RP2040 CPU Support Package Guide

Version: 4.2

Contents

RP2040 Support Package

This guide describes the following features of the RP2040 CPU support package:

Creating RP2040 Projects

Creating an RP2040 C/C++ executable project

To create a new minimal C/C++ executable project:

Select the **File** > **New** > **New Project** menu item.
Select the **A C/C++ executable for Raspberry Pi Foundation RP2040** project template.
Set the required project name and location directory.
Click **Next**.
If required, change any of the default project settings.
Click **Finish** to create the project.

Creating an RP2040 library project

To create a new library project:

Select the **File** > **New** > **New Project** menu item.
Select the **A library for Raspberry Pi Foundation RP2040** project template.
Set the required project name and location directory.
Click **Next**.
If required, change any of the default project settings.
Click **Finish** to create the project.

Creating an RP2040 externally built executable project

To create a new project that will allow you to debug an existing externally built executable file:

Select the **File** > **New** > **New Project** menu item.
Select the **An externally built executable for Raspberry Pi Foundation RP2040** project template.
Set the required project name and location directory.
Click **Next**.
Set the **Load File** project property to point to the executable file you want to download and debug.
If required, change any of the other default project settings.
Click **Finish** to create the project.

Creating an RP2040 CrossWorks Tasking Library executable project

To create a new C/C++ executable project configured to use the CrossWorks Tasking Library:

Select the **File** > **New** > **New Project** menu item.
Select the **A CrossWorks Tasking Library executable for Raspberry Pi Foundation RP2040** project template.
Set the required project name and location directory.

Click **Next**.
If required, change any of the other default project settings.
Click **Finish** to create the project.

Creating an RP2040 assembly code only executable project

Please note, this template does not add any C/C++ startup code or libraries and is therefore not suitable for creating projects that include C/C++ code.

To create a new assembly code only executable project without:

Select the **File** > **New** > **New Project** menu item.
Select the **An assembly code only executable for Raspberry Pi Foundation RP2040** project template.
Set the required project name and location directory.
Click **Next**.
If required, change any of the other default project settings.
Click **Finish** to create the project.

Opening RP2040 Sample Solutions

RP2040 Samples Solution

This solution contains general sample projects that run on RP2040 devices. To open the RP2040 Samples Solution:

Select the **Tools** > **Show Installed Packages** menu item.
Select the **Raspberry Pi Foundation RP2040 CPU Support Package** link.
Select the **Samples Solutions** > **RP2040 Samples Solution** link.

RP2040 CMSIS-DSP Samples Solution

This solution contains sample projects that use the CMSIS-DSP library running on RP2040 devices. To open the RP2040 CMSIS-DSP Samples Solution:

Select the **Tools** > **Show Installed Packages** menu item.
Select the **Raspberry Pi Foundation RP2040 CPU Support Package** link.
Select the **Sample Solutions** > **RP2040 CMSIS-DSP Samples Solution** link.

RP2040 Project Properties

Projects creating using the project templates in this support package have the following device specific project properties:

Heap Size

The heap size is set to be 256 bytes when a project is created. The heap size can be modified using the **Heap Size** project property.

Section Placement

You can select the memory configuration you require using the **Section Placement** project property.

For RP2040 projects, the set of placements are:

Flash - The application runs in internal Flash memory *(default)*.
Flash Vectors In RAM - The application runs in internal Flash memory and exception vectors are copied to RAM memory.
Flash Copy To RAM - The application starts in internal flash and copies itself to run from internal RAM memory.
RAM - The application runs from internal RAM memory only.

Stack Sizes

The main stack size is set to be 256 bytes when a project is created.

The process stack size is set to be 0 bytes when a project is created.

The main and process stack sizes can be modified using the **Main Stack Size** and **Process Stack Size** project properties.

To change the location of the stacks, edit the section placement file and place the *.stack* and *.stack_process* sections as required.

Target Processor

Once a project has been created you can target different devices by modifying the **Target Processor** project property. See the **RP2040 Devices** section for details on the files, preprocessor definitions and macro definitions used when a device is selected.

Second Stage Bootloader

The second stage bootloader can be selected using the **Code** > **Build** > **Second Stage Bootloader** project property. The property can be set to one of the following:

Adesto AT25SF128A

Generic 03h
ISSI IS25LP080D
Winbond W25Q080 *(default)*
Winbond W25X10CL
None

RP2040 Project Templates

The project template system simplifies the creation of new projects with the IDE, it also system makes it easy to create new projects with a text editor or script. All that needs to be specified is the project name, the support packages that the project is dependent on, the target processor and the source files you want to add to the project. For example, create a file called *example.hzp* with the following contents:

```
<!DOCTYPE CrossStudio_Project_File>
<solution Name="Example Solution">
  <project Name="Example Project" template_name="RP2040_EXE">
    <configuration Name="Common" package_dependencies="RP2040" Target="RP2040_M0_0" />
    <folder Name="Source Files">
      <file file_name="file1.c" />
      <file file_name="file2.c" />
    </folder>
  </project>
</solution>
```

You can also add any other property settings that the project requires such as preprocessor definitions or include paths using the property save name, for example:

```
<!DOCTYPE CrossStudio_Project_File>
<solution Name="Example Solution">
  <project Name="Example Project" template_name="RP2040_EXE">
    <configuration Name="Common" package_dependencies="RP2040" Target="RP2040_M0_0"
 c_preprocessor_definitions="MYDEF1=1;MYDEF2=TWO" c_user_include_directories="$(ProjectDir)/
include1;$(ProjectDir)/include2" />
    <folder Name="Source Files">
      <file file_name="file1.c" />
      <file file_name="file2.c" />
    </folder>
  </project>
</solution>
```

Available RP2040 project templates

Template Name	Template Description
RP2040_ASM_EXE	RP2040 Assembly Code Only Executable
RP2040_CTL_EXE	RP2040 CTL Executable
RP2040_EXE	RP2040 C/C++ Executable
RP2040_EXT_EXE	RP2040 Externally Built Executable
RP2040_LIB	RP2040 Library

RP2040 Devices

This package supports the following RP2040 devices:

- **RP2040_M0_0**
- **RP2040_M0_1**

RP2040_M0_0

Device Details	
CMSIS Header File	$(TargetsDir)/RP2040/CMSIS/Device/Include/RP2040.h
CMSIS Include Path	$(TargetsDir)/RP2040/CMSIS/Device/Include
CMSIS System File	$(TargetsDir)/RP2040/CMSIS/Device/Source/system_RP2040.c
Family	RP2040
Loader File	$(TargetsDir)/RP2040/Loader/RP2040_Loader.elf
Memory Map File	$(TargetsDir)/RP2040/XML/RP2040_M0_0_MemoryMap.xml
Register Definition File	$(TargetsDir)/RP2040/XML/rp2040_Registers.xml
Vectors File	$(TargetsDir)/RP2040/Source/rp2040_Vectors.s

Preprocessor Definitions
ARM_MATH_CM0PLUS
RP2040_M0_0
__RP2040_FAMILY

Memory Segments	
FLASH	0x10000000 - 0x10FFFFFF
RAM	0x20000000 - 0x20041FFF

Project Macros
DeviceIncludePath=$(TargetsDir)/RP2040/CMSIS/Device/Include
DeviceHeaderFile=$(TargetsDir)/RP2040/CMSIS/Device/Include/RP2040.h
DeviceLoaderFile=$(TargetsDir)/RP2040/Loader/RP2040_Loader.elf
DeviceRegisterDefinitionFile=$(TargetsDir)/RP2040/XML/rp2040_Registers.xml
DeviceSystemFile=$(TargetsDir)/RP2040/CMSIS/Device/Source/system_RP2040.c
DeviceVectorsFile=$(TargetsDir)/RP2040/Source/rp2040_Vectors.s
DeviceFamily=RP2040

RP2040_M0_1

Device Details	
CMSIS Header File	$(TargetsDir)/RP2040/CMSIS/Device/Include/RP2040.h
CMSIS Include Path	$(TargetsDir)/RP2040/CMSIS/Device/Include
CMSIS System File	$(TargetsDir)/RP2040/CMSIS/Device/Source/system_RP2040.c
Family	RP2040
Loader File	$(TargetsDir)/RP2040/Loader/RP2040_Loader.elf
Memory Map File	$(TargetsDir)/RP2040/XML/RP2040_M0_1_MemoryMap.xml
Register Definition File	$(TargetsDir)/RP2040/XML/rp2040_Registers.xml
Vectors File	$(TargetsDir)/RP2040/Source/rp2040_Vectors.s

Preprocessor Definitions
ARM_MATH_CM0PLUS
RP2040_M0_1
__RP2040_FAMILY

Memory Segments	
FLASH	0x10000000 - 0x10FFFFFF
RAM	0x20000000 - 0x20041FFF

Project Macros
DeviceIncludePath=$(TargetsDir)/RP2040/CMSIS/Device/Include
DeviceHeaderFile=$(TargetsDir)/RP2040/CMSIS/Device/Include/RP2040.h
DeviceLoaderFile=$(TargetsDir)/RP2040/Loader/RP2040_Loader.elf
DeviceRegisterDefinitionFile=$(TargetsDir)/RP2040/XML/rp2040_Registers.xml
DeviceSystemFile=$(TargetsDir)/RP2040/CMSIS/Device/Source/system_RP2040.c
DeviceVectorsFile=$(TargetsDir)/RP2040/Source/rp2040_Vectors.s
DeviceFamily=RP2040