CWDSP1670 Lead Vehicle Technical Summary

LSI LOGIC®

Contents

Figures

Tables

1 Introduction

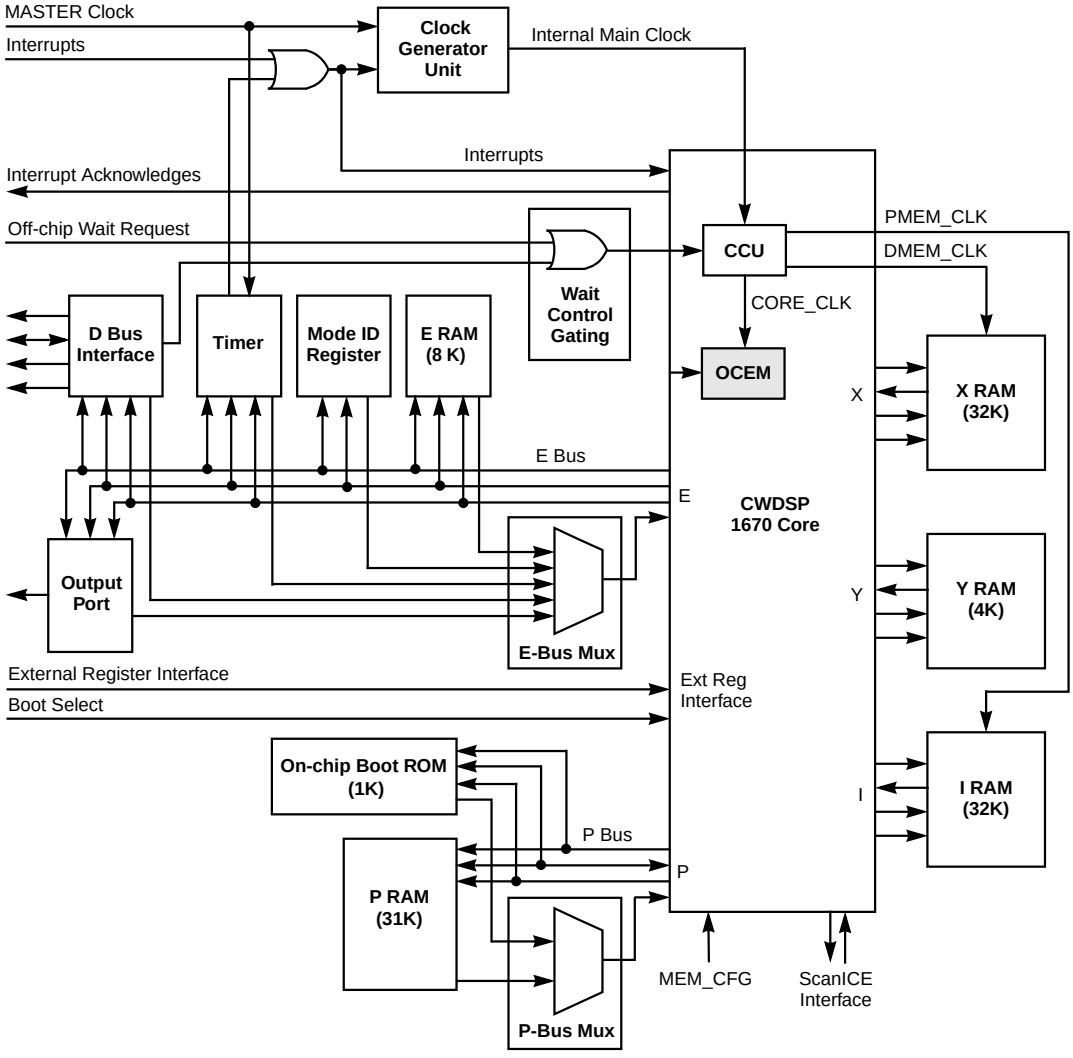
The CWDSP1670 Lead Vehicle is an OakDSPCore®-based, Application-Specific Integrated Circuit (ASIC) that is used as a reference device for system and software development. The CWDSP1670 Lead Vehicle is more suitable to ASIC development than previous generations and includes basic peripheral functions that allow you, with additional hardware, to develop a prototype design. Prototype software also can be debugged and benchmarked on the system.

2 Functional Overview

Figure 1 shows a functional block diagram of the lead vehicle. Its main functional blocks are:

- CWDSP1670 DSP Core
- 32 Kwords of on-chip X RAM
- 4 Kwords of on-chip Y RAM
- 32 Kwords of on-chip instruction RAM (I RAM)
- 1 Kword of on-chip boot ROM
- 31 Kwords of on-chip program RAM (P RAM)
- 8 Kwords of on-chip E RAM
- Clock Generator Unit
- Mode ID Register
- Timer
- D Bus Interface
- Output Port

Figure 1 CWDSP1670 Lead Vehicle Block Diagram



The CWDSP1670 core contains an integral ScanICE Control Unit and On-Chip Emulation Module (OCEM). Refer to the *CWDSP1670 DSP Core Technical Manual* for details on these core components. The core has three data bus interfaces (X, Y, and E) and two instruction bus interfaces (I and P). It can address up to 64 Kwords of data space and 64 Kwords of instruction space. The external memory configuration is

reported to the core on its MEM_CFG inputs which are tied HIGH or LOW appropriately in the lead vehicle.

The Clock Generator Unit derives the internal main clock from the MASTER clock input and provides a register-controlled Stop mode. The core E Bus space is divided among E RAM, the Mode ID register, the Timer, the external D Bus, and the Output Port. The Mode ID register lets you identify normal or Data Memory Map Compatibility (DMMC) memory mapping, two modes of allocating Y RAM and D Bus space. The DMMC mode provides memory mapping compatibility with previous CWDSP16X0 lead vehicles. The DMMC_MODE input to the lead vehicle selects the mapping mode (see Section 3.10, “Miscellaneous”).

The programmable Timer can be used to generate a single or repeated interrupts on any one of the core’s interrupt inputs (INT0–INT3 or NMI). The Timer can be halted and restarted or reset by the host on the D Bus. The D Bus Interface connects an external, 16-bit address and 16-bit, bidirectional data bus to the core in the E Bus address space. D Bus wait states can be programmed into the interface. The Output Port provides eight user-defined output signals which can be used for bank switching when booting from D Bus memories.

The External Register Interface allows CWDSP1670 instructions to access four, user-defined, 16-bit, off-chip registers. The interface includes separate 16-bit input and output data buses, multiplexer controls for selecting one register during a read cycle, and read/write controls for the four registers.

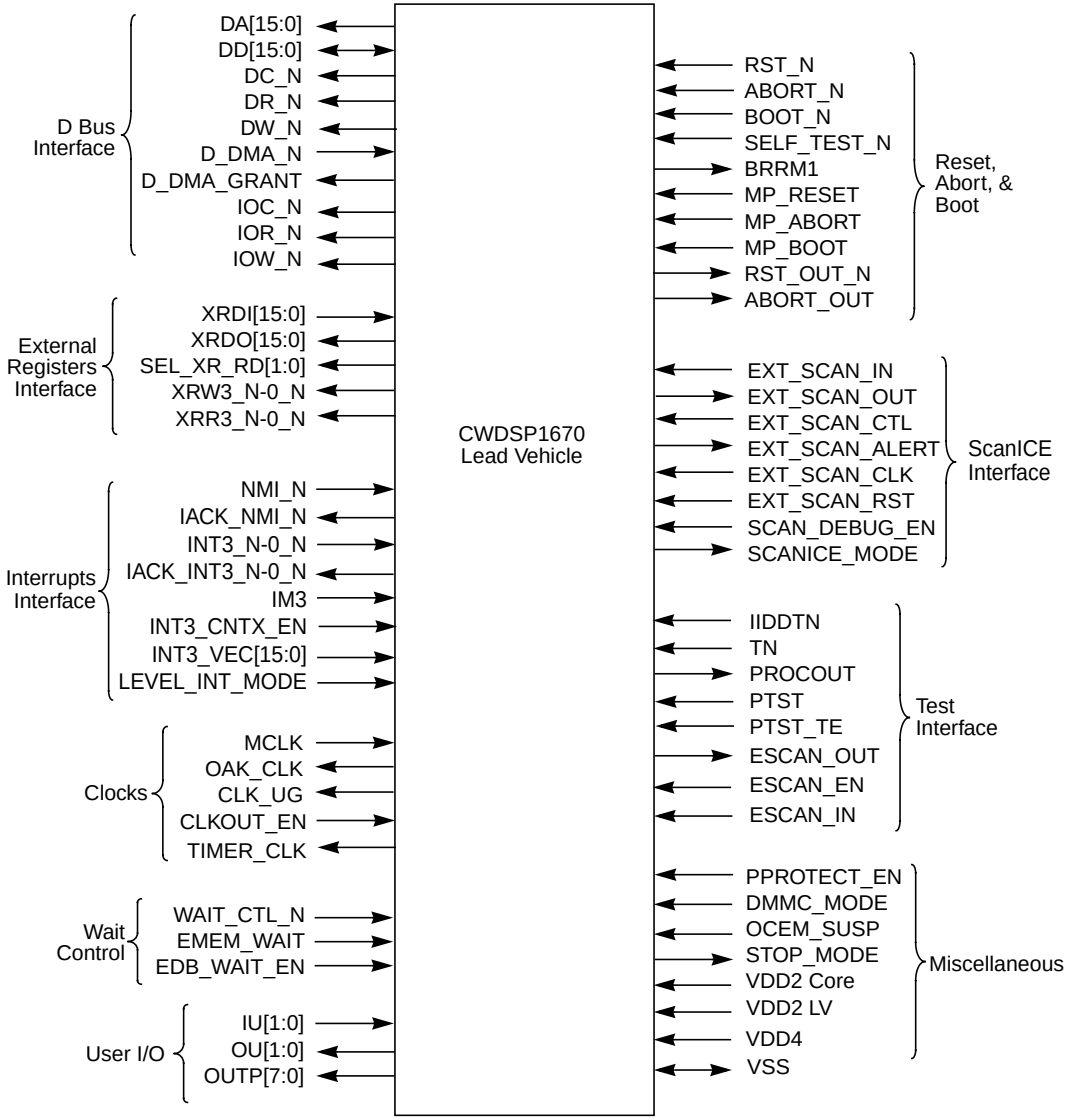
The Boot Select input lets an external host control the source of boot code.

3 Signal Descriptions

This section provides detailed descriptions of CWDSP1670 Lead Vehicle signals. Figure 2 shows all of the lead vehicle’s I/O signals organized by interface or function.

Note: Signals that are LOW when active have an “_N” suffix. Signals that are HIGH when active do not have the suffix.

Figure 2 I/O Signals



3.1 D Bus Interface

DA[15:0] **D Bus Address** **Output**

Data and I/O space address bus. The data and I/O space allocation is shown in the following table.

Normal Mode			DMMC Mode	
Space	**Range**	**Size (Words)**	**Range**	**Size (Words)**
Data	0x8000–0xBFFF	16 K	0x8000–0xBFFF	16 K
	0xE000–0xE7FF	2 K	0xE000–0xEFFF	4 K
I/O	0xE800–0xEBFF	1 K	0xF000–0xF3FF	1 K

DD[15:0] **D Bus Data** **Bidirectional**

Data and I/O space data bus.

DC_N **D Bus Data Space Chip Select** **Output**

Data space device select signal. This signal is asserted when the D Bus address is in the data space range.

DR_N **D Bus Read Enable** **Output**

Data space read enable signal. This signal is asserted when the D Bus address is in the data space range and the lead vehicle requests a read from data space.

DW_N **D Bus Write Enable** **Output**

Data space write enable signal. This signal is asserted when the D Bus address is in the data space range and the lead vehicle wishes to write to data space.

D_DMA_N **DMA Request** **Input**

This signal is asserted by an external DMA controller to request control of the D Bus.

D_DMA_GRANT

D Bus DMA Grant **Output**

The lead vehicle asserts this signal to grant control of the D Bus to the external DMA controller.

IOC_N **D Bus I/O Space Chip Select** **Output**

I/O chip select output signal. This signal is asserted when the D Bus address is in the I/O space range.

IOR_N **D Bus I/O Space Read Enable** **Output**
I/O read enable output signal. This signal is asserted when the D Bus address is in the I/O space range and the lead vehicle requests a read from I/O space.

IOW_N **D Bus I/O Space Write Enable** **Output**
I/O write enable output signal. This signal is asserted when the D Bus address is in the I/O space range and the lead vehicle wishes to write to I/O space.

3.2 External Registers Interface

XRDI[15:0] **External Register Data Input Bus** **Input**
Data bus from the selected external register to the lead vehicle.

XRDO[15:0] **External Register Data Output Bus** **Output**
Data bus from lead vehicle to the selected external register.

SEL_XR_RD[1:0]
External Register Read Select **Output**
These signals allow the multiplexing of the read data from the external registers onto the XRDI[15:0] input data bus.

XRW3_N, XRW2_N, XRW1_N, XRW0_N
External Registers Write Enables **Outputs**
Write enables for external registers 3–0.

XRR3_N, XRR2_N, XRR1_N, XRR0_N
External Registers Read Enables **Outputs**
Read enables for external registers 3–0.

3.3 Interrupts Interface

NMI_N **Nonmaskable Interrupt** **Input**
When this signal is asserted, the CWDSP1670 core in the lead vehicle calls the NMI service routine at vector address 0x0004. This interrupt cannot be internally masked by software. NMI_N is internally synchronized with the rising edge of the MCLK input clock.

IACK_NMI_N **Nonmaskable Interrupt Acknowledge** **Output**
The CWDSP1670 core asserts this signal to acknowledge receipt of the NMI_N interrupt.

INT2_N, INT1_N, INT0_N

Interrupts 0–2 **Inputs**

When any of these signals are asserted, the CWDSP1670 core calls the interrupt service routine at vector address 0x0016 for INT2_N, 0x000E for INT1_N, or 0x0006 for INT0_N. The interrupts can be internally masked by software. The interrupt input signals are internally synchronized with the rising edge of the MCLK input clock.

INT3_N **Externally Vectored Interrupt** **Input**

When this signal is asserted and not masked by the IM3 input, the CWDSP1670 core calls the service routine at the vector address specified by the INT3_VEC[15:0] input. INT3_N is internally synchronized with the rising edge of the MCLK input clock.

IACK_INT2_N, IACK_INT1_N, IACK_INT0_N

Acknowledge Interrupts 2 to 0 **Outputs**

The CWDSP1670 core asserts these signals to acknowledge branching to the service routine for the corresponding interrupt.

IACK_INT3_N **Acknowledge Interrupt 3** **Output**

The CWDSP core asserts this signal to acknowledge branching to the service routine for Interrupt 3.

IM3 **INT3_N Mask** **Input**

When this signal is asserted, the INT3_N interrupt input is masked from the core processor.

INT3_CNTX_EN

INT3_N Context Switching Enable **Input**

Tie this pin HIGH for automatic context switching when an INT3_N interrupt occurs.

INT3_VEC[15:0]

INT3_N Vector **Input**

This 16-bit input is used to specify the address of the service routine for the INT3_N input interrupt. Pins 0 and 1 must be tied to VSS. See Section 9, “Known Limitations” for further hints.

LEVEL_INT_MODE

Level Triggered Interrupt Mode Select **Input**

This signal must be tied LOW for edge-triggered interrupt mode or tied HIGH for level-triggered interrupt mode.

3.4 Clocks

MCLK **Master Clock** **Input**

All chip internal and output clocks are derived from MCLK.

OAK_CLK **Oak Clock** **Output**

Wait-stated copy of MCLK input skewed to align with the internal chip clock.

CLK_UG **Ungated Clock** **Output**

Non-wait-stated copy of MCLK input skewed to align with the internal chip clock to be used by the D Bus interface.

CLKOUT_EN **Clock Out Enable** **Input**

When asserted, enables all output clocks; OAK_CLK, CLK_UG, and TIMER_CLK.

TIMER_CLK **Timer Clock** **Output**

MCLK Input to the Timer gated by CLKOUT_EN.

3.5 Wait Controls

WAIT_CTL_N **Core Clock Wait States Control** **Input**

While this signal is asserted, wait states are inserted into the CWDSP1670 core's PMEM_CLK and CORE_CLK.

EMEM_WAIT **E Memory Wait State** **Input**

When this signal is asserted at reset, the M6 bit in the R3 register is cleared and there are no block 6 E memory wait states. When this signal is deasserted at reset, the M6 bit is set and the number of block 6 E memory wait states is determined by the setting of the W6 field in the R1 register (1 to 16 wait states). The M6 bit can also be set by software.

EDB_WAIT_EN

E Bus Automatic Wait State Enable **Input**

When this signal is asserted, E memory accesses have an automatic, single, wait state.

3.6 User I/O Interface

IU[1:0] **User Input** **Input**

These two signals are the CWDSP1670 core general purpose inputs. Their states are written to bits 10 and 11 of core Status Register 2.

OU[1:0] **User Output** **Output**

These two control signals are the general purpose outputs of the CWDSP1670 Lead Vehicle and reflect bits 9 and 8 of Status Register 2. These signals are deasserted at reset. After self-test boot, the self-test status is indicated on these two signals as in the following table:

OU[1:0]	Status
0b00	All tests pass
0b01	Data memory failure(s)
0b10	Program memory or ROM failure(s)
0b11	Other failure(s)

OUTP[7:0] **User Output Port** **Output**

These signals reflect the states of bits [7:0] of the Output Port register. See Output Port register description.

3.7 Reset, Abort, and Boot

RST_N **Reset** **Input**

Asserting this signal resets the core and clears all of its registers. RST_N must be held asserted for at least six MCLK cycles. After the core terminates the reset period, program execution restarts at program address 0x0000.

ABORT_N **Abort** **Input**

Initiates a breakpoint, allowing the debugger to regain control and halt core code execution.

BOOT_N and SELF_TEST_N **Inputs**

The on-chip boot ROM provides three different modes of boot operation, self-test boot, boot from D Bus, and normal boot, under control of the BOOT_N and SELF_TEST_N signals per the following table. The status

of SELF_TEST_N can be read from the D Bus Interface registers (see the STB bit on page 20).

BOOT_N	SELF_TEST_N	Boot Operation
0	0	Self-test boot. CWDSP1670 executes self test code in ROM during boot. See OU[1:0] signals.
0	1	D Bus boot. CWDSP1670 executes D Bus boot loading data from device located at the base of block 4 (0x8000) in D Bus data memory space.
1	x	Normal boot. CWDSP1670 executes code from address 0x0000.

BRRM1 **Branch to Self Indication** **Output**
BRRM1 is asserted when the core executes a BRR -l instruction, indicating an infinite loop at a single address. This indication can hence be used to determine when the end of a program has been reached or when the boot code branched to self.

MP_RESET **External Host Reset** **Input**
When this signal is asserted by an external host, the core is fully reset. MP_RESET must be held asserted for at least six CLK_UG (MCLK with no wait states) cycles. After the core terminates the reset period, program execution restarts at program address 0x0000 and the core boots. (See also MP_BOOT.) MP_RESET can be masked by the MPRC bit in the ScanICE Control register.

MP_ABORT **External Host Abort** **Input**
When this signal is asserted, it forces a breakpoint in the program and causes the core to go into the debug mode. MP_ABORT can be masked by the MPAC bit in the ScanICE Control register.

MP_BOOT **External Host Boot** **Input**
MP_BOOT can be asserted by an external host when the host deasserts MP_RESET to force the core to boot from the internal boot ROM. MP_BOOT has identical functionality to BOOT_N including the combination with

SELF_TEST_N for different boot modes. MP_BOOT can be masked by the MPBC bit in the ScanICE Control register.

RST_OUT_N Chip Reset Indicator Output

RST_OUT_N is asserted when:

- The RST_N input signal is asserted,
- the EXT_SCAN_RST input signal is asserted,
- the RST bit in the ScanICE Control register is set, or
- the MP_RESET input signal is asserted and not masked by the MPRC bit in the ScanICE Control register.

ABORT_OUT Chip Abort Indicator Output

This signal is asserted to indicate the On-Chip Emulation Module (OCEM) raised a breakpoint interrupt and put the core in debug mode.

3.8 ScanICE Interface

The first six signals listed below are compatible with existing OakDSPCore debugging systems. The last two signals (SCAN_DEBUG_EN and SCANICE_MODE) are additional ScanICE monitor and control signals that can be incorporated in your design.

EXT_SCAN_IN External Scan In Input

Serial scan test data in to the core ScanICE Unit from an external control system.

EXT_SCAN_OUT

External Scan Out Output

Serial scan test data out from the core ScanICE Unit to an external control system.

EXT_SCAN_CTL

External Scan Control Input

When asserted, triggers the core ScanICE Control register loading protocol.

EXT_SCAN_ALERT

External Scan Alert **Output**

This signal is asserted when the core is in ScanICE mode. It is used to alert the external control system that this state has been entered.

EXT_SCAN_CLK

External Scan Clock **Input**

External scan clock input to core.

EXT_SCAN_RST

External Scan Reset **Input**

Core reset input from the external control system.

SCAN_DEBUG_EN

ScanICE Debug Enable **Input**

Asserting this signal enables ScanICE debug.

Note: The debugger cannot be used unless this signal is asserted.

SCANICE_MODE

ScanICE Mode Indicator **Output**

When asserted, indicates that ScanICE debug mode is active.

3.9 Test Interface

The signals listed in this section are for the LSI Logic production testing only. For normal operation, connect them or leave them unconnected as indicated.

IIDDTN **Production Test** **Input**

Tie this pin LOW for normal operation.

TN **Production Test** **Input**

Tie this pin LOW for normal operation.

PROCOUT **Process Monitor Indicator** **Output**

Leave this pin unconnected.

PTST **Production Test Enable** **Input**

Tie this pin LOW for normal operation.

PTST_TE **Production Test Scan Chain Load Enable** **Input**

Tie this pin LOW for normal operation.

ESCAN_OUT **reserved** **Output**
Leave this pin unconnected.

ESCAN_EN **reserved** **Input**
Tie this pin LOW for normal operation.

ESCAN_IN **reserved** **Input**
Tie this pin LOW for normal operation.

3.10 Miscellaneous

PPROTECT_EN
Program Protection Enable **Input**
A typical method for attempting to read program memory is to copy the data in it to data memory using the MOVP instruction in code running from off-chip memory. When the PPROTECT_EN pin is tied HIGH, any attempt to read data from the I Bus using the MOVP instruction (executing from P bus space) results in garbage being read, thus preventing this data from being copied into data memory.

Note: This signal does not serve a useful purpose on the lead vehicle but is provided to demonstrate operation of this core feature.

DMMC_MODE **Input**
When asserted, the lead vehicle is memory mapped in the DMMC mode. When deasserted, normal memory mapping is used. See Section 5.2, “Data Memory.”

OCEM_SUSP **OCEM Suspend** **Input**
Asserting this signal disables the clocks within the OCEM, thus effectively reducing the power consumed by this module to zero. See the SUSPEND signal description in the CWDSP1670 DSP Core Technical Manual.

Note: The debugger cannot be used unless this signal is asserted.

STOP_MODE **Core Stop Mode Indicator** **Output**
When asserted, indicates that the CWDSP1670 core is in the Stop mode, that is, the core clocks are stopped.

VDD2 Core **CWDSP1670 Core Power** **Input**
+2.5 V to core.

VDD2 LV	**Lead Vehicle Power** +2.5 V to lead vehicle components external to the core.	**Input**
VDD4	**Power** +3.3 V to I/O buffers of lead vehicle.	**Input**
VSS	**Ground**	**Bidirectional**

4 Registers

This section describes the registers in the lead vehicle external to the CWDSP1670 DSP Core. Refer to the CWDSP1670 DSP Core Technical Manual for descriptions of the core's registers.

All of the lead vehicle registers are located in the top 1 Kword of the D Bus memory space. The registers have two addresses, one each for the normal and DMMC memory modes. The registers and their addresses are listed in Table 1.

Table 1 Lead Vehicle Registers

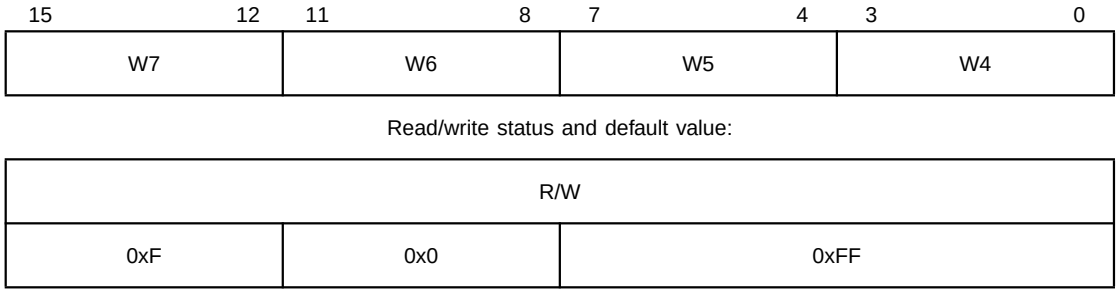
Register	DMMC Mode Address	Normal Mode Address	Function
R0	0xF7E0	0xEC00	Blank, provided for backward compatibility with previous lead vehicles.
R1	0xF7E1	0xEC01	Sets number of D Bus access wait states.
R2	0xF7E2	0xEC02	Set number of I/O access wait states.
R3	0xF7E3	0xEC03	Enable/disable E RAM wait states.
CGU	0xF7E4	0xEC04	Stop mode enable/disable and interrupt select for recovery.
Mode ID	0xF7E6	0xEC06	Holds base address of D Bus memory space
Output Port	0xF7E7	0xEC07	User output bits reflected on OUTP[7:0] output pins.
Timer Reload Value	0xF7E8	0xEC08	Timer starts at this value and counts down to zero.
Timer Control	0xF7E9	0xEC09	Controls timer speed, mode, and interrupt status.
Timer Count	0xF7EA	0xEC0A	Holds the Timer's current count.
Timer Count	0xF7EB	0xEC0B	Holds the Timer's current count.

Note: The register bits and fields labeled “Reserved” are non-functional. Although the descriptions for them tell you to clear them when writing to the registers, writing ones to them will not affect the operation of the lead vehicle.

4.1 D Bus Interface Registers (R1–R3)

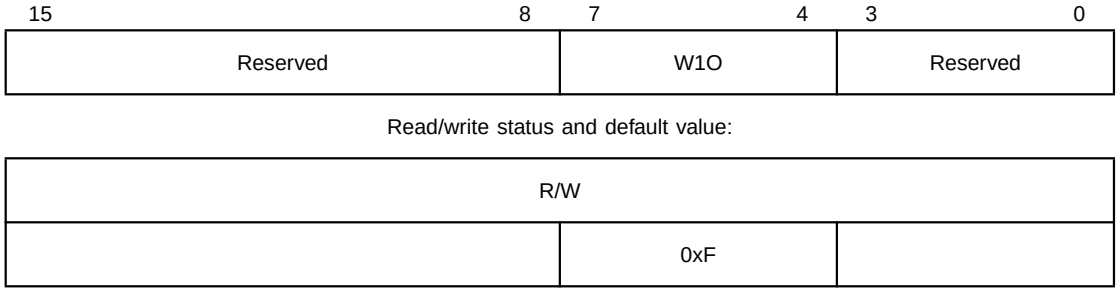
This section shows the layout of the R1 through R3 registers (see Figures 3 through 5) and describes their bits and fields. See Table 1 for register addresses.

Figure 3 R1 Register Layout



The four fields in this register set the number of wait states for memory blocks 4 (W4 field) through 7 (W7 field). An entry of 0x0 causes one wait state, 0x1 causes two wait states, and so on. The W6 field is effective only when the M6 bit in the R3 register is cleared.

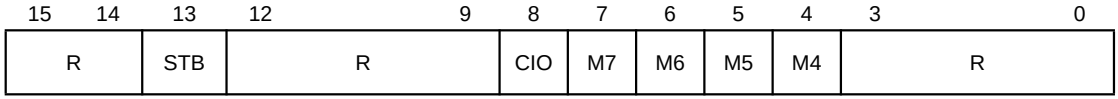
Figure 4 R2 Register Layout



The WIO field in this register sets the number of wait states for the D Bus I/O space. An entry of 0x0 causes one wait state, 0x1 causes two wait states, and so on.

Note: For correct operation of I/O accesses, use a minimum of two wait states (WIO = 0x1).

Figure 5 R3 Register Layout



Read/write status and default value:

R/W				
	See Descrip -tion		0x1F	

R **Reserved** **[3:0], [12:9], [15:14]**
Clear these bits when writing to this register.

M4, M5 **4, 5**
Not used, provided for backward compatibility with previous lead vehicles.

M6 **6**
Block 6 (0xC000–0xDFFF) of E Data memory is assigned to the on-chip E RAM. When M6 is set, the number of wait states for E RAM accesses is controlled by the W6 field in the R1 register. When M6 is cleared, there are no wait states.

M7 **7**
Not used, provided for backward compatibility with previous lead vehicles.

CIO **8**
Not used, provided for backward compatibility with previous lead vehicles.

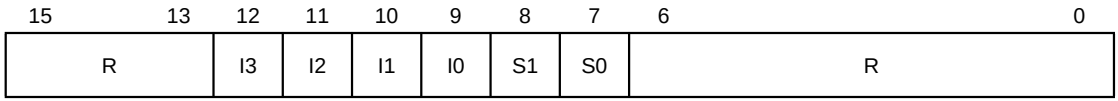
STB **Self-Test Indicator** **13**
This bit is set when the SELF_TEST_N input signal is asserted and cleared when SELF_TEST_N is deasserted.

Note. This bit is not registered, so any value written to it is ignored.

4.2 CGU Register

The Clock Generator Unit register is shown in Figure 6 and its bits are described following the figure. See Table 1 for the register's addresses.

Figure 6 CGU Register Layout



Read/write status and default value:

	R/W	
	0x00	

R **Reserved** **[6:0]**
Clear these bits when writing to this register.

S0, S1 **Clock Stop** **[8:7]**
When either of these bits are set, The CGU in the lead vehicle interrupts the main clock to the CWDSP1670 core stopping all clocks in the core.

I0–I3 **Restart Clock on Interrupt** **[12:9]**
When one of these bits is set, the associated interrupt (INT0–INT3) re-enables the core clocks.

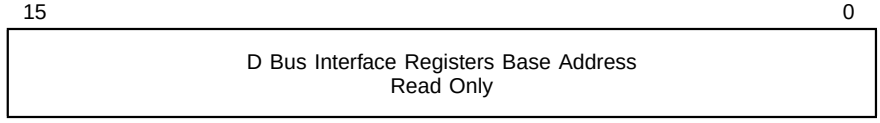
Note: The NMI (Nonmaskable Interrupt) always restarts the core clocks when it occurs during Stop mode.

R **Reserved** **[15:13]**
Clear these bits when writing to this register.

4.3 Mode ID Register

This register holds the base address of the D Bus Interface registers for the memory-mapped mode of the lead vehicle. When the DMMC_MODE input to the lead vehicle is asserted, memory is mapped in the DMMC mode and this register holds address 0xF7E0. When DMMC_MODE is deasserted for Normal mode memory mapping, this register holds address 0xEC00 (see Figure 7). See Table 1 for the addresses of this register.

Figure 7 Mode ID Register Layout



4.4 Output Port Register

The Output Port register is shown in Figure 8. The states of the OUTP7–0 bits are reflected on the OUTP[7:0] pins of the lead vehicle. You can use these bits and pins for such operations as bank switching when booting from D Bus memories. The upper address bits for those memories can be written into this register. See Table 1 for the register's addresses.

Note: The lead vehicle boot ROM uses OUTP[1:0] for generation of additional address lines during a boot from the D Bus. See Section 6.5, "Output Port."

Figure 8 Output Port Register Layout

15 … 8	7	6	5	4	3	2	1	0
R	OUTP 7	OUTP 6	OUTP 5	OUTP 4	OUTP 3	OUTP 2	OUTP 1	OUTP 0

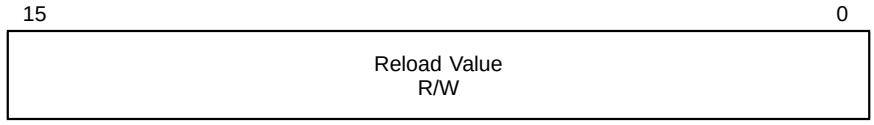
Read/write status and default value:

	R/W
	0x00

4.5 Timer Registers

The Timer registers include the Timer Reload, Timer Control, and Timer Count registers. The figures in this section show their layouts, and their bits and fields are described in the paragraphs following Figures 9 through 11.

Figure 9 Timer Reload Register



The value written into this register is the value the lead vehicle Timer returns to, after it counts down to zero, in Periodic mode (see bit 1 in the Timer Control register). See Table 1 for the addresses of this register.

Figure 10 Timer Control Register Layout

15	14 10	9	8	7	6	5	4	3	2	1	0
Stopped Indicator	R	Disable Reload after Stop	Enable NMI	Enable INT3	Enable INT2	Enable INT1	Enable INT0	Enable Pre-scaler 2	Enable Pre-scaler 1	Periodic Mode Enable	Timer Enable

Read/write status:

Read Only		Write Only

Timer Enable **0**

When this bit is set, the lead vehicle Timer is enabled. When this bit is cleared the Timer is stopped. This bit, with bit 9, can be used to stop and restart the Timer at the stopped count or the Reload Value.

Periodic Mode Enable **1**

When this bit is set, the Timer operates continuously. It starts at the Reload Value in the Timer Reload register, counts down to zero, returns to the Reload Value and continues counting down again.

When this bit is cleared, The Timer starts at the Reload Value, counts down to zero, and stops. The Timer Enable bit is cleared at stop. Setting the Timer Enable bit starts another single countdown.

Enable Prescaler 1 **2**

When this bit is set, Prescaler 1 divides the input clock to the Timer by 16. The timer clock is a copy of MCLK.

Enable Prescaler 2 **3**

When this bit and bit 2 are set, Prescaler 1 and 2 divide the input clock to the Timer by a total of 256. Prescaler 2 is not functional unless both bits are set.

Enable INT0–3 **[7:4]**

When any of these bits are set, the corresponding interrupt is asserted to the CWDSP1670 core when the Timer count reaches zero. This feature is disabled when the bit is cleared. Multiple interrupts can be enabled.

Enable NMI **8**

When this bit is set, NMI is asserted to the CWDSP1670 core when the Timer count reaches zero. This feature is disabled when the bit is cleared. NMI can be enabled in combination with any of the INT0–3 interrupts.

Disable Reload after Stop **9**

When this bit is set and the Timer is stopped and restarted with bit 0, it restarts at the stopped count. When this bit is cleared, the Timer restarts at the Reload Value.

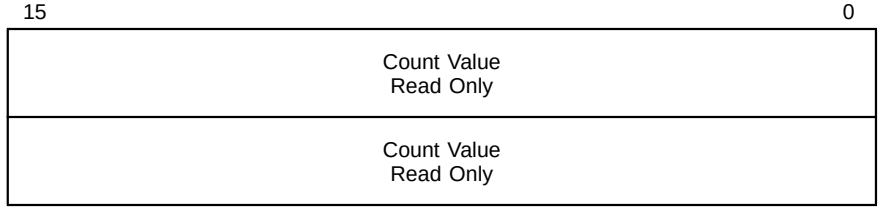
R **Reserved** **[14:10]**

Clear this bit when writing to this register.

Stopped Indicator **15**

This bit is set when the Timer is stopped and cleared when it is counting down.

Figure 11 Timer Count Registers



These two registers each contain the current count of the Timer.

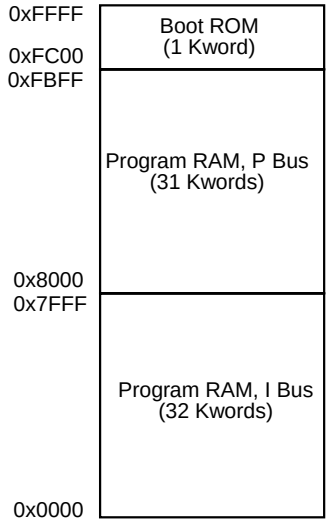
5 Memory Organization

The CWDSP1670 Core can address up to 64 Kwords of program memory and 64 Kwords of data memory. It contains X, Y, and E data memory buses, and I and P program memory buses.

5.1 Program Memory

As shown in Figure 1, 64 Kwords of program memory are included in the lead vehicle as a 32 Kword program RAM on the I Bus, a 1 Kword boot ROM and a 31 Kword program RAM on the P Bus. Figure 12 shows the memory map for these.

Figure 12 Program Memory Map



The D Bus boot program and self-test boot program are located in zero wait-state, on-chip ROM. See Section 6.8.1, “Self-Test Boot.”

5.2 Data Memory

The lead vehicle contains 44 Kwords of data memory configured as 32 Kwords of X RAM, 4 Kwords of Y RAM, and 8 Kwords of E RAM. The remaining 20 Kwords of data space are allocated to the core’s E Bus and

from there to an off-chip D Bus. Part of this space is used for E RAM and on-chip registers.

Previous CWDSP16X0 lead vehicles had two off-chip buses; C and D with programmable memory mapping by 8 Kword blocks. For compatibility with the software written for those lead vehicles, two mapping modes are provided in the CWDSP1670 lead vehicle, Data Memory Map Compatibility (DMMC) and normal modes.

Figure 13 shows the data memory map and Table 2 provides details about each memory range.

Figure 13 Data Memory Map

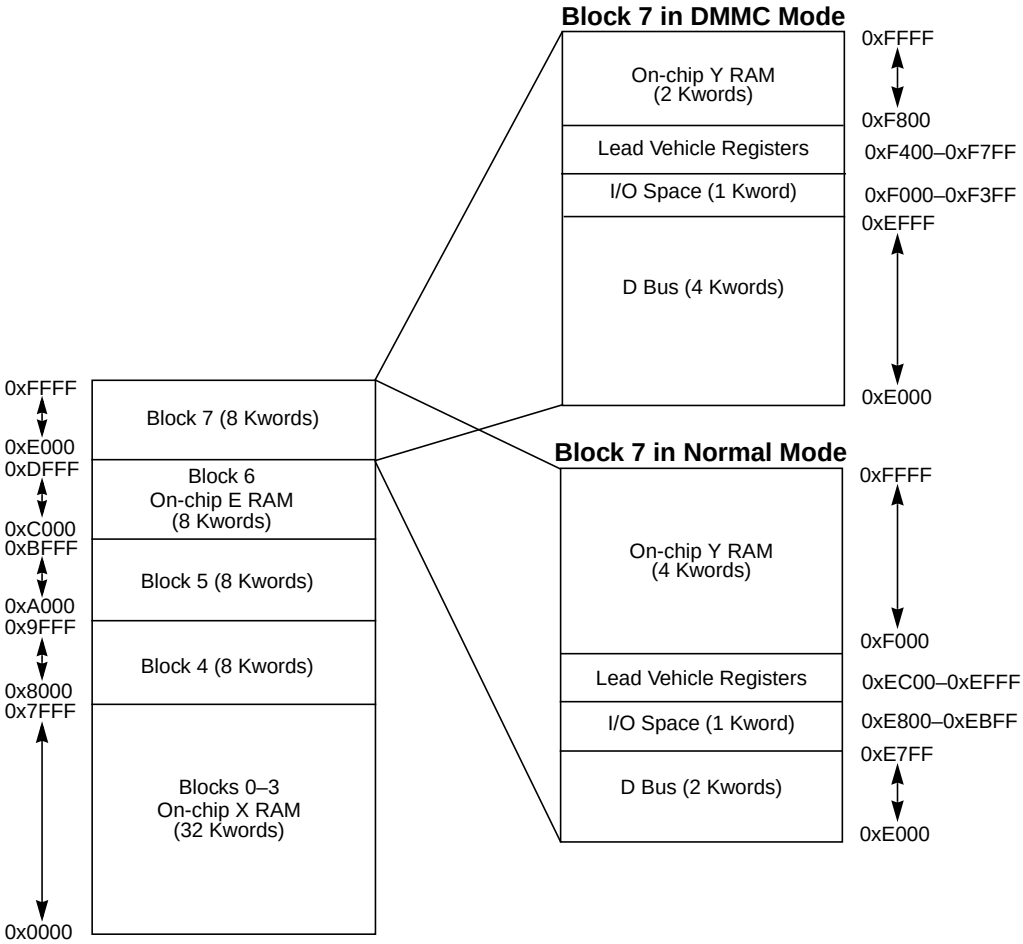


Table 2 Data Memory Allocation

Block	Range (DMMC Mode/ Normal Mode)	Size (Words)	Mapping	Wait States
0–3	0x0000–0x7FFF	32 K	On-chip X RAM	0
4	0x8000–0x9FFF	8 K	D Bus	1–16 (W4 + 1)
5	0xA000–0xBFFF	8 K	D Bus	1–16 (W5 + 1)
6	0xC000–0xDFFF	8 K	On-chip E RAM	0–16[1] (W6 + 1)
7	0xE000–0xEFFF/ 0xE000–0xE7FF	4 K/ 2 K	D Bus	1–16 (W7 + 1)
	0xF000–0xF3FF/ 0xE800–0xEBFF	1 K	D Bus I/O Space	2–16 (WIO + 1)
	0xF400–0xF7DF	992	On-chip Reserved	0
	0xF7E0–0xF7E3/ 0xEC00–0xEC03	4	On-chip D Bus I/F Registers	0
	0xF7E4/ 0xEC04	1	On-chip CGU Register	0
	0xF7E5/ 0xEC05	1	On-chip Reserved	0
	0xF7E6/ 0xEC06	1	On-chip Mode ID Register	0
	0xF7E7/ 0xEC07	1	On-chip Output Port Register	0
	0xF7E8–0xF7EB/ 0xEC08–0xEC0B	4	On-chip Timer Registers	0
	0xF7EC–0xF7FF/ 0xEC0C–0xEFFF	20/ 1012	On-chip Reserved	0
	0xF800–0xFFFF/ 0xF000–0xFFFF	2 K/ 4 K	On-chip Y RAM	0

1. When the M6 bit in the R3 Register is set, the number of wait states is controlled by the setting of the W6 field in the R1 Register. When M6 is cleared, there are no wait states.

Blocks 0-3, the lower 32 Kwords of data address space, are fully mapped as on-chip, closely-coupled, zero wait-state, X RAM, connected through the CWDSP1670 Core X Bus.

Blocks 4 and 5 are mapped onto the CWDSP1670 core's E Bus and, from there to the off-chip D Bus. The W4 and W5 fields in the R1 register (see page 19) determine how many wait states (1–16) are inserted during a transaction in each block. Each block is 8 Kwords in length.

Block 6 is mapped to the on-chip E RAM. When the M6 bit in the R3 register is set, the W6 field in the R1 register is programmed for the number of wait states (1–16). When M6 is cleared, there are no wait states regardless of the value in the W6 field. This feature allows the lead vehicle to operate at 80 MHz with a single wait state on the E memory or at <60 MHz with all of the memory (X,Y,E,I and P) within the lead vehicle zero wait-stated.

The lower section of block 7 is mapped onto the CWDSP1670 core's E Bus and from there to the off-chip D Bus, I/O space, and on-chip register addresses. The I/O space and register addresses take up 2 Kwords above the D Bus. The remainder is used for on-chip Y RAM.

In DMMC mode, Y RAM is 2 Kwords so the first 4 Kwords of this block can be mapped to the D Bus. In normal mode, Y RAM is 4 Kwords so only the first 2 Kwords of Block 7 can be mapped to the D Bus. The W7 field in the R1 register (see page 19) determines how many wait states (1–16) are inserted during a transaction on the D Bus. The W7 field also determines how many wait states (1–16) are inserted during a transaction on the I/O space.

Reading from and writing to memory areas marked reserved in Table 2 will have no effect on the operation of the CWDSP1670 Lead Vehicle. Data read from these areas will be undefined.

6 Functional Description

This section describes the lead vehicle's functional blocks in more detail than is provided in Section 2, "Functional Overview." Refer to that section for an overview and interconnection of the functional blocks

6.1 Core Overview

The CWDSP1670 is a 16-bit, fixed-point digital signal processor (DSP) core designed for middle-end to high-end telecommunications and consumer applications. This core provides a low-cost, high-performance solution for applications where low-power, high integration, and portability are a necessity. This core is a component of the LSI Logic CoreWare® Library, which contains cores for control, high-speed communication, and mixed-signal functions to complement quick time-to-market, customizable solutions. The CWDSP1670 DSP Core is designed by LSI Logic to be fully compatible with the DSP Group OakDSPCore® Instruction Set architecture allowing direct porting of existing code. The OakDSPCore family of cores are modified Harvard architectures, based on DSP Group's PineDSPCore® architecture.

The CWDSP1670 architecture contains dedicated buses for program and data memory. The core, shown in Figure 14, is composed of the following major components:

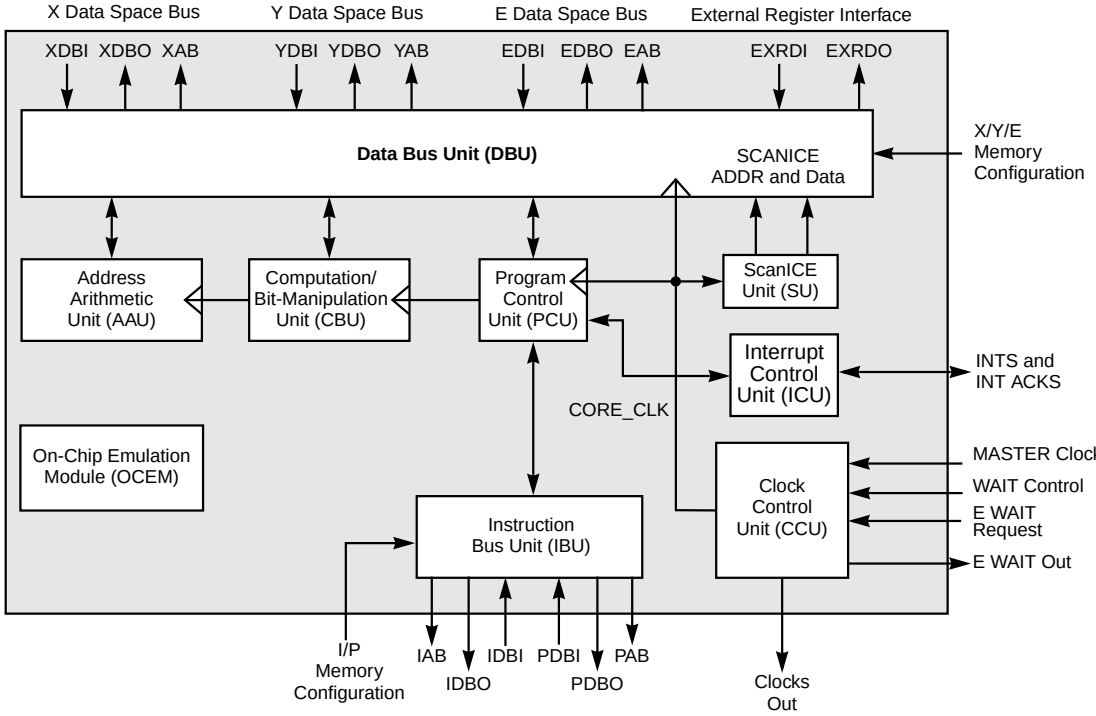
- Data Bus Unit (DBU)
- Address Arithmetic Unit (AAU)
- Program Control Unit (PCU)
- Computation/Bit Manipulation Unit (CBU)
- ScanICE Unit (SU)
- On Chip Emulation Module (OCEM)
- Instruction Bus Unit (IBU)
- Clock Control Unit (CCU)
- Interrupt Control Unit (ICU)

The DBU interfaces internal core components to the off-core X, E, and Y data memories and peripherals. Each memory interface includes an address bus; a separate input and output data bus; memory, read, and write enables; and a synchronous/asynchronous mode select. Sixteen memory bank enable outputs associated with contiguous, 4096-word, memory areas provide faster memory selection. The core has a data address space of 64 Kwords. The X/Y/E Memory Configuration input strap pins define the address boundaries between the X, E, and Y data memories for the core.

The AAU contains the general purpose registers, stack pointer, and index register. It also contains two identical arithmetic units to generate sequences of addresses using the AAU registers. Two addresses may be generated on each cycle for simultaneous access of both X- or E-memory and Y-memory spaces.

The PCU controls the sequencing of the core program. It fetches instructions; generates program memory addresses; handles interrupts with the ICU; and sequences branches, calls, and instruction repeats. The PCU generates the controls for the rest of the CWDSP1670 DSP Core.

Figure 14 CWDSP1670 DSP Core Block Diagram



Instruction Bus Interfaces

The CBU contains arithmetic, bit, and word manipulation logic and the accumulators. It includes the multiplier ALU, barrel shifter, bit function unit (BFU), and the Ax and Bx accumulators. There is also a saturation unit and bus alignment/sign extension logic.

The CWDSP1670 DSP Core incorporates support for both the standard OakDSPCore Combo Debug Interface (CDI) and the LSI Logic ScanICE in-circuit debugging systems. The ScanICE Unit (SU) uses the serial test scan chain and On-Chip Emulation Module (OCEM) to provide full debug functionality without the need for off-chip parallel buses, dual-ported memory or a monitor program. Single-stepping, interrupt, repeat loop, trap, program, data and address value match breakpoint facilities are all provided. ScanICE is fully integrated into the CWDSP1670 core and is 100% compatible with the standard OakDSPCore system development software tools.

Note: The CWDSP1670 Lead Vehicle supports only ScanICE debug.

The IBU interfaces internal core components to the off-core I and P program memories. Each memory interface includes an address bus; a separate input and output program data bus; memory, read, and write enables; 16 memory bank enables; and a synchronous/asynchronous mode select. The core has a program address space of 64 Kwords. The I/P Memory Configuration input strap pins define the address boundary between the I and P program memories to the core.

Note: The CWDSP1670 Lead Vehicle internal memories are all synchronous.

The CCU generates all the clocks for the core and the output clocks from the MASTER clock input. It also implements the WAIT functionality. Connecting the EDB_WAIT_OUT pin to the EDB_WAIT_REQ pin causes an automatic, single, wait state in each E Bus access. The CCU also supplies optimized clocks for data and program memory access. Clock division and clock stop modes are not provided by the core. This ensures that the core is fully static and provides flexibility for power-optimized clock control logic to be implemented off-core.

The ICU handles the interrupt protocols for each of the six interrupts and generates a separate acknowledge signal for each one. The ICU generates the interrupt vector and status signal for the PCU and also prioritizes incoming interrupts.

Note: This functionality can be seen using the EDB_WAIT_EN input of the lead vehicle.

6.2 Clock Generator Unit

All clocks in the lead vehicle are derived from the input master clock (MCLK) by the Clock Generation Unit (CGU). Clocking inside the CWDSP1670 Core is isolated from the rest of the lead vehicle. The specially-generated, moving-edge, memory clocks are only used within the lead vehicle for the closely coupled memories. All transactions on the E Bus interface between the core and the rest of the lead vehicle have at least one inserted wait state except for the E RAM which can be programmed for 0-16 wait states.

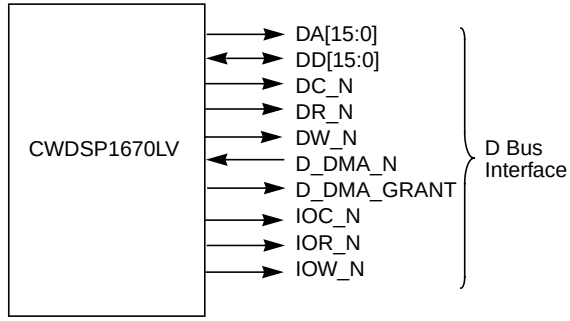
The CGU contains logic that interrupts its output clock to the core when the S0 or S1 bit in the CGU register is set and to restart the core input clock when it receives an interrupt on the INT0–3 or NMI inputs as programmed into the CGU register.

6.3 D Bus Interface

The D Bus interface buffers the E Bus data and address lines off chip for the selected D Bus address ranges. These ranges are selected by the DMMC_MODE input signal (see page 17) and bits in the R3 register of the interface (see page 20). One Kword of D Bus space is for I/O and the remainder is for memory.

The interface also generates asynchronous read, write, and chip enable signals for both the D Bus space and I/O space. Figure 15 Shows the interface signals. They include a 16-bit address and bidirectional data bus, memory chip select and read/write enables, I/O chip select and read/write enables, and a DMA request input and bus grant output. The data or I/O controls (chip enable, read, and write) become active in the appropriate address ranges.

Figure 15 D Bus Interface Signals



The interface also controls the generation of wait states for D Bus device accesses. The number of wait states generated for each block of D Bus memory space is determined by a four-bit field in the R1 register (see page 19) of the D Bus interface. Another 4-bit field in the R2 register (see page 19) controls the number of wait states for I/O space accesses.

The DDMA_N input allows an off-chip device to gain control over the D Bus. When this signal is asserted, the CWDSP1670 core is wait-stated when it writes to or reads from the D Bus. The D Bus data and address lines are tristated by the CWDSP1670 Lead Vehicle after the completion of the current instruction. The core and buses remain in this state until the next rising edge of MCLK after DDMA_N is released.

Figures 16 through 19 show the timing for the various D Bus read and write transactions.

Figure 16 D Bus Data Read Timing

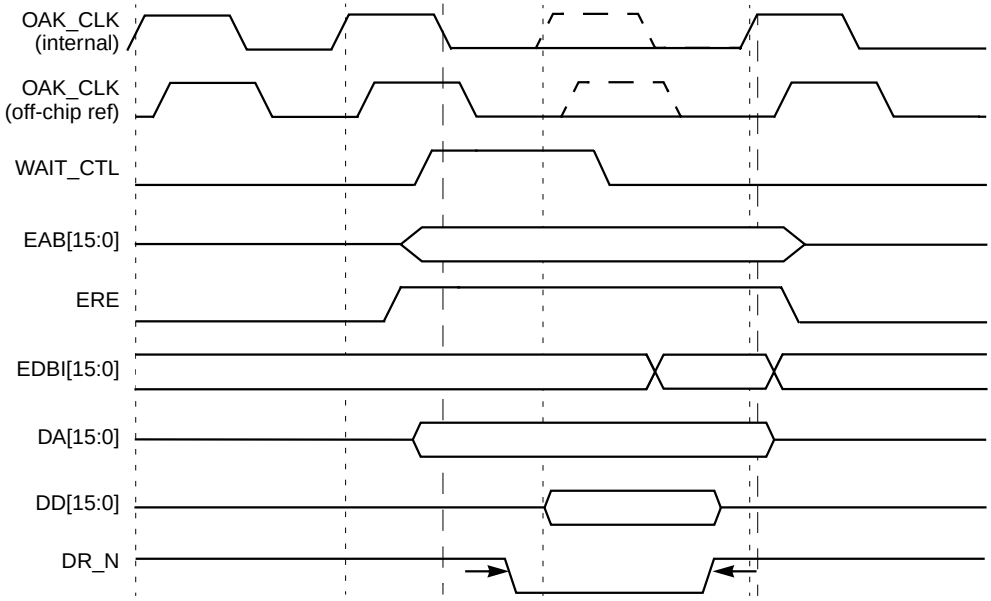


Figure 17 D Bus Data Write Timing

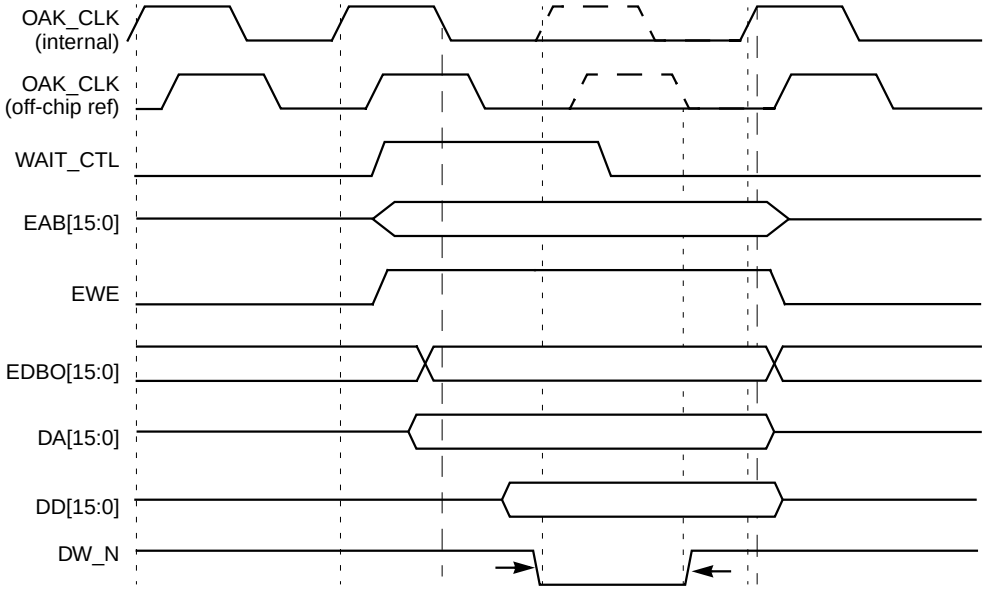


Figure 18 D Bus I/O Read Timing

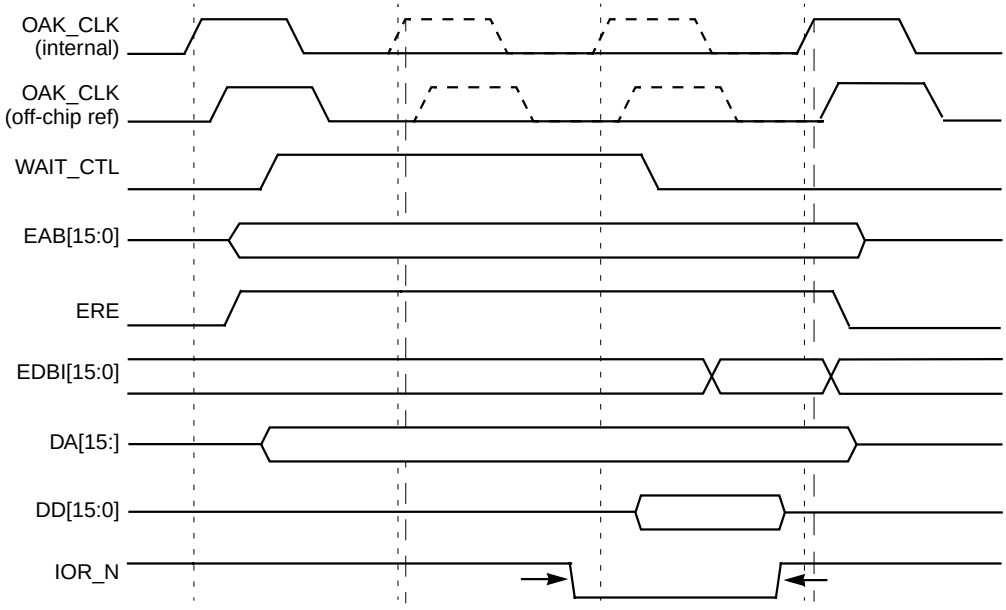
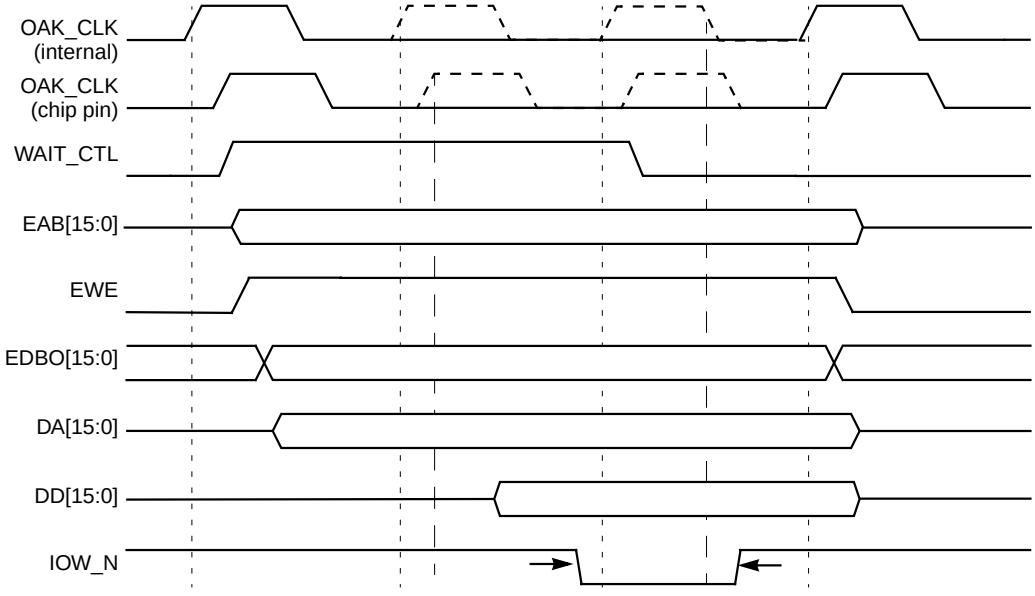


Figure 19 D Bus I/O Write Timing



6.4 Timer

A Timer with three memory mapped registers resides in E Bus space. The Timer generates an interrupt when its count reaches zero. This interrupt is routable to any of the user defined interrupts (INT0, INT1, INT2, INT3, or NMI) under control of the Timer Control register (see page 23).

The input clock to the Timer is MCLK gated by the CLKOUT_EN input signal. The Timer clock and TIMER_CLK output of the chip are enabled when CLKOUT_EN is HIGH. The Timer has two, 4-bit prescalers which can be enabled or disabled using the Timer Control register. Each prescaler divides the clock input to the Timer by 16. If only one prescaler is required, then prescaler 1 should be enabled. The Timer has a modulo count capability with a 16-bit, modulo, reload value specified in the Timer Reload register (see page 23).

Another bit in the Timer Control register determines whether the Timer operates in periodic or single count-down mode. In periodic mode, the Timer counts down to zero, reloads the value specified in the Timer Reload register, and starts counting down again. In single count-down mode, the counter stops when it reaches zero and clears the Timer Enable bit.

The Timer may also be halted at any point and restarted by clearing and resetting the Enable bit in the Timer Control register. Restart can continue from the stop count or from the Reload Value depending on the state of the Disable Reload after Stop bit in the Timer Control register. The Timer Control register also contains a Timer Stopped indicator bit.

The Timer Count registers are two 16-bit registers that are continually updated to the current Timer count.

6.5 Output Port

The Output Port includes a 16-bit register in the D Bus address space and eight, general-purpose output signals, OUTP[7:0]. The states on the output pins directly reflect the states of the first eight bits in the register. These outputs may be used as general purpose outputs or to perform bank-switching when booting from D Bus memories by providing the upper address bits for those memories.

Note: The boot code on the CWDSP1670 Lead Vehicle assumes the OUTP[1:0] pins carry the upper address bits for bank switching of memory on the D Bus when a boot from the D Bus occurs and the code size to be loaded is greater than 16383 words, as shown below:

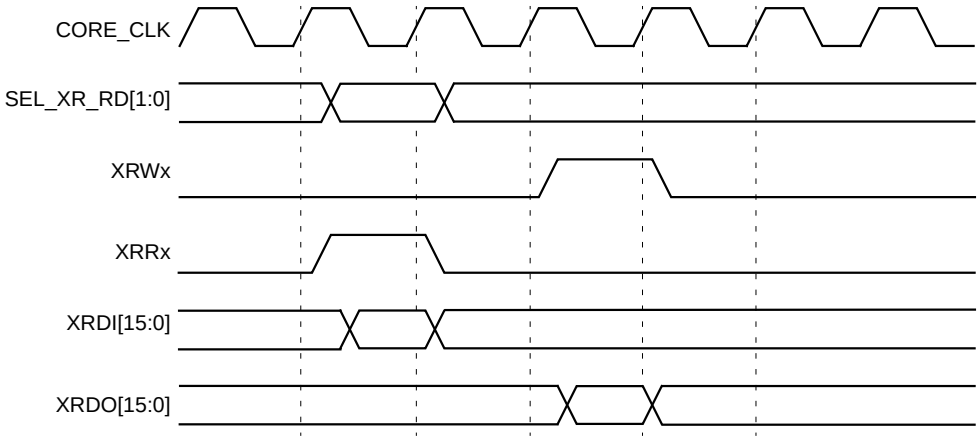
Code Size (words)	OUTP1 (A15)	OUTP0 (A14)
0–16383	0	0
16384–32767	0	1
32768–40151	1	0
49152–65535	1	1

6.6 External Register Interface

The external register interface of the CWDSP1670 Lead Vehicle permits you to use four, 16-bit, external registers in your design. The interface is separated from the other data interfaces and not part of the lead vehicle memory map. The registers can be accessed by most of the CWDSP1670 instructions. Figure 20 shows the timing of the interface signals.

Note: The external registers are not automatically cleared at reset.

Figure 20 External Registers Interface Timing



6.7 Interrupts

Four maskable interrupts and one nonmaskable interrupt are available as chip inputs. The chip outputs acknowledges for each. All interrupt inputs at the chip level are active LOW. The CWDSP1670 Lead Vehicle supports both edge- and level-triggered interrupts. Selection between these modes of operation is made using the LEVEL_INT_MODE input.

Any of the interrupt inputs can be programmed to occur when the Timer reaches zero as described in Section 4.5, “Timer Registers.” See the *CWDSP1670 DSP Core Technical Manual* for further detail on interrupts.

Note: All interrupts pass through two resynchronization flip-flops clocked by the MCLK input. There is no need to provide additional resynchronization off-chip.

6.8 On-chip Boot ROM

The states of the BOOT_N/MP_BOOT and SELF_TEST_N input signals to the lead vehicle when RST_N or MP_RESET is deasserted determine the boot mode as shown in Table 3.

Table 3 Boot Mode Selection

BOOT_N/ MP_BOOT	SELF_TEST_N	Boot Mode
0	0	Self-test Boot. CWDSP1670 executes self-test code in ROM at boot.
0	1	D Bus boot. CWDSP1670 executes D Bus boot loading data from device located at the base of block 4 (0x8000) in D Bus data memory space.
1	X	Normal boot. CWDSP1670 executes boot code from address 0.

The status of the SELF_TEST_N signal is indicated by the STB bit in the R3 register (see page 20).

6.8.1 Self-Test Boot

The self-test boot program includes testing of all RAM memories and checksum tests of the on-chip ROM as well as tests of basic

CWDSP1670 core functionality and timer operation. Upon completion of the testing, or at any point where a failure is detected, the code sets up self-test status indication on the OU[1:0] outputs as shown in Table 4. It then executes a BRR $-1 (branch to self) instruction, driving the chip's BRRM1 output HIGH to indicate completion of the self test.

Table 4 Self-Test Status Outputs

OU[1:0]	Status
0b00	All tests pass
0b01	Data memory failure(s)
0b10	Program memory or ROM failure(s)
0b11	Other failure(s)

6.8.2 Boot from D Bus

D Bus booting uses an approach similar to that which was used for C Bus booting on previous CWDSP16xx lead vehicles. The boot code reads:

- An initialization value for the R1 Register from address 0x8000.
- The length of the program to be loaded from the D Bus device at address 0x8001.
- The program load address from 0x8002.
- The run address from 0x8003.

The program is then copied from the D Bus device starting at address 0x8004 into program RAM at the specified load address. Once the program has been copied, the boot code branches to the specified run address. To support loading of large programs from the D Bus, the boot code assumes that the OUTP[1:0] output pins of the CWDSP1670 Lead Vehicle are being used for addressing purposes if the code length is greater than 16383 words in length. See Section 3.6, "User I/O Interface," for further information.

6.8.3 ROM Guard

There is a BRR -1 instruction at the start of the boot ROM (address 0xFC00). This branch prevents the core from accidently running into the boot ROM area.

6.9 ScanICE Interface

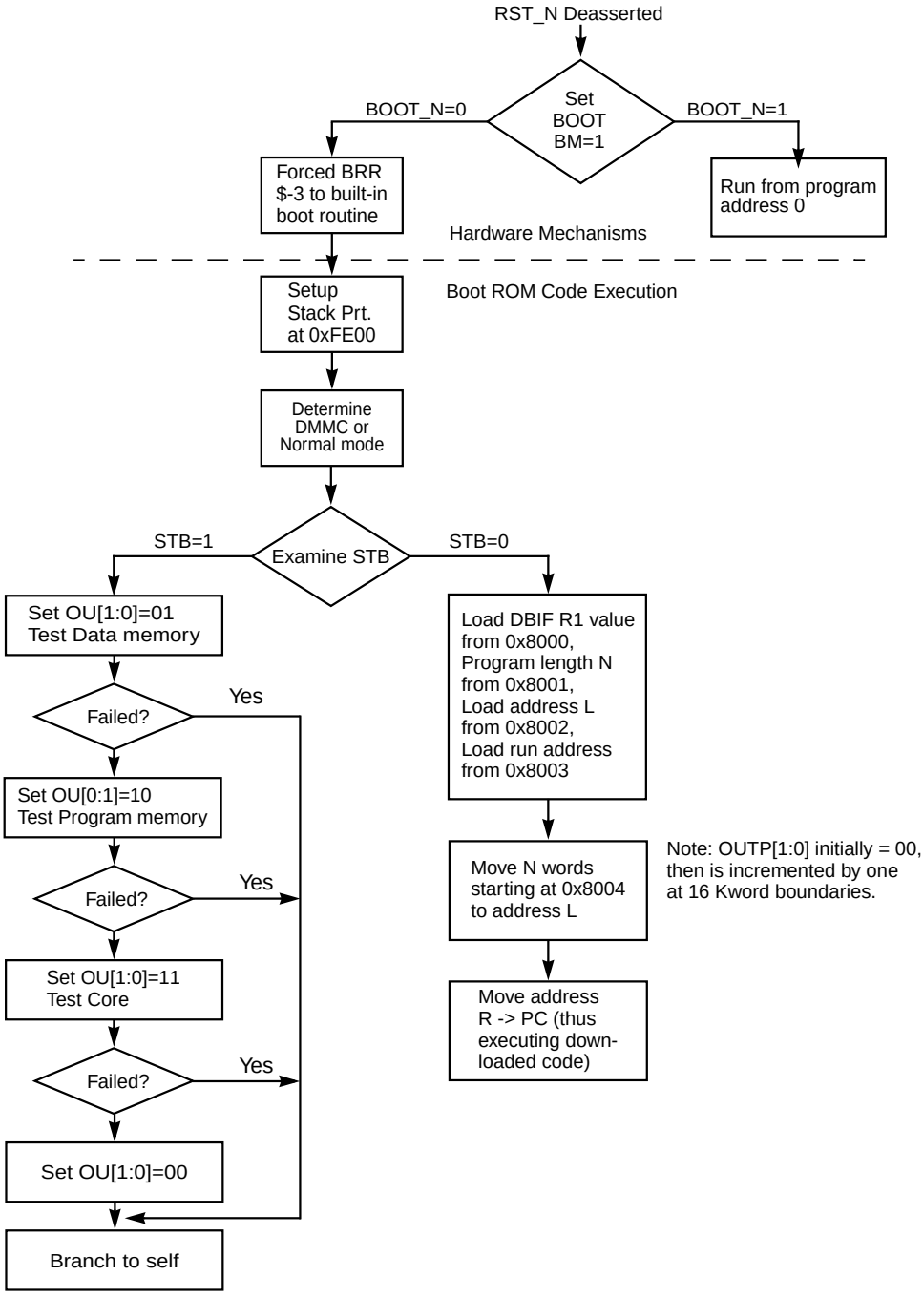
The ScanICE I/O signals of the CWDSP1670 core are brought out to the lead vehicle pins. The standard six signals, compatible with the existing PDIC card, are provided. Two additional ScanICE control and monitor signals are also made available. See Section 3.8, “ScanICE Interface.”

---

7 Sample Boot Program

Figure 21 shows a flowchart for the boot process. Sample boot code is provided following the flowchart.

Figure 21 Lead Vehicle Boot Flow



```
;LSI LOGIC;LSI LOGIC

;26 MARCH 1998

;Boot and Softbist program
;for CWDSP1670 Lead Vehicle

;Boot Program
;R0 IO Base address E800/F000
;R1 retrieval address on the D Bus 8000-bfff
;R2 Output port register address F7E7/EC07
;R3 Start address of y data space
;R4 Target address in Program space on chip 0000-fbff
;R5 Beginning of program address in Program space
;RB Contains base address for the other memory mapped peripherals
;A0 Contains program length value which is decremented with each move
;A1 Used as register for doing arithmetic manipulations
;B0 Contains first illegal program space address FC00
;B1 Contains the current D Bus memory boundary limit address c000/bc03
;

;************** FIR Definitions ***************************
.EQU QUANT 14                       ; Sample quantisation
.EQU            Ntaps   8           ; Number of filter taps
.EQU            Nout    16          ; Number of output data
.EQU DLYsaveAddr 0x0008
.EQU INaddr0x0019
.EQU DLYaddr    0x0000
.EQU OUTaddr0x0009
.EQU COEFaddr0xfd00
; Input samples
; Apply an impulse to get the filter transfer function

.CODE S_Main
.ORG 0x0000
    br 0xfffe
    nop
    nop
.ORG 0xfffe
    br main

.ORG 0xfc00
tg1:brr -1                  ;ROM guard
    nop
chksum:     DW 0x9d29
Parity:     DW 0x913f
Version:DW 0x111d;version number of chip / boot rom code
samples:DW FRACT(1.0,QUANT),0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
coeffs:DW FRACT(0.5,QUANT)          ; h(n)=0.5 x h(n-1)
    DW FRACT(0.25,QUANT)
    DW FRACT(0.125,QUANT)
    DW FRACT(0.0625,QUANT)
    DW FRACT(0.03125,QUANT)
```

```
    DW FRACT(0.015625,QUANT)
    DW FRACT(0.0078125,QUANT)
    DW FRACT(0.00390625,QUANT)
main:dint                       ;disable the interrupts
    set ##0x0300, st2           ;set output bits to indicate operation
    mov ##0xfe00, sp            ;puts the stack pointer in y space
    call setmode                ;calls a subroutine which sets up registers for the
                                        ;mode
mode:mov (rb+#0x3), a1          ;read value of dbi r3
    tstb a1l, #13               ;testing boot/self test control bit
    br test, eq                 ;goes to softbist if control is 1
    rst ##0x0100, st2           ;initialize user output pins

    call transfer               ;calls transfer routine
    rst ##0x0300, st2           ;initialize user output pins to indicate
                                        ;successful boot
runpoint: mov r5, pc            ;transfer start address to PC
    nop                         ;required for above instruction
    nop
error:  brr -1                  ;something has gone wrong
    nop

;*************************************************************************

setmode:mov ##0xF7E0, rb        ;starts the check for ID mode
    mov ##0xf000, r0            ;sets up IO base address for DMMC mode
    mov ##0xf800, r3            ;sets y data space address
    mov r3, a1                  ;move y address value into a1
    mov ##0xf7e6, r1            ;read the contents of id register into r2
    mov (r1), r2
    mov a1l, (rb+#0x06)         ;try to write value to the ID register
    mov (rb+#0x06), a1          ;read back the value in the ID register into a1
    cmp rb, a1                  ;if the values are the same then ID register
                                ;correct, should not read IO base address value
    ret eq                      ;Return if the test is OK
    mov r2, (r1)                ;move the original value back to the tested address

    mov pc, cfgj
    mov ##0xEC00, rb            ;Assuming normal mode otherwise
    mov ##0xe800, r0            ;sets up IO base address for normal mode
    mov ##0xf000, r3            ;sets y data space address
    mov r3, a1                  ;move y address value into a1
    mov ##0xec06, r1            ;read the contents of id register into r2
    mov (r1), r2
    mov a1l, (rb+#0x06)         ;try to write value to the ID register
    mov (rb+#0x06), a1          ;read back the value in the ID register into a1
    cmp rb, a1                  ;if the values are the same then ID register
                                ;correct, should not read IO base address value
    ret eq                      ;Return if the test is OK
    mov r2, (r1)                ;move the original value back to the tested address
    mov pc, cfgj
    br error
```

```
;*************************************************************************
transfer: mov ##0x8000, r1        ;move first byte add of Dbus address into r1
    mov rb, a0                    ;move mode base address in to a0
    add #0x07, a0                 ;address of the output port
    mov a0l, r2                   ;move output port address into r2
    mov (r2), a0l                 ;move contents of output port register to a0
    rst ##0x0003, a0l             ;set output port bits 0 & 1 to 0
    mov a0l, (r2)                 ;send the new value to the output port register
    mov ##0xc000, b1l             ;set first dbus memory boundary
    mov ##0xfc00, b0l             ;set highest program p address for program
    mov (rb+#0x01), a1            ;read value of r1 dbif
    mov (r1)+, a0l                ;move edb wait control word into r5
    and ##0x0f00, a1              ;mask bits from r1 dbif
    and ##0xf0ff, a0              ;mask bits from control word
    or a1l, a0                    ;combine the control word and r1 dbif
    mov a0l, (rb+#0x01)           ;move dbus wait states to r1 dbif
    mov (r1)+, a0l                ;move program length into a0
    mov (r1)+, r4                 ;move program memory destination address
    mov (r1)+, r5                 ;move program start address
    cmp #0x00, a0                 ;check for 0000 utility request
    ret eq                        ;go on to the run 0000 utility
    dec a0                        ;decrement program length control word
    mov a0l, y                    ;set up counter for loop
    bkrep  y, >%tag1-1            ;repeat the sequence for the length of prog
     mov r1, a1l                  ;checking for Dbus memory boundarys
     cmpu b1l, a1                 ;check for Dbus boundary addresses
     call bound, ge               ;go to subroutine if Dbus boundary is met
tag2: movd (r1)+, (r4)+           ;move data memory to program memory
%tag1:ret
bound:mov(r2), a1l                ;read the value of the output port register
    mov ##0x8000, r1              ;start on the next RAM bank at address 8000
    tstb a1l, #0x01               ;test bit 1
    br new, neq
    tstb a1l, #0x00               ;test bit 0, precaution
    br error, eq                  ;should not have reached here with this value
    mov pc, cfgj
    mov ##0xbc03, b1l             ;change highest address for reading from DBus
new:inc a1                        ;point at the next RAM bank
    mov a1l, (r2)                 ;write new bank bits to output port register
    ret                           ;go back to the loading routine
;*************************************************************************
;Soft BIST program
test:rst ##0x0200, st2            ;reset value of user output pins
    mov ##0xfe00, sp
    mov ##0x0000, a0l             ;x first 4k parameter indicates start address
    mov ##0x1000, a1l             ;parameter indicates length of block I
    call test_dmem_rr4k16
    mov ##0x1000, a0l
    call test_dmem_rr4k16
    mov ##0x2000, a0l
    call test_dmem_rr4k16
    mov ##0x3000, a0l
    call test_dmem_rr4k16
```

```
    mov ##0x4000, a0l
    call test_dmem_rr4k16
    mov ##0x5000, a0l
    call test_dmem_rr4k16
    mov ##0x6000, a0l
    call test_dmem_rr4k16
    mov ##0x7000, a0l            ;x 8th 4k start address block VIII
    call test_dmem_rr4k16
    mov pc, cfgj
    mov ##0xc000, a0l            ;e 1st 4k start address block I
    call test_dmem_rr4k16
    mov ##0xd000, a0l            ;e 2nd 4k start address block II
    call test_dmem_rr4k16
    mov ##0x4000, sp             ;y 4k/2k move stack pointer to X
    mov r3, a0l                  ;r3 contains base address of y memory
    cmpv ##0xf800, r3            ;check to determine size of y memory
    moda shr, a1, eq             ;halves value if y=2k
    call test_dmem_rr4k16

    rst ##0x0100, st2            ;reset value of user output pins
    set ##0x0200, st2            ;indicate program memory testing

    mov ##0x0000, a0l            ;start of I program memory
    mov ##0x4000, a1l            ;length of block I
    call test_pmem_rr16k16
    mov ##0x4000, a0l            ;start of I program memory block II
    call test_pmem_rr16k16
    mov ##0x8000, a0l            ;start of P program memory block I
    call test_pmem_rr16k16
    mov ##0xc000, a0l            ;start of P program memory block II
    mov ##0x3c00, a1l            ;length of block II
    call test_pmem_rr16k16
    mov pc, cfgj

    call Crom
    set ##0x0100, st2            ;indicate Other tests in progress
    call Fir
    call Crit
    ; insert lv register tests
    rst ##0x0300, st2
stop:brr -1              ;halt the boot routine
    nop
```

```
;***************************************************************************
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; RCS $Id: bb_rout.asm,v 1.13 1998/09/30 14:23:57 kerryl Exp $
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Filename: rr4k16d_test.asm
;
; Purpose: Memory dependend algorithms library
;
; Context: March C ASIC-driven Memorytest
;
; Author: LSISoftBist RamBist Software Code Compiler
;
; Copyright (c), LSI Logic, 1996
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Update history
;
; RCS $Log: bb_rout.asm,v $
; RCS Revision 1.13  1998/09/30 14:23:57  kerryl
; RCS Adding functionality to boot sequence
; RCS
; RCS Revision 1.12  1998/07/30 15:16:08  kerryl
; RCS update to softbist for new memories
; RCS
; RCS Revision 1.11  1998/07/20 10:33:12  kerryl
; RCS change to boot up routine to satisfy GLS
; RCS
; RCS Revision 1.10  1998/06/23 16:32:51  kerryl
; RCS br change
; RCS
; RCS Revision 1.9  1998/06/11 08:55:41  kerryl
; RCS modifying the boot version value and checksums
; RCS
; RCS Revision 1.8  1998/06/03 16:05:27  kerryl
; RCS modifying the boot version value and checksums
; RCS
; RCS Revision 1.5  1998/05/12 16:41:51  kerryl
; RCS change in script
; RCS
; RCS Revision 1.4  1997/09/26 14:59:09  wiesner
; RCS added checkboard background option
; RCS
; RCS Revision 1.3  1997/09/19 12:53:32  wiesner
; RCS alpha version for marco
; RCS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; March C testroutine
; prototype: void test_dmem_rr4k16(void *sa, unsigned int sz, error_code* err)
;
; formal parameters -----------------------------------------------------
```

```
;
; return <== a0l (void dummy)
;    &sa ==> a0l
;     sz ==> a1l
;   &err ==> b0l
;
; local used registers ------------------------------------------------
;
; r0 = pointer tp testpattern
; r1 = latch addresspointer
; r2 = row size in words
; r3 = temp location
; r4 = pointer to address
; y = size register
; r5 = free pointer
; sv = shift value


test_dmem_rr4k16:

push r0
push r1
push r2
push r3
push r4
push r5
push rb
push y
push sv
push b0l
push a0l
push a1l
; initialize addresspointer
mov a0l, r4

; calculate shifter value and/or (pass)loopcounter
moda shr4, a1
;moda shr, a1
moda dec,a1
mov a1l,y
mov ##0x03, r2

; store addresspointer
mov r4, r1

; Pass 1: increasing addresses ----------------------------------------------

mov ##0xffff, a1l
mov ##0x0000, a0l
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       mov a1l, (r4)             ; write pattern
       cmpu (r4)+, a1
```

```
       br stopd, neq
       mov a0l,   (r4)           ; write pattern
       cmpu (r4)+, a0
       br stopd, neq
       nop
%lab3: bkrep r2, >%lab2-1
       mov a0l, (r4)             ; write pattern
       cmpu (r4)+, a0
       br stopd, neq
       mov a1l,   (r4)           ; write pattern
       cmpu (r4)+, a1
       br stopd, neq
       nop
%lab2: nop
%lab:mov r1, r4                  ; restore addresspointer

; Pass 2: increasing addresses ------------------------------------------------


bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       cmpu (r4), a1
       br stopd, neq
       mov a0l, (r4)             ; write pattern
       cmpu (r4)+, a0
       br stopd, neq
       cmpu (r4), a0
       br stopd, neq
       mov a1l,   (r4)           ; write pattern
       cmpu (r4)+, a1
       br stopd, neq
       nop
%lab3: bkrep r2, >%lab2-1
       cmpu (r4), a0
       br stopd, neq
       mov a1l, (r4)             ; write pattern
       cmpu (r4)+, a1
       br stopd, neq
       cmpu (r4), a1
       br stopd, neq
       mov a0l,   (r4)           ; write pattern
       cmpu (r4)+, a0
       br stopd, neq
       nop

%lab2: nop
%lab: mov r1, r4

; Pass 3: increasing addresses ------------------------------------------------

bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       cmpu (r4), a0
```

```
       br stopd, neq
       mov a1l, (r4)             ; write pattern
       cmpu (r4)+, a1
       br stopd, neq
       cmpu (r4), a1
       br stopd, neq
       mov a0l,   (r4)           ; write pattern
       cmpu (r4)+, a0
       br stopd, neq
       nop
%lab3: bkrep r2, >%lab2-1
       cmpu (r4), a1
       br stopd, neq
       mov a0l, (r4)             ; write pattern
       cmpu (r4)+, a0
       br stopd, neq
       cmpu (r4), a0
       br stopd, neq
       mov a1l,   (r4)           ; write pattern
       cmpu (r4)+, a1
       br stopd, neq
       nop
%lab2: nop
%lab: modr (r4)-

; Pass 4: decreasing addresses ------------------------------------------------
mov r4, r1
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       cmpu (r4), a1
       br stopd, neq
       mov a0l, (r4)             ; write pattern
       cmpu (r4)-, a0
       br stopd, neq
       cmpu (r4), a0
       br stopd, neq
       mov a1l,   (r4)           ; write pattern
       cmpu (r4)-, a1
       br stopd, neq
       nop
%lab3: bkrep r2, >%lab2-1
       cmpu (r4), a0
       br stopd, neq
       mov a1l, (r4)             ; write pattern
       cmpu (r4)-, a1
       br stopd, neq
       cmpu (r4), a1
       br stopd, neq
       mov a0l,   (r4)           ; write pattern
       cmpu (r4)-, a0
       br stopd, neq
       nop
```

```
%lab2: nop
%lab: mov r1, r4


; Pass 5: decreasing addresses ------------------------------------------------
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       cmpu (r4), a0
       br stopd, neq
       mov a1l, (r4)              ; write pattern
       cmpu (r4)-, a1
       br stopd, neq
       cmpu (r4), a1
       br stopd, neq
       mov a0l,   (r4)            ; write pattern
       cmpu (r4)-, a0
       br stopd, neq
       nop
%lab3: bkrep r2, >%lab2-1
       cmpu (r4), a1
       br stopd, neq
       mov a0l, (r4)              ; write pattern
       cmpu (r4)-, a0
       br stopd, neq
       cmpu (r4), a0
       br stopd, neq
       mov a1l,   (r4)            ; write pattern
       cmpu (r4)-, a1
       br stopd, neq
       nop
%lab2: nop
%lab:
test_dmem_rr4k16_end:

pop a1l
pop a0l
pop b0l
pop sv
pop y
pop rb
pop r5
pop r4
pop r3
pop r2
pop r1
pop r0
ret
stopd: brr -1
    nop
```

```
;****************************************************************
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; RCS $Id: bb_rout.asm,v 1.13 1998/09/30 14:23:57 kerryl Exp $
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Filename: rr16k16p_test.asm
;
; Purpose: Memory dependend algorithms library
;
; Context: March C ASIC-driven Memorytest
;
; Author: LSISoftBist RamBist Software Code Compiler
;
;                                Copyright (c), LSI Logic, 1996
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Update history
;
; RCS $Log: bb_rout.asm,v $
; RCS Revision 1.13  1998/09/30 14:23:57  kerryl
; RCS Adding functionality to boot sequence
; RCS
; RCS Revision 1.12  1998/07/30 15:16:08  kerryl
; RCS update to softbist for new memories
; RCS
; RCS Revision 1.11  1998/07/20 10:33:12  kerryl
; RCS change to boot up routine to satisfy GLS
; RCS
; RCS Revision 1.10  1998/06/23 16:32:51  kerryl
; RCS br change
; RCS
; RCS Revision 1.9  1998/06/11 08:55:41  kerryl
; RCS modifying the boot version value and checksums
; RCS
; RCS Revision 1.8  1998/06/03 16:05:27  kerryl
; RCS modifying the boot version value and checksums
; RCS
; RCS Revision 1.5  1998/05/12 16:41:51  kerryl
; RCS change in script
; RCS
; RCS Revision 1.4  1997/09/26 14:59:09  wiesner
; RCS added checkboard background option
; RCS
; RCS Revision 1.3  1997/09/19 12:53:32  wiesner
; RCS alpha version for marco
; RCS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; March C testroutine
; prototype: void test_pmem_rr16k16(void *sa, unsigned int sz, error_code* err)
;
; formal parameters ---------------------------------------------------------
;
```

```
; return <== a0l (void dummy)
;    &sa ==> a0l
;     sz ==> a1l
;   &err ==> b0l
;
; local used registers ------------------------------------------------
;
; r0 = pointer tp testpattern
; r1 = latch addresspointer
; r2 = adr to errorstruct
; r3 = temp location
; r4 = pointer to address
; y = size register
; r5 = free pointer
; sv = shift value


test_pmem_rr16k16:

push r0
push r1
push r2
push r3
push r4
push r5
push rb
push y
push sv
push b0l
push a0l
push a1l
; initialize addresspointer
mov a0l, r4

; calculate shifter value and/or (pass)loopcounter
moda shr4, a1
moda shr, a1
;moda shr, a1
moda dec,a1
mov a1l,y
mov ##0x07, r2
mov ##0x0000, r3
; store addresspointer
mov r4, r5

; Pass 1: increasing addresses ----------------------------------------------
mov ##0x0001,r0
mov ##0x0002, r1
mov ##0xff00, a1l
mov a1l,(r0)
mov ##0x00ff, a0l
mov a0l, (r1)
bkrep y,>%lab-1
```

```
    bkrep r2, >%lab3-1
       movd (r0), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1),  (r4)          ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a0
       br stopp, neq
       nop
%lab3: bkrep r2, >%lab2-1
       movd (r1), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0),   (r4)         ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a1
       br stopp, neq
       nop
%lab2: nop
%lab:mov r5, r4                  ; restore addresspointer

; Pass 2: increasing addresses ------------------------------------------------
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a0
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)+,(r3)
       cmpu (r3), a1
       br stopp, neq
       nop
%lab3: bkrep r2, >%lab2-1
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a1
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1), (r4)           ; write pattern
```

```
       movp (r4)+, (r3)
       cmpu (r3), a0
       br stopp, neq
       nop

%lab2: nop
%lab: mov r5, r4

; Pass 3: increasing addresses ------------------------------------------------
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a1
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a0
       br stopp, neq
       nop
%lab3: bkrep r2, >%lab2-1
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1), (r4)           ; write pattern
       movp (r4)+, (r3)
       cmpu (r3), a0
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)+,(r3)
       cmpu (r3), a1
       br stopp, neq
       nop

%lab2: nop
%lab: modr (r4)-

; Pass 4: decreasing addresses ------------------------------------------------
mov r4, r5
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
```

```
       movd (r1), (r4)           ; write pattern
       movp (r4)-, (r3)
       cmpu (r3), a0
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)-,(r3)
       cmpu (r3), a1
       br stopp, neq
       nop
%lab3: bkrep r2, >%lab2-1
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)-, (r3)
       cmpu (r3), a1
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1), (r4)           ; write pattern
       movp (r4)-, (r3)
       cmpu (r3), a0
       br stopp, neq
       nop

%lab2: nop
%lab: mov r5, r4

; Pass 5: decreasing addresses ------------------------------------------------
bkrep y,>%lab-1
    bkrep r2, >%lab3-1
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)-, (r3)
       cmpu (r3), a1
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a1
       br stopp, neq
       movd (r1), (r4)           ; write pattern
       movp (r4)-, (r3)
       cmpu (r3), a0
       br stopp, neq
       nop
%lab3: bkrep r2, >%lab2-1
       movp (r4), (r3)
       cmpu (r3), a1
```

```
       br stopp, neq
       movd (r1), (r4)           ; write pattern
       movp (r4)-, (r3)
       cmpu (r3), a0
       br stopp, neq
       movp (r4), (r3)
       cmpu (r3), a0
       br stopp, neq
       movd (r0), (r4)           ; write pattern
       movp (r4)-,(r3)
       cmpu (r3), a1
       br stopp, neq
       nop

%lab2: nop
%lab:
test_pmem_rr16k16_end:
pop a1l
pop a0l
pop b0l
pop sv
pop y
pop rb
pop r5
pop r4
pop r3
pop r2
pop r1
pop r0
ret
stopp: brr -1
    nop
;*********************************************************************
;************************** FIR **********************************
;*********************************************************************

; INreg  is register r1
; OUTreg is register r2
; DLYptr is register r0

;-------------------------------------
Fir:   mov     ##0x0800,st1    ; Init SP, page

                               ; Shift P register bits (SP) - 1 left
                               ;  (suitable to the presentation where the
                               ;  binary point is placed after the 1st bit).
                               ; Also initial page into page 0
                               ; DLY[i] = 0

    ; Copy test sample data into X data memory
    mov ##samples, r4
    mov ##INaddr,r2
    rep #Nout-1
```

```
      movp (r4)+,(r2)+

    ; copy co-efficients into Y data memory
    mov ##coeffs, r4
    mov ##COEFaddr,r3
    rep #Ntaps-1
      movp (r4)+,(r3)+

    ; Clear samples buffer
        mov     ##DLYaddr,r0
        clr     a0
        rep     #Ntaps-1
          mov     a0l,(r0)+


;******************** Program Code *************************
; Initializations for TEST_FIR_REAL routine

Init_TEST_FIR_REAL:

        mov     ##INaddr,r3        ; r3 points to INaddr
        mov     ##OUTaddr,r5       ; r5 points TO OUTaddr

; ----------------------------------------


BenchmarkInit:

        mov     ##DLYaddr+Ntaps-1,r0  ; r0 points to DLY (end of buffer)
        mov     r0,DLYsaveAddr        ; Store DLY pointer
        mov     ##COEFaddr,r4         ; r4 points to COEF

        mov     #0x11,a0l               ; Set Modulo operations -
        or      st2,a0                  ;  M0 M4 bit at st2
        mov     a0l,st2                 ; for r0, r4

        mov     ##(Ntaps-1)*128,cfgi    ; Define MODI , no step
        mov     ##(Ntaps-1)*128,cfgj    ; Define MODJ , no step


; ----------------------------------------


TEST_FIR_REAL:

; Run FIR_REAL benchmark Nout times :

        bkrep #Nout-1, >%Testend
        mov     (r3)+,r1          ; INreg = IN[n]
        mov     DLYsaveAddr,r0    ; Restore DLY pointer
```

```
;************ FIR_REAL Sample by sample processing ***************
FIR_REAL:
        modr   (r0)+              ;r0 points to the oldest sample
        mov     r1,(r0)           ; Input new sample
                                  ; Ntaps
        moda    clrr,a0           ; a0 = 0x8000
        mpy     (r4)+,(r0)-       ; p  = h(0) * x(i)
        rep     #Ntaps-2
        mac     (r4)+,(r0)-,a0    ; a0 = a0 + p
                                  ; p  = h(k) * x(i-k)
        add     p,a0              ;              Ntaps-1
        mov     a0h, r2           ; OUTreg=y(i)= Sum( h(k) x(i-k) )
                                             k=0
BenchmarkTerminate:
        mov     r0,DLYsaveAddr    ; Store DLY pointer
%Testend:
        mov     r2,(r5)+           ; OUTreg = OUT[n]

    ;
    ; Check results were as expected
    ;

    ; Set up pointer to copy of filter co-efficients in Y memory
    mov ##OUTaddr,r0

    ; Set up pointer to actual results in X memory
    mov ##COEFaddr,r3

    ; Disable modulo addressing used by FIR filter
        rst     ##0x003f,st2

    ; Loop to check samples = 1/2 original co-efficients
    bkrep #Ntaps-1, %chkloop
           mov (r3)+,a0           ; Copy next co-efficient value into a0
           shr a0                 ; Divide by two using a shift
           cmp (r0)+,a0           ; Compare to output data
           br error,neq           ; Branch to error if comparison failed
           mov pc, cfgj
           nop
%chkloop:nop
endit:ret
;*****************************************************************
;********************** ROM CHECK ********************************
;*****************************************************************
Crom: moda clr, a0                ;clears accumulators performing the checksum
    modb clr, b0
    mov ##0x03fb, r1              ;length of boot rom - top 4 bytes
    mov ##0xffff, a1              ;top of boot rom space
    bkrep r1,>%lab4               ;repeat for each location in ROM - top 4
    movp (a1l), r0                ;mov the value in ROM to r0
    xor r0, a0                    ;perform parity operation
    swap(a0, b0)
    addl r0, a0                   ;perform checksum operation
```

```
    swap(a0, b0)
%lab4:dec a1                    ;next memory location
    movp (a1l), r0              ;mov parity value into r0
    cmpu r0, a0                 ;compare with generated value
    br error ,neq               ;if they are not equal quit
    mov b0l, a0l                ;swap accumulators
    dec a1                      ;decrement memory location
    movp (a1l),r0               ;mov checksum value to r0
    cmpu r0, a0                 ;compare with generated value
    br error, neq               ;if they are not equal quit
    mov pc, cfgj
    ret                         ;successful. ROM is OK
;*********************************************************************
;*********************** CRITICAL PATH ***************************
;*********************************************************************
Crit: rst ##0xffff, st1
    mov ##0xffff, a0
    mov ##0x0001, a1
    add a1l, a0                 ;long carry chain in add/sub
    cmpv ##0x0000, a0l          ;success check
    br error, neq
    sub a1l, a0                 ;long carry chain in add/sub different values
    cmpv ##0xffff, a0l          ;success check
    br error, neq
    mov pc, cfgj
    lpg #0x00
    mov ##0xffff, r0
    mov ##0x0000, r1
    mov ##0x0001, r2
    mov r0, (r1)+
    mov r2, (r1)-
    mov r1, (r0)-
    mov r0, r4
    mov r2, (r0)+
    mac (r4), (r1), a1          ;a long path through the
        mac (r4), (r1), a1      ;multiplier
    mac (r1), ##0xffff, a0      ;a long path through the
    mac (r1), ##0xffff, a0      ;multiplier
    cmpv ##0xffff, a0l
    br error, neq
    cmpv ##0x0000, a1l
    br error, neq
    addv ##0x0001, (r1)         ;long carry chain in add/sub using memory
    subv ##0xffff, (r0)         ;paths
    mov (r1)+, a0l
    mov (r0)-, r3
    sub r3, a0
    cmpv ##0xffff, a0l          ;success check
    br error, neq
    mov pc, cfgj

    mov ##0x5555, r4
    mov ##0x0000, sp
```

```
mov ##0xffff, rb
push r4                         ;pushing and popping on ffff/0000 memory
pop a0l                         ;boundary. completely invert address bits
cmp r4, a0                      ;in consecutive instructions
br error, neq
mov pc, cfgj
mov a0l, (rb+#0x01)             ;reading and writing memory locations
mov (rb+#0x01), a0              ;around the ffff/0000 memory boundary
cmp r4, a0                      ;using the indexed addressing logic
br error, neq
mov pc, cfgj

mov ##0xfff4, sv                ;sets up for right shift
movs 0x00, b0                   ;takes value from mem through barrel shifter
                                       ;into accumulator
divs 0x01, a0                   ;another long path
rep #3                          ;repeat next instruction 4 times
maxd a1, (r0)+, gt              ;compare across ffff/0000 boundary
add a0l, a1
set ##0x0080, st2
shr a1
add b0l, a1
mov a1l, (r0)
mov mixp, a0
chng ##0xffff, (r0)
cmp (r0), a0
br error, neq
mov pc, cfgj

moda clr, a0
moda clr, a1
mov ##0xffff, r4
mov ##0x0000, r0
mov ##0x1111, r2
mov ##0x0005, r1
mov ##0x0002, r3
mov ##0x5555, r5
mov r1, (r0)+
mov r3, (r0)-
mov r2, (r4)-
mov r5, (r4)+
mac (r4)-, (r0)+, a0
mac (r4)+, (r0)-, a1
mac (r4), (r0)+, a0
mac (r4), (r0)+, a1
sub a1l, a0
msu (r4), (r0), a1
mac (r4), (r0), a0
cmpv ##0xeef0, a0l
br error, neq
cmpv ##0x8888, a1l
br error, neq
mov ##0x3fff, sp
ret
```

```
;*****************************************************************
;***************** UTILITIES *************************************
;*****************************************************************


risr: reti         ;redundant interrupt service routine
;*****************************************************************
cisrr:addv ##0x0001, (r0) ; redundant interrupt service routine
    reti    ; which increments a indirect memory location
;*****************************************************************
cisrm:push a0l                    ; redundant interrupt service routine
    push a0h                      ; which increments the contents of a
    mov [##0xffff], a0            ;specific memory location
    moda inc, a0
    mov a0l, [##0xffff]
    pop a0h
    pop a0l
    reti
;*****************************************************************
dnfl:mov ##0xffff, lc             ;Do nothing fast loop
    rep lc
    nop
    brr -3
    ret
;*****************************************************************
scrs0:push a0l                    ;Stop clock routine using s0
    push a0h
    push rb
    push r0
    push r2
    push r3
    call setmode
    mov (rb+#0x04), a0
    set ##0x0080, a0l
    mov a0l, (rb+#0x04)
    nop
    nop
    pop r3
    pop r2
    pop r0
    pop rb
    pop a0h
    pop a0l
    ret
;*****************************************************************
scrs1:push a0l                    ;stop clock routine using s1
    push a0h
    push rb
    push r0
    push r2
    push r3
    call setmode
```

```
    mov (rb+#0x04), a0
    set ##0x0100, a0l
    mov a0l, (rb+#0x04)
    nop
    nop
    pop r3
    pop r2
    pop r0
    pop rb
    pop a0h
    pop a0l
    ret
;*********************************************************************
rver:push r5                        ;Return version number of Chip/ROM code
    push a0l
    push a0h
    mov ##0xfc04, a0
    movp (a0l), r5
    pop a0h
    pop a0l
    ret
;*********************************************************************
ebws:push r5
    push r1
    push r2
    push r4
    call setmode
    mov ##0x0000, r5
    mov ##0xffff, a0
    mov ##0x0001, r1
    mov rb, a1
    add ##0x0009, a1
    mov a1l, r4
    mov r5, (r4)-
    mov a0l, (r4)+
    mov r1, (r4)+
    mov (r4), b0
    sub b0l, a0
    pop  r4
    pop r2
    pop r1
    pop r5
    ret
;*********************************************************************
```

8 Specifications

This section presents the electrical and mechanical specifications for the CWDSP1670 Lead Vehicle.

8.1 Electrical Characteristics

Tables 5 through 8 describe the electrical specifications for the lead vehicle.

Table 5 Absolute Maximum Ratings

Symbol	Parameter	Limits	Units
V_{DD4}	DC supply voltage for I/O buffers	-0.3 to +3.9	V
V_{DD2}	DC supply voltage for core and chip logic	-0.3 to +2.75	V
V_{IN}	Input voltage	-1.0 to V_{DD4} +0.3	V
I_{IN}	DC input current	-10 to +10	mA
T_{STGP}	Storage temperature, plastic	-40 to +125	°C

Table 6 Recommended Operating Conditions for 60 MHz Maximum Operating Frequency

Symbol	Parameter	Limits	Units
V_{DD4}	DC supply voltage	+3.00 to 3.60	V
V_{DD2}	DC supply voltage	+2.25 to +2.75	V
T_A	Ambient temperature, commercial	-20 to +85	°C

Table 7 Recommended Operating Conditions for 80 MHz Maximum Operating Frequency[1]

Symbol	Parameter	Limits	Units
V_{DD4}	DC supply voltage	+3.15 to 3.45	V
V_{DD2}	DC supply voltage	+2.50 to +2.75	V
T_A	Ambient temperature, commercial	-20 to +40	°C

1. EDB_WAIT_EN must be set to Logic 1

Table 8 Capacitance

Symbol	Parameter[1]	Min	Max	Units
C_{IN}	Input Capacitance	–	5	pF
C_{OUT}	Output Capacitance	–	10	pF
C_{IO}	I/O Bus Capacitance	–	15	pF

1. Measurement conditions are V_{IN} = V, T_A = 25 °C, and clock frequency = 80 MHz.

Table 9 DC Characteristics

Symbol	Parameter	Condition[1]	Min	Typ	Max	Units
V_{IH}	High level input voltage	–	2.0	–	–	V
	CLKIN high input level	–	2.0	–	–	
V_{IL}	Low level input voltage	–	–	–	0.8	V
	CLKIN low input level	–	–	–	0.8	
V_{OH}	High level output voltage	I_{OH} = -4.0 mA	2.4	4.5	–	V
V_{OL}	Low level output voltage	I_{OL} = 4.0 mA	–	0.2	0.4	V

Table 9 DC Characteristics (Cont.)

Symbol	Parameter	Condition[1]	Min	Typ	Max	Units
I_{IL}	Input leakage current (output and I/O pins in hi-z or input state).	See Note[2]	-10	-1 to +1	10	uA
I_{CC}	Supply current, dynamic	V_{DD4} = Max f = 50 MHz	–	150	–	mA

1. Measurement conditions are Ta = 25 °C, Vcc = 3.3 V, GND = 0 V, 100 pF load on CLKOUT.
2. V_{DD4} = Max, V_{IN} = V_{DD4} or V_{SS}, V_{OUT} = V_{DD4} or V_{SS}

8.2 AC Timing

All of the timing in this section is referenced to the internal OAK_CLK/CLK_UG. Figure 22 shows how these clocks are related.

Symbol	Description	Typ (ns)
$T_{d\_oc\_int}$	Delay of internal OAK_CLK (WCMIL)	5.3
$T_{d\_oc\_ext}$	Delay of external OAK_CLK (WCMIL)	3.0

Figure 22 Internal Clocks

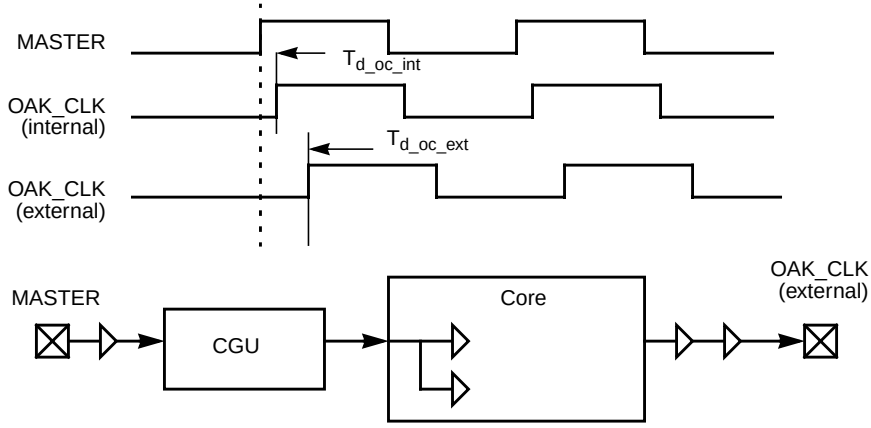


Figure 23, Figure 24, and Table 10 specify the read and write timing of D Bus accesses. Figure 25, Figure 26, and Table 11 specify the read and write timing of D Bus I/O space accesses. The timing was measured at 125˚C (junction temperature) using the WCMIL process and 15 pF loading on all outputs and is valid from WCMIL to BCMIL.

Figure 23 D Bus Read Timing (1 wait state)

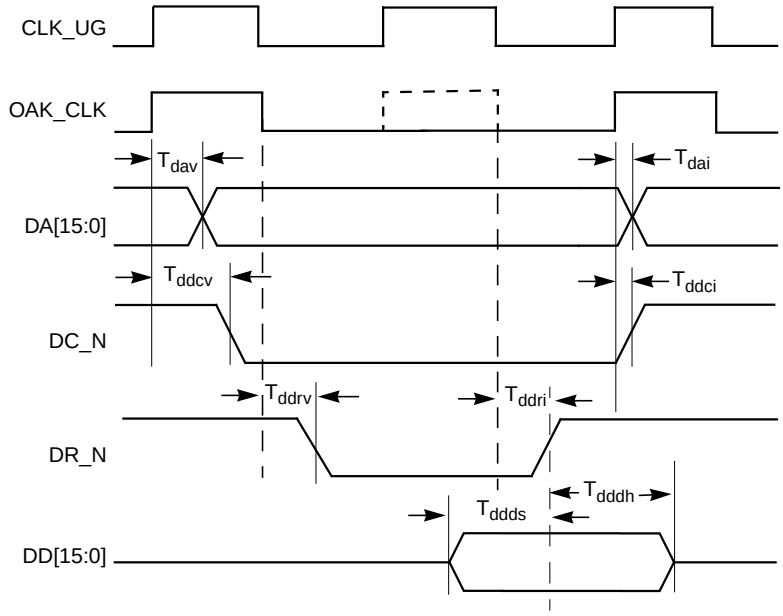


Figure 24 D Bus Write Timing (1 wait state)

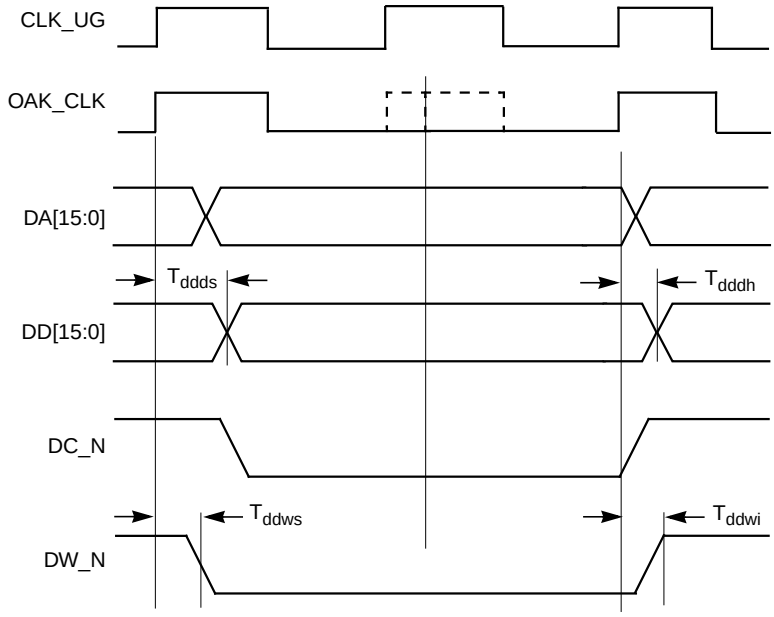


Table 10 D Bus Read/Write Timing

Item	Description	Minimum (ns)	Maximum (ns)
T_{dav}	D Bus Address valid	-	8
T_{dai}	D Bus Address invalid: clock falling edge to address invalid	1	–
T_{ddrv}	D Bus Read strobe valid	–	7
T_{ddri}	D Bus Read strobe invalid	1	–
T_{ddcv}	D Bus Chip select valid	–	8
T_{ddcv}	D Bus Chip select invalid: clock falling edge to chip select invalid	1	–
T_{ddws}	D Bus Write strobe valid	–	7
T_{ddwi}	D Bus Write strobe invalid	1	–
T_{dddv}	D Bus Write data valid	–	9
T_{dddi}	D Bus Write data invalid	1	–

Table 10 D Bus Read/Write Timing (Cont.)

Item	Description	Minimum (ns)	Maximum (ns)
T_{ddds}	D Bus Read data setup: read data valid to read strobe rising edge	6	–
T_{dddh}	D Bus Read data hold: read strobe rising edge to read data invalid	0	–

Figure 25 I/O Space Read Timing (2 wait states)

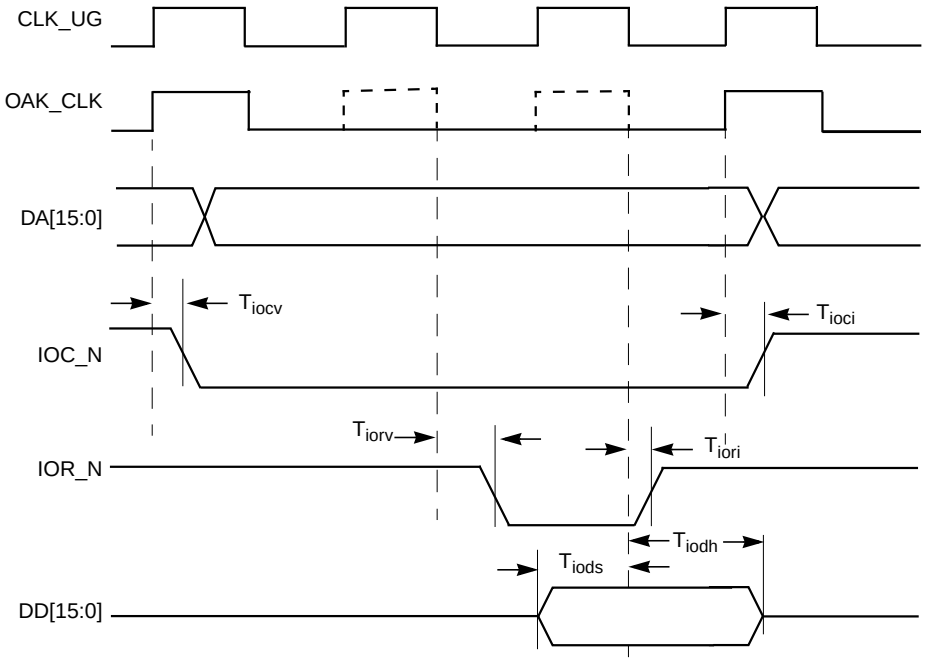


Figure 26 I/O Space Write Timing (2 wait states)

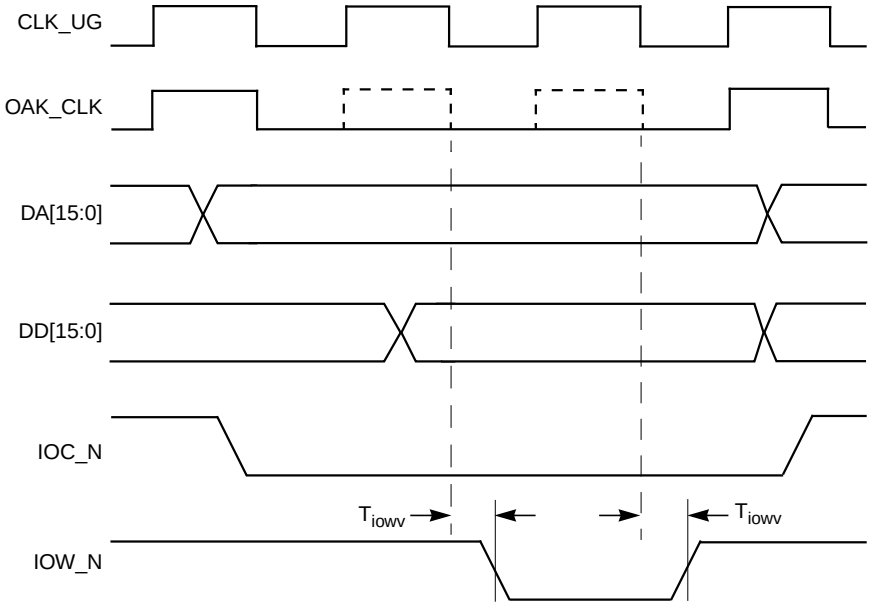


Table 11 I/O Space Read/Write Timing

Item	Description	Minimum (ns)	Maximum (ns)
T_{iocv}	IO chip select valid	–	8
T_{ioci}	IO chip select hold: clock falling edge to chip select invalid	1	–
T_{iorv}	IO read strobe valid	–	7
T_{iori}	IO read strobe hold: clock falling edge to write strobe invalid	1	–
T_{iowv}	IO write strobe valid	–	7
T_{iowi}	IO write strobe hold: clock falling edge to write strobe invalid	1	–
T_{iods}	IO Read data setup: read data valid to read strobe falling edge	6	–
T_{iodh}	IO Read data hold: read strobe falling edge to read data invalid	0	–

8.3 Package Pinout and Mechanical Drawing

Table 12 lists the lead vehicle's I/O signals in alphabetical order, shows their pin numbers and direction, and provides a brief description. Table 13 includes the same information but is sorted by solder ball designation. Figure 27 is the mechanical drawing for the lead vehicle. Figure 28 shows the pin names looking into the solder balls.

Table 12 Alphabetical Signal Listing

Signal Name	Bond Pad	Solder Ball	I/O	Description
ABORT_N	55	P3	in	Abort input
ABORT_OUT	81	P9	out	Abort acknowledge
BOOT_N	54	R2	in	Boot control (Boot from ROM when LOW)
BRRM1	86	N10	out	HIGH = BRR $-1 executed
CLK_UG	59	P4	out	Copy of CWDSP1670 core clock without wait states
CLKOUT_EN	196	D4	in	HIGH = Enable OAK_CLK, CLK_UG, and TIMER_CLK outputs
D_DMA_GRANT	49	P1	out	HIGH = DMA has been granted D Bus. DA, DD, DW_N, DR_N, DC_N, IOW_N, IOR_N, and IOC_N are tristated.
D_DMA_N	47	M3	in	LOW = External DMA controller becomes D Bus master
DA0	1	B2	out/Z	D Bus address lines
DA1	2	C3	out/Z	
DA10	14	E1	out/Z	
DA11	13	F5	out/Z	
DA12	15	F4	out/Z	
DA13	16	F3	out/Z	
DA14	17	F2	out/Z	
DA15	18	F1	out/Z	

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
DA2	3	B1	out/Z	D Bus address lines (cont.)
DA3	4	C2	out/Z	
DA4	6	C1	out/Z	
DA5	5	D3	out/Z	
DA6	7	D2	out/Z	
DA7	11	E5	out/Z	
DA8	8	E4	out/Z	
DA9	12	E2	out/Z	
DC_N	44	L4	out/Z	D Bus chip select
DD0	21	G5	inout/Z	D Bus data lines
DD1	23	G2	inout/Z	
DD10	32	J2	inout/Z	
DD11	37	K4	inout/Z	
DD12	35	K3	inout/Z	
DD13	36	K2	inout/Z	
DD14	34	K1	inout/Z	
DD15	38	L1	inout/Z	
DD2	22	G1	inout/Z	
DD3	24	H5	inout/Z	
DD4	25	H4	inout/Z	
DD5	27	H3	inout/Z	
DD6	28	H2	inout/Z	
DD7	26	H1	inout/Z	
DD8	33	J5	inout/Z	
DD9	31	J3	inout/Z	

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
DMMC_MODE	185	B6	in	HIGH = DMMC mode, LOW = Normal mode
DR_N	42	M1	out/Z	D Bus read enable
DW_N	41	L2	out/Z	D Bus write enable
EDB_WAIT_EN	153	B13	in	HIGH = Enable the automatic, single wait state generation for E Bus accesses.
EMEM_WAIT	60	L6	in	HIGH = 1 wait state for E memory (M6 cleared), LOW = zero wait states for E memory
ESCAN_EN	156	C12	in	Reserved
ESCAN_IN	155	D12	in	Reserved
ESCAN_OUT	199	A2	out	Reserved
EXT_SCAN_ALERT	171	B9	out	ScanICE alert
EXT_SCAN_CLK	165	A11	in	ScanICE clock
EXT_SCAN_CTL	164	B11	in	ScanICE load control
EXT_SCAN_IN	163	D10	in	ScanICE data in
EXT_SCAN_OUT	169	E9	out	ScanICE data out
EXT_SCAN_RST	166	C10	in	ScanICE reset
IACK_INT0_N	125	H13	out	INT0 acknowledge
IACK_INT1_N	126	H15	out	INT1 acknowledge
IACK_INT2_N	127	H12	out	INT2 acknowledge
IACK_INT3_N	128	H11	out	INT3 acknowledge
IACK_NMI_N	129	G14	out	NMI acknowledge
IIDDTN	195	A3	–	Production test
IM3	189	E6	in	INT3 mask control
INT0_N	191	C5	in	Maskable interrupt #0
INT1_N	192	A4	in	Maskable interrupt #1
INT2_N	193	D5	in	Maskable interrupt t#2

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
INT3_CNTX_EN	190	B5	in	HIGH = INT3 automatic context switch enable
INT3_N	194	B4	in	Maskable interrupt #3
INT3_VEC0	132	G13	in	INT3 vector address input
INT3_VEC1	133	G12	in	
INT3_VEC10	144	E12	in	
INT3_VEC11	145	D14	in	
INT3_VEC12	146	C15	in	
INT3_VEC13	147	D13	in	
INT3_VEC14	148	C14	in	
INT3_VEC15	149	B15	in	
INT3_VEC2	134	F15	in	
INT3_VEC3	135	F14	in	
INT3_VEC4	136	F13	in	
INT3_VEC5	137	F12	in	
INT3_VEC6	138	E15	in	
INT3_VEC7	139	F11	in	
INT3_VEC8	142	D15	in	
INT3_VEC9	143	E13	in	
IOC_N	43	L3	out/Z	I/O chip select
IOR_N	45	N1	out/Z	I/O read enable
IOW_N	46	M2	out/Z	I/O write enable
IU0	57	M5	in	User input pins
IU1	58	R3	in	
LEVEL_INT_MODE	158	D11	in	HIGH = Level-triggered interrupt mode, LOW = Edge-triggered interrupt mode

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
MCLK	184	A6	in	Master input clock
MP_ABORT	53	P2	in	Master processor abort control
MP_BOOT	152	C13	in	Master processor boot control
MP_RESET	154	A14	in	Master processor reset control
NMI_N	157	A13	in	Nonmaskable interrupt
Not Connection	84	R10	–	–
Not Connected	155	D12	–	–
Not Connected	51 100 151 156 199 200	R1 N13 A15 C12 A2 A1	–	–
OAK_CLK	85	P10	out	Copy of CWDSP1670 core clock
OCEM_SUSP	186	C6	in	HIGH = Turn off all clocks to OCEM
OU0	88	P11	out	User output pins
OU1	87	R11	out	
OUTP0	174	C8	out	Output port
OUTP1	175	B8	out	
OUTP2	176	A8	out	
OUTP3	177	D8	out	
OUTP4	178	E8	out	
OUTP5	179	E7	out	
OUTP6	180	A7	out	
OUTP7	181	B7	out	
PPROTECT_EN	161	A12	in	HIGH enables P Bus memory protection.
PROCOUT	198	B3	–	Process monitor pin

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
PTST	99	P14	in	Production test enable
PTST_TE	98	R14	in	Production test scan chain load enable
RST_N	162	C11	in	Reset input
RST_OUT_N	83	L9	out	Synchronized to OAK_CLK version of RST_N
SCAN_DEBUG_EN	172	A9	in	HIGH = Enable ScanICE debug mode
SCANICE_MODE	170	C9	out	HIGH = ScanICE mode is active
SEL_XR_RD0	123	J12	out	Multiplexer control signals for read of external registers
SEL_XR_RD1	124	H14	out	
SELF_TEST_N	56	N4	in	Self-test boot enable
STOP_MODE	82	N9	out	Indicates CWDSP1670 core is in Stop mode
TIMER_CLK	84	R10	out	Timer input clock, gated copy of MCLK
TN	52	N3	–	Production test
VDD2 – Oak Core	167 187	B10 D6	in	+2.5 V core power
VDD2 – Ram/LV Logic	–	F6 F7 F9 F10 G6 G10 J6 J10 K6 K7 K9 K10	in	+2.5 V RAM/LV logic power

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
VDD4	9 19 29 39 69 79 89 97 111 121 130 140 159 182	E3 G4 J4 K5 R6 M9 M10 P13 L14 J13 G15 E14 B12 C7	in	+3.3 V I/O buffer power
VSS	10 20 30 40 50 70 80 90 112 122 131 141 150 160 168 173 183 188 197	D1 G3 J1 L5 N2 L7 R9 N11 L11 J15 G11 E11 B14 E10 A10 D9 D7 A5 C4 F8 G7 G8 G9 H6 H7 H8 H9 H10 J7 J8 J9 K8	–	Ground

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
WAIT_CTL_N	48	M4	in	External wait control
XRDI0	61	R4	in	External register data in
XRDI1	62	M6	in	
XRDI10	73	R7	in	
XRDI11	74	L8	in	
XRDI12	75	M8	in	
XRDI13	76	R8	in	
XRDI14	78	N8	in	External register data in
XRDI15	77	P8	in	
XRDI2	63	N5	in	
XRDI3	64	P5	in	
XRDI4	65	R5	in	
XRDI5	66	N6	in	
XRDI6	67	M7	in	
XRDI7	68	P6	in	
XRDI8	71	N7	in	
XRDI9	72	P7	in	
XRDO0	91	R12	out	
XRDO1	92	L10	out	
XRDO10	105	M13	out	
XRDO11	106	M14	out	
XRDO12	107	N15	out	
XRDO13	108	L12	out	
XRDO14	109	L13	out	
XRDO15	110	M15	out	

Table 12 Alphabetical Signal Listing (Cont.)

Signal Name	Bond Pad	Solder Ball	I/O	Description
XRDO2	93	P12	out	External register data out (cont.)
XRDO3	94	M11	out	
XRDO4	95	R13	out	
XRDO5	96	N12	out	
XRDO6	101	R15	out	
XRDO7	102	N14	out	
XRDO8	103	P15	out	
XRDO9	104	M12	out	External register data out
XRR0_N	113	K11	out	Read enable for ext. reg. #0
XRR1_N	114	L15	out	Read enable for ext. reg. #1
XRR2_N	115	K12	out	Read enable for ext. reg. #2
XRR3_N	116	K14	out	Read enable for ext. reg. #3
XRW0_N	117	K13	out	Write enable for ext. reg. #0
XRW1_N	118	K15	out	Write enable for ext. reg. #1
XRW2_N	119	J11	out	Write enable for ext. reg. #2
XRW3_N	120	J14	out	Write enable for ext. reg. #3

Table 13 Signal Listing by Ball Number

Solder Ball	Bond Pad	Signal Name	I/O	Description
A1	200	Not Connected	–	–
A2	199	ESCAN_OUT	out	Reserved, leave unconnected
A3	195	IIDDTN	–	Production test
A4	192	INT1_N	in	Maskable interrupt #1
A5	188	VSS	–	Ground
A6	184	MCLK	in	Master input clock
A7	180	OUTP6	out	User output 6
A8	176	OUTP2	out	User output 2
A9	172	SCAN_DEBUG_EN	in	Enable ScanICE debug mode
A10	168	VSS	–	Ground
A11	165	EXT_SCAN_CLK	in	ScanICE clock
A12	161	PPROTECT_EN	in	HIGH enables P Bus memory protection.
A13	157	NMI_N	in	Nonmaskable interrupt
A14	154	MP_RESET	in	Master processor reset control
A15	151	Not Connected	–	–
B1	3	DA2	out/Z	D Bus address 2
B2	1	DA0	out/Z	D Bus address 0
B3	198	PROCOUT	–	Process monitor pin
B4	194	INT3_N	in	Maskable interrupt #3
B5	190	INT3_CNTX_EN	in	INT3 automatic context switch enable
B6	185	DMMC_MODE	in	HIGH = DMMC mode, LOW = Normal mode
B7	181	OUTP7	out	User output 7
B8	175	OUTP1	out	User output 1
B9	171	EXT_SCAN_ALERT	out	ScanICE alert

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
B10	167	VDD2	in	+2.5 V core power
B11	164	EXT_SCAN_CTL	in	ScanICE load control
B12	159	VDD4	in	+3.3 V I/O buffer power
B13	153	EDB_WAIT_EN	in	HIGH = Enable the automatic, single wait state generation for E Bus accesses
B14	150	VSS	–	Ground
B15	149	INT3_VEC15	in	INT3 interrupt vector address 15
C1	6	DA4	out/Z	D Bus address 4
C2	4	DA3	out/Z	D Bus address 3
C3	2	DA1	out/Z	D Bus address 1
C4	197	VSS	–	Ground
C5	191	INT0_N	in	Maskable interrupt #0
C6	186	OCEM_SUSP	in	Suspend OCEM
C7	182	VDD4	in	+3.3 V I/O buffer power
C8	174	OUTP0	out	User output 0
C9	170	SCANICE_MODE	out	Indication that ScanICE mode is active
C10	166	EXT_SCAN_RST	in	ScanICE reset
C11	162	RST_N	in	Reset input
C12	156	ESCAN_EN	in	Reserved, tie to Vss
C13	152	MP_BOOT	in	Master processor boot control
C14	148	INT3_VEC14	in	INT3 interrupt vector address 14
C15	146	INT3_VEC12	in	INT3 interrupt vector address 12
D1	10	VSS	–	Ground
D2	7	DA6	out/Z	D Bus address 6
D3	5	DA5	out/Z	D Bus address 5

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
D4	196	CLKOUT_EN	in	HIGH = Enable OAK_CLK, CLK_UG, and TIMER_CLK outputs
D5	193	INT2_N	in	Maskable interrupt #2
D6	187	VDD2	in	+2.5 V core power
D7	183	VSS	–	Ground
D8	177	OUTP3	out	User output 3
D9	173	VSS	–	Ground
D10	163	EXT_SCAN_IN	in	ScanICE data in
D11	158	LEVEL_INT_MODE	in	HIGH = Level-triggered interrupt mode, LOW = Edge-triggered interrupt mode
D12	155	ESCAN_IN	in	reserved, tie to Vss
D13	147	INT3_VEC13	in	INT3 interrupt vector address 13
D14	145	INT3_VEC11	in	INT3 interrupt vector address 11
D15	142	INT3_VEC8	in	INT3 interrupt vector 8
E1	14	DA10	out/Z	D Bus address 10
E2	12	DA9	out/Z	D Bus address 9
E3	9	VDD4	in	+3.3 V I/O buffer power
E4	8	DA8	out/Z	D Bus address 8
E5	11	DA7	out/Z	D Bus address 7
E6	189	IM3	in	INT3 mask control
E7	179	OUTP5	out	User output 5
E8	178	OUTP4	out	User output 4
E9	169	EXT_SCAN_OUT	out	ScanICE data out
E10	160	VSS	–	Ground
E11	141	VSS	–	Ground
E12	144	INT3_VEC10	in	INT3 interrupt vector address 10

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
E13	143	INT3_VEC9	in	INT3 interrupt vector address 9
E14	140	VDD4	in	+3.3 V I/O buffer power
E15	138	INT3_VEC6	in	INT3 interrupt vector 6
F1	18	DA15	out/Z	D Bus address 15
F2	17	DA14	out/Z	D Bus address 14
F3	16	DA13	out/Z	D Bus address 13
F4	15	DA12	out/Z	D Bus address 12
F5	13	DA11	out/Z	D Bus address 11
F6	–	VDD2	in	+2.5 V RAM/LV logic
F7	–	VDD2	in	+2.5 V RAM/LV logic
F8	–	VSS	–	Ground
F9	–	VDD2	in	+2.5 V RAM/LV logic
F10	–	VDD2	in	+2.5 V RAM/LV logic
F11	139	INT3_VEC7	in	INT3 interrupt vector address 7
F12	137	INT3_VEC5	in	INT3 interrupt vector address 5
F13	136	INT3_VEC4	in	INT3 interrupt vector address 4
F14	135	INT3_VEC3	in	INT3 interrupt vector address 3
F15	134	INT3_VEC2	in	INT3 interrupt vector address 2
G1	22	DD2	inout/Z	D Bus data 2
G2	23	DD1	inout/Z	D Bus data 1
G3	20	VSS	–	Ground
G4	19	VDD4	in	+3.3 V I/O buffer power
G5	21	DD0	inout/Z	D Bus Data bus
G6	–	VDD2	in	+2.5 V RAM/LV logic
G7	–	VSS	–	Ground

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
G8	–	VSS	–	Ground
G9	–	VSS	–	Ground
G10	–	VDD2	in	+2.5 V RAM/LV logic
G11	131	VSS	–	Ground
G12	133	INT3_VEC1	in	INT3 interrupt vector address 1
G13	132	INT3_VEC0	in	INT3 interrupt vector address 0
G14	129	IACK_NMI_N	out	NMI acknowledge
G15	130	VDD4	in	+3.3 V I/O buffer power
H1	26	DD7	inout/Z	D Bus data 7
H2	28	DD6	inout/Z	D Bus data 6
H3	27	DD5	inout/Z	D Bus data 5
H4	25	DD4	inout/Z	D Bus data 4
H5	24	DD3	inout/Z	D Bus data 3
H6	–	VSS	–	Ground
H7	–	VSS	–	Ground
H8	–	VSS	–	Ground
H9	–	VSS	–	Ground
H10	–	VSS	–	Ground
H11	128	IACK_INT3_N	out	INT3 acknowledge
H12	127	IACK_INT2_N	out	INT2 acknowledge
H13	125	IACK_INT0_N	out	INT0 acknowledge
H14	124	SEL_XR_RD1	out	External register read select 1
H15	126	IACK_INT1_N	out	INT1 acknowledge
J1	30	VSS	–	Ground
J2	32	DD10	inout/Z	D Bus data 10

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
J3	31	DD9	inout/Z	D Bus data 9
J4	29	VDD4	in	+3.3 V I/O buffer power
J5	33	DD8	inout/Z	D Bus data 8
J6	–	VDD2	in	+2.5 V RAM/LV logic
J7	–	VSS	–	Ground
J8	–	VSS	–	Ground
J9	–	VSS	–	Ground
J10	–	VDD2	in	+2.5 V RAM/LV logic
J11	119	XRW2_N	out	Write enable for ext. reg. #2
J12	123	SEL_XR_RD0	out	External register read select 0
J13	121	VDD4	in	+3.3 V I/O buffer power
J14	120	XRW3_N	out	Write enable for ext. reg. #3
J15	122	VSS	–	Ground
K1	34	DD14	inout/Z	D Bus data 14
K2	36	DD13	inout/Z	D Bus data 13
K3	35	DD12	inout/Z	D Bus data 12
K4	37	DD11	inout/Z	D Bus data 11
K5	39	VDD4	in	+3.3 V I/O buffer power
K6	–	VDD2	in	+2.5 V RAM/LV logic
K7	–	VDD2	in	+2.5 V RAM/LV logic
K8	–	VSS	–	Ground
K9	–	VDD2	in	+2.5 V RAM/LV logic
K10	–	VDD2	in	+2.5 V RAM/LV logic
K11	113	XRR0_N	out	Read Enable for ext. reg. #0
K12	115	XRR2_N	out	Read Enable for ext. reg. #2

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
K13	117	XRW0_N	out	Write Enable for ext. reg. #0
K14	116	XRR3_N	out	Read Enable for ext. reg. #3
K15	118	XRW1_N	out	Write Enable for ext. reg. #1
L1	38	DD15	inout/Z	D Bus data 15
L2	41	DW_N	out/Z	D Bus write enable
L3	43	IOC_N	out/Z	I/O chip select
L4	44	DC_N	out/Z	D Bus chip select
L5	40	VSS	–	Ground
L6	60	EMEM_WAIT	in	HIGH = 1 wait state for E memory (M4 cleared), LOW = zero wait states for E memory
L7	70	VSS	–	Ground
L8	74	XRDI11	in	External register data input 11
L9	83	RST_OUT_N	out	Synchronized to OAK_CLK version of RST_N
L10	92	XRDO1	out	External register data output 1
L11	112	VSS	–	Ground
L12	108	XRDO13	out	External register data output 13
L13	109	XRDO14	out	External register data output 14
L14	111	VDD4	in	+3.3 V I/O buffer power
L15	114	XRR1_N	out	Read enable for ext. reg. #1
M1	42	DR_N	out/Z	D Bus read enable
M2	46	IOW_N	out/Z	I/O write enable
M3	47	D_DMA_N	in	LOW = External DMA controller becomes D Bus master
M4	48	WAIT_CTL_N	in	External wait control
M5	57	IU0	in	User input 0

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
M6	62	XRDI1	in	External register data input 1
M7	67	XRDI6	in	External register data input 6
M8	75	XRDI12	in	External register data input 12
M9	79	VDD4	in	+3.3 V I/O buffer power
M10	89	VDD4	in	+3.3 V I/O buffer power
M11	94	XRDO3	out	External register data output 3
M12	104	XRDO9	out	External register data output 9
M13	105	XRDO10	out	External register data output 10
M14	106	XRDO11	out	External register data output 11
M15	110	XRDO15	out	External register data output 15
N1	45	IOR_N	out/Z	I/O read enable
N2	50	VSS	–	Ground
N3	52	TN	–	Production test
N4	56	SELF_TEST_N	in	Self-test boot enable
N5	63	XRDI2	in	External register data input 2
N6	66	XRDI5	in	External register data input 5
N7	71	XRDI8	in	External register data input 8
N8	78	XRDI14	in	External register data input 14
N9	82	STOP_MODE	out	HIGH = CWDSP1670 core is in Stop mode
N10	86	BRRM1	out	HIGH = BRR $-1 execution
N11	90	VSS	–	Ground
N12	96	XRDO5	out	External register data output 5
N13	100	Not Connected	–	–
N14	102	XRDO7	out	External register data output 7
N15	107	XRDO12	out	External register data output 12

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
P1	49	D_DMA_GRANT	out	HIGH = DMA has been granted D Bus. DA, DD, DW_N, DR_N, DC_N, IOW_N, IOR_N, and IOC_N are 3-stated.
P2	53	MP_ABORT	in	Master processor abort control
P3	55	ABORT_N	in	Abort input
P4	59	CLK_UG	out	Copy of CWDSP1670 core clock without wait states
P5	64	XRDI3	in	External register data input 3
P6	68	XRDI7	in	External register data input 7
P7	72	XRDI9	in	External register data input 9
P8	77	XRDI15	in	External register data input 15
P9	81	ABORT_OUT	out	Abort acknowledge
P10	85	OAK_CLK	out	Copy of CWDSP1670 core clock
P11	88	OU0	out	User output 0
P12	93	XRDO2	out	External register data output 2
P13	97	VDD4	in	+3.3 V I/O buffer power
P14	99	PTST	in	Production test enable
P15	103	XRDO8	out	External register data output 8
R1	51	Not Connected	–	–
R2	54	BOOT_N	in	Boot control (Boot from ROM when LOW)
R3	58	IU1	in	User input 1
R4	61	XRDI0	in	External register data input 0
R5	65	XRDI4	in	External register data input 4
R6	69	VDD4	in	+3.3 V I/O buffer power
R7	73	XRDI10	in	External register data input 10
R8	76	XRDI13	in	External register data input 13

Table 13 Signal Listing by Ball Number (Cont.)

Solder Ball	Bond Pad	Signal Name	I/O	Description
R9	80	VSS	–	Ground
R10	84	TIMER_CLK	out	Timer input clock, gated copy of MCLK
R11	87	OU1	out	User output 1
R12	91	XRDO0	out	External register data output 0
R13	95	XRDO4	out	External register data output 4
R14	98	PTST_TE	in	Production test scan chain load enable
R15	101	XRDO6	out	External register data output 6

Figure 27 225 PBGA (IB) Mechanical Drawing

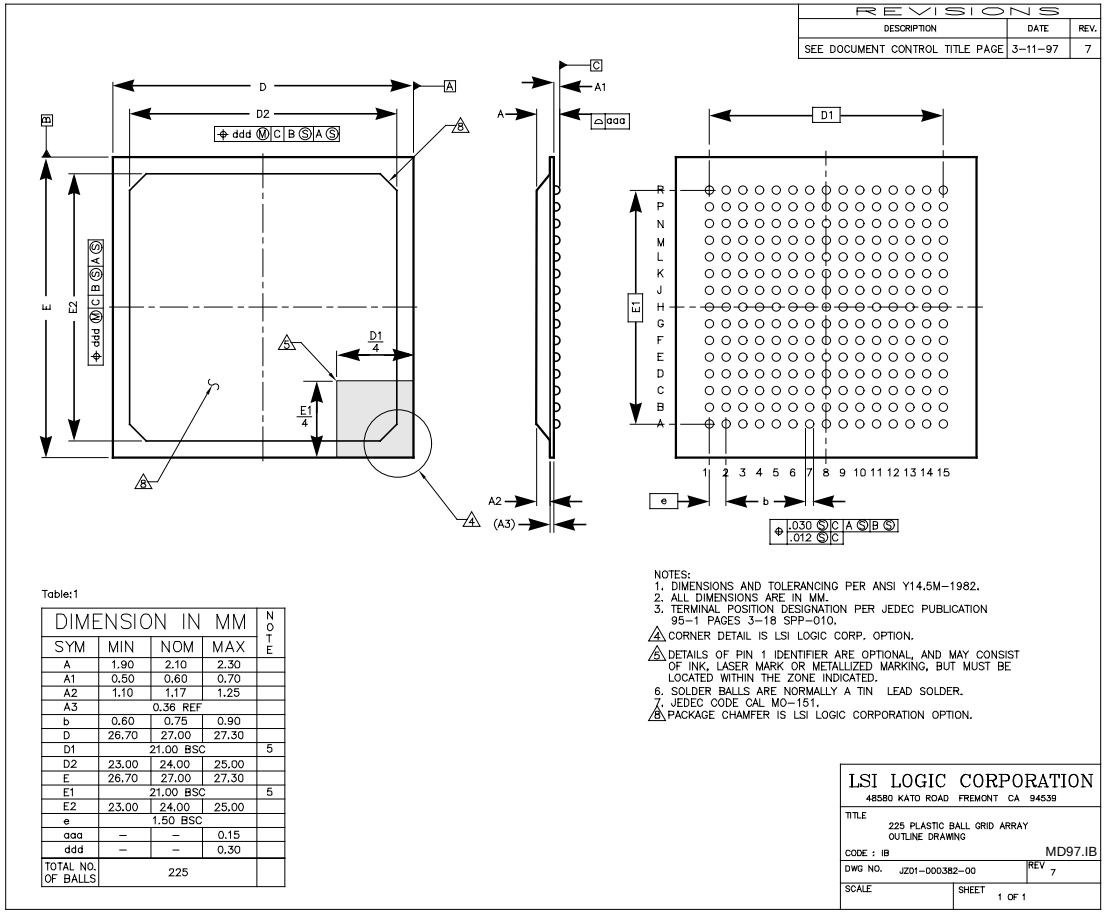


Table:1

DIMENSION IN MM				NOTE
SYM	MIN	NOM	MAX	
A	1.90	2.10	2.30	
A1	0.50	0.60	0.70	
A2	1.10	1.17	1.25	
A3		0.36 REF		
b	0.60	0.75	0.90	
D	26.70	27.00	27.30	
D1		21.00 BSC		5
D2	23.00	24.00	25.00	
E	26.70	27.00	27.30	
E1		21.00 BSC		5
E2	23.00	24.00	25.00	
e		1.50 BSC		
aaa	–	–	0.15	
ddd	–	–	0.30	
TOTAL NO. OF BALLS		225		

Important: **This drawing may not be the latest version. For board layout and manufacturing, obtain the most recent engineering drawings from your LSI Logic marketing representative by requesting the outline drawing for package code IB.**

Figure 28 Pin Assignments Seen From Solder Ball Side

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R
1		DA[2]	DA[4]	vss	DA[10]	DA[15]	DD[2]	DD[7]	vss	DD[14]	DD[15]	DR_N	IOR_N	D_DMA_GRANT	
2	ESCAN_OUT	DA[0]	DA[3]	DA[6]	DA[9]	DA[14]	DD[1]	DD[6]	DD[10]	DD[13]	DW_N	IOW_N	vss	MP_ABORT	BOOT_N
3	IIDDTN	PRO-COUT	DA[1]	DA[5]	vdd4	DA[13]	vss	DD[5]	DD[9]	DD[12]	IOC_N	D_DMA_N	TN	ABORT_N	IU1
4	INT1_N	INT3_N	vss	CLKOUT_EN	DA[8]	DA[12]	vdd4	DD[4]	vdd4	DD[11]	DC_N	WAIT_CTL_N	SELF_TEST_N	CLK_UG	XRDI[0]
5	vss	INT3_CNTX_EN	INT0_N	INT2_N	DA[7]	DA[11]	DD[0]	DD[3]	DD[8]	vdd4	vss	IU0	XRDI[2]	XRDI[3]	XRDI[4]
6	MCLK	DMMC_MODE	OCEM_SUSP	vdd2	IM3						EMEM_WAIT	XRDI[1]	XRDI[5]	XRDI[7]	vdd4
7	OUTP[6]	OUTP[7]	vdd4	vss	OUTP[5]						vss	XRDI[6]	XRDI[8]	XRDI[9]	XRDI[10]
8	OUTP[2]	OUTP[1]	OUTP[0]	OUTP[3]	OUTP[4]						XRDI[11]	XRDI[12]	XRDI[14]	XRDI[15]	XRDI[13]
9	SCAN_DEBUG_EN	EXT_SCAN_ALERT	SCANICE_MODE	vss	EXT_SCAN_OUT						RST_OUT_N	vdd4	STOP_MODE	ABORT_OUT	vss
10	vss	vdd2	EXT_SCAN_RST	EXT_SCAN_IN	vss						XRDO[1]	vdd4	BRRM1	OAK_CLK	TIMER_CLK
11	EXT_SCAN_CLK	EXT_SCAN_CTL	RST_N	LEVEL_INT_MODE	vss	INT3_VEC[7]	vss	IACK_INT3_N	XRW2_N	XRR0_N	vss	XRDO[3]	vss	OU0	OU1
12	PPROTECT_EN	vdd4	ESCAN_EN	ESCAN_IN	INT3_VEC[10]	INT3_VEC[5]	INT3_VEC[1]	IACK_INT2_N	SEL_XR_RD[0]	XRR2_N	XRDO[13]	XRDO[9]	XRDO[5]	XRDO[2]	XRDO[0]
13	NMI_N	EDB_WAIT_EN	MP_BOOT	INT3_VEC[13]	INT3_VEC[9]	INT3_VEC[4]	INT3_VEC[0]	IACK_INT0_N	vdd4	XRW0_N	XRDO[14]	XRDO[10]		vdd4	XRDO[4]
15	MP_RST	vss	INT3_VEC[14]	INT3_VEC[11]	vdd4	INT3_VEC[3]	IACK_NMI_N	SEL_XR_RD[1]	XRW3_N	XRR3_N	vdd4	XRDO[11]	XRDO[7]	PTST	PTST_TE
15		INT3_VEC[15]	INT3_VEC[12]	INT3_VEC[8]	INT3_VEC[6]	INT3_VEC[2]	vdd4	IACK_INT1_N	vss	XRW1_N	XRR1_N	XRDO[15]	XRDO[12]	XRDO[8]	XRDO[6]

9 Known Limitations

This section describes known bugs and limitations.

9.1 INT3_VEC Pin

The two least significant bits of this port must be tied to VSS. This limits the choice of possible interrupt service routine addresses for interrupt 3 to addresses that are divisable by 4.

Notes

Headquarters

LSI Logic Corporation
North American Headquarters
Milpitas CA
Tel: 408.433.8000
Fax: 408.433.8989

LSI Logic Europe Ltd
European Headquarters
Bracknell England
Tel: 44.1344.426544
Fax: 44.1344.481039

LSI Logic K.K.
Headquarters
Tokyo Japan
Tel: 81.3.5463.7821
Fax: 81.3.5463.7820

To receive product literature, visit us at http://www.lsilogic.com.

ISO 9000 Certified

Printed on Recycled Paper

Printed in USA
Order No. C15041
Doc. No. DB09-000092-00