APR27
Rev. 1, 10/2005

Interfacing EPROM and EEPROM Memory with the DSP56300 Family of Digital Signal Processors

by Phil Brewer

This application note describes how to interface electrically erasable and programmable read only memory (EEPROM) and erasable programmable read only memory (EPROM) with the Freescale DSP56300 family of DSPs. This document supplements the *DSP56300 24-Bit Digital Signal Processor Family Manual* and the user's manuals and technical data sheets for devices in the DSP56300 family. It is a condensation of the memory interface details specified in the *DSP56300 24-Bit Digital Signal Processor Family Manual*, user and technical data manuals, and AMD EPROM and ATMEL EEPROM/EPROM data manuals.

This application note describe methods for interfacing various types of memory to the DSP56300 memory expansion port to assist the DSP hardware engineer in fully using the processor resources and generating an optimized memory design. The memory design uses a minimum of additional parts. Taking advantage of the available DSP control lines makes virtually glueless external memory interfaces possible, thereby reducing the cost and using fewer parts in an application memory design.

Specifically, this application note describes implementations for EPROM and EEPROM memory using the DSP56303. The DSP56303 is representative of the DSP56300 family and has all the essential family features. Where appropriate, several memory configurations provide a complete set of examples from which the designer can choose.

CONTENTS

1 DSP56300 Family

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, 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 internal peripherals and hardware accelerators.

2 Non-volatile Memory Families

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

- Mask ROM
- PROM
- OTPROM
- EPROM
  - 5.0V
  - 3.3V
- EEPROM
  - 5.0V
  - 3.3V
  - 2.7V

- Serial EEPROM
  - 5.0V
  - 3.3V
  - 2.7V
  - 2.5V
  - 1.8V

However, only the most popular memory types and those types requiring little or no supporting hardware (that is, glue logic) are included in this application note. The examples in this document give designers the insight necessary to implement other memory families and memory types, if needed, with DSP56300 family devices.

3 Interface Overview

This section describes how external memory devices are interfaced to a DSP56300 family device using the memory and peripheral expansion port. Only the DSP control signals used in the memory implementation examples are described. Other memory configuration implementations may require other available DSP56300 support features. 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 ($\overline{\text{RD}}$). Asserted during an external memory or peripheral read access.
- Write Data Enable ($\overline{\text{WR}}$). An output signal that is asserted during an external memory or peripheral write access.
- Address Bus (A[0–17]). Address lines to allow the DSP56303 direct address 256K words of external memory or peripherals. These active high 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 (D[0–23]). Twenty-four data lines on the DSP56303 that are active high 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]/$\overline{\text{RAS}}$[0–3]). Four address attribute or row address strobe signals. When the address attribute, AA, option is selected for these signal lines, they function as chip selects or additional address lines. As $\overline{\text{RAS}}$, they function as row address strobe lines for DRAM interfacing.

3.1 DSP56300 External Memory Timing

The DSP56300 family derives its core clock from one of various sources, as described in the PLL and clock generator chapter in the *DSP56300 24-Bit Digital Signal Processor Family Manual*. All memory interface timings derive 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. Asynchronous Memory Speeds Based on 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 = PLL Division Factor
   MF = PLL Multiplication Factor
   PDF = PLL Pre-Division Factor
   WS = wait states
2. TC = Clock Cycle Time
3. t_{AA} = Data access time (i.e., address and AA valid to input data valid)
4. t_{AW} = Data access time (i.e., address and AA valid to $\overline{\text{WR}}$ deassertion)

3.1.1 DSP56303 External Memory Bus Asynchronous Read Timing

The DSP56303 memory read access from external asynchronous memory 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 $\overline{\text{RD}}$ assertion time), the read enable signal, $\overline{\text{RD}}$, is asserted.
3. Before a delay of t_{OE} ($\overline{\text{RD}}$ 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 $\overline{\text{RD}}$. The DSP does not require any data hold time, t_{OHZ}, after deassertion of the $\overline{\text{RD}}$ 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.

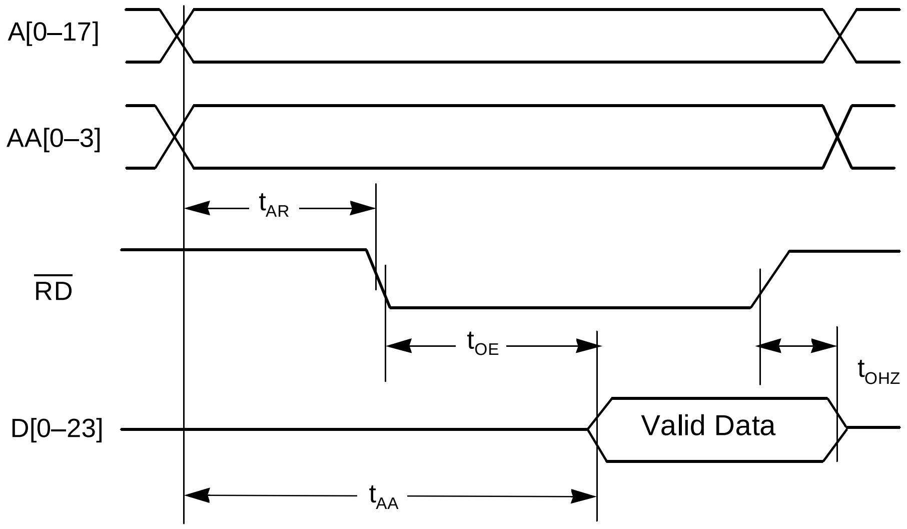

Figure 1. External Memory Bus Asynchronous Read Timing

3.1.2 DSP56300 External Memory Bus Asynchronous Write Timing

When writing to external asynchronous memory, the DSP56303 memory write access is controlled by the following sequence of steps:

1. The memory select address is asserted. This address is created by combining the address bus, A[0–17], and the address attributes AA[0–3].
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. 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} (that is, 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.

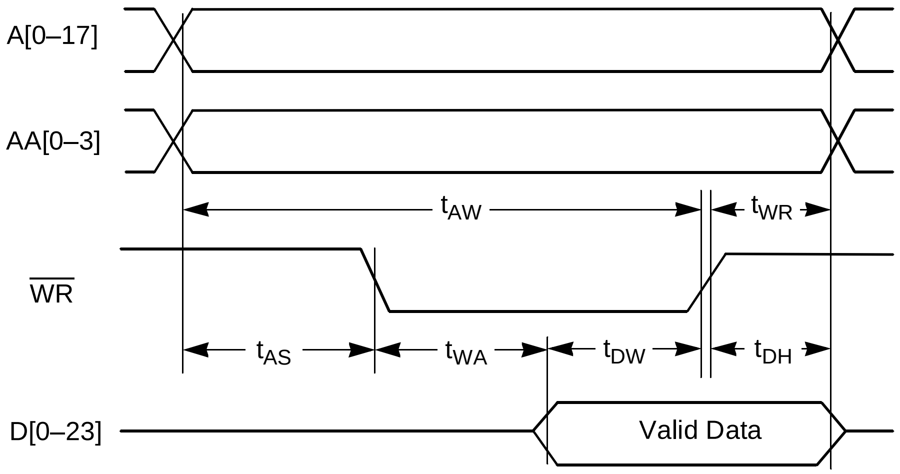

Figure 2. External Asynchronous Memory Bus Write Timing

3.2 DSP Memory Control Registers

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

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

3.2.1 DSP PLL and Clock Generation

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. 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 pre-divided before passing it to the PLL loop frequency multiplier. This frequency predivider has a programmable Division Factor range of 1 to 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 (VCO). 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 (LPD).* Divides the output frequency of the V_{CO} for use by the DSP core. The programmable division factor range if 1–128 is set by controlling the values placed into the PCTL bits 12–14. The low-power division factor is 2^n, where n is the value of 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

The operating core frequency of the DSP 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.

3.2.2 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

3.2.3 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]/$\overline{\text{RAS[0–3]}}$ 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 of the address to the eight most significant bits 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.

3.2.4 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.

3.2.5 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 are applicable 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.

4 EEPROM Memory

EEPROM provides non-volatile program and data storage, which is in-circuit reprogrammable, and offers relatively fast access times. However, even the fastest EEPROM memories would 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 12.5nS.

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 EEPROM devices have this address stability requirement, the DSP operates with at least one wait state when using these external memories.

Two EEPROM memory design examples illustrate how easy and flexible it is to use an EEPROM device with the DSP56300 family. The first example, 2K × 8-bit BOOT EEPROM, shows one solution for an embedded system that loads a program from EEPROM at boot time into the DSP internal RAM and then executes it. The second example, 32K × 8-bit boot EEPROM, shows another solution for a bootable embedded system. The DSP loads internal program RAM from the EEPROM at boot time and then overlays new programs and data from the EEPROM as needed.

4.1 2K × 8-Bit Boot EEPROM Example

The 2K × 8-bit ‘P’ memory space, bootstrap EEPROM implementation uses the ATMEL AT28C16-25 (see **Figure 3** for the memory map layout, **Figure 4** for the block diagram, **Figure 7** for the schematic, and **Example 1** for the code example). A program placed into the EEPROM can be loaded into the DSP at boot time and run. The running program can then save an updated version of the program back into the EEPROM for later use.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 250nS EEPROM, 20 wait states are required. However, during the DSP boot sequence, the DSP generates 31 wait states with the core running at 4.0 MHz, sufficient to accommodate a 7.7mS access time device. This 5.0V device is organized as 2K × 8-bits with a 250nS access time. One memory device is used to achieve the 8-bit wide boot bus.

Level conversion to and from 3.3V and 5V is necessary on the 8-bit data bus to accommodate the 5V memory devices. This is accomplished by using one of Quality Semiconductor's QS3245 QuickSwitch® 8-bit bus switches. This switch allows the connection of 3.3V CMOS logic (the DSP data bus) on one side and 5V TTL-compatible logic (the memory devices) on the other side, effectively providing a 3.3V-to-5V level conversion without adding any significant, 0.25nS, propagation delay.

During reset with Mode 1 selected, the DSP boot code configures Address Attribute Line 1 for program accesses in the address range from $D00000–$DFFFFF. The boot code then loads bytes from the EEPROM, packs them into 24-bit words, and stores them into program RAM. The first word, three packed bytes, read from the EEPROM indicates the number of words to load. The second word from the EEPROM 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. The program listed in **Example 1** initializes the Address Attribute Register, calculates an 8-bit checksum value, and programs the 8-bit checksum value in the last location of the 2K EEPROM bank.

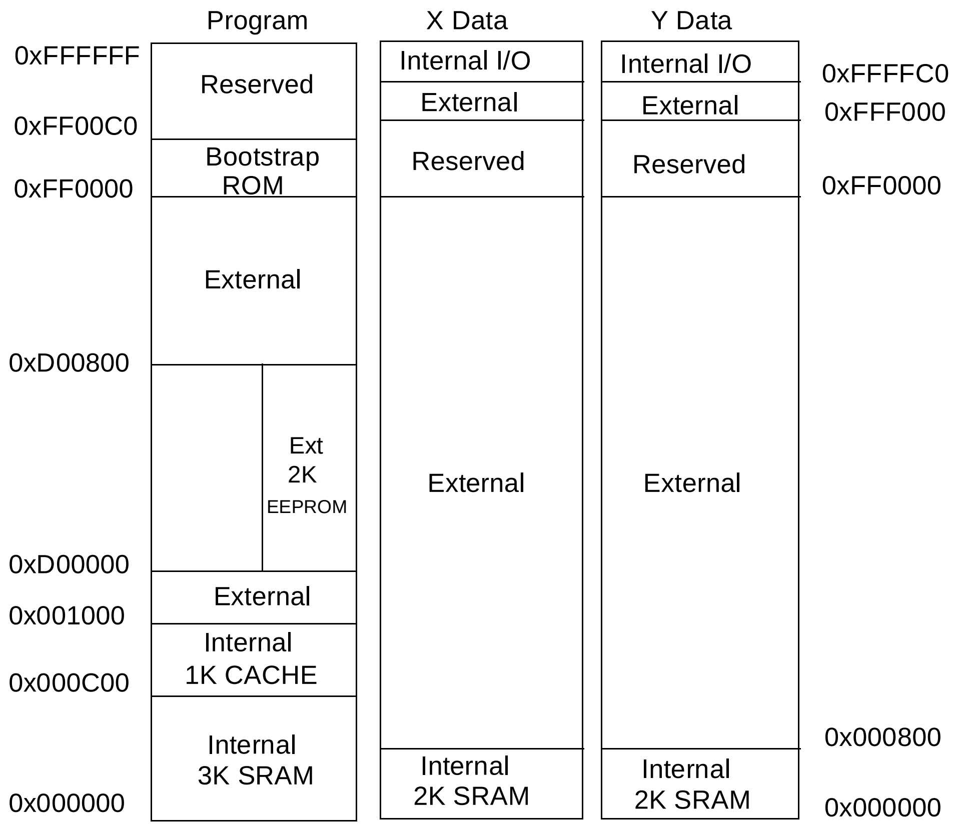

Figure 3. 2K × 8-bit BOOT EEPROM Memory Map

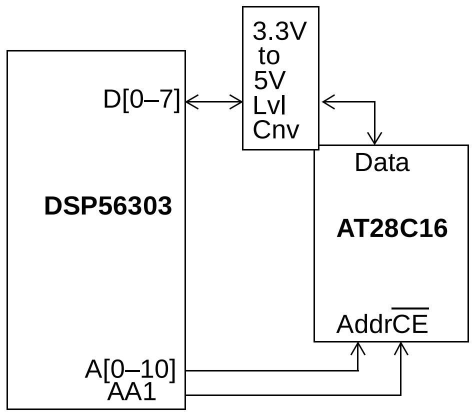

Figure 4. 2K × 8-bit BOOT/Overlay EEPROM Memory Example

4.1.1 EEPROM Timing Requirements

For the EEPROM device to work properly, its timing requirements must be met. Following are the timing requirements for the AT28C16-25 2K × 8-bit 250 nS EEPROM. **Table 2** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 5**.

Table 2. AT28C16-25 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Address to Output Delay	t_{ACC}	—	250nS
Chip Enable to Output Delay	t_{CE}	—	250nS
Output Enable to Output Delay	t_{OE}	10nS	100nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$. Whichever occurs first.	t_{OH}	0nS	—

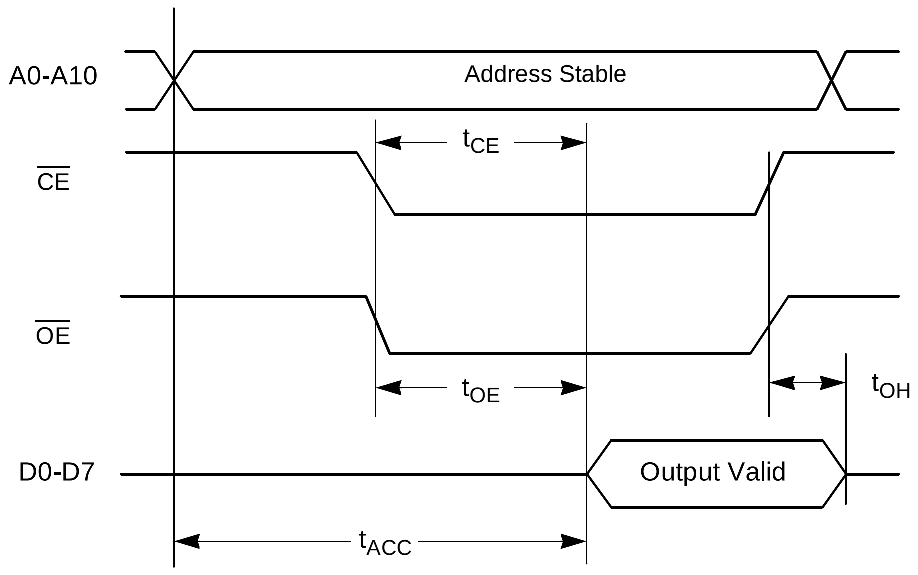

Figure 5. AT28C16 Memory Read Cycle Timing Diagram

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

Table 3. AT28C16-25 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write cycle time (to Program)	t_{WC}	—	1mS
Address setup time	t_{AS}	10nS	—
$\overline{CE}$ setup time	t_{CS}	0nS	—-
Write pulse width	t_{WP}	100nS	1uS
Data setup time	t_{DS}	50nS	—
Data hold time	t_{DH}	10nS	—

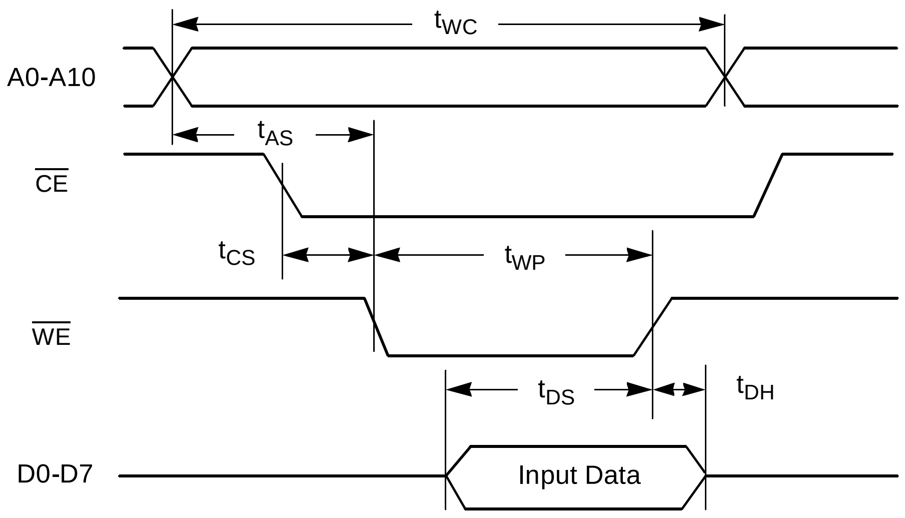

Figure 6. AT8C16 Memory Write Cycle Timing Diagram.

Non-volatile memory locations in the EEPROM can be individually programmed by writing data to that location and then waiting 1mS for the operation to complete. To determine when the data is successfully written, the memory location is read and compared with the last written value. When they compare, the write is complete.

An EEPROM memory location is written as follows:

1. Write the data byte to the EEPROM.
2. Read the data from the address to which the data was written and wait until the data written = data read.

4.1.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 2K × 8-bit boot EEPROM 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 register (PCTL). The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. The PCTL register 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 is used to enable, via EEPROM $\overline{\text{CE}}$, external 2K EEPROM memory bank accesses in the address range $D00000–$D007FF during program space requests. Configure the memory address space requirements for the AA1 using the Address Attribute Register 1 (AAR1). The AAR1 register 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 = 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 8 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 = $D.
- Specify the most significant portion of the address to compare, Bits 12-23 = $D00.

The value loaded into the AAR1 register is $D00D09. The value loaded into AAR0, AAR2 and AAR3 is $000000.

Select the proper number of wait states for the memory configuration using the bus control register (BCR). The BCR register 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 register is $000280

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR register 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 $\overline{\text{BB}}$ 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 $000000.

The value loaded into the OMR register is $004000.

Configure the memory mode of the DSP using the Status Register (SR). The SR register 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 SR bits are selected for their defaults of $000000.

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

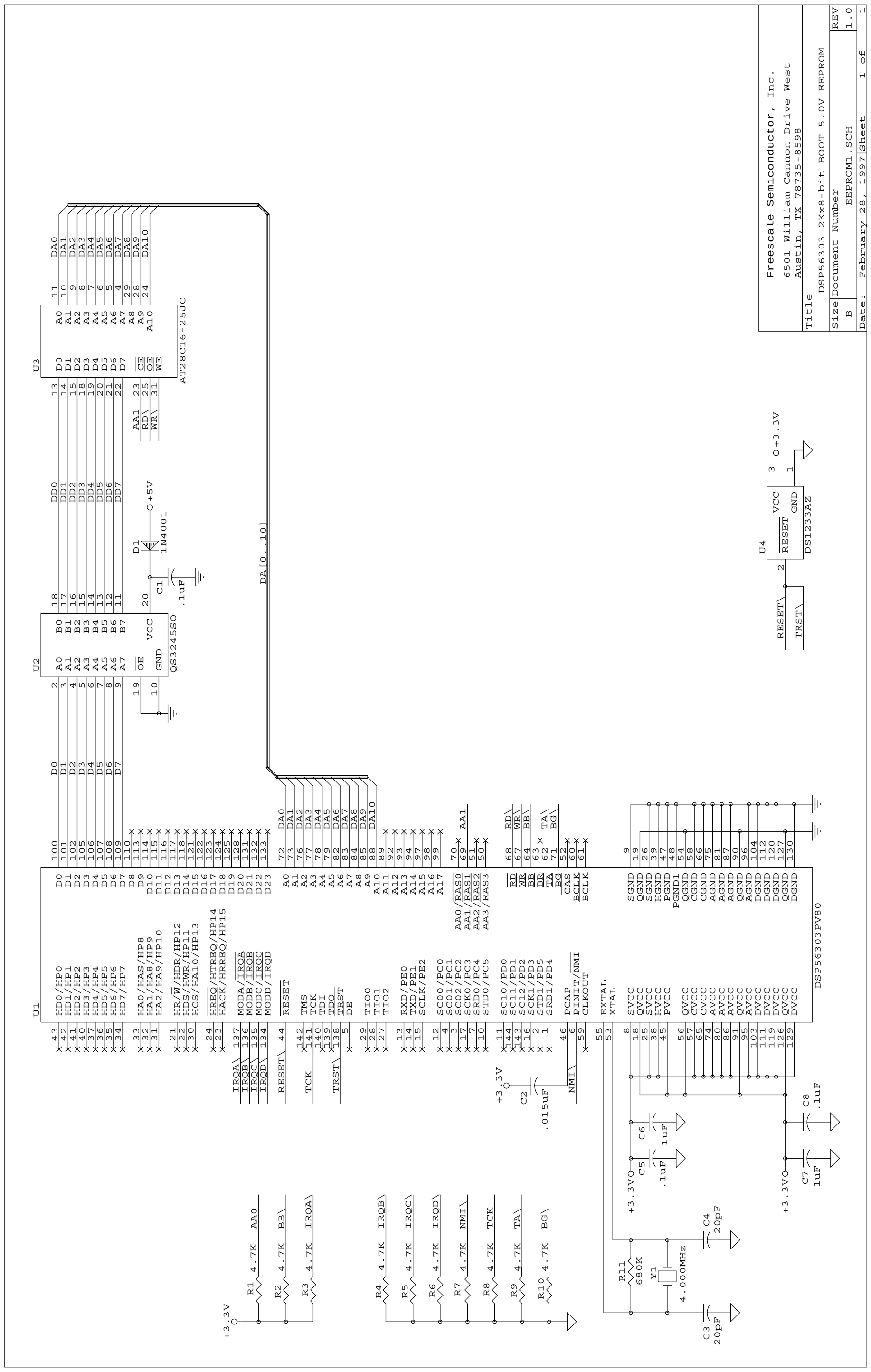

Figure 7. 2K × 8-bit BOOT EEPROM Schematic

Example 1. 2K × 8-bit BOOT EEPROM Checksum Verify Program

```
DSP56300 Assembler Version 6.0.1.6    97-02-27 21:53:54 eeprom1.asm
```

```
                                page    132,60,3,3,
                ;
                ;    eeprom1.asm - Simple program to calculate the 8-bit Checksum for
                ;                 a 2K x 8-bit block of EEPROM memory using a DSP56303.
                ;                 Contains:  Initialization routine,
                ;                            Routine to calculate 8-bit checksum,
                ;                            Routine to write checksum.
                ;
                ;       The program runs in Internal P:RAM to calculate the checksum on
                ;       External P:EEPROM from $D00000 - $D003FF @ 20w/s
                ;

        D00000  MemStart      equ    $D00000
        D00400  MemEnd        equ    $D00400
        000400  MemSize       equ    MemEnd-MemStart ; Last Word is stored Checksum
value

                ;--- Program Specific Storage Locations (X DATA SPACE)
                NEW_CHECKSUM
        000000                equ    $000000        ; Computed Checksum Value
                OLD_CHECKSUM
        000001                equ    $000001        ; Old Checksum from EEPROM

                ;--- DSP56303 Control Registers (X I/O SPACE)
        FFFFFB  BCR           equ    $FFFFFB        ; Bus Control Register
        FFFFFD  PCTL          equ    $FFFFFD        ; PLL Control Register
        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

                ;--- AAR1 value = 0xD00D09
        000001  acctype1      equ    1              ; External Memory access type = 0x1
        000000  aahigh1       equ    0              ; Enable AA1 pin to be low when
selected
        000008  aap1          equ    $8             ; Yes, Enable AA1 pin on ext 'P'
accesses
        000000  aax1          equ    0              ; No, Enable AA1 pin on ext 'X'
accesses
        000000  aay1          equ    0              ; No, Enable AA1 pin on ext 'Y'
accesses
        000000  aswap1        equ    0              ; No, Enable address bus swap
        000000  enpack1       equ    0              ; No, Enable packing/unpacking logic
        000D00  nadd1         equ    $000D00        ; Compare 13 address bits
        D00000  msadd1        equ    $D00000        ; Most significant portion of
address,
                                                    ; $D00000 - D007ff, to compare.
                                                    ; (1101,0000,0000,0xxx,xxxx,xxxx)
D00D09         AAR1_value     equu
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1


               ;--- BCR value = 0x000280
000000         aaa0ws         equ    0              ; Address Attribute Area 0 w/s = 0
000280         aaa1ws         equ    $280           ; Address Attribute Area 1 w/s = 20
000000         aaa2ws         equ    0              ; Address Attribute Area 2 w/s = 0
000000         aaa3ws         equ    0              ; Address Attribute Area 3 w/s = 0
000000         defws          equ    0              ; Default Address Area w/s = 0
000000         busss          equ    0              ; Bus state status = 0
000000         enblh          equ    0              ; Enable Bus Lock Hold = 0
000000         enbrh          equ    0              ; Enable Bus Request Hold = 0
000280         BCR_value      equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh

               ;-----------------------------------------------------------------------
               ;       Header for DSP56303 BOOT Code
               ;-----------------------------------------------------------------------
P:0000FE                      org    p:$100-2

P:0000FE                      dc     pgm_end-eeprom1 ; Number of words in program
P:0000FF                      dc     eeprom1        ; Starting address for program


               ;-----------------------------------------------------------------------
P:000100                      org    p:$100         ; Keep the program in internal RAM

               eeprom1
P:0001000D1080                bsr    init           ; Initialize DSP
        00000D

P:0001020D1080                bsr    calc_checksum  ; Calculate Checksum of EEPROM
        000014

P:000104548100                move   x:OLD_CHECKSUM,a1 ; Get EEPROM's Old Checksum value
P:000105558000                move   x:NEW_CHECKSUM,b1 ; Get Calculated Checksum value
P:00010620000D                cmp    a,b            ; Old Checksum = New Checksum?
P:0001070D104A                beq    _done          ; Yes, we are done
        000005
                                                    ; No
P:000109448000                move   x:NEW_CHECKSUM,x0 ; Get Calculated Checksum value
P:00010A0D1080                bsr    write_checksum ; Write EEPROM Checksum Data value
        000023

               _done
P:00010C050C00                bra    *              ; DONE, Do a dynamic HALT


        ;-----------------------------------------------------------------------------
        ; Initialization Section
        ;-----------------------------------------------------------------------------
               init
        98      P:00010D08F4BD               movep  #PCTL_value,x:PCTL ; Set PLL Control Register
                        0E0013
        99      P:00010F05F439               movec  #$080000,SR        ; Enable 1K Cache
                        080000
        100     P:00011108F4BB               movep  #BCR_value,x:BCR   ; Set external wait states
                        000280
        101     P:00011308F4B8               movep  #AAR1_value,x:AAR1 ; Set Address Attribute Reg1
                        D00D09
        102
        103     P:00011500000C               rts
        104
        105             ;--------------------------------------------------------------------
        106             ; Routine to Calculate 8-bit Checksum
        107             ;--------------------------------------------------------------------
        108                     calc_checksum
        109     P:00011605F420        move   #-1,m0         ; Set LINEAR addressing mode
                        FFFFFF
        110     P:00011860F400        move   #MemStart,r0   ; Set Starting Address of EEPROM
                        D00000
        111     P:00011A70F400        move   #MemSize-1,n0  ; Set to Size of Flash - Checksum
                        0003FF
        112
        113     P:00011C200013        clr    a
        114     P:00011D20001B        clr    b
        115     P:00011E540000        move   a1,x:NEW_CHECKSUM ; Initialize computed checksum ->
$000000
        116     P:00011F540100        move   a1,x:OLD_CHECKSUM ; Initialize read checksum -> $000000
        117
        118                     ; Compute the 8-bit Checksum
        119     P:00012044F400        move   #>$FF,x0 ; Set lower Byte Mask
                        0000FF
        120
        121     P:00012206D810        dor    n0,_ploop
                        000004
        122     P:00012407D88C        move   p:(r0)+,a1 ; Get the EEPROM location Value
        123     P:000125200046        and    x0,a    ; Mask for lower byte
        124     P:000126200018        add    a,b     ; Compute checksum
        125                     _ploop
        126
        127     P:00012707E08C        move   p:(r0),a1 ; Get EEPROM's Old Checksum value
        128     P:00012820004E        and    x0,b    ; Limit calculated Checksum to lower byte
        129     P:000129200046        and    x0,a    ; Limit Old Checksum to lower byte
        130     P:00012A550000        move   b1,x:NEW_CHECKSUM ; Save the Computed Checksum value
        131     P:00012B540100        move   a1,x:OLD_CHECKSUM ; Save the Old Checksum value
        132
        133     P:00012C00000C        rts
        134
        135             ;---------------------------------------------------------------------
        136             ; Routine to Place EEPROM into ERASE/WRITE MODE and send it the checksum Data
        137             ;---------------------------------------------------------------------
        138                     write_checksum
        139     P:00012D60F400        move   #MemEnd-1,r0
                        D003FF
        ld)
        140     P:00012F076084        move   x0,p:(r0)      ; Send the checksum value to the EEPROM
```

```
                                                         ; -- Now in EEPROM's Data Protect State
                                                         ;-- Wait till ERASE/WRITE Cycle is complete
       P:00013044F400         move    #>$FF,x0           ; Lower Byte Mask
               0000FF

                          _write_wait
       P:00013207E08D         move    p:(r0),b1          ; Get current value at EEPROM location
       P:00013320004E         and     x0,b               ; Mask for lower byte
       P:00013420000D         cmp     a,b                ; Value written = Value in EEPROM?
       P:0001350527DD         bne     _write_wait        ; No, wait until it is
                                                         ; Yes
       P:00013600000C         rts

               ;----------------------------------------------------------------------------
               pgm_end

                              end      eeprom1

0      Errors
0      Warning
```

4.2 32K × 8-bit BOOT/Overlay EEPROM Example

The 32K × 8-bit P memory space, bootstrap with program overlay EEPROM implementation uses the ATMEL AT28LV256-25 device (see **Figure 8** for the memory map layout, **Figure 9** for the block diagram, **Figure 12** for the schematic, and **Example 2** for the code example). A program placed into the EEPROM can be loaded into the DSP at boot time and run. The program can then load additional programs or data from the EEPROM into the DSP via overlay techniques. The boot or overlay program saves an updated version of the program or data back into the EEPROM for later use.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 250 nS EEPROM, 20 wait states are required. However, during the DSP boot sequence, the DSP generates 31 wait states with the core running at 4.0 MHz, sufficient to accommodate a 7.7 mS access time device. This 3.3 V device is organized as 32K × 8-bits with a 250 nS access time. One memory device composes the 8-bit wide boot bus.

During reset with Mode 1 selected, the DSP boot code configures AA1 for program accesses in the address range $D00000–$DFFFFF. The boot code then loads bytes from the EEPROM, packs them into 24-bit words, and stores them into program RAM. The first word, three packed bytes read from the EEPROM, indicates the number of words to load. The second word from the EEPROM contains the starting load address for the packed data. This starting load address is also the address which gains program control after the program load is completed. The program listed in **Example 2** initializes the Address Attribute Register, calculates an 8-bit checksum value, and programs the 8-bit checksum value in the last location of the 32K EEPROM bank.

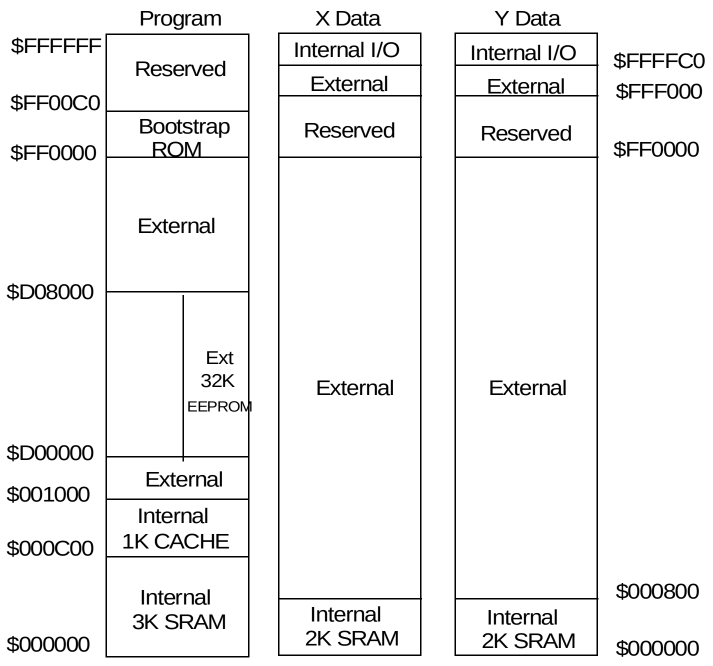

Figure 8. 32K x 8-bit BOOT EEPROM Memory Map

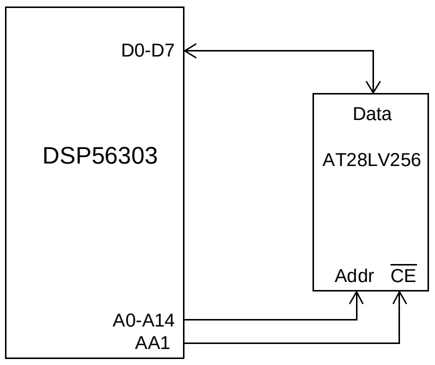

Figure 9. 32K x 8-bit BOOT/Overlay EEPROM Memory Example.

4.2.1 EEPROM Timing Requirements

For the EEPROM device to work properly, its timing requirements must be met. Following are the timing requirements for the AT28LV256-25 32K × 8-bit 250 nS EEPROM. **Table 4** shows the memory read timing specification values used in the memory read cycle timing diagram shown in **Figure 10**.

Table 4. AT28LV256-25 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Address to output delay	t_{ACC}	—	250nS
Chip enable to output delay	t_{CE}	—	250nS
Output enable to output delay	t_{OE}	—	100nS
Output hold time from address, $\overline{CE}$ or $\overline{OE}$, whichever occurs first.	t_{OH}	0nS	—

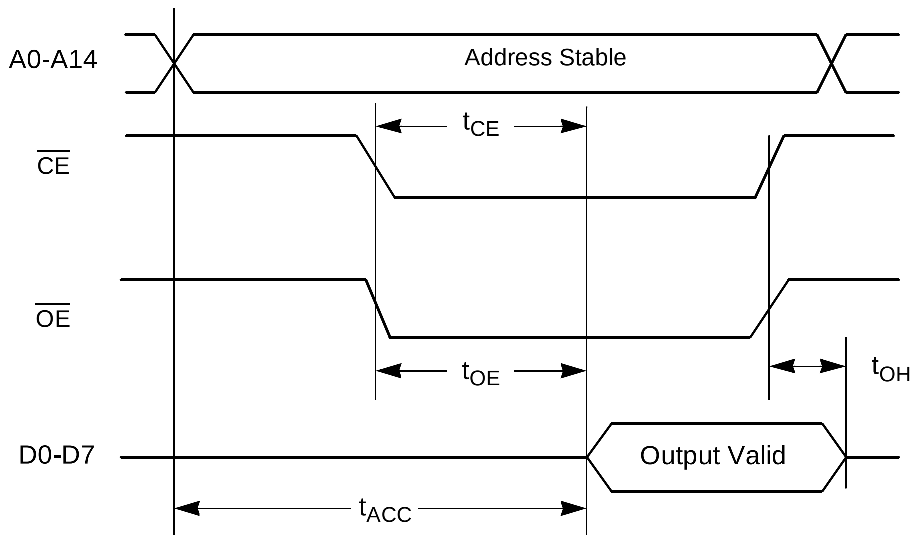

Figure 10. AT28LV256 Memory Read Cycle Timing Diagram

Table 5 shows the memory write timing specification values used in the memory write cycle timing diagram shown in **Figure 11**.

Table 5. AT28LV256-25 Memory Write Timing Specifications

Write Cycle Parameter	Symbol	Min	Max
Write cycle time (to program)	t_{WC}	—	10mS
Address setup time	t_{AS}	0 nS	—
$\overline{CE}$ setup time	t_{CS}	0 nS	—
Write pulse width	t_{WP}	200 nS	—
Data setup time	t_{DS}	50 nS	—
Data hold time	t_{DH}	0 nS	—

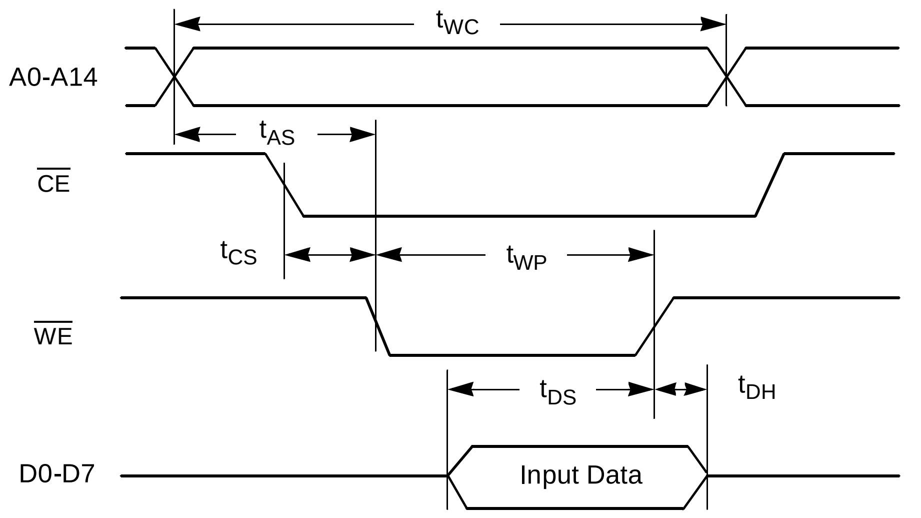

Figure 11. AT28LV256 Memory Write Cycle Timing Diagram.

Non-volatile memory locations in the EEPROM can be individually programmed, or a sector of 64 locations can be written during one program cycle. The AT28LV256 device is organized as 128 sectors of 64 bytes each. Writing to an EEPROM memory location requires writing one byte, then waiting for the erase/program cycle to complete; or the data for a complete sector, 64 bytes, is written to the EEPROM before the erase/program cycle is started. The following data sequence is written to the EEPROM:

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

4.2.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 32K × 8-bit boot EEPROM 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 Register (PCTL). 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 DSP56300 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 EEPROM $\overline{\text{CE}}$, external 32K EEPROM memory bank accesses in the address range $D00000–$D07FFF during program space requests. Configure the memory address space requirements for AA1 using the Address Attribute Register 1 (AAR1). The AAR1 register 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 = 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 = $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 $000000.

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

- Address Attribute Area Zero wait states, bits 0–4 = $0
- Address Attribute Area One wait states, bits 5–9 = $14
- Address Attribute Area Two wait states, bits 10–12 = $0
- Address Attribute Area Three 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 register is $000280.

Configure the operating mode and external memory controls using the Operating Mode Register (OMR). The OMR register 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 $\overline{\text{BB}}$ 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 $000000.

The value loaded into the OMR register 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 SR bits are selected for their defaults of $000000.

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

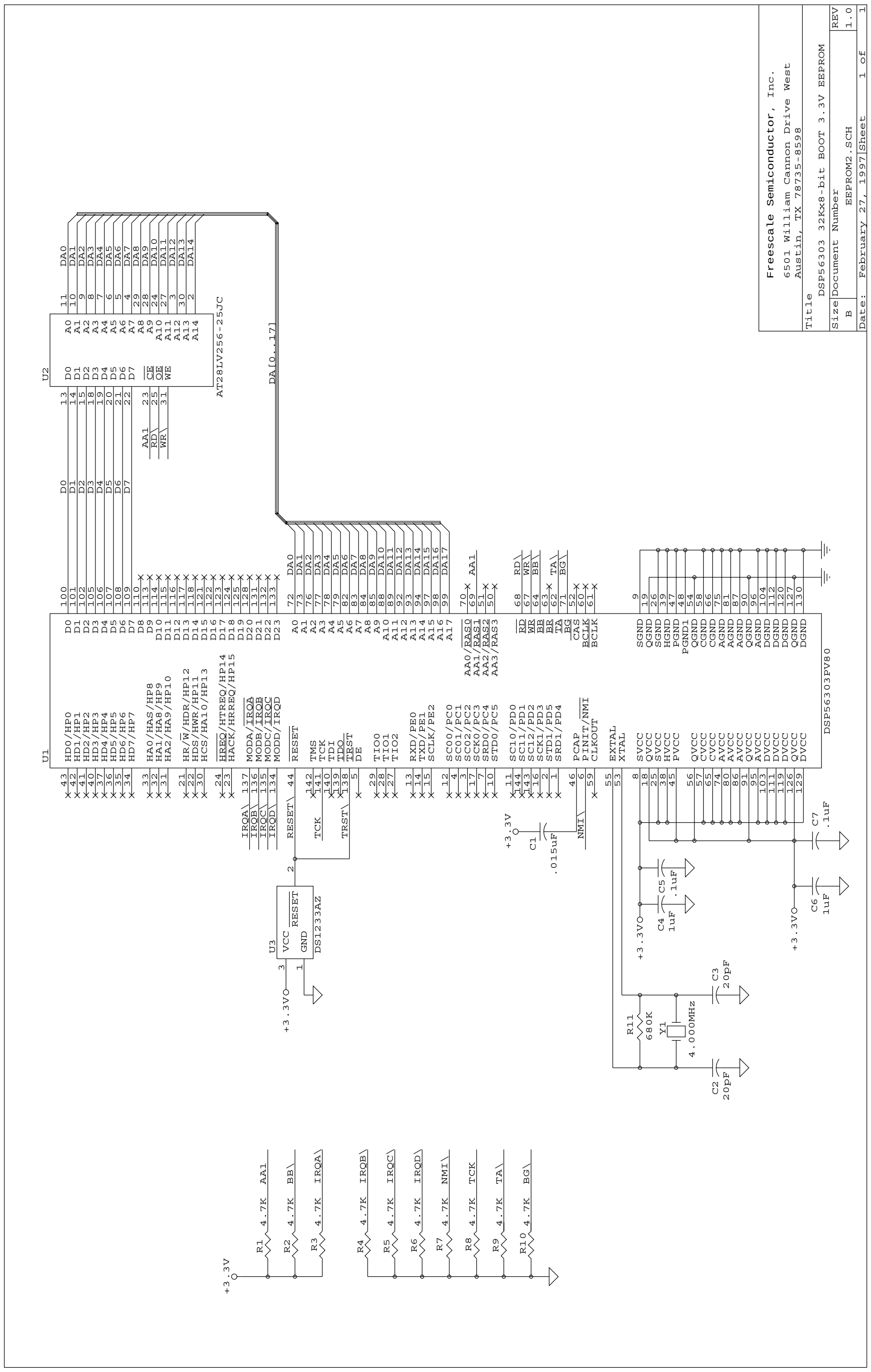

Figure 12. 32Kx8-bit BOOT EEPROM Schematic

Example 2. 32K x 8-bit BOOT EEPROM Checksum Verify Program

```
DSP56300 Assembler Version 6.0.1.6   97-02-26 21:21:55 eeprom2.asm
```

```
                                    page    132,60,3,3,
                    ;
                    ;    eeprom2.asm - Simple program to calculate the 8-bit Checksum for
                    ;                  a 32K x 8-bit block of EEPROM memory using a DSP56303.
                    ;                  Contains:  Initialization routine,
                    ;                             Routine to calculate 8-bit checksum,
                    ;                             Routine to write checksum.
                    ;
                    ;       The program runs in Internal P:RAM to calculate the checksum on
                    ;       External P:EEPROM from $D00000 - $D07FFF @ 20w/s
                    ;

    D00000          MemStart        equ     $D00000
    D08000          MemEnd          equ     $D08000
    008000          MemSize         equ     MemEnd-MemStart ; Last Word is stored Checksum
value

                    ;--- Program Specific Storage Locations (X DATA SPACE)
                    NEW_CHECKSUM
    000000                          equ     $000000        ; Computed Checksum Value
                    OLD_CHECKSUM
    000001                          equ     $000001        ; Old Checksum from PEROM

                    ;--- DSP56303 Control Registers (X I/O SPACE)
    FFFFFB          BCR             equ     $FFFFFB        ; Bus Control Register
    FFFFFD          PCTL            equ     $FFFFFD        ; PLL Control Register
    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
rediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk

                    ;--- AAR1 value = 0xD00909
    000001          acctype1        equ     1       ; External Memory access type = 0x1
    000000          aahigh1         equ     0       ; Enable AA1 pin to be low when selected
    000008          aap1            equ     $8      ; Yes, Enable AA1 pin on ext 'P' accesses
    000000          aax1            equ     0       ; No, Enable AA1 pin on ext 'X' accesses
    000000          aay1            equ     0       ; No, Enable AA1 pin on ext 'Y' accesses
    000000          aswap1          equ     0       ; No, Enable address bus swap
    000000          enpack1         equ     0       ; No, Enable packing/unpacking logic
    000900          nadd1           equ     $000900 ; Compare 9 address bits
    D00000          msadd1          equ     $D00000 ; Most significant portion of address,
                                                    ; $D00000 - D07fff, to compare.
                                                    ; (1101,0000,0xxx,xxxx,xxxx,xxxx)
    D00909          AAR1_value      equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1

```

```

               ;--- BCR value = 0x000280
000000         aaa0ws        equ    0              ; Address Attribute Area 0 w/s = 0
000280         aaa1ws        equ    $280           ; Address Attribute Area 1 w/s = 20
000000         aaa2ws        equ    0              ; Address Attribute Area 2 w/s = 0
000000         aaa3ws        equ    0              ; Address Attribute Area 3 w/s = 0
000000         defws         equ    0              ; Default Address Area w/s = 0
000000         busss         equ    0              ; Bus state status = 0
000000         enblh         equ    0              ; Enable Bus Lock Hold = 0
000000         enbrh         equ    0              ; Enable Bus Request Hold = 0
000280         BCR_value     equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh

               ;----------------------------------------------------------------------
               ;       Header for DSP56303 BOOT Code
               ;----------------------------------------------------------------------
P:0000FE                     org    p:$100-2

P:0000FE                     dc     pgm_end-eeprom2    ; Number of words in program
P:0000FF                     dc     eeprom2            ; Starting address for program


               ;----------------------------------------------------------------------
P:000100                     org    p:$100         ; Keep the program in internal RAM

               eeprom2
P:0001000D1080               bsr    init           ; Initialize DSP
       00000D

P:0001020D1080               bsr    calc_checksum  ; Calculate Checksum of EEPROM
       000014

P:000104548100               move   x:OLD_CHECKSUM,a1 ; Get EEPROM's Old Checksum value
P:000105558000               move   x:NEW_CHECKSUM,b1 ; Get Calculated Checksum value
P:00010620000D               cmp    a,b            ; Old Checksum = New Checksum?
P:0001070D104A               beq    _done          ; Yes, we are done
       000005
                                                   ; No
P:000109448000               move   x:NEW_CHECKSUM,x0 ; Get Calculated Checksum value
P:00010A0D1080               bsr    write_checksum ; Write EEPROM Checksum Data value
       000023

               _done
P:00010C050C00               bra    *              ; DONE, Do a dynamic HALT


       ;-----------------------------------------------------------------------------
       ; Initialization Section
       ;-----------------------------------------------------------------------------
               init
P:00010D08F4BD        movep  #PCTL_value,x:PCTL    ; Set PLL Control Register
       0E0013
P:00010F05F439        movec  #$080000,SR           ; Enable 1K Cache
       080000
P:00011108F4BB        movep  #BCR_value,x:BCR      ; Set external wait states
```

```
       000280
P:00011308F4B8        movep  #AAR1_value,x:AAR1    ; Set Address Attribute Reg1
       D00909

P:00011500000C        rts


       ;---------------------------------------------------------------------------
       ; Routine to Calculate 8-bit Checksum
       ;---------------------------------------------------------------------------
               calc_checksum
P:00011605F420        move   #-1,m0                ; Set LINEAR addressing mode
       FFFFFF
P:00011860F400        move   #MemStart,r0          ; Set Starting Address of EEPROM
       D00000
P:00011A70F400        move   #MemSize-1,n0         ; Set to Size of Flash - Checksum
       007FFF

P:00011C200013        clr    a
P:00011D20001B        clr    b
P:00011E540000        move   a1,x:NEW_CHECKSUM ; Initialize computed checksum ->
$000000
P:00011F540100        move   a1,x:OLD_CHECKSUM ; Initialize read checksum -> $000000

               ; Compute the 8-bit Checksum
P:00012044F400        move   #>$FF,x0              ; Set lower Byte Mask
       0000FF

P:00012206D810        dor    n0,_ploop
       000004
P:00012407D88C        move   p:(r0)+,a1            ; Get the EEPROM location Value
P:000125200046        and    x0,a                  ; Mask for lower byte
P:000126200018        add    a,b                   ; Compute checksum
               _ploop

P:00012707E08C        move   p:(r0),a1             ; Get EEPROM's Old Checksum value
P:00012820004E        and    x0,b           ; Limit calculated Checksum to lower byte
P:000129200046        and    x0,a                  ; Limit Old Checksum to lower byte
P:00012A550000        move   b1,x:NEW_CHECKSUM     ; Save the Computed Checksum value
P:00012B540100        move   a1,x:OLD_CHECKSUM     ; Save the Old Checksum value

P:00012C00000C        rts


       ;----------------------------------------------------------------------------
       ; Routine to Place EEPROM into ERASE/WRITE MODE and send it the checksum data
       ;----------------------------------------------------------------------------
               write_checksum
P:00012D60F400        move   #MemEnd-1,r0
       D07FFF
ld)
P:00012F076084        move   x0,p:(r0)      ; Send the checksum value to the EEPROM
                                            ; -- Now in EEPROM's Data Protect State
               ;-- Wait till ERASE/WRITE Cycle is complete
P:00013044F400        move   #>$FF,x0       ; Lower Byte Mask
       0000FF

```

```
                                _write_wait
P:00013207E08D        move    p:(r0),b1      ; Get current value at EEPROM location
P:00013320004E        and     x0,b           ; Mask for lower byte
P:00013420000D        cmp     a,b            ; Value written = Value in EEPROM?
P:0001350527DD        bne     _write_wait    ; No, wait until it is
                                             ; Yes
P:00013600000C        rts

               ;---------------------------------------------------------------------------
               pgm_end

                      end      eeprom2

0       Errors
0       Warning
```

5 Erasable Programmable Read-Only Memory

EPROM, PROM and ROM memory provide non-volatile program and data storage with relatively fast access times. However, even the fastest EPROM 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 must reside in a stable address during the entire external access, a DSP56300 family device incurs an automatic one wait state penalty. Since EPROM devices have this address stability requirement, the DSP operates with at least one wait state when using these external memories.

The following three EPROM design examples illustrate how easy and flexible it is to use EPROM devices with the DSP56300 family. The first example, 128K × 24-bit BOOT, 128K × 24-bit P, 128K × 24-bit ‘X’ and 128K × 24-bit ‘Y’, shows a solution for an embedded system using four unique banks of EPROM memory. This allows for one bank to be used for booting the DSP, another bank for external program code execution, and the remaining two banks for external coefficient data use. The second example, 16K × 8-bit BOOT, shows a solution for a bootable embedded system. The DSP loads internal program RAM from the EPROM at boot time and then executes it. The third example, 512K × 8-bit BOOT/Overlay, shows another solution for a bootable embedded system. This implementation loads internal DSP program RAM from the EPROM at boot time and then overlays new programs and data from the EPROM when needed.

5.1 128K × 24-Bit Boot, P, X’ and Y EPROM Example

This section describes a 128K × 24-bit BOOT, 128K × 24-bit ‘P’, 128K × 24-bit X and 128K × 24-bit Y memory space, EPROM implementation using the AMD AM27C040 device (see **Figure 13** for boot memory map layout, **Figure 14** for user memory map layout, **Figure 15** for the block diagram, and **Figure 13** for the address attribute associations for boot, program, X-data and Y-data spaces). Since the EPROM is present in the DSP program space, program control can be turned over to the program in the EPROM at boot time. This allows an embedded application to boot and run from EPROM, permitting the DSP internal 1K cache to help speed repeated program functions. Also, since X and Y data space is available in the EPROM, program and data variables can be referenced after boot-up.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 90 nS EPROM, eight wait states are required. This 5 V device is organized as 512K × 8-bits with a 90 nS access time. Three memory devices compose the 24-bit wide 'P', 'X' and 'Y' buses.

Level conversion to and from 3.3 V and 5 V is necessary on the 24-bit data bus to accommodate the 5 V memory devices. This is accomplished by using three Quality Semiconductor's 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 without adding any significant (0.25 nS) propagation delay.

During DSP reset with Mode 0 selected, the DSP starts fetching instructions from external memory location P:$C00000. Since the DSP56303 device uses only 18 address lines, A[0–17], to select external memory, address P:$C00000 appears as P:$000000 on the external address bus. Therefore, the EPROM is configured to respond to all external memory requests. Data aliasing occurs at every 256K boundary if no additional address decoding is provided. Since the 512K EPROM requires 19 address lines to select all of its 512K memory locations, EPROM address line A18 is controlled by AA1 (see the memory space selection chart in **Figure 13**, the schematic in **Figure 17** and the program listed in **Example 3**). The EEPROM address line A17 which is controlled by AA0, selects the EEPROM used during X-data or Y-data accesses. Using these two address attribute lines, the 512K × 24-bit EPROM memory bank can be segmented into four regions of 128K × 24-bits each (see **Figure 13**). The program listed in **Example 3** initializes the address attribute registers and calculates a 24-bit checksum value for each of the four 128K EPROM memory spaces to compare with the checksum values stored in the last location of each 128K EPROM bank.

Table 6. Memory Space Selections Selected by Address Attributes

	Address Attribute Line **EPROM Address Line**	**AA1 A18**	**AA0 A17**
Y-data space	Y:$100000–$1FFFFF	0	0
X-data space	X:$100000–$1FFFFF	0	1
Program space	P:$100000–$1FFFFF	1	0
Program boot space	P:$C00000–$C1FFFF Aliased from $0–$FFFFFF at every fourth 128K page boundary while AA0 and AA1 are unconfigured.	1	1

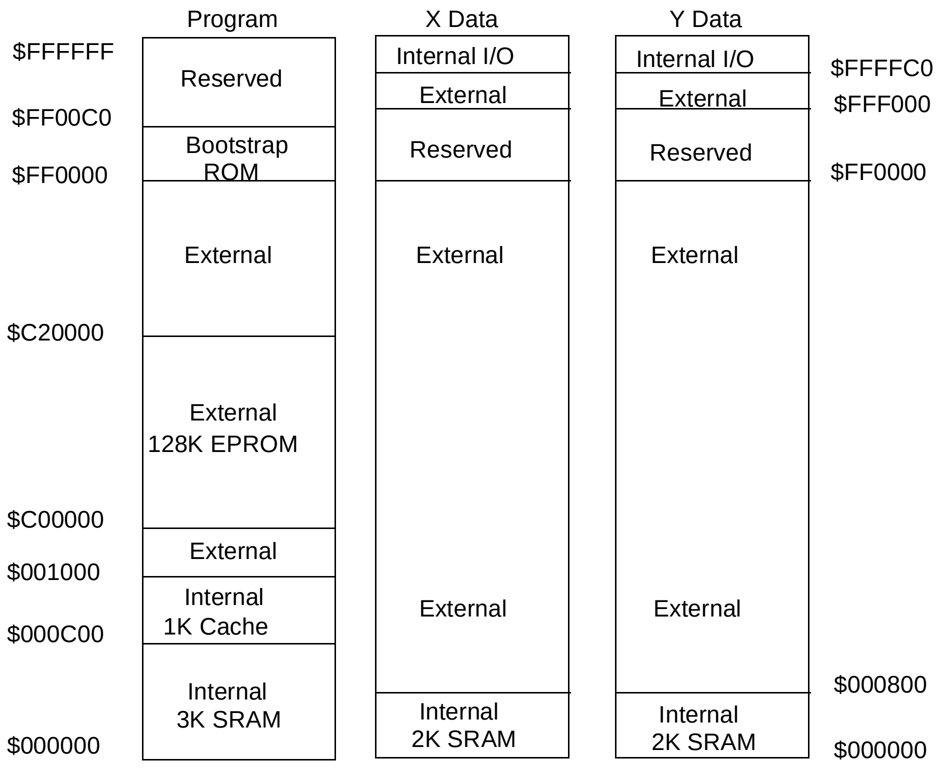

Figure 13. 128K × 24-bit BOOT EPROM Memory Map

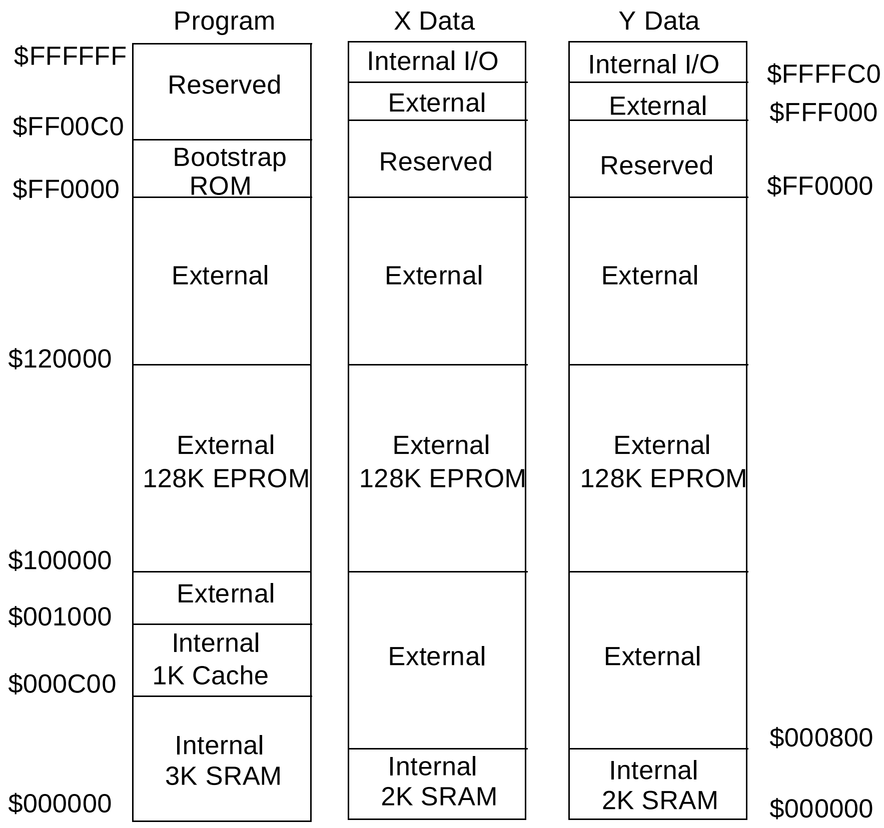

Figure 14. 128K × 24-bit P, X and Y EPROM Memory Map

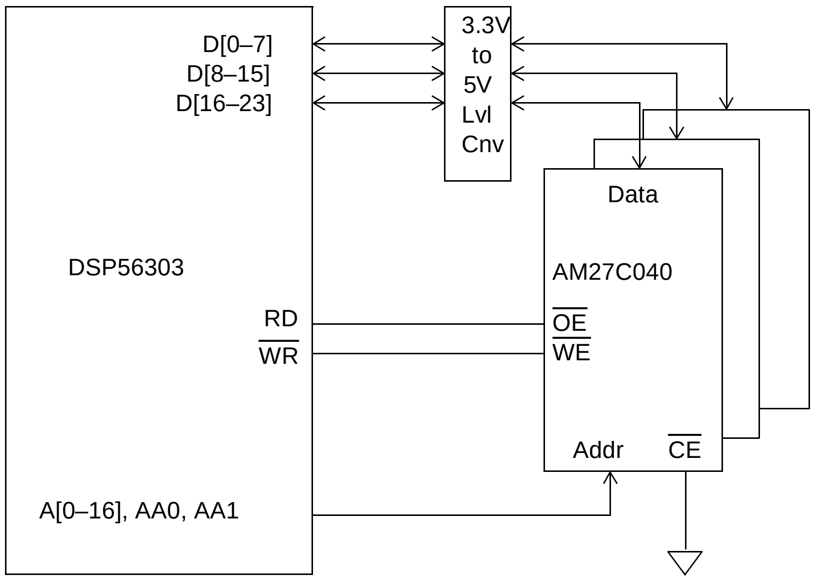

Figure 15. 512K × 24-bit EPROM Memory Example.

5.1.1 EPROM Timing Requirements

Following are the timing requirements for the AM27C040-90 512K × 8-bit 90 nS EPROM. **Table 7** lists the memory read timing specification values used in the memory read cycle timing diagram shown in **Figure 16**.

Table 7. AM27C040-90 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}	—	40 nS
Output hold time from address, $\overline{\text{CE}}$ or $\overline{\text{OE}}$, whichever occurs first.	t_{OH}	0 nS	—

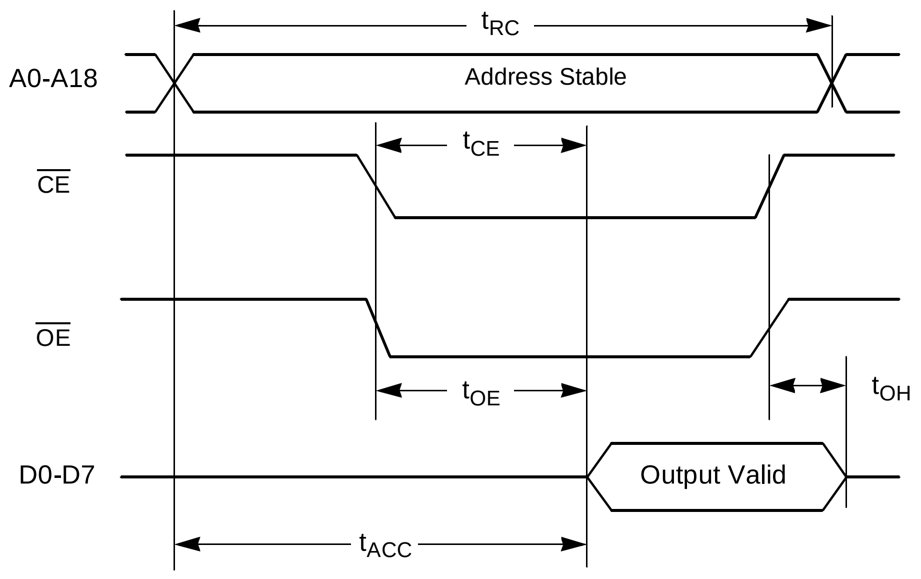

Figure 16. AM27C040 Memory Read Cycle Timing Diagram.

5.1.2 DSP56303 Port A Timing Requirements and Register Settings

For optimal use of the 512K × 24-bit ‘P’ 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 Register (PCTL). 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 = 0x0
- Low-power Divider value = 1, bits 12-14 = 0x0
- VCO Multiplication value = 20, bits 0-11 = 0x013
- 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 0x0E0013.

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

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = 0x1.
- 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 = 0x7.
- Specify the most significant portion of the address to compare, bits 12–23 = 0x100.

The value loaded into the AAR1 register is 0x10070D.

AA0 selects, via EPROM A17, the external 128K EPROM memory bank accesses in the address range from $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 Address Attribute Register 0 (AAR0). The AAR0 register value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = 0x1.
- 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 = 0x7.
- Specify the most significant portion of the address to compare, bits 12–23 = 0x100.

The value loaded into the AAR0 register is 0x100715. The value loaded into AAR2 and AAR3 is 0x000000.

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

- Address attribute area zero wait states, bits 0–4 = 0x8
- Address attribute area one wait states, bits 5–9 = 0x8
- Address attribute area two wait states, bits 10–12 = 0x0
- Address attribute area three wait states, bits 13–15 = 0x0
- Default address area wait states, bits 16–20 = 0x8
- 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 register is 0x080108.

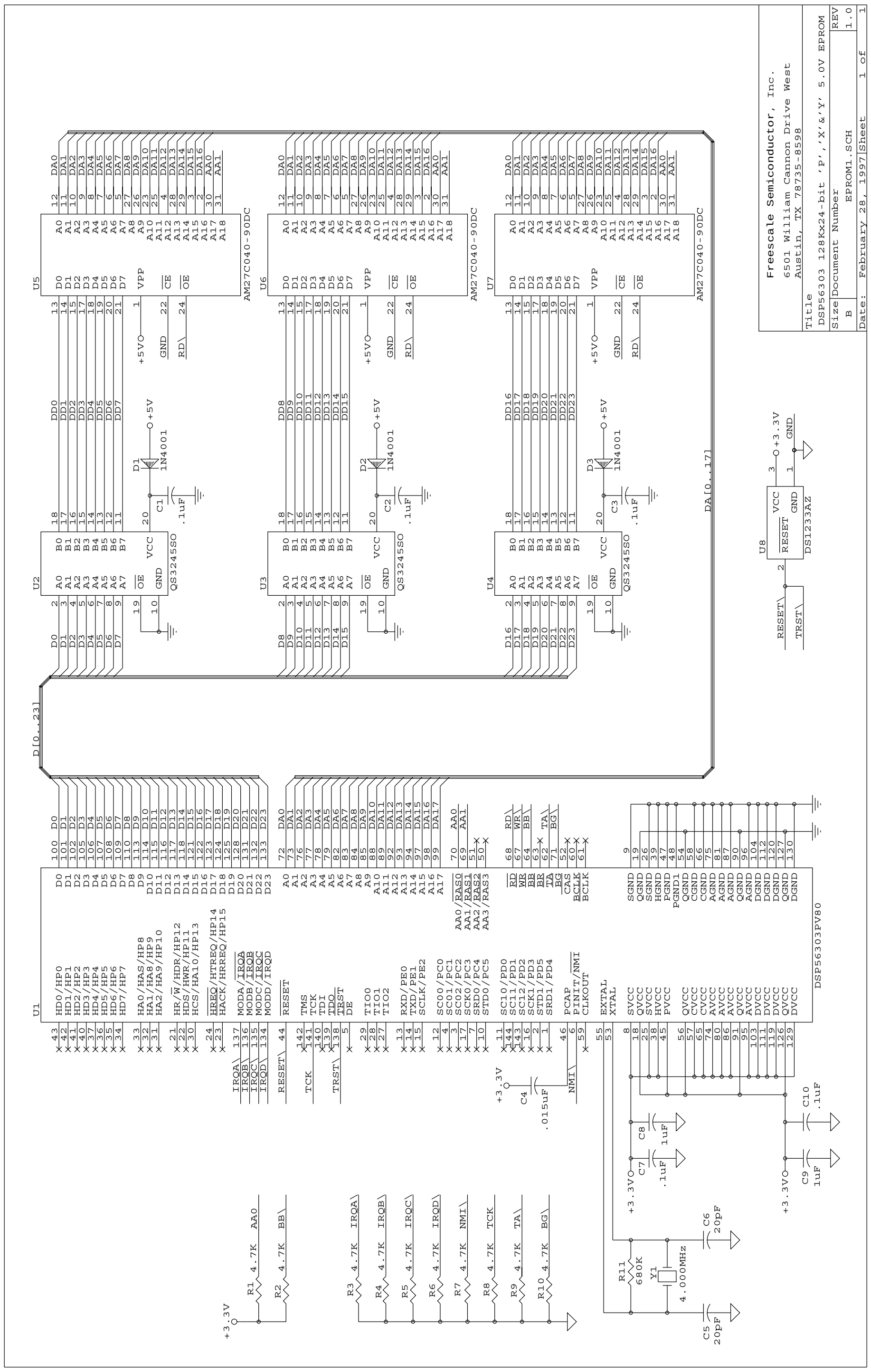

Figure 17. 128Kx24 BOOT, ‘P’, ‘X’ and ‘Y’ Space EPROM Schematic

Example 3. 128K × 24-bit ‘P’, 128K × 24-bit ‘X’ and 128K × 24-bit ‘Y’ Space EPROM Checksum Program

```
DSP56300 Assembler Version 6.0.1.6    97-02-27 21:54:56 eprom1.asm
```

```
                                          page     132,60,3,3,
                        ;
                        ;       eprom1.asm - Simple program to calculate the 24-bit Checksum
                        ;                   for a 128K x 24-bit block of BOOT, 128K x 24-bit block
                        ;                   of Program, 128K x 24-bit block of X-Data and
                        ;                   a 128K x 24-bit block of Y-Data memory using a
DSP56303.
                        ;
                        ;
                        ;        The program runs in Internal P:RAM to calculate the checksum;
                        ;        on External BOOT:EPROM from $C00000 - $C1FFFF @ 8w/s,
                        ;        on External P:EPROM from $100000 - $11FFFF @ 8w/s,
                        ;        on External X:EPROM from $100000 - $11FFFF @ 8w/s and
                        ;        on External Y:EPROM from $100000 - $11FFFF @ 8w/s.
                        ;

          C00000        BootStart     equ     $C00000
          C20000        BootEnd       equ     $C20000
          020000        BootSize      equ     BootEnd-BootStart   ; Last Word is stored Checksum
value

          100000        PMemStart     equ     $100000
          120000        PMemEnd       equ     $120000
          020000        PMemSize      equ     PMemEnd-PMemStart   ; Last Word is stored Checksum
value

          100000        XMemStart     equ     $100000
          120000        XMemEnd       equ     $120000
          020000        XMemSize      equ     XMemEnd-XMemStart   ; Last Word is stored Checksum
value

          100000        YMemStart     equ     $100000
          120000        YMemEnd       equ     $120000
          020000        YMemSize      equ     YMemEnd-YMemStart   ; Last Word is stored Checksum
value

                        ;--- Program Specific Storage Locations (X DATA SPACE)
                        CKSUM_CALC_P
          000000                      equ     $000000       ; P Space Computed Checksum Value
                        CKSUM_READ_P
          000001                      equ     $000001       ; P Space Checksum in Last Memory
Location
                        CKSUM_CALC_X
          000002                      equ     $000002       ; X Space Computed Checksum Value
                        CKSUM_READ_X
          000003                      equ     $000003       ; X Space Checksum in Last Memory
Location
                        CKSUM_CALC_Y
          000004                      equ     $000004       ; Y Space Computed Checksum Value
                        CKSUM_READ_Y
          000005                      equ     $000005       ; Y Space Checksum in Last Memory
Location
                        CKSUM_CALC_B
          000006                      equ     $000006       ; BOOT Space Computed Checksum Value
                        CKSUM_READ_B
```

```
000007                  equ     $000007        ; BOOT Space Checksum in Last Memory
Location

                ;--- 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)*4.00 MHz=80.00 MHz
000000          crystal         equ     0              ; No, Crystal not less than 200 kHz
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

                ;--- AAR1 value = 0x10070D
000001          acctype1        equ     1              ; External Memory access type = 0x1
000001          aahigh1         equ     1              ; Enable AA1 pin to be high when
selected
000008          aap1            equ     $8             ; Yes, Enable AA1 pin on ext 'P'
accesses
000000          aax1            equ     0              ; No, Enable AA1 pin on ext 'X'
accesses
000000          aay1            equ     0              ; No, Enable AA1 pin on ext 'Y'
accesses
000000          aswap1          equ     0              ; No, Enable address bus swap
000000          enpack1         equ     0              ; No, Enable packing/unpacking logic
000700          nadd1           equ     $000700        ; Compare 7 address bits
100000          msadd1          equ     $100000        ; Most significant portion of
address,
                                                       ; $100000 - 11ffff, to compare.
                                                       ; (0001,000x,xxxx,xxxx,xxxx,xxxx)
10070A          AAR1_value      equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1

                ;--- AAR0 value = 0x100715
000001          acctype         equ     1              ; External Memory access type = 0x1
000004          aahigh          equ     $4             ; Enable AA0 pin to be High when
selected
000000          aap             equ     0              ; No, Enable AA0 pin on ext 'P'
accesses
000010          aax             equ     $10            ; Yes, Enable AA0 pin on ext 'X'
accesses
000000          aay             equ     0              ; No, Enable AA0 pin on ext 'Y'
accesses
000000          aswap           equ     0              ; No, Enable address bus swap
000000          enpack          equ     0              ; No, Enable packing/unpacking logic
000700          nadd            equ     $000700        ; Compare 7 address bits
```

```
       90     100000         msadd          equ    $100000        ; Most significant portion 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 = 0x080108
       96     000008         aaa0ws         equ    $8             ; Address Attribute Area 0 w/s = 8
       97     000100         aaa1ws         equ    $100           ; Address Attribute Area 1 w/s = 8
       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    080000         defws          equ    $080000        ; Default Address Area w/s = 8
       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    080108         BCR_value      equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       105
       106                   ;----------------------------------------------------------------------
       107    P:000100                      org    p:$100         ; Keep the program in internal RAM
       108
       109                   eprom1
       110
       111                   ;---------- Initialization Section ----------
       112    P:00010008F4BD        movep          #PCTL_value,x:PCTL ; Set PLL Control Register
                     0E0013
       113    P:00010205F43A        movec          #$004000,OMR       ; Disable Address Attribute
Priority
                     004000
       114    P:00010405F439        movec   #$080000,SR           ; Enable 1K Cache
                     080000
       115    P:00010608F4BB        movep          #BCR_value,x:BCR   ; Set external wait states
                     080108
       116
       117           ;----------------------------------------------------------------------------
       118           ;----------------------------------------------------------------------------
       119                   do_b_checksum
       120    P:00010808F4B9        movep          #>$0,x:AAR0        ; Clear Address Attribute Reg0
                     000000
       121    P:00010A08F4B8        movep          #>$0,x:AAR1        ; Clear Address Attribute Reg1
                     000000
       122
       123    P:00010C05F420        move           #-1,m0             ; Set LINEAR addressing mode
                     FFFFFF
       124    P:00010E60F400        move           #BootStart,r0      ; Set Starting Address of
BOOT:EPROM
                     C00000
       125    P:00011070F400        move           #BootSize,n0       ; Set to Size of BOOT:EPROM
                     020000
       126
       127    P:000112200013        clr            a
       128    P:00011320001B        clr            b
       129    P:000114560600        move           a,x:CKSUM_CALC_B   ; Initialize computed checksum -
> $000000
```

```
               P:000115560700       move    a,x:CKSUM_READ_B      ; Initialize read checksum ->
$000000

                             ;---------- Compute the 24-bit BOOT: Space Checksum ----------
               P:00011606D810       dor            n0,b_loop          ;
                      000003
               P:00011807D88E       move           p:(r0)+,a          ; Get the BOOT:EPROM location
Value
               P:000119200018       add            a,b                ; Compute checksum
                             b_loop

               P:00011A07E08E       move           p:(r0),a           ; Get Checksum from BOOT:EPROM
               P:00011B570600       move           b,x:CKSUM_CALC_B   ; Save the Computed Checksum
value
               P:00011C560700       move           a,x:CKSUM_READ_B   ; Save the Old Checksum value

                      ;-------------------------------------------------------------------------
                      ;-------------------------------------------------------------------------
                             do_p_checksum
               P:00011D08F4B9       movep   #AAR0_value,x:AAR0    ; Set Address Attribute Reg0
                      100715
               P:00011F08F4B8       movep   #AAR1_value,x:AAR1    ; Set Address Attribute Reg1
                      10070A

               P:00012105F420       move    #-1,m0                ; Set LINEAR addressing mode
                      FFFFFF
               P:00012360F400       move    #PMemStart,r0         ; Set Starting Address of P:EPROM
                      100000
               P:00012570F400       move    #PMemSize,n0          ; Set to Size of P:EPROM
                      020000

               P:000127200013       clr     a
               P:00012820001B       clr     b
               P:000129560000       move    a,x:CKSUM_CALC_P ; Initialize computed checksum -> $000000
               P:00012A560100       move    a,x:CKSUM_READ_P      ; Initialize read checksum ->
$000000

                             ;---------- Compute the 24-bit P: Space Checksum ----------
               P:00012B06D810       dor     n0,p_loop             ;
                      000003
               P:00012D07D88E       move    p:(r0)+,a             ; Get the P:EPROM location Value
               P:00012E200018       add     a,b                   ; Compute checksum
                             p_loop

               P:00012F07E08E       move    p:(r0),a              ; Get Checksum from P:EPROM
               P:000130570000       move    b,x:CKSUM_CALC_P      ; Save the Computed Checksum value
               P:000131560100       move    a,x:CKSUM_READ_P      ; Save the Old Checksum value


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

;************************************************************************
                             do_x_checksum
               P:00013205F420       move           #-1,m0             ; Set LINEAR addressing mode
                      FFFFFF
```

```
P:00013460F400        move           #XMemStart,r0      ; Set Starting Address of
X:EPROM
       100000
P:00013670F400        move           #XMemSize,n0       ; Set to Size of X:EPROM
       020000

P:000138200013        clr            a
P:00013920001B        clr            b
P:00013A560200        move           a,x:CKSUM_CALC_X   ; Initialize computed checksum -
> $000000
P:00013B560300        move           a,x:CKSUM_READ_X   ; Initialize read checksum ->
$000000

               ;---------- Compute the 24-bit X: Space Checksum ----------
P:00013C06D810        dor            n0,x_loop          ;
       000003
P:00013E56D800        move           x:(r0)+,a          ; Get the X:EPROM location Value
P:00013F200018        add            a,b                ; Compute checksum
               x_loop

P:00014056E000        move           x:(r0),a           ; Get Checksum from X:EPROM
P:000141570200        move           b,x:CKSUM_CALC_X   ; Save the Computed Checksum
value
P:000142560300        move           a,x:CKSUM_READ_X   ; Save the Old Checksum value

        ;****************************************************************************
        ;****************************************************************************
               do_y_checksum
P:00014305F420        move           #-1,m0             ; Set LINEAR addressing mode
       FFFFFF
P:00014560F400        move           #YMemStart,r0      ; Set Starting Address of
Y:EPROM
       100000
P:00014770F400        move           #YMemSize,n0       ; Set to Size of Y:EPROM
       020000

P:000149200013        clr            a
P:00014A20001B        clr            b
P:00014B560400        move           a,x:CKSUM_CALC_Y   ; Initialize computed checksum -
> $000000
P:00014C560500        move           a,x:CKSUM_READ_Y   ; Initialize read checksum ->
$000000

               ;---------- Compute the 24-bit Y: Space Checksum ----------
P:00014D06D810        dor            n0,y_loop          ;
       000003
P:00014F5ED800        move           y:(r0)+,a          ; Get the Y:EPROM location Value
P:000150200018        add            a,b                ; Compute checksum
               y_loop

P:0001515EE000        move           y:(r0),a           ; Get Checksum from Y:EPROM
P:000152570400        move           b,x:CKSUM_CALC_Y   ; Save the Computed Checksum
value
P:000153560500        move           a,x:CKSUM_READ_Y   ; Save the Read Checksum value

```

```
                 ;************************************************************************

      P:000154050C00          bra      *                          ; Done, Do a dynamic Halt

                              end      eprom1

0     Errors
0     Warnings
```

5.2 16K × 8-bit BOOT EPROM Example

The 16K × 8-bit ‘P’ memory space bootstrap EPROM implementation uses the AMD AM27C128 device (see **Figure 18** for the memory map layout, **Figure 19** for the block diagram and **Figure 21** for the schematic). Using an EPROM in this configuration, a program placed in the EPROM can be loaded into the DSP at boot time and run.

For this example, the DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 250 nS EPROM, 20 wait states are required. This 5 V device is organized as 16K × 8-bits with a 250 nS access time. One memory device composes the 8-bit wide P bus.

Level conversion to and from 3.3 V and 5 V is necessary on the 8-bit data bus to accommodate the 5 V memory device. This is accomplished by using one of Quality Semiconductor’s QS3245 QuickSwitch® 8-bit bus switches. This switch allows 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 without adding any significant (0.25 nS) propagation delay.

During reset with Mode 1 selected, the DSP boot code configures AA1 for program accesses in the address range from $D00000–$DFFFFF. The boot code then loads bytes from the EPROM, packs them into 24-bit words and stores them into program RAM. The first word, three packed bytes, read from the EPROM indicates the number of words to load. The second word from the EPROM contains the starting load address for the packed data, which is also the address that gains program control after the program load is completed. The program listed in **Example 4** initializes the Address Attribute Register and calculates an 8-bit checksum value to compare with the checksum value stored in the last location of the 16K EPROM bank.

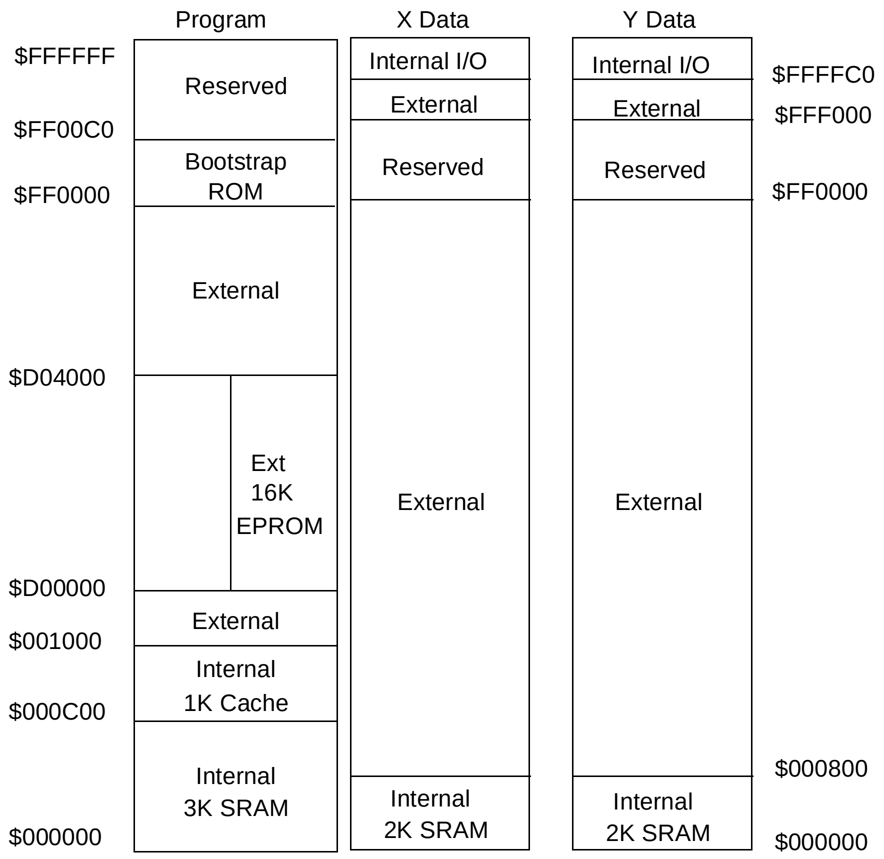

Figure 18. 16K × 8-bit Boot EPROM Memory Map

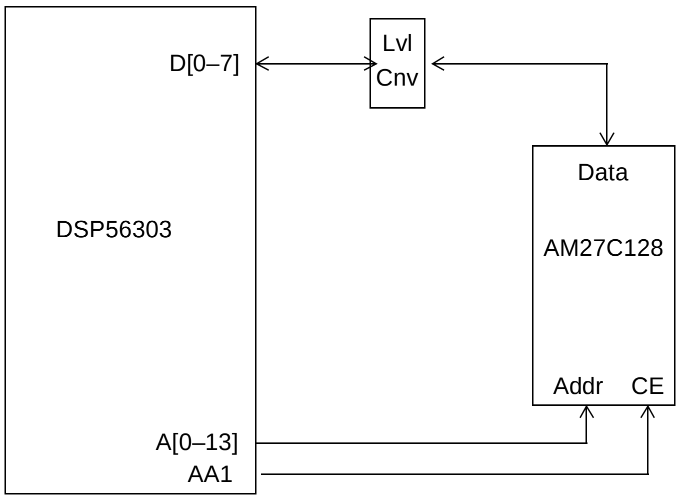

Figure 19. 16K × 8-bit EPROM Memory Example.

5.2.1 EPROM Timing Requirements

For the EPROM device to work properly, its timing requirements must be met. Following are the timing requirements for the AM27C128-250 16K × 8-bit 250 nS EPROM. **Table 8** shows the memory read timing specification values used in the memory read cycle timing diagram, **Figure 20**

Table 8. AM27C128-250 Memory Read Timing Specifications

Read Cycle Parameter	Symbol	Min	Max
Read Cycle Time	t_{RC}	250 nS	—
Address to Output Delay	t_{ACC}	—	250 nS
Chip Enable to Output Delay	t_{CE}	—	250 nS
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	—

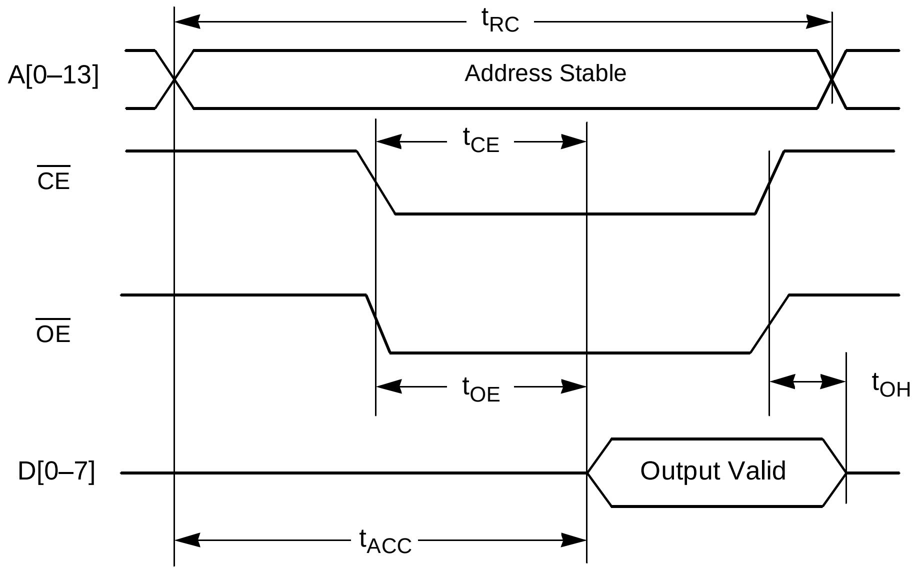

Figure 20. AM27C128 Memory Read Cycle Timing Diagram

5.2.2 DSP56303 Port A Timing Requirements and Register Settings

For optimal use of the 64K × 8-bit BOOT EPROM 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 Register (PCTL). 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 = 0x0
- Low-power divider value = 1, bits 12-14 = 0x0
- VCO multiplication value = 20, bits 0-11 = 0x013
- 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 0x0E0013.

Address Attribute Pin 1 enables, via EPROM $\overline{\text{CE}}$, external 16K EPROM memory bank accesses in the address range from $D00000–$D03FFF during program space requests. Configure the memory address space requirements for the AA1 using the Address Attribute Register 1(AAR1). The AAR1 register value combines the following bits for each feature:

- Specify the external memory access type as asynchronous SRAM, bits 0-1 = 0x1.
- Pull the AA pin high when selected, bit 2 = 0.
- 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 = 0xA;
- Specify the most significant portion of the address to compare, bits 12–23 = 0x100.

The value loaded into the AAR1 register is 0xD00A09. The value loaded into AAR0, AAR2, and AAR3 is 0x000000.

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

- Address attribute area zero wait states, bits 0–4 = 0x0
- Address attribute area one wait state, bits 5–9 = 0x14
- Address attribute area two wait states, bits 10–12 = 0x0
- Address attribute area three wait states, bits 13–15 = 0x0
- Default address area wait states, bits 16-20 = 0x0
- 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 register is 0x000280.

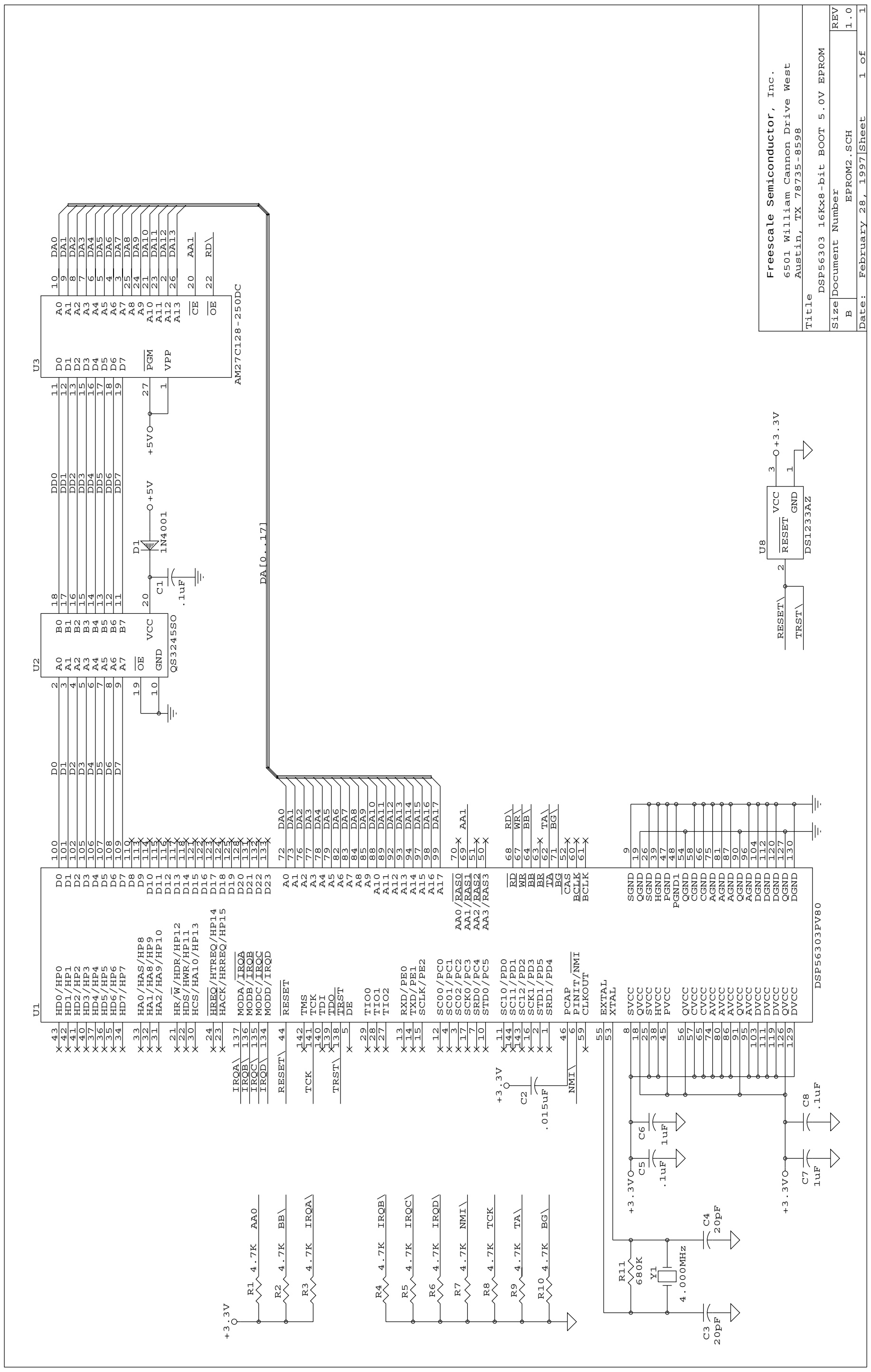

Figure 21. 16Kx8 BOOT EPROM Schematic

Example 4. 16K x 8-bit BOOT EPROM Checksum Program

```
DSP56300 Assembler Version 6.0.1.6   97-02-26 21:35:23 eprom2.asm
```

```
                                   page    132,60,3,3,
                       ;
                       ;     eprom2.asm - Simple program to calculate the 8-bit Checksum for
                       ;                  a 16K x 8-bit block of EPROM memory using a DSP56303.
                       ;                  Contains: Initialization routine,
                       ;                                 Routine to calculate 8-bit checksum.
                       ;
                       ;       The program runs in Internal P:RAM to calculate the checksum on
                       ;       External P:EPROM from $D00000 - $D03FFF @ 20w/s
                       ;

       D00000          MemStart       equ     $D00000
       D04000          MemEnd         equ     $D04000
       004000          MemSize        equ     MemEnd-MemStart    ; Last Word is stored Checksum
value

                       ;--- Program Specific Storage Locations (X DATA SPACE)
                       NEW_CHECKSUM
       000000                         equ     $000000            ; Computed Checksum Value
                       OLD_CHECKSUM
       000001                         equ     $000001            ; Old Checksum from PEROM

                       ;--- DSP56303 Control Registers (X I/O SPACE)
       FFFFFB          BCR            equ     $FFFFFB            ; Bus Control Register
       FFFFFD          PCTL           equ     $FFFFFD            ; PLL Control Register
       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)*4.00 MHz=80.00 MHz
       000000          crystal        equ     0                  ; No, Crystal not less than
200 kHz
       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

                       ;--- AAR1 value = 0xD00A09
       000001          acctype1       equ     1              ; External Memory access type = 0x1
       000000          aahigh1        equ     0              ; Enable AA1 pin to be low when
selected
       000008          aap1           equ     $8             ; Yes, Enable AA1 pin on ext 'P'
accesses
       000000          aax1           equ     0              ; No, Enable AA1 pin on ext 'X'
accesses
       000000          aay1           equ     0              ; No, Enable AA1 pin on ext 'Y'
accesses
       000000          aswap1         equ     0              ; No, Enable address bus swap
       000000          enpack1        equ     0              ; No, Enable packing/unpacking logic
       000A00          nadd1          equ     $000A00        ; Compare 10 address bits
       D00000          msadd1         equ     $D00000        ; Most significant portion of
address,
       48                                                    ; $D00000 - D03fff, to compare.
       49                                                    ; (1101,0000,00xx,xxxx,xxxx,xxxx)
       50      D00A09         AAR1_value     equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       51
       52
       53                     ;--- BCR value = 0x000280
       54      000000         aaa0ws         equ    0            ; Address Attribute Area 0 w/s = 0
       55      000280         aaa1ws         equ    $280         ; Address Attribute Area 1 w/s = 20
       56      000000         aaa2ws         equ    0            ; Address Attribute Area 2 w/s = 0
       57      000000         aaa3ws         equ    0            ; Address Attribute Area 3 w/s = 0
       58      000000         defws          equ    0            ; Default Address Area w/s = 0
       59      000000         busss          equ    0            ; Bus state status = 0
       60      000000         enblh          equ    0            ; Enable Bus Lock Hold = 0
       61      000000         enbrh          equ    0            ; Enable Bus Request Hold = 0
       62      000280         BCR_value      equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       63
       64                     ;-------------------------------------------------------------------
       65                     ;       Header for DSP56303 BOOT Code
       66                     ;-------------------------------------------------------------------
       67      P:0000FE                      org    p:$100-2
       68
       69      P:0000FE                      dc     pgm_end-eprom2      ; Number of words in program
       70      P:0000FF                      dc     eprom2              ; Starting address for program
       71
       72
       73                     ;-------------------------------------------------------------------
       74      P:000100                      org           p:$100                ; Keep the program in
internal RAM
       75
       76                                          eprom2
       77      P:0001000D1080                bsr           init                  ; Initialize DSP
                      000005
       78
       79      P:0001020D1080                bsr           calc_checksum         ; Calculate Checksum
of EPROM
                      00000C
       80
       81                     _done
       82      P:000104050C00                bra    *                            ; DONE, Do a dynamic
HALT
       83
       84
       85              ;----------------------------------------------------------------------
       86              ; Initialization Section
       87              ;----------------------------------------------------------------------
       88                     init
       89      P:00010508F4BD                movep         #PCTL_value,x:PCTL           ; Set PLL
Control Register
                      0E0013
       90      P:00010705F439                movec         #$080000,SR                  ; Enable 1K
Cache
                      080000
       91      P:00010908F4BB                movep         #BCR_value,x:BCR             ; Set external
wait states
```

```
                    000280
       92      P:00010B08F4B8              movep   #AAR1_value,x:AAR1                  ; Set Address
Attribute Reg1
                    D00A09
       93
       94      P:00010D00000C              rts
       95
       96              ;-------------------------------------------------------------------------
       97              ; Routine to Calculate 8-bit Checksum
       98              ;-------------------------------------------------------------------------
       99                      calc_checksum
       100     P:00010E05F420        move    #-1,m0                 ; Set LINEAR addressing mode
                    FFFFFF
       101     P:00011060F400        move    #MemStart,r0           ; Set Starting Address of EPROM
                    D00000
       102     P:00011270F400        move    #MemSize-1,n0          ; Set to Size of Flash - Checksum
                    003FFF
       103
       104     P:000114200013        clr     a
       105     P:00011520001B        clr     b
       106     P:000116540000        move    a1,x:NEW_CHECKSUM          ; Initialize computed checksum -
> $000000
       107     P:000117540100        move    a1,x:OLD_CHECKSUM          ; Initialize read checksum ->
$000000
       108
       109                     ; Compute the 8-bit Checksum
       110     P:00011844F400        move    #>$FF,x0                   ; Set lower Byte Mask
                    0000FF
       111
       112     P:00011A06D810        dor     n0,_ploop
                    000004
       113     P:00011C07D88C        move    p:(r0)+,a1                 ; Get the EPROM location Value
       114     P:00011D200046        and     x0,a                       ; Mask for lower byte
       115     P:00011E200018        add     a,b                        ; Compute checksum
       116                     _ploop
       117
       118     P:00011F07E08C        move    p:(r0),a1              ; Get EPROM's Old Checksum value
       119     P:00012020004E        and     x0,b                   ; Limit calculated Checksum to lower
byte
       120     P:000121200046        and     x0,a                   ; Limit Old Checksum to lower byte
       121     P:000122550000        move           b1,x:NEW_CHECKSUM            ; Save the Computed
Checksum value
       122     P:000123540100        move           a1,x:OLD_CHECKSUM            ; Save the Old
Checksum value
       123
       124     P:000124 00000C       rts
       125
       126             ;-------------------------------------------------------------------------
       127             pgm_end
       128
       129                           end      eprom2

       0       Errors
       0       Warnings
```

5.3 512K × 8-bit BOOT/Overlay EPROM Example

The 512K × 8-bit boot with X data space program overlay EPROM implementation uses the Atmel AT27LV040-15 device (see **Figure 22** for the boot memory map layout, **Figure 23** for the overlay memory map layout, **Figure 24** for the block diagram, and **Figure 26** for the schematic). A program placed into the EPROM can be loaded into the DSP at boot time and run. The booted program can then load additional programs and data from the EPROM into the DSP, via overlay techniques.

The DSP core runs at 80 MHz and the input frequency source is a 4.000 MHz crystal. For a 150 nS EPROM, 12 wait states are required. This 3.3 V device is organized as 512K x 8-bits with a 150 nS access time. One memory device composes the 8-bit wide boot bus.

During reset with Mode 1 selected, the DSP boot code configures AA1 for program accesses in the address range from $D00000–$DFFFFF (see **Figure 22**). The boot code then loads bytes from the EPROM, packs them into 24-bit words, and stores them into program RAM. The first word, three packed bytes, read from the EPROM indicates the number of words to load. The second word from the EPROM contains the starting load address for the packed data, which is also the address which gains program control after the program is loaded.

Under user program control, AA1 can then be configured for X data space accesses in the address range from $100000–$17FFFF (see **Figure 24**). AA0 can then be configured to select the upper half of the 512K EPROM by enabling EPROM A18 as a one when X:$14FFFF–X:$17FFFF is selected. The program listed in **Example 5** initializes the Address Attribute Register and calculates an 8-bit checksum value used to compare with the checksum value stored in the last location of the 512K EPROM bank.

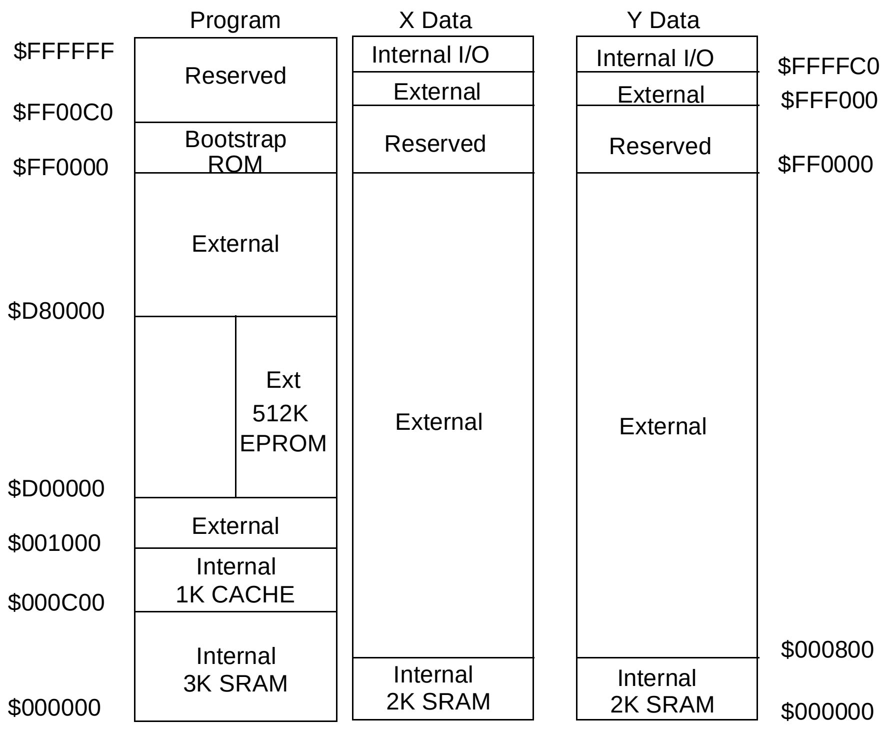

Figure 22. 512K x 8-bit Boot EPROM Memory Map

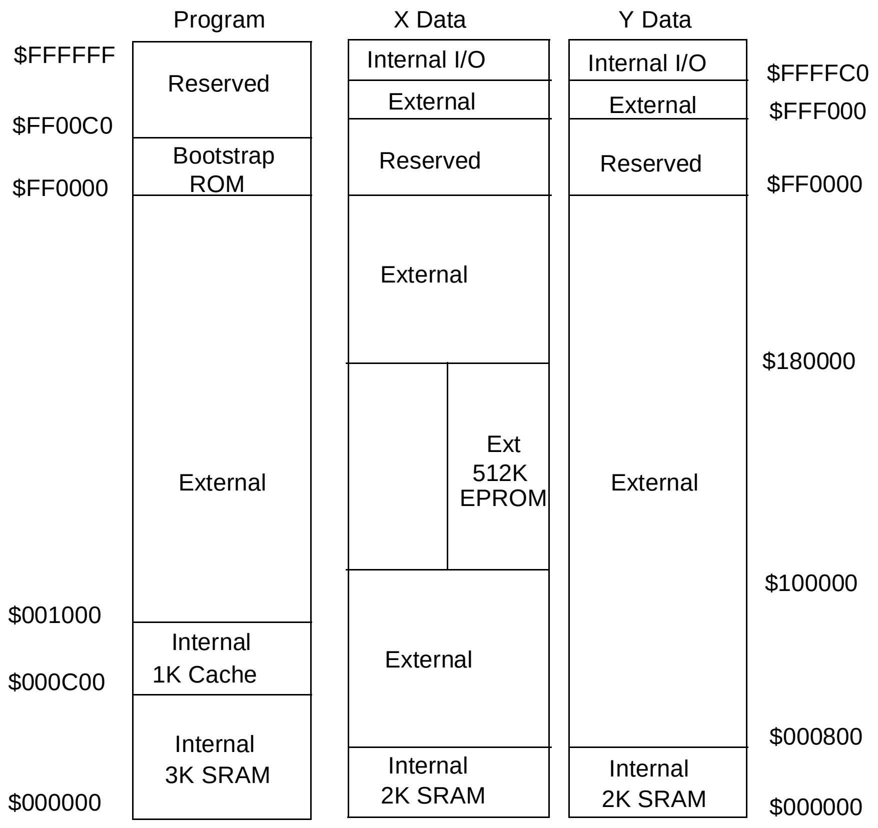

Figure 23. 512K x 8-bit Overlay EPROM Memory Map

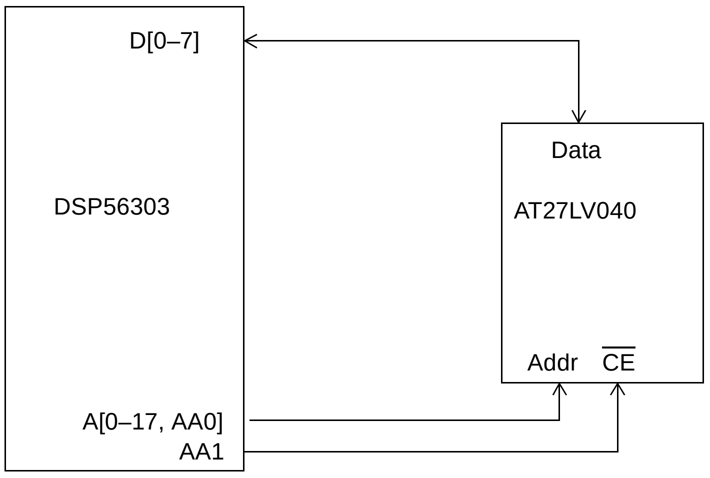

Figure 24. 512K x 8-bit EPROM Memory Example.

5.3.1 EPROM Timing Requirements

Following are the timing requirements for the AT27LV040-15 512K × 8-bit 150 nS EPROM. **Table 9** contains the memory read timing specification values used in the memory read cycle timing diagram shown in **Figure 25**

Table 9. AT27LV040-15 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}	—	60 nS
Output Hold Time from Address, $\overline{CE}$ or $\overline{OE}$, whichever occurs first.	t_{OH}	0 nS	—

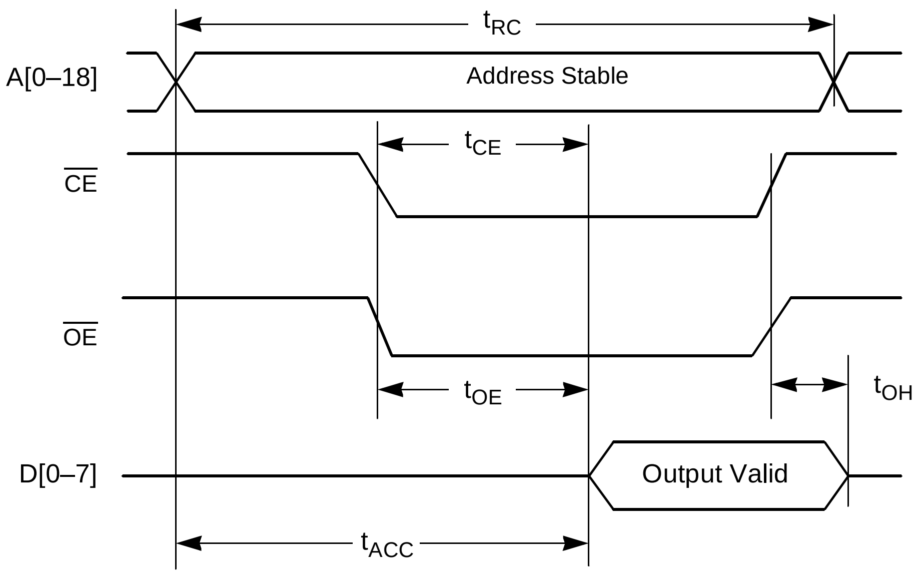

Figure 25. AT27LV040 Memory Read Cycle Timing Diagram.

5.3.2 DSP56303 Port A Timing Requirements and Register Settings

For most efficient use of the 512K × 8-bit boot/overlay EPROM 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 Register (PCTL). 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 = 0x0
- Low-power Divider value = 1, bits 12-14 = 0x0
- VCO Multiplication value = 20, bits 0-11 = 0x013
- 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 0x0E0013.

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

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = 0x1.
- 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 = 0x5.
- Specify the most significant portion of the address to compare, bits 12–23 = 0x100.

The value loaded into the AAR1 register is 0x100591.

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

- Specify the external memory access type as asynchronous SRAM, bits 0–1 = 0x1.
- 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 = 0x6.
- Specify the most significant portion of address to compare, bits 12–23 = 0x140.

The value loaded into the AAR0 register is 0x140695. The value loaded into AAR2 and AAR3 is 0x000000.

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

- Address attribute area zero wait states, bits 0–4 = 0xC
- Address attribute area one wait states, bits 5–9 = 0xC
- Address attribute area two wait states, bits 10–12 = 0x0
- Address attribute area three wait states, bits 13–15 = 0x0

- Default address area wait states, bits 16–20 = 0x0
- 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 register is 0x00018C.

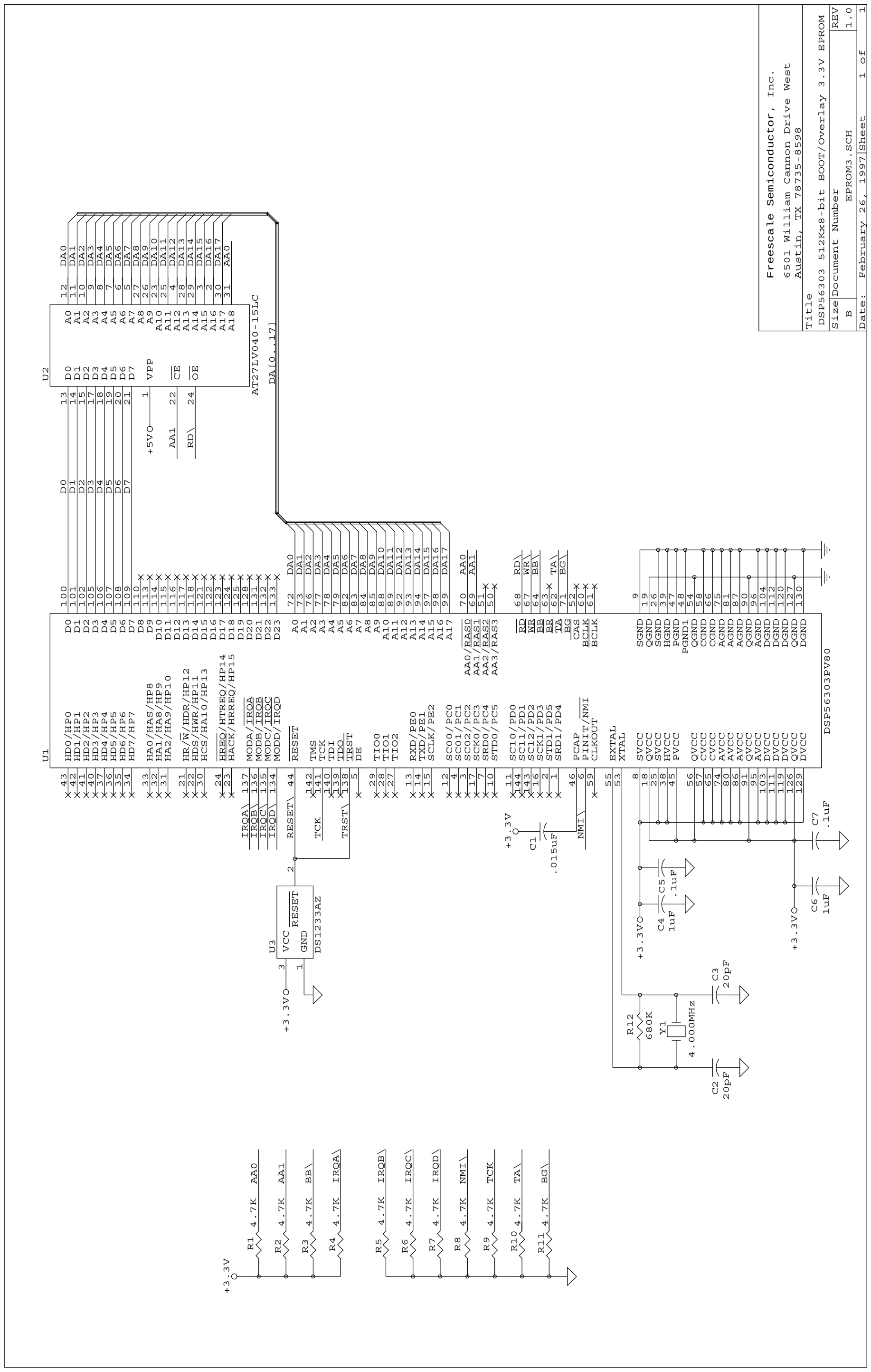

Figure 26. 512Kx8 BOOT/Overlay EPROM Schematic

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

```
        DSP56300 Assembler  Version 6.0.1.6   97-02-27  21:07:11  eprom3.asm



        1                                   page    132,60,3,3,
        2                       ;
        3                       ;     eprom3.asm - Simple program to calculate the 8-bit Checksum for
        4                       ;                   a 512K x 8-bit block of EPROM memory using a DSP56303.
        5                       ;                   Contains:  Initialization routine,
        6                       ;                               Routine to calculate 8-bit checksum.
        7                       ;
        8                       ;       The program runs in Internal P:RAM to calculate the checksum on
        9                       ;       External X:EPROM from $100000 - $17FFFF @ 12w/s
        10                      ;
        11
        12      100000          MemStart       equ     $100000
        13      180000          MemEnd         equ     $180000
        14      080000          MemSize        equ     MemEnd-MemStart    ; Last Word is stored Checksum
value
        15
        16                      ;--- Program Specific Storage Locations (X DATA SPACE)
        17                      NEW_CHECKSUM
        18      000000                         equ     $000000        ; Computed Checksum Value
        19                      OLD_CHECKSUM
        20      000001                         equ     $000001        ; Old Checksum from EPROM
        21
        22                      ;--- DSP56303 Control Registers (X I/O SPACE)
        23      FFFFFB          BCR            equ     $FFFFFB        ; Bus Control Register
        24      FFFFFD          PCTL           equ     $FFFFFD        ; PLL Control Register
        25      FFFFF9          AAR0           equ     $FFFFF9        ; Address Attribute Register 0
        26      FFFFF8          AAR1           equ     $FFFFF8        ; Address Attribute Register 1
        27
        28                      ;--- PCTL value = 0x0E0013
        29      000000          prediv         equ     0              ; Pre-Divider = 1
        30      000000          lowdiv         equ     0              ; Low Power Divider = 1
        31      000013          pllmul         equ     19             ; VCO Mult = 20;
(19+1)*4.00 MHz=80.00 MHz
        32      000000          crystal        equ     0              ; No, Crystal not less than 200 kHz
        33      000000          disXTAL        equ     0              ; No, do not disable crystal use
        34      020000          pllstop        equ     $020000        ; Yes, PLL runs during STOP
        35      040000          enpll          equ     $040000        ; Yes, enable PLL operation
        36      080000          disclk         equ     $080000        ; Yes, disable CORE clock output
        37      0E0013          PCTL_value     equ
prediv+lowdiv+pllmul+crystal+disXTAL+pllstop+enpll+disclk
        38
        39                      ;--- AAR0 value = 0x140695
        40      000001          acctype0       equ     1              ; External Memory access type = 0x1
        41      000004          aahigh0        equ     $4             ; Enable AA0 pin to be high when
selected
        42      000000          aap0           equ     0              ; No, Enable AA0 pin on ext 'P'
accesses
        43      000010          aax0           equ     $10            ; Yes, Enable AA0 pin on ext 'X'
accesses
        44      000000          aay0           equ     0              ; No, Enable AA0 pin on ext 'Y'
accesses
```

```
       45     000000         aswap0        equ     0              ; No, Enable address bus swap
       46     000080         enpack0       equ     $80            ; Yes, Enable packing/unpacking
logic
       47     000600         nadd0         equ     $000600        ; Compare 6 address bits
       48     140000         msadd0        equ     $140000        ; Most significant portion of
address,
       49                                                         ; $140000 - 17ffff, to compare.
       50                                                         ; (0001,01xx,xxxx,xxxx,xxxx,xxxx)
       51     140695         AAR0_value    equ
acctype0+aahigh0+aap0+aax0+aay0+aswap0+enpack0+nadd0+msadd0
       52
       53                    ;--- AAR1 value = 0x100591
       54     000001         acctype1      equ     1              ; External Memory access type = 0x1
       55     000000         aahigh1       equ     0              ; Enable AA1 pin to be low when
selected
       56     000000         aap1          equ     0              ; No, Enable AA1 pin on ext P
accesses
       57     000010         aax1          equ     $10            ; Yes, Enable AA1 pin on ext X
accesses
       58     000000         aay1          equ     0              ; No, Enable AA1 pin on ext Y
accesses
       59     000000         aswap1        equ     0              ; No, Enable address bus swap
       60     000080         enpack1       equ     $80            ; Yes, Enable packing/unpacking
logic
       61     000500         nadd1         equ     $000500        ; Compare 5 address bits
       62     100000         msadd1        equ     $100000        ; Most significant portion of
address,
       63                                                         ; $100000 - 17ffff, to compare.
       64                                                         ; (0001,0xxx,xxxx,xxxx,xxxx,xxxx)
       65     100591         AAR1_value    equ
acctype1+aahigh1+aap1+aax1+aay1+aswap1+enpack1+nadd1+msadd1
       66
       67
       68                    ;--- BCR value = 0x00018C
       69     00000C         aaa0ws        equ     $C             ; Address Attribute Area 0 w/s = 12
       70     000180         aaa1ws        equ     $180           ; Address Attribute Area 1 w/s = 12
       71     000000         aaa2ws        equ     0              ; Address Attribute Area 2 w/s = 0
       72     000000         aaa3ws        equ     0              ; Address Attribute Area 3 w/s = 0
       73     000000         defws         equ     0              ; Default Address Area w/s = 0
       74     000000         busss         equ     0              ; Bus state status = 0
       75     000000         enblh         equ     0                  ; Enable Bus Lock Hold = 0
       76     000000         enbrh         equ     0                  ; Enable Bus Request Hold = 0
       77     00018C         BCR_value     equ
aaa0ws+aaa1ws+aaa2ws+aaa3ws+defws+busss+enblh+enbrh
       78
       79                    ;---------------------------------------------------------------------
       80                    ;       Header for DSP56303 Boot Code
       81                    ;---------------------------------------------------------------------
       82     P:0000FE                     org     p:$100-2
       83
       84     P:0000FE                     dc      pgm_end-eprom3     ; Number of words in program
       85     P:0000FF                     dc      eprom3             ; Starting address for program
       86
       87
       88                    ;---------------------------------------------------------------------
```

```
        P:000100                      org     p:$100             ; Keep program in internal RAM

                      eprom3
        P:0001000D1080                bsr            init        ; Initialize DSP
                000005

        P:0001020D1080                bsr            calc_checksum ; Calculate Checksum of EPROM
                000010

                      _done
        P:000104050C00                bra            *           ; DONE, Do a dynamic HALT


                      ;----------------------------------------------------------------------
                      ; Initialization Section
                      ;----------------------------------------------------------------------
                      init
        P:00010508F4BD        movep         #PCTL_value,x:PCTL ; Set PLL Control Register
                0E0013
        P:00010705F43A        movec         #$004000,OMR   ; Disable Address Attribute
Priorities
                004000
        P:00010905F439        movec         #$080000,SR        ; Enable 1K Cache
                080000
        P:00010B08F4BB        movep         #BCR_value,x:BCR   ; Set external wait states
                00018C
        P:00010D08F4B9        movep         #AAR0_value,x:AAR0 ; Set Address Attribute Reg0
                140695
        P:00010F08F4B8        movep         #AAR1_value,x:AAR1 ; Set Address Attribute Reg1
                100591

        P:00011100000C        rts

                ;----------------------------------------------------------------------
                ; Routine to Calculate 8-bit Checksum
                ;----------------------------------------------------------------------
                      calc_checksum
        P:00011205F420        move          #-1,m0             ; Set LINEAR addressing mode
                FFFFFF
        P:00011460F400        move          #MemStart,r0       ; Set Starting Address EPROM
                100000
        P:00011670F400        move          #MemSize-1,n0      ; Set to Size of Flash -
Checksum
                07FFFF

        P:000118200013        clr           a
        P:00011920001B        clr           b
        P:00011A540000        move          a1,x:NEW_CHECKSUM ; Initialize computed checksum ->
$000000
        P:00011B540100        move          a1,x:OLD_CHECKSUM ; Initialize read checksum ->
$000000

                      ; Compute the 8-bit Checksum
        P:00011C44F400        move          #>$FF,x0           ; Set lower Byte Mask
                0000FF

        P:00011E06D810        dor           n0,_ploop
                000004
        P:00012054D800        move          x:(r0)+,a1          ; Get the EPROM location Value
        P:000121200046        and           x0,a                ; Mask for lower byte
        P:000122200018        add           a,b                 ; Compute checksum
                          _ploop

        P:00012354E000        move          x:(r0),a1           ; Get EPROM's Old Checksum value
        P:00012420004E        and           x0,b            ; Limit calculated Checksum to lower
byte
        P:000125200046        and           x0,a                ; Limit Old Checksum to lower
byte
        P:000126550000        move          b1,x:NEW_CHECKSUM   ; Save Computed Checksum value
        P:000127540100        move          a1,x:OLD_CHECKSUM   ; Save the Old Checksum value

        P:00012800000C        rts


               ;----------------------------------------------------------------------------
               pgm_end

                              end     eprom3

0       Errors
0       Warnings
```

6 Advantages

EPROM, EEPROM, PROM, and ROM memory provide the designer with a non-volatile memory storage at a reasonable speed. EPROM, EEPROM, PROM and ROM memory allows the DSP to load programs and run immediately after RESET, as in an embedded application. Programs stored in the EPROM's non-volatile memory storage can be read and run directly as 24-bit data or indirectly as 8-bit data, using the DMA to access, pack and store the data into DSP Program memory. Additionally, programs and data can be stored in the EEPROM's non-volatile memory with the device still in the circuit. However, for new programs or data to be stored in the EPROM's non-volatile memory, the EPROM must be removed from the circuit, erased by exposure to Ultraviolet Light (UV), and then reprogrammed by an EPROM programmer. For development, this is very inconvenient. However, for production this is possibly very cost-effective. ROM parts are purchased with program data already stored on the part. This allows easier assembly and lower cost compared to EPROM, EEPROM, and PROM.

6.1 Flexible Memory Configuration Capabilities

EPROM, EEPROM, PROM, and ROM memory provide the designer with flexible memory configurations for the DSP56300 family. Using a common hardware memory design with the DSP56300 family memory expansion port allows the memory bank to be logically configured for use in various memory space arrangements by simply setting up the memory expansion port's address attribute control registers. Configuring the memory expansion port address attribute control registers allows the memory bank to be configured for the following:

- Seven different memory space arrangements if one address attribute control line is used:

1. Memory bank P space
2. Memory bank X space

3. Memory bank Y space
4. Memory bank P/X space
5. Memory bank P/Y space
6. Memory bank X/Y space
7. Memory bank P/X/Y space

- Thirteen different memory space arrangements if two address attribute control lines are used:

1. Memory bank P space
2. Memory bank X space
3. Memory bank Y space
4. Memory bank P/X space
5. Memory bank P/Y space
6. Memory bank X/Y' space
7. Memory bank P/X/Y space
8. Memory bank/2 P/X and memory bank/2 Y space
9. Memory bank/2 P/Y and memory bank/2 X space
10. Memory bank/2 P and memory bank/2 X/Y space
11. Memory bank/2 P and memory bank/2 X space
12. Memory bank/2 P and memory bank/2 Y space
13. Memory bank/2 X and memory bank/2 Y space

- Fourteen different memory space arrangements if three address attribute control lines are used:

1. Memory bank/2 P space
2. Memory bank/2 X space
3. Memory bank/2 Y space
4. Memory bank/2 P/X space
5. Memory bank/2 P/Y space
6. Memory bank/2 X/Y space
7. Memory bank/2 P/X/Y space
8. Memory bank/2 P/X and memory bank/2 Y space
9. Memory bank/2 P/Y and memory bank/2 X space
10. Memory bank/2 P and memory bank/2 X/Y space
11. Memory bank/2 P and memory bank/2 X space

12. Memory bank/2 P and memory bank/2 Y space
13. Memory bank/2 ‘X’ and memory bank/2 Y space
14. Memory bank/4 boot, memory bank/4 P, memory bank/4 X and memory bank/4 Y space

6.1.1 Immediate Load and Run Capability

EPROM, EEPROM, PROM, and ROM memory allows the DSP to load programs and run them immediately after reset as in an embedded application. Programs stored in the EPROM non-volatile memory storage can be read and run directly as 24-bit data. Arranging the EPROM, EEPROM, PROM, or ROM memory in a 24-bit bank of memory allows the DSP to run directly from the memory after a reset. This was shown in the 128K × 24-bit boot, 128K × 24-bit program, 128K × 24-bit X, and 128K × 24-bit Y EPROM example.

6.1.2 Program and Data Overlay Capability

Programs stored in the EPROM non-volatile memory storage can be read and run indirectly as 8-bit data, using a program or DMA to access, pack and store the data into DSP program memory. Arranging the EPROM, EEPROM, PROM, and ROM memory in an 8-bit bank allows the DSP at boot time to load and pack the 8-bit memory values into 24-bit words for storage into program RAM. After the data loads, the DSP turns over program control to this newly loaded program. This program, overlay loader, can then programmatically or via DMA control, load additional data from the 8-bit memory bank into program, X data or Y data spaces for later execution or use. This was shown in the 2K × 8-bit Boot EEPROM, 32K × 8-bit EEPROM, 16K × 8-bit EPROM and 512K × 8-bit EPROM examples.

6.1.3 EEPROM In-Circuit Reprogrammability

Programs and data can be stored in the EEPROM non-volatile memory with the device still in the circuit. This allows for easy in-circuit program and data modification or upgrading. For development, this is very convenient, allowing for fast test-program-retest development cycles.

6.2 Disadvantages

For new programs or data to be stored in an EPROM, the EPROM must be removed from the circuit, erased by exposure to Ultraviolet Light (UV), and then reprogrammed by an EPROM programmer. For development, this is very inconvenient.

New programs or data storage in a PROM requires a new unprogrammed PROM to be programmed by a PROM programmer for the part. Then the old parts have to be removed from existing boards. This is very inconvenient for a product and impractical during development.

For new programs or data to be stored in a ROM part, a new ROM mask and new parts must be manufactured. Then the old parts have to be removed from existing boards. This is very inconvenient for a product and impractical during development.

6.3 Speed Selection

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

7 References

[1] *DSP56300 24-bit Digital Signal Processor Family Manual* (DSP56300FM), Freescale Semiconductor, Inc.

[2] *DSP56303 Technical Data Sheet* (DSP56303).

[3] *DSP56303 24-bit Digital Signal Processor User's Manual* (DSP56303UM).

[4] *DSP56303EVM User's Manual*.

[5] *Atmel Corporation Nonvolatile Memory Data Book*, Atmel, May 1996.

[6] *Advanced Micro Devices CMOS Memory Products 1991 Data Book/Handbook*, AMD, 1991.

[7] *Quality Semiconductor QuickSwitch® Products Databook*, Quality Semiconductor, 1995.

[8] Quality Semiconductor Application Note AN-11, *Bus Switches Provide 5v and 3v Logic Conversion with Zero Delay*, 1995.

NOTES:

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.: APR27
Rev. 1
10/2005