Source-level debugging of ROM images using a Majic (Green Hills)

Majic along with the MULTI debugger allows for debugging binary images from ROM. To debug your ROM code (set breakpoints, single-step, etc) with source-level symbolic debug information perform the following steps:

Preparing to debug

The following steps need to be done prior to starting a debug session. Note that you need to make changes to the startice.cmd file, when you go back to debugging from RAM you need to undo these changes, **so it's a good idea to back up this file before starting**.

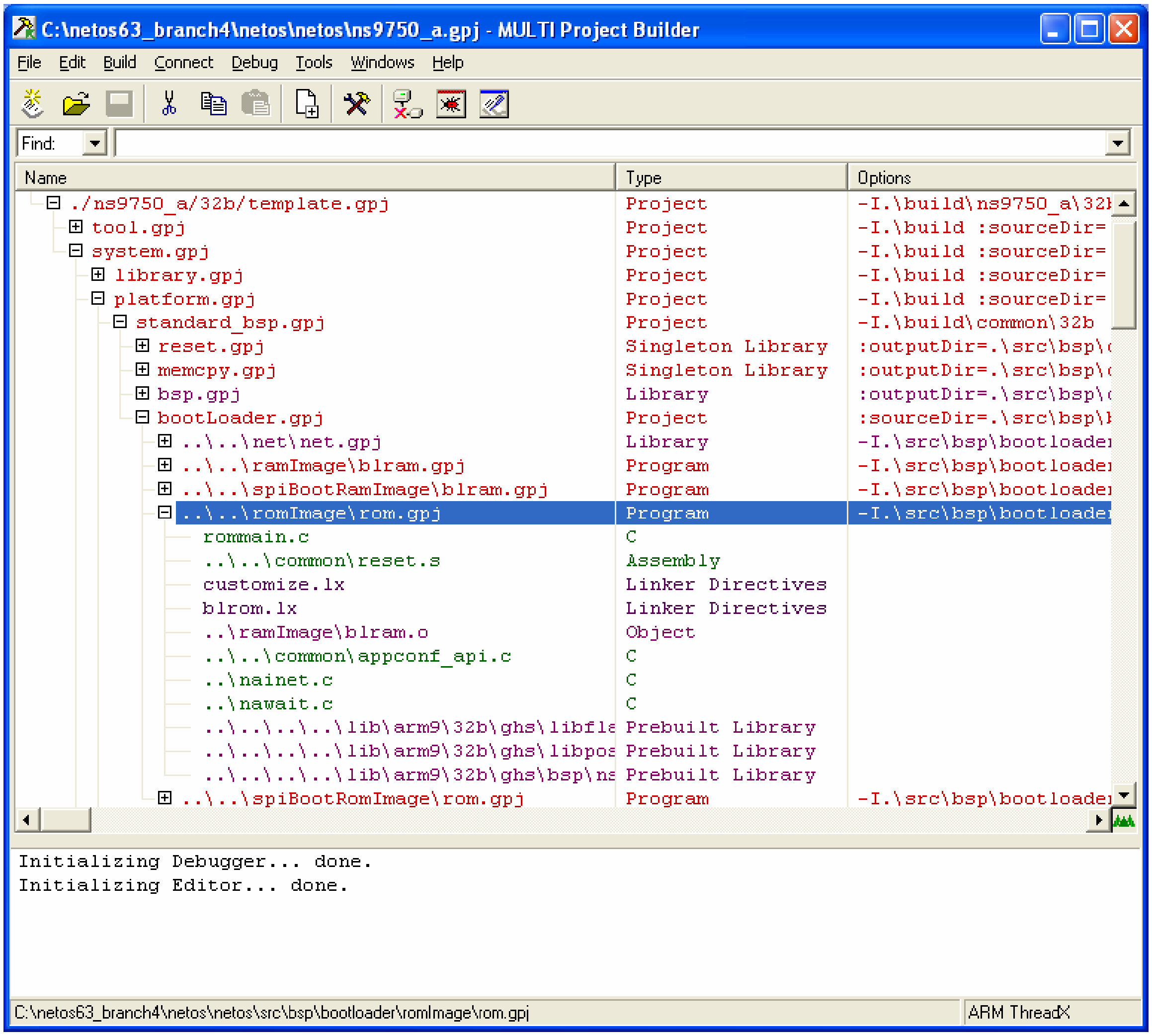
1. First build and program your image into ROM. NET+OS typically first loads a bootloader (rom.bin) and then a compressed image (image.bin), follow the normal procedure for downloading to flash.

2. Edit the startice.cmd file, when you setup the connection organizer you selected the location of this file. Comment out the line 'fr c ns9xxx.cmd' this is done by placing a '//' before this command . You are commenting out the reading of the initialization script which initializes RAM since the ROM image will now initialize RAM. This line is shown below:

```
// fr c ns9xxx.cmd
```

Debug Session

1. Power the board ON.
2. Start MULTI as normal and navigate to your application (if you are using netcentral). If you want to debug the bootloader, navigate into the bootloader directory as shown below:

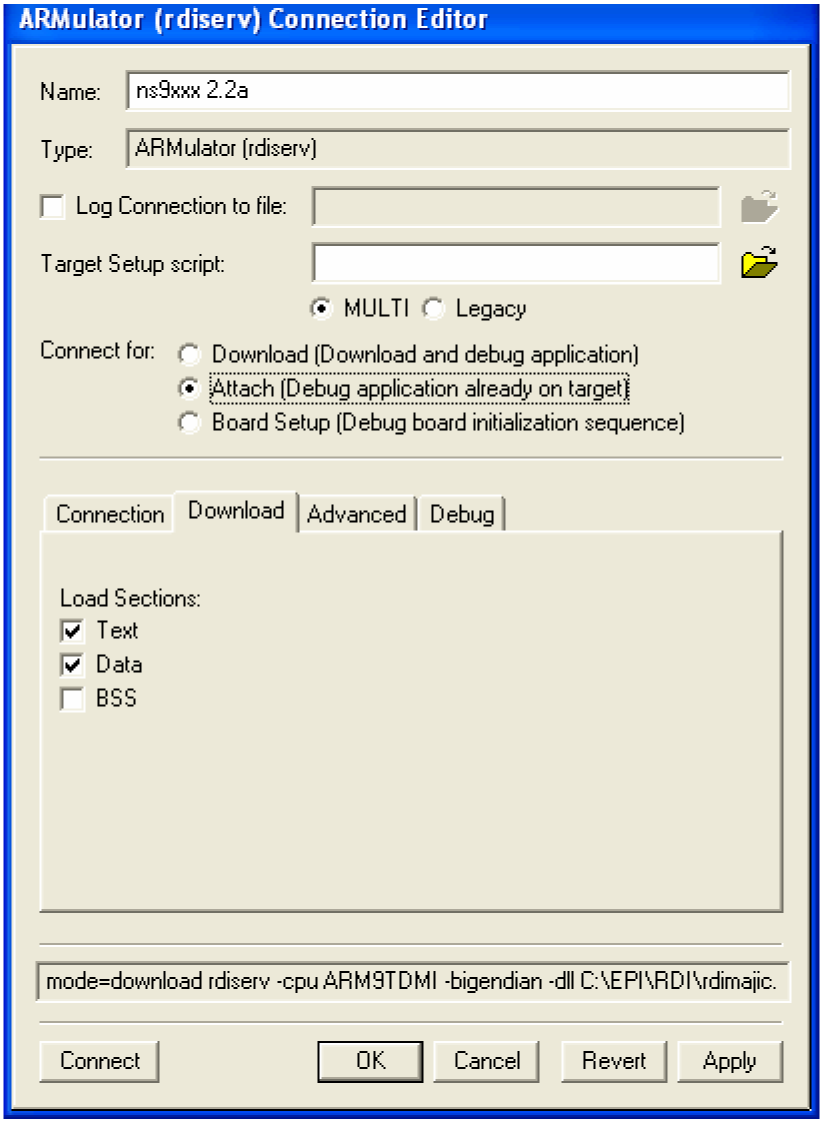

C:\netos63_branch4\netos\netos\ns9750_a.gpj - MULTI Project Builder
File Edit Build Connect Debug Tools Windows Help
Find:

Name	Type	Options
./ns9750_a/32b/template.gpj	Project	-I.\build\ns9750_a\32
tool.gpj	Project	-I.\build :sourceDir=
system.gpj	Project	-I.\build :sourceDir=
library.gpj	Project	-I.\build :sourceDir=
platform.gpj	Project	-I.\build :sourceDir=
standard_bsp.gpj	Project	-I.\build\common\32b
reset.gpj	Singleton Library	:outputDir=.\src\bsp\
memcpy.gpj	Singleton Library	:outputDir=.\src\bsp\
bsp.gpj	Library	:outputDir=.\src\bsp\
bootLoader.gpj	Project	:sourceDir=.\src\bsp\
..\..\net\net.gpj	Library	-I.\src\bsp\bootloade
..\..\ramImage\blram.gpj	Program	-I.\src\bsp\bootloade
..\..\spiBootRamImage\blram.gpj	Program	-I.\src\bsp\bootloade
..\..\romImage\rom.gpj	Program	-I.\src\bsp\bootloade
rommain.c	C	
..\..\common\reset.s	Assembly	
customize.lx	Linker Directives	
blrom.lx	Linker Directives	
..\ramImage\blram.o	Object	
..\..\common\appconf_api.c	C	
..\nainet.c	C	
..\nawait.c	C	
..\..\..\..\..\lib\arm9\32b\ghs\libfla	Prebuilt Library	
..\..\..\..\..\lib\arm9\32b\ghs\libpos	Prebuilt Library	
..\..\..\..\..\lib\arm9\32b\ghs\bsp\na	Prebuilt Library	
..\..\spiBootRomImage\rom.gpj	Program	-I.\src\bsp\bootloade

```
Initializing Debugger... done.
Initializing Editor... done.
```

C:\netos63_branch4\netos\netos\src\bsp\bootloader\romImage\rom.gpj
ARM ThreadX

3. Go into the connection organizer by clicking on Connect->Connection Organizer.
4. In the ‘User Methods’ window double click on the connection which you setup in the tutorial.
5. In the ‘Connect for:’ list click on the ‘Attach (Debug application already on target)’ button as shown below.



ARMulator (rdiserv) Connection Editor

Name: ns9xxx 2.2a

Type: ARMulator (rdiserv)

□ Log Connection to file:

Target Setup script:

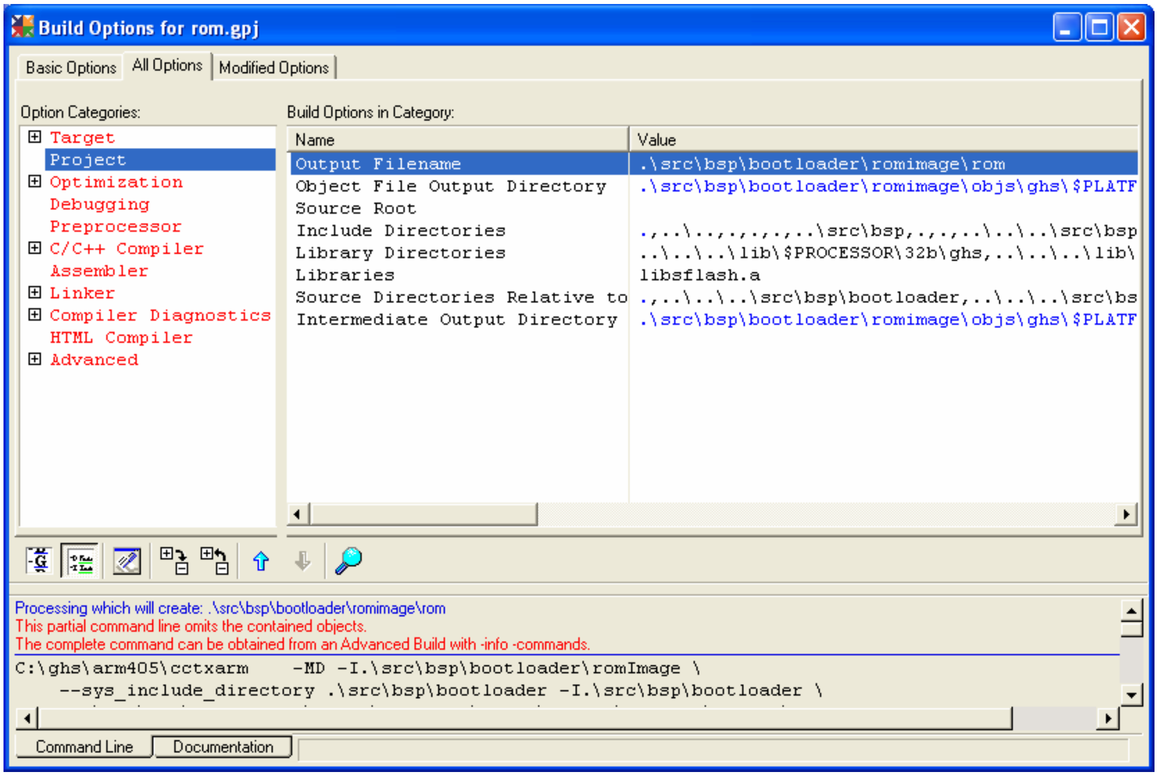
● MULTI ○ Legacy

Connect for:
○ Download (Download and debug application)
● Attach (Debug application already on target)
○ Board Setup (Debug board initialization sequence)

Load Sections:
☑ Text
☑ Data
□ BSS

mode=download rdiserv -cpu ARM9TDMI -bigendian -dll C:\EPI\RDI\rdimajic.

6. Click OK
7. Right mouse click on your user method and select Connect to Target, you should see the RDI server window.
8. Load the symbol file for the image you wish to debug; symbols are in the output file for your project. You see the name of the output file for your project by right mouse clicking on your project (rom.gpj) and select 'Set Options' as shown below:



Name	Value
Output Filename	.\src\bsp\bootloader\romimage\rom
Object File Output Directory	.\src\bsp\bootloader\romimage\objs\ghs\$PLATF
Source Root	
Include Directories	.,..\..,.,.,.,..\src\bsp,.,.,..\..\..\src\bsp
Library Directories	..\..\..\lib\$PROCESSOR\32b\ghs,..\..\..\lib\
Libraries	libsflash.a
Source Directories Relative to	.,..\..\..\src\bsp\bootloader,..\..\..\src\bs
Intermediate Output Directory	.\src\bsp\bootloader\romimage\objs\ghs\$PLATF

```
Processing which will create: .\src\bsp\bootloader\romimage\rom
This partial command line omits the contained objects.
The complete command can be obtained from an Advanced Build with -info -commands.
C:\ghs\arm405\cctxarm    -MD -I.\src\bsp\bootloader\romImage \
    --sys_include_directory .\src\bsp\bootloader -I.\src\bsp\bootloader \
```

9. To load the symbols click on

Debug->Load Symbols->Select Symbols To Load

10. Select your rom file (output files) and click ‘Load’ You should see the following after you click load:

Loading symbols for
C:\netos63_branch4\netos\netos\src\bsp\bootloader\romImage\rom.
Symbols loaded.
PC
0

12. Next set the program counter by typing
$pc=0x50000000

Now you can start stepping through ROM code. ***You can only set two breakpoints at a time***. To break on main, type ‘b main’ as you normally would, this gets converted automatically to a hardware breakpoint. The rom file is the ROM image portion of the bootloader, if you wish to debug the RAM image portion of the bootloader or the RAM image application you will need to reload symbols for that image.

Tips

1. Only have two breakpoints at a time, ARM only supports two hardware breakpoints.
2. If you have optimization turned ON, you can expect to see the program counter ‘jump around’ while single stepping, to disable optimization right mouse click on your project select ‘Set Options’ and set the Optimization Strategy to ‘No Optimizations’ and rebuild your application.
3. Make sure that you exit the RDI Majic console window between debug sessions, it does not always shutdown gracefully, and if it’s still open it won’t allow you to start another debug session.
4. Also, make sure you set the program counter to 0x50000000 before you start stepping.
5. Make sure that flash is enabled.