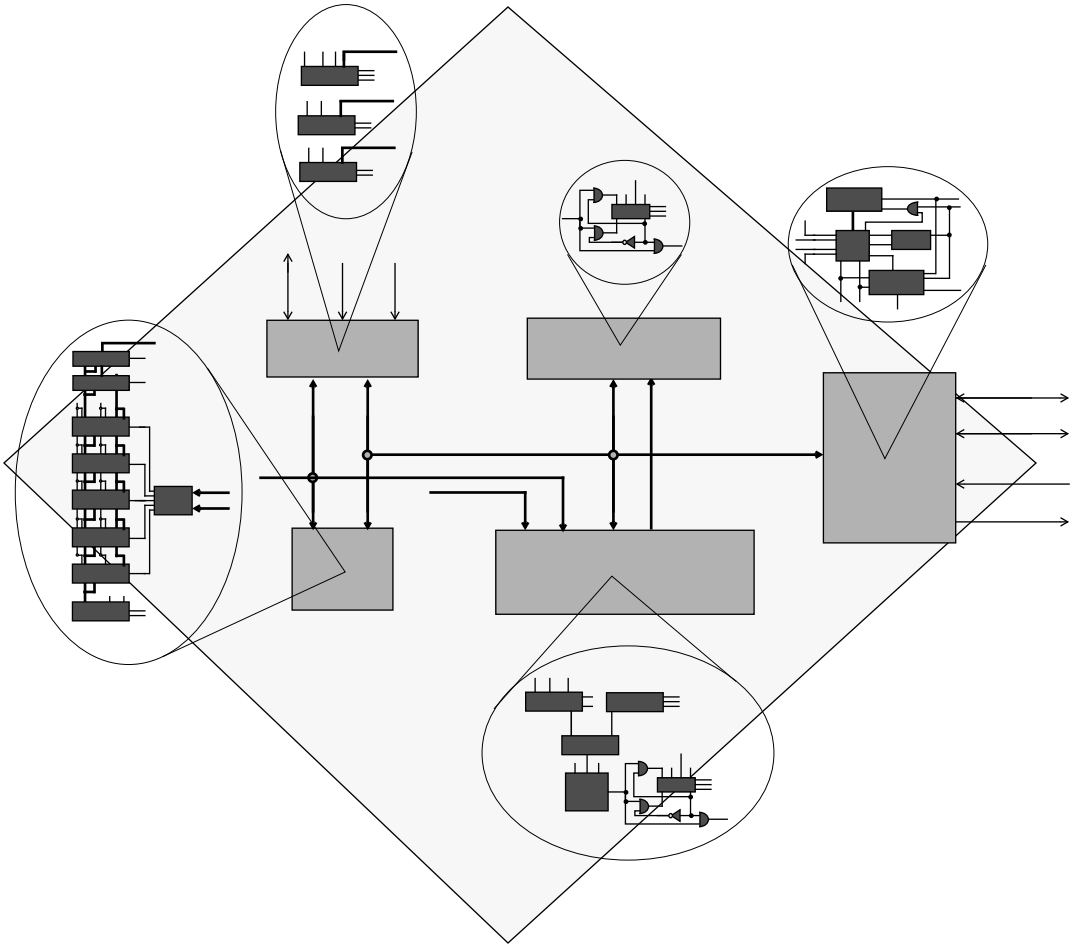
SECTION 10

ON-CHIP EMULATION (OnCE)



SECTION CONTENTS

10.1 INTRODUCTION

The purpose of this Section is to describe a set of circuits which will be used for hardware/ software emulation and debug on the DSP56100 family. OnCE provides a means of interacting with the DSP and any memory mapped peripherals non-intrusively so that a user may examine registers, memory or on-chip peripherals. To achieve this, special circuits and dedicated pins on the DSP are used to avoid sacrificing any user accessible on-chip resource. A key feature of the special OnCE pins is to allow the user to insert the DSP into his target system yet retaining debug control, especially in the cases of devices specified without external bus. The need for a costly cable which brings out the footprint of any chip on traditional emulator systems is eliminated.

Figure 10-1 illustrates a block diagram of the Emulation and test serial interface.

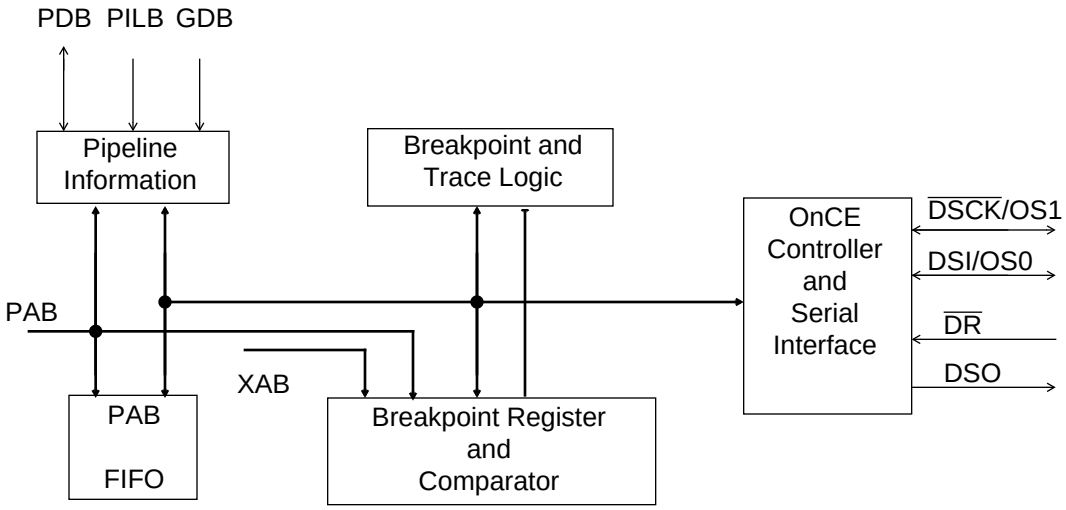
10.2 EMULATION AND TEST PINOUT

10.2.1 Debug Serial Input/OnCE Status 0 (DSI/OS0)

The DSI/OS0 pin, when input, is the pin through which serial data or commands are provided to the OnCE controller. The data received on the DSI pin is recognized only when the DSP has entered the debug mode of operation. Data is always shifted into the OnCE serial port most significant bit (MSB) first on the falling edge of the OnCE serial clock, DSCK. When an output, this pin in conjuction with the OS1 pin, provides information about the chip status when debug mode cannot be entered in response to an external request. The DSI/OS0 pin is an output when not in Debug Mode (i.e., until the acknowledge signal is issued to the Command Controller). When switching from output to input, the pin is three-stated. In order to avoid any possible glitches, an external pull-down resistor should be attached to this pin. During hardware reset, this pin is defined as an output and it is driven low.

10.2.2 Debug Serial Clock/OnCE Status 1 ($\overline{\text{DSCK}}$/OS1)

The $\overline{\text{DSCK}}$/OS1 pin, when an input, is the pin through which the serial clock is supplied to the OnCE controller. The serial clock provides pulses required to shift data into and out of the OnCE serial port. Data is shifted into the chip via the DSI pin on the falling edge of DSCK and is shifted out of the chip via the DSO pin on the rising edge of DSCK. When an output, this pin, in conjunction with the OS0 pin, provides information about the chip status when debug mode cannot be entered in response to an external request. The DSCK/OS1 pin is an output when not in Debug Mode (until the acknowledge signal is issued to the Command Controller). When switching from output to input, the pin is first three-stated. In order to avoid any possible glitches, an external pull-down resistor should be attached to this pin. During hardware reset, this pin is defined as output and it is driven low.



Note: PILB = Program Instruction Latch Bus

Figure 10-1 OnCE Block Diagram

Table 10-1 shows the status of the chip as a function of the two output pins OS0:OS1.

Table 10-1 Function of OS1:OS0

OS1	OS0	Status
0	0	Normal state
0	1	STOP or WAIT mode
1	0	DSP busy state (external accesses with wait state)
1	1	reserved

10.2.3 Debug Serial Output (DSO)

The DSO pin, while in debug mode, is the serial output that permits reading the data contained in one of the OnCE controller registers as specified by the last command received from the external command controller. Data is shifted out of the chip via the DSO pin on the rising edge of DSCK. An acknowledgment pulse will be sent on the DSO pin when:

1. the chip enters the OnCE mode (external, $\overline{DR}$, hardware breakpoint, software breakpoint or trace) to indicate that the chip is ready to accept OnCE commands. This pulse is 3T long.
2. a “do nothing” operation (no go, no exit) is selected to indicate that the input command register is ready to receive a new command. This pulse is 4T long.
3. the requested data (before a read) is available to indicate that the serial shift registers are ready to receive clocks to start transmitting data to the DSO pin. This pulse is 4T long.

4. the shift registers are ready to receive clocks to receive data (before a write) from the DSI pin. This pulse is 4T long.
5. the shift registers have finished shifting in the new data (after a write) to indicate that the input command register is now ready to receive new instruction. This pulse is 4T long.
6. an instruction has completed execution (go, no exit; repeat an instruction). This pulse is 4T long.

Data is always shifted out the OnCE serial port most significant bit (MSB) first on the rising edge of DSCK. When not in debug mode, the DSO pin is driven high. During hardware reset this pin is driven high.

10.2.4 Debug Request Input ($\overline{DR}$)

The $\overline{DR}$ input is an active low pin that provides a means of entering the debug mode of operation from the external command controller. This pin, when asserted, will cause the DSP to finish the current instruction being executed, save the instruction pipeline information, enter the debug mode and wait for commands to be entered from the debug serial input line.

10.3 ONCE CONTROLLER AND SERIAL INTERFACE

The OnCE Controller and Serial Interface contains the following blocks: input shift register, bit counter, OnCE decoder and the status/control register. Figure 10-2 illustrates a block diagram of the OnCE serial interface.

10.3.1 OnCE Input Shift Register (OISR)

The OISR is an 8-bit shift register that receives the serial data from the DSI line. The data is clocked into the register on the falling edge of the clock applied to the DSCK pin. After the 8th bit is received the OISR will stop shifting in new data. The latched data will be used as input for the OnCE Decoder. The data is always shifted into the OISR most significant bit (MSB) first.

10.3.2 OnCE Bit Counter (OBC)

The OBC is a 4-bit counter (0…15) associated with shifting in and out the data bits. The OBC is incremented by the falling edges of the DSCK. The OBC is cleared at reset and whenever the DSP acknowledges that the Debug Mode has been entered. The OBC supplies two signals to the OnCE Decoder: one indicating that the first 8 bits were shifted-in (so a new command is available) and the second indicating that 16 bits were shifted-in (the data associated with that command is available) or that 16 bits were shifted-out (the data required by a read command was shifted out).

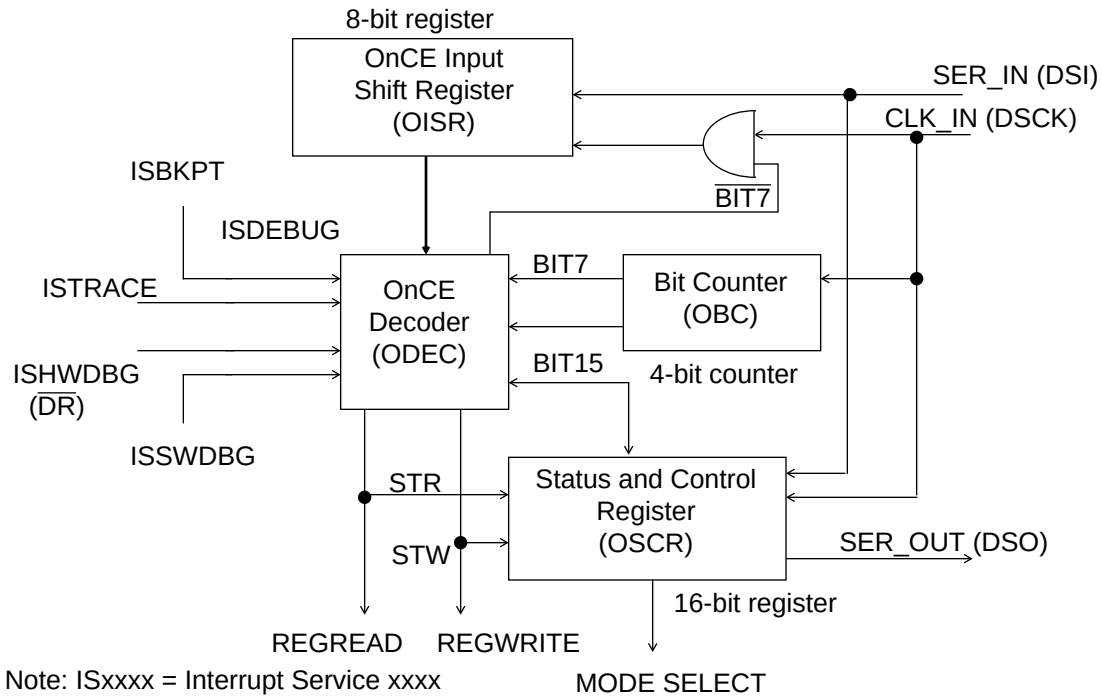


Figure 10-2 OnCE Controller and Serial Interface

10.3.3 OnCE Decoder (ODEC)

The ODEC is the supervisor of the entire OnCE activity. It receives as input the 8-bit command from the OISR, two signals from OBC (one indicating that 8 bits have been received and the other that 16 bits have been received), and one signal indicating that the DSP has halted. The ODEC generates all the strobes required for reading and writing the selected OnCE registers.

10.3.4 OnCE Status and Control Register (OSCR)

The (OSCR is a 16-bit register used to select the events that will put the chip in Debug Mode. Breakpoints may be disabled or enabled on one memory space. The Trace Mode of operation is also selected through OSCR.

OSCR is shown in Table 10-2 and the control bits are described in the following paragraphs.

Table 10-2 OnCE Status and Control Register (OSCR)

Status								Control							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
*	*	*	*	*	TO	HBO	SBO	*	*	*	TME	BS1	BS0	BE1	BE0

10.3.4.1 OSCR Breakpoint Enables (BE0-BE1) Bit 0-1

These control bits enable or disable the breakpoint logic and select the type of memory operations (read; write; access) upon which the breakpoint logic operates. These bits are cleared on hardware reset.

BE1	BE0	Selection
0	0	Breakpoint disabled
0	1	Breakpoint enabled on memory write
1	0	Breakpoint enabled on memory read
1	1	Breakpoint enabled on memory access

10.3.4.2 OSCR Breakpoint Selection (BS0-BS1) Bits 2-3

These control bits select if the Breakpoints will be recognized on program memory fetch, program memory access, X memory access or second X memory read. These bits are cleared on hardware reset.

BS1	BS0	Selection
0	0	Breakpoint on program memory fetch (fetch of the first word of instructions which are actually executed; not of those which are killed, not of those which are the second word of two-word instructions, and not of jumps which are not taken)
0	1	Breakpoint on any program memory access (any MOVEM instructions, fetches of instructions which are executed and of instructions which are killed, fetches of second word of two-word instructions, and fetches of jumps which are not taken
1	0	Breakpoint on first X memory (xab1) access
1	1	Breakpoint on second X memory (xab2) read (xab2 cannot be used to write data into the X memory)

The decoding scheme for BS(1:0) and BE(1:0) is as follows:

Function	BS(1:0)	BE(1:0)
disable	XX	00
program fetch program fetch program fetch	00 00 00	01 10 11
any program write or fetch any program read or fetch any program access or fetch	01 01 01	01 10 11
XAB1 write XAB1 read XAB1 access	10 10 10	01 10 11
disable XAB2 read XAB2 read	11 11 11	01 10 11

10.3.4.3 OSCR Trace Mode Enable (TME) Bit 4

This control bit, when set, enables the Trace Mode. When the Trace Mode is enabled, the chip will enter the Debug Mode whenever the execution of an instruction is completed and the Trace Counter is zero. This bit is cleared on hardware reset.

10.3.4.4 OSCR (Reserved) Bits 5-7

These bits are reserved for future use and read as zero. Reserved bits should be written as zero for future compatibility.

10.3.4.5 OSCR Software Breakpoint Occurrence (SBO) Bit 8

This read-only status bit is set when the debug mode has been entered by a DEBUG or DEBUGcc instruction. It is used by the external command controller to determine how the debug mode was entered. This bit is cleared when leaving the debug mode and is also cleared on hardware reset.

10.3.4.6 OSCR Hardware Breakpoint Occurrence (HBO) Bit 9

This read-only status bit is set when a OnCE hardware breakpoint occurs. It is used by the external command controller to determine how the debug mode was entered. This bit is cleared when leaving the debug mode and it is also cleared on hardware reset.

10.3.4.7 OSCR Trace Occurrence (TO) Bit 10

This read-only status bit is set when the debug mode of operation is entered from a decrement to zero of the trace counter and the trace mode has been armed. This bit is cleared on reset and when leaving the debug mode.

10.3.4.8 OSCR Reserved – Bits 11-15

These bits are reserved for future use and read as zero. Reserved bits should be written as zero for future compatibility.

10.4 OnCE BREAKPOINT LOGIC

Other processors traditionally set a breakpoint in program memory by replacing the instruction at the breakpoint address with an illegal instruction which causes a breakpoint exception. This technique is limiting in that breakpoints can only be set in RAM at the beginning of an opcode and not on an operand. Using such techniques, breakpoints can never be set in data memory.

On the other hand, by using address comparators, breakpoints may be set on program memory opcodes or any data memory location. This significantly increases the programmer's ability to monitor what the program is doing real-time.

The breakpoint logic can be enabled for Program memory breakpoints or for Data memory breakpoints. It contains an address latch, a register that stores the breakpoint address, a comparator and a counter. Figure 10-3 illustrates a block diagram of the OnCE Breakpoint Logic.

10.4.1 OnCE Breakpoint Logic Operation

The address comparator register is useful in halting a program at a specific point to examine/change registers or memory. Using the address comparator to set breakpoints enables the user to set breakpoints in RAM or ROM while in any operating mode.

The address comparator will cause a logic true signal when the comparison of its value is equal to the address on the bus. The breakpoint counter is then decremented if greater than zero. If the breakpoint counter is equal to zero, it is not decremented and a breakpoint occurs.

Conditional jump addresses produced by the instruction pipeline that are equal to the program address being monitored are only valid if the conditional jump instruction occurs, otherwise the conditional jump address is ignored. Program memory address breakpoints occur after the opcode or operand is executed and the breakpoint counter has been decremented to zero.

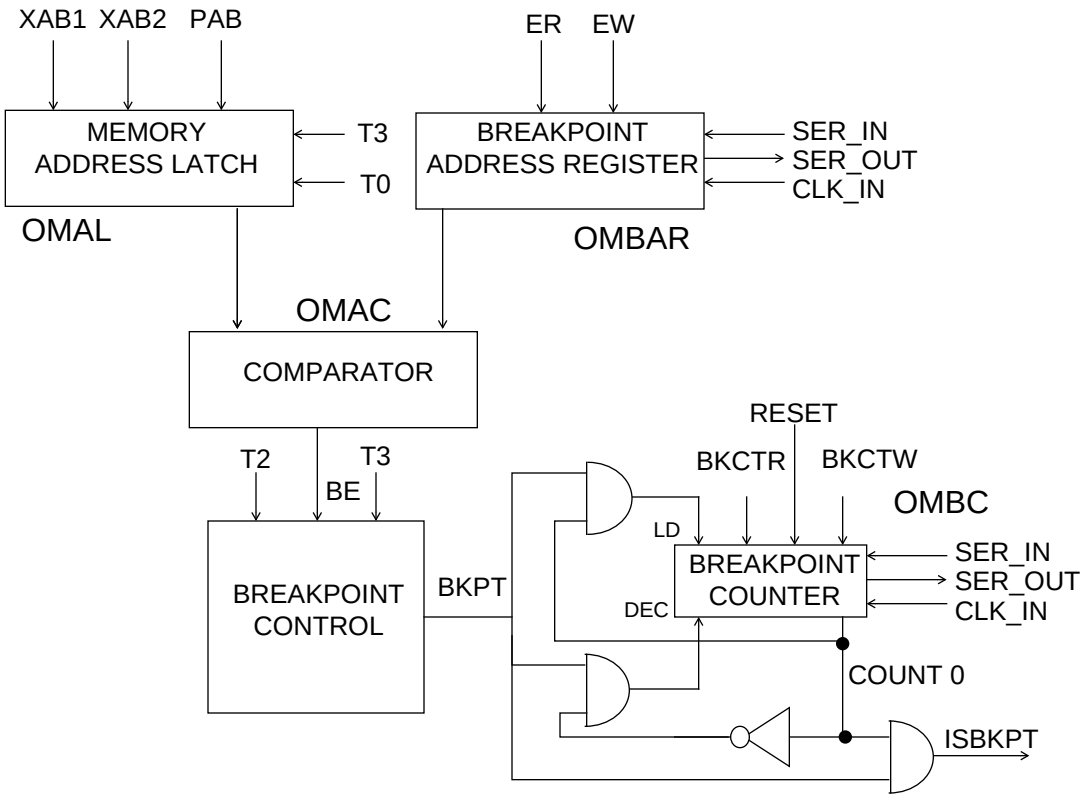


Figure 10-3 Breakpoint Logic

Data memory address breakpoints also occur after the execution of the instruction which formed the data memory address and the breakpoint counter has decremented to zero. The breakpoint registers are controlled by the debug status and control register (OSCR).

10.4.2 Breakpoint Counter

The breakpoint counter is a 16-bit counter that is useful for stopping at the nth iteration of a program loop or when the nth occurrence of a data memory access occurs. This information significantly decreases algorithm debug and provides a means of checking hot spots in program segments as well as peripheral or data memory accesses.

The breakpoint counter becomes a powerful tool when debugging real-time fast interrupt sequences such as servicing an A/D or D/A convertor or stopping after a specific number of host transfers have occurred. The breakpoint counter is cleared by reset.

10.4.3 OnCE Memory Address Latch (OMAL)

The Memory Address Latch (OMAL) is a 16-bit register that latches the PAB, XAB1, or XAB2 on every cycle.

10.4.4 Memory Breakpoint Address Register (OMBAR)

The Memory Breakpoint Address Register (OMBAR) is a 16-bit register that stores the memory breakpoint address. OMBAR is available for read/write operations only through the OnCE serial interface. Before enabling breakpoints, OMBAR must be loaded by the command controller.

10.4.5 Memory Address Comparator (OMAC)

The Memory Address Comparator (OMAC) is a 16-bit comparator that compares the current memory address (stored by OMAL) with Memory Address Register (OMBAR). If OMAC is equal to OMAL then the comparator delivers a signal indicating that the breakpoint address has been reached.

10.4.6 Memory Breakpoint Counter (OMBC)

The Program Memory Breakpoint Counter (OMBC) is a 16-bit counter which is loaded with a value equal to the number of times minus one that a program or data memory address should occur before a breakpoint is acknowledged. On each occurrence the counter is decremented. When the counter has reached the value of zero and a new occurrence takes place, a signal is generated and, if breakpoints are enabled in OSCR, the chip will enter the Debug Mode. OMBC is available for read/write operations only through the OnCE serial interface. Before enabling Memory Breakpoints, OMBC must be loaded by the command controller.

10.5 TRACE/STEP MODE

When in the special trace mode, the DSP will not cause an interrupt exception but instead will enter the debug operation mode and wait for further instructions from the debug serial port. Single or multiple instructions can be traced.

10.5.1 Trace Counter

The trace mode has a 16-bit counter associated with it so that more than one instruction may be executed before returning back to the debug mode of operation. The objective of the counter is to allow the user to take multiple instruction steps in real-time with no interference from the debug mode. This feature helps the software developer debug sections of code which do not have a normal flow or are getting hung up in infinite loops. The trace counter also enables the user to debug areas of code which are time critical.

To enable the trace mode of operation the counter is loaded with a value, the program counter is set to the start location of the instruction(s) to be executed real-time, the trace mode is selected in the debug status register (OSCR) and the DSP exits the debug mode by executing the appropriate command issued by the external command controller. Upon

exiting the debug mode the counter is decremented after each execution of an instruction. Interrupts are serviceable and all instructions executed including fast interrupt services will decrement the trace counter. Upon decrementing to zero the DSP will re-enter the debug mode, the trace occurrence bit in the debug status/control register (OSCR) will be set and the debug serial output pin DSO will be toggled to indicate that the DSP OnCE port is requesting service.

Note: The trace count should be loaded with one less than (i.e., N-1) the number of instructions that the user wants to execute (e.g., to single step one instruction, the trace counter is loaded with a zero).

The Trace counter is cleared by hardware reset. Figure 10-4 illustrates a block diagram of the Trace Counter logic.

10.6 METHODS OF ENTERING THE DEBUG MODE

Entering the Debug Mode is acknowledged by the chip by toggling the DSO line for 3 T cycles. This informs the external command controller that the chip has entered the Debug Mode and is waiting for commands. There are seven ways in which the Debug Mode may be entered. They are:

1. External Request During Hardware Reset
2. External Request During Normal Activity
3. External Request During STOP
4. External Request During WAIT
5. Software Request During Normal Activity
6. Enabling Trace Mode
7. Enabling Breakpoints

10.6.1 External Request During Hardware Reset

Holding the $\overline{\text{DR}}$ line asserted during the assertion of $\overline{\text{RESET}}$ will cause the chip to enter the Debug Mode. After receiving the acknowledge, the command controller must deassert the $\overline{\text{DR}}$ line. Note that in this case the chip does not perform any fetch or memory access before entering the Debug Mode.

10.6.2 External Request During Normal Activity

Holding the $\overline{\text{DR}}$ line asserted during the normal chip activity will cause the chip to finish execution of the current instruction and then enter the Debug Mode. After receiving the

acknowledge the command controller must deassert the $\overline{DR}$ line. Note that in this case the chip completes execution of the current instruction and stops after the newly fetched instruction enters the instruction latch. This process is the same for any newly fetched instruction including instructions fetched during interrupt processing or instructions that will be killed by the interrupt processing.

10.6.3 External Request During STOP

Asserting $\overline{DR}$ when the chip is in the stop state (i.e., it has executed a STOP instruction) causes the chip to exit the stop state and enter the Debug Mode. The chip will wake up from the stop state normally (finish executing STOP) and halt after the next instruction enters the instruction latch. After receiving the acknowledge, the command controller must deassert $\overline{DR}$. Note that in this case the chip completes the execution of the STOP instruction and halts after the next instruction enters the instruction latch.

10.6.4 External Request During WAIT

Asserting $\overline{DR}$ when the chip is in the wait state (i.e. has executed a WAIT instruction) causes the chip to exit wait state and enter the Debug Mode. The chip will wake up from the wait state normally (finish executing WAIT) and halt after the next instruction enters the instruction latch. After receiving the acknowledge, the command controller must deassert $\overline{DR}$. Note that in this case the chip completes execution of the WAIT instruction and halts after the next instruction enters the instruction latch.

10.6.5 Software Request During Normal Activity

Upon executing the DEBUG or DEBUGcc instructions (with condition true for DEBUGcc), the chip will enter Debug Mode after the instruction following the DEBUG/DEBUGcc instruction has entered the instruction latch.

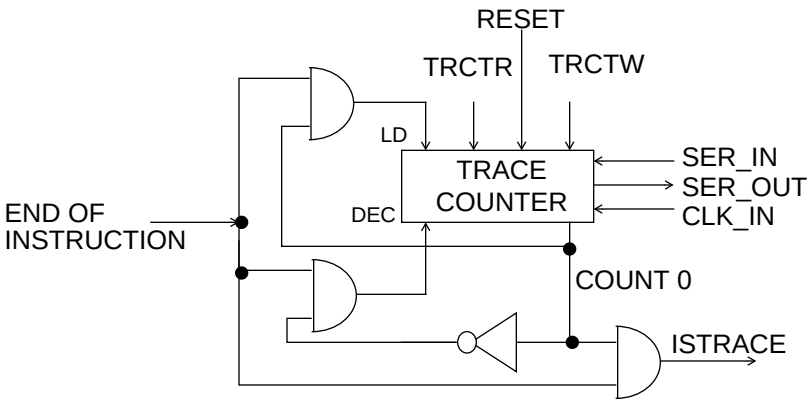


Figure 10-4 Trace Counter Logic

10.6.6 Enabling Trace Mode

When the chip is operating in Trace Mode and the Trace Counter reaches a value of zero, the chip will enter the Debug Mode **after** completing execution of the instruction that caused the Trace Counter to decrement. Only those instructions that are actually executed may cause the Trace Counter to decrement i.e. a killed instruction (instruction discarded during the interrupt process) will not decrement the Trace Counter and will not cause the chip to enter the Debug Mode.

10.6.7 Enabling Breakpoints

The chip will enter the Debug Mode **after** completing execution of the instruction that caused the Breakpoint Counter to decrement when:

1. operating in the Trace Mode when the Breakpoint Counter has reached zero

or

2. when operating in Normal Mode with the Breakpoint mechanism enabled and the Breakpoint Counter has reached zero.

In the case of **breakpointing on:**

1. **Program memory addresses**, the breakpoint will be acknowledged immediately after the execution of the instruction accessed at the specified address.
2. **Data memory addresses** the breakpoint will be acknowledged after the completion of the instruction following the instruction that caused the access at the specified address.

10.7 PIPELINE INFORMATION

The previous chip pipeline state must be reconstructed to resume normal chip activity when returning from the Debug Mode. Figure 10-5 illustrates a block diagram of Pipeline Information Registers. Only the PDB register and the PIL register are used to reconstruct the pipeline as it was before debug. the PAB History Buffer, PAB Register for Fetch and PAB Register for Decode are only used for status information. When loading a one word instruction into the PDB and issuing a GO command, the hardware internally transfers the PDB to the PIL and then executes the instruction. When loading a two word instruction, the first word is loaded into the PDB. As the second word is loaded to the PDB, the first word is automatically transferred to the PIL and then execution takes place.

10.7.1 OnCE PDB Register (OPDBR)

The PDB Register (OPDBR) is a read/write, 16-bit latch that stores the value of the Program Data Bus generated by the last Program Memory access of the DSP before the Debug Mode is entered. OPDBR is available for read/write operations only through the serial interface. This register is affected by the operations performed during the Debug Mode and must be restored by the command controller when returning to normal mode.

10.7.2 OnCE PIL Register (OPILR)

The OPILR is a read only 16-bit latch that stores the instruction present in the Instruction Latch when the Debug Mode is entered. OPILR is available for read operations only through the serial interface. If a write is selected for this register, i.e., R/$\overline{W}$ = 0 and RS4-RS0 = 01011, then zeros will be shifted into the OPILR. This register is affected by the operations performed during the Debug Mode and must be restored by the command controller when returning to normal mode. Since there is no direct write access to this register, this task is accomplished by writing the OPDBR first and then the data from OPDBR is latched in OPILR.

10.7.3 OnCE GDB Register (OGDBR)

The OGDBR is a read only 16-bit latch that stores the value of the Global Data Bus. OGDBR is available for read operations only through the serial interface. OGDBR is required as a means of passing information between the chip and the command controller. OGDBR will be mapped on the X internal IO space at address $FFFF. Whenever the command controller needs information such as a register or memory value it will force the chip to execute an instruction that brings that information to the OGDBR. Then, the contents of the OGDBR will be delivered serially to the command controller by the command “READ GDB REGISTER”.

10.8 PAB HISTORY BUFFER

To ease the debugging activity and keep track of the program flow, a First-In-First-Out, read only, buffer is provided. It stores the addresses of the last five instructions that were executed as well as the addresses of the last fetched instruction and of the instruction currently in the instruction latch.

Figure 10-6 illustrates a block diagram of the Program Address Bus FIFO.

10.8.1 OnCE PAB Register for Fetch (OPABFR)

The OPABFR is a read only 16-bit latch that stores the address of the last instruction that was fetched before the Debug Mode was entered. OPABFR is available for read operations only through the serial interface. This register is not affected by the operations performed during the Debug Mode.

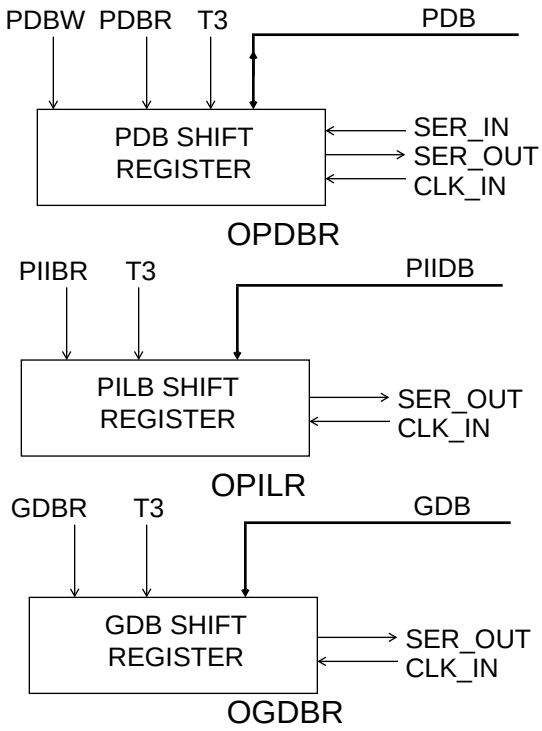


Figure 10-5 Pipeline Information Registers

10.8.2 OnCE PAB Register for Decode (OPABDR)

The 16-bit OPABDR stores the address of the instruction currently in the Instruction Latch. This is the instruction that would have been decoded if the chip would not have entered the Debug Mode. OPABDR is available for read operations only through the serial interface. This register is not affected by the operations performed during the Debug Mode.

10.8.3 OnCE PAB FIFO

The FIFO is implemented as a circular buffer containing five 16-bit registers and one 3-bit counter. All registers have the same address but any read access to the FIFO will cause an increment of the counter thus pointing to the next FIFO register. The registers are serially available for read to the command controller through their common FIFO address. The FIFO is not affected by the operations performed during the Debug Mode except for the FIFO pointer increment when reading the FIFO. Figure 10-6 illustrates a block diagram of the Program Address Bus FIFO.

Caution

To ensure FIFO coherence, a complete set of five reads of the FIFO must be performed. This is necessary due to the fact that each read increments the FIFO pointer thus causing it to point to the next location. After five reads the pointer will point to the same location as before starting the read procedure.

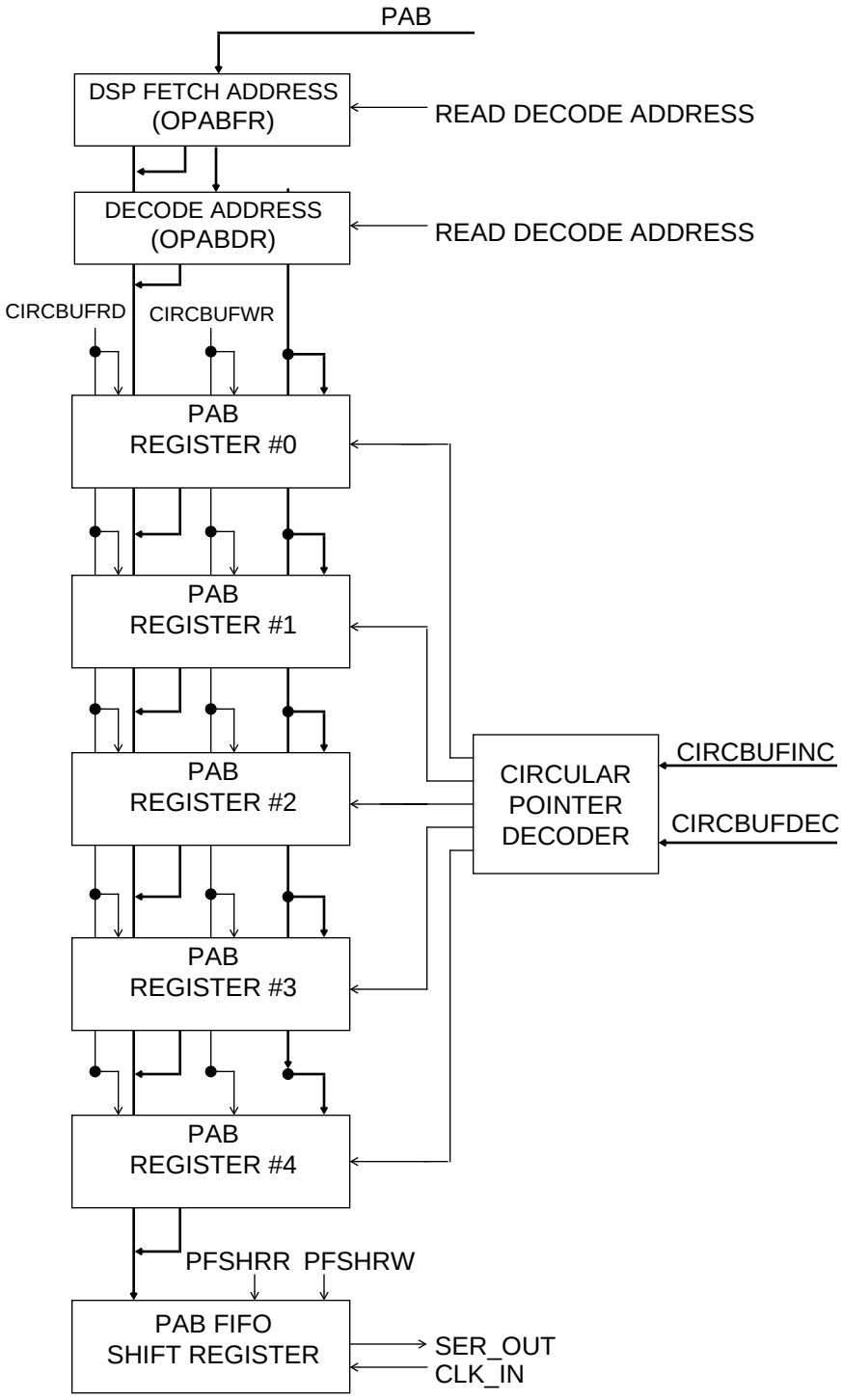


Figure 10-6 Program Address Bus FIFO

7	6	5	4	3	2	1	0
R/W	GO	EX	RS4	RS3	RS2	RS1	RS0

Figure 10-7 OnCE Command Format

10.9 SERIAL PROTOCOL DESCRIPTION

In order to permit an efficient means of communication between the command controller and the DSP chip, the following protocol has been adopted. Before starting any debugging activity, the command controller has to wait for an acknowledge from the chip which informs the command controller that it has entered the Debug Mode. Note that in case of a breakpoint, trace or software DEBUG/DEBUGcc instruction, the acknowledge itself is the one that initiates the debug session. The command controller communicates with the chip by sending 8-bit commands that may be accompanied by 16-bit data. After sending a command, the command processor starts waiting for the chip to acknowledge execution of the command. The command processor may send a new command only after the chip has acknowledged execution of the previous command.

10.9.1 OnCE Commands

There are two type of commands: read commands (when the chip will deliver required data) and write commands (when the chip will receive data and will write it in one of the on chip resources). The commands are 8 bits long and have the format shown in Figure 10-7.

10.9.1.1 OnCE Register Select (RS4-RS0) Bits 0-4

The Register Select bits define which register is source(destination) for the read(write) operation.

RS4-RS0	Register Selected
00000	Debug Status/Control (OSCR)
00001	Memory Breakpoint Counter (OMBC)
00010	Reserved
00011	Trace Counter (OTC)
00100	Memory Breakpoint Address (OMBAR)
00101	Reserved
00110	Reserved
00111	Reserved
01000	Global Data Bus (Transfer) Register (OGDBR)
01001	Program Data Bus (OPDBR) Register
01010	Program Address Bus (OPABFR) Latch for Fetch
01011	Instruction Latch (OPILR)
01100	Clear Breakpoint Counter
01101	Reserved
01110	Clear Trace Counter
01111	Reserved
10000	Reserved
10001	Program Address Bus FIFO and Increment Counter
10010	Reserved
10011	Program Address Bus (OPABDR) Latch for Decode
101xx	Reserved
11xx0	Reserved
11x0x	Reserved
110xx	Reserved
11111	No Register Selected

10.9.1.2 OnCE Exit Command (EX) Bit 5

Bit 5 in the OnCE command word is the exit command. To leave the OnCE mode and re-enter the normal operating mode, both the EX and GO bits must be asserted in the OnCE input command register. There are three exit conditions:

1. **If EX and GO are set**, the chip will leave the Debug Mode, execute the DSP instruction in the pipeline and then resume normal operation. If the register select bits are set to $1F (RS4-RS0 = 11111) then the last instruction (the instruction in the PILB) is re-executed.
2. **If EX is set without GO**, then when the OnCE has finished writing the instruction latch (PILB) register, the OnCE state machine will get another command instead of leaving the OnCE mode.
3. **If EX is set without GO**, then when the OnCE is finished writing the PDB (PILB) register, the OnCE state machine will get another command instead of leaving the OnCE mode.

There is no acknowledgment on the DSO pin when the chip leaves the OnCE mode following a GO or an EX.

EX	Action
0	Remain in Debug Mode
1	Leave Debug Mode

10.9.1.3 OnCE Go Command (GO) Bit 6

If GO is set, execute instruction. There is no acknowledgment on the DSO pin when the chip leaves the OnCE mode following a GO or an EX.

GO	Action
0	Inactive (no action taken)
1	Execute DSP instruction

10.9.1.4 OnCE Read/Write Command (R/$\overline{W}$) Bit 7

R/$\overline{W}$	Action
0	Write the data associated with the command into the register specified by RS4-RS0
1	Read the data contained in the register specified by RS4-RS0

10.10 DSP56100 TARGET SITE DEBUG SYSTEM REQUIREMENTS

A typical debug environment consists of a target system where the DSP resides in the user defined hardware. The debug serial port interfaces to the command convertor over a six wire link consisting of the four debug serial lines, a ground and reset wire. The reset wire is optional and is only used to reset the DSP and its associated circuitry.

The command controller acts as the medium between the DSP target system and a host computer. The host computer interfaces to the controller using a standard RS232 three wire cable or the Application Development System parallel bus. A jumper option on the command controller board will select which method of communications will be used. This allows a variety of different host computers to communicate with the controller circuit. The controller circuit provides several important functions. It acts as a serial debug port driver, host computer command interpreter, and DSP controller. The DSP acts as a slave when in the debug mode and provides data only upon request. The controller issues commands based on the host computer inputs from a user interface program which communicates with the user.

10.11 USING THE OnCE

The following notations are used:

Commands require eight clocks
ACK = Wait for acknowledge on DSO line

CLK = Issue 16 clocks to read out data from selected register

10.11.1 Begin Debug Activity

Debug activity begins on an instruction boundary after the $\overline{DR}$ pin is asserted, a DEBUGcc opcode is executed, a trace countdown occurs, or a breakpoint register countdown occurs. If the instruction executing when the $\overline{DR}$ pin is asserted is a REP instruction or the instruction following a REP instruction, then the debug activity will begin after the instruction following the REP instruction finishes being repeated. The first ACK indicates that the OnCE controller is ready to receive commands and data. Most of the Debug activities will have the following beginning:

ACK

1. Save pipeline information:
   a. Send command READ PDB REGISTER
   b. ACK
   c. CLK
   d. Send command READ OPILR
   e. ACK
   f. CLK
2. Read PAB FIFO and fetch/decode info (this step is optional):
   a. Send command READ PAB address for fetch
   b. ACK
   c. CLK
   d. Send command READ PAB address for decode
   e. ACK
   f. CLK
   g. Send command READ FIFO REGISTER (and increment pointer)
   h. ACK
   i. CLK
   j. Send command READ FIFO REGISTER (and increment pointer)
   k. ACK
   l. CLK

m. Send command READ FIFO REGISTER (and increment pointer)
n. ACK
o. CLK
p. Send command READ FIFO REGISTER (and increment pointer)
q. ACK
r. CLK
s. Send command READ FIFO REGISTER (and increment pointer)
t. ACK
u. CLK

10.11.2 Displaying a Specified Register

1. Send command WRITE PDB REGISTER and GO (no EX)
   (ODEC selects PDB as destination for serial data.)
2. ACK
3. Send the 16-bit opcode: “MOVE reg, x:OGDB
   (After all 16-bits have been received, the PDB register drives the PDB. ODEC generates PRNEW and releases the chip from the “halt” state and the contents of the register specified in the instruction is loaded in the GDB REGISTER. The PRCYC1 signal (an internal signal) that marks the end of the instruction brings the chip again in the “halt” state and an acknowledge is issued to the command controller)
4. ACK
5. Send command READ GDB REGISTER
   (ODEC selects GDB as the source for serial data and an acknowledge is issued to the command controller)
6. ACK
7. CLK

10.11.3 Displaying X Memory Area Starting from Address xxxx

This command uses Rn to minimize serial traffic.

1. Send command WRITE PDB REGISTER and GO (no EX).
   (ODEC selects PDB as destination for serial data.)
2. ACK

3. Send the 16-bit opcode: “MOVE R0,x:OGDB”
   (After all 16-bits have been received, the PDB register drives the PDB. ODEC generates PRNEW and releases the chip from the “halt” state and the contents of R0 are loaded in the GDB REGISTER. The PRCYC1 signal that marks the end of the instruction brings the chip again to the “halt” state and an acknowledge is issued to the command controller)
4. ACK
5. Send command READ GDB REGISTER
   (ODEC selects GDB as the source for serial data and an acknowledge is issued to the command controller)
6. ACK
7. CLK
   (The command controller generates 16 clocks that shift out the contents of the GDB register. The value of R0 is thus saved and will be restored before exiting the Debug Mode)
8. Send command WRITE PDB REGISTER (no GO, no EX).
   (ODEC selects PDB as destination for serial data.)
9. ACK
10. Send the 16-bits of opcode: “MOVE #$xxxx,R0”
    (After all 16-bits have been received, the PDB register drives the PDB. ODEC generates PRNEW so the PILR is loaded with the opcode. An acknowledge is issued to the command controller)
11. ACK
12. Send command WRITE PDB REGISTER and GO (no EX).
    (ODEC selects PDB as destination for serial data.)
13. ACK
14. Send the 16-bits of the 2nd word of: “MOVE #$xxxx,R0” (the xxxx field) where xxxx is the address to be read.
    (After all 16-bits have been received, the PDB register drives the PDB. ODEC releases the chip from the “halt” state and the instruction starts execution. The PRCYC1 signal that marks the end of the instruction brings the chip again to the “halt” state and an acknowledge is issued to the command controller)

15. ACK
16. Send command WRITE PDB REGISTER and GO (no EX).
    (ODEC selects PDB as destination for serial data.)
17. ACK
18. Send the 16-bit opcode: “MOVE X:(R0)+,x:OGDB”
    (After all 16-bits have been received, the PDB register drives the PDB. ODEC generates PRNEW and releases the chip form the “halt” state and the contents of X:(R0) are loaded in the GDB REGISTER. The PRCYC1 signal that marks the end of the instruction brings the chip again in the “halt” state and an acknowledge is issued to the command controller)
19. ACK
20. Send command READ GDB REGISTER
    (ODEC selects GDB as source for serial data and an acknowledge is issued to the command controller)
21. ACK
22. CLK
23. Send command NO SELECTION and GO (no EX).
    (ODEC releases the chip from the “halt” state and the instruction is executed once again (in a “REPEAT-like” fashion. The PRCYC1 signal that marks the end of the instruction brings the chip again to the “halt” state and an acknowledge is issued to the command controller.)
24. ACK
25. Send command READ GDB REGISTER
    (ODEC selects GDB as source for serial data and an acknowledge is issued to the command controller.)
26. ACK
27. CLK
28. Repeat from step 23 until the entire memory area is examined. At the end of the process R0 has to be restored.

10.11.4 Returning from Debug Mode to Normal Mode

There are two cases for returning from the debug mode. In **case 1**, control will be returned to the program that was running before debug was initiated and in **case 2**, the registers will be changed to jump to a different program. There is **no acknowledgment** on the DSO pin when the chip leaves the OnCE mode following a GO, EX. This is a special case of the “write a register” option.

10.11.4.1 Case 1: Returning from Debug Mode to Normal Mode

1. Send command WRITE PDB REGISTER (no GO, no EX).

   (ODEC selects the PDB register as destination for serial data. Also ODEC selects the on-chip PAB register as source for the PAB bus. After the PAB was driven an acknowledge is issued to the command controller)

2. ACK

3. Send the 16-bits of the saved PILB (instruction latch) value.

   (After all 16-bits have been received, the PDB register drives the PDB. ODEC generates PRNEW so the entire chip loads the opcode. An acknowledge is issued to the command controller)

4. ACK

5. Send command WRITE PDB REGISTER (GO, EX).
   (ODEC selects PDB as destination for serial data.)

6. ACK

7. Send the 16-bits of the saved PDB value.
   (After all 16-bits have been received, the PDB register drives the PDB. ODEC releases the chip form the “halt” state and the Debug Mode bit in OSCR is cleared. The chip continues to execute instructions until a Debug Mode condition occurs)

10.11.4.2 Case 2: Jump to a New Program (Go from Address $xxxx).

1. Send command WRITE PDB REGISTER (no GO, no EX).
   (ODEC selects PDB as destination for serial data.)

2. ACK

3. Send 16 bits of the opcode of a two word jump instruction instead of the saved PIL (instruction latch) value.

   (After all the 16-bits have been received, the PDB register drives the PDB. ODEC causes the DSP to load the opcode. An acknowledge is issued to the command controller.)

4. ACK

5. Send command WRITE PDB REGISTER (GO, EX).
   (ODEC selects PDB as destination for serial data.)

6. ACK

7. Send 16 bits of the target absolute address ($xxxx). The chip will resume fetching from the target address (you do not have to worry about the pipeline). Note that the trace counter will count this instruction so the current trace counter may need to be corrected if the trace mode enable bit in the OSCR has been set.

   (e. g., After 16 bits have been received, the PDB register drives the PDB. ODEC releases the chip from the “halt” state and the Debug Mode bit in OSCR is cleared. The chip executes first the jump instruction and will then fetch the instruction from the target address. The chip continues to execute instructions from that address until a Debug Mode condition occurs.)