Technical Notes on using Analog Devices' DSP components and development tools

Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

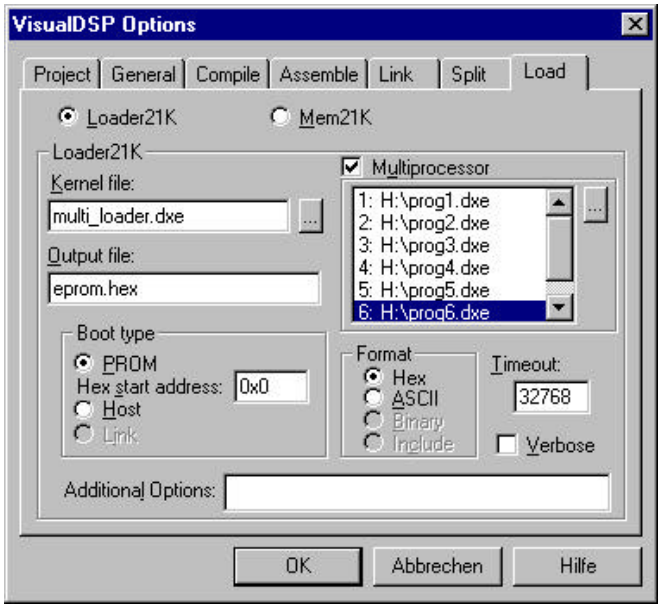
ADSP-2106x: Storing multiple Applications in a single Boot EPROM

March 23, 2000 (bk)

This paper describes how multiple (up to six) VisualDSP executables may be stored in a single Boot EPROM. A simple modified Boot Loader will decide at boot time, which executable will be loaded.

How does it work?

Well, there isn't a lot to do. VisualDSP supports a booting scheme where a cluster of up to six SHARCs can be booted from a single EPROM.



The generated hex file contains the boot loader and the boot images for the several SHARCs. Furthermore also a table is included that contains the EPROM offset address of the executables. After reset the identical boot loader will be loaded into all the SHARCs of the cluster. The original boot loader first determines the ID of the processor it is running on. From the offset table it gets the address, where in the EPROM the corresponding data is stored. Finally the boot loader will boot the application as in single-processor scheme.

The offset table consists of seven entries: Entry 0 corresponds to ID0 in a single-SHARC system. The entries 1 to 6 are used for SHARC clusters. When the Multiprocessor check box in the Loader Settings Window is disabled, only one executable can be stored in the hex file. Entry 0 for ID0 is assumed. If the check box is active, up to six executables can be served.

As explained by EE-72 this technique is used to create executables for single SHARC designs where the ID is 1.

We will take advantage of this feature and use the Multiprocessor Box to group multiple executables in an EPROM. But now something else rather than the processor ID is going to determine which application will be loaded. So the original boot loader has to be modified slightly.

Modifying the boot loader normally requires an in-deeper understanding of the boot scenario (see EE-56), but in this case this is not really necessary.

First you should create a new project to rebuild the boot loader, let's call it multi_loader. Then, please

copy the original source file (060_prom.asm or 065L_prom.asm) and the corresponding linker description file (060_ldr.ldf or 065L_ldr.ldf) from VisualDSP\21k\ldr into your project directory. You can use the 060 files for all the 2106x types except the 21065L. The original 060_ldr.ldf file selects the 21062. Feel free to change the type.

Please search in the original source file for the instructions that determines the processor's ID:

```
R0=DM(SYSTAT);
R0=FEXT R0 BY 8:3;
```

Without caring about the rest of the code, the two code lines above determine the ID stored in R0. The content of R0 decides which executable will be booted. Now, these two instructions will be replaced by any others. Just for example R0 might be controlled by a jumper connected to the flag pins. There are only two issues to take into account:

1) R0 must get a value between 1 and 6
2) The maximal length of the boot loader must not exceed 256 instructions.

The following example is for the 21061's EZ-KIT Lite. Normally this loader will boot the program #1, but if the button FLAG1 is pushed during reset program #2 will be loaded instead.

```
R0=1;
IF NOT FLAG1_IN R0=R0+1;
```

That is all! The next example is for the 21065L's EZ-Lab. The booting is controlled by the push buttons FLAG2, FLAG1 and FLAG0 here.

```
R0=ASTAT;
R0=FEXT R0 BY 16:3;
R0=R0+1;
R7=6;
R0=CLIP R0 BY R7;
```

When all the push buttons are released after reset, program #1 will be loaded. Program #2 will be loaded when button FLAG0 is pushed, etc.

Since you may modify R0 in any way you like a lot of scenarios become possible. Instead of simple checking the flag pins you can set up the SPORT or a Link Port and receive the number of the application to be booted.

```
R0=0x_____;
DM(RDIV0)=R0;
R0=0x_____;
DM(SRCTL0)=R0;
R0=DM(RX0);
```

Additionally, this technique can also be used to load the same executable into more (or all) SHARCs in a cluster. Of course, you can do that also using the original boot loader, but then you have to store the same application more times in the EPROM. Just use the instruction

```
R0=1;
```

or even

```
R0=0;
```

when the multiprocessor check box is not selected.

Furthermore you can load program #1 into SHARC ID2, ID4 and ID6 and program #2 into SHARC ID1, ID3 and ID5.

```
R7=0x01;
R0=R0 AND R7;
R0=R0+1;
```

Or you can even combine the ID with a jumper value…