APR26
Rev. 1, 10/2005

Interfacing Flash Memory with the DSP56300 Family of Digital Signal Processors

by Phil Brewer

This application note describes how to interface Flash memory to the DSP56300 family of DSPs. This document is a supplement to the *DSP56300 24-Bit Digital Signal Processor Family Manual* and to the user's manuals and technical data sheets for devices in the DSP56300 family.

Methods are described for interfacing various types of memory to the DSP56300 memory expansion port to assist the DSP hardware engineer in using the processor resources and generating an optimized memory design. These memory designs use a minimum of additional parts. Taking advantage of the available DSP control lines makes virtually glueless external memory interfaces possible, thereby reducing cost and using fewer parts in an application memory design.

Specifically, this application note describes asynchronous implementations for Flash and programmable erasable read-only memory (PEROM) using the DSP56303. The DSP56303 is representative of the DSP56300 family and has all the essential family features. Several examples of memory configurations assist hardware designers in implementing a DSP-based memory design.

CONTENTS

1 Overview

The DSP56300 family of DSPs uses a programmable 24-bit fixed-point core. This core is a high-performance, single-clock-cycle-per-instruction engine that consists of a peripheral/memory expansion port (port A), external memory and peripheral DRAM controller, data arithmetic logic unit (data ALU), address generation unit (AGU), instruction cache controller, program control unit, internal concurrent six-channel DMA controller, PLL clock generator, on-chip emulation (OnCE™) module, JTAG test access port (TAP) compatible with the **IEEE** Std. 1149.1™ Standard, and a peripheral and memory expansion bus. The main features of the DSP56300 core include:

- Modified Harvard architecture with 24-bit instruction and 24-bit data width
- Fully pipelined 24 × 24-bit parallel multiplier-accumulator (MAC)
- 56-bit parallel barrel shifter
- 16-bit arithmetic mode of operation
- Highly parallel instruction set
- Position Independent Code (PIC) instruction set support
- Unique DSP addressing modes
- Internal memory-expandable hardware stack
- Nested hardware DO loops
- Fast auto-return interrupts
- Instruction cache
- External memory and peripheral access attribute select support
- Phase lock loop (PLL)
- Program address tracing support

The main differences between derivatives of the DSP56300 family are the size of the internal memory and the types of peripherals and hardware accelerators.

2 Non-Volatile Memory Families

The following non-volatile memory families were considered for this application note:

- Flash memory:
  - 12.0 V Flash (bulk erase)
  - 5.0 V Flash (sector erase); 8/16-bit organization
  - 3. 3V Flash (sector erase); 8/16-bit organization
- PEROM memory:
  - 5.0 V PEROM (sector erase); 8/16-bit organization
  - 3.3 V PEROM (sector erase); 8/16-bit organization
  - 2.7 V PEROM (sector erase); 8/16-bit organization

The examples in this application note use the most popular memory types and types requiring little or no supporting hardware (that is, glue logic). The examples provided for these memory types can assist designers in implementing other memory families and memory types with the DSP56300 family.

3 References

- The following data sheets and manuals are available at the Freescale web site listed on the back cover of this application note:
  - *DSP56300 24-Bit Digital Signal Processor Family Manual* (DSP56300FM).
  - *DSP56301 Technical Data Sheet* (DSP56301).
  - *DSP56302 Technical Data Sheet* (DSP56302).
  - *DSP56303 Technical Data Sheet* (DSP56303).
  - *DSP56301 24-bit Digital Signal Processor User's Manual* (DSP56301UM).
  - *DSP56303 24-bit Digital Signal Processor User's Manual* (DSP56303UM).
  - *DSP56303EVM User's Manual*.
- *Atmel Corporation Nonvolatile Memory Data Book*, Atmel, May 1996.
- *Advanced Micro Devices FLASH Memory Products 1994/1995 Data Book/Handbook*, AMD, 1995.
- *Quality Semiconductor QuickSwitch® Products Databook*, Quality Semiconductor, 1995.
- *Quality Semiconductor Application Note AN-11, 'Bus Switches Provide 5v and 3v Logic Conversion with Zero Delay'*, Quality Semiconductor, 1995.

4 DSP56300 Control Signals

This section describes only the DSP control signals which are used in the memory implementation examples. Other memory configuration implementations may require other memory support features of the DSP56300 family. These additional memory control signals are described in the port A chapter of the *DSP56300 24-Bit Digital Signal Processor Family Manual*.

- Read Data Enable ($\overline{RD}$). Output signal asserted during an external memory or peripheral read access.
- Write Data Enable ($\overline{WR}$). Output signal asserted during an external memory or peripheral write access.
- Address Bus (A[0–17]). Address lines that allow the DSP563003 to address 256K words of external memory or peripherals directly. These output signals are asserted only during external memory or peripheral read or write accesses. These signal lines maintain state when external memory spaces are not being accessed.
- Data Bus (D0–D23. Data lines on the DSP56303 that are bidirectional signals asserted only during external program and data memory accesses. These signal lines maintain state when external memory spaces are not being accessed.
- Address Attribute/Row Address Strobe (AA[0–3]/$\overline{RAS}$[0–3]). When the Address Attribute, AA, option is selected for these signal lines, they can function as chip selects or additional address lines. When the $\overline{RAS}$ option is selected, they can function as Row Address Strobe lines for DRAM interfacing.

4.1 DSP External Memory Timing

The DSP56300 family derives its core clock from one of various sources (see the PLL and clock generator chapter in the *DSP56300 24-Bit Digital Signal Processor Family Manual*). All memory interface timings are derived from the period of the DSP core clock. For example, if the DSP core clock frequency is 80 MHz, the memory timings are based on a 12.5 nS clock cycle time, and an external memory typically requires less than 12.5 nS access time for one DSP wait state operation. However, these timings are affected by several factors, such as the use of the phase lock loop, the use of an external frequency over or under 4 MHz, the source of the external frequency, and propagation delays in the DSP itself. Any of these factors can cause this value to deviate from 12.5 nS. **Table 1** represents expected required memory performance data at an 80 MHz DSP core frequency and various wait states using the DSP56303.

Table 1. External Memory Speeds with DSP Wait States

External Clock (MHz)	DF	MF	PDF	WS	Core Clock (MHz)	TC (nS)	t_{AA} – max (nS)	t_{AW} – min (nS)
4.00	1	20	1	1	80.00	12.5	12.4	17.9
4.00	1	20	1	2	80.00	12.5	24.9	30.4
4.00	1	20	1	3	80.00	12.5	37.4	42.9
4.00	1	20	1	4	80.00	12.5	49.9	55.4
4.00	1	20	1	5	80.00	12.5	62.4	67.9
4.00	1	20	1	6	80.00	12.5	74.9	80.4
4.00	1	20	1	7	80.00	12.5	87.4	92.9
4.00	1	20	1	8	80.00	12.5	99.9	105.4
4.00	1	20	1	9	80.00	12.5	112.4	117.9
4.00	1	20	1	10	80.00	12.5	124.9	130.4
4.00	1	20	1	11	80.00	12.5	137.4	142.9
4.00	1	20	1	12	80.00	12.5	149.9	155.4
4.00	1	20	1	13	80.00	12.5	162.4	167.9
4.00	1	20	1	14	80.00	12.5	174.9	180.4
4.00	1	20	1	15	80.00	12.5	187.4	192.9
4.00	1	20	1	16	80.00	12.5	199.9	205.4
4.00	1	20	1	17	80.00	12.5	212.4	217.9
4.00	1	20	1	18	80.00	12.5	224.9	230.4
4.00	1	20	1	19	80.00	12.5	237.4	242.9
4.00	1	20	1	20	80.00	12.5	249.9	255.4

Notes:
1. DF = Division Factor defined by the DF0-DR2 bits in the PCTL Register
2. MF = PLL Multiplication Factor defined by the MF0-MF11 bits in the PCTL Register
3. PDF = Predivision Factor defined by the PD0-PD3 bits in the PCTL Register
4. WS = Wait States
5. TC = Clock Cycle Time
6. t_{AA} = Data access time (address and AA valid to input data valid)
7. t_{AW} = Data access time (address and AA valid to $\overline{WR}$ deassertion)

4.1.1 DSP56303 External Memory Asynchronous Read Timing

During reads from external asynchronous memory, the DSP56303 memory read access is controlled by the following steps:

1. The required memory address is asserted. The memory address is created by combining the address bus, A[0–17], and the Address Attributes, AA[0–3].
2. After a delay of t_{AR} (address valid to $\overline{RD}$ assertion time), the Read enable signal, $\overline{RD}$, is asserted.
3. Before a delay of t_{OE} (i.e., $\overline{RD}$ assertion to input data valid), the memory device puts valid data on the data bus.
4. The DSP latches the data bus data and deasserts $\overline{RD}$. The DSP does not require any data hold time, t_{OHZ}, after deassertion of the $\overline{RD}$ signal.

The data access time, t_{AA} (address and AA valid to input data valid), is the time delay typically used by memory devices to specify data access timing. The t_{AA} for a memory device must be less than or equal to the DSP t_{AA} time for valid data transfers.

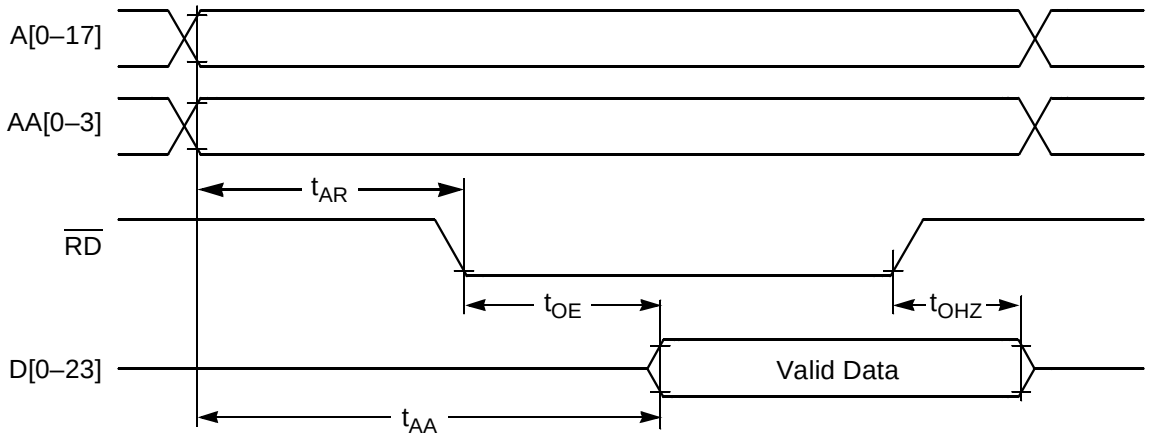

Figure 1. External Memory Bus Asynchronous Read Timing

4.1.2 DSP56303 External Memory Asynchronous Write Timing

During writes to external asynchronous memory, the DSP56303 memory write access is controlled by the following steps:

1. The memory select address is asserted. The memory select address is created by combining the Address bus, A0–A17, and the Address Attributes AA0–AA3.
2. After a delay of t_{AS}—address valid to $\overline{WR}$ assertion time—the Write enable signal, $\overline{WR}$, is asserted.
3. Before a delay of t_{WA}—$\overline{WR}$ assertion to output data valid—the DSP places valid data on the data bus.
4. After a delay of t_{DW}—data valid to $\overline{WR}$ deassertion (data set-up time)—the DSP deasserts the $\overline{WR}$ signal.
5. Then the DSP deasserts the address and address attributes after t_{WR}—$\overline{WR}$ deassertion to address not valid—while holding the data valid for t_{DH}.

The data access time, t_{AW} (i.e., address and AA valid to $\overline{WR}$ deassertion), is typically the critical timing specification for memory devices. The t_{AW} for a memory device must be less than or equal to the DSP t_{AW} time for valid data transfers.

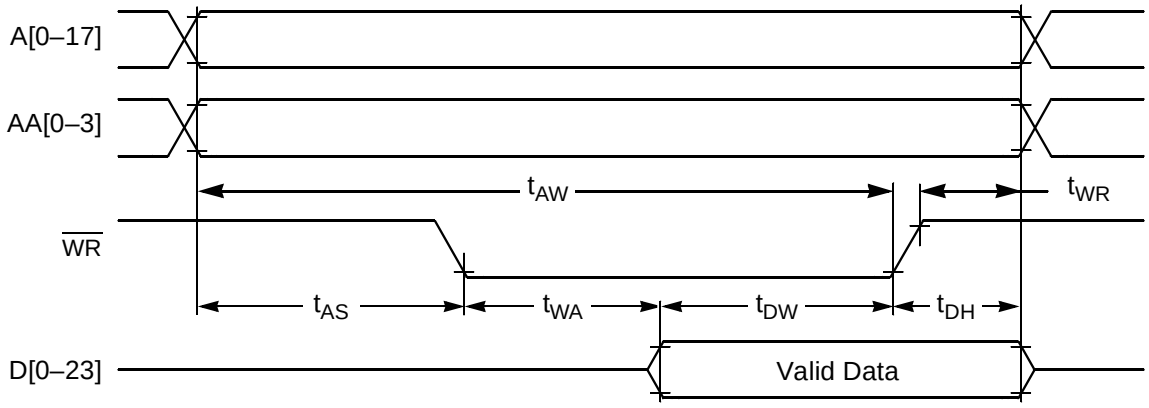

Figure 2. External Asynchronous Memory Bus Write Timing

4.2 DSP Memory Control Registers

You must configure the following control registers to access external memory or peripherals properly when using the DSP56303:

- DSP PLL and Clock Generation register
- Bus Control Register
- DRAM Control Register (if DRAM is used)
- Address Attribute Registers

4.2.1 DSP PLL and Clock Generation

You must set the core speed of the DSP for optimum processor and memory performance by configuring the DSP PLL and Clock Generation in the PLL Control (PCTL) register. For detailed information on the PCTL register, see the PLL and Clock Generator chapter in the *DSP56300 24-Bit Digital Signal Processor Family Manual*. The PLL Control register is an X data I/O mapped 24-bit register. The PCTL register can be separated into four sub-functions:

- *Frequency predivider*. The input clock frequency can be predivided before it is passed it to the PLL loop frequency multiplier. This frequency predivider has a programmable division factor range of 1–16. It is set by controlling the values placed in the PCTL register bits 20–23. The division factor is the binary value stored in bits 20–23, plus one.
- *PLL loop frequency multiplier*. The clock frequency output from the predivider is multiplied by the voltage-controlled oscillator (V_{CO}). The multiplication factor is set by the value in the PCTL register bits 0–11. The multiplication factor is the binary value stored in bits 0–11, plus one.
- *Frequency low-power divider*. The low-power divider (LPD) can divide the output frequency of the V_{CO} before it is used by the DSP core. This frequency low-power divider has a programmable division factor range of 1–128. It is set by controlling the values placed in the PCTL register bits 12–14. The low-power division factor is 2^n, where n is the value in PCTL bits 12–14.

- *Frequency control bits.* The following five control bits control the input frequency source, the PLL during Stop mode, the activation of the PLL V_{CO}, and the external availability of the core clock:
  - — Crystal frequency is less than 200 kHz, bit 15
  - — Disable XTAL drive output, bit 16
  - — PLL runs during STOP mode, bit 17
  - — Enable PLL operation, bit 18
  - — Disable core clock output, bit 19

Device operating core frequency is set by the control bits in the PCTL register as follows:

$$F_{CORE} = \frac{F_{EXTAL} \times MF}{PDF \times DF}$$

where

- F_{CORE} is the DSP core frequency.
- F_{EXTAL} is the external input frequency source present on the EXTAL pin.
- PDF is the predivider factor defined by the PD[0–3] bits in PCTL.
- MF is the PLL Multiplication Factor defined by the MF[0–11] bits in PCTL.
- DF is the Division Factor defined by the DF[0–2] bits in PCTL.

4.2.2 Bus Control Register (BCR)

The Bus Control Register (BCR) is a 24-bit X data I/O register that controls the external bus wait states generated for each address attribute area 0–3 and assigns a default value to all memory areas not covered by an address attribute area. Each area can have up to 31 wait states. Select the correct number of wait states for each memory configuration using this register.

- Wait states for address attribute area 0, allowing 0–31 wait states, bits 0–4
- Wait states for address attribute area 1, allowing 0–31 wait states, bits 5–9
- Wait states for address attribute area 2, allowing 0–7 wait states, bits 10–12
- Wait states for address attribute area 3, allowing 0–7 wait states, bits 13–15
- Wait states for address areas not specified by areas 0–3, allowing 0–31 wait states, bits 16–20
- The bus state status, bit 21
- Enable Bus Lock Hold, bit 22
- Enable Bus Request Hold, bit 23

4.2.3 Address Attribute Control Registers (AAR0–AAR3)

Four 24-bit Address Attribute Control registers in the X data I/O memory space control the activity of the AA[0–3]/$\overline{\text{RAS0}}$–$\overline{\text{RAS3}}$ pins. Each AA/$\overline{\text{RAS}}$ pin is asserted if the address and memory space of the appropriate AARx matches the requested external memory address and address space.

- Specify external memory access type; select from Synchronous SRAM, Asynchronous SRAM, and DRAM accesses, bits 0–1
- Pull the AA pin high, bit 2
- Activate the AA pin during external program space accesses, bit 3
- Activate the AA pin during external X data space accesses, bit 4
- Activate the AA pin during external Y data space accesses, bit 5
- Move the eight Least Significant bits (LSBs) of the address to the eight Most Significant bits (MSBs) of the external address bus, bit 6
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7
- Specify the number of address bits to compare, allowing the use of 0–12 address bits, bits 8–11
- Specify the most significant portion of the address to compare, bits 12–23

4.2.4 Operating Mode Register (OMR)

The Operating Mode Register (OMR) is a 24-bit I/O register that selects the operating mode of the DSP, external memory controls, and stack extension controls. The following flags are applicable to memory interfacing:

- The DSP operating mode is specified by MA–MD, bits 0–3.
- The External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4.
- The Memory Switch mode bit reconfigures internal memory spaces, bit 7.
- The Transfer Acknowledge synchronize select bit selects the synchronization method for the Transfer Acknowledge ($\overline{\text{TA}}$) pin, bit 11.
- The Bus Release Timing bit selects between a fast and slow bus release of the $\overline{\text{BB}}$ pin, bit 12.
- The Address Attribute Priority Disable bit allows the address attribute pins, AA[0–3], to be used in any combination, bit 14.

4.2.5 Status Register (SR)

The Status Register (SR) is a 24-bit I/O register that selects and monitors the results of arithmetic computations and the current state of the DSP. The following flags apply to memory interfacing:

- Sixteen-Bit Compatibility mode enables full compatibility with object code written for the DSP56000 family, bit 13.
- Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, bit 19.

5 +5 V Flash Memory

Flash memory provides non-volatile program and data storage, which is in-circuit reprogrammable and offers relatively fast access times. However, even the fastest Flash memories require a DSP core running at 80 MHz to generate several external memory wait states, with each wait state equivalent to one clock period of the DSP core (i.e., one wait state would be approximately 12.5 nS for a core running at 80 MHz).

When the external memory device must reside in a stable address during an entire external access, a DSP56300 family device incurs an automatic one wait state penalty. Since Flash devices require address stability, the DSP operates with at least one wait state when using these external memories.

The following three design examples illustrate how easy and flexible it is to use AMD Flash memory with the Freescale DSP56300 family.

- The first example uses three 512K × 8-bit Flash devices to form a block of 512K × 24-bit words, all of which are accessible as program, X data, and Y data memory. This example shows one solution for an embedded system, which executes code and references data tables from the shared 512K word bank of Flash memory.
- The second example uses three 512K × 8-bit Flash devices to form a block of 512K × 24-bit words. In this example, the memory is accessed as four separate memory areas: a 128K word boot area, a 128K word program area, a 128K word X data area, and a 128K word Y data area.
- The third example uses a 256K × 16-bit Flash device that is accessed as a 128K × 16-bit X data area and a 128K × 16-bit Y data area. This example demonstrates how a system can change or upgrade its 16-bit reference data tables.

5.1 512K × 24-bit Shared Memory Flash Example

This example implements a 512K × 24-bit word memory space that is accessible as program, X data, and Y data memory. This design uses three AMD Am29F040 devices. See **Figure 3** for the memory map layout, **Figure 4** for the block diagram, **Example 1** for the program, and **Figure 5** for the schematic. Since the program, X data and Y data spaces are shared, any location referenced by a program memory access also accesses the same memory location during an X data or Y data access. Since the Flash memory is in the DSP program space, program control can be turned over to the program in Flash at $C00000 at reset. This allows an embedded application to boot and run directly from Flash, using the internal 1 K program cache to help speed repeated program operations. Since the X data and Y data spaces also access the Flash, data can be stored in the Flash for use after boot.

For this example, the DSP core runs at 80 MHz, generated by the PLL from a 4.000 MHz crystal. The memory bank consists of three AMD Am29F040 devices, which are organized as 512K × 8-bits to provide a 24-bit word. These are 5 V devices, with a 90 nS access time. The 90 nS access time requires eight wait states for correct operation.

The DSP56303 data bus is not 5 V tolerant, so level converters are needed to interface with the 5 V memory devices. The devices used are Quality Semiconductor QS3245 QuickSwitch® 8-bit Bus Switches. These switches allow the connection of 3.3 V CMOS logic (the DSP data bus) on one side and 5 V TTL-compatible logic (the memory devices) on the other side, effectively providing a 3.3 V to 5 V level conversion. The propagation delay of the switch is 0.25 nS, which is not significant in this design.

This example demonstrates a minimal external memory design using a single bank of Flash memory in the external memory spaces. If you add more memory or other devices to the bus, additional address decode logic may be needed.

After reset with Mode 0 selected, the DSP starts fetching instructions from external memory at location P:$C00000. Although this DSP core uses 24-bit addressing internally, only eighteen external address lines (A0–A17) are provided. Consequently, the top six bits are stripped, and the address presented on the address bus is $000000. The Flash memory is configured to respond to all external memory requests, as $\overline{CE}$ is asserted at all times. Since the eighteen bits can address only 256K words, only the first 256K words of the Flash are accessible, and data aliasing occurs every 256K words (i.e., accessing location 256K returns the contents of the same Flash

location as accessing location 0 or location 512K, and so on). To access the upper 256K words of Flash, address attribute 0 (AA0) is used as Flash address bit A18. AAR0 is configured so that AA0 is set for all address accesses (program, X data, and Y data) in the range $C40000–$C7FFFF and cleared otherwise. However, AA0 is set on exit from Reset and stays set until AAR0 is initialized. This means that initially, when instructions are fetched from $C00000, they are not fetched from Flash location $000000 but from $040000.

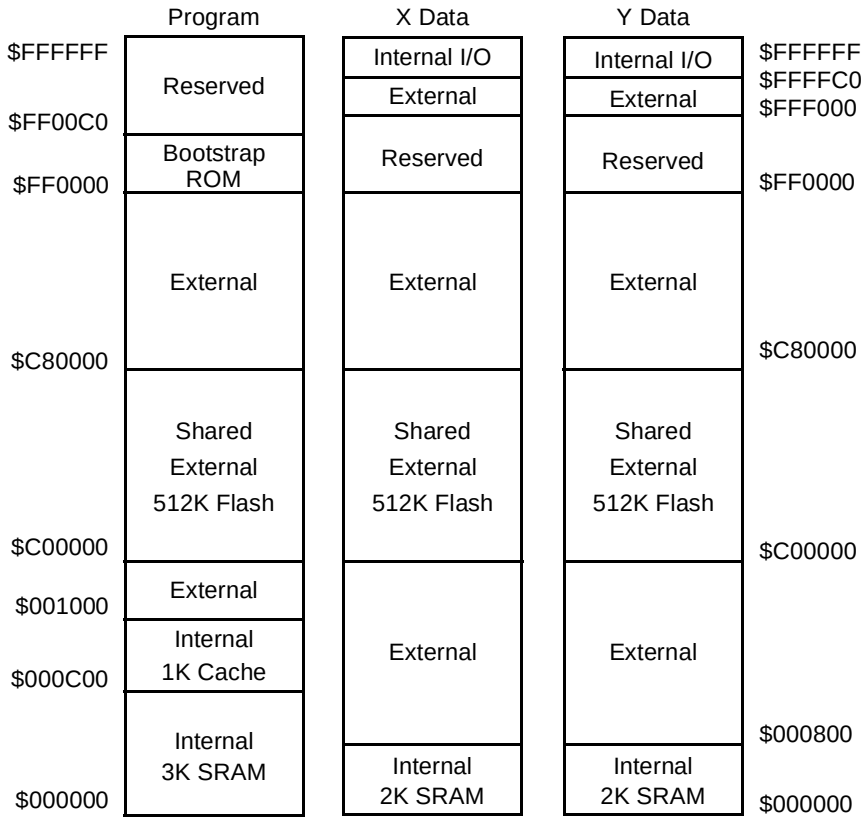

Figure 3. 512K × 24-bit Flash Memory Map

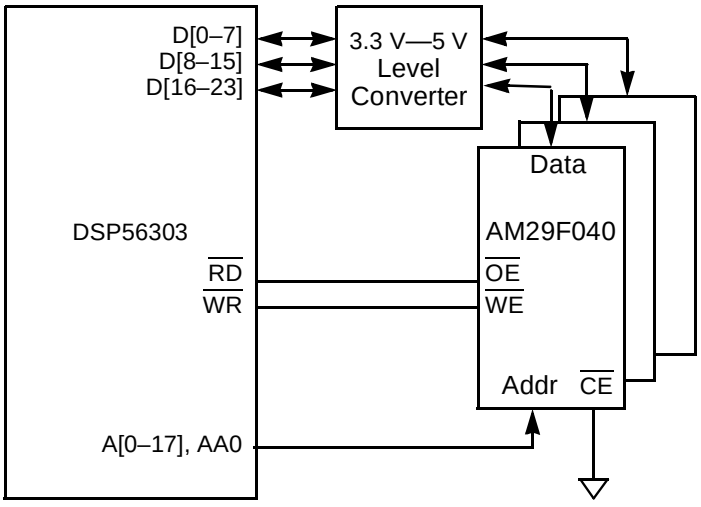

Figure 4. 128K × 24-bit Boot, Program, X Data, Y Data Flash Block Diagram

5.1.1 Flash Timing Requirements

For the Flash device to work properly, its timing requirements must be met. Following are the timing requirements for the Am29F040-90 512K × 8-bit 90 nS Flash. **Table 2** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 5**.

Table 2. Am29F040-90 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Read Cycle Time	t_{RC}	90 nS	—
Address to Output Delay	t_{ACC}	—	90 nS
Chip Enable to Output Delay	t_{CE}	—	90 nS
Output Enable to Output Delay	t_{OE}	—	35 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0 nS	—

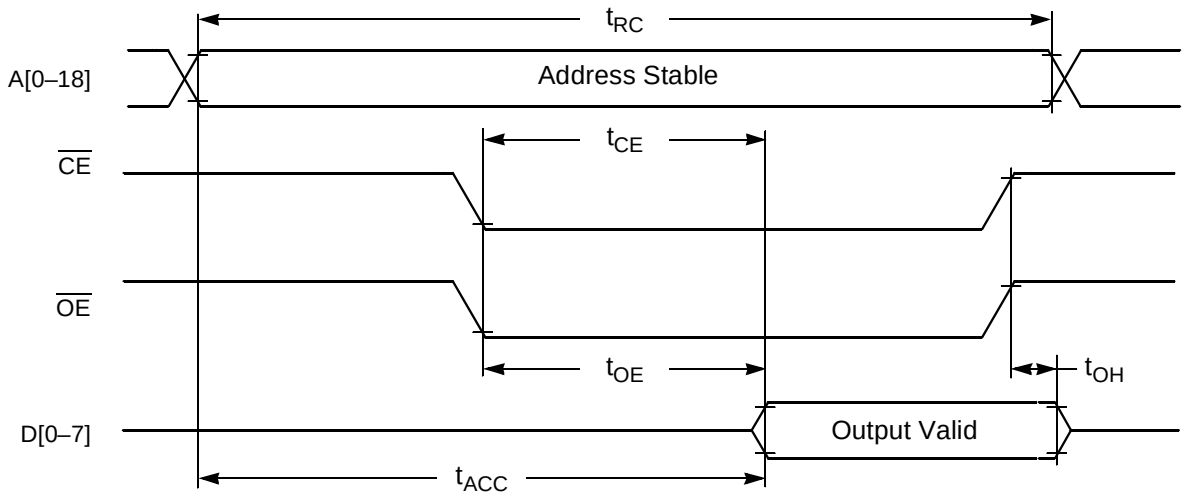

Figure 5. Am29F040 Memory Read Cycle Timing Diagram

Table 3 shows the memory write timing specification values used in the memory write cycle timing diagram, **Figure 6**.

Table 3. Am29F040-90 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write Cycle Time	t_{WC}	90 nS	—
Address Set-up Time	t_{AS}	0 nS	—
$\overline{CE}$ Set-up Time	t_{CS}	0 nS	—
Write Pulse Width	t_{WP}	35 nS	—
Data Set-up Time	t_{DS}	35 nS	—
Data Hold Time	t_{DH}	0 nS	—

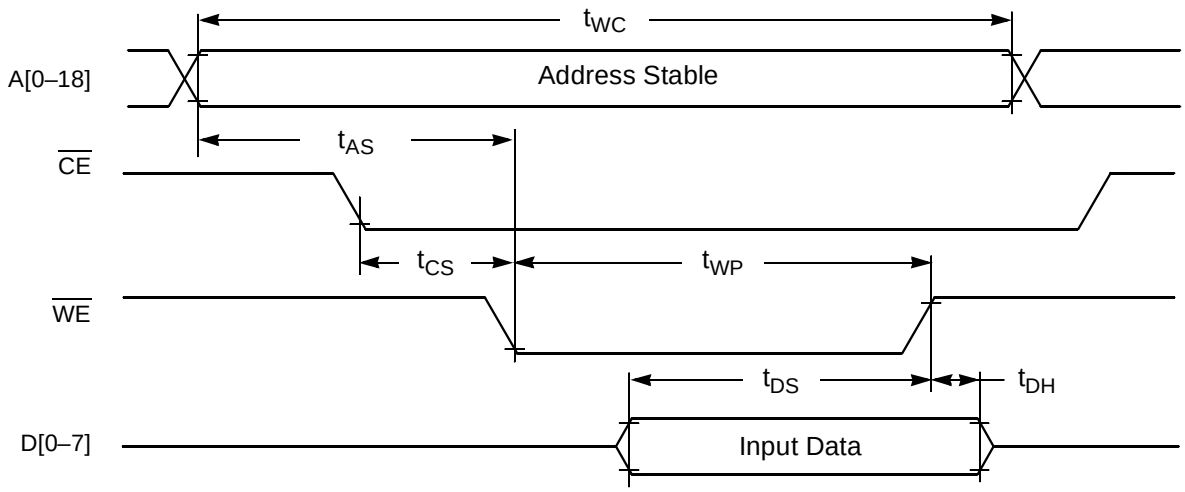

Figure 6. Am29F040 Memory Write Cycle Timing Diagram

A location in the Flash can be programmed if that location has not been written with a zero in any of the eight bits. If the memory location has been previously written, then it must first be erased. The Am29F040 device is organized as eight sectors of 64K bytes each, and to erase one memory location, the sector containing that memory location must be erased.

To write to an erased memory location, write the following data sequence to the Flash memory:

1. Write `$AAAAAA` to location `$5555` relative to the Flash.
2. Write `$555555` to location `$2AAA` relative to the Flash.
3. Write `$A0A0A0` to location `$5555` relative to the Flash.
4. Write 24-bit data to Address in the Flash.
5. Read Address until data read = data written.

To erase a sector, write the following data sequence to the Flash memory:

1. Write `$AAAAAA` to location `$5555` relative to the Flash.
2. Write `$555555` to location `$2AAA` relative to the Flash.
3. Write `$808080` to location `$5555` relative to the Flash.
4. Write `$AAAAAA` to location `$5555` relative to the Flash.
5. Write `$555555` to location `$2AAA` relative to the Flash.
6. Write `$303030` to Sector Address Location relative to the Flash.
7. Read location in erasing sector until data read = `$FFFFFF`.

5.1.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 512K × 24-bit Program, X data, and Y data space memory configuration, set up the following DSP control registers. Set the core speed of the DSP for optimum processor and memory performance using the DSP PLL and Clock Generation (PCTL) register. For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal.

The PCTL value combines the following bits for each feature:

- Desired core frequency = 80 MHz
- Given the external frequency = 4.000 MHz
- Predivider value = 1, bits 20–23 = $0
- Low-power divider value = 1, bits 12–14 = $0
- V_{CO} Multiplication value = 20, bits 0–11 = $013
- Crystal is not less than 200 kHz, bit 15 = 0
- Disable XTAL drive output, bit 16 = 0
- PLL runs during STOP, bit 17 = 1
- Enable PLL operation, bit 18 = 1
- Disable core clock output, bit 19 = 1

The value loaded into the PCTL is `$0E0013`.

AA0 is used as Flash A18, the most significant address line of the external 512K Flash memory bank accesses in the address range from `$C40000` to `$C7FFFF` during program space requests. Using Address Attribute Register 0 (AAR0), configure the memory address space requirements for the Address Attribute Pin 0. The AAR0 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA0 pin high when selected, bit 2 = $1.
- Activate the AA0 pin during external program space accesses, bit 3 = $1.
- Activate the AA0 pin during external X data space accesses, bit 4 = $1.
- Activate the AA0 pin during external Y data space accesses, bit 5 = $1.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = $0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = $0.
- Specify the number of address bits to compare, bits 8–11 = $6.
- Specify the most significant portion of the address to compare, bits 12–23 = $C40.

The value loaded into the AAR0 is `$C4063D`. The value loaded into AAR1, AAR2 and AAR3 is `$000000`.

Using the Bus Control Register (BCR), select the proper number of wait states for the memory configuration. The BCR value combines the following bits for each feature:

- Address attribute area 0 wait states, bits 0–4 = $8.
- Address attribute area 1 wait states, bits 5–9 = $0.
- Address attribute area 2 wait states, bits 10–12 = $0.
- Address attribute area 3 wait states, bits 13–15 = $0.
- Default address area wait states, bits 16–20 = $8.
- Bus state status, bit 21 = 0.

- Enable Bus Lock Hold, bit 22 = 0.
- Enable Bus Request Hold, bit 23 =0.

The value loaded into the BCR is `$080008`.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR value combines the following bits for each feature:

- MA-MD bits specify the DSP operating mode, bits 0–3 = $0.
- External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4 = $0.
- Memory Switch Mode bit reconfigures internal memory spaces, bit 7 = $0.
- Transfer Acknowledge Synchronize Select bit selects the synchronization method for the Transfer Acknowledge ($\overline{TA}$) pin, bit 11 = $0
- Bus Release Timing bit selects between a fast and slow bus release of the $\overline{BB}$ pin, bit 12 = $0.
- Address Attribute Priority Disable bit allows the Address Attribute pins, AA0–AA3, to be used in any combination, bit 14 = $1.
- All other OMR bits are selected for their defaults of $0.

The value loaded into the OMR is `$004000`.

Configure the memory mode of the DSP using the Status Register (SR). The SR value combines the following bits for each feature:

- Sixteen-bit Compatibility mode enables full compatibility with object code written for the DSP56000 family of DSPs, bit 13 = $0.
- Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, bit 19 = $1.
- All other Status Register bits are selected for their defaults of $0.

The value loaded into the SR is `$080000`, which is the value loaded during reset.

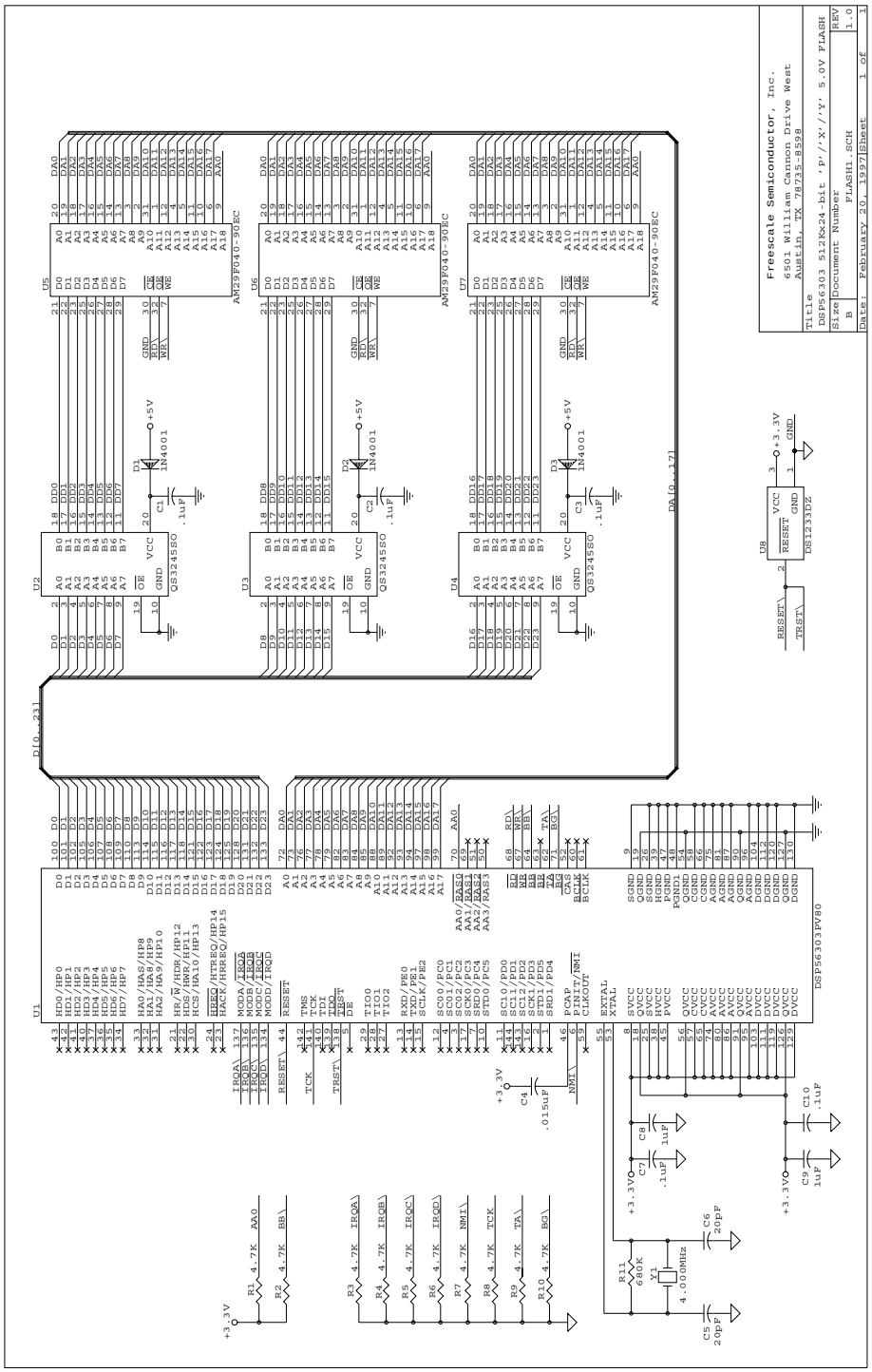

Figure 7. 512K × 24-bit Program, X data, Y data Flash Schematic

Example 1. 512K x 24-bit Program, X Data, and Y Data Space Flash Memory Checksum Program

```
       Freescale DSP56300 Assembler Version 6.0.1.6   97-02-23 11:05:32 flash1.asm
       1                                  page    132,60,3,3,
       2                  ;
       3                  ;     flash1.asm - Simple program to calculate and write a 24-bit
Checksum
       4                  ;                  for a 512K x 24-bit block of program memory using
       5                  ;                    a DSP56303.
       6                  ;
       7                  ;       The program runs in Internal P:RAM to calculate the checksum
       8                  ;        on External P:FLASH from $C00000 - $C7FFFF @ 8w/s
       9                  ;
       10
       11     C00000      FMemStart     equ    $C00000
       12     C80000      FMemEnd       equ    $C80000
       13     080000      FMemSize      equ    FMemEnd-FMemStart ; Last Word is stored
Checksum
       14
       15                 ;--- Program Specific Storage Locations (X DATA SPACE)
       16                 CKSUM_COMPUTED
       17     000000                    equ    $000000          ; Computed Checksum Value
       18     000001      CKSUM_READ    equ    $000001          ; Old Checksum Value
       19
       20                 ;--- DSP56303 Control Registers (X I/O SPACE)
       21     FFFFFB      BCR           equ    $FFFFFB    ; Bus Control Register
       22     FFFFFD      PCTL          equ    $FFFFFD    ; PLL Control Register
       23     FFFFF9      AAR0          equ    $FFFFF9    ; Address Attribute Register 0
       24
       25                 ;--- PCTL value = 0x0E0013
       26     000000      prediv        equ    0          ; Predivider = 1
       27     000000      lowdiv        equ    0          ; Low Power Divider = 1
       28     000013      pllmul        equ    19         ; VCO Mult = 20;
(19+1)*4.00MHz=80.00MHz
       29     000000      crystal       equ    0          ; No, Crystal not less than 200kHz
       30     000000      disXTAL       equ    0          ; No, do not disable crystal use
       31     020000      pllstop       equ    $020000    ; Yes, PLL runs during STOP
       32     040000      enpll         equ    $040000    ; Yes, enable PLL operation
       33     080000      disclk        equ    $080000    ; Yes, disable CORE clock output
       34     0E0013      PCTL_value    equ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk
       35
       36                 ;--- AAR0 value = 0xC4063D
       37     000001      acctype       equ    1          ; External Memory access type = 0x1
       38     000004      aahigh        equ    $4         ; Enable AA0 pin high when selected
       39     000008      aap           equ    $8         ; Yes, Enable AA0 pin on ext P access
       40     000010      aax           equ    $10        ; Yes, Enable AA0 pin on ext X access
       41     000020      aay           equ    $20        ; Yes, Enable AA0 pin on ext Y access
       42     000000      aswap         equ    0          ; No, Enable address bus swap
       43     000000      enpack        equ    0          ; No, Enable packing/unpacking logic
       44     000600      nadd          equ    $000600    ; Compare 6 address bits
       45     C40000      msadd         equ    $C40000    ; Most significant part of address,
       46                                                  ; $C40000 - C7ffff, to compare.
       47                                                  ; (1101,01xx,xxxx,xxxx,xxxx,xxxx)
       48     C4063D      AAR0_value    equ
acctype+aahigh+aap+aax+aay+aswap+enpack+nadd+msadd
       49
       50                 ;--- BCR value = 0x080008
       51     000008      aaa0ws        equ    $8         ; Address Attribute Area 0 w/s = 8
       52     000000      aaa1ws        equ    0          ; Address Attribute Area 1 w/s = 0
       53     000000      aaa2ws        equ    0          ; Address Attribute Area 2 w/s = 0
```

```
       54     000000       aaa3ws        equ    0          ; Address Attribute Area 3 w/s = 0
       55     080000       defws         equ    $080000    ; Default Address Area w/s = 8
       56     000000       busss         equ    0          ; Bus state status = 0
       57     000000       enblh         equ    0          ; Enable Bus Lock Hold = 0
       58     000000       enbrh         equ    0          ; Enable Bus Request Hold = 0
       59     080008       BCR_value     equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       60
       61                  ;---------------------------------------------------------------
       62     P:000100             org           p:$100     ;Keep the program in internal RAM
       63
       64                  flash1
       65
       66                  ;---------- Initialization Section ----------
       67     P:000100 08F4BD      movep  #PCTL_value,x:PCTL; Set PLL Control Register
                       0E0013
       68     P:000102 05F439      movec  #$080000,SR      ; Enable 1K Cache
                       080000
       69     P:000104 08F4BB      movep  #BCR_value,x:BCR ; Set FLASH's wait states to 12
                       080008
       70     P:000106 08F4B9      movep  #AAR0_value,x:AAR0 ; Set Address Attribute Reg0
                       C4063D
       71
       72     P:000108 05F420      move   #-1,m0           ; Set LINEAR addressing mode
                       FFFFFF
       73     P:00010A 60F400      move   #FMemStart,r0    ; Set starting address of Flash
                       C00000
       74     P:00010C 70F400      move   #FMemSize-1,n0   ; Set to Size of Flash - Checksum
                       07FFFF
       75
       76     P:00010E 200013      clr    a
       77     P:00010F 20001B      clr    b                ; Clear Checksum Accumulator
       78     P:000110 560000      move   a,x:CKSUM_COMPUTED; Initialize computed checksum ->
$000000
       79     P:000111 560100      move   a,x:CKSUM_READ   ; Initialize read checksum -> $000000
       80
       81                  ;---------- Compute the 24-bit Checksum ----------
       82     P:000112 06D810      dor    n0,_loop
                       000003
       83     P:000114 07D88E      move   p:(r0)+,a        ; Get the FLASH location Value
       84     P:000115 200018      add    a,b              ; and Compute the Checksum
       85                  _loop
       86
       87     P:000116 07E08E      move   p:(r0),a         ; Get FLASH Old Checksum Value
       88     P:000117 570000      move   b,x:CKSUM_COMPUTED; Save Computed 24-bit Checksum
       89     P:000118 560100      move   a,x:CKSUM_READ ; Save 24-bit Checksum read from FLASH
       90     P:000119 20001B      clr    b                ; Zero b0 and b2 registers
       91     P:00011A 578000      move   x:CKSUM_COMPUTED,b ; Put computed 24-bit checksum in
b1
       92     P:00011B 20000D      cmp    a,b              ; Old Checksum = New Checksum?
       93     P:00011C 0D104A      beq    _done            ; Yes
                       000035
       94                                                  ; No
       95                  ;---------- See if Checksum Location is Erased ----------
       96     P:00011E 44F400      move   #$FFFFFF,x0      ; FLASH erased value
                       FFFFFF
       97     P:000120 200045      cmp    x0,a             ; Contents of checksum location
Erased?
       98     P:000121 0D104A      beq    _write_checksum  ; Yes
                       000020
```

```
99                                              ; No
100         ;-------------------------------------------------------------------
101         ;                       N O T E
102         ;-------------------------------------------------------------------
103         ; If at least 64Kx24-bit words of external RAM is available in the system
104         ; then; 1)  the last sector of the FLASH can be read into external RAM
105         ;       2)  the last sector of the FLASH can be erased,
106         ;       3)  and then the last sector of data along with the new checksum
107         ;           can be written back into the FLASH.
108         ; otherwise;  1)  erase the last sector of the FLASH,
109         ;             2)  and write the checksum to the FLASH.
110         ;-------------------------------------------------------------------
111         ;-- Copy the last sector of the FLASH into external RAM (if RAM available)
112         ;-------------------------------------------------------------------
113
114         ;---------- Erase last sector of FLASH ----------
115    P:000123 44F400     move   #$AAAAAA,x0
               AAAAAA
116    P:000125 077084     move   x0,p:FMemStart+$5555 ; Unlock FLASH cycle 1
               C05555
117
118    P:000127 44F400     move   #$555555,x0
               555555
119    P:000129 077084     move   x0,p:FMemStart+$2AAA ; Unlock FLASH cycle 2
               C02AAA
120
121    P:00012B 44F400     move   #$808080,x0
               808080
122    P:00012D 077084     move   x0,p:FMemStart+$5555; Send set-up command
               C05555
123
124    P:00012F 44F400     move   #$AAAAAA,x0
               AAAAAA
125    P:000131 077084     move   x0,p:FMemStart+$5555 ; Unlock FLASH cycle 1
               C05555
126
127    P:000133 44F400     move   #$555555,x0
               555555
128    P:000135 077084     move   x0,p:FMemStart+$2AAA ; Unlock FLASH cycle 2
               C02AAA
129
130    P:000137 44F400     move   #$303030,x0
               303030
131    P:000139 077084     move   x0,p:FMemEnd-$800 ; Send sector erase command
               C7F800
132                                             ; and specify sector to erase
133    P:00013B 44F400     move   #$FFFFFF,x0
               FFFFFF
134                 _wait_til_erased
135    P:00013D 07F08E     move   p:FMemEnd-$800,a ; Get value of location in last
sector
               C7F800
136    P:00013F 200045     cmp    x0,a             ; Fully Erased?
137    P:000140 0527DD     bne    _wait_til_erased ; No
138                                             ; Yes
139         ;-------------------------------------------------------------------
140         ; --- Write the last sector Data back to the FLASH   (if RAM available) ---
141         ;-------------------------------------------------------------------
142
143         ;---------- FLASH Write Routine ----------
```

```
       144           ;  b  = contains data to be written to FLASH.
       145           ;  r0 = points to location in FLASH to be written.
       146          _write_checksum
       147    P:000141 44F400     move   #$AAAAAA,x0
                       AAAAAA
       148    P:000143 077084     move   x0,p:FMemStart+$5555    ; Unlock FLASH cycle 1
                       C05555
       149
       150    P:000145 44F400     move   #$555555,x0
                       555555
       151    P:000147 077084     move   x0,p:FMemStart+$2AAA    ; Unlock FLASH cycle 2
                       C02AAA
       152
       153    P:000149 44F400     move   #$A0A0A0,x0
                       A0A0A0
       154    P:00014B 077084     move   x0,p:FMemStart+$5555    ; Send FLASH write command
                       C05555
       155
       156    P:00014D 07608F     move   b,p:(r0)                ; Write Data to FLASH
       157
       158          _write_wait
       159    P:00014E 07E084     move   p:(r0),x0     ; Get current FLASH value at write
location
       160    P:00014F 20004D     cmp    x0,b          ; Write Done?
       161    P:000150 0527DE     bne    _write_wait   ; No
       162                                             ; Yes
       163          _done
       164    P:000151 050C00     bra    *             ; DONE, do a dynamic HALT
       165
       166
       167                         end     flash1

       0    Errors
       0    Warnings
```

5.2 128K × 24-bit Boot, Program, X and Y Data Flash

In this example, three AMD Am29F040 devices serve as a bank of 512K × 24-bit words of Flash memory divided into four areas: 128K words boot memory, 128K words program memory, 128K words X data, and 128K words Y data. See **Figure 8** for the boot memory map, **Figure 9** for the user memory map, **Figure 10** for the block diagram, and **Table 4** for the association of AA0 and AA1 with the memory spaces.

Since the Flash memory is present in the DSP program space, program control can be turned over to the program in Flash memory at boot time. This allows an embedded application to boot and run from Flash memory so that the DSP internal 1K cache can help speed repeated program functions. Also, since X and Y data space storage is available in Flash memory, program and data variables can be stored in the Flash bank for use after boot.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 70 nS Flash, six wait states are required. This 5 V device is organized as 512K × 8-bits with a 70 nS access time. Three memory devices compose the 24-bit memory bus.

Level conversion between 3.3 V and 5 V is necessary on the 24-bit data bus to accommodate the 5 V Flash memory devices. This is accomplished using three Quality Semiconductor QS3245 QuickSwitch® 8-bit Bus Switches. These switches allow the connection of 3.3 V CMOS logic (the DSP data bus) on one side and 5 V TTL-compatible logic (the memory devices) on the other side, effectively providing 3.3 V to 5 V level conversion. The propagation delay is 0.25 nS, which is not significant in this design.

During DSP reset with Mode 0 selected, the DSP starts fetching instructions from external memory location P:$C00000. Since the DSP56303 device uses only 18 address lines, A0–A17, to select external memory, address P:$C00000 appears as P:$000000 on the external address bus. Therefore, the Flash memory is configured to respond to all external memory requests and data aliasing occurs at every 256K boundary if no additional address decoding is provided. Since the 512K Flash requires nineteen address lines to select all of its 512K memory locations, Flash address line A18 is controlled by Address Attribute Line 1 (AA1). See the memory space selection chart in **Table 4**, the schematic in **Figure 13** and the program listed in **Example 2**. The Flash address line A17, which is controlled by Address Attribute Line 0 (AA0), selects the Flash for use during X data or Y data accesses. Using these two address attribute lines, the 512K × 24-bit Flash memory bank can be segmented into four regions of 128K × 24-bits, as shown in **Table 4**. The program listed in **Example 2** initializes the Address Attribute registers, calculates a 24-bit checksum value for each of the four 128K Flash memory spaces, and programs the checksum value of each space into the last location in each of the four 128K Flash memory spaces.

Table 4. Association of AA0 and AA1 with Memory Spaces

AA1 Flash A18	AA0 Flash A17	DSP Address Space	Address Range
0	0	Y data	Y:$100000–$1FFFFF
0	1	X data	X:$100000–$1FFFFF
1	0	Program	P:$100000–$1FFFFF
1	1	Program Boot	P:$C00000–$C1FFFF (see Note)
Aliased from $0–$FFFFFF at every fourth 128K page boundary while AA0 and AA1 are unconfigured.			

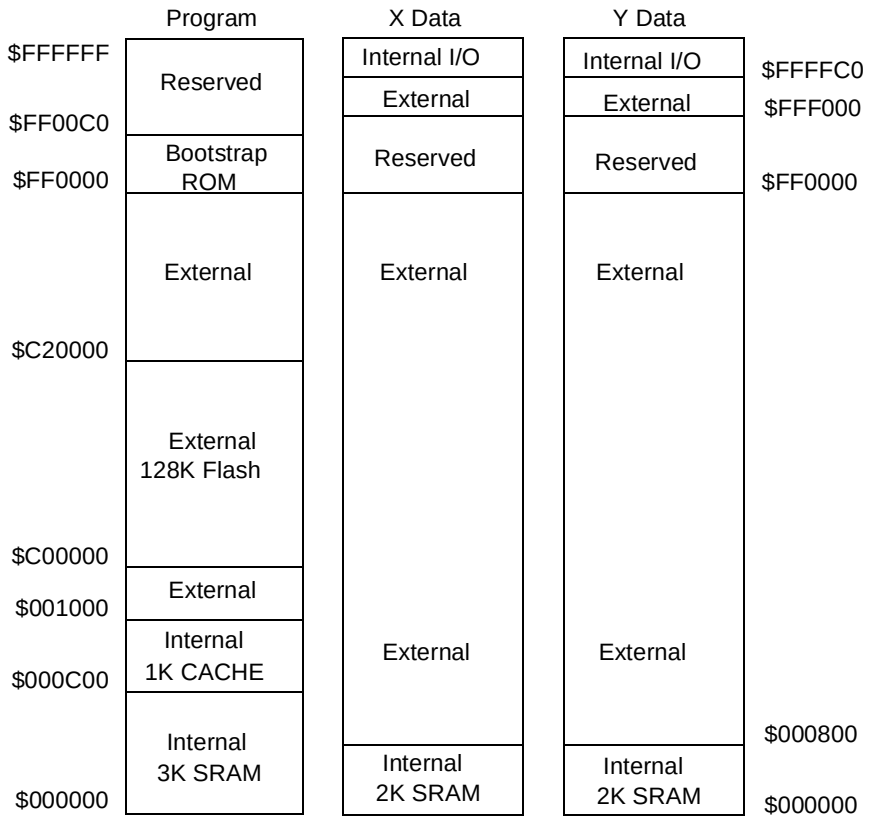

Figure 8. 128K × 24-bit Boot Flash Memory Map

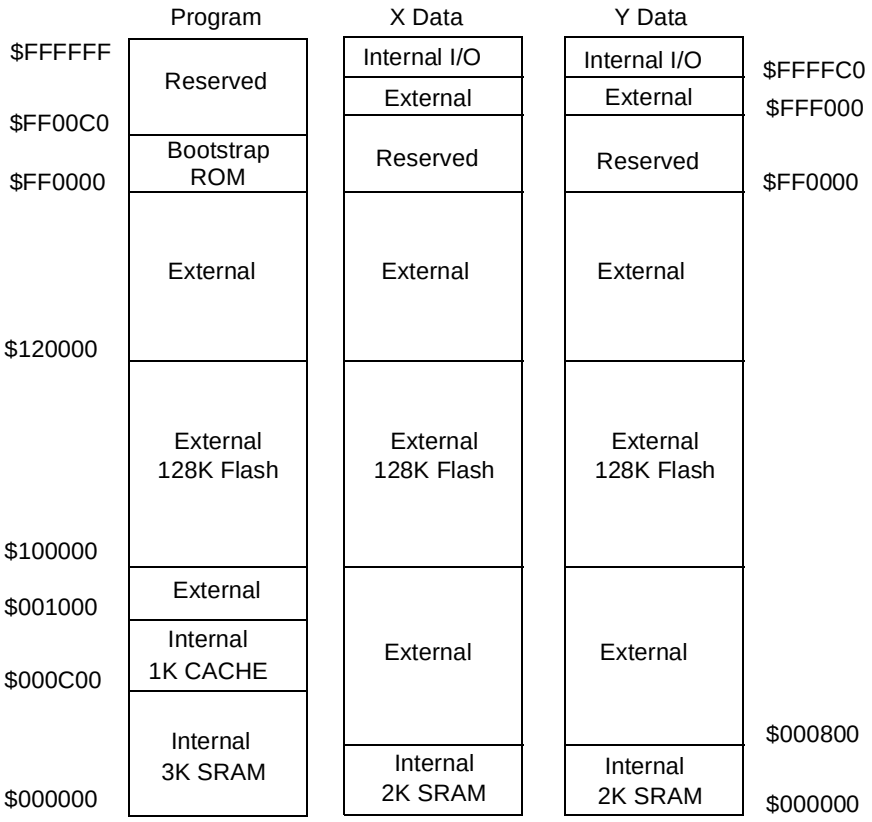

Figure 9. 128K × 24-bit Boot, Program, X Data, Y Data Flash Memory Map

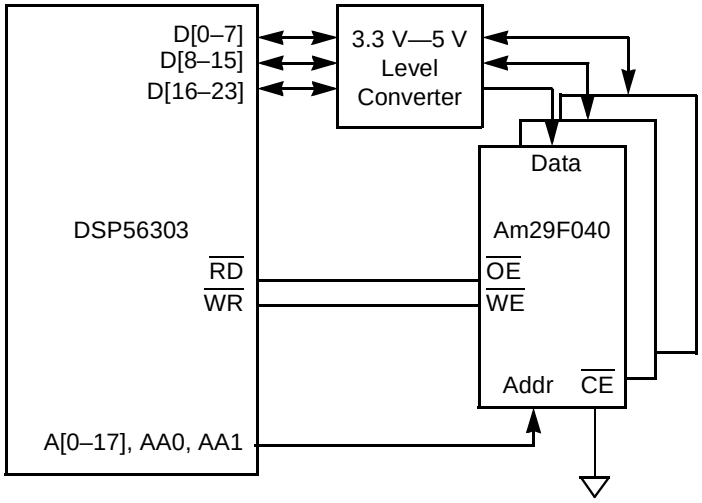

Figure 10. 128K × 24-bit Boot, Program, X Data, Y Data Flash Block Diagram

5.2.1 Flash Timing Requirements

Following are the timing requirements for the Am29F040-70 512K × 8-bit 70 nS Flash. **Table 5** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 11**.

Table 5. Am29F040-70 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Read Cycle Time	t_{RC}	70 nS	—
Address to Output Delay	t_{ACC}	—	70 nS
Chip Enable to Output Delay	t_{CE}	—	70 nS
Output Enable to Output Delay	t_{OE}	—	30 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0 nS	—

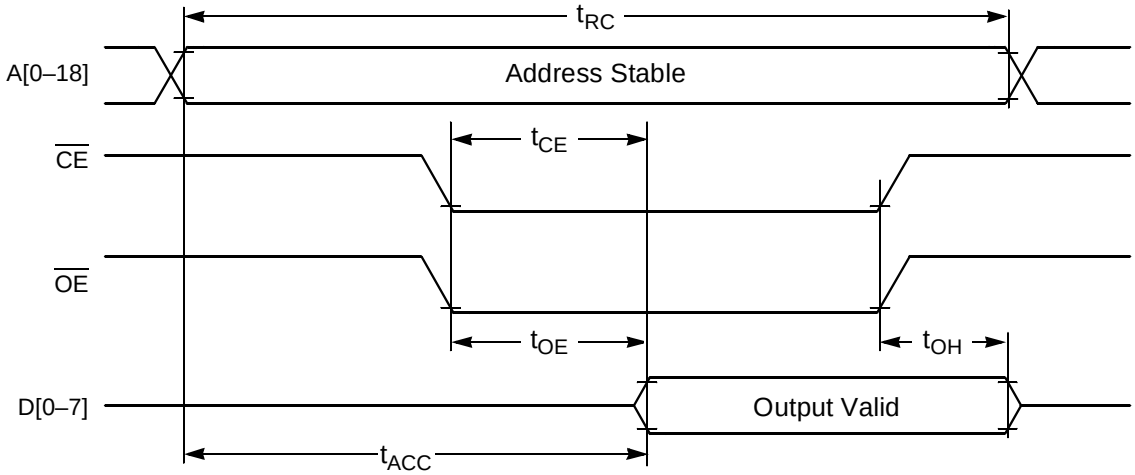

Figure 11. Am29F040 Memory Read Cycle Timing Diagram.

Table 6 shows the memory write timing specification values for the write cycle timing diagram, **Figure 12**.

Table 6. Am29F040-70 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write Cycle Time	t_{WC}	70 nS	—
Address set-up Time	t_{AS}	0 nS	—
$\overline{CE}$ set-up Time	t_{CS}	0 nS	—
Write Pulse Width	t_{WP}	30 nS	—
Data Set-up Time	t_{DS}	30 nS	—
Data Hold Time	t_{DH}	0 nS	—

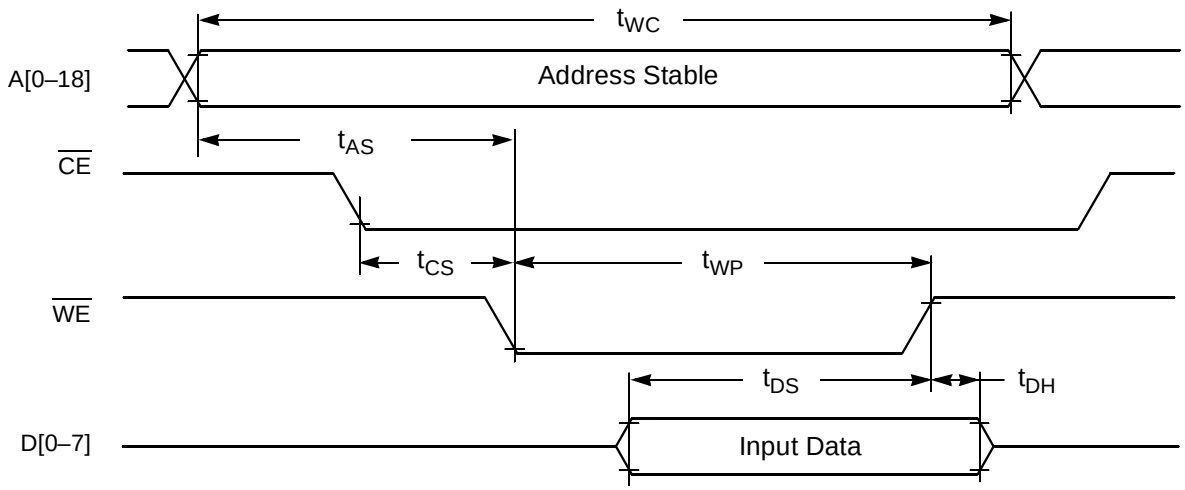

Figure 12. Am29F040 Memory Write Cycle Timing Diagram.

You can program a non-volatile memory location in the Flash memory if the location has not been written with a zero in any of the eight bits. However, if the memory location has been written, you must first erase it. The Am29F040 device is organized as eight sectors of 64 KB each, and to erase a memory location, you must erase the sector in which the memory location resides.

To write to an erased memory location, write the following data sequence to the Flash memory:

1. Write $AAAAAA to location $5555 relative to the Flash memory.
2. Write $555555 to location $2AAA relative to the Flash memory.
3. Write $A0A0A0 to location $5555 relative to the Flash memory.
4. Write 24-bit data to Address in the Flash memory.
5. Read Address until data read = data written.

To erase a sector, write the following data sequence to the Flash memory:

1. Write $AAAAAA to location $5555 relative to the Flash memory.
2. Write $555555 to location $2AAA relative to the Flash memory.
3. Write $808080 to location $5555 relative to the Flash memory.
4. Write $AAAAAA to location $5555 relative to the Flash memory.
5. Write $555555 to location $2AAA relative to the Flash memory.
6. Write $303030 to Sector Address Location relative to the Flash memory.
7. Read location in erasing sector until data read = $FFFFFF.

5.2.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 128K × 24-bit program, X data, and Y data space memory configuration, set the core speed of the DSP for optimum processor and memory performance using the DSP PLL and Clock Generation (PCTL) register. For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal.

The PCTL register value combines the following bits for each feature:

- Desired Core Frequency = 80 MHz
- Given the External Frequency = 4.000 MHz
- Predivider value = 1, bits 20–23 = $0.
- Low-power Divider value = 1, bits 12–14 = $0.
- VCO Multiplication value = 20, bits 0–11 = $013.
- Crystal less than 200 kHz, bit 15 = 0.
- Disable XTAL drive output, bit 16 = 0.
- PLL runs during STOP, bit 17 = 1.
- Enable PLL operation, bit 18 = 1.
- Disable core clock output, bit 19 = 1.

The value loaded into the PCTL is $0E0013.

AA1 selects, via Flash memory A18, the external 128K Flash memory bank during accesses in the address range $100000–$11FFFF between program space requests and X data/Y data requests. Configure the memory address space requirements for Address Attribute Pin 1 with Address Attribute Register 1 (AAR1). The AAR1 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 1.
- Activate the AA pin during external program space accesses, bit 3 = 1.
- Activate the AA pin during external X data space accesses, bit 4 = 0.
- Activate the AA pin during external Y data space accesses, bit 5 = 0.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $7.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR1 is $10070D.

AA0 selects, via Flash memory A17, the external 128K Flash memory bank accesses in the address range $100000–$11FFFF between X data space requests and Y data space requests. Configure the memory address space requirements for the Address Attribute Pin 0 using the Address Attribute Register 0 (AAR0). The AAR0 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 1.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the AA pin during external Y data space accesses, bit 5 = 0.

- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $7.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR0 is $100715. The value loaded into AAR2 and AAR3 is $000000.

Select the proper number of wait states for the memory configuration using the Bus Control Register (BCR). The BCR value combines the following bits for each feature:

- Address attribute area 0 wait states, bits 0–4 = $6.
- Address attribute area 1 wait states, bits 5–9 = $6.
- Address attribute area 2 wait states, bits 10–12 = $0.
- Address attribute area 3 wait states, bits 13–15 = $0.
- Default address area wait states, bits 16–20 = $6.
- Bus state status, bit 21 = 0.
- Enable Bus Lock Hold, bit 22 = 0.
- Enable Bus Request Hold, bit 23 =0.

The value loaded into the BCR is $0600C6.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR value combines the following bits for each feature:

- MA-MD bits specify the DSP operating mode, bits 0–3 = $0.
- External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4 = $0.
- Memory Switch Mode bit reconfigures internal memory spaces, bit 7 = $0.
- Transfer Acknowledge Synchronize Select bit selects the synchronization method for the Transfer Acknowledge (TA) pin, bit 11 = $0.
- Bus Release Timing bit selects between a fast and slow bus release of the $\overline{BB}$ pin, bit 12 = $0.
- Address Attribute Priority Disable bit allows the Address Attribute pins, AA0–AA3, to be used in any combination, bit 14 = $1.
- All other OMR bits are selected for their defaults of $0.

The value loaded into the OMR is $004000.

Configure the memory mode of the DSP using the Status Register (SR). The SR value combines the following bits for each feature:

- Sixteen-bit Compatibility mode enables full compatibility with object code written for the DSP56000 family of DSPs, bit 13 = $0.
- Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, bit 19 = $1.
- All other Status Register bits are selected for their defaults of $0.

The value loaded into the SR is $080000, which is the value loaded during reset.

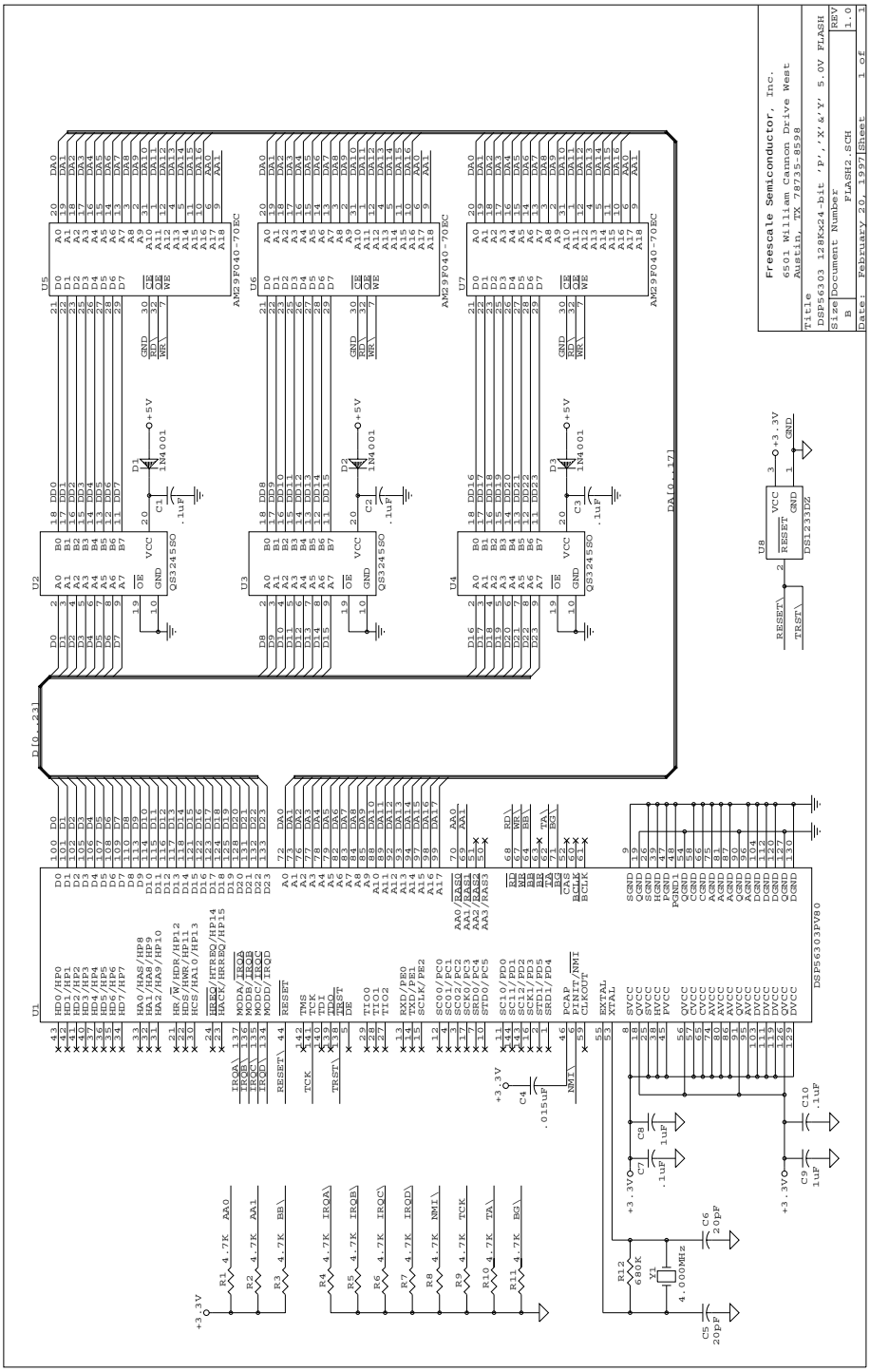

Figure 13. 128K × 24-Bit Boot, Program, X Data, Y Data Flash Schematic

Example 2. 128K x 24-bit BOOT, ‘P’, ‘X’ & ‘Y’ Space FLASH Memory Checksum Verify Program

```
Freescale DSP56300 Assembler Version 6.0.1.6   97-02-23 10:59:24 flash2.asm

1                                       page    132,60,3,3,
2                       ;
3                       ;       flash2.asm, Program to calculate and write a 24-bit Checksum
4                       ;        for a 128K x 24-bit block of BOOT, 128K x 24-bit block
5                       ;        of Program, 128K x 24-bit block of X-Data and
6                       ;        a 128K x 24-bit block of Y-Data memory using a DSP56303.
7                       ;
8                       ;
9                       ;         The program runs in Internal P:RAM to calculate checksum;
10                      ;         on External BOOT:FLASH from $C00000 - $C1FFFF @ 6w/s,
11                      ;         on External P:FLASH from $100000 - $11FFFF @ 6w/s,
12                      ;         on External X:FLASH from $100000 - $11FFFF @ 6w/s and
13                      ;         on External Y:FLASH from $100000 - $11FFFF @ 6w/s.
14                      ;
15
16      C00000          BootStart       equ     $C00000
17      C80000          BootEnd         equ     $C80000
18      080000          BootSize        equ     BootEnd-BootStart ;Last Word is stored Checksum
19
20      100000          PMemStart       equ     $100000
21      180000          PMemEnd         equ     $180000
22      080000          PMemSize        equ     PMemEnd-PMemStart ;Last Word is stored Checksum
23
24      100000          XMemStart       equ     $100000
25      180000          XMemEnd         equ     $180000
26      080000          XMemSize        equ     XMemEnd-XMemStart ;Last Word is stored Checksum
27
28      100000          YMemStart       equ     $100000
29      180000          YMemEnd         equ     $180000
30      080000          YMemSize        equ     YMemEnd-YMemStart ;Last Word is stored Checksum
31
32                      ;--- Program Specific Storage Locations (X DATA SPACE)
33                      CKSUM_CALC_P
34      000000                          equ     $000000 ;P Space Computed Checksum Value
35                      CKSUM_READ_P
36      000001                          equ     $000001 ;P Space Checksum in last memory
Location
37                      CKSUM_CALC_X
38      000002                          equ     $000002 ; X Space Computed Checksum Value
39                      CKSUM_READ_X
40      000003                          equ     $000003 ; X Space Checksum in last memory
Location
41                      CKSUM_CALC_Y
42      000004                          equ     $000004 ; Y Space Computed Checksum Value
43                      CKSUM_READ_Y
44      000005                          equ     $000005 ; Y Space Checksum in Last Memory
Location
45                      CKSUM_CALC_B
46      000006                          equ     $000006 ; BOOT Space Computed Checksum Value
47                      CKSUM_READ_B
48      000007                          equ     $000007 ; BOOT Space Checksum in Last Memory
Location
49
50                      ;--- DSP56303 Control Registers (X I/O SPACE)
51      FFFFFB          BCR             equ     $FFFFFB ; Bus Control Register
52      FFFFFD          PCTL            equ     $FFFFFD ; PLL Control Register
53      FFFFF9          AAR0            equ     $FFFFF9 ; Address Attribute Register 0
```

```
       54     FFFFF8        AAR1          equ    $FFFFF8 ; Address Attribute Register 1
       55
       56                   ;--- PCTL value = 0x0E0013
       57     000000        prediv        equ    0         ; Pre-Divider = 1
       58     000000        lowdiv        equ    0         ; Low Power Divider = 1
       59     000013        pllmul        equ    19        ; VCO Mult = 20;
(19+1)*4.00MHz=80.00MHz
       60     000000        crystal       equ    0         ; No, Crystal not less than 200kHz
       61     000000        disXTAL       equ    0         ; No, do not disable crystal use
       62     020000        pllstop       equ    $020000   ; Yes, PLL runs during STOP
       63     040000        enpll         equ    $040000   ; Yes, enable PLL operation
       64     080000        disclk        equ    $080000   ; Yes, disable CORE clock output
       65     0E0013        PCTL_value    equ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk
       66
       67                   ;--- AAR1 value = 0x10070D
       68     000001        acctype1      equ    1         ; External Memory access type = 0x1
       69     000001        aahigh1       equ    1         ; Enable AA1 pin high when selected
       70     000008        aap1          equ    $8        ; Yes, Enable AA1 pin on ext P access
       71     000000        aax1          equ    0         ; No, Enable AA1 pin on ext X access
       72     000000        aay1          equ    0         ; No, Enable AA1 pin on ext Y access
       73     000000        aswap1        equ    0         ; No, Enable address bus swap
       74     000000        enpack1       equ    0         ; No, Enable packing/unpacking logic
       75     000700        nadd1         equ    $000700   ; Compare 7 address bits
       76     100000        msadd1        equ    $100000   ; Most significant part of address,
       77                                                  ; $100000 - 11ffff, to compare.
       78                                                  ;(0001,000x,xxxx,xxxx,xxxx,xxxx)
       79     10070A        AAR1_value equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       80
       81                                                  ;--- AAR0 value = 0x100715
       82     000001        acctype       equ    1         ; External Memory access type = 0x1
       83     000004        aahigh        equ    $4        ; Enable AA0 pin High when selected
       84     000000        aap           equ    0         ; No, Enable AA0 pin on ext P access
       85     000010        aax           equ    $10       ; Yes, Enable AA0 pin on ext X access
       86     000000        aay           equ    0         ; No, Enable AA0 pin on ext Y access
       87     000000        aswap         equ    0         ; No, Enable address bus swap
       88     000000        enpack        equ    0         ; No, Enable packing/unpacking logic
       89     000700        nadd          equ    $000700   ; Compare 7 address bits
       90     100000        msadd         equ    $100000   ; Most significant part of address,
       91                                                  ; $100000 - 11ffff, to compare.
       92                                                  ; (0001,000x,xxxx,xxxx,xxxx,xxxx)
       93     100715        AAR0_value    equ
acctype+aahigh+aap+aax+aay+aswap+enpack+nadd+msadd
       94
       95              ;--- BCR value = 0x0600C6
       96     000006        aaa0ws        equ    $6        ; Address Attribute Area 0 w/s = 6
       97     0000C0        aaa1ws        equ    $C0       ; Address Attribute Area 1 w/s = 6
       98     000000        aaa2ws        equ    0         ; Address Attribute Area 2 w/s = 0
       99     000000        aaa3ws        equ    0         ; Address Attribute Area 3 w/s = 0
      100     060000        defws         equ    $060000   ; Default Address Area w/s = 6
      101     000000        busss         equ    0         ; Bus state status = 0
      102     000000        enblh         equ    0         ; Enable Bus Lock Hold = 0
      103     000000        enbrh         equ    0         ; Enable Bus Request Hold = 0
      104     0600C6        BCR_value     equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
      105
      106              ;---------------------------------------------------------------------
      107     P:000100                    org    p:$100    ;Keep program in internal RAM
      108
```

```
       109           flash2
       110
       111           ;---------- Initialization Section ----------
       112    P:000100 08F4BD     movep         #PCTL_value,x:PCTL ; Set PLL Control Register
                       0E0013
       113    P:000102 05F43A     movec         #$004000,OMR     ; Disable Address Attribute
Priority
                       004000
       114    P:000104 05F439     movec         #$080000,SR      ; Enable 1K Cache
                       080000
       115    P:000106 08F4BB     movep         #BCR_value,x:BCR ; Set external wait states
                       0600C6
       116
       117           ;---------------------------------------------------------------------
       118           ;---------------------------------------------------------------------
       119           do_b_checksum
       120    P:000108 08F4B9     movep         #>$0,x:AAR0      ; Clear Address Attribute Reg0
                       000000
       121    P:00010A 08F4B8     movep         #>$0,x:AAR1      ; Clear Address Attribute Reg1
                       000000
       122
       123    P:00010C 05F420     move          #-1,m0           ; Set LINEAR addressing mode
                       FFFFFF
       124    P:00010E 60F400     move          #BootStart,r0    ; Set Starting Address of
BOOT:FLASH
                       C00000
       125    P:000110 70F400     move          #BootSize,n0     ; Set to Size of BOOT:Flash
                       080000
       126
       127    P:000112 200013     clr           a
       128    P:000113 20001B     clr           b
       129    P:000114 560600     move          a,x:CKSUM_CALC_B ;Initialize computed checksum
-> $000000
       130    P:000115 560700     move          a,x:CKSUM_READ_B ;Initialize read checksum ->
$000000
       131
       132                 ;---------- Compute the 24-bit BOOT: Space Checksum ----------
       133    P:000116 06D810     dor           n0,b_loop        ;
                       000003
       134    P:000118 07D88E     move          p:(r0)+,a ; Get BOOT:FLASH location Value
       135    P:000119 200018     add           a,b              ; Compute checksum
       136                 b_loop
       137
       138    P:00011A 07E08E     move          p:(r0),a         ;Get Checksum from BOOT:FLASH
       139    P:00011B 570600     move          b,x:CKSUM_CALC_B ;Save Computed Checksum value
       140    P:00011C 20001B     clr           b                ;Clear b0, b1 and b2
       141    P:00011D 560700     move          a,x:CKSUM_READ_B ;Save the Read Checksum value
       142    P:00011E 578600     move          x:CKSUM_CALC_B,b ;Put Calculated Checksum in b1
       143    P:00011F 20000D     cmp           a,b              ; Old Checksum = New Checksum?
       144    P:000120 0D104A     beq           b_done           ; Yes
                       000035
       145                                                      ; No
       146                 ;---------- See if Checksum Location is Erased ----------
       147    P:000122 44F400     move   #$FFFFFF,x0   ; FLASH erased value
                       FFFFFF
       148    P:000124 200045     cmp    x0,a          ; Contents of checksum location =
Erased?
       149    P:000125 0D104A     beq    b_write_checksum; Yes, write new checksum to Flash
                       000020
       150                                             ; No
```

```
151          ;---------------------------------------------------------------------
152          ;                       N O T E
153          ;---------------------------------------------------------------------
154          ; If at least 64Kx24-bit words of external RAM is available in the system
155          ; then; 1)  the last sector of the FLASH can be read into external RAM
156          ;       2)  the last sector of the FLASH can be erased,
157          ;       3)  and the last sector of data along with the new checksum
158          ;           can be written back into the FLASH.
159          ; otherwise;  1)  erase the last sector of the FLASH,
160          ;             2)  and write the checksum to the FLASH.
161          ;---------------------------------------------------------------------
162          ;-- Copy the last sector of FLASH into external RAM (if RAM available) --
163          ;---------------------------------------------------------------------
164
165          ;---------- Erase last sector of FLASH ----------
166     P:000127 44F400     move   #$AAAAAA,x0
                 AAAAAA
167     P:000129 4C7000     move   x0,y:BootStart+$5555    ; Unlock BOOT:FLASH cycle 1
                 C05555
168
169     P:00012B 44F400     move   #$555555,x0
                 555555
170     P:00012D 4C7000     move   x0,y:BootStart+$2AAA    ; Unlock BOOT:FLASH cycle 2
                 C02AAA
171
172     P:00012F 44F400     move   #$808080,x0
                 808080
173     P:000131 4C7000     move   x0,y:BootStart+$5555    ; Send set-up command
                 C05555
174
175     P:000133 44F400     move   #$AAAAAA,x0
                 AAAAAA
176     P:000135 4C7000     move   x0,y:BootStart+$5555    ; Unlock BOOT:FLASH cycle 1
                 C05555
177
178     P:000137 44F400     move   #$555555,x0
                 555555
179     P:000139 4C7000     move   x0,y:BootStart+$2AAA    ; Unlock BOOT:FLASH cycle 2
                 C02AAA
180
181     P:00013B 44F400     move   #$303030,x0
                 303030
182     P:00013D 077084     move   x0,p:BootEnd-$800       ; Send sector erase command
                 C7F800
183                                                        ; and select sector to erase
184     P:00013F 44F400     move   #$FFFFFF,x0
                 FFFFFF
185                  b_wait_til_erased
186     P:000141 07F08E     move   p:BootEnd-$800,a; Get current value of location in
last sector
                 C7F800
187     P:000143 200045     cmp    x0,a                    ; Fully Erased?
188     P:000144 0527DD     bne    b_wait_til_erased       ; No
189                                                        ; Yes
190
191                  ;---------- Write Data Buffer Back to FLASH (if RAM available) ----
192
193                  ;---------- Write BOOT:FLASH Routine ----------
194                  ;  b  = contains data to be written to FLASH.
195                  ;  r0 = points to location in FLASH to be written.
```

```
       196                b_write_checksum
       197    P:000145 44F400      move   #$AAAAAA,x0
                       AAAAAA
       198    P:000147 4C7000      move   x0,y:BootStart+$5555    ; Unlock BOOT:FLASH cycle 1
                       C05555
       199
       200    P:000149 44F400      move   #$555555,x0
                       555555
       201    P:00014B 4C7000      move   x0,y:BootStart+$2AAA    ; Unlock BOOT:FLASH cycle 2
                       C02AAA
       202
       203    P:00014D 44F400      move   #$A0A0A0,x0
                       A0A0A0
       204    P:00014F 4C7000      move   x0,y:BootStart+$5555    ; Send FLASH write command
                       C05555
       205
       206    P:000151 07608F      move   b,p:(r0)          ; Send data to write to BOOT:FLASH
       207
       208                b_write_wait
       209    P:000152 07E084      move   p:(r0),x0 ; Get BOOT:FLASH value at write location
       210    P:000153 20004D      cmp    x0,b              ; Write Done?
       211    P:000154 0527DE      bne    b_write_wait      ; No
       212                                                  ; Yes
       213                b_done
       214                ;---------------------------------------------------------------------
       215                ;---------------------------------------------------------------------
       216                do_p_checksum
       217    P:000155 08F4B9      movep  #AAR0_value,x:AAR0 ; Set Address Attribute Reg0
                       100715
       218    P:000157 08F4B8      movep  #AAR1_value,x:AAR1 ; Set Address Attribute Reg1
                       10070A
       219
       220    P:000159 05F420      move   #-1,m0            ; Set LINEAR addressing mode
                       FFFFFF
       221    P:00015B 60F400      move   #PMemStart,r0     ; Set Starting Address of P:FLASH
                       100000
       222    P:00015D 70F400      move   #PMemSize,n0      ; Set to Size of P:Flash
                       080000
       223
       224    P:00015F 200013      clr    a
       225    P:000160 20001B      clr    b
       226    P:000161 560000      move   a,x:CKSUM_CALC_P ; Initialize computed checksum ->
$000000
       227    P:000162 560100      move   a,x:CKSUM_READ_P ; Initialize read checksum ->
$000000
       228
       229                ;---------- Compute the 24-bit P: Space Checksum ----------
       230    P:000163 06D810      dor    n0,p_loop
                       000003
       231    P:000165 07D88E      move   p:(r0)+,a         ; Get the P:FLASH location Value
       232    P:000166 200018      add    a,b               ; Compute checksum
       233                p_loop
       234
       235    P:000167 07E08E      move   p:(r0),a          ; Get Checksum from P:FLASH
       236    P:000168 570000      move   b,x:CKSUM_CALC_P ; Save the Computed Checksum value
       237    P:000169 20001B      clr    b                 ; Clear b0, b1 and b2
       238    P:00016A 560100      move   a,x:CKSUM_READ_P ; Save the Read Checksum value
       239    P:00016B 578000      move   x:CKSUM_CALC_P,b ; Put Calculated Checksum value in b1
       240    P:00016C 20000D      cmp    a,b               ; Old Checksum = New Checksum?
       241    P:00016D 0D104A      beq    p_done            ; Yes
```

```
                000035
       242                                          ; No
       243                ;---------- See if Checksum Location is Erased ----------
       244    P:00016F 44F400     move   #$FFFFFF,x0   ; FLASH erased value
                FFFFFF
       245    P:000171 200045     cmp    x0,a          ; Contents of checksum location =
Erased?
       246    P:000172 0D104A     beq    p_write_checksum; Yes, write new checksum value to
FLASH
                000020
       247                                          ; No
       248          ;---------------------------------------------------------------------
       249          ;                       N O T E
       250          ;---------------------------------------------------------------------
       251          ; If at least 64Kx24-bit words of external RAM is available in the system
       252          ; then; 1)  the last sector of the FLASH can be read into external RAM
       253          ;       2)  the last sector of the FLASH can be erased,
       254          ;       3)  and the last sector of data along with the new checksum
       255          ;           can be written back into the FLASH.
       256          ; otherwise;  1)  erase the last sector of the FLASH,
       257          ;              2)  and write the checksum to the FLASH.
       258          ;---------------------------------------------------------------------
       259          ;-- Copy the last sector of FLASH into external RAM (if RAM available) --
       260          ;---------------------------------------------------------------------
       261
       262          ;---------- Erase last sector of FLASH ----------
       263    P:000174 44F400     move   #$AAAAAA,x0
                AAAAAA
       264    P:000176 4C7000     move   x0,y:PMemStart+$5555    ; Unlock P:FLASH cycle 1
                105555
       265
       266    P:000178 44F400     move   #$555555,x0
                555555
       267    P:00017A 4C7000     move   x0,y:PMemStart+$2AAA    ; Unlock P:FLASH cycle 2
                102AAA
       268
       269    P:00017C 44F400     move   #$808080,x0
                808080
       270    P:00017E 4C7000     move   x0,y:PMemStart+$5555    ; Send set-up command
                105555
       271
       272    P:000180 44F400     move   #$AAAAAA,x0
                AAAAAA
       273    P:000182 4C7000     move   x0,y:PMemStart+$5555    ; Unlock P:FLASH cycle 1
                105555
       274
       275    P:000184 44F400     move   #$555555,x0
                555555
       276    P:000186 4C7000     move   x0,y:PMemStart+$2AAA    ; Unlock P:FLASH cycle 2
                102AAA
       277
       278    P:000188 44F400     move   #$303030,x0
                303030
       279    P:00018A 077084     move   x0,p:PMemEnd-$800       ; Send sector erase command
                17F800
       280                                                     ; and select sector to erase
       281    P:00018C 44F400     move   #$FFFFFF,x0
                FFFFFF
       282                p_wait_til_erased
```

```
       283    P:00018E 07F08E     move   p:PMemEnd-$800,a ; Get value of location in last
sector
                17F800
       284    P:000190 200045     cmp    x0,a                    ; Fully Erased?
       285    P:000191 0527DD     bne    p_wait_til_erased       ; No
       286                                                        ; Yes
       287
       288                ;---------- Write Data Buffer Back to FLASH (if RAM available) ----
       289
       290                ;---------- Write P:FLASH Routine ----------
       291                ;  b  = contains data to be written to FLASH.
       292                ;  r0 = points to location in FLASH to be written.
       293                p_write_checksum
       294    P:000192 44F400     move   #$AAAAAA,x0
                AAAAAA
       295    P:000194 4C7000     move   x0,y:PMemStart+$5555    ; Unlock P:FLASH cycle 1
                105555
       296
       297    P:000196 44F400     move   #$555555,x0
                555555
       298    P:000198 4C7000     move   x0,y:PMemStart+$2AAA    ; Unlock P:FLASH cycle 2
                102AAA
       299
       300    P:00019A 44F400     move   #$A0A0A0,x0
                A0A0A0
       301    P:00019C 4C7000     move   x0,y:PMemStart+$5555    ; Send FLASH write command
                105555
       302
       303    P:00019E 07608F     move   b,p:(r0)                ; Send data to write to
P:FLASH
       304
       305                p_write_wait
       306    P:00019F 07E084     move   p:(r0),x0     ; Get current P:FLASH value at write
location
       307    P:0001A0 20004D     cmp    x0,b                    ; Write Done?
       308    P:0001A1 0527DE     bne    p_write_wait            ; No
       309                                                        ; Yes
       310                p_done
       311
;*********************************************************************
       312
;*********************************************************************
       313                do_x_checksum
       314    P:0001A2 05F420     move   #-1,m0           ; Set LINEAR addressing mode
                FFFFFF
       315    P:0001A4 60F400     move   #XMemStart,r0    ; Set Starting Address of X:FLASH
                100000
       316    P:0001A6 70F400     move   #XMemSize,n0     ; Set to Size of X:Flash
                080000
       317
       318    P:0001A8 200013     clr    a
       319    P:0001A9 20001B     clr    b
       320    P:0001AA 560200     move   a,x:CKSUM_CALC_X ; Initialize computed checksum ->
$000000
       321    P:0001AB 560300     move   a,x:CKSUM_READ_X ; Initialize read checksum ->
$000000
       322
       323                ;---------- Compute the 24-bit X: Space Checksum ----------
       324    P:0001AC 06D810     dor    n0,x_loop
                000003
```

```
       325    P:0001AE 56D800     move   x:(r0)+,a        ; Get the X:FLASH location Value
       326    P:0001AF 200018     add    a,b              ; Compute checksum
       327                  x_loop
       328
       329    P:0001B0 56E000     move   x:(r0),a         ; Get Checksum from X:FLASH
       330    P:0001B1 570200     move   b,x:CKSUM_CALC_X ; Save the Computed Checksum value
       331    P:0001B2 20001B     clr    b                ; Clear b0, b1 and b2
       332    P:0001B3 560300     move   a,x:CKSUM_READ_X ; Save the Read Checksum value
       333    P:0001B4 578200     move   x:CKSUM_CALC_X,b ; Put Calculated Checksum value in b1
       334    P:0001B5 20000D     cmp    a,b              ; Old Checksum = New Checksum?
       335    P:0001B6 0D104A     beq    x_done           ; Yes
                      000035
       336                                              ; No
       337                  ;---------- See if Checksum Location is Erased ----------
       338    P:0001B8 44F400     move   #$FFFFFF,x0      ; FLASH erased value
                      FFFFFF
       339    P:0001BA 200045     cmp    x0,a             ; Contents of checksum location =
Erased?
       340    P:0001BB 0D104A     beq    x_write_checksum ; Yes, Go write new checksum value to
FLASH
                      000020
       341                                              ; No
       342                  ;---------------------------------------------------------------
       343                  ;                       N O T E
       344                  ;---------------------------------------------------------------
       345                  ; If there is at least 64Kx24-bit words of external RAM
       346                  ; then 1) last sector of the FLASH can be read into external RAM
       347                  ;       2)  the last sector of the FLASH can be erased,
       348                  ;       3)  and the last sector of data along with the new checksum
       349                  ;           can be written back into the FLASH.
       350                  ; otherwise;  1)  erase the last sector of the FLASH,
       351                  ;             2)  and write the checksum to the FLASH.
       352                  ;---------------------------------------------------------------
       353                  ;-- Copy last sector of FLASH into external RAM (if available) ----
       354                  ;---------------------------------------------------------------
       355
       356                  ;---------- Erase last sector of FLASH ----------
       357    P:0001BD 44F400     move   #$AAAAAA,x0
                      AAAAAA
       358    P:0001BF 4C7000     move   x0,y:XMemStart+$5555; Unlock X:FLASH cycle 1
                      105555
       359
       360    P:0001C1 44F400     move   #$555555,x0
                      555555
       361    P:0001C3 4C7000     move   x0,y:XMemStart+$2AAA; Unlock X:FLASH cycle 2
                      102AAA
       362
       363    P:0001C5 44F400     move   #$808080,x0
                      808080
       364    P:0001C7 4C7000     move   x0,y:XMemStart+$5555 ; Send set-up command
                      105555
       365
       366    P:0001C9 44F400     move   #$AAAAAA,x0
                      AAAAAA
       367    P:0001CB 4C7000     move   x0,y:XMemStart+$5555; Unlock X:FLASH cycle 1
                      105555
       368
       369    P:0001CD 44F400     move   #$555555,x0
                      555555
       370    P:0001CF 4C7000     move   x0,y:XMemStart+$2AAA; Unlock X:FLASH cycle 2
```

```
                102AAA
       371
       372    P:0001D1 44F400     move   #$303030,x0
                303030
       373    P:0001D3 447000     move   x0,x:XMemEnd-$800  ; Send sector erase command
                17F800
       374                                                  ; and select sector to erase
       375    P:0001D5 44F400     move   #$FFFFFF,x0
                FFFFFF
       376                 x_wait_til_erased
       377    P:0001D7 56F000     move   x:XMemEnd-$800,a ; Get value of location in last
sector
                17F800
       378    P:0001D9 200045     cmp    x0,a                    ; Fully Erased?
       379    P:0001DA 0527DD     bne    x_wait_til_erased       ; No
       380                                                       ; Yes
       381
       382                 ;---------- Write Data Buffer Back to FLASH (if RAM available) ----
       383
       384                 ;---------- Write X:FLASH Routine ----------
       385                 ;  b  = contains data to be written to FLASH.
       386                 ;  r0 = points to location in FLASH to be written.
       387                 x_write_checksum
       388    P:0001DB 44F400     move   #$AAAAAA,x0
                AAAAAA
       389    P:0001DD 4C7000     move   x0,y:XMemStart+$5555; Unlock X:FLASH cycle 1
                105555
       390
       391    P:0001DF 44F400     move   #$555555,x0
                555555
       392    P:0001E1 4C7000     move   x0,y:XMemStart+$2AAA; Unlock X:FLASH cycle 2
                102AAA
       393
       394    P:0001E3 44F400     move   #$A0A0A0,x0
                A0A0A0
       395    P:0001E5 4C7000     move   x0,y:XMemStart+$5555; Send FLASH write command
                105555
       396
       397    P:0001E7 576000     move   b,x:(r0)         ; Send data to write to X:FLASH
       398
       399                 x_write_wait
       400    P:0001E8 44E000     move   x:(r0),x0     ; Get X:FLASH value at write location
       401    P:0001E9 20004D     cmp    x0,b                    ; Write Done?
       402    P:0001EA 0527DE     bne    x_write_wait            ; No
       403                                                       ; Yes
       404                 x_done
       405               ;*********************************************************************
       406               ;*********************************************************************
       407                 do_y_checksum
       408    P:0001EB 05F420     move   #-1,m0           ; Set LINEAR addressing mode
                FFFFFF
       409    P:0001ED 60F400     move   #YMemStart,r0    ; Set Starting Address of Y:FLASH
                100000
       410    P:0001EF 70F400     move   #YMemSize,n0     ; Set to Size of Y:Flash
                080000
       411
       412    P:0001F1 200013     clr    a
       413    P:0001F2 20001B     clr    b
       414    P:0001F3 560400     move   a,x:CKSUM_CALC_Y ; Initialize computed checksum ->
$000000
```

```
       415    P:0001F4 560500     move   a,x:CKSUM_READ_Y ; Initialize read checksum ->
$000000
       416
       417                ;---------- Compute the 24-bit Y: Space Checksum ----------
       418    P:0001F5 06D810     dor    n0,y_loop
                       000003
       419    P:0001F7 5ED800     move   y:(r0)+,a        ;Get Y:FLASH location value
       420    P:0001F8 200018     add    a,b              ; Compute checksum
       421                y_loop
       422
       423    P:0001F9 5EE000     move   y:(r0),a         ; Get Checksum from Y:FLASH
       424    P:0001FA 570400     move   b,x:CKSUM_CALC_Y ;Save computed checksum
       425    P:0001FB 20001B     clr    b                ;Clear b0, b1 and b2
       426    P:0001FC 560500     move   a,x:CKSUM_READ_Y ;Save Read Checksum value
       427    P:0001FD 578400     move   x:CKSUM_CALC_Y,b ;Put calculated checksum value in b1
       428    P:0001FE 20000D     cmp    a,b              ;Old Checksum=New Checksum?
       429    P:0001FF 0D104A     beq    y_done           ; Yes
                       000035
       430                                             ; No
       431                ;---------- See if Checksum Location is Erased --------
       432    P:000201 44F400     move   #$FFFFFF,x0   ; FLASH erased value
                       FFFFFF
       433    P:000203 200045     cmp    x0,a          ; Contents of checksum location =
Erased?
       434    P:000204 0D104A     beq    y_write_checksum; Yes, Go write new checksum value to
FLASH
                       000020
       435                                          ; No
       436                ;---------------------------------------------------------------
       437                ;                        N O T E
       438                ;---------------------------------------------------------------
       439                ; If there is at least 64Kx24-bit words of external RAM
       440                ; then 1) last sector of FLASH can be read into RAM
       441                ;       2)  the last sector of the FLASH can be erased,
       442                ;       3)  and last sector of data with new checksum
       443                ;           can be written back into the FLASH.
       444                ; otherwise;  1)  erase the last sector of the FLASH,
       445                ;             2)  and write the checksum to the FLASH.
       446                ;---------------------------------------------------------------
       447                ;-- Copy last sector of FLASH into external RAM (if RAM available)
       448                ;---------------------------------------------------------------
       449
       450                ;---------- Erase last sector of FLASH ----------
       451    P:000206 44F400     move   #$AAAAAA,x0
                       AAAAAA
       452    P:000208 4C7000     move   x0,y:YMemStart+$5555; Unlock Y:FLASH cycle 1
                       105555
       453
       454    P:00020A 44F400     move   #$555555,x0
                       555555
       455    P:00020C 4C7000     move   x0,y:YMemStart+$2AAA; Unlock Y:FLASH cycle 2
                       102AAA
       456
       457    P:00020E 44F400     move   #$808080,x0
                       808080
       458    P:000210 4C7000     move   x0,y:YMemStart+$5555; Send set-up command
                       105555
       459
       460    P:000212 44F400     move   #$AAAAAA,x0
                       AAAAAA
```

```
461    P:000214 4C7000      move   x0,y:YMemStart+$5555; Unlock Y:FLASH cycle 1
               105555
462
463    P:000216 44F400      move   #$555555,x0
               555555
464    P:000218 4C7000      move   x0,y:YMemStart+$2AAA; Unlock Y:FLASH cycle 2
               102AAA
465
466    P:00021A 44F400      move   #$303030,x0
               303030
467    P:00021C 4C7000      move   x0,y:YMemEnd-$800; Send sector erase command
               17F800
468                                                  ; and select sector to erase
469    P:00021E 44F400      move   #$FFFFFF,x0
               FFFFFF
470                 y_wait_til_erased
471    P:000220 5EF000      move   y:YMemEnd-$800,a; Get value of location in last sector
               17F800
472    P:000222 200045      cmp    x0,a                    ; Fully Erased?
473    P:000223 0527DD      bne    y_wait_til_erased       ; No
474                                                        ; Yes
475
476                 ;---------- Write Data Buffer Back to FLASH (if RAM available) ----
477
478                 ;---------- Write Y:FLASH Routine ----------
479                 ;  b  = contains data to be written to FLASH.
480                 ;  r0 = points to location in FLASH to be written.
481                 y_write_checksum
482    P:000224 44F400      move   #$AAAAAA,x0
               AAAAAA
483    P:000226 4C7000      move   x0,y:YMemStart+$5555; Unlock Y:FLASH cycle 1
               105555
484
485    P:000228 44F400      move   #$555555,x0
               555555
486    P:00022A 4C7000      move   x0,y:YMemStart+$2AAA; Unlock Y:FLASH cycle 2
               102AAA
487
488    P:00022C 44F400      move   #$A0A0A0,x0
               A0A0A0
489    P:00022E 4C7000      move   x0,y:YMemStart+$5555;Send FLASH write command
               105555
490
491    P:000230 5F6000      move   b,y:(r0)         ; Send data to write to Y:FLASH
492
493                 y_write_wait
494    P:000231 4CE000      move   y:(r0),x0     ; Get Y:FLASH value at write location
495    P:000232 20004D      cmp    x0,b                    ; Write Done?
496    P:000233 0527DE      bne    y_write_wait            ; No
497                                                        ; Yes
498                 y_done
499
;*********************************************************************
500
501    P:000234 050C00      bra    *                       ; Done, Do dynamic Halt
502
503                         end     flash2

0    Errors
0    Warnings
```

5.3 128K × 16-Bit X Data and Y Data Flash Example

The 128K × 16-bit X data and 128K × 16-bit Y data memory space Flash implementation uses the AMD Am29F400-150 device. See **Figure 14** for the memory map layout and **Figure 15** for the block diagram. Sixteen-bit coefficient and data arrays are stored externally in non-volatile storage, allowing results from previous operations to be stored before power is removed and to be recalled on power up.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 150 nS Flash memory, twelve wait states are required. This 5 V device is organized as 256K × 16-bit words with a 150 nS access time. One memory device is used to achieve the 16-bit memory bus.

Level conversion to and from 3.3 V and 5 V is necessary on the 16-bit data bus to accommodate the 5 V Flash memory devices. This is accomplished using two Quality Semiconductor QS3245 QuickSwitch® 8-bit Bus Switches. These switches allow the connection of 3.3 V CMOS logic, DSP data bus, on one side and 5 V TTL-compatible logic, memory devices, on the other side, effectively providing a 3.3 V-to-5 V level conversion. The 0.25 nS propagation delay is not significant. **Figure 18** shows the schematic for this example.

Address Attribute Line 1 is configured to select the Flash memory device when an X data or Y data access is requested in the address range $100000–$11FFFF. AA0 is configured to select the 256K Flash memory bank between 128K of X data space and 128K of Y data space. The Address Attribute implementation details are presented in **Section 5.3.2**, *DSP56303 Port A Timing Requirements and Register Settings*, on page 41 and in the program listing in **Example 3** on page 44.

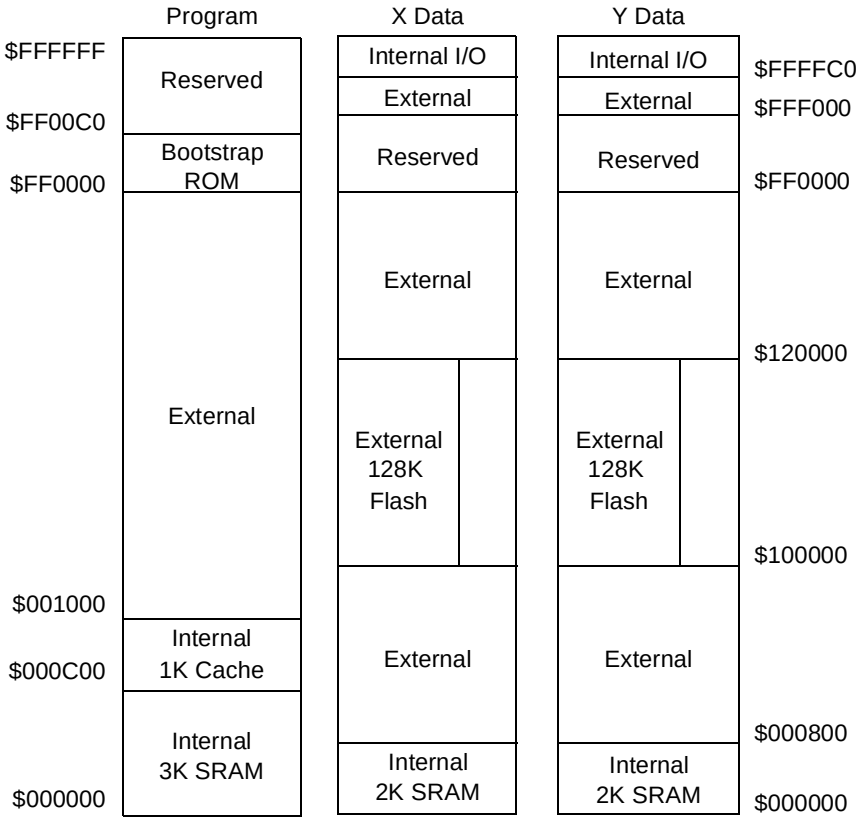

Figure 14. 128K × 16 X Data and Y Data Memory Map

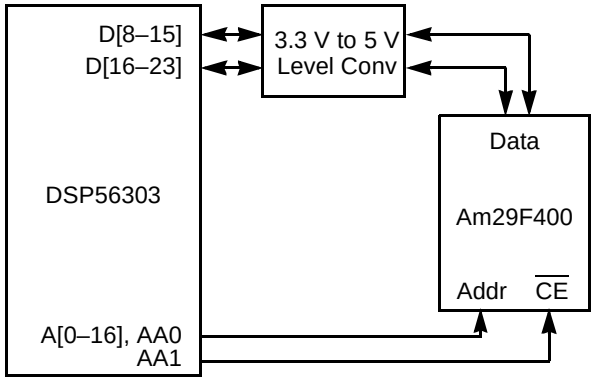

Figure 15. 128K × 16-Bit X Data and Y Data Flash memory Block Diagram

5.3.1 Flash Memory Timing Requirements

Following are the timing requirements for the Am29F400-150 256K × 16-bit 150nS Flash memory. **Table 7** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 16**.

Table 7. Am29F400-150 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Read Cycle Time	t_{RC}	150 nS	—
Address to Output Delay	t_{ACC}	—	150 nS
Chip Enable to Output Delay	t_{CE}	—	150 nS
Output Enable to Output Delay	t_{OE}	—	55 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0 nS	—

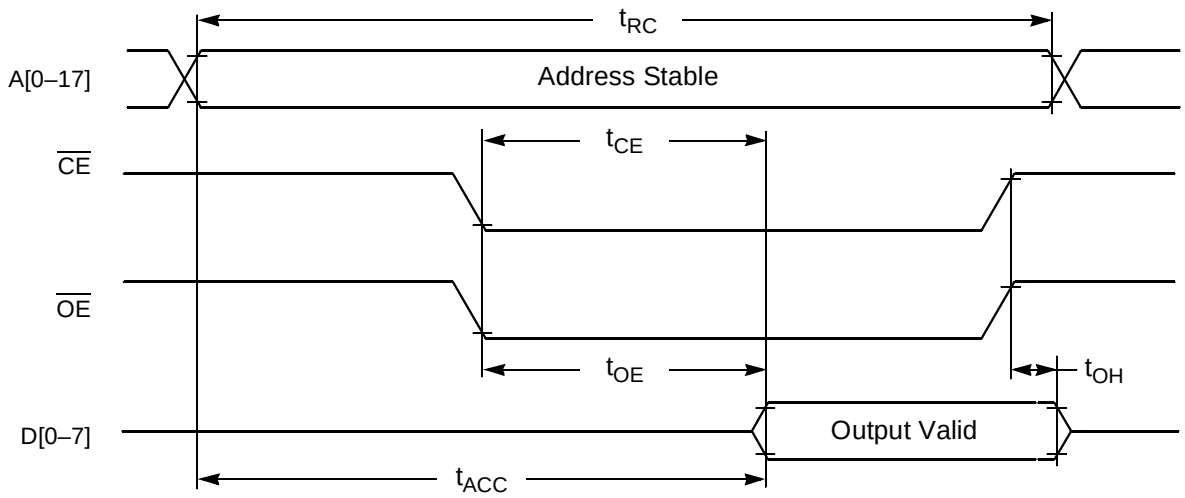

Figure 16. Am29F400 Memory Read Cycle Timing Diagram

Table 8 shows the memory write timing specification values used in the memory write cycle timing diagram, **Figure 17**.

Table 8. AM29F400 Memory Write Cycle Timing Diagram

Write Cycle Parameter	Symbol	Min	Max
Write Cycle Time	t_{WC}	150 nS	—
Address set-up Time	t_{AS}	0 nS	—
$\overline{CE}$ set-up Time	t_{CS}	0 nS	—
Write Pulse Width	t_{WP}	50 nS	—
Data set-up Time	t_{DS}	50 nS	—
Data Hold Time	t_{DH}	0 nS	—

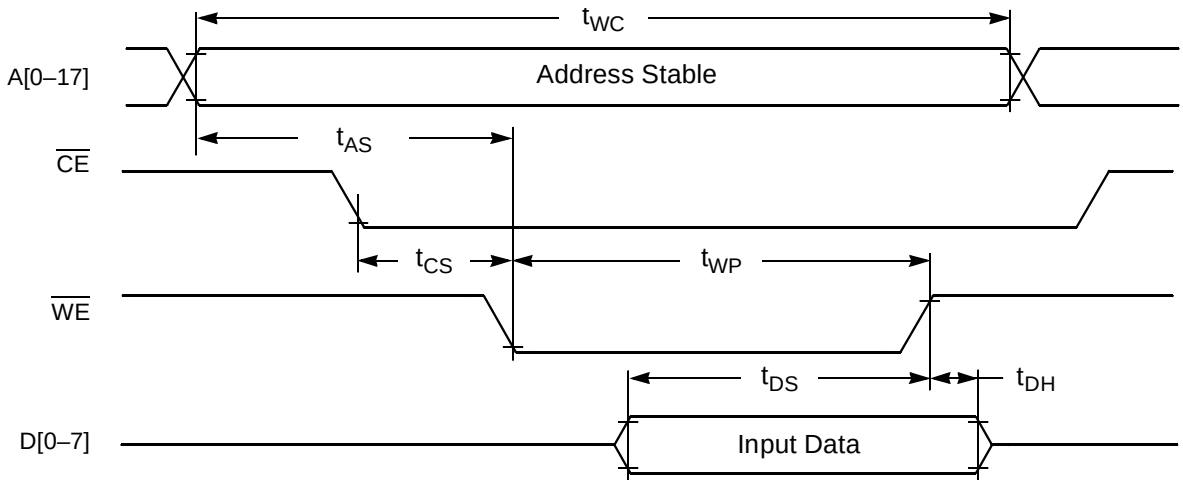

Figure 17. Am29F400 Memory Write Cycle Timing Diagram.

You can program non-volatile memory locations in the Flash memory if the location has not been written with a zero in any of the eight bits. However, if the memory location has been written, then you must first erase it. The Am29F400 device is organized as eight sectors of 64K bytes each, and to erase a memory location, you must erase the sector in which the memory location resides.

To write to an erased memory location, write the following data sequence to the Flash:

1. Write $AAAA00 to location $5555 relative to the Flash memory.
2. Write $555500 to location $2AAA relative to the Flash memory.
3. Write $A0A000 to location $5555 relative to the Flash memory.
4. Write 16-bit data to address in the Flash memory.
5. Read Address until DATA written = DATA read.

To erase a sector, write the following data sequence to the Flash memory:

1. Write $AAAA00 to location $5555 relative to the Flash memory.
2. Write $555500 to location $2AAA relative to the Flash memory.
3. Write $808000 to location $5555 relative to the Flash memory.
4. Write $AAAA00 to location $5555 relative to the Flash memory.
5. Write $555500 to location $2AAA relative to the Flash memory.
6. Write $303000 to Sector Address Location relative to the Flash memory.

7. Read location in erasing Sector until DATA read = $FFFFxx.

   Note: Only the upper 16 bits of the 24-bit data word are significant.

5.3.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 128K × 16-bit X data and Y data space memory configuration, set the core speed of the DSP for optimum processor and memory performance using the DSP PLL and Clock Generation (PCTL) register. For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. The PCTL value combines the following bits for each feature:

- Desired core frequency = 80 MHz
- Given the external frequency = 4.000 MHz
- Predivider value = 1, bits 20–23 = $0.
- Low-power divider value = 1, bits 12–14 = $0.
- VCO multiplication value = 20, bits 0–11 = $013.
- Crystal less than 200 kHz, bit 15 = 0.
- Disable XTAL drive output, bit 16 = 0.
- PLL runs during STOP, bit 17 = 1.
- Enable PLL operation, bit 18 = 1.
- Disable core clock output, bit 19 = 1.

The value loaded into the PCTL is $0E0013.

AA1 enables, via Flash memory $\overline{CE}$, external 128K Flash bank accesses in the address range $100000–$11FFFF during X data and Y data space requests. Configure the memory address space requirements for the AA1 using Address Attribute Register 1 (AAR1). The AAR1 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 0.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the AA pin during external Y data space accesses, bit 5 = 1.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $7.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR1 is $100731.

AA0 switches, via Flash memory A17, between X data and Y data space 128K Flash bank accesses in the address range $100000–$11FFFF during X and Y data space requests. Configure the memory address space requirements for the AA0 using the Address Attribute Register 0 (AAR0).

The AAR0 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 1.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the AA pin during external Y data space accesses, bit 5 = 0.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $7.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR0 is $100715. The value loaded into AAR2 and AAR3 is $000000.

Select the proper number of wait states for the memory configuration using the Bus Control Register (BCR). The BCR value combines the following bits for each feature:

- Address attribute area 0 wait states, bits 0–4 = $C.
- Address attribute area 1 wait states, bits 5–9 = $C.
- Address attribute area 2 wait states, bits 10–12 = $0.
- Address attribute area 3 wait states, bits 13–15 = $0.
- Default address area wait states, bits 16–20 = $0.
- Bus state status, bit 21 = 0.
- Enable Bus Lock Hold, bit 22 = 0.
- Enable Bus Request Hold, bit 23 =0.

The value loaded into the BCR is $00018C.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR value combines the following bits for each feature:

- MA–MD bits specify the DSP operating mode, bits 0–3 = $0.
- External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4 = $0.
- Memory Switch Mode bit reconfigures internal memory spaces, bit 7 = $0.
- Transfer Acknowledge Synchronize Select bit selects the synchronization method for the Transfer Acknowledge ($\overline{\text{TA}}$) pin, bit 11 = $0.
- Bus Release Timing bit selects between a fast and slow bus release of the $\overline{\text{BB}}$ pin, bit 12 = $0.
- Address Attribute Priority Disable bit allows the Address Attribute pins, AA[0–3], to be used in any combination, bit 14 = $1.
- All other OMR bits are selected for their defaults of $0.

The value loaded into the OMR is $004000.

Configure the memory mode of the DSP using the Status Register (SR). The SR value combines the following bits for each feature: Sixteen-bit Compatibility mode enables full compatibility with object code written for the DSP56000 family of DSPs, bit 13 = $0. The Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, bit 19 = $1. All other Status Register bits are selected for their defaults of $0.

The value loaded into the SR is $080000, which is the value loaded during reset.

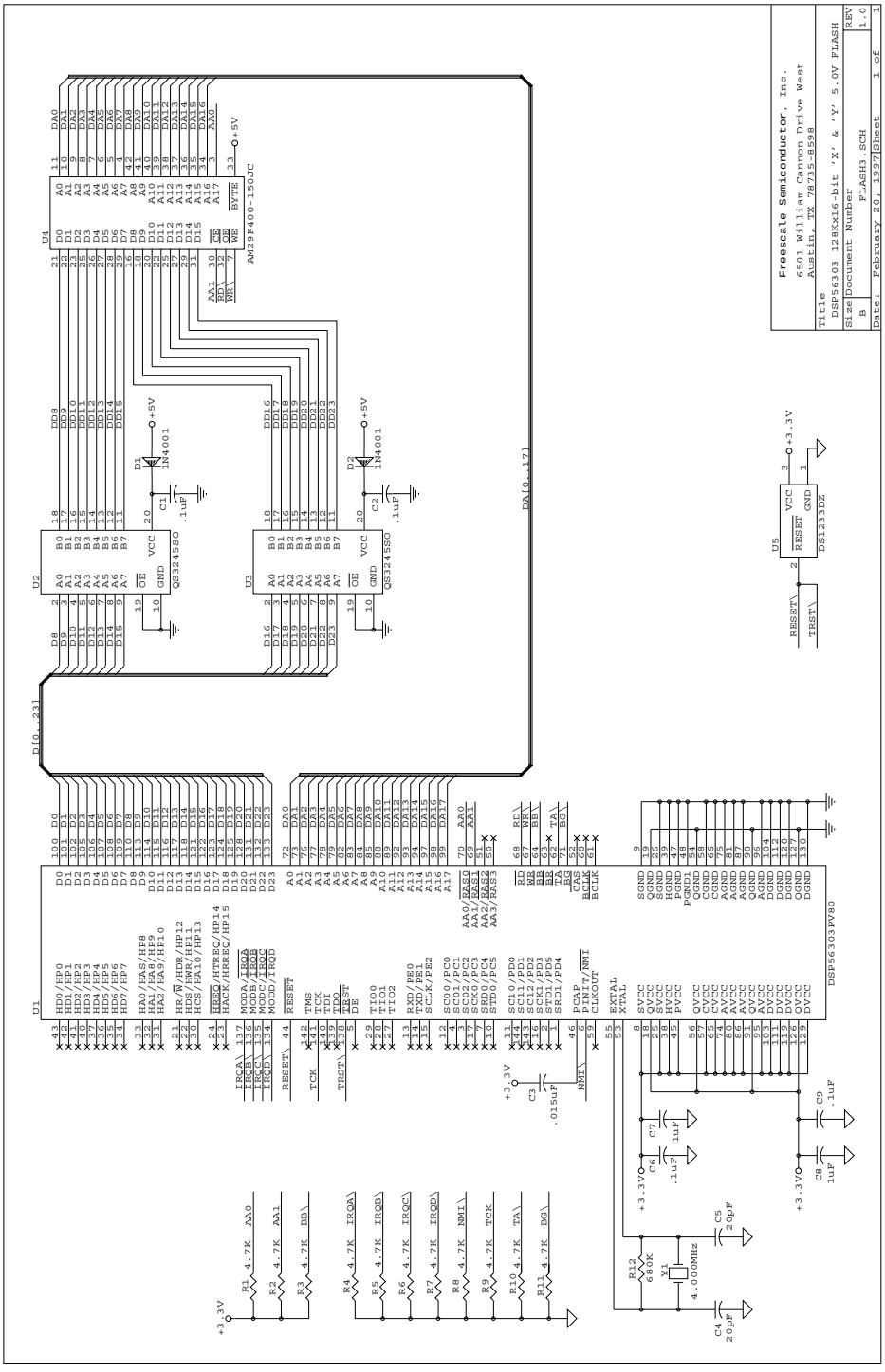

Figure 18. 256K x 16-bit X and Y Data Flash Schematic

Interfacing Flash Memory with the DSP56300 Family of Digital Signal Processors, Rev. 1

Example 3. 256K × 16-Bit X Data and Y Space Flash Memory Checksum Verify Program

```
       Freescale DSP56300 Assembler Version 6.0.1.6   97-02-23 11:05:36 flash3.asm

       1                                    page    132,60,3,3,
       2                    ;
       3                    ; flash3.asm - Simple program to calculate and write a 16-bit
Checksum
       4                    ; for a 256K x 16-bit block of X-Data and a
       5                    ;                   256K x 16-bit block of Y-Data memory using a
DSP56303.
       6                    ;
       7                    ; Program runs in Internal P:RAM to calculate checksum;
       8                    ; on External X:FLASH from $100000-$11FFFF @ 12w/s and
       9                    ; on External Y:FLASH from $100000-$11FFFF @ 12w/s.
       10                   ;
       11
       12     100000        XMemStart     equ    $100000
       13     120000        XMemEnd       equ    $120000
       14     020000        XMemSize      equ    XMemEnd-XMemStart ;Last Word is stored Checksum
       15
       16     100000        YMemStart     equ    $100000
       17     120000        YMemEnd       equ    $120000
       18     020000        YMemSize      equ    YMemEnd-YMemStart ;Last word is stored Checksum
       19
       20                   ;--- Program Specific Storage Locations (X DATA SPACE)
       21                   CKSUM_CALC_X
       22     000000                      equ    $000000 ; X Space Computed Checksum
       23                   CKSUM_READ_X
       24     000001                      equ    $000001 ; X Space Checksum in Last Memory
Location
       25                   CKSUM_CALC_Y
       26     000002                      equ    $000002 ; Y Space Computed Checksum
       27                   CKSUM_READ_Y
       28     000003                      equ    $000003 ; Y Space Checksum in Last Memory
Location
       29
       30                   ;--- DSP56303 Control Registers (X I/O SPACE)
       31     FFFFFB        BCR           equ    $FFFFFB   ; Bus Control Register
       32     FFFFFD        PCTL          equ    $FFFFFD   ; PLL Control Register
       33     FFFFF9        AAR0          equ    $FFFFF9   ; AAR0
       34     FFFFF8        AAR1          equ    $FFFFF8   ; AAR1
       35
       36                   ;--- PCTL value = 0x0E0013
       37     000000        prediv        equ    0         ; Pre-Divider = 1
       38     000000        lowdiv        equ    0         ; Low Power Divider = 1
       39     000013        pllmul        equ    19        ; VCO Mult = 20;
(19+1)*4.00MHz=80.00MHz
       40     000000        crystal       equ    0         ; No, Crystal not less than 200kHz
       41     000000        disXTAL       equ    0         ; No, do not disable crystal use
       42     020000        pllstop       equ    $020000   ; Yes, PLL runs during STOP
       43     040000        enpll         equ    $040000   ; Yes, enable PLL operation
       44     080000        disclk        equ    $080000   ; Yes, disable CORE clock output
       45     0E0013        PCTL_value    equ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk
       46
       47                   ;--- AAR1 value = 0x100731
       48     000001        acctype1      equ    1 ;External Memory access type = 0x1
       49     000000        aahigh1       equ    0 ; Enable AA1 pin low when selected
       50     000000        aap1          equ    0 ; No, Enable AA1 pin on ext P access
       51     000010        aax1          equ    $10    ; Yes, Enable AA1 pin on ext X access
```

```
       52     000020        aay1          equ    $20    ; Yes, Enable AA1 pin on ext Y access
       53     000000        aswap1        equ    0      ; No, Enable address bus swap
       54     000000        enpack1       equ    0      ; No, Enable packing/unpacking logic
       55     000700        nadd1         equ    $000700 ; Compare 7 address bits
       56     100000        msadd1        equ    $100000 ; Most significant portion of address,
       57                                             ; $100000 - 11ffff, to compare.
       58                                             ; (0001,000x,xxxx,xxxx,xxxx,xxxx)
       59     100731        AAR1_value equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       60
       61                   ;--- AAR0 value = 0x100715
       62     000001        acctype       equ    1      ; External Memory access type = 0x1
       63     000004        aahigh        equ    $4     ; Enable AA0 pin to be High when
selected
       64     000000        aap           equ    0      ; No, Enable AA0 pin on ext 'P' access
       65     000010        aax           equ    $10    ; Yes, Enable AA0 pin on ext 'X' access
       66     000000        aay           equ    0      ; No, Enable AA0 pin on ext 'Y' access
       67     000000        aswap         equ    0      ; No, Enable address bus swap
       68     000000        enpack        equ    0      ; No, Enable packing/unpacking logic
       69     000700        nadd          equ    $000700 ; Compare 7 address bits
       70     100000        msadd         equ    $100000 ; Most significant portion of address,
       71                                             ; $100000 - 11ffff, to compare.
       72                                             ;(0001,000x,xxxx,xxxx,xxxx,xxxx)
       73     100715        AAR0_value    equ
acctype+aahigh+aap+aax+aay+aswap+enpack+nadd+msadd
       74
       75                   ;--- BCR value = 0x00018C
       76     00000C        aaa0ws        equ    $C     ; Address Attribute Area 0 w/s = 12
       77     000180        aaa1ws        equ    $180   ; Address Attribute Area 1 w/s = 12
       78     000000        aaa2ws        equ    0      ; Address Attribute Area 2 w/s = 0
       79     000000        aaa3ws        equ    0      ; Address Attribute Area 3 w/s = 0
       80     000000        defws         equ    0      ; Default Address Area w/s = 0
       81     000000        busss         equ    0      ; Bus state status = 0
       82     000000        enblh         equ    0      ; Enable Bus Lock Hold = 0
       83     000000        enbrh         equ    0      ; Enable Bus Request Hold = 0
       84     00018C        BCR_value     equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       85
       86                   ;---------------------------------------------------------------
       87     P:000100                    org    p:$100 ; Keep the program in internal RAM
       88
       89                   flash3
       90
       91                   ;---------- Initialization Section ----------
       92     P:000100 08F4BD     movep  #PCTL_value,x:PCTL ; Set PLL Control Register
                       0E0013
       93     P:000102 05F43A     movec  #$004000,OMR     ; Disable Address Attribute Priority
                       004000
       94     P:000104 05F439     movec  #$080000,SR      ; Enable 1K Cache
                       080000
       95     P:000106 08F4BB     movep  #BCR_value,x:BCR ; Set external wait states
                       00018C
       96     P:000108 08F4B9     movep #AAR0_value,x:AAR0 ; Set Address Attribute Reg0
                       100715
       97     P:00010A 08F4B8     movep  #AAR1_value,x:AAR1 ; Set Address Attribute Reg1
                       100731
       98
       99                   ;---------------------------------------------------------------
      100                   ;---------------------------------------------------------------
      101                   do_x_checksum
```

```
       102    P:00010C 05F420     move          #-1,m0    ; Set LINEAR addressing mode
                       FFFFFF
       103    P:00010E 60F400     move          #XMemStart,r0 ; Set Starting Address of X:FLASH
                       100000
       104    P:000110 70F400     move          #XMemSize,n0 ; Set to Size of X:FLASH
                       020000
       105
       106    P:000112 200013     clr           a
       107    P:000113 20001B     clr           b
       108    P:000114 560000     move          a,x:CKSUM_CALC_X ; Initialize computed checksum
-> $000000
       109    P:000115 560100     move          a,x:CKSUM_READ_X ; Initialize read checksum ->
$000000
       110
       111                 ;---------- Compute the Checksum ----------
       112    P:000116 44F400     move          #$FFFF00,x0 ; Read Mask
                       FFFF00
       113    P:000118 06D810     dor           n0,x_loop
                       000004
       114    P:00011A 56D800     move          x:(r0)+,a ; Get the X:FLASH location Value
       115    P:00011B 200046     and           x0,a      ; Force input value to be $xxxx00
       116    P:00011C 200018     add           a,b       ; Compute checksum
       117                 x_loop
       118
       119    P:00011D 56E000     move          x:(r0),a  ; Get Old Checksum value
       120    P:00011E 20004E     and           x0,b      ; Force calculated Checksum to be
$xxxx00
       121    P:00011F 200046     and           x0,a      ; Force Old Checksum to be $xxxx00
       122    P:000120 570000     move          b,x:CKSUM_CALC_X ; Save Computed Checksum value
       123    P:000121 20001B     clr           b
       124    P:000122 560100     move          a,x:CKSUM_READ_X ; Save the Old Checksum value
       125    P:000123 578000     move          x:CKSUM_CALC_X,b ; Put Calculated Checksum
value in b1
       126    P:000124 20000D     cmp           a,b       ; Old Checksum = New Checksum?
       127    P:000125 0D104A     beq           x_done    ; Yes
                       000039
       128                                       ; No
       129                 ;---------- See if Checksum Location is Erased ----------
       130    P:000127 44F400     move          #$FFFF00,x0 ; FLASH erased value
                       FFFF00
       131    P:000129 200045     cmp           x0,a      ; Contents of checksum location =
Erased?
       132    P:00012A 0D104A     beq           x_write_checksum ; Yes
                       000021
       133                                               ; No
       134                 ;-----------------------------------------------------------------
       135                 ;                       N O T E
       136                 ;-----------------------------------------------------------------
       137                 ; If at least 64Kx16-bit words of external RAM is available
       138                 ; then; 1) last sector of the FLASH can be read into external RAM
       139                 ;       2)  the last sector of the FLASH can be erased,
       140                 ;       3)  and last sector of data along with the new checksum
       141                 ;           can be written back into the FLASH.
       142                 ; otherwise;  1)  erase the last sector of the FLASH,
       143                 ;             2)  and write the checksum to the FLASH.
       144                 ;-----------------------------------------------------------------
       145                 ;-----------------------------------------------------------------
       146                 ;--- Copy last sector of FLASH into external RAM (if available) ---
       147                 ;-----------------------------------------------------------------
       148
```

```
       149                ;---------- Erase last sector of FLASH ----------
       150    P:00012C 44F400     move          #$AAAA00,x0
                       AAAA00
       151    P:00012E 4C7000     move          x0,y:XMemStart+$5555; Unlock X:FLASH cycle 1
                       105555
       152
       153    P:000130 44F400     move          #$555500,x0
                       555500
       154    P:000132 4C7000     move          x0,y:XMemStart+$2AAA; Unlock X:FLASH cycle 2
                       102AAA
       155
       156    P:000134 44F400     move          #$808000,x0
                       808000
       157    P:000136 4C7000     move          x0,y:XMemStart+$5555; Send set-up command
                       105555
       158
       159    P:000138 44F400     move          #$AAAA00,x0
                       AAAA00
       160    P:00013A 4C7000     move          x0,y:XMemStart+$5555; Unlock X:FLASH cycle 1
                       105555
       161
       162    P:00013C 44F400     move          #$555500,x0
                       555500
       163    P:00013E 4C7000     move          x0,y:XMemStart+$2AAA; Unlock X:FLASH cycle 2
                       102AAA
       164
       165    P:000140 44F400     move          #$303000,x0
                       303000
       166    P:000142 447000     move    x0,x:XMemEnd-$800       ; Send sector erase command
                       11F800
       167
       168    P:000144 44F400     move    #$FFFF00,x0
                       FFFF00
       169                      x_wait_til_erased
       170    P:000146 56F000     move    x:XMemEnd-$800,a ; Get current value of location in
last sector
                       11F800
       171    P:000148 200046     and     x0,a
       172    P:000149 200045     cmp     x0,a                    ; Fully Erased?
       173    P:00014A 0527DC     bne     x_wait_til_erased       ; No
       174                                                 ; Yes
       175
       176                ;---------- Write Data Buffer Back to FLASH (if RAM available) ---
       177
       178                ;---------- X:FLASH Write Routine ----------
       179                ;  b  = contains data to be written to FLASH.
       180                ;  r0 = points to location in FLASH to be written.
       181                x_write_checksum
       182    P:00014B 44F400     move          #$AAAA00,x0
                       AAAA00
       183    P:00014D 4C7000     move          x0,y:XMemStart+$5555; Unlock X:FLASH cycle #1
                       105555
       184
       185    P:00014F 44F400     move          #$555500,x0
                       555500
       186    P:000151 4C7000     move          x0,y:XMemStart+$2AAA; Unlock X:FLASH cycle #2
                       102AAA
       187
       188    P:000153 44F400     move          #$A0A000,x0
                       A0A000
```

```
       189    P:000155 4C7000      move          x0,y:XMemStart+$5555; Send FLASH write command
                       105555
       190
       191    P:000157 576000      move          b,x:(r0)   ; Send data to write to X:FLASH
       192
       193    P:000158 44F400      move          #$FFFF00,x0
                       FFFF00
       194                  x_write_wait
       195    P:00015A 56E000      move          x:(r0),a  ;Get X:FLASH value at write location
       196    P:00015B 200046      and           x0,a      ; Force read value to be $xxxx00
       197    P:00015C 20000D      cmp           a,b               ; Write Done?
       198    P:00015D 0527DD      bne           x_write_wait      ; No
       199                                                         ; Yes
       200                  x_done
       201
;*************************************************************************
       202
;*************************************************************************
       203                  do_y_checksum
       204    P:00015E 05F420      move          #-1,m0    ; Set LINEAR addressing mode
                       FFFFFF
       205    P:000160 60F400      move          #YMemStart,r0 ; Set Starting Address of Y:FLASH
                       100000
       206    P:000162 70F400      move          #YMemSize,n0 ; Set to Size of Y:Flash
                       020000
       207
       208    P:000164 200013      clr           a
       209    P:000165 20001B      clr           b
       210    P:000166 560200      move          a,x:CKSUM_CALC_Y ; Initialize computed checksum
-> $000000
       211    P:000167 560300      move          a,x:CKSUM_READ_Y ; Initialize read checksum ->
$000000
       212
       213                  ;---------- Compute the Checksum ----------
       214    P:000168 44F400      move          #$FFFF00,x0
                       FFFF00
       215    P:00016A 06D810      dor           n0,y_loop
                       000004
       216    P:00016C 5ED800      move          y:(r0)+,a ; Get the Y:FLASH location Value
       217    P:00016D 200046      and           x0,a      ; Force read value to $xxxx00
       218    P:00016E 200018      add           a,b       ; Compute checksum
       219                  y_loop
       220
       221    P:00016F 5EE000      move          y:(r0),a  ; Get Old Checksum value
       222    P:000170 20004E      and           x0,b      ; Force calculated checksum to
$xxxx00
       223    P:000171 200046      and           x0,a      ; Force Old Checksum to $xxxx00
       224    P:000172 570200      move          b,x:CKSUM_CALC_Y ; Save Computed Checksum value
       225    P:000173 20001B      clr           b
       226    P:000174 560300      move          a,x:CKSUM_READ_Y ; Save the Old Checksum value
       227    P:000175 578200      move          x:CKSUM_CALC_Y,b ; Get calculated Checksum
value in b1
       228    P:000176 20000D      cmp           a,b       ; Old Checksum = New Checksum?
       229    P:000177 0D104A      beq           y_done    ; Yes
                       000039
       230                                                  ; No
       231                  ;---------- See if Checksum Location is Erased ----------
       232    P:000179 44F400      move          #$FFFF00,x0 ; FLASH erased value
                       FFFF00
       233    P:00017B 200045      cmp           x0,a   ;Contents of checksum location = Erased?
```

```
234    P:00017C 0D104A     beq          y_write_checksum ; Yes
               000021
235
236                  ;---------------------------------------------------------------
237                  ;                        N O T E
238                  ;---------------------------------------------------------------
239                  ; If at least 64Kx16-bit words of external RAM is available
240                  ; then 1) last sector of FLASH can be read into external RAM
241                  ;       2)  the last sector of the FLASH can be erased,
242                  ;       3)  and the last sector of data along with the new checksum
243                  ;           can be written back into the FLASH.
244                  ; otherwise;  1)  erase the last sector of the FLASH,
245                  ;             2)  and write the checksum to the FLASH.
246                  ;---------------------------------------------------------------
247                  ;--- Copy last sector of FLASH into external RAM (if available) ---
248                  ;---------------------------------------------------------------
249
250                  ;---------- Erase last sector of FLASH ----------
251    P:00017E 44F400     move         #$AAAA00,x0
               AAAA00
252    P:000180 4C7000     move         x0,y:YMemStart+$5555; Unlock Y:FLASH cycle 1
               105555
253
254    P:000182 44F400     move         #$555500,x0
               555500
255    P:000184 4C7000     move         x0,y:YMemStart+$2AAA; Unlock Y:FLASH cycle 2
               102AAA
256
257    P:000186 44F400     move         #$808000,x0
               808000
258    P:000188 4C7000     move         x0,y:YMemStart+$5555; Send set-up command
               105555
259
260    P:00018A 44F400     move         #$AAAA00,x0
               AAAA00
261    P:00018C 4C7000     move         x0,y:YMemStart+$5555; Unlock Y:FLASH cycle 1
               105555
262
263    P:00018E 44F400     move         #$555500,x0
               555500
264    P:000190 4C7000     move         x0,y:YMemStart+$2AAA; Unlock Y:FLASH cycle 2
               102AAA
265
266    P:000192 44F400     move         #$303000,x0
               303000
267    P:000194 4C7000     move         x0,y:YMemEnd-$800; Send sector erase command
               11F800
268
269    P:000196 44F400     move   #$FFFF00,x0
               FFFF00
270                  y_wait_til_erased
271    P:000198 5EF000     move   y:YMemEnd-$800,a; Get value of location in last sector
               11F800
272    P:00019A 200046     and    x0,a             ; Mask value to $xxxx00
273    P:00019B 200045     cmp    x0,a             ; Fully Erased?
274    P:00019C 0527DC     bne    y_wait_til_erased ; No
275                                                 ; Yes
276
277                  ;---------- Write Data Buffer Back to FLASH (if RAM available) ----
278
```

```
       279                ;---------- Y:FLASH Write Routine ----------
       280                ;  b  = contains data to be written to FLASH.
       281                ;  r0 = points to location in FLASH to be written.
       282                y_write_checksum
       283    P:00019D 44F400     move          #$AAAA00,x0
                       AAAA00
       284    P:00019F 4C7000     move          x0,y:YMemStart+$5555; Unlock Y:FLASH cycle #1
                       105555
       285
       286    P:0001A1 44F400     move          #$555500,x0
                       555500
       287    P:0001A3 4C7000     move          x0,y:YMemStart+$2AAA; Unlock Y:FLASH cycle #2
                       102AAA
       288
       289    P:0001A5 44F400     move          #$A0A000,x0
                       A0A000
       290    P:0001A7 4C7000     move          x0,y:YMemStart+$5555; Send FLASH write command
                       105555
       291
       292    P:0001A9 5F6000     move          b,y:(r0)   ; Send data to write to Y:FLASH
       293
       294    P:0001AA 44F400     move          #$FFFF00,x0
                       FFFF00
       295                y_write_wait
       296    P:0001AC 5EE000     move          y:(r0),a   ; Get current Y:FLASH value at write
location
       297    P:0001AD 200046     and           x0,a
       298    P:0001AE 20000D     cmp           a,b              ; Write Done?
       299    P:0001AF 0527DD     bne           y_write_wait     ; No
       300                                                     ; Yes
       301                y_done
       302
;*********************************************************************
       303
       304    P:0001B0 050C00     bra           *                ; Done, Do a Dynamic Halt
       305
       306                        end           flash3

       0    Errors
       0    Warnings
```

6 +3.3 V PEROM Memory

Programmable erasable read-only memory (PEROM) provides non-volatile program and data storage that is in-circuit reprogrammable and offers relatively fast access times. However, even the fastest PEROM memories require a DSP core running at 80 MHZ to generate external memory wait states. With each wait state equivalent to one clock period of the DSP core, one wait state for a core running at 80 MHZ is roughly 12.5 nS. When the external memory device requires address stability during an entire external access, a DSP56300 family derivative incurs an automatic one wait state penalty. Since PEROM devices require address stability, the DSP operates with at least one wait state when these external memories are used.

The following three Atmel PEROM design examples illustrate the ease and flexibility in using a single 3.3 V PEROM device with the DSP56300 family.

- The first example, 32KK × 8-bit boot PEROM, shows one solution for an embedded system, which loads a program from PEROM at boot time into the DSP internal RAM and then executes it.

- The second example, 512K × 8-bit boot/overlay PEROM, shows another solution for a bootable embedded system. In this implementation, the DSP loads internal program RAM from the PEROM at boot time and then overlays new programs and data from the PEROM when needed.
- The third example, 32K × 16-bit X data and 32K × 16-bit Y data PEROM, shows a solution that allows a user to modify or upgrade the 16-bit reference data tables.

6.1 32K × 8-Bit Boot PEROM Example

The 32K × 8-bit bootstrap PEROM implementation uses the Atmel AT29LV256-25. See **Figure 19** for the memory map layout, **Figure 20** for the block diagram, and **Figure 23** for the schematic. A program placed in the PEROM can be loaded into the DSP RAM at boot time and executed. The program can then save an updated version of the program back into the PEROM for later restarts.

The DSP core runs at 80 MHz and the input frequency source is from a 4.000 MHz crystal. For a 250 nS PEROM, twenty wait states are required. During the DSP boot sequence, the DSP generates thirty-one wait states to accommodate slow memories. On exit from reset, the core for our example runs at 4.0 MHz, allowing a 7.7 mS access time device to be used. This 3.3 V device is organized as 32K × 8 bits with a 250 nS access time. One memory device is used to achieve the 8-bit wide boot bus.

During reset with Mode 1 selected, the DSP boot code configures Address Attribute Line 1 for program accesses in the address range $D00000–$DFFFFF. The boot code then loads bytes from PEROM, packs them into 24-bit words and stores them into program RAM. The first word, three packed bytes read from the PEROM, indicates the number of words to load, and the second word contains the starting load address for the packed data. This starting load address is also the address that gains program control after the program load is completed.

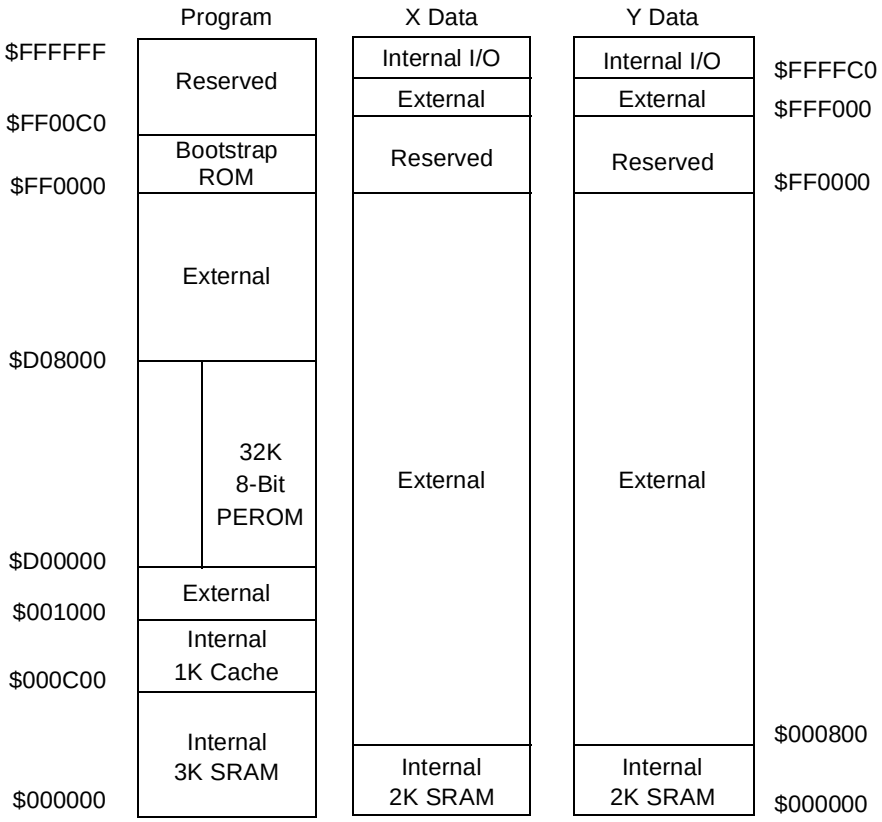

Figure 19. 32K × 8-Bit Boot PEROM Memory Map

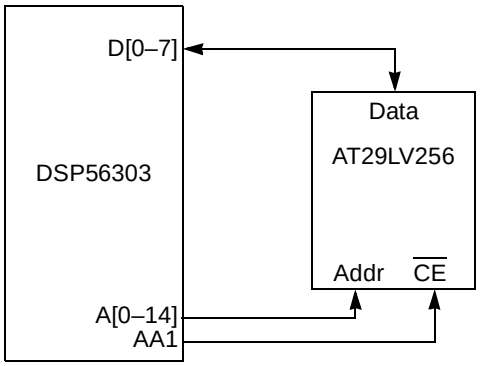

Figure 20. 32K × 8-Bit BOOT/Overlay PEROM Memory Example

6.1.1 PEROM Timing Requirements

Following are the timing requirements for the AT29LV256-25 32K × 8-bit 250 nS PEROM. **Table 9** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 21**.

Table 9. AT29LV256-25 Memory Read TIming Specifications

Read Cycle Parameter	Symbol	Min	Max
Address to Output Delay	t_{ACC}	—	250 nS
Chip Enable to Output Delay	t_{CE}	—	250 nS
Output Enable to Output Delay	t_{OE}	—	120 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0 nS	—

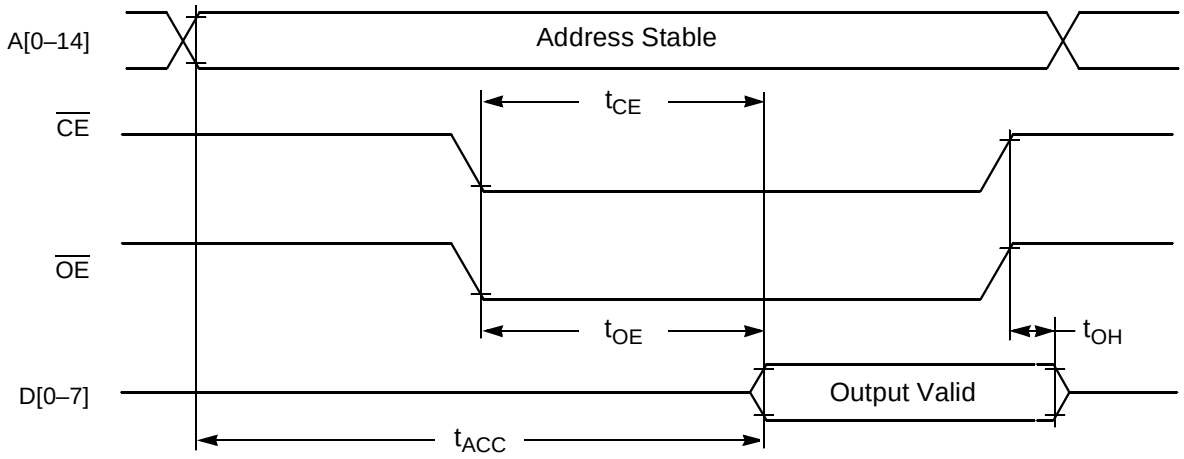

Figure 21. AT29LV256 Memory Read Cycle Timing Diagram

Table 10 shows the memory write timing values used in the memory write cycle timing diagram, **Figure 22**.

Table 10. AT29LV256-25 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write Cycle Time (to Program)	t_{WC}	—	20 mS
Address set-up Time	t_{AS}	10 nS	—

Table 10. AT29LV256-25 Memory Write Timing Specifications (Continued)

Write Cycle Parameter	Symbol	Min	Max
$\overline{CE}$ set-up Time	t_{CS}	0 nS	—
Write Pulse Width	t_{WP}	200 nS	—
Data set-up Time	t_{DS}	100 nS	—
Data Hold Time	t_{DH}	10 nS	—

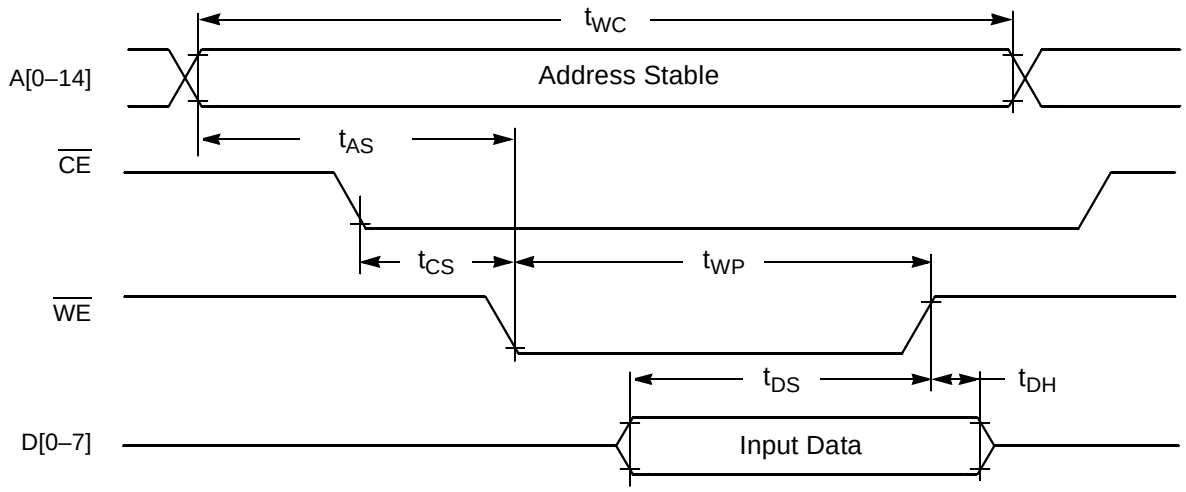

Figure 22. AT29LV256 Memory Write Cycle Timing Diagram

Non-volatile memory locations in the PEROM are not individually programmable, but must be programmed as a sector of sixty-four locations. The AT29LV256 device is organized as 512 sectors of sixty-four bytes each. To change one memory location, program the entire sector in which the memory location resides. The programming cycle erases and programs all sixty-four locations; there is no separate erase cycle.

Writing to a PEROM memory location requires writing a complete sector, or 64 bytes of data, to the PEROM in the following sequence:

1. Write $0000AA to location $5555 relative to the PEROM.
2. Write $000055 to location $2AAA relative to the PEROM.
3. Write $0000A0 to location $5555 relative to the PEROM.
4. Write 64 bytes of data to sector in the PEROM.
5. Read last address in sector until data read = data written.

6.1.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 32K × 8-bit Boot PEROM memory configuration, set the core speed of the DSP for optimum processor and memory performance using the DSP PLL and Clock Generation (PCTL) register. For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. The PCTL register value combines the following bits for each feature:

- Desired core frequency = 80 MHz
- Given the external frequency = 4.000 MHz
- Predivider value = 1, bits 20–23 = $0

- Low-power divider value = 1, bits 12–14 = $0
- VCO multiplication value = 20, bits 0–11 = $013
- Crystal less than 200 kHz, bit 15 = 0
- Disable XTAL drive output, bit 16 = 0
- PLL runs during STOP, bit 17 = 1
- Enable PLL operation, bit 18 = 1
- Disable core clock output, bit 19 = 1

The value loaded into the PCTL register is $0E0013.

AA1 connects to PEROM pin $\overline{CE}$ to enable the external 32K PEROM bank for program space accesses between $D00000–$D07FFF. Configure the memory address space requirements for AA1 using the Address Attribute Register 1 (AAR1). The AAR1 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA1 pin high when selected, bit 2 = 0.
- Activate the AA1 pin during external program space accesses, bit 3 = 1.
- Activate the AA1 pin during external X data space accesses, bit 4 = 0.
- Activate the AA1 pin during external Y data space accesses, bit 5 = 0.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $9.
- Specify the most significant portion of the address to compare, bits 12–23 = $D00.

The value loaded into the AAR1 is $D00909.

The value loaded into AAR0, AAR2 and AAR3 is $00000.

Select the proper number of wait states for the memory configuration using the Bus Control Register (BCR). The BCR value combines the following bits for each feature:

- Address attribute area 0 wait states, bits 0-4 = $0.
- Address attribute area 1 wait states, bits 5-9 = $14.
- Address attribute area 2 wait states, bits 10-12 = $0.
- Address attribute area 3 wait states, bits 13-15 = $0.
- Default address area wait states, bits 16-20 = $0.
- Bus state status, bit 21 = 0.
- Enable Bus Lock Hold, bit 22 = 0.
- Enable Bus Request Hold, bit 23 =0.

The value loaded into the BCR is $000280.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR value combines the following bits for each feature:

- MA–MD bits specify the DSP operating mode, bits 0–3 = $0.
- External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4 = $0.
- Memory Switch Mode bit, reconfigures internal memory spaces, bit 7 = $0.
- Transfer Acknowledge Synchronize Select bit selects the synchronization method for the Transfer Acknowledge, TA, pin, bit 11 = $0.
- Bus Release Timing bit selects between a fast and slow bus release of the $\overline{BB}$ pin, bit 12 = $0.
- Address Attribute Priority Disable bit allows the Address Attribute pins, AA0–AA3, to be used in any combination, bit 14 = $1.
- All other OMR bits are selected for their defaults of $0.

The value loaded into the OMR is $004000.

Configure the memory mode of the DSP using the Status Register (SR). The SR value combines the following bits for each feature:

- Sixteen-Bit Compatibility Mode enables full compatibility to object code written for the DSP56000 Family of DSPs, SR Bit 13 = $0.
- Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, SR Bit 19 = $1.
- All other Status Register bits are selected for their defaults of $0.

The value loaded into the SR is $080000, which is the value loaded during reset.

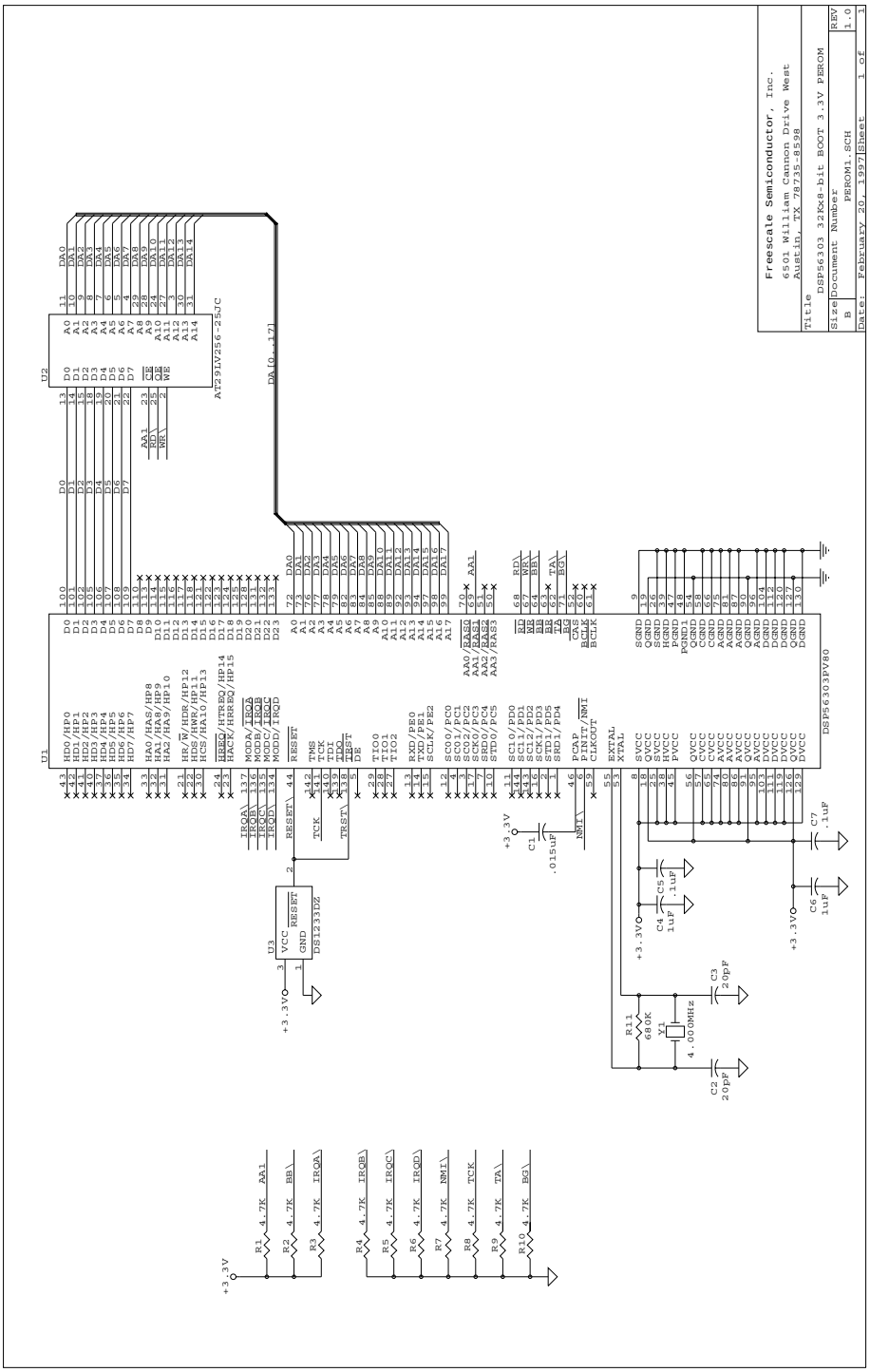

Figure 23. 32K × 8-Bit BOOT PEROM Schematic

Example 4. 32K x 8-bit BOOT PEROM Checksum Verify Program

```
        Freescale DSP56300 Assembler Version 6.0.1.6   97-02-23 09:09:05 perom1.asm

        1                   page    132,60,3,3,
        2             ;
        3             ; perom1.asm, simple program to calculate the 8-Bit
                      ; checksum for
        4             ; a 32K x 8-Bit block of PEROM memory using a DSP56303.
        5             ; Contains: Initialization routine,
        6             ; Routine to calculate 8-Bit checksum,
        7             ; Routine to read checksum sector,
        8             ; Routine to write checksum sector.
        9             ;
        10            ; Program runs in Internal P:RAM to calculate the checksum on
        11            ; External P:PEROM from $D00000 - $D07FFF @ 20w/s
        12            ;
        13
        14    D00000 MemStart     equ    $D00000
        15    D08000 MemEnd       equ    $D08000
        16    008000 MemSize      equ    MemEnd-MemStart  ; Last Word is stored Checksum value
        17
        18            SectorStart
        19    D07FC0              equ    $D07FC0          ; Start of Checksum Sector
        20    000040 SectorSize   equ    64               ; Size of Checksum Sector
        21
        22            ;--- Program Specific Storage Locations (X DATA SPACE)
        23            NEW_CHECKSUM
        24    000000              equ    $000000          ; Computed Checksum Value
        25            OLD_CHECKSUM
        26    000001              equ    $000001          ; Old Checksum from PEROM
        27
        28    000002 DataBuffer   equ    $000002          ; Start of Last Sector Storage Buffer
        29
        30            ;--- DSP56303 Control Registers (X I/O SPACE)
        31    FFFFFB BCR          equ    $FFFFFB          ; Bus Control Register
        32    FFFFFD PCTL         equ    $FFFFFD          ; PLL Control Register
        33    FFFFF8 AAR1         equ    $FFFFF8          ; Address Attribute Register 1
        34
        35            ;--- PCTL value = 0x0E0013
        36    000000       prediv equ    0                ; Predivider = 1
        37    000000       lowdiv equ    0                ; Low Power Divider = 1
        38    000013       pllmul equ    19             ; VCO Mult = 20; (19+1)*4.00MHz=80.00MHz
        39    000000       crystalequ    0                ; No, Crystal not less than 200kHz
        40    000000       disXTALequ    0                ; No, do not disable crystal use
        41    020000       pllstop equ   $020000          ; Yes, PLL runs during STOP
        42    040000       enpll  equ    $040000          ; Yes, enable PLL operation
        43    080000       disclk equ    $080000          ; Yes, disable CORE clock output
        44    0E0013       PCTL_valueequ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk
        45
        46            ;--- AAR1 value = 0xD00909
        47    000001       acctype1 equ  1      ; External Memory access type = 0x1
        48    000000       aahigh1 equ   0      ; Enable AA1 pin low when selected
        49    000008       aap1   equ    $8     ; Yes, Enable AA1 pin on ext P access
        50    000000       aax1   equ    0      ; No, Enable AA1 pin on ext X access
        51    000000       aay1   equ    0      ; No, Enable AA1 pin on ext Y access
        52    000000       aswap1 equ    0      ; No, Enable address bus swap
        53    000000       enpack1 equ   0      ; No, Enable packing/unpacking logic
        54    000900       nadd1  equ    $000900      ; Compare 9 address bits
        55    D00000       msadd1 equ    $D00000      ; Most significant portion of address,
        56                                            ; $D00000 - D07fff, to compare.
        57                                            ;  (1101,0000,0xxx,xxxx,xxxx,xxxx)
```

```
       58     D00909       AAR1_valueequ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       59
       60
       61           ;--- BCR value = 0x000280
       62     000000       aaa0ws equ    0      ; Address Attribute Area 0 w/s = 0
       63     000280       aaa1ws equ    $280   ; Address Attribute Area 1 w/s = 20
       64     000000       aaa2ws equ    0      ; Address Attribute Area 2 w/s = 0
       65     000000       aaa3ws equ    0      ; Address Attribute Area 3 w/s = 0
       66     000000       defws  equ    0      ; Default Address Area w/s = 0
       67     000000       busss  equ    0      ; Bus state status = 0
       68     000000       enblh  equ    0      ; Enable Bus Lock Hold = 0
       69     000000       enbrh  equ    0      ; Enable Bus Request Hold = 0
       70     000280       BCR_valueequ  aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       71
       72           ;-------------------------------------------------------
       73           ;       Header for DSP56303 BOOT Code
       74           ;-------------------------------------------------------
       75     P:0000FE             org    p:$100-2
       76
       77     P:0000FE     dc      pgm_end-perom1          ; Number of words in program
       78     P:0000FF     dc      perom1                  ; Starting address for program
       79
       80
       81           ;-------------------------------------------------------
       82     P:000100     org     p:$100                  ;Keep program in internal RAM
       83
       84                  perom1
       85     P:0001000D1080       bsr    init             ; Initialize DSP
                   000011
       86
       87     P:0001020D1080       bsr    calc_checksum    ; Calculate Checksum of PEROM
                   000018
       88
       89     P:000104548100       move   x:OLD_CHECKSUM,a1 ; Get PEROM's Old Checksum value
       90     P:000105558000       move   x:NEW_CHECKSUM,b1 ; Get Calculated Checksum value
       91     P:00010620000D       cmp    a,b              ; Old Checksum = New Checksum?
       92     P:0001070D104A       beq    _done            ; Yes, we are done
                   000009
       93                                                  ; No
       94     P:000109 0D1080      bsr    save_sector   ; Save contents of PEROM Checksum Sector
                   000028
       95
       96     P:00010B 448000      move   x:NEW_CHECKSUM,x0 ; Get Calculated Checksum value
       97     P:00010C 447000      move   x0,x:DataBuffer+SectorSize-1; Update Checksum
location in buffer
                   000041
       98     P:00010E 0D1080      bsr    write_sector  ; Write saved PEROM Checksum sector Data
                   000033
       99                                                  ; with New Checksum value to PEROM
       100          _done
       101    P:000110 050C00      bra    *                ; DONE, Do a dynamic HALT
       102
       103
       104          ;---------------------------------------------------------------------
       105          ; Initialization Section
       106          ;---------------------------------------------------------------------
       107          init
       108    P:000111 08F4BD      movep   #PCTL_value,x:PCTL     ; Set PLL Control Register
                   0E0013
       109    P:000113 05F439      movec   #$080000,SR            ; Enable 1K Cache
```

```
                080000
       110      P:000115 08F4BB      movep   #BCR_value,x:BCR       ; Set external wait states
                000280
       111      P:000117 08F4B8      movep   #AAR1_value,x:AAR1     ; Set Address Attribute Reg1
                D00909
       112
       113      P:000119 00000C      rts
       114

       115              ;---------------------------------------------------------------------
       116              ; Routine to Calculate 8-Bit Checksum
       117              ;---------------------------------------------------------------------
       118              calc_checksum
       119      P:00011A 05F420      move    #-1,m0           ; Set LINEAR addressing mode
                FFFFFF
       120      P:00011C 60F400      move    #MemStart,r0     ; Set Starting Address of PEROM
                D00000
       121      P:00011E 70F400      move    #MemSize-1,n0    ; Set to Size of Flash - Checksum
                007FFF
       122
       123      P:000120 200013      clr     a
       124      P:000121 20001B      clr     b
       125      P:000122 540000      move    a1,x:NEW_CHECKSUM ;Initialize computed checksum ->
$000000
       126      P:000123 540100      move    a1,x:OLD_CHECKSUM ;Initialize read checksum ->
$000000
       127
       128              ; Compute the 8-Bit Checksum
       129      P:000124 44F400      move    #>$FF,x0         ; Set lower Byte Mask
                0000FF
       130
       131      P:000126 06D810      dor     n0,_ploop
                         000004
       132      P:000128 07D88C      move    p:(r0)+,a1    ; Get the PEROM location Value
       133      P:000129 200046      and     x0,a          ; Mask for lower byte
       134      P:00012A 200018      add     a,b           ; Compute checksum
       135              _ploop
       136
       137      P:00012B 07E08C      move    p:(r0),a1 ; Get PEROM Old Checksum value
       138      P:00012C 20004E      and     x0,b    ; Limit calculated Checksum to lower byte
       139      P:00012D 200046      and     x0,a          ; Limit Old Checksum to lower byte
       140      P:00012E 550000      move    b1,x:NEW_CHECKSUM ; Save the Computed Checksum value
       141      P:00012F 540100      move    a1,x:OLD_CHECKSUM ; Save the Old Checksum value
       142
       143      P:000130 00000C      rts
       144
       145              ;---------------------------------------------------------------------
       146              ; Routine to Read the current contents of the PEROM
       147              ;        where the Checksum is stored
       148              ;---------------------------------------------------------------------
       149              save_sector
       150      P:000131310200       move    #DataBuffer,r1 ; Point to start of data storage buffer
       151      P:00013205F421       move    #-1,m1           ; Set modulo to linear
                FFFFFF
       152
       153      P:000134 60F400      move    #SectorStart,r0  ; Point to Start of Sector in PEROM
                D07FC0
       154      P:000136 384000      move    #SectorSize,n0   ; Get size of Sector
       155      P:000137 05F420      move    #-1,m0           ; Set modulo to linear FFFFFF
       156
```

```
157    P:000139 44F400     move   #>$FF,x0         ; Lower Byte Mask 0000FF
158
159    P:00013B 06D810     dor    n0,_read_loop 000004
160    P:00013D 07D88C     move   p:(r0)+,a1       ; Read a word from the PEROM sector
161    P:00013E 200046     and    x0,a             ; Mask data to lower byte, ie $0000xx
162    P:00013F 545900     move   a1,x:(r1)+       ; Save off a word in storage buffer
163                 _read_loop
164
165    P:000140 00000C     rts
166
167           ;---------------------------------------------------------------------
168           ; Routine to Place PEROM into ERASE/WRITE MODE and send it sector of Data
169           ;---------------------------------------------------------------------
170           write_sector
171    P:000141310200      move   #DataBuffer,r1 ; Point to start of data storage buffer
172    P:00014205F421      move   #-1,m1           ; Set modulo to linear FFFFFF
173
174    P:000144 60F400     move   #SectorStart,r0  ; Point to Start of Sector in PEROM
              D07FC0
175    P:000146 384000     move   #SectorSize,n0   ; Get size of sector
176    P:000147 05F420     move   #-1,m0           ; Set modulo to linear FFFFFF
177
178           ;-- Place PEROM into ERASE/WRITE MODE
179    P:000149 44F400     move   #>$AA,x0
              0000AA
180    P:00014B 077084     move   x0,p:MemStart+$5555 ; Unlock PEROM Cycle #1
              D05555
181
182    P:00014D 44F400     move   #>$55,x0
              000055
183    P:00014F 077084     move   x0,p:MemStart+$2AAA ; Unlock PEROM Cycle #2
                   D02AAA
184
185    P:000151 44F400     move   #>$A0,x0
              0000A0
186    P:000153 077084     move   x0,p:MemStart+$5555 ; Send PEROM Write Command
              D05555
187           ; -- PEROM Writes are now enabled
188           ;-- Send a Sector of data to the PEROM
189    P:000155 06D810     dor    n0,_write_loop
              000004
190    P:000157 54D900     move   x:(r1)+,a1       ; Read a byte from the storage buffer
191    P:000158 07588C     move   a1,p:(r0)+       ; Write the byte to PEROM
192    P:000159 000000     nop
193           _write_loop
194           ; -- Now in PEROM's Data Protect State
195           ;-- Wait till ERASE/WRITE Cycle is complete
196    P:00015A 205000     move   (r0)-            ; Point to last PEROM's location
197    P:00015B 44F400     move   #>$FF,x0         ; Lower Byte Mask
              0000FF
198
199           _write_wait
200    P:00015D 07E08D     move   p:(r0),b1     ; Get current value at PEROM location
201    P:00015E 20004E     and    x0,b          ; Mask for lower byte
202    P:00015F 20000D     cmp    a,b           ; Last value written = value in PEROM?
203    P:000160 0527DD     bne    _write_wait      ; No, wait until it is
204                                                ; Yes
205    P:000161 00000C     rts
206
207           ;---------------------------------------------------------------------
```

```
208           pgm_end
209
210           end     perom1

0       Errors
0       Warnings
```

6.2 512K × 8-Bit Boot/Overlay PEROM Example

The 512K × 8-bit bootstrap with X data space program overlay PEROM implementation uses the Atmel AT29LV040A device. See **Figure 24** for the boot memory map layout, **Figure 25** for the overlay memory map layout, **Figure 26** for the block diagram, and **Figure 29** for the schematic. A program placed in the PEROM can be loaded into the DSP at boot time, and then executed. The program can then load additional programs or data from the PEROM into the DSP using overlay techniques. The boot or overlay program can then save an updated version of the program or data back into the PEROM for later use.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 200 nS PEROM, sixteen wait states are required. However, during the DSP boot sequence, the DSP generates thirty-one wait states to accommodate slow memories. At boot time the core for this example runs at 4.0 MHz, allowing a 7.7 mS access time device to be used. This 3.3 V device is organized as 512K × 8-bits with a 200 nS access time. One memory device is used to achieve the 8-bit wide boot bus.

During reset with Mode 1 selected, the DSP boot code configures Address Attribute Line 1 for program accesses in the address range $D00000–$DFFFFF, as shown in **Figure 24**. The boot code then reads bytes from PEROM, packs them into 24-bit words, and stores them into program RAM. The first word, three packed bytes, read from the PEROM indicates the number of words to load. The second word from the PEROM contains the load address for the packed data. This is also the address to which control is transferred after the program load is completed. Under user program control, the AA1 can then be configured for X data space accesses in the address range $100000–$17FFFF, as shown in **Figure 25**. Address Attribute Line 0 can then be configured to select the upper half of the 512K PEROM by enabling PEROM A18 high when X:$14FFFF–X:$17FFFF is selected.

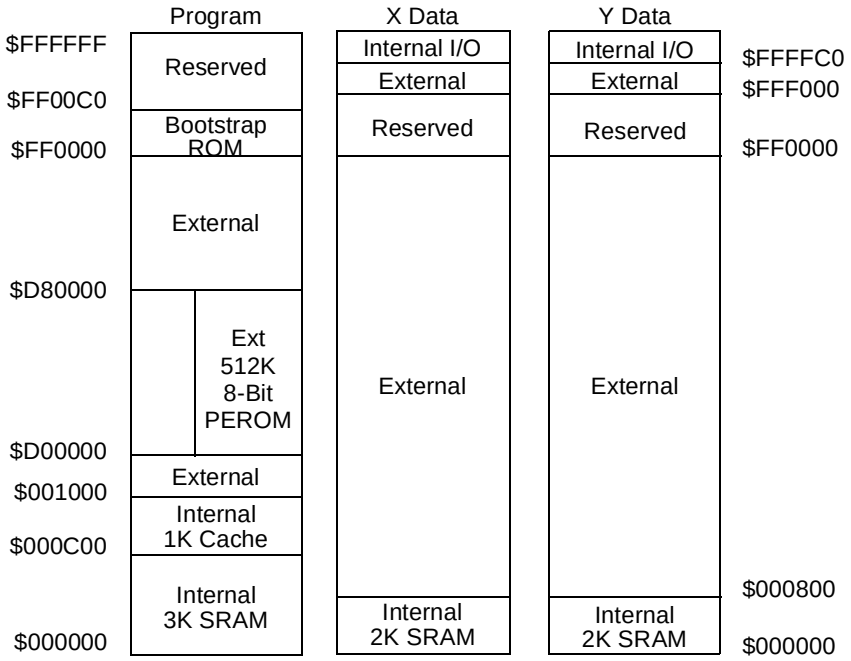

Figure 24. 512K × 8-Bit BOOT PEROM Memory Map

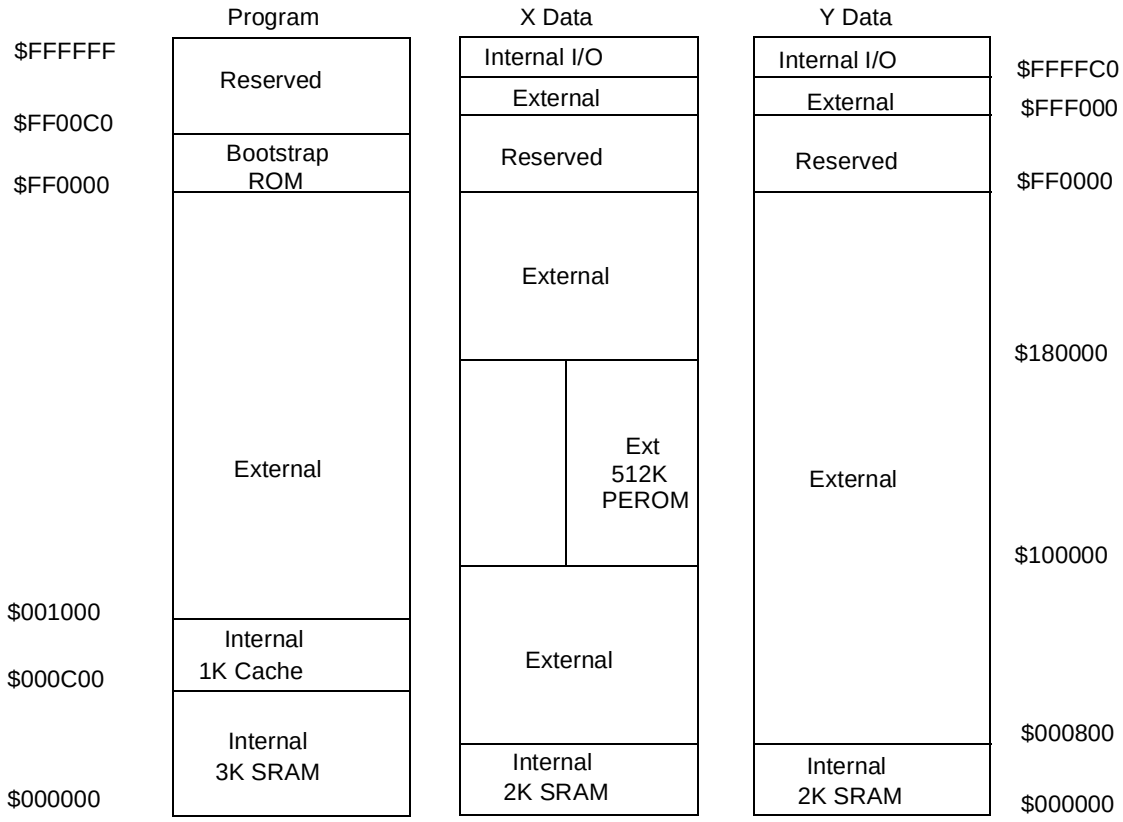

Figure 25. 512K x 8-bit Overlay PEROM Memory Map

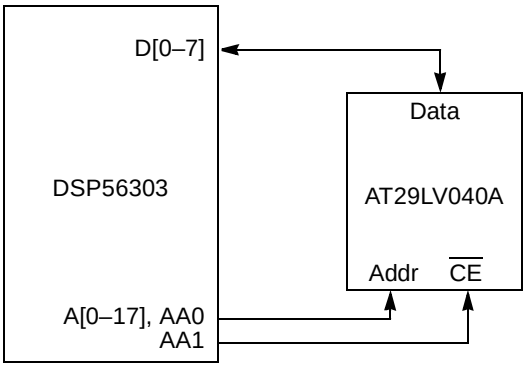

Figure 26. 512K × 8-Bit Boot/Overlay PEROM Block Diagram

6.2.1 PEROM Timing Requirements

For the PEROM device to work properly, its timing requirements must be met. Following are the timing requirements for the AT29LV040A-20 512K × 8-bit 200 nS Flash memory. **Table 11** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 27**.

Table 11. AT29LV040A-20 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Address to Output Delay	t_{ACC}	—	200 nS
Chip Enable to Output Delay	t_{CE}	—	200 nS

Table 11. AT29LV040A-20 Memory Read Timing Specifications (Continued)

Read Cycle Parameter	Symbol	Min	Max
Output Enable to Output Delay	t_{OE}	0 nS	100 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0 nS	—

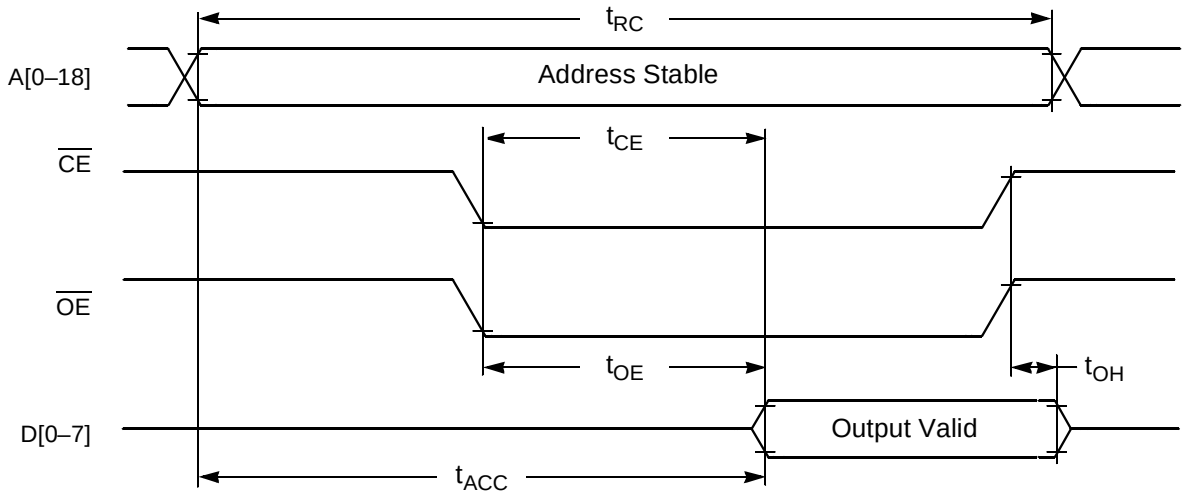

Figure 27. AT29LV040A Memory Read Cycle Timing Diagram

Table 12 shows the memory write timing specification values used in the memory write cycle timing diagram, **Figure 28**.

Table 12. AT29LV049A-20 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write Cycle Time (for Programming)	t_{WC}	—	20 mS
Address set-up Time	t_{AS}	10 nS	—
$\overline{CE}$ set-up Time	t_{CS}	0 nS	—
Write Pulse Width	t_{WP}	200 nS	—
Data set-up Time	t_{DS}	100 nS	—
Data Hold Time	t_{DH}	10 nS	—

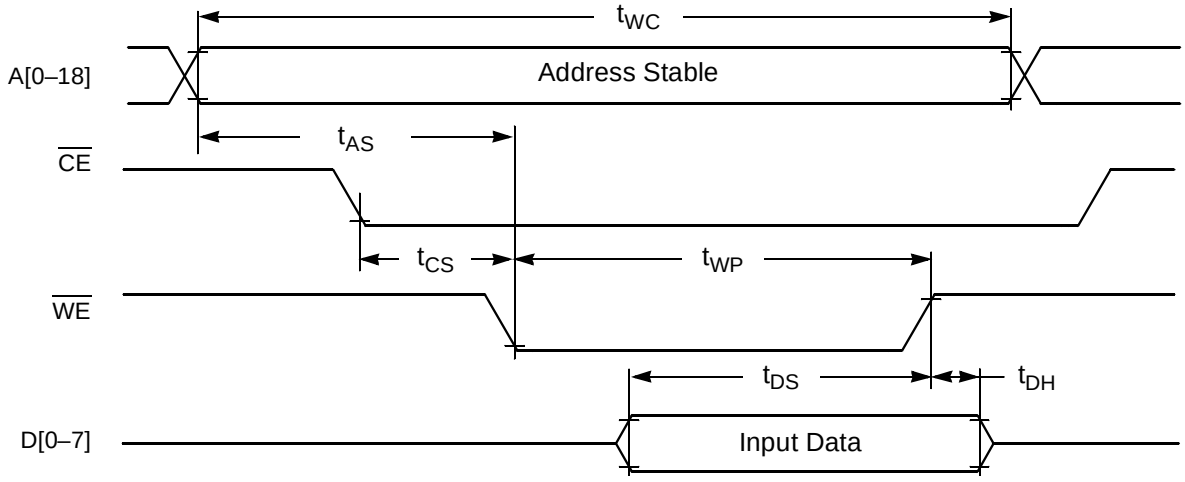

Figure 28. AT29LV040A Memory Write Cycle Timing Diagram.

Non-volatile memory locations in the PEROM cannot be individually programmed. However, a sector consisting of 256 locations can be programmed. The AT29LV040 device is organized as 2048 sectors of 256 bytes each, and to erase a memory location the sector it resides in must be erased. Writing to a PEROM memory location requires writing a complete sector, or 256 bytes of data, to the PEROM in the following sequence:

1. Write $0000AA to location $5555 relative to the PEROM
2. Write $000055 to location $2AAA relative to the PEROM
3. Write $0000A0 to location $5555 relative to the PEROM
4. Write 256 bytes of 8-bit data to sector in the PEROM
5. Read last sector address until data read = data written.

Note: Only the last eight bits of a 24-bit word are significant.

6.2.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 512K × 8-bit Boot/Overlay PEROM memory configuration, set the core speed of the DSP for optimum processor and memory performance using the DSP PLL and Clock Generation (PCTL) register. For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. The PCTL register value combines the following bits for each feature:

- Desired Core Frequency = 80 MHz
- Given the External Frequency = 4.000 MHz
- Predivider value = 1, bits 20–23 = $0
- Low-power Divider value = 1, bits 12–14 = $0
- VCO Multiplication value = 20, bits 0–11 = $013
- Crystal less than 200 kHz, bit 15 = 0
- Disable XTAL drive output, bit 16 = 0
- PLL runs during STOP, bit 17 = 1
- Enable PLL operation, bit 18 = 1
- Disable core clock output, bit 19 = 1

The value loaded into the PCTL register is $0E0013.

AA1 enables, via PEROM $\overline{CE}$, external 512K PEROM bank accesses in the address range $100000–$17FFFF during X data space requests. Configure the memory address space requirements for AA1 using the Address Attribute Register 1 (AAR1). The AAR1 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 0.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the AA pin during external Y data space accesses, bit 5 = 0.

- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 1.
- Specify the number of address bits to compare, bits 8–11 = $5.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR1 is $100591.

Address Attribute Pin 0 (AA0) selects, via PEROM A18, the upper addresses of the external 512K PEROM memory bank accesses in the address range from $140000 to $17FFFF during Program space requests. Configure the memory address space requirements for AA0 using the Address Attribute Register 0 (AAR0). The AAR0 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 1.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the AA pin during external Y data space accesses, bit 5 = 0.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 1.
- Specify the number of address bits to compare, bits 8–11 = $6.
- Specify the most significant portion of the address to compare, bits 12–23 = $140.

The value loaded into the AAR0 is $140695.

The value loaded into AAR2 and AAR3 is $000000.

Select the proper number of wait states for the memory configuration using the Bus Control Register (BCR). The BCR value combines the following bits for each feature:

- Address attribute area 0 wait states, bits 0–4 = $10.
- Address attribute area 1 wait states, bits 5–9 = $10.
- Address attribute area 2 wait states, bits 10–12 = $0.
- Address attribute area 3 wait states, bits 13–15 = $0.
- Default address area wait states, bits 16–20 = $0.
- Bus state status, bit 21 = 0.
- Enable Bus Lock Hold, bit 22 = 0.
- Enable Bus Request Hold, bit 23 = 0.

The value loaded into the BCR is $000210.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR value combines the following bits for each feature:

- MA–MD bits specify the DSP operating mode, bits 0–3 = $0.

- External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4 = $0.
- Memory Switch Mode bit reconfigures internal memory spaces, bit 7 = $0.
- Transfer Acknowledge Synchronize Select bit selects the synchronization method for the Transfer Acknowledge, TA, pin, bit 11 = $0.
- Bus Release Timing bit selects between a fast and slow bus release of the $\overline{BB}$ pin, bit 12 = $0.
- Address Attribute Priority Disable bit allows the Address Attribute pins, AA0-AA3, to be used in any combination, bit 14 = $1.
- All other OMR bits are selected for their defaults of $0.

The value loaded into the OMR is $004000.

Configure the memory mode of the DSP using the Status Register (SR). The SR value combines the following bits for each feature:

- Sixteen-bit Compatibility Mode enables full compatibility to object code written for the DSP56000 Family of DSPs, bit 13 = $0.
- Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, bit 19 = $1.
- All other Status Register bits are selected for their defaults of $0.

The value loaded into the SR is $080000, which is the value loaded during reset.

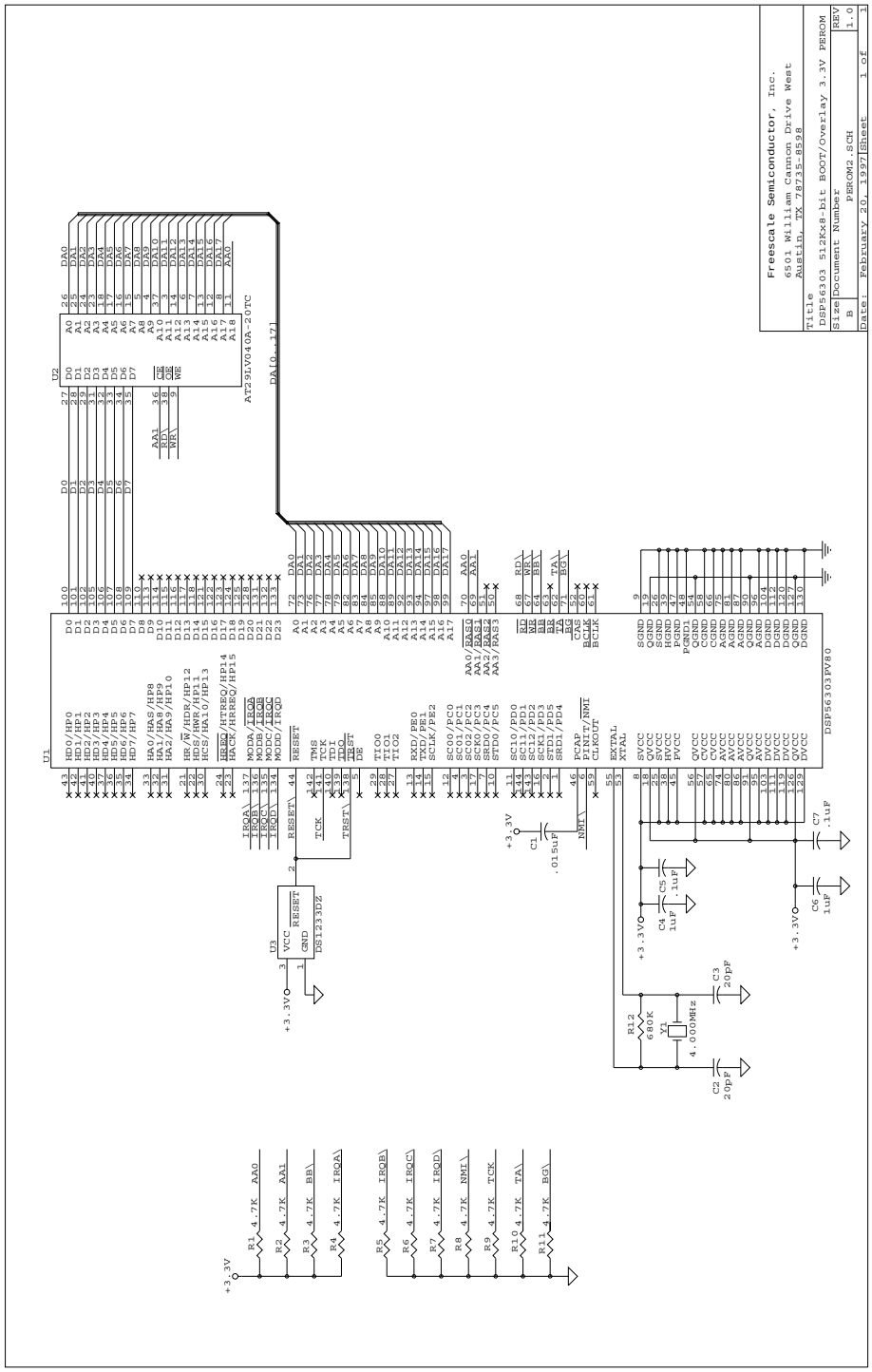

Figure 29. 512K × 8-Bit BOOT/Overlay PEROM Schematic

Example 5. 512K x 8-bit BOOT/Overlay Checksum Verify Program

```
Freescale DSP56300 Assembler Version 6.0.1.6   97-02-23 09:07:50 perom2.asm

1                       page    132,60,3,3,
2               ;
3               ; perom2.asm - Simple program to calculate the 8-Bit Checksum for
4               ; a 512K x 8-Bit block of PEROM memory using a DSP56303.
5               ; Contains: Initialization routine,
```

```
6             ; Routine to calculate 8-Bit checksum,
7             ; Routine to read checksum sector,
8             ; Routine to write checksum sector.
9             ;
10            ; Program runs in Internal P:RAM to calculate the checksum on
11            ; External X:PEROM from $100000 - $17FFFF @ 16w/s
12            ;
13
14    100000 MemStart     equ    $100000
15    180000 MemEnd       equ    $180000
16    080000 MemSize      equ    MemEnd-MemStart  ; Last Word is stored Checksum value
17
18    000100 SectorSize   equ    256              ; Size of Checksum Sector
19           SectorStart
20    17FF00              equ    MemEnd-SectorSize ; Start of Checksum Sector
21
22           ;--- Program Specific Storage Locations (X DATA SPACE)
23           NEW_CHECKSUM
24    000000              equ    $000000          ; Computed Checksum Value
25           OLD_CHECKSUM
26    000001              equ    $000001          ; Old Checksum from PEROM
27
28    000002 DataBuffer   equ    $000002          ; Start of Last Sector Storage Buffer
29                                                ; for 256 locations
30
31           ;--- DSP56303 Control Registers (X I/O SPACE)
32    FFFFFB BCR          equ    $FFFFFB          ; Bus Control Register
33    FFFFFD PCTL         equ    $FFFFFD          ; PLL Control Register
34    FFFFF9 AAR0         equ    $FFFFF9          ; Address Attribute Register 0
35    FFFFF8 AAR1         equ    $FFFFF8          ; Address Attribute Register 1
36
37           ;--- PCTL value = 0x0E0013
38    000000 prediv       equ    0             ; Predivider = 1
39    000000 lowdiv       equ    0             ; Low Power Divider = 1
40    000013 pllmul       equ    19            ; VCO Mult = 20; (19+1)*4.00MHz=80.00MHz
41    000000 crystal      equ    0             ; No, Crystal not less than 200kHz
42    000000 disXTAL      equ    0             ; No, do not disable crystal use
43    020000 pllstop      equ    $020000          ; Yes, PLL runs during STOP
44    040000 enpll        equ    $040000          ; Yes, enable PLL operation
45    080000 disclk       equ    $080000          ; Yes, disable CORE clock output
46    0E0013 PCTL_value   equ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk
47
48           ;--- AAR0 value = 0x140695
49    000001 acctype0     equ    1      ; External Memory access type = 0x1
50    000004 aahigh0      equ    $4     ; Enable AA0 pin high when selected
51    000000 aap0         equ    0      ; No, Enable AA0 pin on ext P access
52    000010 aax0         equ    $10    ; Yes, Enable AA0 pin on ext X access
53    000000 aay0         equ    0      ; No, Enable AA0 pin on ext Y access
54    000000 aswap0       equ    0      ; No, Enable address bus swap
55    000080 enpack0      equ    $80    ; Yes, Enable packing/unpacking logic
56    000600 nadd0        equ    $000600      ; Compare 6 address bits
57    140000 msadd0       equ    $140000      ; Most significant part of address,
58                                            ; $140000 - 17ffff, to compare.
59                                            ; (0001,01xx,xxxx,xxxx,xxxx,xxxx)
60    140695 AAR0_valueequ acctype0+aahigh0+aap0+aax0+aay0+aswap0+enpack0+nadd0+msadd0
61
62           ;--- AAR1 value = 0x100591
63    000001 acctype1     equ    1      ; External Memory access type = 0x1
64    000000 aahigh1      equ    0      ; Enable AA1 pin low when selected
65    000000 aap1         equ    0      ; No, Enable AA1 pin on ext 'P' accesses
```

```
       66     000010 aax1          equ    $10    ; Yes, Enable AA1 pin on ext X access
       67     000000 aay1          equ    0      ; No, Enable AA1 pin on ext Y access
       68     000000 aswap1        equ    0      ; No, Enable address bus swap
       69     000080 enpack1       equ    $80    ; Yes, Enable packing/unpacking logic
       70     000500 nadd1         equ    $000500      ; Compare 5 address bits
       71     100000 msadd1        equ    $100000      ; Most significant portion of address,
       72                                              ; $100000 - 17ffff, to compare.
       73                                              ; (0001,0xxx,xxxx,xxxx,xxxx,xxxx)
       74     100591 AAR1_value    equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       75
       76
       77             ;--- BCR value = 0x000210
       78     000010 aaa0ws        equ    $10    ; Address Attribute Area 0 w/s = 16
       79     000200 aaa1ws        equ    $200   ; Address Attribute Area 1 w/s = 16
       80     000000 aaa2ws        equ    0      ; Address Attribute Area 2 w/s = 0
       81     000000 aaa3ws        equ    0      ; Address Attribute Area 3 w/s = 0
       82     000000 defws         equ    0      ; Default Address Area w/s = 0
       83     000000 busss         equ    0      ; Bus state status = 0
       84     000000 enblh         equ    0      ; Enable Bus Lock Hold = 0
       85     000000 enbrh         equ    0            ; Enable Bus Request Hold = 0
       86     000210 BCR_value     equ    aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       87
       88             ;------------------------------------------------------------
       89             ;       Header for DSP56303 Boot Code
       90             ;------------------------------------------------------------
       91     P:0000FE             org    p:$100-2
       92
       93     P:0000FE             dc     pgm_end-perom2   ; Number of words in program
       94     P:0000FF             dc     perom2           ; Starting address for program
       95
       96
       97             ;------------------------------------------------------------
       98     P:000100             org    p:$100           ;Keep program in internal RAM
       99
      100                   perom2
      101    P:000100 0D1080      bsr    init             ; Initialize DSP
                     000011
      102
      103    P:000102 0D1080      bsr    calc_checksum    ; Calculate Checksum of PEROM
                     00001C
      104
      105    P:000104 548100      move   x:OLD_CHECKSUM,a1 ; Get PEROM Old Checksum
      106    P:000105 558000      move   x:NEW_CHECKSUM,b1 ; Get Calculated Checksum
      107    P:000106 20000D      cmp    a,b              ; Old Checksum = New Checksum?
      108    P:000107 0D104A      beq    _done            ; Yes, we are done
                     000009
      109                                                 ; No
      110    P:000109 0D1080      bsr    save_sector   ; Save contents of PEROM's Checksum
Sector
                     00002C
      111
      112    P:00010B 448000      move   x:NEW_CHECKSUM,x0 ; Get Calculated Checksum value
      113    P:00010C 447000      move   x0,x:DataBuffer+SectorSize-1; Update Checksum
location in buffer
                     000101
      114    P:00010E 0D1080      bsr    write_sector  ; Write saved PEROM Checksum sector Data
                     000038
      115                                               ; with New Checksum to PEROM
      116                   _done
```

```
117    P:000110 050C00 bra *                   ; DONE, Do a dynamic HALT
118
119
120           ;-----------------------------------------------------------
121           ; Initialization Section
122           ;-----------------------------------------------------------
123           init
124    P:000111 08F4BD movep #PCTL_value,x:PCTL ; Set PLL Control Register
                  0E0013
125    P:000113 05F43A movec #$004000,OMR       ; Disable Address Attribute Priorities
                  004000
126    P:000115 05F439 movec #$080000,SR        ; Enable 1K Cache
                  080000
127    P:000117 08F4BB movep #BCR_value,x:BCR   ; Set external wait states
                  000210
128    P:000119 08F4B9 movep #AAR0_value,x:AAR0 ; Set Address Attribute Reg0
                  140695
129    P:00011B 08F4B8 movep #AAR1_value,x:AAR1 ; Set Address Attribute Reg1
                  100591
130
131    P:00011D 00000C rts
132
133           ;-----------------------------------------------------------
134           ; Routine to Calculate 8-Bit Checksum
135           ;-----------------------------------------------------------
136           calc_checksum
137    P:00011E 05F420 move #-1,m0              ; Set LINEAR addressing mode
                  FFFFFF
138    P:000120 60F400 move #MemStart,r0        ; Set Starting Address of PEROM
                  100000
139    P:000122 70F400 move #MemSize-1,n0       ; Set to Size of Flash Checksum
                  07FFFF
140
141    P:000124 200013      clr    a
142    P:000125 20001B      clr    b
143    P:000126 540000      move   a1,x:NEW_CHECKSUM ; Initialize computed checksum ->
$000000
144    P:000127 540100      move   a1,x:OLD_CHECKSUM ; Initialize read checksum ->
$000000
145
146           ; Compute the 8-Bit Checksum
147    P:000128 44F400      move   #>$FF,x0          ; Set lower Byte Mask
                  0000FF
148
149    P:00012A 06D810      dor    n0,_ploop
                  000004
150    P:00012C 54D800      move   x:(r0)+,a1        ; Get the PEROM location Value
151    P:00012D 200046      and    x0,a              ; Mask for lower byte
152    P:00012E 200018      add    a,b               ; Compute checksum
153           _ploop
154
155    P:00012F 54E000      move   x:(r0),a1         ; Get PEROM Old Checksum value
156    P:000130 20004E      and    x0,b              ; Limit calculated Checksum to lower
byte
157    P:000131 200046      and    x0,a   ; Limit Old Checksum to lower byte
158    P:000132 550000      move   b1,x:NEW_CHECKSUM ; Save Computed Checksum
159    P:000133 540100      move   a1,x:OLD_CHECKSUM ; Save Old Checksum value
160
161    P:000134 00000C      rts
162
```

```
163           ;------------------------------------------------------------
164           ; Routine to Read a Sector of Data from the PEROM
165           ;       where the Checksum is stored
166           ;------------------------------------------------------------
167           save_sector
168     P:000135 310200     move   #DataBuffer,r1 ; Point to start of data storage buffer
169     P:000136 05F421     move   #-1,m1         ; Set modulo to linear
                FFFFFF
170
171     P:000138 60F400     move   #SectorStart,r0 ; Point to Start of Sector in PEROM
                17FF00
172     P:00013A70F400      move   #SectorSize,n0 ; Get size of Sector
                000100
173     P:00013C 05F420     move   #-1,m0         ; Set modulo to linear
                FFFFFF
174
175     P:00013E 44F400     move   #>$FF,x0       ; Lower Byte Mask
                0000FF
176
177     P:000140 06D810     dor    n0,_read_loop
                000004
178     P:000142 54D800     move   x:(r0)+,a1     ; Read a word from the PEROM sector
179     P:000143 200046     and    x0,a           ; Mask data to lower byte, ie $0000xx
180     P:000144 545900     move   a1,x:(r1)+     ; Save a word in storage buffer
181           _read_loop
182
183     P:000145 00000C rts
184
185           ;------------------------------------------------------------
186           ; Routine to Place PEROM into ERASE/WRITE MODE and send it the sector of
Data
187           ;------------------------------------------------------------
188           write_sector
189     P:000146 310200     move   #DataBuffer,r1 ; Point to start of data storage buffer
190     P:000147 05F421     move   #-1,m1           ; Set modulo to linear
                FFFFFF
191
192     P:000149 60F400     move   #SectorStart,r0  ; Point to Start of Sector in PEROM
                17FF00
193     P:00014B 70F400     move   #SectorSize,n0   ; Get size of sector
                000100
194     P:00014D 05F420     move   #-1,m0           ; Set modulo to linear
                FFFFFF
195
196           ;-- Place PEROM into ERASE/WRITE MODE
197     P:00014F 44F400     move   #>$AA,x0
                0000AA
198     P:000151 447000     move   x0,x:MemStart+$5555 ; Unlock PEROM Cycle #1
                105555
199
200     P:000153 44F400     move   #>$55,x0
                000055
201     P:000155 447000     move   x0,x:MemStart+$2AAA ; Unlock PEROM Cycle #2
                102AAA
202
203     P:000157 44F400     move   #>$A0,x0
                0000A0
204     P:000159 447000     move   x0,x:MemStart+$5555 ; Send PEROM Write Command
                105555
205                                          ; -- PEROM Writes now enabled
```

```
       206           ;-- Send a Sector of data to the PEROM
       207     P:00015B 06D810     dor    n0,_write_loop
                     000004
       208     P:00015D 54D900     move   x:(r1)+,a1    ; Read byte from storage buffer
       209     P:00015E 545800     move   a1,x:(r0)+    ; Write the byte to PEROM
       210     P:00015F 000000     nop
       211           _write_loop
       212                                              ; -- Now in PEROM Data Protect State
       213           ;-- Wait till ERASE/WRITE Cycle is complete
       214     P:000160 205000     move   (r0)-           ; Point to last PEROM location
       215     P:000161 44F400     move   #>$FF,x0        ; Lower Byte Mask
                     0000FF
       216
       217           _write_wait
       218     P:000163 55E000     move   x:(r0),b1       ; Get value at PEROM location
       219     P:000164 20004E     and    x0,b            ; Mask for lower byte
       220     P:000165 20000D     cmp    a,b             ; Last value written = value in
PEROM?
       221     P:000166 0527DD     bne    _write_wait     ; No, wait until it is
       222                                              ; Yes
       223     P:000167 00000C     rts
       224
       225           ;---------------------------------------------------------
       226           pgm_end
       227
       228           end     perom2

       0       Errors
       0       Warnings
```

6.3 32K × 16-Bit X Data and Y Data PEROM Example

The 32K × 16-bit X data and 32K × 16-bit Y data memory PEROM implementation uses the Atmel AT29LV1024-20. See **Figure 30** for the memory map layout and **Figure 31** for the block diagram. Sixteen-bit coefficient and data arrays are stored externally in non-volatile storage so that results from previous operations can be stored before power is removed and recalled on power-up.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 200 nS Flash memory, sixteen wait states are required. This 3.3 V device is organized as 64K × 16-bits with a 200 nS access time. One memory device is used to achieve the 16-bit wide X data and Y data buses. **Figure 34** shows the schematic for this example.

AA1 is configured to select the PEROM device when an X data or Y data space access in the address range $100000 –$107FFF. AA0 is configured to select the 64K PEROM between 32K of X data space and 32K of Y data space. The address attribute implementation details are discussed in **Section 6.3.2**, *DSP56303 Port A Timing Requirements and Register Settings*, on page 75 and the program listing in **Example 6** on page 78.

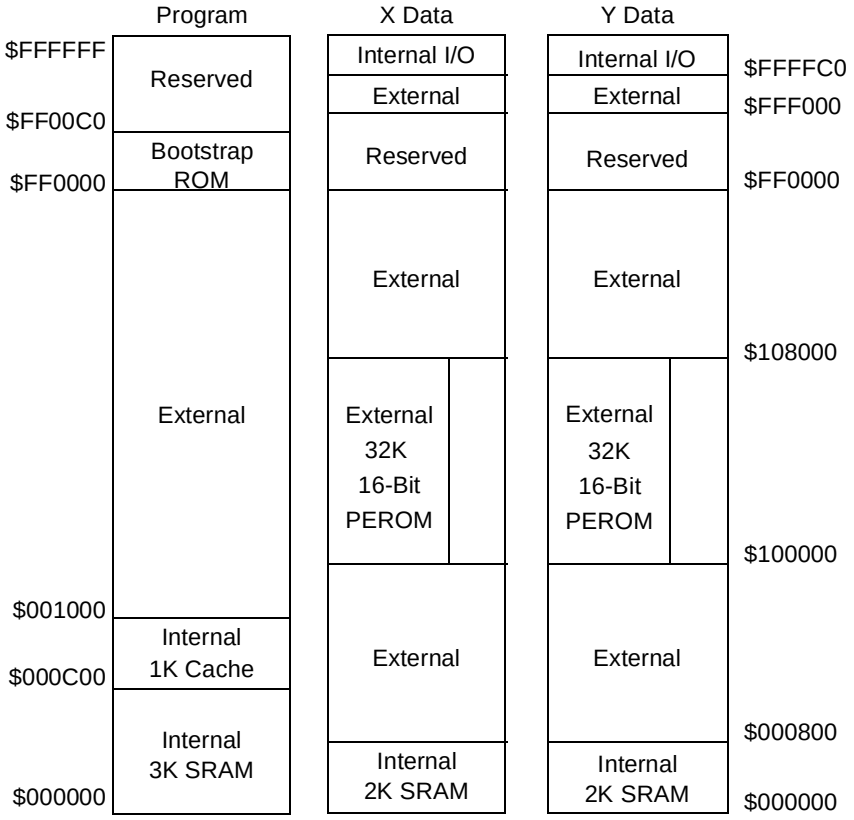

Figure 30. 32K × 16 X Data and 32K × 16 Y Data Memory Map

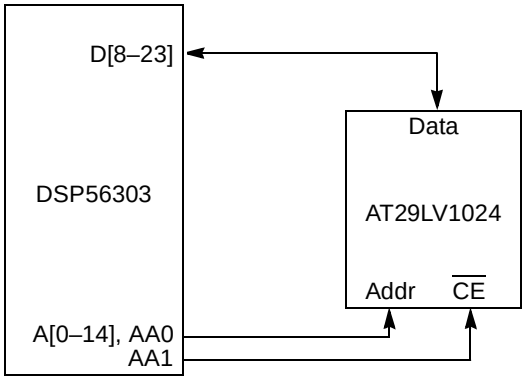

Figure 31. 32K × 16-Bit X Data and 32K × 16 Y Data PEROM Block Diagram

6.3.1 PEROM Timing Requirements

Following are the timing requirements for the AT29LV1024-20 64K × 16-bit 200 nS PEROM. **Table 13** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 32**.

Table 13. AT29LV1024-20 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Read Cycle Time	t_{RC}	200 nS	—
Address to Output Delay	t_{ACC}	—	200 nS
Chip Enable to Output Delay	t_{CE}	—	200 nS

Table 13. AT29LV1024-20 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Output Enable to Output Delay	t_{OE}	—	100 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0 nS	—

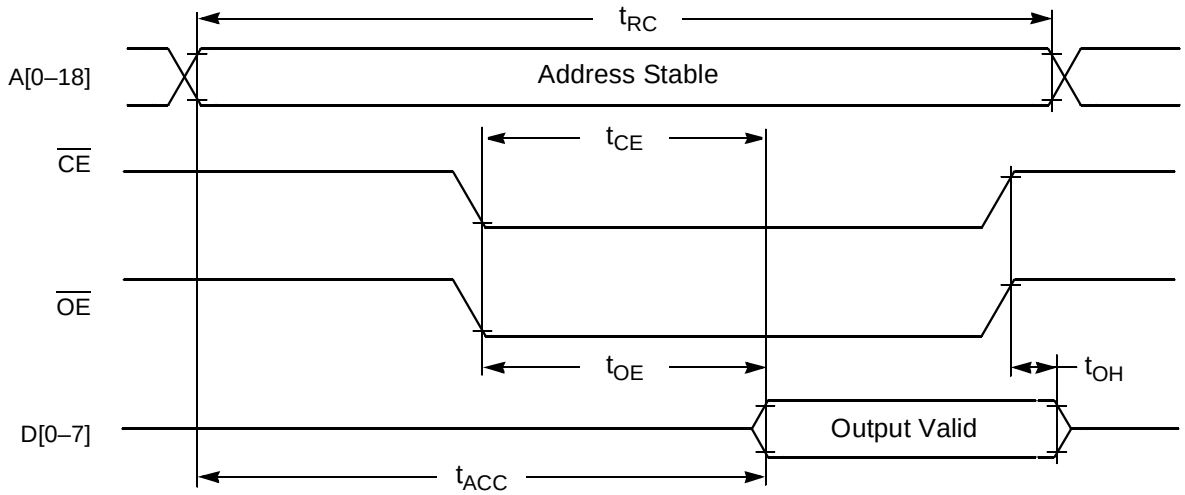

Figure 32. AT29LV1024 Memory Read Cycle Timing Diagram.

Table 14 shows the memory write timing specification values used in the memory write cycle timing diagram, **Figure 33**.

Table 14. AT29LV1024-20 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write Cycle Time (during Program)	t_{WC}	—	20 mS
Address set-up Time	t_{AS}	0 nS	—
$\overline{CE}$ set-up Time	t_{CS}	0 nS	—
Write Pulse Width	t_{WP}	200 nS	—
Data set-up Time	t_{DS}	100 nS	—
Data Hold Time	t_{DH}	0 nS	—

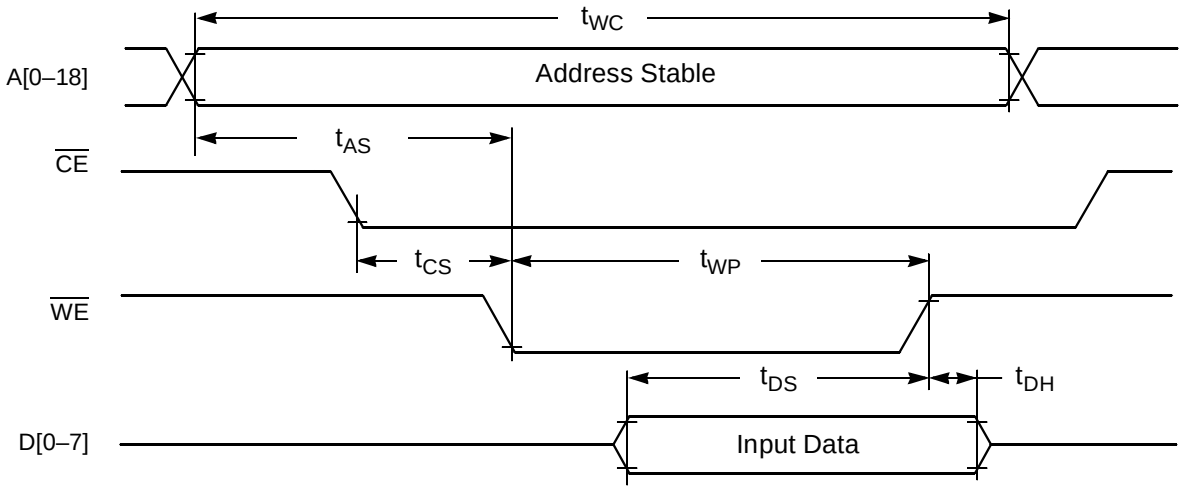

Figure 33. AT29LV1024 Memory Write Cycle Timing Diagram.

Non-volatile memory locations in the PEROM cannot be programmed individually, but must be programmed as a sector of 128 locations. The AT29LV1024 device is organized as 512 sectors of 128 16-bit words. To program a memory location, the entire sector it resides in must be programmed. No separate erase cycle exists; the program cycle erases and programs the entire sector as one operation.

Writing to a PEROM memory location requires writing a complete sector, or 128 words of data, to the PEROM in the following sequence:

1. Write $AAAA00 to location $5555 relative to the PEROM.
2. Write $555500 to location $2AAA relative to the PEROM.
3. Write $A0A000 to location $5555 relative to the PEROM.
4. Write 128 words of 16-bit data to the sector in the PEROM.
5. Read the last address in the sector until data written = data read.

Note: Only upper sixteen bits of a 24-bit word are significant.

6.3.2 DSP56303 Port A Timing Requirements and Register Settings

For optimal use of the 32K × 16-bit X data and 32K × 16-bit Y data space memory configuration, set up the following DSP control registers.

Set the core speed of the DSP for optimum processor and memory performance using the DSP PLL and Clock Generation (PCTL) register. For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. The PCTL value combines the following bits for each feature:

- Desired core frequency = 80 MHz
- Given the external frequency = 4.000 MHz
- Predivider value = 1, bits 20–23 = $0
- Low-power divider value = 1, bits 12–14 = $0
- V_{CO} multiplication value = 20, bits 0–11 = $013

- Crystal less than 200 kHz, bit 15 = 0
- Disable XTAL drive output, bit 16 = 0
- PLL runs during STOP, bit 17 = 1
- Enable PLL operation, bit 18 = 1
- Disable core clock output, bit 19 = 1

The value loaded into PCTL is $0E0013.

AA1 enables, via PEROM $\overline{CE}$, external 32K PEROM bank accesses in the address range $100000–$107FFF during X data and Y data space requests. Configure the memory address space requirements for AA1 using the Address Attribute Register 1 (AAR1). The AAR1 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 0.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the AA pin during external Y data space accesses, bit 5 = 1.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $9.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR1 is $100931.

AA0 switches, via PEROM A15, between X data and Y data space 32K PEROM bank accesses in the address range $100000–$107FFF during X data and Y data space requests. Configure the memory address space requirements for AA0 using Address Attribute Register 0 (AAR0). The AAR0 value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = $1.
- Pull the AA pin high when selected, bit 2 = 1.
- Activate the AA pin during external program space accesses, bit 3 = 0.
- Activate the AA pin during external X data space accesses, bit 4 = 1.
- Activate the ~~AA~~ pin during external Y data space accesses, bit 5 = 0.
- Move the eight least significant bits of the address to the eight most significant bits of the external address bus, bit 6 = 0.
- Enable the internal packing/unpacking logic during external DMA accesses, AAR0 bit 7 = 0.
- Specify the number of address bits to compare, bits 8–11 = $9.
- Specify the most significant portion of the address to compare, bits 12–23 = $100.

The value loaded into the AAR0 is $100915; the value loaded into AAR2 and AAR3 is $00000.

Select the proper number of wait states for the memory configuration using the Bus Control Register (BCR). The BCR value combines the following bits for each feature:

- Address attribute area 0 wait states, bits 0–4 = $10.
- Address attribute area 1 wait states, bits 5–9 = $10.
- Address attribute area 2 wait states, bits 10–12 = $0.
- Address attribute area 3 wait states, bits 13–15 = $0.
- Default address area wait states, bits 16–20 = $0.
- Bus state status, bit 21 = 0.
- Enable Bus Lock Hold, bit 22 = 0.
- Enable Bus Request Hold, bit 23 =0.

The value loaded into the BCR is $000210.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR value combines the following bits for each feature:

- MA–MD bits specify DSP operating mode, bits 0–3 = $0.
- External Bus Disable bit disables the external bus controller for power conservation when external memory is not used, bit 4 = $0.
- Memory Switch Mode bit reconfigures internal memory spaces, bit 7 = $0.
- Transfer Acknowledge Synchronize Select bit selects the synchronization method for the Transfer Acknowledge, TA, pin, bit 11 = $0.
- Bus Release Timing bit selects between a fast and slow bus release of the $\overline{BB}$ pin, bit 12 = $0.
- Address Attribute Priority Disable bit allows the Address Attribute pins, AA0–AA3, to be used in any combination, bit 14. = $1.
- All other OMR bits are selected for their defaults of $0.

The value loaded into the OMR is $004000.

Configure the memory mode of the DSP using the Status Register (SR). The SR value combines the following bits for each feature:

- Sixteen-Bit Compatibility Mode enables full compatibility to object code written for the DSP56000 Family of DSPs, Bit 13 = $0.
- Instruction Cache Enable bit enables the instruction cache controller and changes the last 1K of internal program memory into cache memory, Bit 19 = $1.
- All other Status Register bits are selected for their defaults of $0.

The value loaded into the SR is $080000, which is the value loaded during reset.

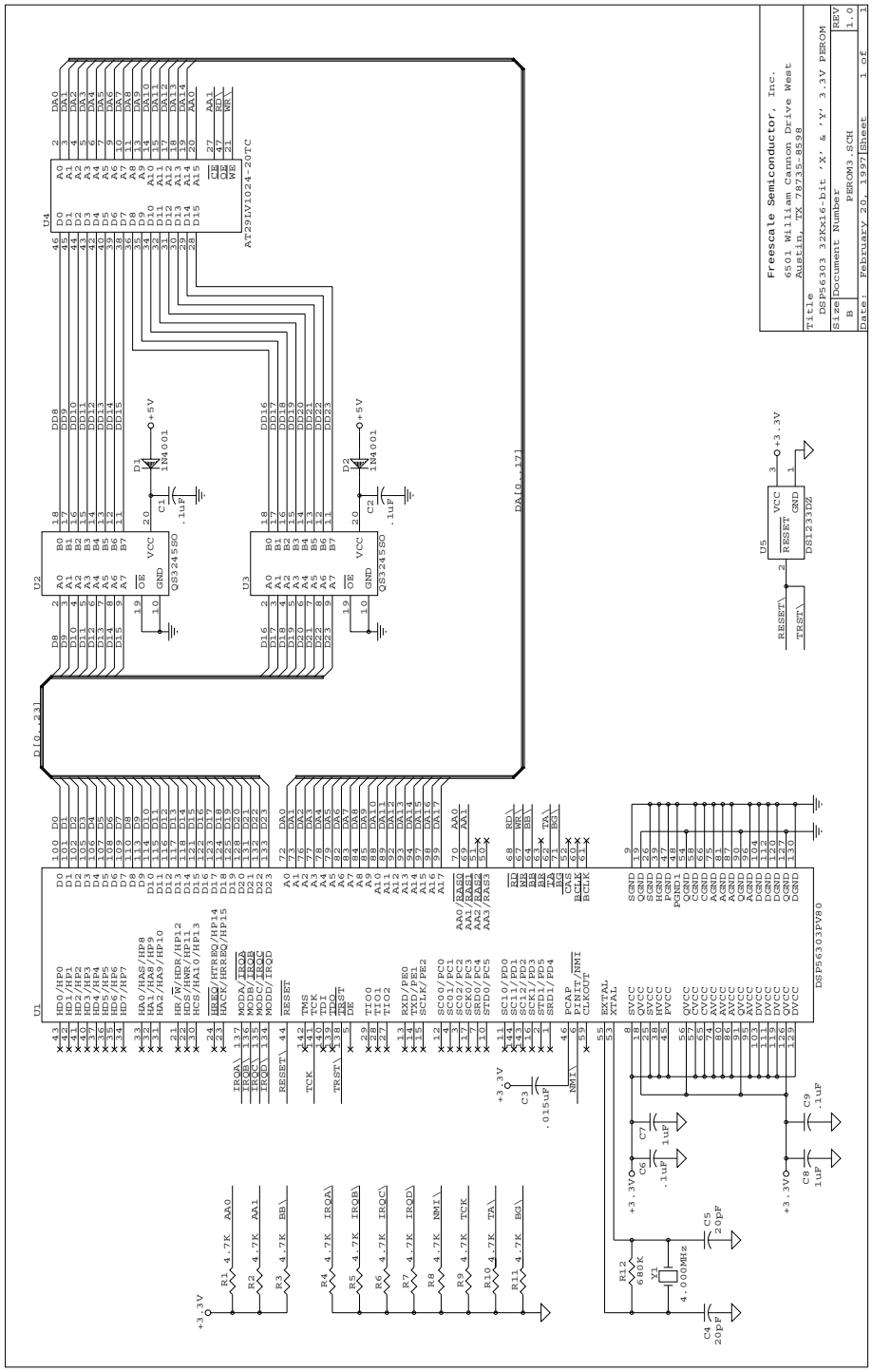

Figure 34. 32K × 16-Bit X and 32K × 16-Bit Y Data Space PEROM Schematic

Example 6. 32K x 16-bit X and Y Space PEROM Checksum Verify Program

```
Freescale DSP56300 Assembler Version 6.0.1.6   97-02-23 09:26:58 perom3.asm

1                           page    132,60,3,3,
2                    ;
3                    ; perom3.asm, program to calculate the 16-Bit Checksum for
4                    ; two 32K x 16-Bit blocks of PEROM memory using a DSP56303.
```

```
                        ; Contains: Initialization routine,
                        ;                               Routine to calculate 16-Bit checksum,
                        ;                               Routine to read checksum sector,
                        ;                               Routine to write checksum sector.
                        ;
                        ; The program runs in Internal P:RAM to calculate the checksum on
                        ;       External X:PEROM from $100000 - $107FFF @ 16w/s and
                        ;       External Y:PEROM from $100000 - $107FFF @ 16w/s.
                        ;

     100000 MemStart     equ    $100000
     180000 MemEnd       equ    $180000
     080000 MemSize      equ    MemEnd-MemStart  ; Last Word is stored Checksum value

     000080 SectorSize   equ    128              ; Size of Checksum Sector
            SectorStart
     17FF80              equ    MemEnd-SectorSize ; Start of Checksum Sector

            ;--- Program Specific Storage Locations (X DATA SPACE)
            NEW_X_CHECKSUM
     000000              equ    $000000          ; X SPACE Computed Checksum Value
            OLD_X_CHECKSUM
     000001              equ    $000001          ; X SPACE Old Checksum from PEROM
            NEW_Y_CHECKSUM
     000002              equ    $000002          ; Y SPACE Computed Checksum Value
            OLD_Y_CHECKSUM
     000003              equ    $000003          ; Y SPACE Old Checksum from PEROM

     000004 DataBuffer   equ    $000004          ; Start of Last Sector Storage Buffer
                                                 ; for 128 locations

            ;--- DSP56303 Control Registers (X I/O SPACE)
     FFFFFB BCR          equ    $FFFFFB       ; Bus Control Register
     FFFFFD PCTL         equ    $FFFFFD       ; PLL Control Register
     FFFFF9 AAR0         equ    $FFFFF9       ; Address Attribute Register 0
     FFFFF8 AAR1         equ    $FFFFF8       ; Address Attribute Register 1

            ;--- PCTL value = 0x0E0013
     000000 prediv       equ    0             ; Pre-Divider = 1
     000000 lowdiv       equ    0             ; Low Power Divider = 1
     000013 pllmul       equ    19            ; VCO Mult = 20; (19+1)*4.00MHz=80.00MHz
     000000 crystal      equ    0             ; No, Crystal not less than 200kHz
     000000 disXTAL      equ    0             ; No, do not disable crystal use
     020000 pllstop      equ    $020000       ; Yes, PLL runs during STOP
     040000 enpll        equ    $040000       ; Yes, enable PLL operation
     080000 disclk       equ    $080000       ; Yes, disable CORE clock output
     0E0013 PCTL_value   equ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk

            ;--- AAR0 value = 0x100915
     000001 acctype0     equ    1             ; External Memory access type = 0x1
     000004 aahigh0      equ    $4            ; Enable AA0 pin high when selected
     000000 aap0         equ    0             ; No, Enable AA0 pin on ext P access
     000010 aax0         equ    $10           ; Yes, Enable AA0 pin on ext X access
     000000 aay0         equ    0             ; No, Enable AA0 pin on ext Y access
     000000 aswap0       equ    0             ; No, Enable address bus swap
     000000 enpack0      equ    0             ; No, Enable packing/unpacking logic
     000900 nadd0        equ    $000900       ; Compare 9 address bits
     100000 msadd0       equ    $100000       ; Most significant portion of address,
                                                ; $100000 - 107fff, to compare.
```

```
       64                                          ; (0001,0000,0xxx,xxxx,xxxx,xxxx)
       65     100915 AAR0_value   equ
acctype0+aahigh0+aap0+aax0+aay0+aswap0+enpack0+nadd0+msadd0
       66
       67           ;--- AAR1 value = 0x100931
       68     000001 acctype1     equ    1            ; External Memory access type = 0x1
       69     000000 aahigh1      equ    0            ; Enable AA1 pin to be low when selected
       70     000000 aap1         equ    0            ; No, Enable AA1 pin on ext P access
       71     000010 aax1         equ    $10          ; Yes, Enable AA1 pin on ext X access
       72     000020 aay1         equ    $20          ; Yes, Enable AA1 pin on ext Y access
       73     000000 aswap1       equ    0            ; No, Enable address bus swap
       74     000000 enpack1      equ    0            ; No, Enable packing/unpacking logic
       75     000900 nadd1        equ    $000900      ; Compare 9 address bits
       76     100000 msadd1       equ    $100000      ; Most significant portion of address,
       77                                             ; $100000 - 107fff, to compare.
       78                                             ; (0001,0000,0xxx,xxxx,xxxx,xxxx)
       79     100931 AAR1_value   equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       80
       81
       82           ;--- BCR value = 0x000210
       83     000010 aaa0ws       equ    $10          ; Address Attribute Area 0 w/s = 16
       84     000200 aaa1ws       equ    $200         ; Address Attribute Area 1 w/s = 16
       85     000000 aaa2ws       equ    0            ; Address Attribute Area 2 w/s = 0
       86     000000 aaa3ws       equ    0            ; Address Attribute Area 3 w/s = 0
       87     000000 defws        equ    0            ; Default Address Area w/s = 0
       88     000000 busss        equ    0            ; Bus state status = 0
       89     000000 enblh        equ    0            ; Enable Bus Lock Hold = 0
       90     000000 enbrh        equ    0            ; Enable Bus Request Hold = 0
       91     000210 BCR_value    equ    aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       92
       93           ;---------------------------------------------------------------------
       94     P:000100            org    p:$100           ; Keep the program in internal RAM
       95
       96           perom3
       97     P:0001000D1080             bsr    init      ; Initialize DSP
                   00001F
       98
       99     P:0001020D1080      bsr    calc_x_checksum  ; Calculate Checksum of X:PEROM
                   00002A
      100
      101    P:000104548100      move   x:OLD_X_CHECKSUM,a1 ; Get X:PEROM Old Checksum value
      102    P:000105558000      move   x:NEW_X_CHECKSUM,b1 ; Get Calculated Checksum value
      103    P:00010620000D      cmp    a,b               ; Old Checksum = New Checksum?
      104    P:0001070D104A      beq    _do_y_perom       ; Yes, we are done
                   000009
      105                                                 ; No
      106    P:0001090D1080      bsr    save_x_sector ; Save contents of X:PEROM Checksum
Sector
                   00003A
      107
      108    P:00010B448000      move   x:NEW_X_CHECKSUM,x0 ; Get Calculated Checksum value
      109    P:00010C447000 move x0,x:DataBuffer+SectorSize-1; Update Checksum location in
buffer
                   000083
      110    P:00010E0D1080      bsr    write_x_sector ; Write saved X:PEROM Checksum sector
Data
                   000045
      111                                           ; with New Checksum value to X:PEROM
      112          ;---------------------------------------------------------------------
```

```
       113           _do_y_perom
       114    P:0001100D1080      bsr    calc_y_checksum  ; Calculate Checksum of Y:PEROM
                     000064
       115
       116    P:000112548300      move   x:OLD_Y_CHECKSUM,a1 ; Get Y:PEROM Old Checksum value
       117    P:000113558200      move   x:NEW_Y_CHECKSUM,b1 ; Get Calculated Checksum value
       118    P:00011420000D      cmp    a,b               ; Old Checksum = New Checksum?
       119    P:0001150D104A      beq    _done             ; Yes, we are done
                     000009
       120                                                  ; No
       121    P:0001170D1080      bsr    save_y_sector ; Save contents of Y:PEROM Checksum
Sector
                     000074
       122
       123    P:000119448200      move   x:NEW_Y_CHECKSUM,x0 ; Get Calculated Checksum value
       124    P:00011A447000      move   x0,x:DataBuffer+SectorSize-1; Update Checksum
location in buffer
                     000083
       125    P:00011C0D1080      bsr    write_y_sector ; Write saved Y:PEROM Checksum sector
Data
                     00007F
       126                                             ; with New Checksum value to Y:PEROM
       127
       128           _done
       129    P:00011E050C00      bra    *              ; DONE, Do a dynamic HALT
       130
       131
       132           ;---------------------------------------------------------------------
       133           ; Initialization Section
       134           ;---------------------------------------------------------------------
       135           init
       136    P:00011F08F4BD      movep  #PCTL_value,x:PCTL ; Set PLL Control Register
                     0E0013
       137    P:00012105F43A      movec  #$004000,OMR  ; Disable Address Attribute Priorities
                     004000
       138    P:00012305F439      movec  #$080000,SR       ; Enable 1K Cache
                     080000
       139    P:00012508F4BB      movep  #BCR_value,x:BCR ; Set external wait states
                     000210
       140    P:00012708F4B9      movep  #AAR0_value,x:AAR0; Set Address Attribute Reg0
                     100915
       141    P:00012908F4B8      movep  #AAR1_value,x:AAR1 ; Set Address Attribute Reg1
                     100931
       142
       143    P:00012B00000C      rts
       144
       145           ;---------------------------------------------------------------------
       146           ; Routine to Calculate 16-Bit Checksum in X:PEROM
       147           ;---------------------------------------------------------------------
       148           calc_x_checksum
       149    P:00012C05F420      move   #-1,m0           ; Set LINEAR addressing mode
                     FFFFFF
       150    P:00012E60F400      move   #MemStart,r0     ; Set Starting Address of PEROM
                     100000
       151    P:00013070F400      move   #MemSize-1,n0    ; Set to Size of Flash - Checksum
                     07FFFF
       152
       153    P:000132200013      clr    a
       154    P:00013320001B      clr    b
```

```
       155    P:000134540000      move   a1,x:NEW_X_CHECKSUM ; Initialize computed checksum ->
$000000
       156    P:000135540100      move   a1,x:OLD_X_CHECKSUM ; Initialize read checksum ->
$000000
       157
       158           ; Compute the 16-Bit Checksum
       159    P:00013644F400      move   #$FFFF00,x0      ; Set 16-Bit Mask
              FFFF00
       160
       161    P:00013806D810      dor    n0,_ploop
              000004
       162    P:00013A54D800      move   x:(r0)+,a1       ; Get the PEROM location Value
       163    P:00013B200046      and    x0,a             ; Mask for lower byte
       164    P:00013C200018      add    a,b              ; Compute checksum
       165           _ploop
       166
       167    P:00013D54E000      move   x:(r0),a1        ; Get PEROM's Old Checksum value
       168    P:00013E20004E      and    x0,b             ; Limit calculated Checksum
       169    P:00013F200046      and    x0,a             ; Limit Old Checksum value
       170    P:000140550000      move   b1,x:NEW_X_CHECKSUM ; Save Computed Checksum value
       171    P:000141540100      move   a1,x:OLD_X_CHECKSUM ; Save the Old Checksum value
       172
       173    P:00014200000C      rts
       174
       175           ;---------------------------------------------------------------------
       176           ; Routine to Read a Sector of Data from the X:PEROM
       177           ;       where the Checksum is stored
       178           ;---------------------------------------------------------------------
       179           save_x_sector
       180    P:000143310400      move   #DataBuffer,r1 ; Point to start of data storage buffer
       181    P:00014405F421      move   #-1,m1         ; Set modulo to linear
              FFFFFF
       182
       183    P:00014660F400      move   #SectorStart,r0 ; Point to Start of Sector in PEROM
              17FF80
       184    P:000148388000      move   #SectorSize,n0 ; Get size of Sector
       185    P:00014905F420      move   #-1,m0         ; Set modulo to linear
              FFFFFF
       186
       187    P:00014B44F400      move   #$FFFF00,x0   ; Set 16-Bit Mask
              FFFF00
       188
       189    P:00014D06D810      dor    n0,_read_loop
              000004
       190    P:00014F54D800      move   x:(r0)+,a1    ; Read a word from the PEROM sector
       191    P:000150200046      and    x0,a          ; Mask data, ie $xxxx00
       192    P:000151545900      move   a1,x:(r1)+    ; Save off a word in storage buffer
       193           _read_loop
       194
       195    P:000152 00000C rts
       196
       197           ;---------------------------------------------------------------------
       198           ; Routine to Place X:PEROM into ERASE/WRITE MODE
       199           ;       and send it the sector of Data
       200           ;---------------------------------------------------------------------
       201           write_x_sector
       202    P:000153310400      move   #DataBuffer,r1 ; Point to start of data storage buffer
       203    P:00015405F421      move   #-1,m1           ; Set modulo to linear
              FFFFFF
       204
```

```
205    P:00015660F400      move   #SectorStart,r0  ; Point to Start of Sector in PEROM
              17FF80
206    P:000158388000      move   #SectorSize,n0   ; Get size of sector
207    P:00015905F420      move   #-1,m0           ; Set modulo to linear
              FFFFFF
208
209           ;-- Place PEROM into ERASE/WRITE MODE
210    P:00015B44F400      move   #$AAAA00,x0
              AAAA00
211    P:00015D4C7000      move   x0,y:MemStart+$5555; Unlock PEROM Cycle #1
              105555
212
213    P:00015F44F400      move   #$555500,x0
              555500
214    P:0001614C7000      move   x0,y:MemStart+$2AAA; Unlock PEROM Cycle #2
              102AAA
215
216    P:00016344F400      move   #$A0A000,x0
              A0A000
217    P:0001654C7000      move   x0,y:MemStart+$5555; Send PEROM Write Command
              105555
218                                           ; -- PEROM Writes are now enabled
219           ;-- Send a Sector of data to the PEROM
220    P:00016706D810      dor    n0,_write_loop
              000004
221    P:00016954D900      move   x:(r1)+,a1       ; Read a byte from the storage buffer
222    P:00016A545800      move   a1,x:(r0)+       ; Write the byte to PEROM
223    P:00016B000000      nop
224                 _write_loop
225                                              ; -- Now in PEROM Data Protect State
226           ;-- Wait till ERASE/WRITE Cycle is complete
227    P:00016C205000      move   (r0)-            ; Point to last PEROM's location
228    P:00016D44F400      move   #$FFFF00,x0      ; Set 16-Bit Mask
              FFFF00
229
230                 _write_wait
231    P:00016F55E000      move   x:(r0),b1        ; Get current value at PEROM location
232    P:00017020004E      and    x0,b             ; Mask for lower byte
233    P:00017120000D      cmp    a,b              ; Last value written = value in
PEROM?
234    P:0001720527DD      bne    _write_wait      ; No, wait until it is
235                                             ; Yes
236    P:00017300000C      rts
237
238           ;---------------------------------------------------------------------
239           ; Routine to Calculate 16-Bit Checksum in Y:PEROM
240           ;---------------------------------------------------------------------
241           calc_y_checksum
242    P:00017405F420      move   #-1,m0           ; Set LINEAR addressing mode
              FFFFFF
243    P:00017660F400      move   #MemStart,r0     ; Set Starting Address of PEROM
              100000
244    P:00017870F400      move   #MemSize-1,n0    ; Set to Size of Flash - Checksum
              07FFFF
245
246    P:00017A200013      clr    a
247    P:00017B20001B      clr    b
248    P:00017C540200      move   a1,x:NEW_Y_CHECKSUM ; Initialize computed checksum ->
$000000
```

```
       249    P:00017D540300      move   a1,x:OLD_Y_CHECKSUM ; Initialize read checksum ->
$000000
       250
       251                  ; Compute the 16-Bit Checksum
       252    P:00017E44F400      move   #$FFFF00,x0      ; Set 16-Bit Mask
              FFFF00
       253
       254    P:00018006D810      dor    n0,_ploop
              000004
       255    P:0001825CD800      move   y:(r0)+,a1       ; Get the PEROM location Value
       256    P:000183200046      and    x0,a             ; Mask for lower byte
       257    P:000184200018      add    a,b              ; Compute checksum
       258                  _ploop
       259
       260    P:0001855CE000      move   y:(r0),a1        ; Get PEROM's Old Checksum value
       261    P:00018620004E      and    x0,b             ; Limit calculated Checksum
       262    P:000187200046      and    x0,a             ; Limit Old Checksum value
       263    P:000188550200      move   b1,x:NEW_Y_CHECKSUM ; Save Computed Checksum value
       264    P:000189540300      move   a1,x:OLD_Y_CHECKSUM ; Save the Old Checksum value
       265
       266    P:00018A00000C      rts
       267
       268          ;-------------------------------------------------------------
       269          ; Routine to Read a Sector of Data from the Y:PEROM
       270          ;       where the Checksum is stored
       271          ;-------------------------------------------------------------
       272          save_y_sector
       273    P:00018B310400      move   #DataBuffer,r1 ; Point to start of data storage buffer
       274    P:00018C05F421      move   #-1,m1           ; Set modulo to linear
              FFFFFF
       275
       276    P:00018E60F400      move   #SectorStart,r0  ; Point to Start of Sector in PEROM
              17FF80
       277    P:000190388000      move   #SectorSize,n0   ; Get size of Sector
       278    P:00019105F420      move   #-1,m0           ; Set modulo to linear
              FFFFFF
       279
       280    P:00019344F400      move   #$FFFF00,x0      ; Set 16-Bit Mask
              FFFF00
       281
       282    P:00019506D810      dor    n0,_read_loop
              000004
       283    P:0001975CD800      move   y:(r0)+,a1       ; Read a word from the PEROM sector
       284    P:000198200046      and    x0,a             ; Mask data, ie $xxxx00
       285    P:000199545900      move   a1,x:(r1)+       ; Save off a word in storage buffer
       286                  _read_loop
       287
       288    P:00019A00000C      rts
       289
       290          ;---------------------------------------------------------------------
       291          ; Routine to Place Y:PEROM into ERASE/WRITE MODE
       292          ;       and send it the sector of Data
       293          ;---------------------------------------------------------------------
       294                  write_y_sector
       295    P:00019B310400      move   #DataBuffer,r1 ; Point to start of data storage buffer
       296    P:00019C05F421      move   #-1,m1           ; Set modulo to linear
              FFFFFF
       297
       298    P:00019E60F400      move   #SectorStart,r0  ; Point to Start of Sector in PEROM
              17FF80
```

```
299    P:0001A0388000      move   #SectorSize,n0   ; Get size of sector
300    P:0001A105F420      move   #-1,m0           ; Set modulo to linear
              FFFFFF
301
302           ;-- Place PEROM into ERASE/WRITE MODE
303    P:0001A344F400      move   #$AAAA00,x0
              AAAA00
304    P:0001A54C7000      move   x0,y:MemStart+$5555; Unlock PEROM Cycle #1
              105555
305
306    P:0001A744F400      move   #$555500,x0
              555500
307    P:0001A94C7000      move   x0,y:MemStart+$2AAA; Unlock PEROM Cycle #2
              102AAA
308
309    P:0001AB44F400      move   #$A0A000,x0
              A0A000
310    P:0001AD4C7000      move   x0,y:MemStart+$5555; Send PEROM Write Command
              105555
311                                                ; -- PEROM Writes are now enabled
312           ;-- Send a Sector of data to the PEROM
313    P:0001AF06D810      dor    n0,_write_loop
              000004
314    P:0001B154D900      move   x:(r1)+,a1       ; Read a byte from the storage buffer
315    P:0001B25C5800      move   a1,y:(r0)+       ; Write the byte to PEROM
316    P:0001B3000000      nop
317           _write_loop
318                                                ; -- Now in PEROM Data Protect State
319           ;-- Wait till ERASE/WRITE Cycle is complete
320    P:0001B4205000      move   (r0)-            ; Point to last PEROM's location
321    P:0001B544F400      move   #$FFFF00,x0      ; Set 16-Bit Mask
              FFFF00
322
323                  _write_wait
324    P:0001B75DE000      move   y:(r0),b1     ; Get current value at PEROM location
325    P:0001B820004E      and    x0,b          ; Mask for lower byte
326    P:0001B920000D      cmp    a,b           ; Last value written = value in PEROM?
327    P:0001BA0527DD      bne    _write_wait   ; No, wait until it is
328                                             ; Yes
329    P:0001BB00000C      rts
330
331                 ;---------------------------------------------------------------
332
333                         end     perom3

0      Errors
0      Warnings
```

7 Advantages

Flash and PEROM memory offer non-volatile memory storage with a reasonable speed and flexible memory configuration capabilities for the DSP56300 Family. Flash and PEROM memory allow the DSP to load programs and run immediately after reset, as required in an embedded application. Programs stored in Flash or PEROM non-volatile memory can be read and run directly as 24-bit data, or indirectly as 8-bit data by using the DSP DMA controller to access, pack, and store the data into DSP program memory. Programs and data can be stored in Flash or PEROM non-volatile memory without the need to remove the device from the board, promoting convenient program and data updates.

Because PEROM devices use smaller sector sizes than Flash devices, less system RAM is required to update sectors in PEROM devices than in Flash devices. PEROM sector sizes typically range from 64–512 memory locations, but Flash sectors range from 8–64 K memory locations. For bulk Flash memory, the whole device is in effect one sector. Less RAM is therefore needed to read a PEROM sector, make the required changes, and write the modified sector back to the PEROM device. However. Flash devices have the advantage that a single (unprogrammed) location can be programmed without saving and reloading the entire sector. Therefore, data can be added progressively to Flash memory without sector saving and reloading.

7.1 Disadvantages

Flash and PEROM memory have a limited number of reprogram cycles. The reprogram cycle limit for Flash devices is approximately 100,000 write/erase cycles. The reprogram cycle limit for PEROM devices is approximately 10,000 write/erase cycles. If Flash and PEROM memory are used properly and are not reprogrammed continually, these reprogram limits suffice for most applications.

Flash and PEROM memories require a special program to execute when the memories are written or programmed. This programming process requires approximately two orders of magnitude longer than a read access.

Some Flash memory architectures have large sectors that require special storage or handling when reprogramming is desired. All data to be written to Flash memory must be stored in local memory or transferred from an external source during the reprogramming process.

Flash and PEROM memories are not as fast as fast static RAM, so a DSP slows down for Flash and PEROM accesses. Data read access times from Flash and PEROM are approximately one order of magnitude longer than from fast static RAM.

7.2 Speed Selection

For Flash and PEROM memory speed selection, the critical timing specification used is typically based on the Flash and PEROM data access time, t_{AA}. However, all timing specifications must be met and should always be reviewed for compliance.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations not listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GMBH
Technical Information Center
Schatzbogen 7
81829 München, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
+800 2666 8080

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Order No.: APR26
Rev. 1
10/2005