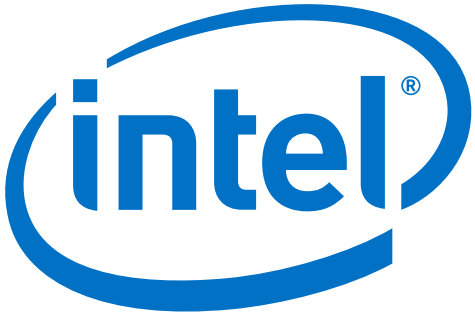

Movidius™

moviConvert

Manual

Version 00.90.0 / 2018-05-18

Table of Contents

Copyright and Proprietary Information Notice

Movidius Ltd.
1730 South El Camino Real, Suite 200
San Mateo, CA 94402
http://www.movidius.com/

1. Usage examples

The usage diagram is presented below in **Figure 1**.

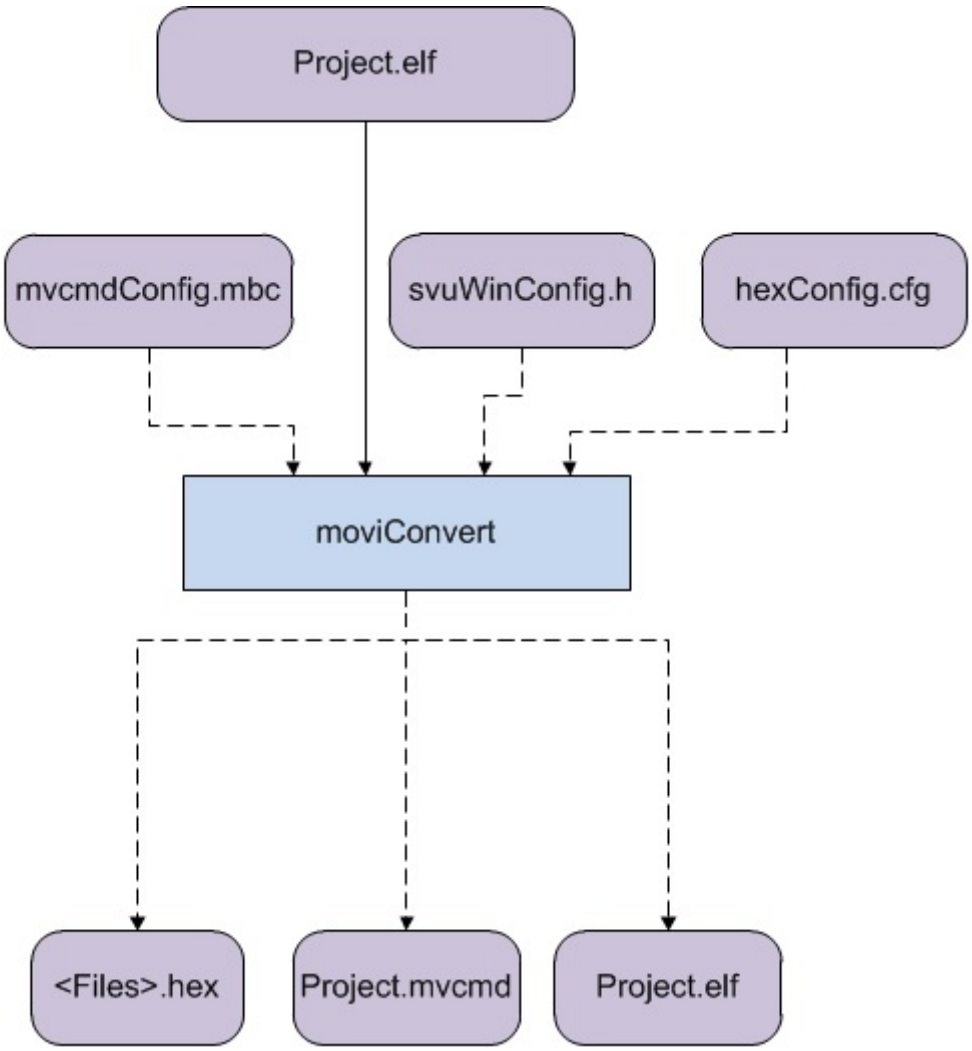

Figure 1: moviConvert Usage Diagram

The `moviConvert` tool takes the final `.elf` file of the project and some optional files as input:

- `SHAVE` windows configuration files, `svuWinConfig.h`, (if the SHAVE window configuration is present or not in the `.elf` file needs to be updated). Description of the file is presented in Section 3 `mvcmdConfig.mbc`, a configuration file for the `.mvcmd` output file. It is described in Section 5.
- mvcmd configuration file, `.mbc` (Movidius Boot Configuration), which contains boot options and which describes the final output image.
- Hex configuration file, `.cfg`, which contains descriptions of the memories that hex files are needed for

Output files:

- `.hex` files
- `.mvcmd` file: boot image file

- `.elf` file

2. Command Line Parameters

2.1. Used syntax

Syntax	Description
`[foo]`	Optional string
`<fooBar>`	Specified string (e.g. The name of a file)

Table 1. Used Syntax

2.2. Available Parameters

Parameter	Description
`-h[elp]`	Display the help screen
`-version`	Display the version and exit
`-v[erbose]`	Display detailed information about the process
`-winConfig [:<configFileName>]`	Specify the `SHAVE` windows config file
`-hex:<outHexDir>`	Output `.hex` files specifying the folder for them
`-hexConfig:<fileName>`	Specify a hex config file (Myriad2 only)
`-hexDdrPad`	Pad ddr memories (Myriad2 only)
`-hexPad`	pad non ddr memories (Myriad2 only)
`-elf[:<fileName>]`	enable .elf output with optional file name
`-mvcmd[:<fileName>]`	enable .mvcmd output with optional file name
`-mvcmdConfig:<fileName>`	use <fileName> for configuring `.mvcmd`
`-mvcmdVersion:<XXYYZZ>`	specify version string (6 characters) which is inserted in the .mvcmd header. (e.g.: `-mvcmdVersion:503000`)
`-mvcmdDescription:<string>`	specify the description string (max. 10 characters) which is inserted in the `.mvcmd header`. For using this, the `mvcmdVersion` needs to be active, too. (e.g.: `-mvcmdDescription:3D Convert`)
`-ddrSize:<sizeInMB>`	specify the DDR size in MB (e.g.: `-ddrSize:16`)
`-cv:<chipVersion>`	specify the chip version
`-map[:<fileName>]`	output the map of the input file
`-elfInput`	Specifies that the input file is an `elf` format file
`-noMvcmdSignature`	Disable signature output at start of `.mvcmd` file
`-callThenContinue`	Enable the “call and continue” feature in the `Myriad 2 ROM`

Parameter	Description
-mvcmdInit:<fileName>	Specify an initialization .mvcmd file to be pre-pended to the one generated from the specified elf file.
-initOnlyForDdrSections	Enable DDR initialization only when DDR sections are present in the input elf file
-mvcmdPadSize:<sizeInKB>	Specifies the size of the padding that is to be inserted after the DDR initialization mvcmd file in the final mvcmd
-secureBoot	Enable the secured boot mode for mvcmd creation
-generateKeys	Enable the generate secure keys mode
-secureBufferSize:<sizeInKB>	Specify the DDR buffer size to be used by the encrypted boot
-securePrivateKey:<path>	Specifies the private key file for secure boot
-secureAesKey:<path>	Specifies the AES key file for secure boot
-securePublicKey:<path>	Specifies the public key file for secure boot
-D:symName=symValue	Define a script variable. The variable can be referenced within the mvcmd configuration script using the $symName directive
-script fileScriptName	Specifies the script file to be used for generating the .mvcmd. In Table 3 are described the available commands.
-ddrinitPBAddress:<value>	Specifies the DDR init process block address.
-overridePayloadPaddingBytes:<value>	set to the value specified rounded up to the nearest multiple of 16 bytes. (only ma2x5x).

Table 2. Available Parameters

3. Secure boot MVCMD

3.1. Generating secure boot keys

The moviConvert tool can be used to generate the public key, private key and AES encryption key, which can be used afterwards for creating secure boot mvcmd images.

The user is required to provide the appropriate file names for the key files.

An example of the command line arguments that can be for generating encryption keys is listed below:

```
moviConvert -generateKeys -securePrivateKey:key.priv -securePublicKey:key.pub -secureAesKey:key.aes
```

3.2. Generating a secure MVCMD image file

In order to generate a secure boot MVCMD, several command line options are mandatory to be present:

- The -secureBoot switch
- The application elf file to convert
- An architecture that supports encrypted boot needs to be specified via the -cv switch (e.g.: -cv:ma2155)
- The files for the secure private key and the secure AES key must be specified

In addition, the user can specify the following optional switches:

- Using -secureBufferSize:<sizeInKB> the user can configure the buffer size at the top of DDR used by the ROM in order to decrypt the MVCMD. This will affect how moviConvert packs the elf data into PROCESS_BLOCK sections (more information on what a PROCESS_BLOCK is can be found in the ROM specification). The default size of 128MB will be used if the argument is not present or if the value given is 0 (zero). If the input elf contains sections în DDR which overlap with the DDR buffer, the tool will try to reduce the size of the buffer so that it avoids overlap and generate an error if the resulting buffer is smaller than any of the elf sections sizes.
- Using -mvcmdInit:<fileName> will instruct to securely prepend the specified initialization elf file to the elf application
- The -mvcmdPadSize:<sizeInKB> switch can be used to configure the padding that is to be inserted right after the initialization elf. The default value is 0.

Example:

- Generate a secure boot file for ma2155 out of the app.elf application binary, using default settings:

```
moviConvert -cv:ma2155 -secureBoot app.elf -securePrivateKey:key.priv -secureAesKey:key.aes
```

- Generate a secure boot file for ma2155 out of the app.elf application, using a DDR buffer of 128 KB, the ddrinit.mvcmd special DDR initialization boot file and 10 KB of padding between the initialization mvcmd and the elf application:

```
moviConvert -cv:ma2155 -secureBoot app.elf -securePrivateKey:key.priv -secureAesKey:key.aes
-secureBufferSize:128 -mvcmdInit:ddrinit.mvcmd -mvcmdPadSize:10
```

4. SHAVE Windows Configuration File

The user can set the SHAVE window registers using one of the following methods:

- Using a .asm section which has the start address at the WINDOW_A address and having a variable length in which the user can set the window registers

Example: .data windowRegs0 0x80140010 winRegs0: .int 0x10008000, 0x10000000, 0x10018000, 0x10010000

- Using a debugger batch file or debugger commands like:

Example:

```
set 0x80140010, 0x10000000
```

- Using the LEON code:

Example:

```
//Set window registers for a specified SHAVE
void SetWindowRegisters(unsigned int shaveNumber, unsigned int windowA, unsigned int windowB, unsigned int windowC,
unsigned int windowD)
{
   /Calculate address of the WindowA register
   unsigned int address = 0x80140010 + 0x10000 * shaveNumber;
   /Set each register
   SET_REG_WORD(address + 0x0, windowA);
   SET_REG_WORD(address + 0x4, windowB);
   SET_REG_WORD(address + 0x8, windowC);
   SET_REG_WORD(address + 0xC, windowD);
}
/Configure Window registers
SetWindowRegisters(0, 0x10008000, 0x10000000, 0x10010000, 0x10018000);
```

In order to cover all the cases (or as many as possible), a `-winConfig` command line switch has been added to the `moviConvert` tool, which has the following syntax:

```
-winConfig[:<configFileName]
```

If a `<configFileName>` is specified (e.g.: `moviConvert.elf -winConfig:../Test/file1.h fileName.elf`), that file is used for window configuration, even if the `.elf` file contains sections for setting window registers of the `SHAVEs`.

If `no <configFileName>` is specified (e.g.: `moviConvert.elf -winConfig fileName1.elf`), a default config file is considered, which is the name of the input `.elf` file with the file extension changed to `.h`.

The format of the file name should be in C format which defines an array of values which are considered as default window registers. The config file could be something like this:

```
u32 svu_win_regs[8][4]=
 {
  /*winRegs0*/ {0x10008000, 0x10000000, 0x10018000, 0x10010000},
  /*winRegs1*/ {0x10028000, 0x10020000, 0x10038000, 0x10030000},
  /*winRegs2*/ {0x10048000, 0x10040000, 0x10058000, 0x10050000},
  /*winRegs3*/ {0x10068000, 0x10060000, 0x10078000, 0x10070000},
  /*winRegs4*/ {0x10088000, 0x10080000, 0x10098000, 0x10090000},
  /*winRegs5*/ {0x100a8000, 0x100a0000, 0x100b8000, 0x100b0000},
  /*winRegs6*/ {0x100C8000, 0x100C0000, 0x100d8000, 0x100d0000},
  /*winRegs7*/ {0x100E8000, 0x100E0000, 0x100f8000, 0x100f0000}
 };
```

The config file should contain a marker string which is considered by the `moviConvert` config parser. For the current implementation, the marker is `/*winRegs` followed by the `SHAVE` number.

The config file can also be included in the `C/C++ project` of the `main` application.

The config file may contain settings just for some `SHAVEs`. The pseudo code for parsing the config file is something like this:

- get each line from the config file
- left trim the line
- check if the line starts with marker string
- if yes, get the `SHAVE` number and parse the window values. If any error is given, ignore the entire line
- if not, process the next line

5. MVCMD configuration (.mbc) file

The `.mbc` file is a file used for configuring the `.mvcmd` (using `-mvcmdConfig:<fileName>`) boot image output file.

It is a text file which may contain the following elements:

- comments: starting with '//' sequence
- pairs of property-value considered when building the final boot image:

```
<property> = <propertyValue>
```

For `<propertyValue>`, `moviConvert` accepts the following types:

- For `bool` type : `ON`, `on`, `TRUE`, `true`, `OFF`, `off`, `FALSE`, `false`
- For `integer` type: values expressed in `decimal`, `hexadecimal` (`0x…`) and `binary` (`0b…`). Values are stored as 32-bit integers

- For `string` types: strings with/without quotes

For <property>, moviConvert supports:

- `IsFibbed(bool)` - if the chip is `Myriad ES` (fibbed)
- `LittleEndian (bool)` - little endian output
- `DsuEnable (bool)` - enable `Leon DSU`
- `Strip (bool)` - strip all strings from the output file
- `AllClocks(int)` - enable all clocks
- `Hex (int)` - encoding (8 : 8-bit, 16 : 16-bit BE, 17 : 16-bit LE, 32 : 32-bit BE, 33 : 32-bit LE)
- `ForceInitializeDDR` – always do DDR initialization
- `ForceDontInitializeDDR` – never do DDR initialization

Other options may be added to this list in the future, as new features may require new properties to be added.

Example:

```
AllClocks    = 1
IsFibbed     = ON
Strip        = ON
LittleEndian = ON
DsuEnable    = ON
Hex          = 33
```

6. Hex output configuration file (.cfg)

The `moviConvert` tool supports also `hex` output configuration files.

The configuration file is used to specify the characteristics of the memories available in the system.

Example of a piece of the `.cfg` file for Myriad 2:

```
// ROM - 32KB -
    ROM, rom, LE, 0x76000000, 2048, 128
// DRAM - 128MB
    DRAM, dram, LE, 0x80000000, 33554432, 32
// CMX - 2MB
    CMX, cmx, LE, 0x70000000, 131072, 128
```

Each valid line contains:

- MemoryName – the name of the memory – can be used multiple times for the same memory, to describe mirrored address spaces

- FileName – file name to be used for the hex file generation
- Endianness – the memory endianness
- Offset – the address map offset for the memory
- Height – number of lines, each line being LineSize bits wide (Myriad 2 ROM has 2048 lines of 128 bits per line)
- LineSize – width of memory line in bits ()
- "//" can be used to add comments to the file

7. Script commands

In fileScriptName from `-script` switch, it can be used any `TCL` specific command and one or more of the commands that are presented in the table below:

Parameter	Description
`INCLUDE_MVCMD_HEADER`	Include the `mvcmd` header.
`INCLUDE_BINARY_IF_TRUE filename $initializeDDr`	Include binary filename if initializeDDr is true.
`INCLUDE_BINARY filename`	Include binary `filename`.
`DELETE_FILE filename`	Delete the file `filename`.
`LOAD_ELF_SECTIONS filename`	Load sections from elf file `filename`.
`OUTPUT_FILE filename`	Set the output file.
`LABEL labelName:`	Define a label.
`<ENCRYPT>`	Start the encrypt block.
`</ENCRYPT>`	End the encrypt block.
`SKIP_TO_OFFSET offset`	Skip to `offset`.
`WRITE_MEM32 address value`	Write the value to address.
`ALIGN_WITH_NOPS count`	Added `count` NOPS.
`LOAD_MEM_BIN address binFileName`	Load a binary file(`binFileName`) at address.
`VECTOR_TO_ELF_ENTRY elfFile`	Set the entry point of file `elfFile`.

Table 3. Commands available in a script file