TAS Cycles

A Study of the Viability of the TAS
Instruction on the Zorro II Bus

Document Revision 1.00

Preliminary Release

by Dave Haynie
June 20, 1990

There has been a considerable amout of confusion about the use of the TAS, or Test And Set, instruction on Amiga systems for quite some time. Historically, the 68000 implementation of the bus lock required by TAS has been a headache for many 68000 system designers, and Amiga engineers haven't been excluded from this. While the problem of TAS alone is an important issue, the evolution of official TAS support on Amiga systems further complicates the matter. This article won't give a clear answer to the "can I use TAS" question, since that answer changes based on the context of TAS's use. What I'm trying to do here is explain why TAS is a problem, what various Amiga systems do with TAS, and when TAS *might* be safely used by a Zorro II expansion board.

The Need For TAS?

The TAS instruction is designed to support hardware locked semaphores between multiple processors in a system. The idea is that making a read cycle, ALU operation, and write cycle an atomic operation will allow the 68000 to check on the value of a semaphore in memory and set this semaphore if currently unset, all without the possibility of another processor gaining access to the memory in between the read and write phases. Presumably, as long as shared structures in a multiprocessing system are all guarded by semaphores, and all semaphores are obtained under bus lock protection, then the possibility of uncontrolled simultaneous access of these shared resources is eliminated.

In the general case, this is called bus locking, and it's a perfectly valid concept. The problem stems from the way the 68000 designers chose to lock the bus. This TAS-generated bus lock becomes a special case 68000 cycle that's often not correctly supported by hardware. For this reason, TAS should *never* be used on Zorro II devices that aren't specifically designed to support it. As long as you're not dealing with multiprocessor systems, TAS isn't necessary. The remainder of the document will be concerned with multiprocessing systems; the cases in which TAS is very useful, if not actually necessary.

A 68000 Bus Locking Primer

As mentioned, the problem with the use of TAS is primarily based on the way the 68000 implement its bus locking. Unlike 68020 and 68030 processors, which drive a special Read-Modify Cycle signal (RMC*) to indicate bus locking, the 68000 runs a modification of the normal 68000 bus cycle.

The normal 68000 bus cycle starts with the assertion of addresses and the read strobe (R/W), followed shortly by the assertion of address strobe (AS*) and the data strobes (UDS* and LDS*). AS* is asserted during state S_2, while the data strobes are asserted during state S_2 for read cycles, state S_4 for write cycles. The 68000 drives data for write cycles just before the data

strobes, and samples data on read cycles on the falling edge of its clock at state S_6. In either case, it samples the data transfer acknowledge strobe (DTACK*) on the falling edge of state S_4, and inserts wait states until DTACK* is asserted. Once DTACK* is sampled, the cycle concludes with AS*, UDS*, LDS*, and most other signals being negated during S_7 or shortly thereafter. *Figure 1* illustrates a standard 68000 read and write cycle.

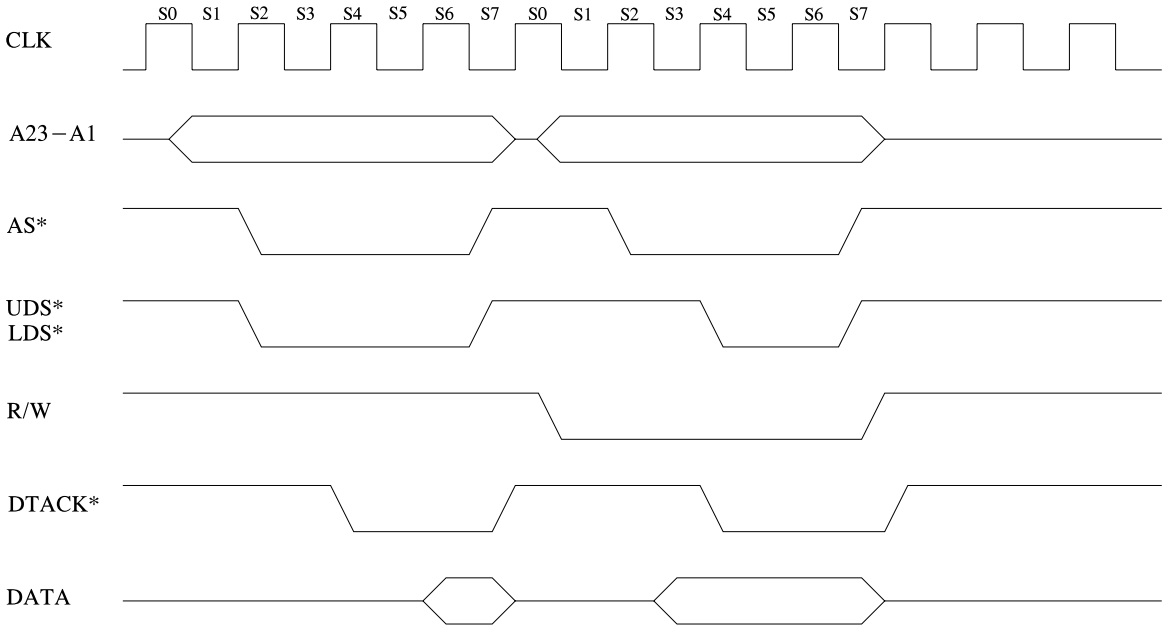

Figure 1: Standard 68000 Read and Write

The above read and write cycle is quite sufficient for handling semaphores in a single processor system, if supported within a single instruction. The 68000 family CPUs don't permit interrupts to be serviced except at cycle boundaries. So the 68000 bit test instructions are adequate to support software semaphores in an interrupt driven multitasking single processor system. Bus arbitration, on the other hand, is managed at cycle boundaries for the most part. Ordinary I/O DMA devices such as hard disks don't interfere with this locking either, since such DMA devices don't access system structures. All 680x0 locked cycles, including TAS, complete atomically even in the presence of a pending DMA request, so true bus locking isn't strictly necessary to handle semaphores between DMA processors using normal 680x0 bus arbitration to share memory. A bus locking mechanism is, however, the proper way to handle resource locking of sturctures a cycle shared mailbox memory, such as the shared memory that's implemented on Amiga BridgeCards.

The 68000 TAS cycle works much like a normal read followed by a normal write, except that AS* is not negated between the read and write portions of the compound cycle. Bus devices that expect AS* to qualify cycle boundaries, or devices that anticipate a minimum delay between the assertion of DTACK* and the negation of AS*, will fail with TAS cycles. Both AS* and the data strobes must be used to identify the subcycle boundaries within the full TAS cycle. The bus address doesn't change throughout the cycle, while the R/W line and data bus obey the timing of the individual subcycles. DTACK* and other slave device signals will similarly respect the

timing of the individual subcycles, as based on UDS* or LDS*. Additionally, the TAS cycle contains extra clock cycles for the ALU operation, so that instead of the minimal eight clocks for a read cycle immediately followed by a write cycle, the minimum TAS cycle takes ten clocks. *Figure 2* illustrates a 68000 TAS cycle.

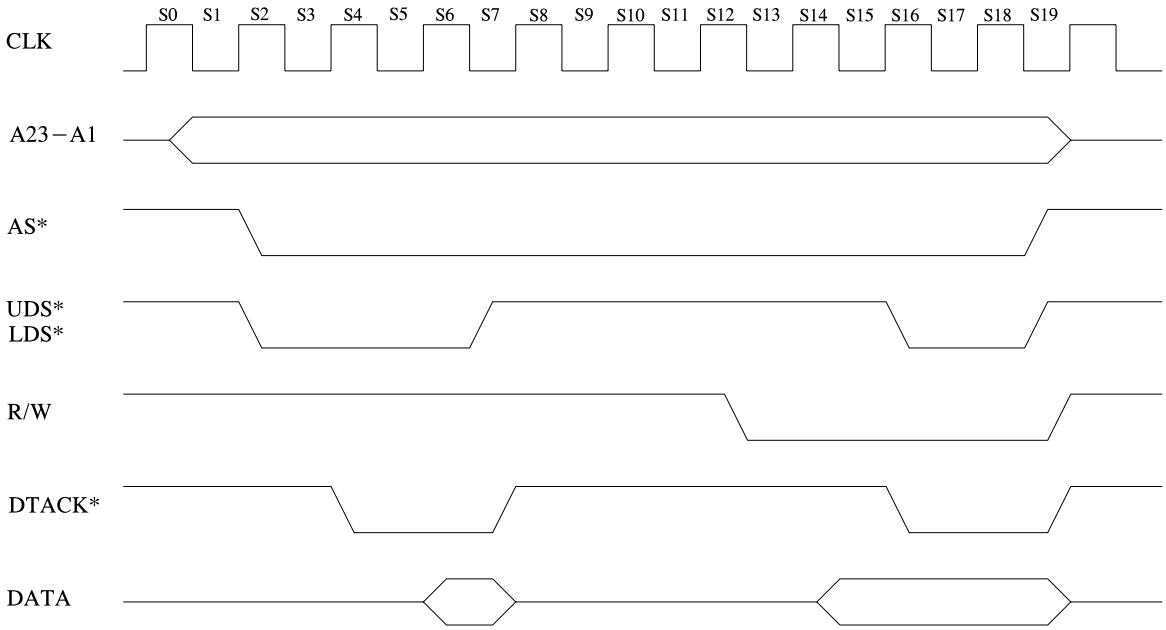

Figure 2: 68000 Read-Modify-Write

Zorro II Bus Issues

The Zorro II Expansion Bus, being originally based on the 68000 CPU bus, by extension can run TAS cycles. However, their use on the Zorro II bus is far from straight forward. Although the original Zorro II specifications, and the updated specifications published in the *A500/A2000 Technical Reference Manusl* both require a 68000 Refence Manusl as a companion specification, the use of TAS was never explicitly discussed for use on the bus. Additionally, the use of TAS in Amiga Chip memory was explicitly forbidden, and at the time of the Zorro II specification writing, generally considered unsuppored anywhere on Amiga computers.

Because of this ambiguity, many Zorro II devices don't support TAS cycles at all. Therefore, the use of TAS is explicitly forbidden on any Zorro II device that isn't specifically known to support TAS. Even simple memory boards probably act unpredictably when driven with a TAS cycle.

Even circuits that support TAS must be very carefully designed to support it properly. Due to clock skewing between the 68000 and the backplane, or alternate bus masters and the backplane, no Zorro II slave PIC that controls DTACK*, either via OVR* or XRDY, must assume it knows the delay between its assertion of DTACK* and the bus master's negation of the data strobes. The end of a Zorro II cycle is defined by the negation of AS*, and nothing else. The end of the read half of the TAS cycle is defined by the negation of UDS*/LDS*, and nothing else. Zorro II bus controllers in the A2000 and A3000 are designed with this in mind. For designs in which the

DTACK* assertion to end of cycle time is critical, the best practice is to assert XRDY or DTACK* on the rising edge of the 7MHz Zorro II bus clock. The strobes must still be considered to end the cycle, but this does minimize the effect of any clock skews that may arise in the system.

TAS and the A2500

At the time the A2620 and A2630 were designed, the use of TAS on the expansion bus was still considered a dubious practice. While TAS can, within the limits setforth in this document, be used successfully on some Amiga system, it is not completely supported in either A2500 configuration. As mentioned, the TAS cycle performs two major functions. The first, which is supported by the A2500, is to provide an atomic cycle that will arbitrate a semaphore between DMA devices on the bus. With A2500s, TAS is guaranteed to be atomic with respect to bus arbitration. However, it does not generate 68000 compatible TAS cycles on the expansion bus. The TAS instruction executed on either A2500 system will result in a standard 68000 read cycle followed by a standard 68000 write cycle. Thus, there is no bus locking, so TAS won't be sufficient to arbitrate semaphores in shared mailbox type memory.

The best solution for such mailbox arbitration is to use software based spin lock semphores or other methods which permit safe multiprocessing semaphores without the hardware bus locking. If that's unacceptable for all Amigas, the TAS solution can be invoked on Amiga systems that support it, while the software-only solution can be invoked on A2500s. Software can determine the existence of an A2500 by querying for an A2500 via the expansion library. A2500s show up as Commodore products (manufacturer number $0202) numbers $50 (for A2500/20) or $51 (for A2500/30).

TAS and the A3000

The A3000 does support 68000 compatible TAS cycles on its implementation of the Zorro II bus, but there are some caveats that must be considered.