

INST Pin Failure Following DMA Sequences with the UT80CXX196KD

Table 1: Cross Reference of Affected Product

| Product Name: | SMD #: | Device Type: | Internal PIC Number: |
|---------------|------------|--------------|----------------------|
| UT80CRH196KD | 5962R98583 | 01 | JD02A through JD02D |
| UT80C196KD | 5962-98583 | 01 | JD12A through JD12D |

1.0 Overview

This product errata is provided by Aeroflex UTMC to describe the INST pin's failure to assert when DMA arbitration is used in an UT80CXX196KD based design. Any user that intends to couple the DMA (aka., Bus Hold) functionality provided by the UT80CXX196KD with an overlapped instruction/data memory architecture, should use this errata to guide their circuit design and software development efforts.

Specifically, when DMA is employed in a circuit design, there exists a number of scenarios where the INST pin will not be active (e.g. logic high) during instruction fetch immediately following the exit from the DMA sequence. Although Aeroflex UTMC has identified a hardware only work-around, there still exists a very small possibility that the INST failure will occur. The only 100% guaranteed method of avoiding this DMA/INST failure is by coupling the hardware implementation with an appropriately designed software architecture.

2.0 DMA/INST Failure Discussion

The expected operation of a DMA sequence on the UT80CXX196KD is as follows:

1. A DMA arbiter will request the memory bus from the UT80CXX196KD by driving the hold request (HOLD) signal low.
2. When the UT80CXX196KD finds a suitable stopping point, it will drive the hold acknowledge (HLDA) signal low and subsequently tri-state all relevant memory signals.
3. The UT80CXX196KD will also drive the bus request (BREQ) signal whenever it determines that an external bus access is required. The assertion of the BREQ signal can occur simultaneously with the assertion of the HLDA signal or anytime during a falling CLKOUT edge within a DMA sequence.
4. When the DMA arbiter senses an active HLDA signal, it should provide a DMA acknowledge signal to the device requesting the bus.
5. When the requesting device gains access to the memory bus, as approved by the DMA acknowledge, it will take control of all relevant memory signals and perform the necessary memory accesses.
6. When this memory access is complete, the previously requested device will tri-state its memory bus control signals, and remove its bus request signal.
7. When the DMA arbiter identifies the requesting device has relinquished the memory bus, it will subsequently remove the HOLD signal to the UT80CXX196KD by driving it high.
8. Upon the deactivation of the HOLD signal, the UT80CXX196KD will remove the HLDA and BREQ signals, and subsequently take control of the memory bus to perform a queued data or instruction memory access.

Depending upon a number of critical events that can occur within the UT80CXX196KD during a DMA sequence, the instruction fetch performed by the UT80CXX196KD following the exit from the DMA sequence may not work correctly. This failure mechanism exhibited by the UT80CXX196KD is characterized by an instruction fetch without an active INST pin (logic low). For all but one case, the occurrence of this failure depends on the instance during the DMA sequence that the UT80CXX196KD drives the BREQ signal low, and the subsequent number of CLKOUT rising edges occurring before the HOLD signal is removed by the DMA arbiter. Both the UT80CXX196KD's assertion of the BREQ signal, and its determination of whether an instruction or data access occurs following a DMA sequence, is based on a three-tiered memory access prioritization sequence. Table 2 below describes these prioritization scenarios.

Table 2: Memory Access Prioritization Summary

| Internal Signal | Description |
|----------------------------|---|
| Instruction Fetch Priority | Highest priority memory access control signal. When active the next memory access will be an instruction fetch. |
| Data Access | The middle priority memory access control signal. If a data access is pending and the Instruction Fetch Priority signal is not active, the next memory access will be a data cycle. |
| Instruction Fetch Request | The least significant memory access control signal. As long as the Instruction Fetch Priority and Data Access signals are not active, the next memory cycle will be an instruction fetch. |

2.1 Cases Resulting in a Successful Instruction Fetch Following a DMA Sequence:

The Instruction Fetch Priority signal will always be active to the UT80CXX196KD memory controller whenever the instruction queue is less than half full. If the Instruction Fetch Priority signal activates during a DMA cycle, the UT80CXX196KD will always perform a proper instruction fetch upon the completion of the DMA sequence. For this scenario, no failures will occur. Similarly, if the DMA sequence concludes without a pending data access, and between 4 and 7 bytes are loaded into the instruction queue, the UT80CXX196KD will again successfully fetch an instruction due to the active Instruction Fetch Request signal. Lastly, if the Data Access signal is active and neither the Instruction Fetch Priority nor the Instruction Fetch Request are active, then a data read or write will occur upon the conclusion of the DMA sequence.

2.2 Cases Resulting in a Failed Instruction Fetch Following a DMA Sequence:

However, for the scenario where the Data Access and the Instruction Fetch Request signals are active, the ensuing memory cycle will be either a successful data access or an instruction fetch with the INST pin being driven low by the UT80CXX196KD. Specifically, the internal control logic of the Data Access signal in the UT80CXX196KD causes the Data Access signal to pulse between active and inactive at every falling edge of the CLKOUT signal. Therefore, if the Data Access signal is active when the UT80CXX196KD exits a DMA cycle, a successful data memory read or write will occur. However, if the Data Access signal pulses inactive as the UT80CXX196KD exits the DMA cycle, an instruction fetch will occur, but the INST pin will have been gated off by the active Data Access signal during the previous clock cycle.

In order to work around this failure scenario from a hardware perspective, the DMA arbiter should remove the HOLD signal such that the UT80CXX196KD exits the DMA cycle with valid memory access control signals defined. In order to properly time the removal of the HOLD signal, the DMA arbiter must pay close attention to the activation timings of the HLDA and BREQ signals. In short, the DMA arbiter can prevent a failure by accounting for two key cases. The first case is when the HLDA and BREQ signals are simultaneously activated by the UT80CXX196KD. The second case is when the HLDA signal activates first, followed sometime later by the UT80CXX196KD's activation of the BREQ signal.

2.3 Failure Scenarios Preventable Solely Through Hardware Implementation:

For the first case, the DMA arbiter should drive the HOLD signal low for an odd number of CLKOUT rising edges. Figure 1 shows an example signal diagram that would satisfy the first failure scenario. Because the HOLD signal is held active for the correct number of states times, the Data Access signal is inactive during the DMA exit sequence and therefore does not prevent the INST pin from activating. Following the same exit philosophy, Figure 2 shows the signal diagram for the scenario where the BREQ signal activates at least one state time after the HLDA signal activates. In this second scenario, the HOLD signal should be held active for an even number of CLKOUT rising edges following the assertion of the BREQ signal. Again, this timing sequence would result in the Data Access signal being inactive upon the removal of the HLDA signal, thus allowing the INST signal to activate properly during the subsequent instruction fetch.

2.4 Failure Scenarios NOT Preventable Solely Through Hardware Implementation:

There is, however, a case where the $\overline{\text{HLDA}}$ and $\overline{\text{BREQ}}$ signals will activate simultaneously (similar to the first case) but the $\overline{\text{HOLD}}$ signal must be held low for an even number of CLKOUT rising edges (similar to the second case). This scenario occurs when the UT80CXX196KD completes loading the 8th byte of the instruction queue while it simultaneously asserts the $\overline{\text{HLDA}}$ signal and internally decides to request an external data access. Unfortunately, the behavior and timing associated with all of the external signals on the UT80CXX196KD does not offer any insight into this situation. Consequently, there is no hardware-only work-around to protect against this last scenario. As a result, implementing a DMA arbiter that protects against the two failure scenarios presented earlier is not sufficient. Instead, a complimentary hardware and software work-around must be employed to establish 100% protection from a DMA/INST failure.

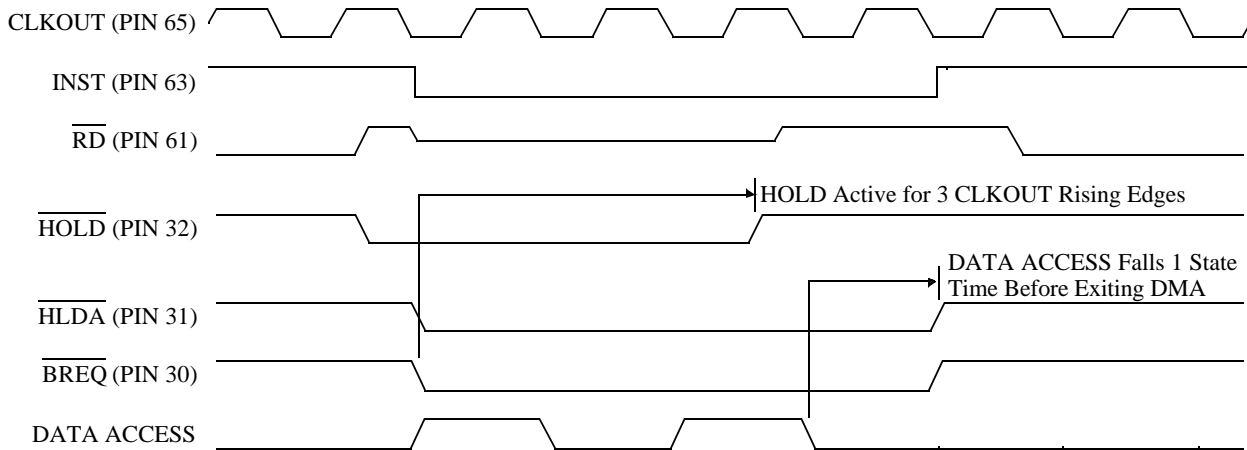


Figure 1. Signal Diagram for Case of $\overline{\text{HLDA}}$ and $\overline{\text{BREQ}}$ Simultaneous Activation

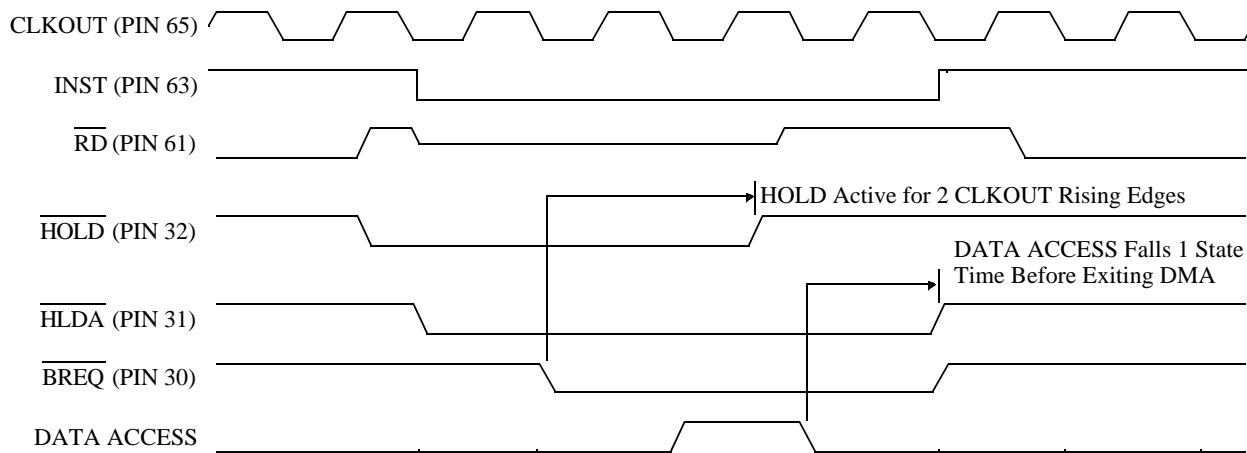


Figure 2. Signal Diagram for Case of $\overline{\text{BREQ}}$ Delayed Activation from $\overline{\text{HLDA}}$

3.0 DMA/INST Failure Work-Around

This section of the errata discusses two solutions to working-around the instruction fetch failures caused by the UT80CXX196KD’s DMA arbitration exit behavior. The first solution works for all cases, but it places a large dependence on the software programmer’s consistent implementation and linking of the application software. If the software is not properly linked and segregated in accordance with the following solution, the frequency and probability of failure increases significantly. In order to maximize the probability of success and protect against errors resulting from the programmer’s software implementation, Aeroflex UTMC recommends that both work-arounds be implemented by the software and hardware designers.

3.1 DMA/INST Failure Solution:

This solution requires a coordinated hardware and software implementation. From a hardware perspective, the system memory map should provide an area of memory that will always decode to the instruction memory space independent of the state of the INST pin. This area of memory must be large enough to hold all of the application software that will be run while DMA is active.

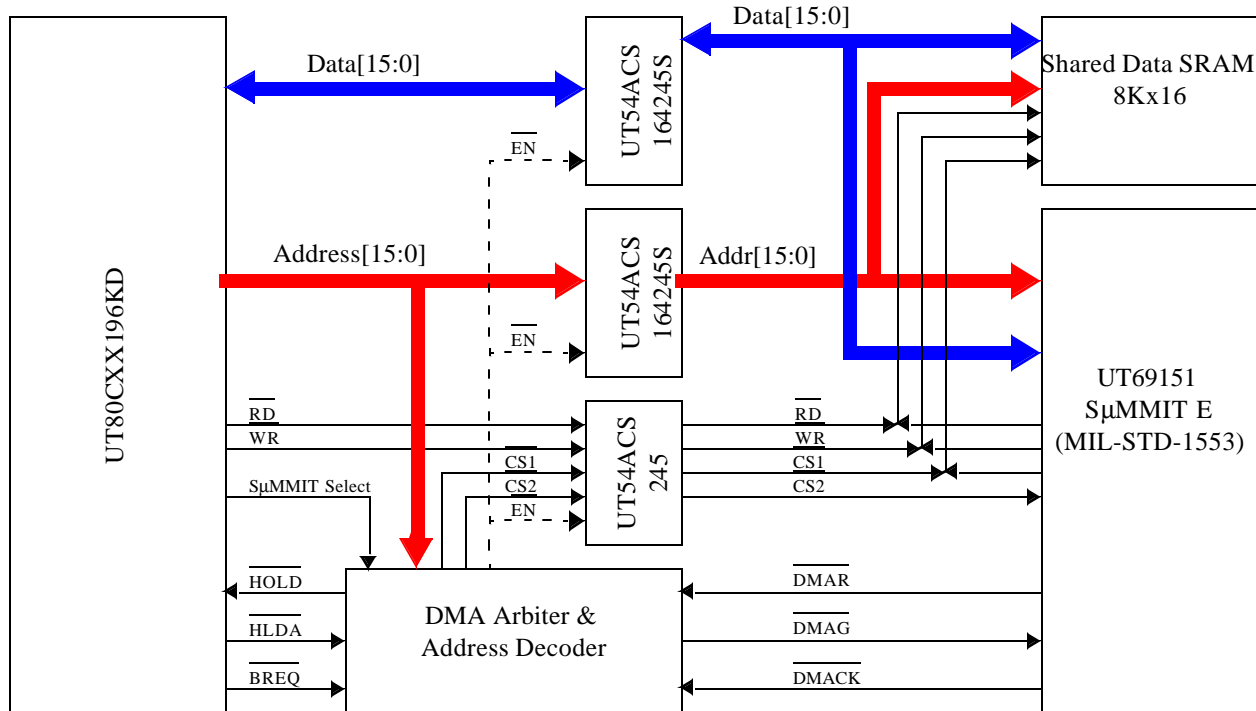


Figure 3. Isolation Scheme Between UT80CXX196KD and SμMMIT™ E

3.2 Hardware Implementation:

In addition to dedicating an instruction memory area, the system design shall include the ability to isolate the UT80CXX196KD from all other DMA capable devices. An example of this isolation is depicted in Figure 3. In this example, we show a system where the UT80CXX196KD will access either the SμMMIT™ or the SμMMIT™’s memory whenever it asserts the discrete output -- SμMMIT Select. The logic block containing the DMA Arbiter and the Address Decoder will monitor the state of the SμMMIT Select signal to determine (a) whether DMA arbitration should be active, and (b) when the isolation buffers should be enabled. If the SμMMIT Select is inactive, the DMA Arbiter shall keep the HOLD signal inactive and constantly assert the DMAG signal to the SμMMIT. Generally speaking, both the UT80CXX196KD and SμMMIT will have total control of their respective memory busses whenever the SμMMIT Select is inactive. This capability is ensured because the Address Decoder is designed to disable the isolation buffers unless the SμMMIT Select is active and the UT80CXX196KD addresses either the SμMMIT or the SμMMIT™’s memory. As a result of this design technique, the application software is able to control the activation of the DMA arbiter. A capability that is paramount in avoiding a DMA/INST failure.

Once the DMA capable devices have been isolated from the UT80CXX196KD, the system designer must implement a memory map that supports the software controlled activation of DMA arbitration. Table 3 lists an example memory map that would support the system design shown in Figure 3. Applying the memory map below to the example design in Figure 3, you should notice two key areas. The first key area is the memory range of 400H through 207FH. Based on the memory map, the address decoder should always provide access to the instruction memory whenever the UT80CXX196KD addresses the memory range 400H through 207FH independent of the state of the INST and SμMMIT Select pins. For the software designer, all software that operates while the SμMMIT Select pin is active must be linked into this dedicated instruction memory range.

Table 3: Sample Memory Map For Figure 3

| Memory Segment: | Memory Segment Decode: | | |
|--------------------------|------------------------|-----------------------------|----------------|
| | INST | SμMMIT Select (Active High) | Address [15:0] |
| Instruction Memory | X | X | 400H-207FH |
| | 1 | X | 2080H-FFFFH |
| UT80CXX196KD Data SRAM | 0 | X | 2080H-3FFFH |
| | 0 | 0 | 4000H-7FFFH |
| | 0 | X | 8000H-FFFFH |
| SμMMIT Memory SRAM | 0 | 1 | 4000H-7FBFH |
| SμMMIT E Protocol Device | 0 | 1 | 7FC0H-7FFFH |

The second key area of memory is in the range of 4000H through 7FFFH. Whenever the UT80CXX196KD accesses data in this memory range, the state of the SμMMIT Select must be evaluated by the Address Decoder to determine what devices are being accessed by the UT80CXX196KD. If the SμMMIT Select is inactive (logic 0), then the UT80CXX196KD's data SRAM will be selected. If, however, the SμMMIT Select is asserted (logic 1), then the buffer enable will be activated and the appropriate chip select will be supplied to the SμMMIT side of the system. As soon as the Address Decoder enables the isolation buffers, the address bus from the UT80CXX196KD and its control signals will be supplied to the SμMMIT side of the system, and the data bus will be driven in the direction of either the UT80CXX196KD, if the RD is asserted, or in the direction of the SμMMIT if the WR signal is asserted.

Lastly, the DMA Arbiter should be enabled/disabled by the SμMMIT Select signal. Whenever the SμMMIT Select is active, the arbiter should be active independent of whether the UT80CXX196KD is actually addressing data memory in the range of 4000H-7FFFH. This important distinction ensures there will be no bus contention between the UT80CXX196KD and the SμMMIT sides of the system. As long as the DMA Arbiter is active before the UT80CXX196KD addresses the SμMMIT side of the system, it will prevent the UT80CXX196KD from making the access while the SμMMIT has control of the memory bus. On the flip side, when the SμMMIT Select is inactive, the DMA Arbiter will be disabled, and, therefore provide both sides of the system with complete control of their respective memory buses.

3.3 Software Implementation:

From a software perspective, all code that executes while the SμMMIT Select is active must reside within the dedicated instruction space (400H-207FH). In order to control linking of the appropriate code segments in this dedicated instruction area, the software engineer must create distinct object files and define their locations in the memory setup dialog box found in the Tasking EDE Linker Options. In order to create stand-alone object files, the software engineer must create self contained C or C++ source files that will be included with the entire project build. As an example, let's assume that a project includes the main file: "MY_MAIN.C", and two supporting source files: "SUMMIT_APPS.C" and "CONTROL.C". Now, if the "SUMMIT_APPS.C" contains all the code that will run whenever the SμMMIT Select is active, it must be linked in the address range of 400H-207FH. Figure 4 shows how the memory setup can be defined in the Tasking EDE Linker Options dialog box.

3.4 Summary of the DMA/INST Failure Solution:

By implementing all of the aforementioned system and software design recommendations, the DMA/INST failure will be avoided. However, it is critical that the software engineer identify all code segments, including those that control activation of the DMA Arbiter, and ensure that they are linked into the dedicated instruction memory areas. Because it is difficult to ensure that all of these code segments are linked correctly, Aeroflex UTMC recommends that the DMA Arbiter include the DMA/INST failure mitigation technique described in section 3.0.1 below.

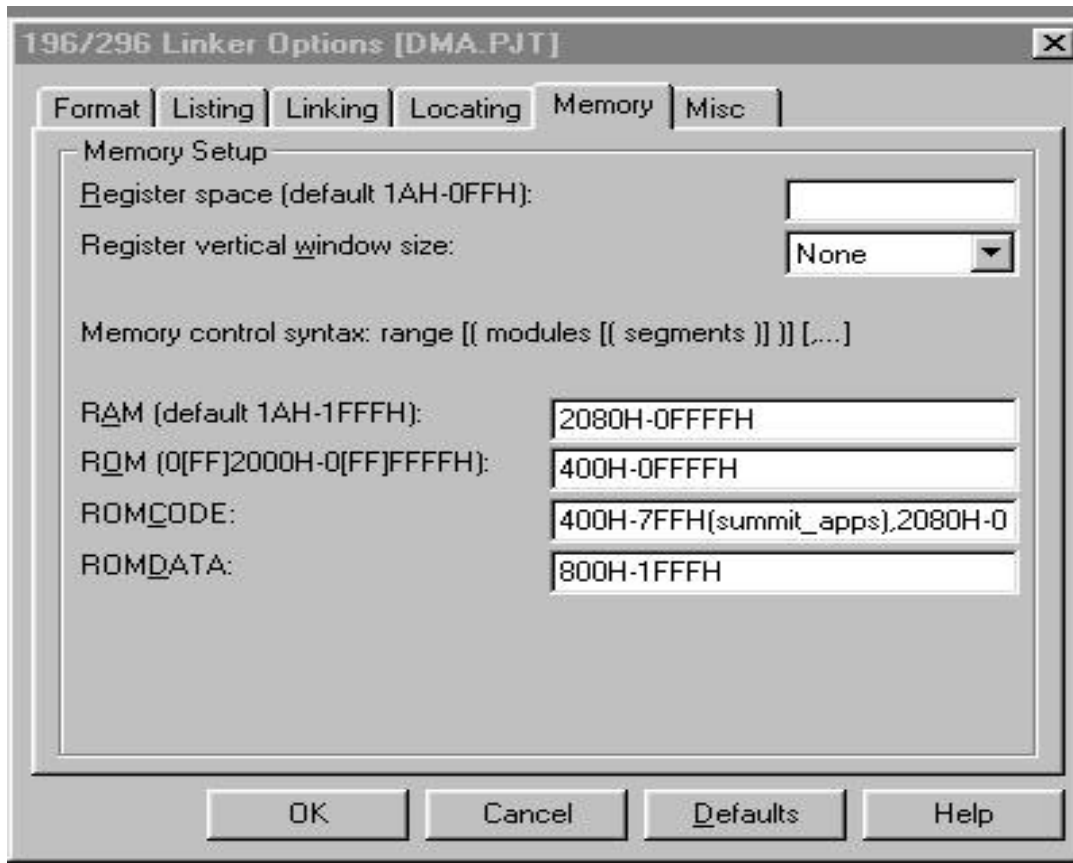


Figure 4. Memory Setup Defining Linker Location for SUMMIT_APPS Code Segment

3.0.1 DMA/INST Failure Mitigation Technique:

Although the following workaround does not provide a 100% protection against all the DMA/INST failure scenarios, it does serve as a great backstop to implementation errors from the master workaround described in section 3.1 above. This DMA/INST failure mitigation technique offers a complete solution to the first two failure scenarios described earlier in this errata (see section 2.3), but it does not solve the third scenario defined previously in section 2.4. Although the third scenario is not protected by this mitigation technique, it occurs at a sufficiently low frequency to ensure a high probability of success from the implementation of this technique.

The diagram in Figure 5 offers an example $\overline{\text{HOLD}}$ delay circuit that correctly delays the $\overline{\text{HOLD}}$ signal depending on the relationship between assertions of the $\overline{\text{HLDA}}$ and $\overline{\text{BREQ}}$ signals. The “Internal HOLD#” signal located in the diagram is generated by the standard DMA arbitration logic and should be run through a delay filter like the one shown in the following diagram. Referring to Figure 5, the delay filter should supply both the original and the one-clock delayed versions of the Internal HOLD# signal to a 2-to-1 multiplexor (as shown by the two tri-stateable buffers with complimentary enable signals). The output of this multiplexor, Delayed HOLD#, should be tied directly to the $\overline{\text{HOLD}}$ input on the UT80CXX196KD. The proper version of the Internal HOLD# signal is selected by the output of the toggle flip-flop located in the upper right hand portion of the diagram.

The toggle flip-flop is held in reset until the $\overline{\text{HLDA}}$ signal has been asserted for at least one rising clock edge and the $\overline{\text{BREQ}}$ signal is asserted low. Once taken out of reset, the toggle flip-flop will alternately select the appropriate version of the Internal HOLD# signal. The resulting behavior of the Delayed HOLD# signal is depicted in the simulation waveform shown in Figure 6. The simulation depicts both failure scenarios that are corrected by this mitigation technique. The first scenario is characterized by the $\overline{\text{HLDA}}$ and $\overline{\text{BREQ}}$ signals being driven simultaneously from the falling edge of the CLKOUT signal. After the UT80CXX196KD asserts these signals, the DMA Arbiter drives the Internal HOLD# signal active until the fourth rising CLK-

OUT edge is received. Although the Internal HOLD# signal is de-asserted, the current state of the toggle flip-flop ensures that the delayed version of the Internal HOLD# signal is selected for an additional rising clock edge. As a result, the Delayed HOLD# signal driven to the UT80CXX196KD remains active for an odd number of rising CLKOUT edges following the coincident assertion of the BREQ and HLDA signals.

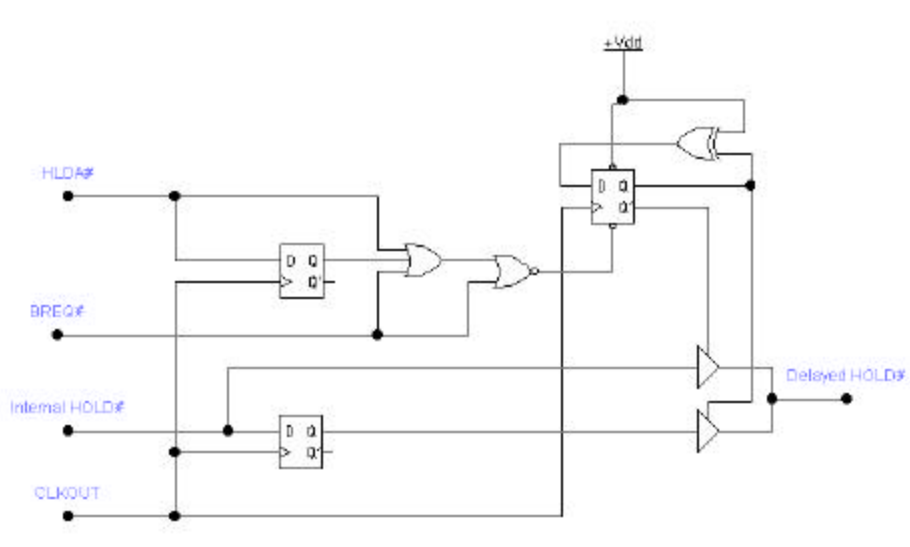


Figure 5. HOLD Delay Circuit

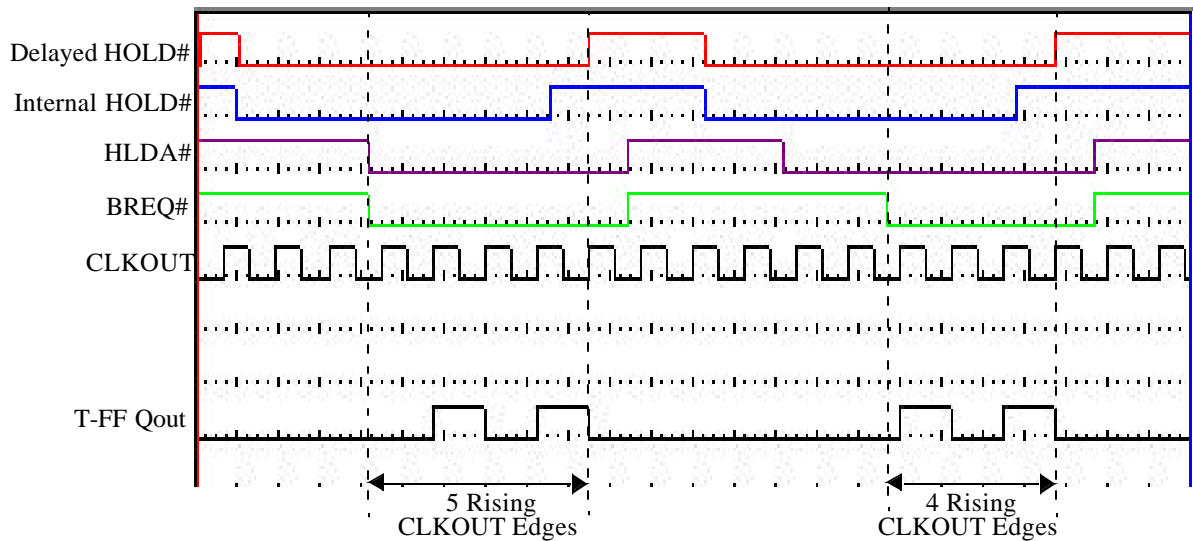


Figure 6. HOLD Delay Circuit Simulation

Similarly, the waveform shows the behavior of this mitigation technique applied to the second scenario, characterized by the BREQ signal activating on a non-coincident falling CLKOUT edge from the HLDA signal. Under this scenario, the HLDA signal would be active for at least one CLKOUT cycle before the BREQ signal is asserted and, therefore, the toggle flip-flop will be taken out of reset as soon as the BREQ signal is activated. Referring to the simulation waveform in Figure 6 the DMA Arbiter removes the Internal HOLD# signal on the third rising CLKOUT edge after BREQ is asserted. However, due to the state of the multiplexor select signal generated by the toggle flip-flop, the Delayed HOLD# signal is held for one additional CLKOUT rising edge resulting in an even number of rising CLKOUT edges occurring before the Delayed HOLD# is deasserted to the UT80CXX196KD.

In order to help you implement the DMA Arbiter and the recommended HOLD delay circuit, Aeroflex UTMC provides an ABEL HDL program for use in a UT22VP10 RADPAL that would work for the system design example depicted in Figure 3. This source file can be downloaded from UTMC's website at http://www.utmc.com/products/80196_examples.html.

4.0 Conclusion

Although the potential risk of the DMA/INST failure is highly dependent upon the system architecture and software implementation, Aeroflex UTMC recommends that all UT80CXX196KD based design using DMA, coupled with overlapping instruction and data memory, should include all solutions defined in section 3.0 of this errata. Aeroflex UTMC plans to supersede all UT80CXX196KD product (JDXXX) with a radiation qualified device in accordance with the following table:

Table 1: Cross Reference of Superseding UT80CXX196KD Product

| Product Name: | SMD #: | Device Type: | Internal PIC Number: |
|---------------|------------|--------------|----------------------|
| UT80CRH196KD | 5962R98583 | 02 | KC01X |
| UT80C196KD | 5962-98583 | 02 | KC11X |

The new product offering is scheduled for production by July 2002.