supervisory Block	$0B $0F
1 1	A24 Addressing	No	0 0 0 1 1 0 1 1	User Data User Program Supervisory Data Supervisory Program	$39 $3A $3D $3E
1 1	A24 Addressing	Yes	0 X 1 X	User Block Supervisory Block	$3B $3F
1 0	A16 Addressing	No	0 X 1 X	User Access Supervisory Access	$29 $2D
0 0	User Defined AMSR[7] = 0	Yes/No		User Defined	AMSR[5:0]
0 0	User Defined AMSR[7] = 1	Yes/No	 0 0 0 1 1 0 1 1	User Defined	AMSR[5:3] + $01 $02 $05 $06