# 5 ADSP-21161N LOADER

## Overview

The loader (`elfloader.exe`) generates boot-loadable file for ADSP-21161N DSPs by processing executable files. The loader's output is a boot-loadable file with the file name extension `.LDR`. To make a loadable file, the loader processes data from a boot-kernel file (`.DXE`) and one or more other executable files (`.DXE`).

You can load the results from the loader into the VisualDSP++ debugger for simulation. Once you have fully debugged your program, use the loader to generate a set of boot-loadable files for your target system.

This chapter contains the following information on the loader:

- "Loader Command-Line Reference" on page 5-2

  Contains reference information on the loader commands and operations.

- "Loader Guide" on page 5-10

- Contains information on booting and kernel use."Loader Glossary" on page 5-35

  Contains a glossary of loader-related terms.

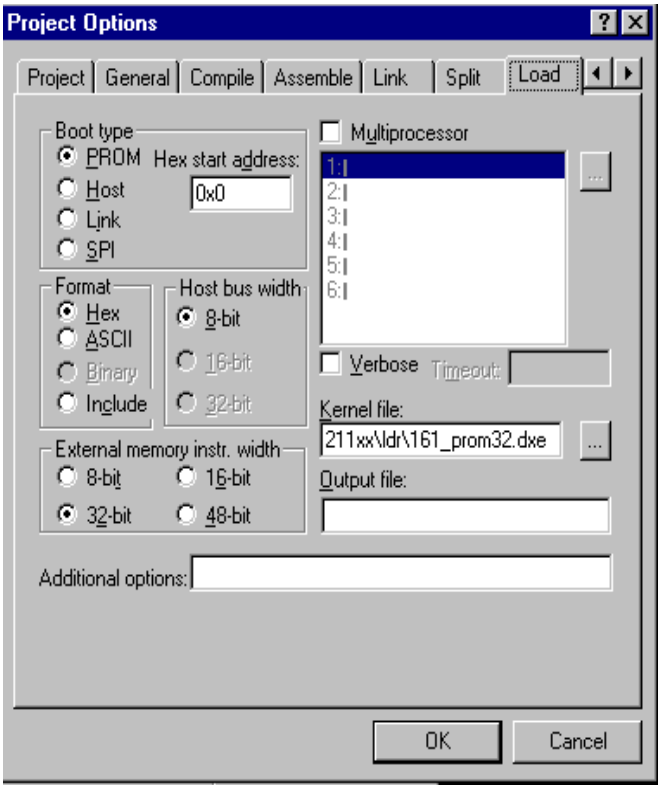
The examples in this chapter cover the ADSP-21161 DSP loader ONLY. For information on the ADSP-21060, ADSP-21061, ADSP-21062, ADSP-21065L and ADSP-21160 DSPs, see Chapter 4.

# Loader Command-Line Reference

This section provides reference information on the loader command line and settings. Refer to Table 5-2 on page 5-6 for switch descriptions.

## Setting Loader Options

When developing a DSP project, you may find it useful to modify the loader's default option settings in the VisualDSP++ environment. This is done via the Load tab on the Project Options dialog box selected from the Project menu. For example,



Use the Additional Options field in the Load tab to enter the appropriate file names and functional options to control the loader operation. The loader has command-line switches that correspond to the Load dialog's options. For more information, see the *VisualDSP++ User's Guide for ADSP-21xxx DSPs*.

## Running the Loader

Use the following syntax for a generic single-processor loader command line:

```
elfloader sourcefile -switch [-switch …] -proc ADSP-21161
```

where:

- *sourcefile* — This is the name of the executable file (`.DXE`) to be processed for a single-processor boot-loadable file. A file name can include the drive, directory, file name and file extension; enclose long file names within straight-quotes, `"long file name"`.
- *-switch* — This is the name of the switch to be processed. The loader has many optional switches. These select the operations and modes for the loader.
- `-proc` *ADSP-21161* — This is the name of a target processor: ADSP-21161.

You must specify your larger as `ADSP-21161` using either the `-Dprocessor` or `-proc processor` switch. Refer to Table 5-2 on page 5-6 for switch descriptions.

The loader command line is case sensitive.

In the ADSP-21161N DSPs, for a single processor system, there is only one input `.dxe` file without any prefix and suffix to the input file name, for example:

```
elfloader -proc ADSP-21161 -bSPI Input.dxe <switches>
```

However, a distinct processor ID number for each input file must be present in the command line for supporting a multiprocessor system. A processor ID is given through the switch "`-id#exe=`", where # is the ID number from 1 to 6.

For example:

```
elfloader -proc ADSP-21161 -bprom -id1exe=Input1.dxe
-id2exe=Input2.dxe <switch>
```

The loader is processing the input file `Input1.dxe` for the processor with the ID of 1 and the input file `Input2.dxe` for the processor with the ID of 2, and so on.

Some loader switches take a file name as an optional parameter. File searches are important in the loader's process. The loader supports relative and absolute directory names, default directories, and user-selected directories for file search paths. File searches occur as follows:

1. *`Specified path`* — If you include relative or absolute path information in a file name, the loader only searches in that location for the file.
2. *`Default directory`* — If you do not include path information in the file name, the loader searches for the file in the current working directory.

When you provide an input or output file name as a command-line parameter, use the following guidelines:

- Enclose long file names within straight-quotes, "`long file name`".
- Append the appropriate file name extension to each file.

The loader follows the conventions for file name extensions that appear in Table 5-1.

Table 5-1. File Name Extension Conventions

| Extension | File Description |
|---|---|
| .dxe[1] | Executable files and boot-kernel files |
| .ldr | Loader output file |

[1] The loader recognizes shared memory (.SM) and overly memory (.OVL) files, but does not expect these files on the command line. Place .SM or .OVL files in the same directory as the .DXE that refers to them, and the loader finds the .SM and .OVL files when processing the .LDR file.

## ADSP-21161 Loader Command-Line Switches

A summary and descriptions of the loader's command-line switches appears in Table 5-2.

Table 5-2. Loader Command-Line Switch Description

| Switch | Description |
|---|---|
| *filename* | The *filename* parameter without a switch selects an executable file for single-processor boot-loadable files. For multiprocessor boot-loadable files, use the -id#exe=file switch. |
| -b *type* | The -b (boot-type) switch directs the loader to prepare a boot-loadable file for the *type* booting-mode. Valid *type* selections are prom, host, link and SPI. If the -b switch does not appear on the command line, the default setting is -bprom.<br><br>Note that the boot-type that you select with the -b switch must correspond to the boot-kernel that you select with the -l switch and the file format that you select with the -f switch. |
| -d *processor* | The -d (processor) switch directs the loader to specify a defined processor: ADSP-21161. See also the -proc switch.<br><br>The processor parameter allows the loader to select a kernel from the default kernel directory based on the processor specified. Alternatively, you can use the -l option to specify a particular loader kernel. |
| -e *filenames* | The -e (except shared memory) switch directs the loader to omit the shared memory (.SM) file from the output loader file. You can use this switch multiple times to omit multiple shared memory files. |

Table 5-2. Loader Command-Line Switch Description (Cont'd)

| Switch | Description |
|---|---|
| -ExternalMemoryWidth # | The -ExternalMemoryWidth # switch is used to specify the external memory word packing width. The valid values are 48, 32, 16 and 8 independent of the boot mode. |
| -f *format* | The -f (boot file format) switch directs the loader to prepare a boot-loadable file in the specified *format*. The available *format* selections are hex (Intel Hex-32), ASCII, binary, and include (C-style include), but valid formats depend on the -b switch's boot-type selection. The boot-type formats are as follows:<br>For PROM — hex, ASCII or INCLUDE<br>For HOST or LINK — ASCII, binary or INCLUDE<br>For SPI — ASCII<br>If the -f switch does not appear on the command line, the default format for boot-types are: hex for PROM and ASCII for HOST, LINK or SPI. |
| -h | The -h (command line help) switch directs the loader to output the list of command line switches to standard output and exit. |
| -HostWidth # | The -HostWidth # switch is used to specify the word width in a loader output file. The valid values are generally 32, 16 and 8 depending on the boot mode.<br>The default values for HostWidth are 8 for PROM and HOST, 16 for LINK, and 32 for SPI. The valid word widths are as follows:<br>BOOT TYPE — BOOT FORMAT — HOSTWIDTH #<br>PROM — hex, ASCII — 8<br>INCLUDE — 8, 16<br>HOST — ASCII, binary — 8, 16<br>INCLUDE — 16<br>LINK — ASCII, binary — 16<br>INCLUDE — 16<br>SPI — hex, ASCII — 8, 16, 32 |

Table 5-2. Loader Command-Line Switch Description (Cont'd)

| Switch | Description |
|---|---|
| `-id#exe=`*`file`* | The `-id#exe` (multiprocessor ID) switch directs the loader to use # processor ID for the corresponding executable *`file`* when producing a boot-loadable file for EPROM booting a multiprocessor system. This switch is only for producing a boot-loadable file that boots multiple DSPs in turn from a single EPROM.<br><br>Valid values are `1`, `2`, `3`, `4`, `5` and `6`. Do not use this switch for single-processor systems; enter the executable *`filename`* as a parameter without a switch to select an executable file for single-processor systems. |
| `-l` *`kernel`* | The `-l` (boot-kernel) switch directs the loader to use the *`kernel`* file for the boot-loading routine in the output boot-loadable file. Note that the boot-kernel that you select with the `-l` switch must correspond to the boot-type that you select with the `-b` switch and the external memory packing width (EMPW). If the `-l` switch does not appear on the command line, the loader searches for a default boot-kernel file.<br><br>Based on the `-b` switch setting and the specified external memory packing width, the loader searches for the default boot-kernel for `ADSP-21161` in your DSP tools installation directory `...\211xx\ldr`, as follows:<br><br>BOOT TYPE — DEFAULT KERNEL FILE — EMPW<br>`PROM` `161_prom8.dxe` 8<br>`161_prom16.dxe` 16<br>`161_prom32.dxe` 32<br>`161_prom48.dxe` 48<br>`HOST` `161_host8.dxe` 8<br>`161_host16.dxe` 16<br>`161_host32.dxe` 32<br>`161_host48.dxe` 48 |

Table 5-2. Loader Command-Line Switch Description (Cont'd)

| Switch | Description |
|---|---|
| | LINK 161_link8.dxe 8<br>161_link16.dxe 16<br>161_link32.dxe 32<br>161_link48.dxe 48<br><br>SPI 161_SPI8.dxe 8<br>161_SPI16.dxe 16<br>161_SPI32.dxe 32<br>161_SPI48.dxe 48 |
| -o *filename* | The -o (output file) switch directs the loader to use the *filename* for the loader's output file. If not specified, the default name for the loader's output is "*source-file*.ldr." |
| -p *address* | The -p (prom start address) switch directs the loader to start the boot-loadable file at the specified *address* in the EPROM. If the -p switch does not appear on the command line, the loader starts the EPROM file at address 0x0 in the EPROM. |
| -proc *ProcessorID* | The -proc (processor) switch directs the loader to select a target processor: ADSP-21161. |
| -t *timeout* | The -t (timeout cycles) switch directs the loader to set the *timeout* number of cycles as a limit on the number of cycles that the DSP spends initializing external memory. Valid *timeout* values range from 3 to 32765 cycles; 32765 is the default *timeout* value. The *timeout* value is directly related to the number of cycles that the DSP locks the bus for boot-loading, instructing the DSP to lock the bus for no more than 2x *timeout* number of cycles. If you are working with a fast host that cannot tolerate being locked out of the bus, use a relatively small value *timeout*. |
| -v | The -v (verbose loader messages) switch directs the loader to output status information as the loader processes your files. |

# Loader Guide

The loader converts executable files into boot-loadable files for an ADSP-21161N DSP. Programs can be automatically downloaded to the internal memory of a DSP after power-up or after a software reset. This process is called booting.

Loader options let you control how the loader processes your executable files, letting you select features such as loader kernel, boot type, and file format among others. These options appear on the loader command line (or on the Load tab, available from the Project Options window in the VisualDSP++ environment).

See "ADSP-21161 Loader Command-Line Switches" on page 5-6 for more information.

All software developers using the loader should be familiar with the following operations:


- "Working With Different Boot Types" on page 5-11
- "ADSP-21161N Boot-Kernels" on page 5-14
- "EPROM Booting" on page 5-19
- "Link Port Booting" on page 5-27
- "SPI Booting" on page 5-30
- "Multiprocessor Booting" on page 5-32

## Working With Different Boot Types

Before selecting options for the loader's operation, you should read about how the loader's features apply to the boot-type that you are using. This section talks about the boot-types for ADSP-21xxx DSPs and relates loader features that support these boot-types.

The ADSP-21xxx DSPs have a special hardware feature that boot-loads a small, 256-instruction program into the DSP's internal memory at chip reset. This program can come from an external PROM, a host processor, SPI or another ADSP-21xxx DSP. You select the boot-type using three of the DSP's external pins as shown in Table 5-3 and Table 5-4. After the small program (the boot-kernel) loads, this program loads the rest of your program and data into the DSP. The combination of the boot-kernel and your program make up a boot-loadable file.

The ADSP-21161N processor needs a loader and a number of kernels to support the booting process. The basic boot modes to support the ADSP-21161N processor are PROM, HOST, SPI HOST and LINK. The basic loader file formats are Intel Hex, ASCII and Binary.

### Output File Formats

Each of boot modes supports several output file formats and word packing formats. These formats are described in the folfowing sections.

#### PROM Output File Format

The output file format to support the PROM boot mode are Intel Hex, ASCII and INCLUDE. The default is Intel Hex. The `HostWidth` setting is 8 (or 8-bit). For the INCLUDE format output, the `HostWidth` setting can be 16 (16 bit).

To support external memory word packing, the loader needs to re-calculate the external memory address according to the packing width selected. The default external memory packing width is 32 bits. The other packing

widths available are 8, 16 and 48 bits. To select packing width, use the `-ExternalMemoryWidth` command-line switch.

### Host Output File Format

There are three output file formats to support the HOST boot mode: ASCII, INCLUDE and Binary, and the default is ASCII. The value of the `HostWidth` setting is either 8 or 16. The default is 8. For the INCLUDE format output, the `HostWidth` setting is 16 (or 16 bit) only.

Similar to the PROM, the loader needs to re-calculate a correct external memory addresses for the external memory instructions to support the external memory word packing. The default external memory packing width is 32 bits. The other packing widths available are 8, 16 and 48 bits. To select packing width, use the `-ExternalMemoryWidth` command-line switch.

### Link Output File Format

There are three output file formats to support the LINK boot mode: ASCII, INCLUDE and Binary, and the default is ASCII. The value of the `HostWidth` setting is 16 (or 16 bit) only.

The loader supports the LINK boot mode in the same way as in the ADSP-21160 processor. In addition to that, the loader needs to re-calculate a correct address for external memory instructions similar to the PROM and HOST boot modes.

### SPI HOST Output File Format

In order to support the different word lengths in an output file and external memory packing modes, the loader uses the following command-line switches (or VisualDSP++ IDDE settings):

- `-HostWidth #` to specify the word width in a loader output file. The valid values are generally 32, 16 and 8 (bits) depending on the boot mode. The default value is 32.

- `-ExternalMemoryWidth #` to specify the external memory word packing width. The valid values are generally 48, 32, 16 and 8 (bits). The default value is 32.

## Selecting A Booting Mode

The booting mode is selected using the LBOOT, EBOOT, and $\overline{\text{BMS}}$ pins. Table 5-3 and Table 5-4 show how to select among the booting modes.

Table 5-3. Boot Mode Pins (Descriptions)

| Pin | Type | Description |
|---|---|---|
| EBOOT | I | EPROM Boot — When EBOOT is high, the ADSP-21xxx boot-loads from an 8-bit EPROM through the DSP's external port. When EBOOT is low, the LBOOT and $\overline{\text{BMS}}$ pins determine booting mode. |
| LBOOT | I | Link Boot — When LBOOT is high (and EBOOT is low), the ADSP-21xxx boots from another ADSP-21xxx DSP through the DSP's link port. When LBOOT is low (and EBOOT is low), the ADSP-2106x boots from a host processor through the DSP's external port. |
| $\overline{\text{BMS}}$ | I/O/T[1] | Boot Memory Select — When boot-loading from an EPROM (EBOOT=1 and LBOOT=0), this pin is an *output* and serves as the chip select for the EPROM. In a multiprocessor system, $\overline{\text{BMS}}$ is output by the bus master. When host-booting, link-booting or SPI-booting, (EBOOT=0 and LBOOT=0 or =1), $\overline{\text{BMS}}$ is an *input* and must be high for host-, link-, or SPI-booting. |

[1] Three-statable in EPROM boot mode (when $\overline{\text{BMS}}$ is an output).

Table 5-4. Boot Mode Pins (States)

| EBOOT | LBOOT | $\overline{\text{BMS}}$ | Booting Mode |
|---|---|---|---|
| 1 | 0 | output | EPROM Boot (connect $\overline{\text{BMS}}$ to EPROM chip select) |
| 0 | 0 | 1 (input) | Host Boot |

Table 5-4. Boot Mode Pins (States) (Cont'd)

| EBOOT | LBOOT | $\overline{\text{BMS}}$ | Booting Mode |
|---|---|---|---|
| 0 | 1 | 1 (input) | Link Boot |
| 0 | 1 | 0 (input) | Serial Port Boot (SPI) |
| 0 | 0 | 0 (input) | No Booting (processor executes from the external memory) |

## ADSP-21161N Boot-Kernels

For all boot-kernels, the boot-loading process begins by downloading the boot-kernel into the DSP's memory. Boot kernels are loaded at reset into program segment defined in the `161_ldr.ldf` linker description file. This file is stored in your installation directory ...`\211xx\ldr` directory.

The boot-kernel sets up the DSP as necessary and loads boot-data. After the boot-kernel finishes initializing the rest of the system, the boot-kernel loads boot-data over itself with a final DMA transfer. The following are default boot kernels that come with VisualDSP++ software for the ADSP-21161N DSP.

| PROM Kernel | Function |
|---|---|
| `161_prom8.asm` | Initializing 8-bit packed instructions in external memory |
| `161_prom16.asm` | Initializing 16-bit packed instructions in external memory |
| `161_prom32.asm` | Initializing 32-bit packed instructions in external memory |
| `161_prom48.asm` | Initializing 48-bit non-packed instructions in external memory |

| LINK BOOT KERNELS | HOST BOOT KERNELS | SPI BOOT KERNELS |
|---|---|---|
| 161_link8.asm | 161_host8.asm | 161_spi8.asm |
| 161_link16.asm | 161_host16.asm | 161_spi16.asm |
| 161_link32.asm | 161_host32.asm | 161_spi32.asm |
| 161_link48.asm | 161_host48.asm | 161_spi48.asm |

The default `ExternalMemoryWidth` for all the boot modes is 32 (bits). If you do not specify any kernel file, the loader picks up a correct default kernel and places it in the loader output file according to the specified value of the `ExternalMemoryWidth`. Otherwise, it will use the user kernel file as specified by the user.

## Boot-Kernel Changes and Software Issues

For some systems, customization of the boot-kernel is required. Also, the operation of other tools (notably the C/C++ compiler) is influenced by whether or not you plan to use the loader. This section describes these types of loader-usage issues.

The loader reads your DSP executable file, identifying the DSP type and the system's memory configuration. When producing a boot-loadable file, the loader uses the information in the executable file to initialize memory correctly. The loader cannot, however, determine how the DSP's `SYSCON`, `WAIT` and `SDRAM` registers should be configured for external memory loading in the system.

If you must modify the boot-kernel by inserting the correct values for your system, you will need to rebuild the kernel before generating the boot-loadable file. The loader kernel contains default values for `SYSCON`. The `WAIT`, `SDCTL` and `SDRAM` initialization code may be found in the comments for the kernel file.

After modifying a copy of the applicable boot-kernel sourcefile (for example, `161_link.asm`, `161_host.asm`, or `161_prom.asm`, etc., refer to Table 5-2 on page 5-6 for the list of kernel files) to apply the appropriate initializations of the `SYSCON` and `WAIT` registers, use the loader-kernel (`-l`) command-line switch.

When using the VisualDSP++ IDDE, select your modified kernel executable in the Load option dialog box) to apply this customized boot-kernel.

If you have to modify the boot-kernel for EPROM-, host-, SPI-, or link-booting modes, make sure that segment named `seg_ldr` is defined in your linker description file. You can find source for this segment in the `.ldf` file, that is located in `...\211xx\ldr\)` directory.

Because the loader uses the address for the first location of the reset vector during the boot-load process, your code should avoid using this address anytime. When using any of the DSP's power-up booting-modes, it is even more important that this address does not contain a critical instruction, because that address is not executed during the booting sequence. Place a `NOP` or `IDLE` instruction at this location. The loader generates a warning if the vector address `0x0004 0004` does not contain `NOP` or `IDLE`.

The load kernel project can be rebuild from the VisualDSP++ IDDE. If it is not used, the following command lines demonstrate how to rebuild various default boot-kernels for the ADSP-21161N DSPs:

- The default boot-kernel for EPROM booting is `161_prom32.asm`. After copying the default file to `my_prom32.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

  ```
  easm21k -adsp21161 my_prom32.asm
  ```

  - or -

  ```
  easm21k -proc ADSP21161 my_prom32.asm
  linker -T 161_ldr.ldf my_prom32.doj
  ```

- The default boot kernel for host booting is `161_host32.asm`. After copying the default file to `my_host32.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

```
easm21k -adsp21161 my_host32.asm
```

- or -

```
easm21k -proc ADSP21161 my_host32.asm
linker -T 161_ldr.ldf my_host32.doj
```

- The default boot kernel for link booting is `161_link32.asm`. After copying the default file to `my_link32.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

```
easm21k -adsp21161 my_link32.asm
```

- or -

```
easm21k -proc ADSP21161 my_link32.asm
linker -T 161_ldr.ldf my_link32.doj
```

- The default boot kernel for SPI booting is `161_SPI32.asm`. After copying the default file to `my_link32.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

```
easm21k -adsp21161 my_SPI32.asm
```

- or -

```
easm21k -proc ADSP21161 my_SPI32.asm
linker -T 161_ldr.ldf my_SPI32.doj
```

## Loader File Header Words

The loader inserts header words at the beginning of data blocks in the loader file. These header words are used by the loader kernel to properly place data and instruction blocks in memory. The header format for PROM, host, and link boot loader files is as follows:

```
0x00000000DDDD
0xAAAAAAAALLLL
```

where: `D` = data type tag, `A` = address, and `L` = length

A data block header for the SPI boot type in a loader file consists of three 32-bit words.

For single processor systems, the data header has three 32-bit words in SPI boot type, as follows:

| | |
|---|---|
| `0x0000LLL` | • First word: Data length or data word count of the data block |
| `0xAAAAAAAA` | • Second word: Data block start address |
| `0x000000DD` | • Third word: Tag of data type. |

The data structures and packing schemes in each data block for multiprocessor systems are the same as the ones for single processor systems currently used.

The tag is used by the loader kernel to determine what type of data or instruction is being loaded. Table 5-5 is a list of tags for the ADSP-21161N processor.

Table 5-5. ADSP-21161N Tag Information

| TAG NUMBER | SECTION TYPE | TAG NUMBER | SECTION TYPE |
|---|---|---|---|
| 0x0000 | final init | 0x000E | init pm48 |
| 0x0001 | zero dm16 | 0x000F | zero dm64 |
| 0x0002 | zero dm32 | 0x0010 | init dm64 |
| 0x0003 | zero dm40 | 0x0011 | zero pm64 |
| 0x0004 | init dm16 | 0x0012 | init pm64 |
| 0x0005 | init dm32 | 0x0013 | init pm8 ext |
| 0x0006 | init dm40 | 0x0014 | init pm16 ext |
| 0x0007 | zero pm16 | 0x0015 | init pm32 ext |
| 0x0008 | zero pm32 | 0x0016 | init pm48 ext |
| 0x0009 | zero pm40 | 0x0017 | zero pm8 ext |
| 0x000A | zero pm48 | 0x0018 | zero pm16 ext |
| 0x000B | init pm16 | 0x0019 | zero pm32 ext |
| 0x000C | init pm32 | 0x001A | zero pm48 ext |
| 0x000D | init pm40 | | |

## EPROM Booting

The DSP can boot from an EPROM through the external port. EPROM booting through the external port is selected when the EBOOT input is high and the LBOOT input is low. These settings cause the $\overline{\text{BMS}}$ pin to become an output, serving as chip select for the EPROM.

The DMAC10 control register is initialized for booting packing boot data into 48-bit instructions. The EPROM boot mode uses Channel 10 of the I/O processor's DMA controller to transfer the instructions to internal memory. For EPROM booting, the DSP reads data from an 8-bit external EPROM.

After the boot process loads 256 words into memory locations 0x40000 through 0x400FF, the DSP begins executing instructions. Because most DSP programs require more than 256 words of instructions and initialization data, the 256 words typically serve as a loading routine for the

application. Analog Devices supplies loading routines (loader kernels) that can load entire programs. These routines come with the development tools. See “ADSP-21161N Boot-Kernels” on page 5-14 for more information.

Refer to “*ADSP-21161 SHARC DSP Hardware Reference*” for more detailed information on DMA and system configurations.

It is important to note that DMA channel differences between the ADSP-21161 DSP and previous SHARC DSPs (ADSP-2106x) introduce some booting differences. Even with these differences, the ADSP-21161 DSP supports the same boot capability and configuration as the ADSP-2106x DSPs. The DMAC register default values differ because the ADSP-21161 DSP has additional parameters and different DMA channel assignments. The EPROM boot mode uses EPB0, DMA channel 10. Like the ADSP-2106x DSPs, the ADSP-21161 DSP boots from DATA23-16.

The DSP determines the booting mode at reset from the EBOOT, LBOOT, and $\overline{\text{BMS}}$ pin inputs. When EBOOT=1 and LBOOT=0, the DSP boots from an EPROM through the external port and uses $\overline{\text{BMS}}$ as the memory select output. For a list showing how to select different boot modes, see the Boot Memory Select pin descriptions in Table 5-3 and Table 5-4 on page 5-13.

When using any of the power-up booting modes, address 0x0004 0004 should not contain a valid instruction since it is not executed during the booting sequence. A NOP or IDLE instruction should be placed at this location.

In EPROM booting through the external port, an 8-bit wide boot EPROM must be connected to data bus pins 23-16 (DATA23-16). The lowest address pins of the DSP should be connected to the EPROM’s address lines. The EPROM’s chip select should be connected to $\overline{\text{BMS}}$ and its output enable should be connected to $\overline{\text{RD}}$.

In a multiprocessor system, the $\overline{\text{BMS}}$ output is only driven by the ADSP-21161N bus master. This allows wire-ORing of multiple $\overline{\text{BMS}}$ signals for a single common boot EPROM.

Systems can boot any number of ADSP-21161N DSPs from a single EPROM using the same code for each processor or differing code for each.

During reset, the DSP's ACK line is internally pulled high with a 20 kΩ equivalent resistor and is held high with an internal keeper latch. It is not necessary to use an external pullup resistor on the ACK line during booting or at any other time.

When the EPROM boot mode is configured, the external port DMA Channel 10 (DMAC10) becomes active following reset; it is initialized to 0x04A1. This enables the external port DMA and selects DTYPE for instruction words. The packing mode bits (PMODE) are ignored, BSO is set in SYSCON, and 8- to 48-bit packing is forced with least-significant-word first.

The RBWS and RBAM fields of the WAIT register are initialized to perform asynchronous access and to generate seven wait states (eight cycles total) for the EPROM access in external memory space. Note that wait states defined for boot memory are applied to $\overline{\text{BMS}}$-asserted accesses.

Table 5-6 shows how the DMA channel 10 parameter registers are initialized at reset for EPROM booting. The count register (CEP0) is initialized to 0x0100 for transferring 256 words to internal memory. The external count register (ECEP0), which is used when external addresses are generated by the DMA controller, is initialized to 0x0600 (i.e., 0x0100 words with six bytes per word). The DMAC10 control register is initialized to 0000 0160.

The default value sets up external port transfers as follows:

- DEN = 1, external port enabled
- MSWF = 0, LSB first

- PMODE = 100, 8- to 48-bit packing
- DTYPE = 1, three-column data

Table 5-6. DMA Channel 10 Parameter Register Initialization for EPROM Booting

| Parameter Register | Initialization Value |
|---|---|
| IIEP0 | 0x0004 0000 |
| IMEP0 | uninitialized (increment by 1 is automatic) |
| CEP0 | 0x0100 (256 instruction words) |
| CPEP0 | uninitialized |
| GPEP0 | uninitialized |
| EIEP0 | 0x0080 0000 |
| EMEP0 | uninitialized (increment by 1 is automatic) |
| ECEP0 | 0x0600 (256 words x 6 bytes/word) |

At system start-up, when the DSP's $\overline{\text{RESET}}$ input goes inactive, the following sequence occurs:

1. The DSP goes into an idle state, identical to that caused by the IDLE instruction. The program counter (PC) is set to address 0x0004 0004.
2. The DMA parameter registers for channel 10 are initialized as shown in Table 5-6.
3. $\overline{\text{BMS}}$ becomes the boot EPROM chip select.

4. 8-bit Master Mode DMA transfers from EPROM to internal memory begin, on the external port data bus lines 23-16.
5. The external address lines (ADDR23-0) start at 0x0080 0000 and increment after each access.
6. The $\overline{\text{RD}}$ strobe asserts as in a normal memory access with seven wait states (eight cycles).

The DSP's DMA controller reads the 8-bit EPROM words, packs them into 48-bit instruction words, and transfers them to internal memory until 256 words have been loaded. The EPROM is automatically selected by the $\overline{\text{BMS}}$ pin; other memory select pins are disabled.

The DMA external count register (ECEP0) decrements after each EPROM transfer. When ECEP0 reaches zero, the following wake-up sequence occurs:

1. The DMA transfers stop.
2. The External Port DMA Channel 10 interrupt (EP0I) is activated.
3. $\overline{\text{BMS}}$ is deactivated and normal external memory selects are activated.
4. The DSP vectors to the EP0I interrupt vector at 0x0004 0050.

At this point the DSP has completed its booting mode and is executing instructions normally. The first instruction at the EP0I interrupt vector location, address 0x0004 0050, should be an RTI (Return from Interrupt). This process returns execution to the reset routine at location 0x0004 0005 where normal program execution can resume. After reaching this point, a program can write a different service routine at the EP0I vector location 0x0004 0050.

## Host Booting

The DSP can boot from a host processor through the external port. Host booting is selected when the EBOOT and LBOOT inputs are low and $\overline{\text{BMS}}$ is high. Configured for host booting, the DSP enters the slave mode after reset and waits for the host to download the boot program.

The DMAC10 control register is initialized for booting packing boot data into 48-bit instructions. Host boot uses Channel 10 of the I/O processor's DMA controller to transfer the instructions to internal memory. For host booting, the DSP accepts data from a 8-, 16- or 32-bit host microprocessor (or other external device).

After the boot process loads 256 words into memory locations 0x4 0000 through 0x4 00FF, the DSP begins executing instructions. Because most DSP programs require more than 256 words of instructions and initialization data, the 256 words typically serve as a loading routine for the application. Analog Devices supplies loading routines (loader kernels) that can load entire programs. These routines come with the development tools. See "ADSP-21161N Boot-Kernels" on page 5-14 for more information.

Refer to "*ADSP-21161 SHARC DSP Hardware Reference*" for more detailed information on DMA and system configurations.

It is important to note that DMA channel differences between the ADSP-21161 DSP and previous SHARC DSPs (ADSP-2106x) introduce some booting differences. Even with these differences, the ADSP-21161 DSP supports the same boot capability and configuration as the ADSP-2106x DSPs. The DMAC register default values differ because the ADSP-21161 DSP has additional parameters and different DMA channel assignments. The Host boot mode uses EPB0, DMA channel 10. Like the ADSP-2106x DSPs, the ADSP-21161 DSP boots from DATA23-16.

The DSP determines the booting mode at reset from the EBOOT, LBOOT, and $\overline{\text{BMS}}$ pin inputs. When EBOOT=0, LBOOT=0, and $\overline{\text{BMS}}$ =1, the DSP boots from a host through the external port. Refer to Table 5-3 and Table 5-4 on page 5-13 on how to select different boot modes.

When using any of the power-up booting modes, address 0x0004 0004 should not contain a valid instruction since it is not executed during the booting sequence. A NOP or IDLE instruction should be placed at this location.

During reset, the DSP's ACK line is internally pulled high with a 20 kΩ equivalent resistor and is held high with an internal keeper latch. It is not necessary to use an external pullup resistor on the ACK line during booting or at any other time.

When the Host boot mode is configured, the external port DMA Channel 10 (DMAC10) becomes active following reset; it is initialized to 0x04A1. This enables the external port DMA and selects DTYPE for instruction words. The packing mode bits (PMODE) are ignored, BSO is set in SYSCON, and 8- to 48-bit packing is forced with least-significant-word first.

Table 5-6 on page 5-22 shows how the DMA channel 10 parameter registers are initialized at reset for host booting. The count register (CEP0) is initialized to 0x0100 for transferring 256 words to internal memory. The external count register (ECEP0), which is used when external addresses are generated by the DMA controller, is initialized to 0x0600 (i.e., 0x0100 words with six bytes per word). The DMAC10 control register is initialized to 0000 0160.

The default value sets up external port transfers as follows:

- DEN = 1, external port enabled
- MSWF = 0, LSB first
- PMODE = 100, 8- to 48-bit packing
- DTYPE = 1, three-column data

Table 5-7. DMA Channel 10 Parameter Register Initialization for Host Booting

| Parameter Register | Initialization Value |
|---|---|
| IIEP0 | 0x0004 0000 |
| IMEP0 | uninitialized (increment by 1 is automatic) |
| CEP0 | 0x0100 (256 instruction words) |
| CPEP0 | uninitialized |
| GPEP0 | uninitialized |
| EIEP0 | 0x0080 0000 |
| EMEP0 | uninitialized (increment by 1 is automatic) |
| ECEP0 | 0x0600 (256 words x 6 bytes/word) |

At system start-up, when the DSP's $\overline{\text{RESET}}$ input goes inactive, the following sequence occurs:

1. The DSP goes into an idle state, identical to that caused by the IDLE instruction. The program counter (PC) is set to address 0x0004 0004.
2. The DMA parameter registers for channel 10 are initialized as shown in Table 5-6 on page 5-22.
3. The host uses HBR and CS to arbitrate for the bus.

4. The host can write to SYSCON to change boot width from default.
5. The host writes boot information to external port buffer 0.

The DMA external count register (ECEPO) decrements after each transfer. When ECEPO reaches zero, the following wake-up sequence occurs:

1. The DMA transfers stop.
2. The External Port DMA Channel 10 interrupt (EPOI) is activated.
3. The DSP vectors to the EPOI interrupt vector at 0x0004 0050.

At this point the DSP has completed its booting mode and is executing instructions normally. The first instruction at the EPOI interrupt vector location, address 0x0004 0050, should be an RTI (Return from Interrupt). This process returns execution to the reset routine at location 0x0004 0005 where normal program execution can resume. After reaching this point, a program can write a different service routine at the EPOI vector location 0x0004 0050.

## Link Port Booting

Link port booting uses DMA channel 8 of the I/O processor to transfer the instructions to internal memory. In this boot mode, the DSP receives 4-bit wide data in link buffer 0.

After the boot process loads 256 words into memory locations 0x40000 through 0x400FF, the DSP begins executing instructions. Because most DSP programs require more than 256 words of instructions and initialization data, the 256 words typically serve as a loading routine for the application. Analog Devices supplies loading routines (loader kernels) that load an entire program through the selected port. These routines come with the development tools. See “ADSP-21161N Boot-Kernels” on page 5-14 for more information.

Refer to "*ADSP-21161 SHARC DSP Hardware Reference*" for more detailed information on DMA and system configurations.

It is important to note that DMA channel differences between the ADSP-21161 DSP and previous SHARC DSPs (ADSP-2106x) introduce some booting differences. Even with these differences, the ADSP-21161 DSP supports the same boot capability and configuration as the ADSP-2106x DSPs.

The DSP determines the booting mode at reset from the EBOOT, LBOOT, and $\overline{\text{BMS}}$ pin inputs. When EBOOT=0, LBOOT=1, and $\overline{\text{BMS}}$=1, the DSP boots through the link port. For a list showing how to select different boot modes, see the Boot Memory Select pin description in Table 5-3 and Table 5-4 on page 5-13.

When using any of the power-up booting modes, address 0x0004 0004 should not contain a valid instruction since it is not executed during the booting sequence. A NOP or IDLE instruction should be placed at this location.

In link port booting, the DSP gets boot data from another DSP's link port or 4-bit wide external device after system power-up.

The external device must provide a clock signal to the link port assigned to link buffer 0. The clock can be any frequency, up to a maximum of the DSP clock frequency. The clock's falling edges strobe the data into the link port. The most significant 4-bit nibble of the 48-bit instruction must be downloaded first.

Table 5-8 shows how the DMA channel 8 parameter registers are initialized at reset. The count register (CLB0) is initialized to 0x0100 for transferring 256 words to internal memory. The LCTL register is overridden during link port booting to allow link buffer 0 to receive 48-bit data.

Table 5-8. DMA Channel 8 Parameter Register Initialization for Link Port Booting

| Parameter Register | Initialization Value |
|---|---|
| IILB0 | 0x0004 0000 |
| IMLB0 | uninitialized (increment by 1 is automatic) |
| CLB0 | 0x0100 (256 instruction words) |
| CPLB0 | uninitialized |
| GPLB0 | uninitialized |

In systems where multiple DSPs are not connected by the parallel external bus, booting can be accomplished from a single source through the link ports. To simultaneously boot all of the DSPs, a parallel common connection should be made to link buffer 0 on each of the processors. If only a daisy chain connection exists between the processors' link ports, then each DSP can boot the next one in turn. Link buffer 0 must always be used for booting.

## SPI Booting

One of the DSP's booting modes is booting via the SPI port. The SPI port booting uses DMA channel 8 of the I/O processor to transfer the instructions to internal memory. In this boot mode, the DSP receives 8-bit wide data in the SPIRX register.

During the boot process the program loads 256 words into memory locations 0x40000 through 0x400FF. The DSP subsequently begins executing instructions. Because most DSP programs require more than 256 words of instructions and initialization data, the 256 words typically serve as a loading routine for the application. Analog Devices supplies loading routines (loader kernels) that load an entire program through the selected port. These routines come with the development tools. See "ADSP-21161N Boot-Kernels" on page 5-14 for more information.

Refer to "*ADSP-21161 SHARC DSP Hardware Reference*" for more detailed information on DMA and system configurations.

The DSP determines the booting mode at reset from the EBOOT, LBOOT, and $\overline{\text{BMS}}$ pin inputs. When EBOOT=0, LBOOT=1, and $\overline{\text{BMS}}$=0, the DSP boots through the SPI Port. For a list showing how to select different boot modes, see the Boot Memory Select pin description in Table 5-3 and Table 5-4 on page 5-13.

When using any of the power-up booting modes, address 0x0004 0004 should not contain a valid instruction since it is not executed during the booting sequence. A NOP or IDLE instruction should be placed at this location.

In SPI port booting, the DSP gets boot data from another DSP's SPI port or another SPI compatible device after system power-up.

Table 5-9 shows how the DMA channel 8 parameter registers are initialized at reset. The SPI Control Register (SPICTL) is configured to 0x0A001F81 upon reset during on SPI boot.

This configuration sets up the SPIRX register for 32-bit serial transfers. The SPIRX DMA channel 8 parameter registers are configured to DMA in 0x180 number of 32-bit words into internal memory normal word address space starting at 0x40000. Once the 32-bit DMA transfer completes, the data is then accessed as 3-column 48-bit instructions. The DSP executes a 256 (0x100) word loader kernel upon completion of the 32-bit, 0x180 word DMA.

Note that for 16-bit SPI hosts, two words are shifted into the 32-bit receive shift register before a DMA transfer to internal memory occurs. For 8-bit SPI hosts, four words are shifted into the 32-bit receive shift register before a DMA transfer to internal memory occurs.

Table 5-9. DMA Channel 8 Parameter Register Initialization for SPI Port Booting

| Parameter Register | Initialization Value |
|---|---|
| IISRX | 0x0004 0000 |
| IMSRX | uninitialized (increment by 1 is automatic) |
| CSRX | 0x0180 (256 instruction words) |
| GPSRX | uninitialized |

## Multiprocessor Booting

Multiprocessor systems can be booted from a host processor, from external EPROM, through a link port, or from SPI port. The loader currently generates single-processor loader files for SPI, host and link port booting. As a result, the loader currently supports multiprocessor EPROM booting only. The user should have the application code modified to be responsible for properly setting up multiprocessor booting in SPI, host and link port booting modes.

For information on the setup and DMA processes for booting a single processor, see:

- "EPROM Booting" on page 5-19
- "Host Booting" on page 5-24
- "Link Port Booting" on page 5-27
- "SPI Booting" on page 5-30

Refer to "*ADSP-21161 SHARC DSP Hardware Reference*" for more detailed information on DMA and system configurations.

### Multiprocessor EPROM Booting

There are two methods of booting a multiprocessor system from an EPROM. Processors perform the following steps in these methods:

- Arbitrate for the bus
- DMA the 256 word boot stream, after becoming bus master
- Release the bus
- Execute the loaded instructions

### Booting From a Single EPROM

The $\overline{\texttt{BMS}}$ signals from each DSP may be wire-ORed together to drive the chip select pin of the EPROM. Each DSP can boot in turn, according to its priority. When the last one has finished booting, it must inform the others (which may be in the idle state) that program execution can begin (if all DSPs are to begin executing instructions simultaneously).

When multiple DSPs boot from one EPROM, the DSPs can boot either identical code or different code from the EPROM. If the processors load differing code, a jump table (based on processor ID) can be used to select the code for each processor.

### Sequential EPROM Booting

The `EBOOT` pin of the DSP with `IDx=1` must be set high for EPROM booting. All other DSPs should be configured for host booting (`EBOOT=0`, `LBOOT=`0, and `BMS=1`), which leaves them in the idle state at startup and allows the DSP with `IDx=1` to become bus master and boot itself. Only the $\overline{\texttt{BMS}}$ pin of DSP #1 is connected to the chip select of the EPROM. When DSP #1 has finished booting, it can boot the remaining DSPs by writing to their external port DMA buffer 0 (`EPB0`) via the multiprocessor memory space.

## Supporting the Include Format for PROM Booting

In ADSP-21161N DSPs, the "`include`" format has been extended to support the PROM booting, with two word widths available: 8 and 16 bit. The word width is set using the `-HostWidth #` switch.

The switch "`-f include`" can be combined with the switch "`-b prom`" to produce the loader file in the "`include`" format. See Table 5-2 on page 5-6 for information on command-line switches.

The loader output file in the "include" format would, similar to the PROM HEX output, have five basic parts in the following order:

1. PROM start address (from -p # switch)
2. PROM kernel
3. multiprocessor jump table
4. user application code
5. the saved user code which is conflicted with the kernel code in address.

The first two bytes of an output file from the loader will be the 16 bits from the bits 16-31 of the PROM start address set via the -p # switch. The default value is 0x0000. The kernel code will be followed immediately. The processor jump table including the processor IDs and the associated jump addresses is next. Then the user application code is followed with the save user code being the last.

A loader output file produced in the "include" format is in ASCII only.

# Loader Glossary

Boot-kernel—The boot-kernel refers to the executable file that performs the memory initialization on the target.

Boot-loadable file—The boot file refers to the output of the loader that contains the boot loader and the formatted system configurations.

Boot-loading or booting—Booting refers to the process of loading the boot loader, initializing system memory, and starting the application on the target.

Loader—The loader refers to `elfloader` contained in the software release.