

5 ADSP-2191 LOADER

Overview

The ADSP-2191 loader tool, `elfloader`, is used for creating the boot stream file. This loader utility accepts as its input one executable file (`.DXX` file) and produces as an output a loader file (`.LDR` file).

When working within the VisualDSP++ environment, you can set options from the loader command line (or the **Load** tab of the **Project Options** dialog box) for the type of boot-loadable file that the loader produces.

This document contains the following information on the loader:

- [“Bootting in ADSP-2191” on page 5-2](#)

This section contains procedures for using the loader within the VisualDSP environment.

- [“ADSP-2191 Boot Loader Utility” on page 5-9](#)

This section contains reference information on the loader commands and operations.

- [“Loader Glossary” on page 5-14](#)

This section contains a glossary of loader-related terms.

For more information on IDDE, see the *VisualDSP++ 2.0 User’s Guide for ADSP-21xx DSPs*.

Booting in ADSP-2191

Upon power-up, booting can occur either from an 8-bit wide EPROM via the Memory DMA Port, from a host agent via the Host Port DMA, from the UART port, or from an SPI port. Booting can also be initiated in software after reset.

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 and related loader features that support these boot-types.

This section contains the following information:

- [“ADSP-2191 Boot Kernel” on page 5-2](#)
- [“Booting Pins - Modes of Operation” on page 5-3](#)
- [“Boot Stream Contents” on page 5-4](#)
- [“Booting from External EPROM or Host” on page 5-4](#)
- [“No-Boot Mode Booting” on page 5-6](#)
- [“8-bit EPROM Booting” on page 5-6](#)
- [“UART Booting” on page 5-7](#)
- [“Serial EPROM Booting” on page 5-7](#)

ADSP-2191 Boot Kernel

The “Loader/Boot kernel” refers to the resident program in the BOOT ROM space that is responsible for booting the DSP. When the ADSP-2191 comes out of a hardware /RESET, program control jumps to 0xFF0000 and begins the execution of the boot ROM code.

In the ADSP-2191, the highest 16 locations in Page 0 program memory in addition to the highest 272 locations in Page 0 data memory are reserved for use by the ROM boot routines (typically for setting up DMA data

structures and for doing book-keeping operations). You should make sure that neither boot sequence entry code nor boot loaded program is allowed into this space.

Booting Pins - Modes of Operation

The dedicated `BMODE2-0` pins are sampled during hardware reset and are used to control the boot mode that takes affect following the hardware reset.

These pins are the `BMODE0`, `BMODE1`, and `OPMODE`. The state of these pins are captured in corresponding bits 0, 1, and 2 of the Reset Configuration Register, also known as the System Configuration Register (IO address 0x00204).

Table 5-1. Modes of Operation for ADSP-2191

BMODE 1 Pin	BMODE 0 Pin	OPMODE Pin	Description
0	0	0	No Boot, RUN from external 16-bit memory
0	1	0	Boot from EPROM
1	0	0	Boot from Host
1	1	0	Reserved
0	0	1	No Boot, RUN from external 8-bit memory, bypass ROM
0	1	1	Boot from UART

Since the `OPMODE` pin now has a dual role (that of acting as a boot-mode-select during `/RESET`, and also in determining whether the third SPORT on the DSP functions as a SPORT or SPI), it is possible that an application might require the `OPMODE` to be different at runtime

Booting in ADSP-2191

than it is at /RESET (e.g., boot from a UART but want to use SPORT2 during runtime). In such cases, it is the responsibility of the boot kernel to set it accordingly at the end of the booting process.

This means that the `OPMODE` bit can be changed in software at anytime during runtime, as long as the corresponding peripherals are disabled at that time.

Boot Stream Contents

In ADSP-2191s, the boot kernel is located on-chip and stored in a 24-bit wide, 1K location ROM. The starting address of this boot ROM will be at memory address `0xFF0000` (i.e., the first location of page 256), in 1-wait-stated memory.

To bypass the boot ROM, you can use a command-line option in the loader utility to include a user-specified loader kernel as part of and the first component of the boot-stream. In this case, the boot ROM will set up an initial DMA to load in the loader kernel via the medium of booting and then completely hand over control to the loader kernel. It will have no further role in the booting process.

Booting from External EPROM or Host

When booting from an external EPROM or from a host agent, the boot stream format consists of a “header” and “data blocks”.

The ADSP-2191 ROM resident loader is designed to parse and load a specific booting stream format. The boot stream contains a header field that specifies whether the stream is guarded by a checksum. The ADSP-2191 on-chip ROM is located at addresses `0xFF 0000` to `0xFF 03FF`. A boot interrupt will vector to address `0xFF 0000`.

The first word (or header) in the boot stream is a common word that applies to all booting formats. Individual bits within this word are set or cleared based on the method of booting and specific command line

options specified by the user and loader utility (“[Loader Command-Line Switches](#)” on page 5-11). This will be a 16-bit field that contains among other things, information on the number of wait-states and the width of the Host or EPROM (8-bit or 16-bit).

This first header is followed by the regular boot stream, i.e., a series of "headers" and "data blocks", with each header optionally followed by a corresponding block of data. While data blocks typically follows each header, headers that indicate regions of memory that need to be "zero-filled" are not followed by data blocks.

Each header consists of four or six 16-bit words:

1. The first word of a header is a 16-bit field consisting of a flag that indicates whether the block of data to follow is either a 24-bit or 16-bit payload or zero-initialized data. The flag will also uniquely identify the last block that needs to be transferred.
2. The second word of a header (16-bit field) contains the lower 16 bits of the 24-bit start address to begin loading the data (destination). The first octet will be the 8 LSBs, followed by the next most significant bits (8-15), and so on.
3. The third word (also a 16-bit field) contains the upper-most 8 bits of the 24-bit destination address, padded (suffixed) with a byte of zeros.
4. The fourth word (16-bit field) contains the word count of the payload. As with the address, the first octet will be the 8 LSBs, the second octet will be the 8 MSBs.

In cases where a checksum is used to verify accuracy of booting, there will be an extra word in the checksum. This fifth word (also a 16-bit field) will be the CRC-16 checksum for the header and the data block immediately following it. The header will be buffered with a dummy word of zeros to ease the EMI addressing issue.

The header is followed by the payload data itself. 16-bit data is sent in a 16-bit field while 24-bit data is sent in a 32-bit field.



24-bit data will be represented differently in the boot stream from 24-bit addresses. 32-bit data will be transmitted the following way - a byte of zeros, bits 0-7, followed by bits 8-15, and finally bits 16-24.

No-Boot Mode Booting

The BMODE2-0 is strapped to either a 000 or 001, the ADSP-2191 comes out of hardware reset and begins to execute code from Page 1 memory space (0x01 0000). The packing mode selected is dependent on the state of the BMODE0 pin (0 = 8-bit ext/24-bit int or 1 = 16-bit ext/24-bit int). The External Port Interface is configured to operate with the default divide by 128 clock and read wait state count of 7.

Note: This mode does not use the boot stream format.

8-bit EPROM Booting

When booting from an 8-bit EPROM, direct DSP core accesses and Memory DMA are used under control of an EPROM boot routine located in the ROM space to load a boot stream formatted program located in 0x000000 of the boot space. Appropriate packing modes will be selected based on the requirements of the boot stream (either 8-bit ext/24-bit int or 8-bit ext/16-bit int).

Each page of boot space is 64K words long, so 16-bits will address the EPROM per page. The upper 8 address bits will specify the boot page. Upon hardware reset, booting will be from page 0.

The External Port Interface will use its default configuration of divide by 128 clock and 7 wait states to access the EPROM. While booting via EPROM boot space, the highest 16 locations in Page 0 program memory block (0x7FF0 to 0x7FFF) and the top 272 locations of Page 0 data memory block (0xFE00 to 0xFFFF) are reserved for use by the ROM boot routines.

UART Booting

When booting via the UART port, a host downloads a boot stream formatted program to the DSP following an auto-baud handshake sequence. The auto-baud feature provides for less complication during system design since the user is not required to calculate boot clock rates as a function of crystal frequency, core clock divider, peripheral clock mode, etc.

The booting host can select a baud rate (including a non-standard rate) anywhere within the UART clocking capabilities.

Following a hardware or software reset, the ADSP-2191 will begin transmitting 0xFF values and will expect to receive 0xAA. The DSP will continue receiving the boot stream formatted serial stream until the `<end_of_strm>` field is received.

This boot operation is controlled by a UART boot routine in the internal ROM space. While booting via UART, the highest 16 locations in Page 0 program memory block (0x7FF0 to 0x7FFF) and the top 272 locations of Page 0 data memory block (0xFE00 to 0xFFFF) are reserved for use by the ROM boot routine.

Serial EPROM Booting

When booting from a SPI compatible EPROM, SPI0 port is used. The SPI port will select a single serial EPROM device using the PF0 pin as a chip select, submit a read command and address 0x00, and begin to clock consecutive data into memory (either internal memory or external memory) at a `SCK` clock frequency of `core_clock/288`.

Two types of SPI EEPROM devices are supported: devices of 4K bytes and smaller (12-bit address range), and those larger than 4K bytes (16-bit address range).

Data will continue to be accessed until the `<end_of_strm>` field is read.

Booting in ADSP-2191

This boot operation is controlled by a SPI boot routine in internal ROM space. While booting via Serial EPROM, the highest 16 locations in Page 0 program memory block (0x7FF0 to 0x7FFF) and the top 272 locations of Page 0 data memory block (0xFEF0 to 0xFFFF) are reserved for use by the ROM boot routine.

ADSP-2191 Boot Loader Utility

The loader (`elfloader.exe`) produces a boot stream file by processing an executable file (`.DXE` file) generated by the linker (`linker.exe`) and outputting a loader file (`.LDR` file).

This section provides reference information on the loader command line and boot-loading. A description for each switch appears in [Table 5-3 on page 5-11](#).



When using the linker within the VisualDSP++ environment, the settings in the linker options dialog box correspond to linker command line switches. For more information, see the *VisualDSP++ 2.0 User's Guide for ADSP-21xx DSP*

ADSP-2191 Loader Command-Line Reference

Use the following syntax for the loader command line:

```
elfloader sourcefile -dADSP-2191 -switch [-switch ...]
```

where:

- `sourcefile` — This is the name of the executable file or files (`.DXE`) to be processed. 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.
- `-dADSP-2191` — This is the mandatory switch that tells the loader to generate an output file for the ADSP-2191 processor (`.ldr`).

ADSP-2191 Boot Loader Utility

In the following example, the loader command line,

```
elfloader p0.dxe -dADSP-2191
```

runs the loader with:

p0.dxe — Provides an executable file(s) to process into a boot-loadable file.

-dADSP-2191 — Provides a loader output. By default, the output filename will be **po.ldr** if it is not specified on the command line using the **-o <filename>** switch.

Many loader switches take a file name as an optional parameter. [Table 5-2 on page 5-11](#) 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. 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 5-2](#).

Table 5-2. File Name Extension Conventions

Extension	File Description
.dxe ¹	Executable files and boot-kernel files
.ldr	Loader output file for EPROMs

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

Descriptions for each switch appear in [Table 5-3](#).

Table 5-3. Loader Command-Line Switches¹

Switch	Description
<i>filename</i>	The <i>filename</i> parameter without a switch selects an executable file for single-processor boot-loadable files. For multiprocessor boot-loadable files, use the <code>-id#exe=file</code> switch.
<code>-b type</code>	The <code>-b</code> (boot-type) switch directs the loader to prepare a boot-loadable file for the <i>type</i> booting-mode. Valid <i>type</i> selections are PROM (default), HOST, UART, SPI, and NOBOOT. Note that the boot-type that you select with the <code>-b</code> switch must correspond to the boot-kernel that you select with the <code>-l</code> switch and the file format that you select with the <code>-f</code> switch.
<code>-blocksize #</code>	This switch specifies in decimal the size in words, of each block of data in the bootstream. Defaults to 6K words. Valid block sizes can range upto a maximum of 64K words.
<code>-checksum</code>	This switch directs the loader to calculate and generate checksums for each block of code/data. Currently, the only valid input is "CRC16"

ADSP-2191 Boot Loader Utility

Table 5-3. Loader Command-Line Switches¹ (Cont'd)

Switch	Description
-clkdivide <i>#</i>	This switch specifies the base clock divide factor. Acceptable values are 0 to 7 (both inclusive). Defaults to 5. Applies only to EPROM and HOST boot modes.
-dADSP-2191	This mandatory switch tells the loader to generate an output file for the ADSP-2191 processor (.ldr).
-endian	This switch specifies the big endian format; without it, the little endian format (default) is used.
-f <i>format</i>	The -f (boot file format) switch directs the loader to prepare a boot-loadable file in the specified <i>format</i> . The available <i>format</i> selections are hex (Intel Hex-32), ASCII and binary. Hex is the default.
-h	The -h (command line help) switch directs the loader to output the list of command line switches to standard output and exit.
-l <i>file</i>	The -l switch allow the user to bypass the BOOT ROM kernel entirely. If this switch is specified, the first word of the boot stream will contain a predetermined format. On seeing this, the BOOT ROM kernel proceeds to set up a DMA to transfer 256 words that correspond to the loader specified in the command line, into the first 256 locations of internal memory (from where the DSP is booting from) and then perform a JUMP to location 0. At this point, it relinquishes any control over the booting process.
-o <i>filename</i>	The -o (output file) switch specifies the output file name. Defaults to the executable file. Extension is .LDR
-opmode <i>#</i>	This switch specifies whether the boot kernel should set DSP in 3-SPORT mode (0) or 2-SPORT/1-SPI mode (1) at the end of booting. Default is 0 (3-SPORT mode).
-v	The -v (verbose loader messages) switch directs the loader to output status information as the loader processes your files.

Table 5-3. Loader Command-Line Switches¹ (Cont'd)

Switch	Description
-waits #	This switch determines the number of wait states for external accesses. Acceptable inputs are 0 to 7 (both inclusive). Defaults to 7. Only applies to EPROM and HOST boot modes.
-width #	This [external bus width] switch specifies the bit width for EPROM/FLASH or HOST: 8 (default) or 16.
-M	The -M switch directs the boot loader utility to show the dependencies only
-MM	The -MM switch directs the boot loader utility to show the dependencies while processing the input files and producing an output file.
-Mo <i>filename</i>	The -Mo switch directs the boot loader utility to put the dependency information to the output file specified in name filename. -Mo switch must be used in conjunction with -M or -MM.
Mt <i>TargetName</i>	The -Mt switch directs the boot loader utility to use the <i>targetname</i> in the dependency file. -Mo switch must be used in conjunction with -M or -MM.

¹ Items shown in [] are optional. Items shown in *italics* are user defined and are described with each switch.

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 `ld219x` contained in the software release.