



Interfacing NAND Flash Memory with ADSP-21161 SHARC® Processors

Contributed by Aseem Vasudev Prabhugaonkar

Rev 1 – November 17, 2005

Introduction

NAND flash provides an alternative to hard drives, especially in portable and handheld systems. Cellular phones, personal digital assistants (PDAs), digital cameras, MP3 players, and other mobile computing, communications, and consumer products use flash memory as their primary storage media. Applications that use NAND flash memory to store information such as images, data, and music require high density.

This application note describes how to interface NAND flash memory to ADSP-21161 SHARC® processors, even though the ADSP-21161 does not have a NAND flash controller on-chip. The interface is demonstrated by code examples, which are validated on a hardware platform. The code examples are provided in “C” and assembly languages. Two different interfacing schemes are used to provide more flexibility to the system designer while designing a flash memory interface. The memory device used to demonstrate this interface is the K9F1208U0B, manufactured by Samsung Electronics.

NAND Flash versus NOR Flash

The most important parameter regarding memories is cost per bit. In semiconductor memory, bit cost depends on memory cell area per bit. The cell area of NAND flash memory is smaller than that of NOR flash, hence NAND flash can be less expensive than NOR flash.

Functionally, the primary difference between NAND flash and NOR flash is that NAND flash is block accessible and NOR flash is byte accessible.

NAND flash has the advantage of short erasing and programming times. The programming current into the floating gate is very small because NAND flash uses Fowler-Nordheim tunneling for erasing and programming. Therefore, power consumption for programming does not significantly increase, even as the number of memory cells being programmed increases. As a result, many NAND flash memory cells can be programmed simultaneously, decreasing the programming time per byte. Conversely, NOR flash can be programmed by byte or word only. Since it uses the hot electron injection mechanism for programming, it also consumes more power and the programming time per byte is longer. Typically, the programming time for NOR flash is more than an order of magnitude greater than that of NAND flash.

From the system designer’s point of view, the most striking difference between NAND flash and NOR flash is the hardware interface. NOR flash has a fully memory-mapped random access interface like an EPROM, with dedicated address lines and data lines. This interface makes it easy to “boot” NOR flash systems. On the other hand, NAND flash has no dedicated address lines. It is controlled using an indirect I/O-like interface and by sending commands and addresses through an 8-bit bus to an internal command and address

register. Therefore, NOR flash memories are mainly used to boot-load a processor, and NAND flash memories are used to store large amounts of audio and video data.

Overview of the ADSP-21161 Processor's External Port

NAND flash is interfaced to the external port of ADSP-21161 processor, which extends address and data buses off-chip. Using these buses and external control lines, systems can interface the processor with external devices such as memory, 8-, 16-, or 32-bit host processors, and other processors. In addition to its on-chip SRAM, the processor provides addressing of up to 64 mega words SRAM or SBSRAM or 254 mega words of off-chip SDRAM memory through its external port. The external memory map is divided in to four equal-sized banks so that the systems can accommodate I/O devices with different timing requirements.

K9F1208U0B NAND Flash

The K9F1208U0B is offered in a 64M x 8-bit configuration. The K9F1208X0B is 512 Mbits with spare 16 Mbits capacity. The device is offered in 1.8V, 2.7V, and 3.3V supply voltage ranges. Its NAND cell provides the most cost-effective solution for solid-state mass storage. A program operation can be performed in 200 μ s (typical) on the 528-byte page. An erase operation can be performed in 2 ms (typical) on a 16-Kbyte block. Data in the page can be read out at 50 ns (60 ns for K9F1208R0B) cycle time per byte. The I/O pins serve as the ports for address and data input/output as well as command input.

NAND Flash Interface

NAND flash, which is controlled using I/O pins, does not have dedicated address and data lines.

Thus, the programmer must use additional flash control signals to send commands that perform various tasks like read and write. The basic commands supported by NAND flash memories are block erase, block program, read status, and block read. Figure 1 shows the pinout of NAND flash in a TSOP1 package.

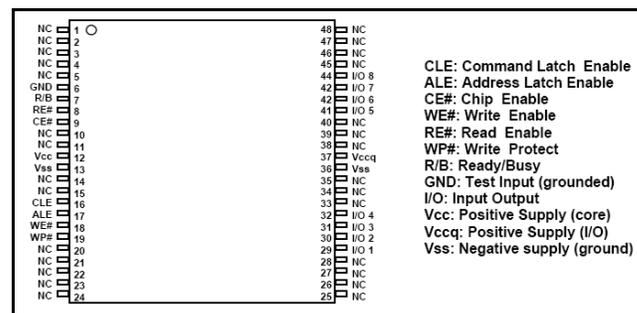


Figure 1. NAND Flash Pinout in a TSOP1 Package

Assert chip enable (CE#) logic low to access the device. It enables the NAND flash to accept bytes written to the chip when write enable (WE#) is asserted low (and to enable the output of a data byte when read enable (RE#) is asserted low). When CE# is high, the chip ignores RE# and WE# and the I/O is three-stated. Command Latch Enable (CLE) is used to send commands to the device when CE# is asserted. Address Latch Enable (ALE) is used to latch the address into the flash's address register. Table 1 indicates the selected internal register for valid combinations of CLE and ALE.

ALE	CLE	Register Selected
0	0	Data Register
0	1	Command Register
1	0	Address Register
1	1	Not Defined

Table 1. Combinations of ALE and CLE, when CS# is Asserted

The flash memory is accessed in terms of column, page, and block. Read and program operations take place on a per-page basis. An erase operation takes place on a block basis. One page consists of 528 bytes. The size of the data

register is therefore 528 bytes. One block comprises 32 such pages. There are three basic operations in a NAND flash: reading a page, programming a page, and erasing a block. Figure 2 shows the organization of K9F1208U0B flash memory in terms of pages and blocks.

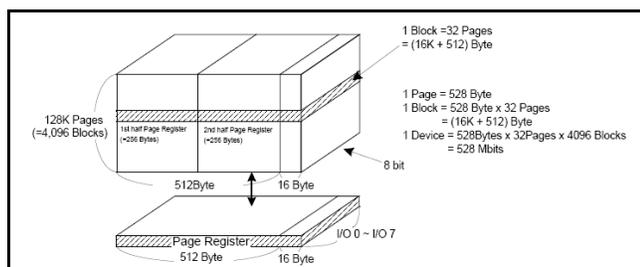


Figure 2. K9F1208U0B Flash Memory Organization

The command phase comprises a single byte transfer to the command register. The address phase comprises a series of byte transfers, which depend on the size of the flash. The address phase consists of the column address byte followed by a series of row address bytes. The number of row address bytes depends on the number of blocks, and hence the size of the memory device.

Flash Commands

This discusses the primary NAND flash commands. The primary and very commonly used flash memory commands are read page, program page, read status, and erase block. Every command (except read status) consists of a command write phase followed by address write phase. The read status command does not have an address write phase. The command is written into the flash's command register followed by the start address for the read or program operation latched into address register.

READ Operation

After issuing the read command and the address, the read operation is performed from the flash memory into its internal data register. Once the internal read operation is completed, as indicated

by the R/B# (BUSY) signal, the data register can be read over the external data bus. The various phases are explained as follows.

- **Command Phase:** With CS# asserted, CLE=1, and ALE=0, the command byte (00H) is placed on the I/O pins. This is a command write operation. On the rising edge of WE#, the “read mode 1” command is latched into the command register.
- **Address Phase:** With CS# asserted, ALE=1, and CLE =0, the column and row address is latched into the address register. This operation is usually a series of writes. The byte-wide addresses are latched on the rising edge of every WR# pulse. The first address byte is the column address, and the subsequent bytes are the row addresses. The five least-significant bits of first row address indicate the page number within a block. The three most-significant bits of the first row address and the rest of the higher row address bits determine the block.
- **Data Transfer Phase:** With CS# asserted, CLE=0, and ALE=0, the device goes into a busy state to transfer data from memory into the on-chip data register. This is indicated by the BUSY (R/B#) signal. R/B# goes to logic low and remains at logic low until the data-transfer phase is completed. Once the R/B# goes back to logic level high, the data can be read from data register.
- **Read-Out Phase:** This is indicated by R/B# going to logic level high after the busy period. With CS# asserted, ALE=0, and CLE=0, the data can be read over I/O with a series of RD# pulses.

Figure 3 shows the timing diagram for a read operation.

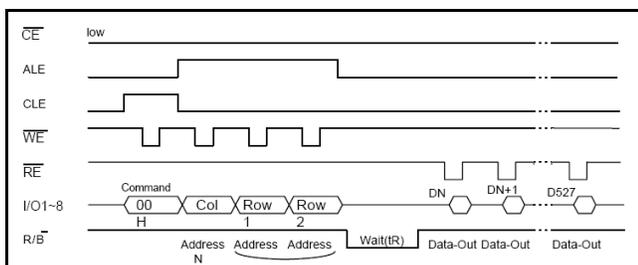


Figure 3. Read Operation Timing

Program Operation

This is a memory write operation. The data in the on-chip data register is written into memory. Similar to a read operation, the device asserts R/B# to logic level low, indicating its busy status. The various phases of this operation are explained below.

- **Command Phase:** With CS# asserted, CLE=1, and ALE=0, the command byte (0x80) is placed on the I/O pins. This is a command write operation. On the rising edge of WE#, the “serial data input” command is latched into the command register.
- **Address Phase:** This is similar to the address phase (See [READ Operation](#) on page 1). With CS# asserted, ALE=1, and CLE=0, the column and row addresses are latched into the address register.
- **Data Input Phase:** With CS# asserted, ALE=0, and CLE=0, the data is written into the data register. The data is driven on the I/O lines, and every byte is latched on the rising edge of WR#.
- **Program Phase:** With CS# asserted, ALE=0, and CLE=1, the auto program command is written into the flash’s command register. The device then goes to the BUSY state, indicated by R/B# going to a logic level low state.
- **Timeout Check Phase:** The status is checked after the completion of the program phase to determine whether the programming was successful.

Figure 4 shows the timing diagram for a program operation.

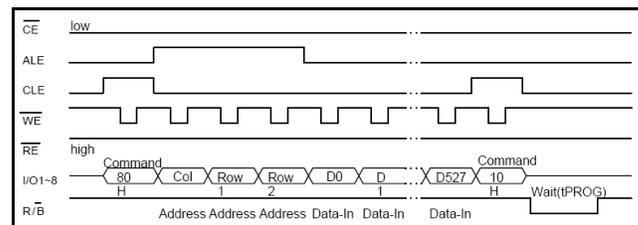


Figure 4. Program Operation Timing

Block Erase Operation

In a block erase operation, a group of consecutive pages (typically 32) is erased in a single operation. Although programming turns bits from “1” to “0”, block erasure is necessary to turn bits from “0” back to “1”.

- **Command Phase:** With CS# asserted, ALE=0, and CLE=1, command byte 0x60 (auto block erase) is written to the flash’s command register. This is done by placing the command byte (0x60) on the I/O lines. The rising edge of WR# latches this value into the command register.
- **Address Phase:** Since this is a block erase, column and page addresses are not required. With CS# asserted, ALE=1, and CLE=0, the block address is written to the address register with series of writes into the flash.
- **Erase Phase:** With CS# asserted, ALE=0, and CLE=1, the auto block erase confirm command (0xD0) is written to the command register. The device then goes to a BUSY state to complete the block erase operation.
- **Timeout Check Phase:** The status is checked after the completion of the program phase to determine whether the erasure was successful.

Figure 5 shows the timing diagram for a block erase operation.

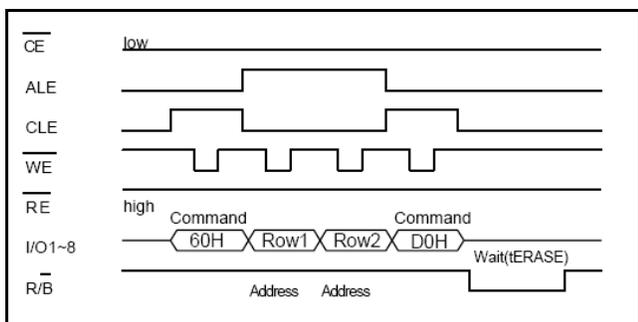


Figure 5. Block Erase Operation Timing

Interface with ADSP-21161 Processors

Since the ADSP-21161 processor does not have a dedicated on-chip NAND flash controller, a software driver is required to control and use the NAND flash. Two approaches are described for this interface. Standard NAND flash memories require that the chip enable remains asserted during the read busy period. Thus, using processor’s FLAG pin to drive the flash’s chip enable makes the driver code compatible with standard NAND flash and CEDC (chip enable don’t care) NAND flash memories. Approach A uses the processor’s dedicated FLAG pins to drive CS#, ALE, and CLE on the flash device. This approach relies on the delays mentioned in the flash device’s data sheet for various operations and does not probe R/B# signal driven by the flash device.

Approach B utilizes address lines “A0” and “A1” to drive CLE and ALE, respectively. Approach B also uses the flash’s R/B# signal to determine the memory’s BUSY state. From a system standpoint, Approach B is a more optimized and reliable method, but it engages one of the processor’s interrupt (IRQ) signals.

Approach A

Refer to Figure 6 for details on the connections between the ADSP-21161 processor and the flash memory. This approach relies completely on the delays mentioned for various tasks in the

flash device data sheet. To be on the safe side, the delay provided in the code should be more than necessary for a given task.

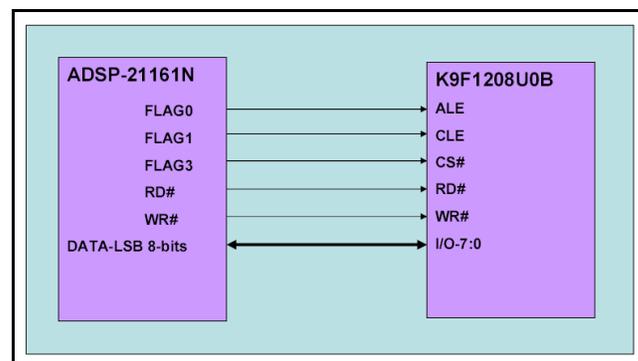


Figure 6. ADSP-21161 and Flash Memory Interface – Approach A

Figure 7 shows the flowchart for the block erase code implemented in Approach A.

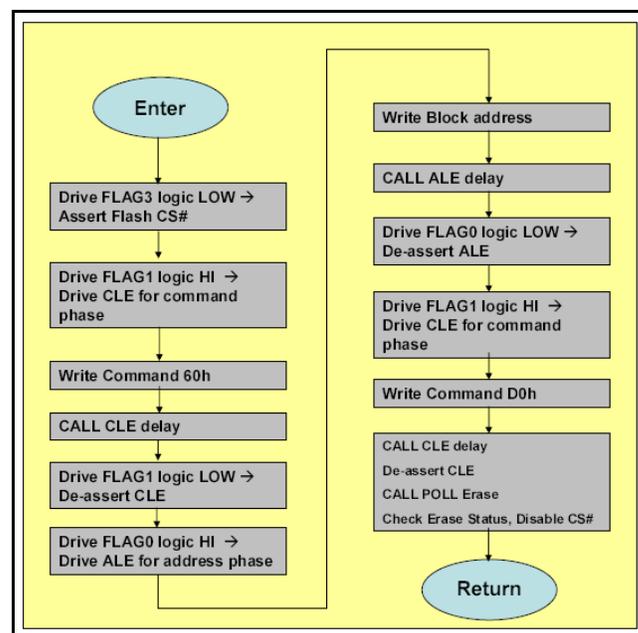


Figure 7. Flowchart of Block Erase Function – Approach A

Figure 8 shows the flowchart for the block program code implemented in Approach A.

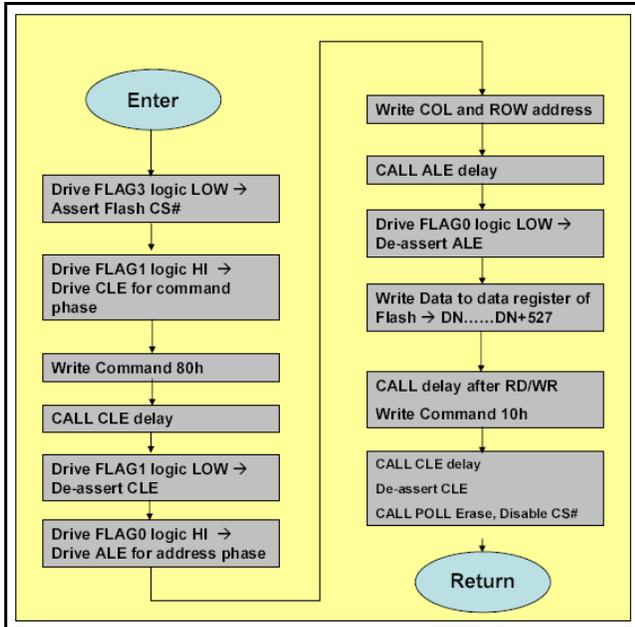


Figure 8. Flowchart of Block Program Function – Approach A

Figure 9 shows the flowchart for the block read code implemented in Approach A.

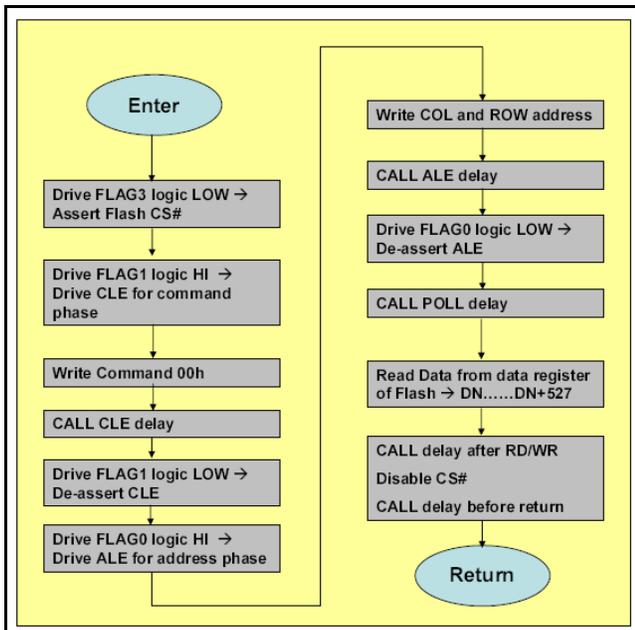


Figure 9. Flowchart of Block Read Function – Approach A

Approach B

Refer to Figure 10 for details on the connections between an ADSP-21161 processor and a flash memory. R/B# (BUSY) from the flash is an active low signal; hence, it is inverted before being connected to the processor's external interrupt. Being an open drain output signal, R/B# requires an external pull-up resistor. Note that the processor's external interrupt must be configured to be edge sensitive.

It is also important to use the hold cycle for address and data during the processor's external memory accesses. This ensures that the CLE and ALE driven by A0 and A1, respectively, are still active for a cycle of external port after the RD# or WR# strobes are inactivated by the processor. This can be ensured by configuring two or more wait states. The code configures seven wait states.

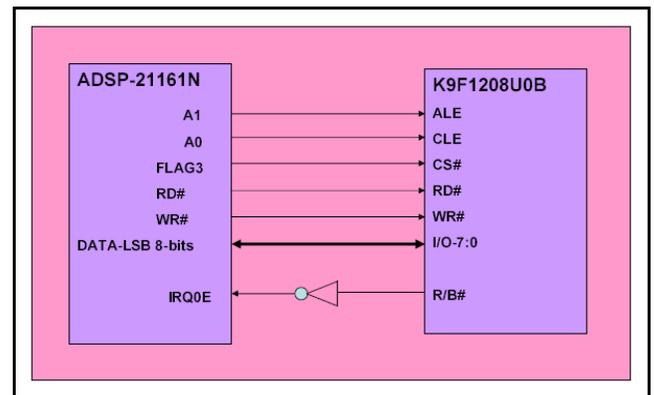


Figure 10. ADSP-21161 Processor and Flash Memory Interface – Approach B

Figure 11 shows the flowchart for the block erase code implemented in Approach B.

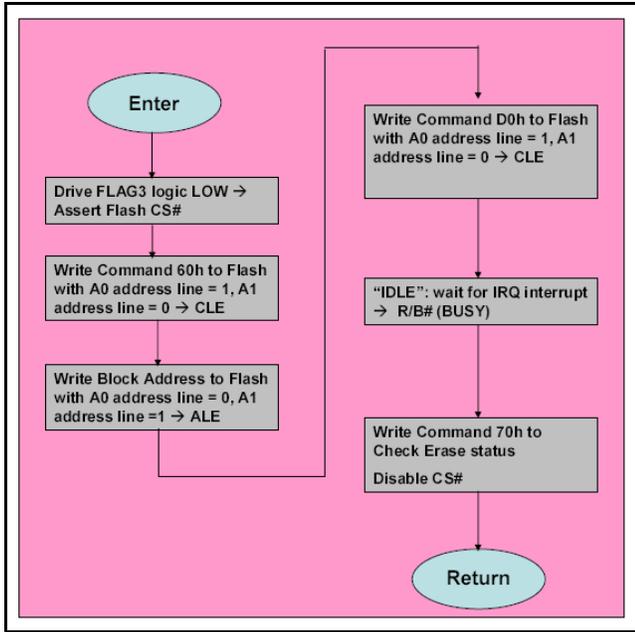


Figure 11. Flowchart of Block Erase Function – Approach B

Figure 12 shows the flowchart for the block program code implemented in Approach B.

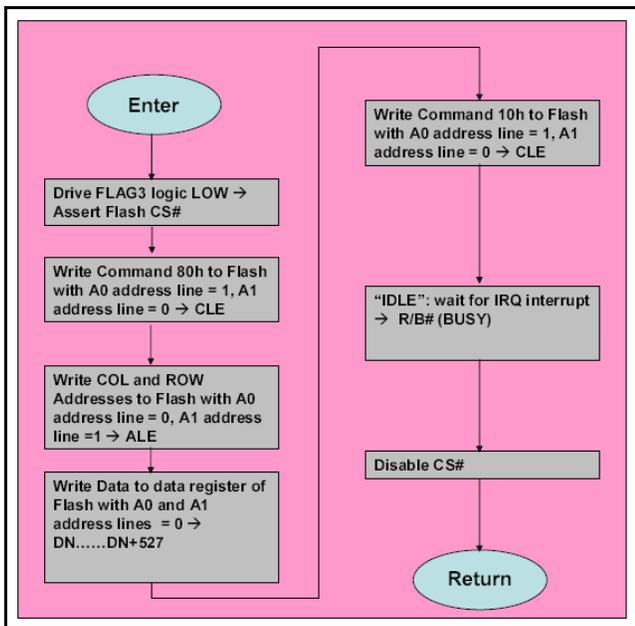


Figure 12. Flowchart of Block Program Function – Approach B

Figure 13 shows the flowchart for the block read code implemented in Approach B.

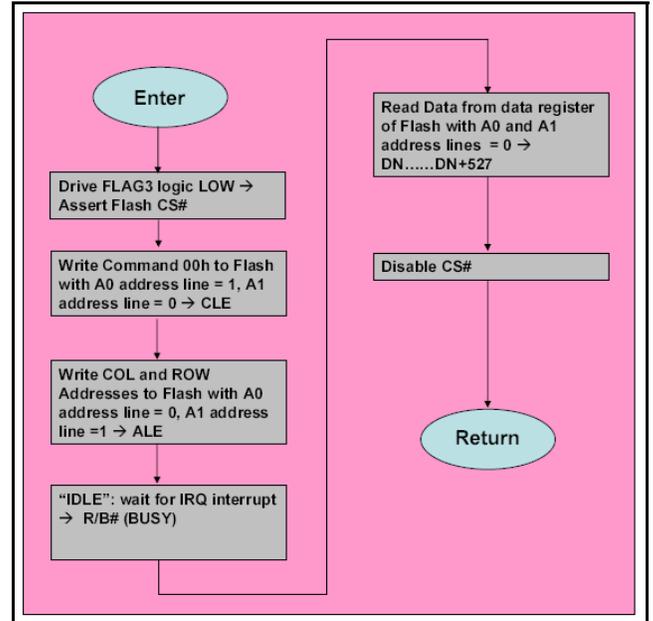


Figure 13. Flowchart of Block Read Function – Approach B

The DMA mode of data transfer can also be used to perform read data and write data operations to the flash memory. In such a case, ensure that no other core or DMA accesses happen to any of the external memory banks. Any such accesses cause spurious RD# and/or WR# pulses to the flash memory, which may cause unexpected results.

Screenshots representing timing diagrams for this interface are shown in the following figures.



Write Command -1



Erase Command -1



Write Command -2



Read Command -2



Read Command -1



Write Command -3

Figure 14. Timing Plots

Conclusion

The interface between an ADSP-21161 processor and a NAND flash can be achieved seamlessly. This interface can be implemented using multiple approaches, which provide system designers with even more flexibility.

Appendix

Code is supplied in the .ZIP file associated with this EE-Note.

References

- [1] *ADSP-21161N Bring-Up-Board schematics*. Analog Devices, Inc.
- [2] *Preliminary data sheet of K9F1208U0B 64M X 8-bit NAND Flash memory*, Samsung Electronics.
- [3] *NAND Flash Applications Design Guide*, Revision 1.0 April 2003, System Solutions from Toshiba America Electronic Components, Inc.
- [4] *ADSP-21161 Hardware Reference Manual*, Third Edition, May 2002. Analog Devices, Inc.

Document History

Revision	Description
<i>Rev 1 – November 17, 2005 by Aseem Vasudev Prabhugaonkar</i>	Initial Release