4 ADSP-2106X/21160 LOADER

Overview

The loader (`elfloader.exe`) generates boot-loadable files for ADSP-21xxx DSPs by processing executable files. The loader's output is a boot-loadable file (`.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 includes the following information on the loader:

- "Loader Command-Line Reference" on page 4-2

  Contains reference information on the loader commands and operations.

- "Loader Guide" on page 4-11

  Contains information on booting and kernel use.

- "Loader Glossary" on page 4-35

  Contains a glossary of loader-related terms.

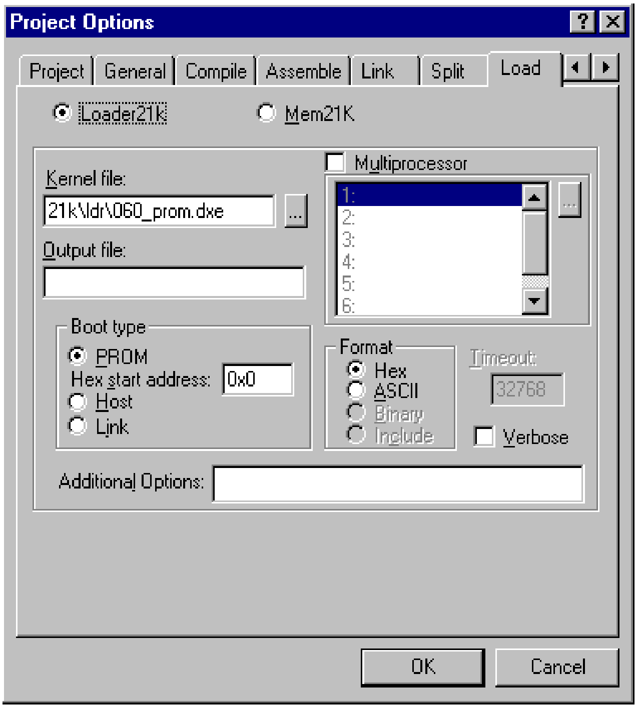
This chapter covers the loader and booting process information for the ADSP-21060, ADSP-21061, ADSP-21062, ADSP-21065L and ADSP-21160 DSPs. For more information on ADSP-21161N loader, see Chapter 5.

Loader Command-Line Reference

This section provides reference information on the loader command line and settings. Refer to Table 4-2 on page 4-6 for switch descriptions.

Setting Loader Options

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



● Loader21k ○ Mem21K

Kernel file: 21k\ldr\060_prom.dxe

Output file:

☐ Multiprocessor

Boot type
● PROM
Hex start address: 0x0
○ Host
○ Link

Format
● Hex
○ ASCII
○ Binary
○ Include

Timeout: 32768

☐ Verbose

Additional Options:

Use the **Additional Options** field in the **Load** tab of the **Project Options** dialog box 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

This section provides reference information on the loader command line and boot-loading. A list of all switches and their descriptions appears in Table 4-2 on page 4-6.

Use the following syntax for the loader command line:

```
elfloader sourcefile -switch [-switch …] -proc processor
```

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` *`processor`* — This is the selection of the processor (target).

ⓘ If you do not specify your targer, the default is ADSP-21062.

The loader command line is case sensitive. The following command line, for example:

```
elfloader p0.dxe -bprom -fhex -l 060_prom.dxe -proc ADSP-21060
```

runs the loader with:

p0.dxe — Provides an executable file to process into a boot-loadable file; because the `-o` switch does not appear on this command line, the name for the output file is `p0.ldr`.

-bprom — Selects EPROM booting as the boot-type for the boot-loadable file

-fhex — Selects Intel Hex-32 as the format for the boot-loadable file

-l 060_prom.dxe — Selects the `060_prom.dxe` file as the boot-kernel for the boot-loadable file

-proc ADSP-21060 — Selects the ADSP-21060 processor

When used with 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-21060 -bprom 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-21060 -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. Table 4-1 lists the type of files, names, and extensions that the loader expects on files.

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 4-1.

Table 4-1. 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.

The loader's command-line switches and their descriptions appear in Table 4-2.

Table 4-2. Loader Command-Line Switches

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`, and `link`. If the `-b` switch does not appear on the command line, the default setting is `-bprom`. 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.
`-c` *`option`*	The `-c` (custom option) switch directs the loader to select one of the addresses: `20004` and `20040` for ADSP-2106x DSPs, `8004` and `8040` for ADSP-21065L DSPs, and `40000` and `40050` for ADSP-21160 DSPs. The loader can pick up a correct address even if this switch is absent in the command line.

Table 4-2. Loader Command-Line Switches (Cont'd)

Switch	Description
`-d` *`processor`*	The `-d` (processor) switch directs the loader to specify a defined processor: • For ADSP-2106x DSP (where x = 0,1, 2 or 5L), the processor selections are `ADSP-21060`, `ADSP-21061`, `ADSP-21062`, and `ADSP-21065L` • For ADSP-21160 DSP, the processor selection is `ADSP-21160` 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. If you do not specify your target, the default is `ADSP-21062`. To select ADSP-21065L DSP, you must enter ADSP-21065L. To select ADSP-21160 DSP, you must enter ADSP-21160.
`-e` *`filenames`*	The `-e` (except shared memory) switch directs the loader to omit the shared memory (.SM) file from the output loader file. If you prepare a boot-loadable file for multiple processors that contains shared memory, only one processor boot image contains the shared memory items. You can use this switch multiple times to omit multiple shared memory files.
`-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. For a `prom` boot-type, use the `hex` or `ASCII` formats; for the `host` or `link` boot-types, use the `ASCII`, `binary`, or `include` formats. 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` or `link`.

Table 4-2. Loader Command-Line Switches (Cont'd)

<table>
<tr><th>Switch</th><th>Description</th></tr>
<tr><td><code>-h|help</code></td><td>The <code>-h</code> (command-line help) switch directs the loader to output the list of command line switches to standard output and exit.</td></tr>
<tr><td><code>-id#exe=</code><i><code>file</code></i></td><td>The <code>-id#exe</code> (multiprocessor ID) switch directs the loader to use # processor ID for the corresponding executable <i><code>file</code></i> 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.
Valid # values are <code>1</code>, <code>2</code>, <code>3</code>, <code>4</code>, <code>5</code> and <code>6</code>. Do not use this switch for single-processor systems; input the executable <i><code>filename</code></i> as a parameter without a switch to select an executable file for single-processor systems.</td></tr>
<tr><td><code>-l</code> <i><code>kernel</code></i></td><td>The <code>-l</code> (boot-kernel) switch directs the loader to use the <i><code>kernel</code></i> file for the boot-loading routine in the output boot-loadable file. Note that the boot-kernel that you select with the <code>-l</code> switch must correspond to the boot-type that you select with the <code>-b</code> switch and the processor-type from the <code>-d</code> or <code>-proc</code> switch. If the <code>-l</code> switch does not appear on the command line, the loader searches for a default boot-kernel file.
Based on the <code>-b</code> switch setting, the loader searches for the default boot-kernel in your DSP Tools installation directory <code>...\21k\ldr</code> or <code>...\211xx\ldr</code>, as follows:
<table>
<tr><th>-b switch</th><th>Default Boot-Kernel</th><th>DSP (from -d or -proc)</th></tr>
<tr><td><code>prom</code></td><td><code>060_prom.dxe</code></td><td><code>ADSP-21060</code>
<code>ADSP-21061</code>
<code>ADSP-21062</code></td></tr>
<tr><td></td><td><code>065L_prom.dxe</code></td><td><code>ADSP-21065L</code></td></tr>
<tr><td></td><td><code>160_prom.dxe</code></td><td><code>ADSP-21160</code></td></tr>
</table>
</td></tr>
</table>

Table 4-2. Loader Command-Line Switches (Cont'd)

Switch	Description

	`host` `060_host.dxe` `ADSP-21060`
	`ADSP-21061`
	`ADSP-21062`
	`065L_host.dxe` `ADSP-21065L`
	`160_host.dxe` `ADSP-21160`
	
	`link` `060_link.dxe` `ADSP-21060`
	`ADSP-21062`
	`160_link.dxe` `ADSP-21160`
	
	Note: ADSP-21160 boot-kernels are in the `..\211xx\ldr` directory.

`-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 "*`sourcefile`*`.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; this EPROM address corresponds to address 0x400000 in an ADSP-2106x DSP.
`-proc` *`ProcessorID`*	The `-proc` (processor) switch directs the loader to select a target processor: `ADSP-21060, ADSP-21061, ADSP-21062`, `ADSP-21065`L, or `ADSP-21160`. If you do not specify your target, the default is `ADSP-21062`. To select ADSP-21065L DSP, you must enter ADSP-21065L. To select ADSP-21160 DSP, you must enter ADSP-21160.

Table 4-2. Loader Command-Line Switches (Cont'd)

Switch	Description
`-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 (`elfloader.exe`) processes your executable files, producing a boot-loadable file. You can automatically download programs to the processor's internal memory after power-up or after a software reset. This process is called *booting*.

The ADSP-21xxx DSPs supports three booting modes: EPROM, host, and link. The DSPs have a special hardware feature that boot-loads a small, 256-instruction program, called "boot-kernel", into the DSP's internal memory at chip reset. When executed, the boot-kernel facilitates booting the user's application code.

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

- "Working With Different Boot Types" on page 4-12
- "Boot-Kernel Features and Boot-Loading Process" on page 4-15
- "Boot-Kernel Changes and Software Issues" on page 4-19
- "EPROM Booting" on page 4-22
- "Host Booting" on page 4-27
- "Link Booting" on page 4-31
- "Multiprocessor Booting" on page 4-32
- "No Boot Mode" on page 4-33
- "Interrupt Vector Table Location" on page 4-34

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 the appropriate boot-types.

Booting

The ADSP-21xxx DSPs always supports three booting modes: EPROM, host, and link, as shown in Table 4-3 and Table 4-4. Boot-loadable files for these modes pack the boot-data into 48-bit instructions and use an appropriate DMA channel of the DSP's DMA controller to boot-load the instructions.

ⓘ The ADSP-2106x DSPs use `DMAC6` when booting. In ADSP-21160 DSPs, the `DMAC8` channel is used for the link port booting and the `DMAC10` channel is used for the host and EPROM booting.

- For booting from an EPROM through the external port, the ADSP-21xxx DSP reads boot-data from an 8-bit external EPROM.
- For booting from a host processor through the external port, the ADSP-21xxx DSP accepts boot-data from a 16- or 32-bit host microprocessor.
- For booting through the link port, the ADSP-21xxx DSP receives boot-data through its link port as 4-bit wide data in link buffer 4.
- For the NO BOOT mode, the ADSP-21xxx DSP starts to execute instructions from external memory.

After the boot process loads 256 words into appropriate memory locations using any boot method: `0x20000` (ADSP-21060/61/62 DSPs), `0x8000` (ADSP-21065L DSP), and `0x40000` (ADSP-21160 DSP), the processor begins executing instructions. Because most applications require more than 256 words of instructions and initialization data, the 256 words typically serve as a loading routine for the application. This loading routine (Loader Kernel or Boot Kernel) is used to load an entire program.

General Powerup Booting Process (PROM/Link/Host)

The ADSP-21xxx DSPs have a special hardware feature that boot-loads a small, 256-instruction program, called “boot-kernel”, into the DSP’s internal memory at chip reset. When executed, the boot-kernel facilitates booting the user’s application code. The combination of the boot-kernel and your program make up a boot-loadable file.

At powerup, a general booting process includes,

1. A DMA channel is automatically configured for a 256 word (48-bit) transfer, starting at: `0x20000` (ADSP-21060/61/62 DSPs), `0x8000` (ADSP-21065L DSP), and `0x40000` (ADSP-21160 DSP).
2. The first 256 instructions are called the “loader kernel” or “boot kernel.”
3. The loader kernel then runs and starts up 48-bit word DMAs to load the application executable code and data, then uses core writes with the data to internal and external memory.
4. Finally the loader kernel overwrites itself with the applications first 256 words from location `0x20000` to `0x200FF` (ADSP-21060/61/62 DSPs), `0x8000` to `0x80FF` (ADSP-21065L DSP), and `0x40000` to `0x400FF` (ADSP-21160 DSP).

Note: ADSP-21061 and ADSP-21065L DSPs do not support link booting.

The boot-kernel can come from an external PROM, a host processor, or another ADSP-21xxx DSP. Choosing the boot type directs the system to prepare the appropriate boot-kernel. You select the boot type using three of the DSP’s external pins as shown in Table 4-3 and Table 4-4.

Selecting A Booting Mode

The booting mode is selected using the `LBOOT`, `EBOOT`, and $\overline{\texttt{BMS}}$ pins. Table 4-3 and Table 4-4 show how to select among the EPROM-, host-, and link-booting modes for an ADSP-2106x/21160 DSP.

Table 4-3. 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 $\overline{\texttt{BMS}}$ 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, $\overline{\texttt{BMS}}$ is output by the bus master. When host-booting, link-booting or SPI-booting, (`EBOOT`=0 and `LBOOT`=0 or =1), $\overline{\texttt{BMS}}$ is an *`input`* and must be high for host-, link-, or SPI-booting.

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

Note that when using any of the power-up booting modes, the reset vector address `0x0002 0004` (ADSP-21060/61/62 DSPs), `0x40004` (ADSP-21160 DSP), or `0x8004` (ADSP-21065L DSP) 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.

Table 4-4. Boot Mode Pins (States)

EBOOT	LBOOT	$\overline{\text{BMS}}$	Booting Mode
1	0	output	EPROM ($\overline{\text{BMS}}$ is chip select)
0	0	1 (input)	Host processor
0	1	0 (input)	Reserved
0	1	1 (input)	Link port
0	0	0 (input)	No booting -- processor executes from external memory
1	1	x (input)	Reserved

Boot-Kernel Features and Boot-Loading Process

For all boot-kernels, the boot-loading process begins by downloading the boot-kernel into the DSP's memory. 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. There are three boot-kernels that come with VisualDSP++ software. Boot kernels are loaded at reset into program segment `seg_ldr`, which is defined in the `06x_ldr.ldf` linker description file (for ADSP-2106x DSPs) and in the `160_ldr.ldf` (for ADSP-21160 DSP).

This file is stored in your `DSP Tools installation...\21k\ldr`) directory for all ADSP-2106x DSPs and in `...\211xx\ldr` directory for ADSP-21160 DSP. For example,

- `060_prom.asm` is the source file for the ADSP-2106x PROM boot-kernel.
- `060_host.asm` is the source file for the ADSP-2106x host boot-kernel.

- `060_link.asm` is the source file for the ADSP-2106x link boot-kernel.

For the ADSP-21065L PROM boot-kernel, the source file is `065L_prom.asm`. For the ADSP-21160 PROM boot-kernel, the source file is `160_prom.asm`, and so on.

Each kernel begins with macros, defining the addresses of IOP registers that the kernel uses. Following the macros, the kernels have a table of interrupt vectors. Of these interrupts, the kernel only uses the reset interrupt and the appropriate `DMA` interrupt. After finishing the DMA transfers that load the boot-kernel into the DSP, the boot-kernel performs the following sequence (for all booting modes):

1. Activates the external port DMA Channel interrupt (`EP0I`), stores the `DMACx` control setting in `R2` for later restore, and clears `DMACx` for new setting.
2. Enables interrupts and nesting for DMA transfer, sets up the `IMASK` register to allow DMA transfer, and sets up the `MODE1` register to enable interrupts and nesting.
3. Gets the processor's ID from the `SYSTAT` register and finds the start address of boot section for the processor. The table starts after the boot-kernel, ends before the first target executable, and contains seven entries in the following 48-bit format:

```
Address(32 bits)      Processor ID (16 bits)
```

Each block is formatted as shown in Table 4-5.

Table 4-5. Boot Memory Formats

Word	Usage	Bits 47-0	
0	Not used	Most significant 32 bits	Boot data block header for both zero and non-zero blocks
	Data type of data block	Least significant 16 bits	
1	Address of internal memory	Most significant 32 bits	
	Length of data block	Least significant 16 bits	
	Boot Data - word 1 - word 2 ; - word n	48 bits 48 bits	Boot Data only if it is not a zero block

Each initialization block is identified by a 16-bit tag which precedes the block. Each type of initialization has a unique tag number (Tag Number - `Initialization Type`):

0-`FINAL INIT`, 1-`ZERO DM16`, 2-`ZERO DM32`, 3-`ZERO DM40`, 4-`INIT DM16`, 5-`INIT DM32`, 6-`INIT DM40`, 7-`ZERO PM16`, 8-`ZERO PM32`, 9-`ZERO PM40`, 10-`ZERO PM48`, 11-`INIT PM16`, 12-`INIT PM32`, 13-`INIT PM40`, and 14-`INIT PM48`

For ADSP-21060 DSPs only:
15-`ZERO DM64`, 16-`INIT DM64`, 17-`ZERO PM64`, and 18-`INIT PM64`

4. Initializes internal and external memory.

5. Loads the first 256 words of target executable file during the final initialization stage and (finally) the boot-kernel overwrites itself.

   The final initialization is indicated by a 0 value tag. When the loader detects this tag, it reads the next 48-bit word. This word indicates the instruction when the loading is completing. This instruction is loaded into the 48-bit `PX` register after the boot loader has finished initializing internal memory.

   The kernel requires that address `0x020040` (ADSP-2106x DSPs), `0x8040` (ADSP-21065L DSPs), or `0x40050` (ADSP-21060 DSPs) contains an RTI instruction because a `DMACx` interrupt is generated when the initialization is complete.

   Note that the kernel assumes `IMDW` is 0 during booting process before it is set to 1 in the final boot stage of the kernel. Also remember that when using any of the power-up booting modes, the first location of the reset vector should not contain a valid instruction because it is not executed during the booting sequence. A `NOP` or `IDLE` instruction should be placed at this location.

   If the boot-kernel is going to initialize external memory, set appropriate values in the `SYSCON` and `WAIT` registers by creating a custom boot-kernel. Note that the value in `DMACx` is nonzero, and `IMASK` is set to allow `DMACx` interrupts. Because the `DMACx` interrupt remains enabled in `IMASK`, clear this interrupt before using this DMA channel again. Otherwise, unintended interrupts may occur.

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. 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` and `WAIT` 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 initialization code may be found in the comments for the kernel file.

After modifying a copy of the applicable boot-kernel sourcefile (for example, `060_link.asm`, `060_host.asm`, or `060_prom.asm`, etc., refer to Table 4-2 on page 4-6 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++ environment, select your modified kernel executable in the **Load** dialog box to apply this customized boot-kernel.

If you have to modify the boot-kernel for EPROM-, host-, 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 `...\21k\ldr\` (or `...\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 `0x20004` (ADSP-21060/61/62 DSPs), `0x40004` (ADSP-21160 DSP), or `0x8004` (ADSP-21065L DSP) does not contain `NOP` or `IDLE`.

If you are working with the C/C++ compiler and using the loader, do not use the `mem21k` utility that initializes DSP runtime-memory for compiled C/C++ code. Disable the `mem21k` utility with the compiler's `-no-mem` switch. For code compiled with the `-no-mem` switch, the `seg_init` segment that is defined in the linker description file need only contain 16 words (address locations).

If you are working with the C/C++ compiler and using the splitter instead of the loader, disable the loader and use the `mem21k` utility. Select the `mem21k` radio button in the **Load** dialog box to disable the loader.

The load kernel project can be rebuild from the VisualDSP++ IDDE. If the VisualDSP++ IDDE is not used, the following command lines demonstrate how to rebuild various default boot-kernels for the ADSP-2106x DSPs:

- The default boot-kernel for EPROM booting is `060_prom.asm`. After copying the default file to `my_prom.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

  ```
  easm21k -ADSP21060 my_prom.asm
  ```

  - or -

  ```
  easm21k -proc ADSP21060 my_prom.asm
  linker -T 060_ldr.ldf my_prom.doj
  ```

- The default boot kernel for host booting is `060_host.asm`. After copying the default file to `my_host.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

  ```
  easm21k -ADSP21060 my_host.asm
  ```

  - or -

  ```
  easm21k -proc ADSP21060 my_host.asm
  linker -T 060_ldr.ldf my_host.doj
  ```

- The default boot kernel for link booting is `060_link.asm`. After copying the default file to `my_link.asm` and modifying it to suit your system, use the following command lines to rebuild the boot-kernel:

  ```
  easm21k -ADSP21060 my_link.asm
  ```

  - or -

  ```
  easm21k -proc ADSP21060 my_link.asm
  linker -T 060_ldr.ldf my_link.doj
  ```

To rebuild boot-kernels for ADSP-21065L or ADSP-21160 DSPs, you can use the appropriate command lines. For example,

```
easm21k -ADSP21065L my_prom.asm
```

- or -

```
easm21k -proc ADSP21065L my_prom.asm
linker -T 065L_ldr.ldf my_prom.doj
```

```
easm21k -ADSP21160 my_prom.asm
```

- or -

```
easm21k -proc ADSP21160 my_prom.asm
linker -T 160_ldr.ldf my_prom.doj
```

EPROM Booting

EPROM booting through the external port is selected when the `EBOOT` input is high and the `LBOOT` input is low. These settings cause the $\overline{\texttt{BMS}}$ pin to become an output, serving as chip select for the EPROM. Table 4-6 lists all PROM connections to DSPs.

Table 4-6. PROM Connections to any ADSP-21xxx DSP

DSP Model	Connection
21060/61/62	PROM/EPROM connected to DATA23-16 pins
21065L	PROM/EPROM connected to DATA0-7 pins
21160	PROM/EPROM connected to DATA32-39 pins
21xxx	Address pins of PROM connect to lowest address pins of any DSP
21xxx	Chip Select connect to the $\overline{\text{BMS}}$ pin
21060/61/62/65L	Output enable connect to the $\overline{\text{RD}}$ pin
21160	Output enable connect to $\overline{\texttt{RDH}}$ pin

Note that during reset, the `ACK` line is internally pulled high with a 2Kohm 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.

The DMA Channel parameter registers are initialized at reset for EPROM booting as shown in Table 4-7 and Table 4-8. The count is initialized to `0x0100` for transferring 256 words to internal memory. The external count register (`ECx)` which is used when external addresses are generated by the

DMA controller, is initialized to `0x0600` (that is, `0x0100` words with six bytes per word).

Table 4-7. ADSP-2106x DMA Settings for EPROM Booting

		ADSP-21xxx DSP Models	
		21060/61/62	21065L
SYSCON		`0x00000010`	`0x00000020`
DMA Channel		`DMAC6`	`DMAC8`
`II6`	`IIEP0`	`0x20000`	`0x8000`
`IM6`	`IMEP0`	`0x1`(implied)	`0x1`(implied)
`C6`	`CEP0`	`0x100`	`0x100`
`EI6`	`EIEP0`	`0x400000`	`0x00000000`
`EM6`	`EMEP0`	`0x1`(implied)	`0x1`(implied)
`EC6`	`ECEP0`	`0x600`	`0x600`
IRQ Vector		`0x20040`	`0x8040`

Table 4-8. ADSP-21160 DMA Settings for EPROM Booting

	ADSP-21160 DSP
SYSCON	`0x00`
DMA Channel	`DMAC10 (EPB0)`
`II10`	`0x40000`
`IM10`	`0x1`(implied)

	ADSP-21160 DSP
`C10`	`0x100`
`EI10`	`0x800000`
`EM10`	`0x1`(implied)
`EC10`	`0x600`
IRQ Vector	`0x40050`

After the DSP’s $\overline{\texttt{RESET}}$ pin goes inactive on started-up, an ADSP-21xxx system that is configured for EPROM booting goes through the following boot-loading sequence:

1. The $\overline{\texttt{BMS}}$ pin becomes the boot EPROM chip select.
2. The DSP goes into an idle state, identical to that caused by the IDLE instruction. The program counter (PC) is set to the DSP reset vector address:

   – `0x20004` for ADSP-21060/61/62 DSPs

   – `0x8004` for the ADSP-21065L DSP

   – `0x40004` for the ADSP-21160 DSP
3. The DMA controller reads 8-bit EPROM words, packs them into 48-bit instruction words and transfers them into internal memory (packing order from low to high bytes) until 256 words have been loaded.
4. The DSP’s DMA parameter registers for appropriate DMA channels initialize, as shown in Table 4-7 and Table 4-8. The External Port DMA Channel (6 or 10) becomes active following reset; it is initialized to set external port DMA enable and selects `DTYPE` for instruction words. The packing mode bits `PMODE` are ignored, and

48-to-8 bit packing is forced with least-significant-word first. The `UBWS` and `UBWM` fields of the wait register are initialized to generate six wait states for the EPROM access in unbanked external memory space.

5. The DSP begins 8-bit DMA transfers from the EPROM to internal memory, using external port data bus lines:

   `D23-D16` for ADSP-21060/61/62 DSPs

   `D0-D8` for the ADSP-21065L DSP

   `D32-D39` for the ADSP-21160 DSP

6. The data transfers begin and increment after each access. The external address lines (`ADDR 31-0`), start at:

   – `0x0040 0000` for ADSP-21060/61/62 DSPs

   – `0x0000 0000` for the ADSP-21065L DSP

   – `0x0080 0000` for the ADSP-21160 DSP

7. The DSP $\overline{\mathtt{RD}}$ pin asserts as in a normal memory access, with six wait states (seven cycles).

8. After finishing DMA transfers to load boot-kernel into the DSP, the `BSO` bit is cleared in `SYSCON` register, deactivating $\overline{\mathtt{BMS}}$ and activating normal external memory select

   The boot-kernel uses three copies of `SYSCON`; one that contains the original value of `SYSCON`, a second that contains `SYSCON` with `BSO` bit set which allows the DSP to gain access to the boot EPROM, and a third with the `BSO` bit cleared.

   When `BSO=1`, the EPROM packing mode bits in the `DMACx` control register are ignored and 8-48 bit packing is forced. (Note that 8-bit packing is only available during EPROM booting or when `BSO` is set). When one of the external port DMA channels is being used in

> conjunction with the `BSO` bit, none of the other three channels may be used. In this mode, $\overline{\texttt{BMS}}$ is not asserted by a core processor access, only by a DMA transfer. This feature lets the boot-kernel perform other external accesses to non-boot memory.

The EPROM is automatically selected by the $\overline{\texttt{BMS}}$ pin after reset; other memory select pins are disabled. The ADSP-21xxx 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 DMA external count register decrements after each EPROM transfer. When it reaches zero, the DMA transfer stops and `EPxx` is activated. `RTI` return the program counter to the setting where kernel begins.

For EPROM booting mode of a single processor with `ID=0`, do not use the `-id#exe` switch; just include the executable on the command line without a switch. For more information, see "Loader Glossary" on page 4-35.

The `WAIT` register `UBWM` (used for EPROM booting) is initialized at reset to both internal wait and external acknowledge required. The internal keeper-latch on the `ACK` pin initially holds acknowledge high (asserted). If acknowledge is driven low by another device during an EPROM boot, it is possible the keeper latch may latch acknowledge low.

The DSP views the deasserted (low) acknowledge as a hold off from the EPROM. In this condition, wait states are continually inserted, preventing the EPROM boot from completing. If you are writing a custom boot-kernel, it is recommended that you change the `WAIT` register early within the boot kernel so that `UBWM` is set to internal wait mode (`01`).

Host Booting

An ADSP-2106x/21160 DSP accepts data from a 16-bit host microprocessor (or other external device) through the external port `EPB0` and pack boot data into 48-bit instructions using appropriate DMA channel. Host is selected when the `EBOOT` and `LBOOT` inputs are low and $\overline{\texttt{BMS}}$ is high. Configured for host booting, the DSP enters the slave mode after reset and waits for the host to download the boot program.

Table 4-9 lists all HOST connections to DSPs.

Table 4-9. HOST Connections to any ADSP-2106x/21160 DSP

DSP Model	Connection/Data Bus Pins
21060/61/62	HOST connected to DATA31-16 pins
21065L	HOST connected to DATA15-0 pins
21160	HOST connected to DATA47-32 pins
21xxx	Address pins of PROM connect to lowest address pins of any DSP
21xxx	Chip Select connect to the $\overline{\text{BMS}}$ pin
21060/61/62/65L	Output enable connect to the $\overline{\text{RD}}$ pin
21160	Output enable connect to $\overline{\text{RDH}}$ pin

After reset, the DSP goes into an idle stage with:

– PC set to address `0x20004` for ADSP-21060/61/62 DSPs

– PC set to address `0x8004` for the ADSP-21065L DSP

– PC set to address `0x40004` for the ADSP-21160 DSP

The parameter registers for External Port DMA Channel (6, 8 or 10) are initialized as shown in Table 4-7 and Table 4-8, except that Registers `GPx`, `IEx` and `EMx` are not initialized. However, no DMA transfers start.

The DMA Channel Control register (`DMAC0` for ADSP-21060/61/62 DSPs), `DMAC8` (for the ADSP-21065L DSP), or `DMAC10` (for the ADSP-21160 DSP) is initialized, which allows external port DMA enable and selects `DTYPE` for instruction words, `PMODE` for 16-to-48 bit word packing, and least-significant-word first.

Because the host processor is accessing the `EPB0` external port buffer, the `HPM` host packing mode bits of the `SYSCON` register must be set to correspond to the external bus width specified by the `PMODE` bits of `DMACx` control register. If a different packing mode is desired, the host must write to `DMACx` and `SYSCON` to change the `PMODE` and `HPM` setting. The host boot file created by the loader requires the host processor to perform the following sequence of actions:

1. The host initiates the booting operation by asserting the DSP's $\overline{\text{HBR}}$ input. This tells the DSP that the default 16-bit bus width is used. The host may also optionally assert the $\overline{\text{CS}}$ chip select input to allow asynchronous transfers.
2. After the host receives the $\overline{\text{HBG}}$ signal back from the DSP, the host can start downloading instructions by writing directly to the external port DMA buffer 0 or the host can change the reset initialization conditions of the DSP by writing to any of the `IOP` control registers. The host must use data bus pins as shown in Table 4-9.
3. The host continues to write 16-bit words to `EPB0` until the entire program is boot-loaded. The host must wait between each host write to external port DMA buffer 0; there must be a pause of at least 12 cycles between the host releasing bus and next host bus-request.

After the host boot-loads the first 256-instructions (the boot-kernel), the initial DMA transfers stop, and the boot-kernel performs the following sequence:

1. Activates external port DMA channel interrupt (`EP0I`), stores the `DMACx` control setting in `R2` for later restore, clears `DMACx` for new setting, and sets the `BUSLCK` bit in the `MODE2` register to lock out the host.
2. Stores the `SYSCON` register value in `R12` for restore.
3. Enables interrupts and nesting for DMA transfer, sets up the `IMASK` register to allow DMA interrupts, and sets up the `MODE1` register to enable interrupts and allow nesting.
4. Loads the DMA control register with `0x00a1` and sets up its parameters to read the data word by word from external buffer 0.

   Each word is read in to address `0x20004` (ADSP-21060/61/62 DSPs), `0x8004` (ADSP-21065L DSP), or `0x40004` (ADSP-21160 DSP) for dispatching. The data through this buffer has structure of boot section which could include more than one initialization block.

5. Clears the `BUSLCK` bit in the `MODE2` register to let the host write in the external buffer 0 right after the appropriate DMA channel is activated.

   For information on the data structure of the boot section and initialization, see "EPROM Booting" on page 4-22.

6. Loads the first 256-words of target executable file during the final initialization stage, and then the kernel overwrites itself.

The final initialization works the same way as with EPROM booting, except that the `BUSLCK` bit in the `MODE2` register is cleared to allow the host to write to the external port buffer.

Note that the default boot-kernel for host booting assumes `IMDW` is set to 0 during boot-loading, except during the final initialization stage. Also note that when using any of the power-up booting modes, locations `0x20004` (ADSP-21060/61/62 DSPs), `0x8004` (ADSP-21065L DSP), or `0x40004` (ADSP-21160 DSP) must not contain a valid instruction because it is not executed during the booting sequence. Place a `NOP` or `IDLE` instruction at this location.

If the kernel is going to initialize external memory, create a custom boot-kernel that sets appropriate values in the `SYSCON` and `WAIT` register. Be aware that the value in the DMA channel register is nonzero, and `IMASK` is set to allow DMA channel register interrupts. Because the DMA interrupt remains enabled in `IMASK`, this interrupt must be cleared before using the DMA channel again. Otherwise, unintended interrupts may occur.

Note that a master ADSP-21xxx DSP may boot a slave ADSP-21xxx DSP by writing to its `DMACx` control register and setting the packing mode (`PMODE`) to 00. This allows instructions to be downloaded directly without packing. The wait state setting of 6 on the slave DSP does not affect the speed of the download since wait states only affect bus master operation.

Link Booting

ⓘ Link booting is supported on all ADSP-21xxx DSPs except for ADSP-21061 and ADSP-21065L DSPs.

When link booting the ADSP-21xxx DSP, the DSP receives data from 4-bit link buffer 4 and packs boot data into 48-bit instructions using appropriate DMA channels (`DMAC6` for ADSP-2106x DSPs and `DMAC8` for ADSP-21160 DSP).

Link mode is selected when the `EBOOT` is low and both `LBOOT` and $\overline{\texttt{BMS}}$ are high. The external device must provide a clock signal to the link port assigned to link buffer 4. 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. The link port acknowledge signal generated by the DSP can be ignored during booting since the link port cannot be preempted by another DMA channel.

Link booting is similar to host booting; the `IIx` and `Cx` parameter registers for DMA channels are initialized to the same values. The DMA Channel 6 Control Register (`DMAC6`) is initialized to `0x00A0`, and DMA Channel 10 Control Register (`DMAC10`) is initialized to `0x0010 0000`. This disables external port DMA and selects `DTYPE` for instruction words. The `LCTL` and `LCOM` link-port control-registers are overridden during link booting to allow link buffer 4 to receive 48-bit data.

Note that after booting completes, the `IMASK` remains set to allow DMA channel interrupts. This interrupt must be cleared before link buffer 4 is again enabled. Otherwise, unintended link interrupts may occur.

Refer to "Host Booting" on page 4-27 for more information on booting process.

Multiprocessor Booting

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

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

- “EPROM Booting” on page 4-22
- “Host Booting” on page 4-27
- “Link Booting” on page 4-31

Multiprocessor EPROM Booting

The loader can produce boot-loadable files that let all ADSP-21xxx DSPs in a multiprocessor system boot in turn from a single EPROM. In such a system, the $\overline{\texttt{BMS}}$ signals from each ADSP-21xxx DSP may be wire-OR’ed together to drive the chip select pin of the EPROM. Each ADSP-21xxx DSP boots in turn, according to its priority. When the last one has finished booting, it must inform the others that program execution can begin.

Besides taking turns when booting, EPROM booting multiple DSPs is similar to single-processor EPROM booting. For more information, see “EPROM Booting” on page 4-22.

When booting a multiprocessor system through a single EPROM:

- Connect all $\overline{\texttt{BMS}}$ pins to EPROM.
- Processor ID#1 will boot first then others will follow.

- The EPROM loader kernel accepts multiple `.exe` and `.dxe` files and reads the ID field in `SYSTAT` to determine which area of EPROM to read.
- All processors should have some sort of software flag or hardware signal (`FLAG` pins) to indicate all booting is complete.

When booting a multiprocessor system through an External Port:

- The host can use the Host Interface
- An ADSP-21xxx DSP that is EPROM-, Host or Link-booted can boot the other DSPs through the EP (Host boot mode)

Note that for multiprocessor EPROM booting, you must use the multiprocessor feature in the Load option dialog box or the `-id#exe` switch in the command-line version of the loader. These features specify which executable files are targeted for DSP specific IDs. For EPROM booting mode of a single processor with `ID=0`, do not use the `-id#exe` switch; name the executable on the command line without a switch. For a single processor with `ID=1`, use the `-id1exe` switch.

Also note that multiple files can only be specified in EPROM booting mode. For host or link booting, you may not specify multiple files.

No Boot Mode

The “no boot” mode of the ADSP-21xxx DSPs causes the processor to start fetching and executing instructions at address `0x0040 0004` (for ADSP-2106x DSPs), `0x0002 0004` (for ADSP-21065L DSP) and `0x0080 0004` (for ADSP-21160 DSP) in external memory space. In this mode, all DMA control and parameter registers are set equal to their default initialization values.

Interrupt Vector Table Location

If the ADSP-21xxx DSP is booted from an external source (i.e. EPROM, host, or link port booting), the interrupt vector table will be located in internal memory. If, however, the ADSP-21xxx DSP is not booted, and is executing from external memory, the vector table must be located in the external memory.

The `IIVT` bit in the `SYSCON` control register can be used to override the booting mode in determining where the interrupt vector table is located. If the DSP is not booted (no boot mode), setting `IIVT` to 1 selects an internal vector table while `IIVT=0` selects an external vector table. If the DSP is booted from an external source (any mode other than no boot mode), then `IIVT` has no effect. The default initialization value of `IIVT` is zero.

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.