

4 ADSP-218X LOADER

Overview

The ADSP-218x loader/splitter tool can convert executable files into boot-loadable files for an ADSP-218x DSP. From the loader command line (or the Load tab of the Project Options dialog box when working within the VisualDSP environment), you can set options for the type of boot-loadable file that the loader produces. This document contains the following information on the loader:

- [“ADSP-218x Loader Guide” on page 4-2](#)

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

- [“ADSP-218x Loader Command-Line Reference” on page 4-8](#)

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

- [“Loader Glossary” on page 4-13](#)

This section contains a glossary of loader-related terms.



The ADSP218x splitter/loader tool can also prepare non-bootable PROM image files, which execute from DSP external memory. In most all instances, developers working with an ADSP-21xx DSP should use the loader instead of the splitter. [For more information, see “Splitter”.](#)

ADSP-218x Loader Guide

The loader/splitter tool (`elfspl21.exe`) processes your executable files, producing a boot-loadable file. Loader operations depend on a set of loader options.

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 the Load tab of the Project Options dialog box in the VisualDSP++ environment.



The options in the Load options dialog correspond to command-line switches in the command-line version of the loader.

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

- [“Working With Different Boot Types” on page 4-2](#)
- [“Setting Loader Options” on page 4-7](#)

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-218x DSPs and relates loader features that support these boot-types.

ADSP-218x DSPs have a special hardware feature that boot-loads a small program into the DSP’s internal memory at chip reset. This program can come from an external PROM, a host processor, or another ADSP-218x DSP. You select the boot-type using three of the DSP’s external pins as shown in [Table 4-1](#). 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-218x uses an internal encoded kernel by default, but provides for replacement of this kernel through a command-line switch.

The ADSP-218x DSPs supports boot loading via an EPROM through BDMA port of the processor, or via a host processor through the IDMA host interface port. For the ADSP-2181 and ADSP-2183 processors, the BMODE pin determines the boot mode for the DSP (see [Table 4-1](#)). For the remainder of the ADSP-218x family of DSPs, (ADSP-2184 through ADSP-2189), the `MODE A` pin determines the boot mode for the processor (see [Table 4-2](#)).

Bootstrap Loading (Booting)

The ADSP-218x has two mechanisms to allow automatic loading of the on-chip program memory after reset: normally, these processors can be configured to boot in only one of two methods -- either from an EPROM or from a host processor via the IDMA port.

The ADSP-218x has two mechanisms to allow automatic loading of on-chip memory (both Program Memory and Data Memory) after reset. These processors can be configured to boot in one of two methods - either from an EPROM via the BDMA port or from a host processor via the IDMA port.

The booting mode is determined by the state of an external pin of the processor on the rising edge of the `/RESET` signal. For the ADSP-2181 and ADSP-2183 processors, the `BMODE` pin determines the booting mode: when `BMODE=0`, the DSP is configured for booting via BDMA, when `BMODE=1`, the DSP is configured for an IDMA boot.

For all other ADSP-218x processors, the `MODE A` pin determines the booting mode of the processor; when `MODE A=0`, the processor is configured for booting via BDMA, or when `MODE A=1`, the processor is configured for booting via IDMA.

BDMA - EPROM Booting

When the `BMODE` pin (or `MODE A` pin where applicable) configure the DSP for BDMA booting, the ADSP-218x processor initiates a BDMA boot sequence when the `/RESET` signal goes high (inactive). The BDMA interface is configured by default upon reset to the following default values when BDMA booting is specified: the `BDIR`, `BTYPE`, and `BMPAGE` bit fields of the BDMA Control register are set to zero, the `BIAD` and `BEAD` registers are set to zero, and the `BWCOUNT` register is set to 32. These default register values configure the DSP to read 32 Program Memory words (or 96 bytes) from the EPROM, and load them into the first 32 locations of Program Memory. These 32 words are used to set up the BDMA to load in the remaining program code. The `BCR` bit (of the BDMA Control Register) is also set to one, which causes program execution to be held off until all 32 Program Memory words (or 96 bytes) are loaded into on-chip Program Memory. Once all 32 PM words are read in by the BDMA port interface, the processor resets, and next begins program execution at `PM 0x0000`.

The ADSP-218x loader, `elfsp121.exe`, lets you create BDMA bootable programs by adding a boot loader kernel to your executable program. The loader generates the code that initializes up to six pages of Program Memory and four pages of Data Memory, where each page is 16K bytes in size.

IDMA - Host Booting

The IDMA Port provides an efficient means of communication between a host system and the ADSP-218x. The port is used to access the on-chip program memory and data memory of the DSP. The IDMA port cannot, however, be used to write to the DSP's memory-mapped control registers.

The IDMA port has a 16-bit multiplexed address and data bus and supports 24-bit program memory. The IDMA port is completely asynchronous and can be written to while the ADSP-218x is operating at full speed. The DSP memory address is latched and then automatically incremented after each IDMA transaction. An external device can therefore access a block of sequentially addressed memory by specifying only

the starting address of the block. This increases throughput as the address does not have to be sent for each memory access.

Refer to page 13 for more information on using the .IDM files and IDMA booting.

Table 4-1. Modes of Operation for ADSP-2181 and ADSP-2183 Only

MMAP Pin	BMODE Pin	Description
0	0	BDMA feature is used in default mode to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded.
0	1	IDMA feature is used to load any internal memory as desired. Program execution is held off until internal program memory location 0 is written to.
1	X	Bootstrap features disabled. Program execution immediately starts from location 0.

The following DSPs have the Mode D mode of operation (Mode D pin):

ADSP-2185M	ADSP-2184N
ADSP-2186M	ADSP-2185N
ADSP-2187L	ADSP-2186N
ADSP-2188M	ADSP-2187N
ADSP-2189M	ADSP-2188N
	ADSP-2189N

Table 4-2. Modes of Operation for ADSP-2184 to ADSP-2189

Mode D	Mode C	Mode B	Mode A	Description
X	0	0	0	BDMA feature is used to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded. Chip is configured in Full Memory Mode.*
X	0	1	0	No automatic boot operations occur. Program execution starts at external memory location 0. Chip is configured in Full Memory Mode. BDMA can still be used, but the processor does not automatically use or wait for these operations.
0	1	0	0	BDMA feature is used to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded. Chip is configured in Host Mode. $\overline{\text{IACK}}$ has active pull-down.* (REQUIRES ADDITIONAL HARDWARE).
0	1	0	1	IDMA feature is used to load any internal memory as desired. Program execution is held off until the host writes to internal program memory location 0. Chip is configured in Host Mode. $\overline{\text{IACK}}$ has active pull-down.*

Mode D	Mode C	Mode B	Mode A	Description
1	1	0	0	BDMA feature is used to load the first 32 program memory words from the byte memory space. Program execution is held off until all 32 words have been loaded. Chip is configured in Host Mode; $\overline{\text{IACK}}$ requires external pull down. (REQUIRES ADDITIONAL HARDWARE)
1	1	0	1	IDMA feature is used to load any internal memory as desired. Program execution is held off until the host writes to internal program memory location 0. Chip is configured in Host Mode. $\overline{\text{IACK}}$ requires external pull-down.*

NOTE: * Considered as standard operating settings. Using these configurations allows for easier design and better memory management.

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.

The loader has command line switches that correspond to the Load dialog's options. Use the Additional Options field in the Load tab of the Project Options dialog box to enter the appropriate file names and command line switches to support the loader operation.

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

ADSP-218x Loader Command-Line Reference

The loader (`elfspl21.exe`) generates a boot-loadable image file, (`.BNM` for EPROM image files, `.IDM` for IDMA bootable image files), for the ADSP-218x family DSPs by processing the executable files (`.DXE`) generated by the linker (`linker.exe`). When making a boot-loadable file, the loader adds code from a default boot-kernel file (a user-generated boot-kernel file can also be used), which gets placed before your executable code in the `.BNM` file, to allow initialization of all internal memory segments of the DSP during boot time. You can load and simulate an EPROM boot in a simulator session in the VisualDSP++ debugger. This allows for debugging of the boot loader kernel as well as the debugging of your code.

This section provides reference information on the loader command line and boot-loading. A description for each switch appears in [Table 4-4 on page 4-11](#).



When using the linker within the VisualDSP++ 2.0 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 DSPs*.

Use the following syntax for the ADSP-218x loader command line:

```
elfspl21 sourcefile outputfile -switch [-switch ...]
```

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.
- `outputfile` — This is the name of the output file. A file name can include the drive, directory, file name and file extension (`.bnm` or `.idm`).

In the following example, the loader command line,

```
elfspl21 p0.dxe output.bnm -loader -i -u load test.doj
```

runs the loader with:

`p0.dxe` — Provides an executable file to process into a boot-loadable file.

`output` — Provides a loader output filename.

`-loader` — Selects EPROM booting as the boot-type for the boot-loadable file.

`-i` — Selects Intel Hex-32 as the format for the boot-loadable file (`-u` selects s-record (i.e, S2)).

`-u load <test.doj>` — Selects the `test.doj` file as the boot-kernel for the boot-loadable file.

Many loader switches take a file name as an optional parameter. [Table 4-3 on page 4-11](#) lists the type of files, names, and extensions that the loader expects on files.

ADSP-218x Loader Command-Line Reference

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.

For more information on file searches, see the *Product Bulletin for VisualDSP & ADSP-21xx Family DSPs*.

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 switches may be used in any order on the command line except for the *sourcefile* and *outputfile* files, -2181 (or -218x) and -loader switches. The *sourcefile* and *outputfile* files must be placed first on the command line. If they needed, the -2181 (or -218x) and -loader switches must always follow the *outputfile* file, while the -2181 (or -218x) switches must always be placed before the -loader switch.

The loader follows the conventions for file name extensions that appear in [Table 4-3](#).

Table 4-3. File Name Extension Conventions

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

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 .BNM file.

Descriptions for each switch appear in [Table 4-4](#).

Table 4-4. Loader Command-Line Switches¹

Switch	Description
<i>source file</i>	The <i>sourcefile</i> parameter without a switch selects an executable file for single-processor boot-loadable files. For information on shared and overlay memory files, see Table 4-3 .
<i>output file</i>	The <i>outputfile</i> parameter select the loader's output file.
-byte	This switch directs the loader to use the byte stream output format
-h	The -h (command line help) switch directs the loader to output the list of command line switches to standard output and exit.
-i	This switch directs the loader to use the Intel hex format
-noloader	When used with -2181 -loader, this switch generates a byte memory image without a boot loader.

ADSP-218x Loader Command-Line Reference

Table 4-4. Loader Command-Line Switches¹ (Cont'd)

Switch	Description
-v	The -v (verbose loader messages) switch directs the loader to output status information as the loader processes your files.
-u	The -u (user flags) switch allows to select a byte-stacked format file. -u s1 format (default) -us s1 format with bytes stacked for 8 bit memory -us2 s2 format with bytes stacked for 8 bit memory -ui Intel format with bytes stacked for 8 bit memory
-2181	When used with -2181 -loader, this switch keeps the image.
-218x	This switch supports potential internal PMOVLAY/DMOVLAY pages. Use in place of -2181. For example, -2187, -2188, -2189..
-bdma <i>inputfile</i> <i>start_address</i>	For use with -2181 -loader. This option allows you to specify an additional .dxe file (inputfile) to be placed in byte memory, starting at the specified address. The loader returns an error if the specified address is in use by the boot loader, or another additional -bdma specified file.
-bdmaload <i>inputfile</i> <i>start_address</i>	This switch allows the user to specify an additional load frame (inputfile) to be placed in byte memory, starting at the specified address.
-unload <i>file</i>	This switch reads the BDMA boot loader from a <file.doj>
-idma	This switch produces a set of records suitable for IDMA programming (see page 13 for more information).
offsetaddr #	This switch provides the byte memory address offset (this switch only valid with -2181 -noloader)
offsetpage #	This switch provides the byte memory page offset (switch only valid with -2181 -noloader)

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

The switch "-idma" makes the `elfspl21` produce a series of records suitable for programming the 218x family IDMA unit. This switch implies "-2181", and it overrides "-loader".

The .IDM file is in ASCII text format, consisting of

```
<count> (of 16 bit words in the record; same as the number of DM
words, 2X number of PM words)
<IDMA Control Register value>
<overlay page> (optional, only when invoked with -218x instead of
-2181)
<16 bit words> (PM have 16 MSBs first, then 8lsbs, right-justified).
```

The final record programs the restart vector and is followed by the word "ffff", which serves as an end marker since it occurs in the position of a count word, but the maximum count for a 21xx part would be 4000 (hex).

For example,

```
2002
4000
0000
FABC
:
FFFF
```

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 `elfspl21` contained in the software release.

Loader Glossary