

Using ROM Libraries

A ROM library is an absolute file, containing code, which is not directly executable, but which can be used by an application. Objects (functions and variables) within a ROM library are assigned an absolute address.


A ROM library file can be loaded separately in the debugger or even burnt separately into an EPROM or in Flash.

ROM Library support is fully integrated in CD CW 8-16 V1.0 and higher (Smart Linker V5.0.15 and higher).

Defining ROM Library Project

Project Settings

In order to create a project for a ROM Library:

- Use a generic stationery for the appropriate CPU, when you create your ROM Library project. As the ROM Library does not generate an executable file, there is no need to configure (or set up) a debugger.
- Open the project setting dialog clicking on the  icon in the project window tool bar.
- Select the Linker panel.
- Click on the “Options” button and select the “Output” Tab.
- Check the option “Link as ROM Library” (option –AsROMLib).
- Close the “SmartLinker Option Settings” dialog clicking on “OK”.
- Close the project setting dialog clicking on “OK”.

ROM Library PRM File

When the ENTRIES command is present in a ROM Library PRM file, the objects enumerated there are considered to be entry points in the ROM library. Only these objects and the objects they refer to are inserted in the ROM library.

Example:

```
NAMES END

ENTRIES
  Read ReadString Write WriteLn WriteString  InitTerminal
END

SECTIONS
  MY_RAM = READ_WRITE 0x1000 TO 0x1BFF;
  MY_ROM = READ_ONLY  0xC000 TO 0xCFFF;
PLACEMENT
  DEFAULT_ROM  INTO MY_ROM;
  DEFAULT_RAM  INTO MY_RAM;
END
```

When the ENTRIES command is not present in a ROM Library PRM file, all the objects located in the files enumerated in the NAMES command is inserted in the ROM library.



Example:

```
NAMES END

SECTIONS
    MY_RAM = READ_WRITE 0x1000 TO 0x1BFF;
    MY_ROM = READ_ONLY 0x5000 TO 0xCFFF;
PLACEMENT
    DEFAULT_ROM INTO MY_ROM;
    DEFAULT_RAM INTO MY_RAM;
END
```

Be Careful:

If there is no ENTRY command in the PRM file, and if the ANSI library file is specified in the NAMES block, all the functions and variables defined in this library is inserted in the ROM library. This requires a large amount of RAM and ROM area.

ROM Library Startup File

As a ROM library cannot be executed alone, it does not require any stack initial value, startup or main function. For initialization purpose, a ROM library may require a startup structure. In that case the startup file for a ROM library only consist in the definition of a startup structure.

Example for HC12:

```
/*
    HC12 Small and medium memory model;
    standard startup-structure definition for a ROM library.
-----*/

#include "start12.h"

/*
struct _tagStartup _startupData; // read-only:
                                // _startupData is allocated in ROM and
                                // initialized by the linker
*/
```

Note:

For the HC11 and HC12 processor in banked memory model, the same rules apply for ROM libraries and user application:

- The predefined section STRINGS, ROM_VAR, NON_BANKED, COPY must always be located in the non banked ROM area.

Defining Project using a ROM Library

Project Setting

In order to create a project using a ROM Library:

- Use a stationery for the appropriate CPU and target interface, when you create your project.
- In the Project menu select “Add Files” and add the ROM library MCP file to your project. This way the ROM Library is considered as a sub-project from your application.

Application PRM File

There should be no overlap between the memory areas defined in the ROM library PRM file and the one specified in the application PRM file.

Example:

```

NAMES  END

SECTIONS
  MY_RAM    = READ_WRITE 0x01C00 TO 0x01FFF;
  MY_ROM_1  = READ_ONLY  0x18000 TO 0x18FFF;
  MY_ROM_0  = READ_ONLY  0x0D000 TO 0x0DFFF;
PLACEMENT
  DEFAULT_ROM          INTO  MY_ROM_1;
  _PRESTART, STARTUP
  STRINGS, ROM_VAR,
  NON_BANKED, COPY INTO  MY_ROM_0;
  DEFAULT_RAM         INTO  MY_RAM;
END

VECTOR ADDRESS 0xFFFE _Startup /* set reset vector on _Startup */
STACKSIZE 0x60
  
```

It is the user responsibility to ensure, that the code and data memory area of the application and the ROM library does not overlap.

Note:

For CPU using an OVERLAP segment for parameter passing or local variables (HC05, ST7), you have to define a separate OVERLAP segment for the application and the ROM Library.

Application Startup File

If you use the standard startup code within the application, the global variables from the ROM library may not be initialized appropriately. Usually we recommend you to initialize all global variables within the ROM library using assignment in the source code.

If you choose to have your ROM library global variables initialized during startup, make sure to insert the appropriate startup in your project. In fact look fore a startup file, where the startup-structure field libInits is processed.

Following table shows which startup files can be used to initialize ROM library variables during application startup.

| CPU | Startup files |
|------|---------------|
| HC05 | Startup.c |
| HC08 | Startup.c |



| | |
|------|--|
| HC11 | Startup.c |
| HC16 | Startup.c, Strtroml.c, Strtromm.c, Strtroms.c, |
| ST7 | Startup.c |
| XA | Startup.c |

Debugging a Project using a ROM Library

When you are debugging a project using a ROM library, make sure that both the ROM library and the project are loaded in the target system. The ROM library may be resident in ROM or Flash or loaded separately.

When you are debugging an application using a ROM library, the CodeWarrior debugger can only have one symbol table loaded at a time.

Usually after loading from the application, symbolic debugging is available on the application only. If you intend to perform symbolic debugging on the ROM library:

- In the debugger select Load Application
- In the Load Executable File dialog, select the "Load Symbols only" option button.
- Browse for your ROM library ABS file and click OK. Now symbolic debugging is available for the ROM library.
- If you want to activate symbolic debugging on the application again, perform a Load Symbols for the application ABS file.