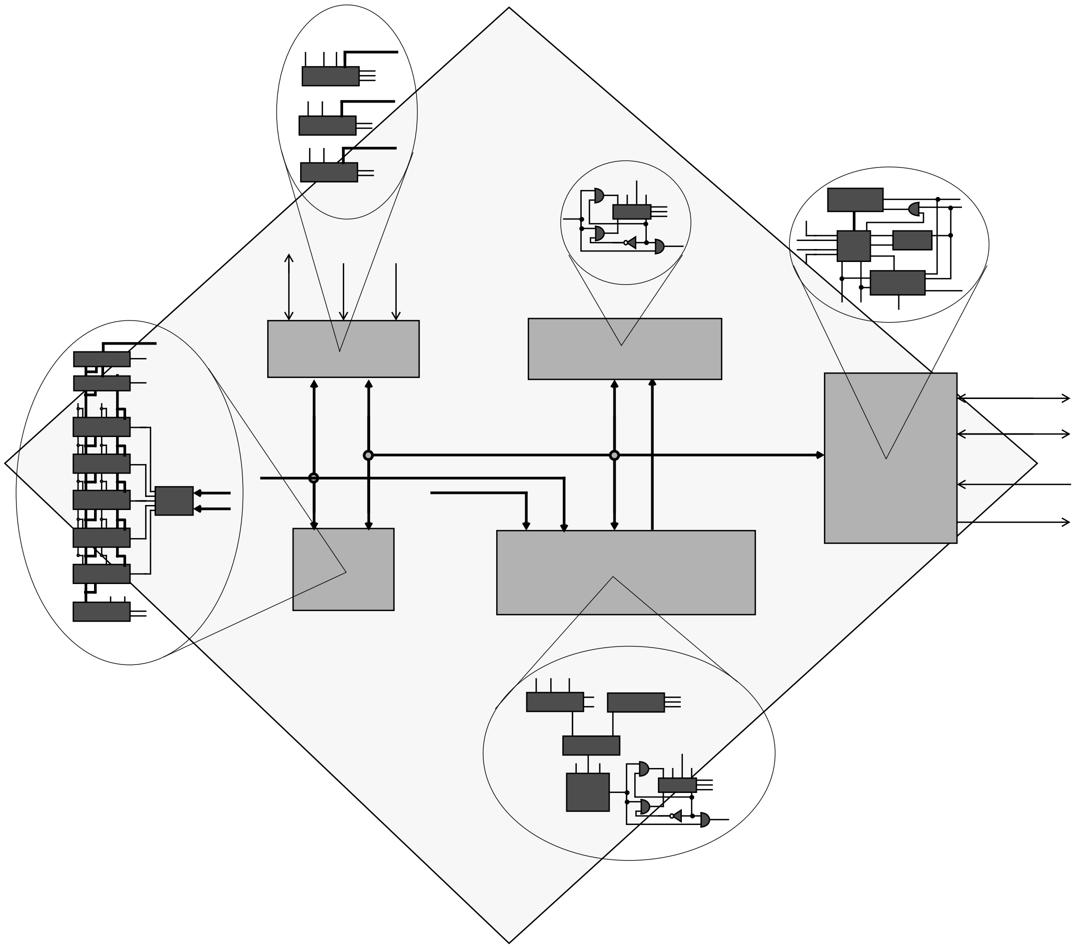
SECTION 10
ON-CHIP EMULATION (OnCE)



SECTION CONTENTS

10.1 ON-CHIP EMULATION INTRODUCTION

The DSP56K on-chip emulation (OnCE) circuitry provides a sophisticated debugging tool that allows simple, inexpensive, and speed independent access to the processor's internal registers and peripherals. OnCE tells application programmers exactly what the status is within the registers, memory locations, buses, and even the last five instructions that were executed. OnCE capabilities are accessible through a standard set of pins which are the same on all of the members of the DSP56K processor family. Figure 10-1 shows the components of the OnCE circuitry. OnCE is shown as part of the DSP56K central processing module in Figure 10-2.

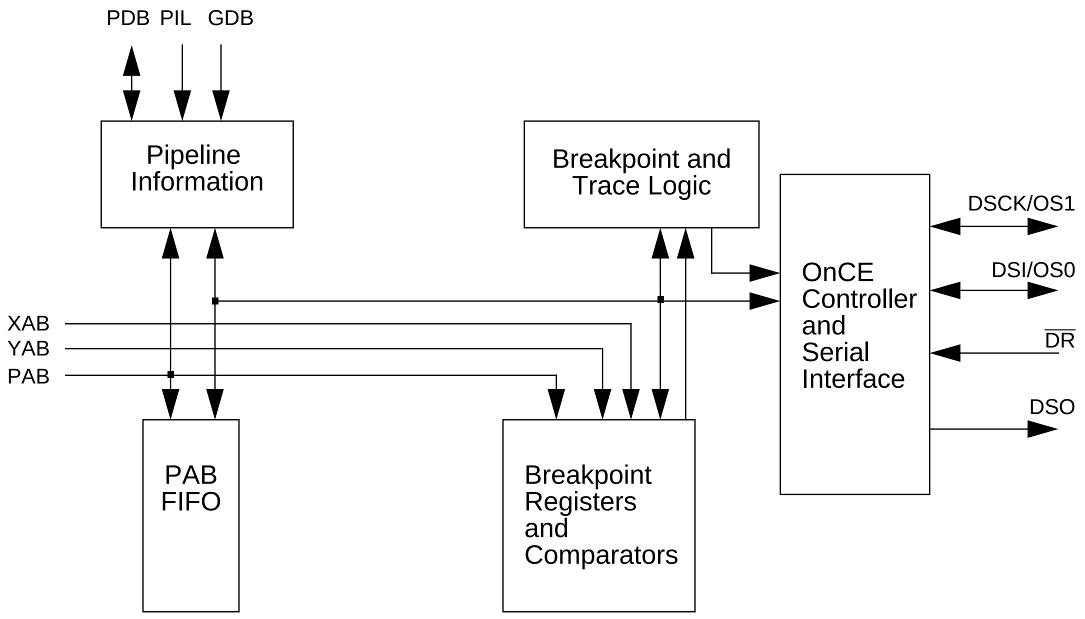

PDB
PIL
GDB
Pipeline Information
Breakpoint and Trace Logic
OnCE Controller and Serial Interface
DSCK/OS1
DSI/OS0
$\overline{DR}$
DSO
XAB
YAB
PAB
PAB FIFO
Breakpoint Registers and Comparators

Figure 10-1 OnCE Block Diagram

10.2 ON-CHIP EMULATION (OnCE) PINS

The following paragraphs describe the OnCE pins associated with the OnCE controller and serial interface component shown in Figure 10-1.

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

Serial data or commands are provided to the OnCE controller through the DSI/OS0 pin when it is an input. The data received on the DSI pin will be recognized only when the DSP56K has entered the debug mode of operation. Data is latched on the falling edge of the DSCK serial clock (described in Section 10.2.2). Data is always shifted into the OnCE serial port most significant bit (MSB) first. When the DSI/OS0 pin is an output, it works in conjunction with the OS1 pin to provide chip status information (see Table 10-1). The

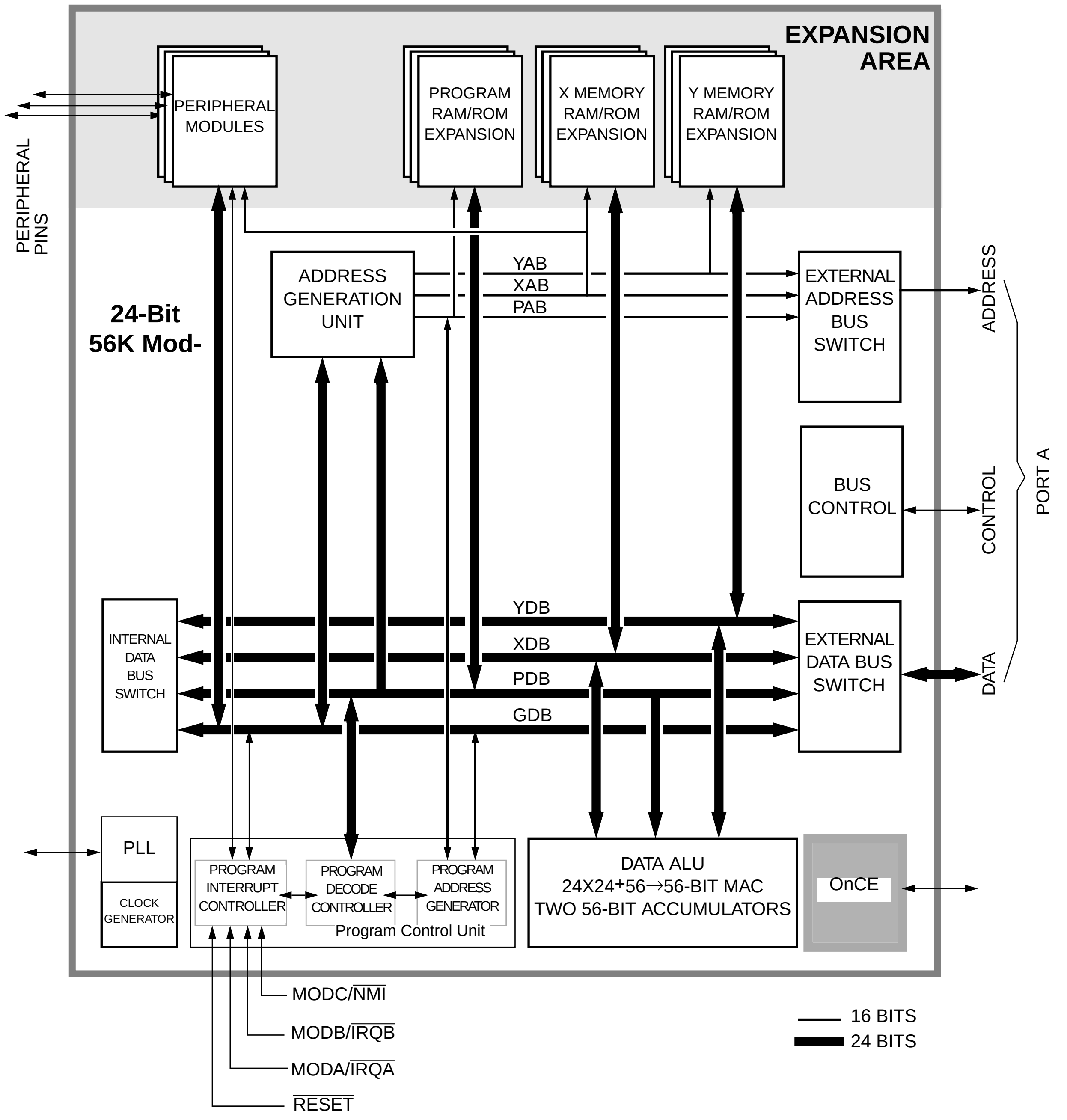


Figure 10-2 DSP56K Block Diagram

DSI/OS0 pin is an output when the processor is not in debug mode. When switching from output to input, the pin is three-stated. During hardware reset, this pin is defined as an output and it is driven low.

Note: To avoid possible glitches, an external pull-down resistor should be attached to this pin.

10.2.2 Debug Serial Clock/Chip Status 1 (DSCK/OS1)

The DSCK/OS1 pin supplies the serial clock to the OnCE when it is an input. The serial clock provides pulses required to shift data into and out of the OnCE serial port. (Data is clocked into the OnCE on the falling edge and is clocked out of the OnCE serial port on the rising edge.) The debug serial clock frequency must be no greater than 1/8 of the processor clock frequency. When an output, this pin, in conjunction with the OS0 pin, provides information about the chip status (see Table 10-1). The DSCK/OS1 pin is an output when the chip is not in debug mode. When switching from output to input, the pin is three-stated. During hardware reset, this pin is defined as an output and it is driven low.

Note: To avoid possible glitches, an external pull-down resistor should be attached to this pin.

Table 10-1 Chip Status Information

OS1	OS0	Status
0	0	Normal State
0	1	Stop or Wait State
1	0	Chip waits for bus mastership
1	1	Chip waits for end of memory wait states (due to $\overline{\text{WT}}$ assertion or BCR)

10.2.3 Debug Serial Output (DSO)

Serial data is read from the OnCE through the DSO pin, as specified by the last command received from the external command controller. Data is always shifted out the OnCE serial port most significant bit (MSB) first. Data is clocked out of the OnCE serial port on the rising edge of DSCK.

The DSO pin also provides acknowledge pulses to the external command controller. When the chip enters the debug mode, the DSO pin will be pulsed low to indicate (acknowledge) that the OnCE is waiting for commands. After receiving a read command, the DSO pin will be pulsed low to indicate that the requested data is available and the OnCE serial port is ready to receive clocks in order to deliver the data. After receiving a write command, the DSO pin will be pulsed low to indicate that the OnCE serial port is ready to receive the data to be written; after the data is written, another acknowledge pulse will be provided.

During hardware reset and when the processor is idle, the DSO pin is held high.

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

The debug request input ($\overline{\mathrm{DR}}$) allows the user to enter the debug mode of operation from the external command controller. When $\overline{\mathrm{DR}}$ is asserted, it causes the DSP56K 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 DSI line. While in debug mode, the $\overline{\mathrm{DR}}$ pin lets the user reset the OnCE controller by asserting it and deasserting it after receiving acknowledge. It may be necessary to reset the OnCE controller in cases where synchronization between the OnCE controller and external circuitry is lost. $\overline{\mathrm{DR}}$ must be deasserted after the OnCE responds with an acknowledge on the DSO pin and before sending the first OnCE command. Asserting $\overline{\mathrm{DR}}$ will cause the chip to exit the STOP or WAIT state.

10.3 OnCE CONTROLLER AND SERIAL INTERFACE

The OnCE Controller and Serial Interface contains the following blocks: OnCE command register, bit counter, OnCE decoder, and the status/control register. Figure 10-3 illustrates a block diagram of the OnCE controller and serial interface

10.3.1 OnCE Command Register (OCR)

The OCR is an 8-bit shift register that receives its serial data from the DSI pin. It holds the 8-bit commands to be used as input for the OnCE Decoder. The Command Register is shown in Figure 10-4.

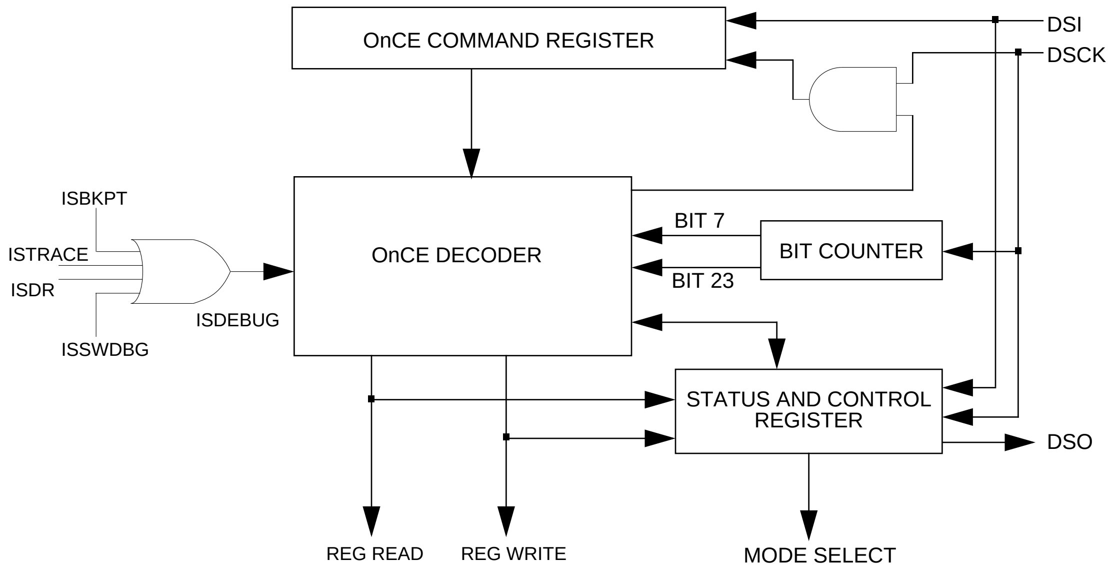


Figure 10-3 OnCE Controller and Serial Interface

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

Figure 10-4 OnCE Command Register

10.3.1.1 Register Select (RS4-RS0) Bits 0-4

The Register Select bits define which register is source (destination) for the read (write) operation. Table 10-2 indicates the OnCE register addresses.

Table 10-2 OnCE Register Addressing

RS4-RS0	Register Selected
00000	OnCE Status and Control Register (OSCR)
00001	Memory Breakpoint Counter (OMBC)
00010	Reserved
00011	Trace Counter (OTC)
00100	Reserved
00101	Reserved
00110	Memory Upper Limit Register (OMULR)
00111	Memory Lower Limit Register (OMLLR)
01000	GDB Register (OGDBR)
01001	PDB Register (OPDBR)
01010	PAB Register for Fetch (OPABFR)
01011	PIL Register (OPILR)
01100	Clear Memory Breakpoint Counter (OMBC)
01101	Reserved
01110	Clear Trace Counter (OTC)
01111	Reserved
10000	Reserved
10001	Program Address Bus FIFO and Increment Counter
10010	Reserved
10011	PAB Register for Decode (OPABDR)
101xx	Reserved
11xx0	Reserved
11x0x	Reserved
110xx	Reserved
11111	No Register Selected

10.3.1.2 Exit Command (EX) Bit 5

If the EX bit is set, the processor will leave the debug mode and resume normal operation. The Exit command is executed only if the Go command is issued, and the operation is write to OPDBR or read/write to “No Register Selected”. Otherwise the EX bit is ignored.

EX	Action
0	Remain in debug mode
1	Leave debug mode

10.3.1.3 Go Command (GO) Bit 6

If the GO bit is set, the chip will execute the instruction which resides in the PIL register. To execute the instruction, the processor leaves the debug mode, and the status is reflected in the OS0-OS1 pins. The processor will return to the debug mode immediately after executing the instruction if the EX bit is cleared. The processor goes on to normal operation if the EX bit is set. The GO command is executed only if the operation is write to OPDBR or read/write to “No Register Selected”. Otherwise the GO bit is ignored.

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

10.3.1.4 Read/Write Command (R/W) Bit 7

The R/W bit specifies the direction of data transfer. The table below describes the options defined by the R/W bit.

R/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

10.3.2 OnCE Bit Counter (OBC)

The OBC is a 5-bit counter associated with shifting in and out the data bits. The OBC is incremented by the falling edges of the DSCK. The OBC is cleared during hardware reset and whenever the DSP56K 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 24 bits were shifted in (the data associated with that command is available) or that 24 bits were shifted out (the data required by a read command was shifted out).

10.3.3 OnCE Decoder (ODEC)

The ODEC supervises the entire OnCE activity. It receives as input the 8-bit command from the OCR, two signals from OBC (one indicating that 8 bits have been received and the other that 24 bits have been received), and two signals indicating that the processor was 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 Status and Control Register is a 16-bit register used to select the events that will put the chip in debug mode and to indicate the reason for entering debug mode. The control bits are read/write while the status bits are read only. See Figure 10-5.

10.3.4.1 Memory Breakpoint Control (BC0-BC3) Bits 0-3

These control bits enable memory breakpoints. They allow memory breakpoints to occur when a memory address is within the low and high memory address registers and will select whether the breakpoint will be recognized for read, write, or fetch (program space) accesses. These bits are cleared on hardware reset. See Table 10-3 for the definition of the BC0-BC3 bits.

When BC3-BC0=0001, program memory breakpoints are enabled for any **fetch** access to the program space (true and false fetches, fetches of 2nd word, etc.). Explicit program memory accesses resulting from MOVEP and MOVEM instructions to/from program memory space are ignored.

When BC3-BC0=0010, program memory breakpoints are enabled for any **read** access to the Program space (MOVEP and MOVEM instructions from P: memory space, true and false fetches, fetches of 2nd word, etc.). Explicit program memory write accesses resulting from MOVEP and MOVEM instructions to P: memory space are ignored.

15 11	10	9	8	7	6	5	4	3	2	1	0
*	TO	MBO	SWO	*	*	*	TME	BC3	BC2	BC1	BC0

* Reserved, read as zero, should be written with zero for future compatibility.

Figure 10-5 OnCE Status and Control Register (OSCR)

When BC3-BC0=0011, program memory breakpoints are enabled for any **read or write** access to the Program space (any kind of MOVE, true and false fetches, fetches of second word, etc.).

When BC3-BC0=0100, program memory breakpoints are enabled only for **fetches of the first instruction word** of instructions that are actually executed. Aborted instructions and prefetched instructions that are discarded (such as jump targets that are not taken) are ignored by the breakpoint logic.

When BC3-BC0=0101, 0110 or 0111, program memory breakpoints are enabled only for explicit program memory access resulting from MOVEP or MOVEM instructions to/from P: memory space.

Table 10-3 Memory Breakpoint Control Table

BC3	BC2	BC1	BC0	DESCRIPTION
0	0	0	0	Breakpoint disabled
0	0	0	1	Breakpoint on any fetch (including aborted instructions)
0	0	1	0	Breakpoint on any P read (any fetch or move)
0	0	1	1	Breakpoint on any P access (any fetch, P move R/W)
0	1	0	0	Breakpoint on executed fetches only
0	1	0	1	Breakpoint on P space write
0	1	1	0	Breakpoint on P space read (no fetches)
0	1	1	1	Breakpoint on P space write or read (no fetches)
1	0	0	0	Reserved
1	0	0	1	Breakpoint on X space write
1	0	1	0	Breakpoint on X space read
1	0	1	1	Breakpoint on X space write or read
1	1	0	0	Reserved
1	1	0	1	Breakpoint on Y space write
1	1	1	0	Breakpoint on Y space read
1	1	1	1	Breakpoint on Y space write or read

10.3.4.2 Trace Mode Enable (TME) Bit 4

The TME control bit, when set, enables the Trace Mode of operation (see Section 10.5). This bit is cleared on hardware reset.

10.3.4.3 Reserved (Bits 5-7, 11-15)

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

10.3.4.4 Software Debug Occurrence (SWO) Bit 8

This read-only status bit is set when the processor enters debug mode of operation as a result of the execution of the DEBUG or DEBUGcc instruction with condition true. This bit is cleared on hardware reset or when leaving the debug mode with the GO and EX bits set.

10.3.4.5 Memory Breakpoint Occurrence (MBO) Bit 9

This read-only status bit is set when a memory breakpoint occurs. This bit is cleared on hardware reset or when leaving the debug mode with the GO and EX bits set.

10.3.4.6 Trace Occurrence (TO) Bit 10

This read-only status bit is set when the processor enters debug mode of operation, when the trace counter is zero and the trace mode has been armed. This bit is cleared on hardware reset or when leaving the debug mode with the GO and EX bits set.

10.4 OnCE MEMORY BREAKPOINT LOGIC

Memory breakpoints may be set on program memory or data memory locations. Also, the breakpoint does not have to be in a specific memory address but within an address range of where the program may be executing. This significantly increases the programmer’s ability to monitor what the program is doing in real-time.

The breakpoint logic contains a latch for the addresses, registers that store the upper and lower address limit, comparators, and a breakpoint counter. Figure 10-6 illustrates the block diagram of the OnCE Memory Breakpoint Logic.

Address comparators help to determine where a program may be getting lost or when data is being written to areas that should not be written to. They are also useful in halting a program at a specific point to examine/change registers or memory. Using address comparators to set breakpoints enables the user to set breakpoints in RAM or ROM in any operating mode. Memory accesses are monitored according to the contents of the OSCR.

The low address comparator will generate a logic true signal when the address on the bus is greater than or equal to the contents of the lower limit register. The high address comparator will generate a logic true signal when the address on the bus is less than or equal to the contents of the upper limit register. If the low address comparator and high address comparator both issue a logic true signal, the address is within the address range and the breakpoint counter is decremented if the contents are greater than zero. If zero, the counter is not decremented and the breakpoint exception occurs (ISBKPT asserted).

10.4.1 Memory Address Latch (OMAL)

The Memory Address Latch is a 16-bit register that latches the PAB, XAB or YAB on every instruction cycle according to the BC3-BC0 bits in OSCR.

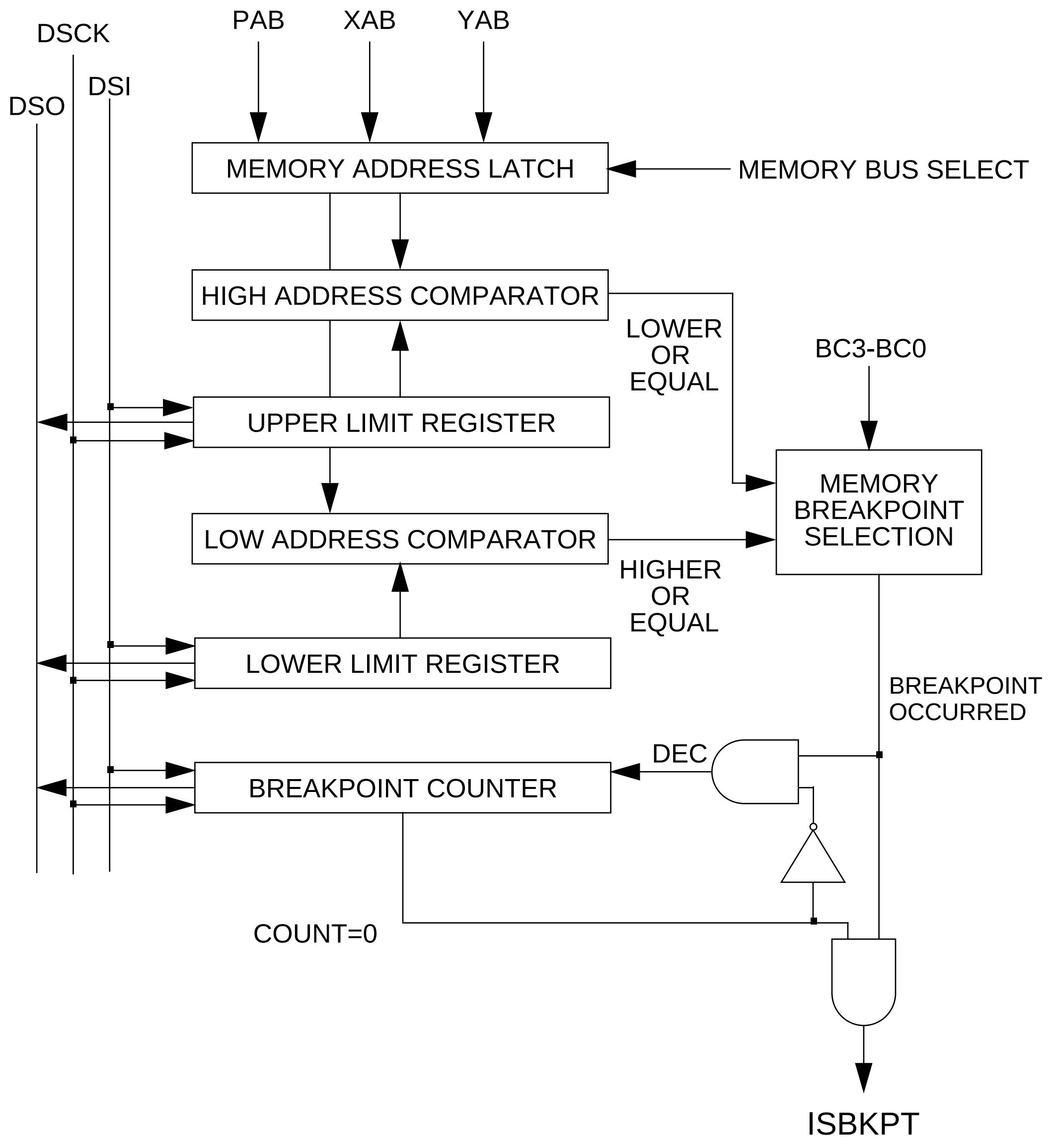


Figure 10-6 OnCE Memory Breakpoint Logic

10.4.2 Memory Upper Limit Register (OMULR)

The 16-bit Memory Upper Limit Register stores the memory breakpoint upper limit. The OMULR can be read or written through the OnCE serial interface. Before enabling breakpoints, OMULR must be loaded by the external command controller.

10.4.3 Memory Lower Limit Register (OMLLR)

The 16-bit Memory Lower Limit Register stores the memory breakpoint lower limit. The OMLLR can be read or written through the OnCE serial interface. Before enabling breakpoints, OMLLR must be loaded by the external command controller.

10.4.4 Memory High Address Comparator (OMHC)

The OMHC compares the current memory address (stored in OMAL) with the OMULR contents. If OMULR is higher than or equal to OMAL then the comparator delivers a signal indicating that the address is lower than or equal to the upper limit.

10.4.5 Memory Low Address Comparator (OMLC)

The OMLC compares the current memory address (stored in OMAL) with the OMLLR contents. If OMLLR is lower than or equal to OMAL then the comparator delivers a signal indicating that the address is higher than or equal to the lower limit.

10.4.6 Memory Breakpoint Counter (OMBC)

The 24-bit OMBC is loaded with a value equal to the number of times, minus one, that a memory access event should occur before a memory breakpoint is declared. The memory access event is specified by the BC3-BC0 bits in the OSCR register and by the memory upper and lower limit registers. On each occurrence of the memory access event, the breakpoint counter is decremented. When the counter has reached the value of zero and a new occurrence takes place, the chip will enter the debug mode. The OMBC can be read, written, or cleared through the OnCE serial interface.

Anytime the upper or lower limit registers are changed, or a different breakpoint event is selected in the OSCR, the breakpoint counter must be written afterward. This assures that the OnCE breakpoint logic is reset and that no previous events will affect the new breakpoint event selected.

The breakpoint counter is cleared by hardware reset.

10.5 OnCE TRACE LOGIC

The OnCE trace logic allows the user to execute instructions in single or multiple steps before the chip returns to the debug mode and awaits OnCE commands from the debug serial port. (The OnCE trace logic is independent of the trace facility of the DSP56000/56001, which is operated through the trace interrupt discussed in Section 7.3.3.3, and started by setting the trace bit in the processor’s status register discussed in Section 5.4.2.12). The OnCE trace logic block diagram is shown in Figure 10-7.

The trace counter allows more than one instruction to be executed in real time before the chip returns to the debug mode of operation. 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 count the number of instructions executed in a code segment.

To initiate 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 TME bit is set in the OSCR, and the processor 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 and the execution of each repeated instruction) will decrement the trace counter.

Upon decrementing the trace counter to zero, the processor will re-enter the debug mode, the trace occurrence bit TO in the OSCR will be set, and the DSO pin will be toggled to indicate that the processor has entered debug mode and is requesting service (ISTRACE asserted).

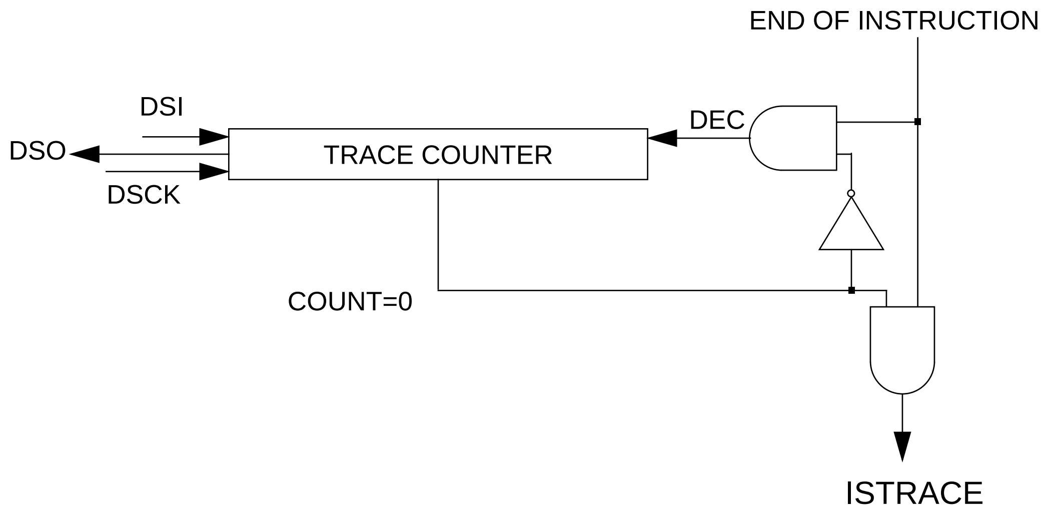


Figure 10-7 OnCE Trace Logic Block Diagram

10.5.1 Trace Counter (OTC)

The OTC is a 24-bit counter that can be read, written, or cleared through the OnCE serial interface. If N instructions are to be executed before entering the debug mode, the Trace Counter should be loaded with N-1. The Trace Counter is cleared by hardware reset.

10.6 METHODS OF ENTERING THE DEBUG MODE

The chip acknowledges having entered the debug mode by pulsing low the DSO line, informing the external command controller that the chip has entered the debug mode and is waiting for commands.The following paragraphs discuss conditions that bring the processor into the debug mode.

10.6.1 External Debug Request During $\overline{\text{RESET}}$

Holding the $\overline{\text{DR}}$ line asserted during the assertion of $\overline{\text{RESET}}$ causes the chip to enter the debug mode. After receiving the acknowledge, the external command controller must deassert the $\overline{\text{DR}}$ line before sending the first command. Note that in this case the chip does not execute any instruction before entering the debug mode.

10.6.2 External Debug Request During Normal Activity

Holding the $\overline{\text{DR}}$ line asserted during normal chip activity causes the chip to finish the execution of the current instruction and then enter the debug mode. After receiving the acknowledge, the external command controller must deassert the $\overline{\text{DR}}$ line before sending the first command. Note that in this case the chip completes the 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 by the interrupt processing, or those that will be aborted by the interrupt processing.

10.6.3 External Debug Request During STOP

Asserting $\overline{\text{DR}}$ when the chip is in the stop state (i. e., has executed a STOP instruction) and keeping it asserted until an acknowledge pulse in DSO is produced causes the chip to exit the stop state and enter the debug mode. After receiving the acknowledge, the external command controller must deassert $\overline{\text{DR}}$ before sending the first command. 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 Debug Request During WAIT

Asserting $\overline{\text{DR}}$ when the chip is in the wait state (i. e., has executed a WAIT instruction) and keeping it asserted until an acknowledge pulse in DSO is produced causes the chip to exit the wait state and enter the debug mode. After receiving the acknowledge, the external command controller must deassert $\overline{\text{DR}}$ before sending the first command. Note that in this case, the chip completes the 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 instruction when the specified condition is true, the chip enters the debug mode after the instruction following the DEBUG instruction has entered the instruction latch.

10.6.6 Enabling Trace Mode

When the trace mode mechanism is enabled and the trace counter is greater than zero, the trace counter is decremented after each instruction execution. The completed execution of an instruction when the trace counter is zero will cause the chip to enter the debug mode.

Note: Only instructions actually executed cause the trace counter to decrement, i.e. an aborted instruction will not decrement the trace counter and will not cause the chip to enter the debug mode.

10.6.7 Enabling Memory Breakpoints

When the memory breakpoint mechanism is enabled with a breakpoint counter value of zero, the chip enters the debug mode after completing the execution of the instruction that caused the memory breakpoint to occur. In case of breakpoints on executed program memory fetches, the breakpoint will be acknowledged immediately after the execution of the fetched instruction. In case of breakpoints on data memory addresses (accesses to X, Y or P memory spaces by MOVE instructions), the breakpoint will be acknowledged after the completion of the instruction following the instruction that accessed the specified address.

10.7 PIPELINE INFORMATION AND GLOBAL DATA BUS REGISTER

A number of on-chip registers store the chip pipeline status to restore the pipeline and resume normal chip activity upon return from the debug mode. Figure 10-8 shows the block diagram of the pipeline information registers with the exception of the program address bus (PAB) registers, which are shown in Figure 10-9.

10.7.1 Program Data Bus Register (OPDBR)

The OPDBR is a 24-bit latch that stores the value of the program data bus which was generated by the last program memory access before the chip entered the debug mode. OPDBR can be read or written through the OnCE serial interface. It is affected by the operations performed during the debug mode and must be restored by the external command controller when the chip returns to normal mode.

10.7.2 Pipeline Instruction Latch Register (OPILR)

The OPILR is a 24-bit latch that stores the value of the instruction latch before the debug mode is entered. OPILR can only be read through the OnCE serial interface. This register is affected by the operations performed during the debug mode and must be restored by the external command controller when returning to normal mode. Since there is no direct write access to this register, this task is accomplished in the first write to OPDBR after entering the debug mode or after executing the GO command; the data from OPDBR is transferred to OPILR only in these cases.

10.7.3 Global Data Bus Register (OGDBR)

The OGDBR is a 24-bit latch that can only be read through the OnCE serial interface. OGDBR is not actually required from a pipeline status restore point of view but is required as a means of passing information between the chip and the external command controller. OGDBR is mapped on the X internal I/O space at address $FFFC. Whenever the external command controller needs the contents of a register or memory location, it will force the chip to execute an instruction that brings that information to OGDBR. Then, the contents of OGDBR will be delivered serially to the external command controller by the command “READ GDB REGISTER”.

10.8 PROGRAM ADDRESS BUS HISTORY BUFFER

There are two read-only PAB registers which give pipeline information when the debug mode is entered. The OPABFR register tells which opcode address is in the fetch stage of the pipeline and OPABDR tells which opcode is in the decode stage. To ease debugging activity and keep track of program flow, a First-In-First-Out (FIFO) buffer stores the

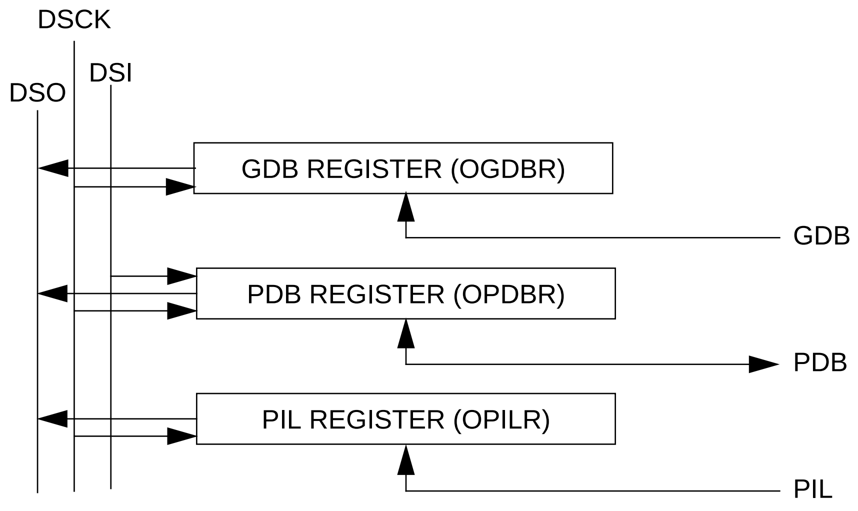


Figure 10-8 OnCE Pipeline Information and GDB Registers

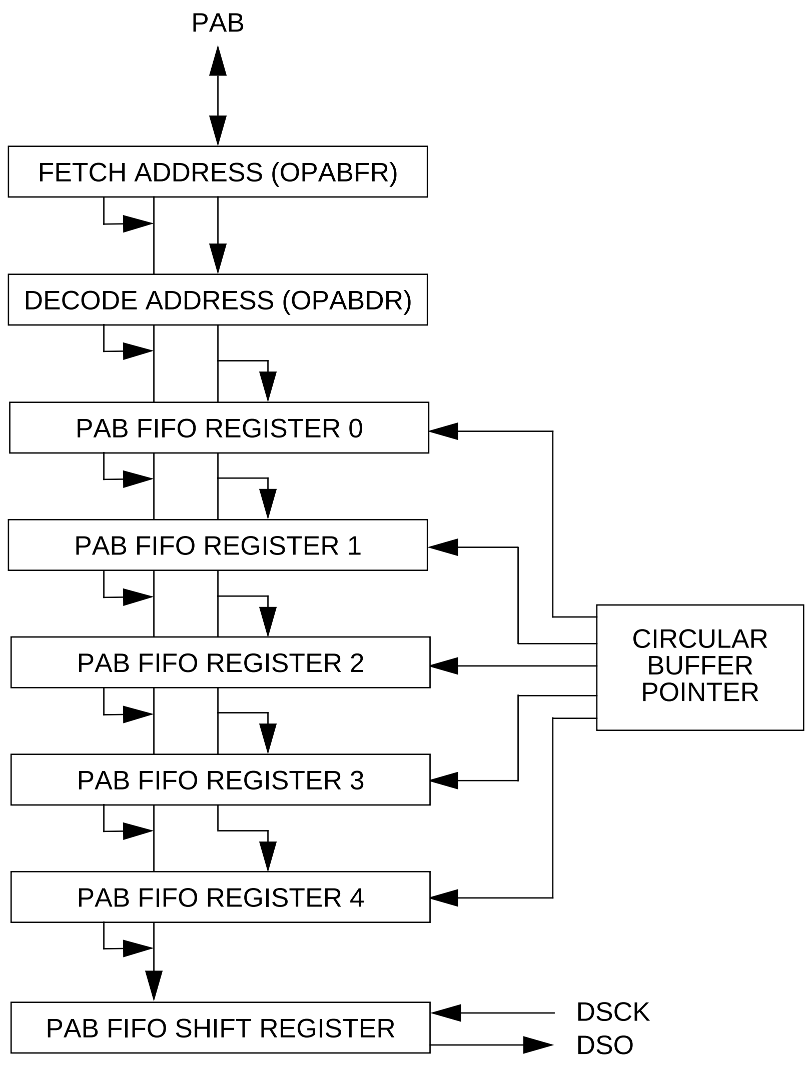


Figure 10-9 OnCE PAB FIFO

addresses of the last five instructions that were executed.

10.8.1 PAB Register for Fetch (OPABFR)

The OPABFR is a 16-bit register that stores the address of the last instruction that was fetched before the debug mode was entered. The OPABFR can only be read through the OnCE serial interface. This register is not affected by the operations performed during the debug mode.

10.8.2 PAB Register for Decode (OPABDR)

The OPABDR is a 16-bit register that 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 can only be read through the serial interface.

This register is not affected by the operations performed during the debug mode.

10.8.3 PAB FIFO

The PAB FIFO stores the addresses of the last five instructions that were executed. The FIFO is implemented as a circular buffer containing five 16-bit registers and one 3-bit counter. All the registers have the same address but any read access to the FIFO address will cause the counter to increment, making it point to the next FIFO register. The registers are serially available to the external command controller through their common FIFO address. Figure 10-9 shows the block diagram of the PAB FIFO. The FIFO is not affected by the operations performed during the debug mode except for the FIFO pointer increment when reading the FIFO. When entering the debug mode, the FIFO counter will be pointing to the FIFO register containing the address of the oldest of the five executed instructions. The first FIFO read will obtain the oldest address and the following FIFO reads will get the other addresses from the oldest to the newest (the order of execution).

To ensure FIFO coherence, a complete set of five reads of the FIFO must be performed because each read increments the FIFO pointer, thus making it point to the next location. After five reads the pointer will point to the same location it pointed to before starting the read procedure.

10.9 SERIAL PROTOCOL DESCRIPTION

The following protocol permits an efficient means of communication between the OnCE's external command controller and the DSP56K chip. Before starting any debugging activity, the external command controller must wait for an acknowledge on the DSO line, indicating that the chip has entered the debug mode. The external command controller communicates with the chip by sending 8-bit commands that may be accompanied by 24 bits of data. Both commands and data are sent or received most significant bit first. After sending a command, the external command controller must wait for the processor to acknowledge execution of the command before it may send a new command.

When accessing OnCE 16-bit registers, the register contents appear in the 16 most significant bits in the 24-bit data field, and the 8 least significant bits are zeroed.

10.9.1 OnCE Commands

The OnCE commands may be classified as follows:

- read commands (when the chip will deliver the required data).
- write commands (when the chip will receive data and write the data in one of the OnCE registers).
- commands that do not have data transfers associated with them.

The commands are 8 bits long and have the format shown in Figure 10-4.

10.10 DSP56K TARGET SITE DEBUG SYSTEM REQUIREMENTS

A typical DSP56K debug environment consists of a target system where the DSP56K resides in the user defined hardware. The debug serial port interfaces to the external command controller over a 6-wire link which includes the 4 OnCE wires, a ground, and a reset wire. The reset wire is optional and is only used to reset the DSP56K and its associated circuitry.

The external command controller acts as the medium between the DSP56K target system and a host computer. The external command controller circuit acts as a DSP56K serial debug port driver and host computer command interpreter. 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:

ACK = Wait for acknowledge on the DSO pin

CLK = Issue 24 clocks to read out data from the selected register

10.11.1 Begin Debug Activity

Most of the debug activities have the following beginning:

1. ACK
2. Save pipeline information:
    a. Send command READ PDB REGISTER (10001001)
    b. ACK
    c. CLK
    d. Send command READ PIL REGISTER (10001011)
    e. ACK
    f. CLK
3. Read PAB FIFO and fetch/decode info (this step is optional):
    a. Send command READ PAB address for fetch (10001010)
    b. ACK
    c. CLK
    d. Send command READ PAB address for decode (10010011)
    e. ACK

f. CLK

g. Send command READ FIFO REGISTER and increment counter (10010001)

h. ACK

i. CLK

j. Send command READ FIFO REGISTER and increment counter (10010001)

k. ACK

l. CLK

m. Send command READ FIFO REGISTER and increment counter (10010001)

n. ACK

o. CLK

p. Send command READ FIFO REGISTER and increment counter (10010001)

q. ACK

r. CLK

s. Send command READ FIFO REGISTER and increment counter (10010001)

t. ACK

u. CLK

10.11.2 Displaying A Specified Register

1. Send command WRITE PDB REGISTER, GO, no EX (01001001). The OnCE controller selects PDB as destination for serial data.
2. ACK
3. Send the 24-bit DSP56K opcode: “MOVE reg,x:OGDB”
   After 24 bits have been received, the PDB register drives the PDB. The OnCE controller releases the chip from the debug mode, the chip executes the MOVE instruction, and the contents of the register specified in the instruction are loaded in the GDB REGISTER. The signal that marks the end of the instruction returns the chip to the debug mode.
4. ACK
5. Send command READ GDB REGISTER (10001000)

The OnCE controller selects GDB as source for serial data.

6. ACK
7. CLK

10.11.3 Displaying X Memory Area Starting From Address XXXX

This command uses R0 to minimize serial traffic.

1. Send command WRITE PDB REGISTER, GO, no EX (01001001).
   The OnCE controller selects PDB as destination for serial data.
2. ACK
3. Send the 24-bit DSP56K opcode: “MOVE R0,x:OGDB”
   After 24 bits have been received the PDB register drives the PDB. The OnCE controller releases the chip from the debug mode and the contents of R0 are loaded in the GDB REGISTER. The signal that marks the end of the instruction returns the chip to the debug mode.
4. ACK
5. Send command READ GDB REGISTER (10001001)
   The OnCE controller selects GDB as source for serial data.
6. ACK
7. CLK
   The external command controller generates 24 clocks that shift out the contents of the GDB register. The value of R0 is thus saved and should be restored before exiting the debug mode.
8. Send command WRITE PDB REGISTER, no GO, no EX (00001001)
   OnCE controller selects PDB as destination for serial data.
9. ACK
10. Send the 24-bit DSP56K opcode: “MOVE #$xxxx,R0”
    After 24 bits have been received, the PDB register drives the PDB. The OnCE controller causes the processor to load the opcode.
11. ACK
12. Send command WRITE PDB REGISTER, GO, no EX (01001001)
    The OnCE controller selects PDB as destination for serial data.
13. ACK
14. Send the 24-bit 2nd word of: “MOVE #$xxxx,R0” (the xxxx field).
    After 24 bits have been received, the PDB register drives the PDB. The OnCE con-

troller releases the chip from the debug mode and the instruction starts execution. The signal that marks the end of the instruction returns the chip to the debug mode.

15. ACK
16. Send command WRITE PDB REGISTER, GO, no EX (01001001)
    The OnCE controller selects PDB as destination for serial data.
17. ACK
18. Send the 24-bit DSP56K opcode: “MOVE X:(R0)+,x:OGDB”
    After 24 bits have been received, the PDB register drives the PDB. The OnCE controller releases the chip from the debug mode and the contents of X:(R0) are loaded in the GDB REGISTER. The signal that marks the end of the instruction returns the chip to the debug mode.
19. ACK
20. Send command READ GDB REGISTER (10001000)
    The OnCE controller selects GDB as source for serial data.
21. ACK
22. CLK
23. Send command NO REGISTER SELECTED, GO, no EX (01011111)
    The OnCE controller releases the chip from the debug mode and the instruction is executed again in a “REPEAT-like” fashion. The signal that marks the end of the instruction returns the chip to the debug mode.
24. ACK
25. Send command READ GDB REGISTER (10001000)
    The OnCE controller selects GDB as source for serial data.
26. ACK
27. CLK
28. Repeat from step 23 until the entire memory area is examined.
29. After finishing reading the memory, R0 should to be restored as follows.
30. Send command WRITE PDB REGISTER, no GO, no EX (00001001)
    OnCE controller selects PDB as destination for serial data.
31. ACK
32. Send the 24-bit DSP56K opcode: “MOVE #saved_r0,R0”
    After 24 bits have been received, the PDB register drives the PDB. The OnCE con-

troller causes the processor to load the opcode.

33. ACK
34. Send command WRITE PDB REGISTER, GO, no EX (01001001)
    The OnCE controller selects PDB as destination for serial data.
35. ACK
36. Send the 24-bit second word of: “MOVE #saved_r0,R0” (the saved_r0 field).
    After 24 bits have been received, the PDB register drives the PDB. The OnCE controller releases the chip from the debug mode and the instruction starts execution. The signal that marks the end of the instruction returns the chip to the debug mode.
37. ACK

10.11.4 Executing a Single-Word DSP56K Instruction While in Debug Mode

1. Send command WRITE PDB REGISTER, GO, no EX (01001001).
   The OnCE controller selects PDB as destination for serial data.
2. ACK
3. Send the single-word 24-bit DSP56K opcode to be executed.
   After 24 bits have been received, the PDB register drives the PDB. The OnCE controller releases the chip from the debug mode and the chip executes the instruction. The signal that marks the end of the instruction returns the chip to the debug mode. Some DSP56K instructions should not be executed in this state: DO, REP, ILLEGAL or any opcode that is considered illegal, and DEBUG.
4. ACK

10.11.5 Executing a Two-Word DSP56K Instruction While in Debug Mode

1. Send command WRITE PDB REGISTER, no GO, no EX (00001001).
   The OnCE controller selects PDB as destination for serial data.
2. ACK
3. Send the first instruction word (24-bit DSP56K opcode)
   After 24 bits have been received, the PDB register drives the PDB. The OnCE controller causes the processor to load the opcode.
   Some DSP56K instructions should not be executed in this state: DO, REP, ILLEGAL or any opcode that is considered illegal, and DEBUG.
4. ACK
5. Send command WRITE PDB REGISTER, GO, no EX (01001001)
   The OnCE controller selects PDB as destination for serial data.
6. ACK
7. Send the second 24-bit instruction word.
   After 24 bits have been received, the PDB register drives the PDB. The OnCE controller releases the chip from the debug mode and the instruction starts execution. The signal that marks the end of the instruction returns the chip to the debug mode.
8. ACK

10.11.6 Returning from Debug Mode to Normal Mode

There are two cases for returning from the debug mode in a single processor:

- Control is returned to the program that was running before debug was initiated.
- Jump to a different program location is executed.

10.11.6.1 Case 1: Return To The Previous Program (Return To Normal Mode)

1. Send command WRITE PDB REGISTER, no GO, no EX (00001001)
   The OnCE controller selects the PDB as the destination for serial data. Also, the OnCE controller selects the on-chip PAB register as the source for the PAB bus.
2. ACK
3. Send the 24 bits of the saved PIL (instruction latch) value.
   After the 24 bits have been received, the PDB register drives the PDB. The OnCE controller causes the PIL to latch the PDB value. In this way, the PIL is restored to the same state as before entering the debug mode.
4. ACK
5. Send command WRITE PDB REGISTER, GO, EX (01101001)
   The OnCE controller selects PDB as destination for the serial data to follow.
6. ACK
7. Send the 24 bits of the saved PDB value.
   After the 24 bits have been received, the PDB register drives the PDB. In this way, the PDB is restored to the same state as before entering the debug mode. The EX bit causes the OnCE controller to release the chip from the debug mode and the status bits in OSCR are cleared. The GO bit causes the chip to start executing instructions.

10.11.6.2 Case 2: Jump To A New Program (Go From Address $xxxx)

1. Send command WRITE PDB REGISTER, no GO, no EX (00001001)
   The OnCE controller selects PDB as destination for serial data. Also, the OnCE controller selects the on-chip PAB register as the source for the PAB bus.
2. ACK
3. Send 24 bits of the opcode of a two-word jump instruction instead of the saved PIL value. After the 24 bits have been received, the PDB register drives the PDB. The OnCE controller causes the PIL to latch the PDB value. In this way, the instruction latch will contain the opcode of the jump instruction which will cause the change in the program flow.
4. ACK
5. Send command WRITE PDB REGISTER, GO, EX (01101001)
   The OnCE controller selects PDB as destination for serial data.
6. ACK
7. Send 24 bits of the jump target absolute address ($xxxxxx).
   After 24 bits have been received, the PDB register drives the PDB. In this way, the PDB contains the second word of the jump as required for the jump instruction ex-

ecution. The EX bit causes the OnCE controller to release the chip from the debug mode and the status bits in OSCR are cleared. The GO bit causes the chip to start executing the jump instruction which will then cause the chip to continue instruction execution from the target address. Note that the trace counter will count the jump instruction so the current trace counter may need to be corrected if the trace mode is enabled.

10.11.7 Debugging Multiprocessor Systems With a Single External Command Controller

In multiprocessor systems, each processor may be individually debugged as described above. When simultaneous exit of the debug state is desired for more than one processor, each processor must first be loaded with the required PIL and PDB values where processing should proceed. This is accomplished by the following sequence as applied to each processor:

1. Send command WRITE PDB REGISTER, no GO, no EX (00001001)
   The OnCE controller selects PDB as destination for serial data. Also, the OnCE controller selects the on-chip PAB register as the source for the PAB bus.
2. ACK
3. Send 24 bits of either the opcode of a 2-word jump instruction or the saved PIL value. After the 24 bits have been received, the PDB register drives the PDB. The OnCE controller causes the PIL to latch the PDB value.
4. ACK
5. Send command WRITE PDB REGISTER, no GO, no EX (00001001)
   The OnCE controller selects PDB as destination for serial data.
6. ACK
7. Send 24 bits of either the jump target absolute address ($xxxxxx) or the saved PDB value. After 24 bits have been received, the PDB register drives the PDB.
8. ACK

At this point, all processors should have the required PIL and PDB values while still in debug mode. To return all processors to the normal execution state simultaneously, the following command should be issued to all processors in parallel:

9. Send command NO REGISTER SELECTED, GO, EX (01111111)
   The OnCE controller releases the chips from the debug mode and instruction execution is resumed.