器件支持包

一个包含**<devices>**元素的软件包称作设备家族包（DFP）。一个 DFP 可以包含额外的**软元件**或者**工程示例**或者其任意组合。典型的 DFP 内容有：

- 解释设备或者设备系列功能的**设备属性**。
- 配置设备的软元件和基本设备驱动程序，典型的有：
  - 需要用来设置 C 运行时库，器件时钟和存储器接口的 CMSIS 兼容的系统和启动文件。
  - 提供给中间件栈使用的物理器件外设的软件例行程序对应的外设驱动程序接口
- 程序员视角的用来描述器件外设的一个或多个系统视图描述文件。驱动程序可遵守 **CMSIS-Driver** 标准。
- 擦除和下载代码到片上闪存用的 **Flash 编程算法**。
- 显示器件及其外设的用法的工程示例。
- 可以用来作为应用开发起点的用户代码模板。

该节是一个说明如何创建 DFP 的教程。起初，在 DFP 中只有描述器件的 PDSC（Package description——包描述）文件。然后向该 DFP 中扩充添加 SVD（System View Description——系统视图描述）文件、Flash 算法和器件相关的如系统和 HAL（Hardware Abstraction Layer——硬件抽象层）的软元件文件。示例项目和代码模板可作为描述节添加到软元件包中。

DFP 使用案例

为支持新的器件，**器件系列包（DFP）**可以作为开发工具的扩展由芯片供应商提供。相对于某一器件系列，DFP 使芯片供应商能独立的分发器件支持工具。

DFP 也能被用来提供显示相关网站的信息。一个例子是 **www.keil.com/dd2/**上的新设备数据库：

Home / MDK5 Device List / SN32F235J

SONiX SN32F235J

ARM Cortex-M0, 50 MHz, 4 kB RAM, 32 kB ROM

SN32F2 series 32-bit micro-controller is a new series of extremely Low Power Consumption and High Performance MCU powered by ARM Cortex M0 processor with Flash ROM architecture.

- **Core** ARM Cortex-M0 50 MHz
- **Timer/Counter/PWM** 3 x 16-bit Timer, 3 x 32-bit Timer
- **I/O & Package** 33-QFN
  - External Interrupts: 4 External Interrupts
  - I/Os: 23 Inputs/Outputs
  - Package, QFN: 33-pad QFN
- **Other**
- **Communication** SPI, I2C, USB, Device, UART
- **Clock & Power** 1.80 V .. 3.60 V, 50 MHz
- **Analog** 4-channel 12-bit ADC
- **Memory** 4 kB RAM, 32 kB ROM

Quick Links

- Board List
- Software Packs
- MDK Version 5
- Legacy Support
- Feedback

Silicon Supplier

- SONiX
- Distributors

Device Family Pack DFP

Support for this device is contained in:

SONiX SN32F2 Series Device Support and Examples

Download

Development Tools

- MDK Version 5 (Microcontroller Development Kit)
- MDK-Lite Download

从 **DFP** 中提取网站上的设备信息

创建一个 **DFP** 的步骤

基本器件系列包

在下面的章节中，将为设备供应商 **Myvendor** 提供的称为 **MVCM3** 的一个虚构的器件系列创建 DFP。器件系列由被分成两个子系列的四个成员组成。所述 MVCM3 系列的规格如下：

MVCM3			
描述：MVCM3器件系列包含一个ARM Cortes-M3处理器，运行频率可达到100MHz以及多种的片上外设。			
处理器		ARM Cortes-M3处理器（r2p1版），小端	
内存保护单元		无	
浮点运算单元		无	
外部中断		16	
运行温度范围		-40℃～+105℃（扩展级温度）	
运行电压		+2.5V～+3.6V	
实时时钟		32.768kHz	
看门狗定时器		1	
MVCM3100		**MVCM3200**	
MVCM3100子系列最高运行频率为50MHz。		MVCM3200子系列最高运行频率为100MHz。	
I/O数量	26	I/O数量	38
I^2C	1	I^2C	3
UART	4	UART	5
Timer/Counter	6×32-bit	Timer/Counter	8×32-bit
		PWM	4×16-bit
封装	32-Pin LQFP	封装	48-Pin LQFP
MVCM3110特征		**MVCM3250特征**	
RAM	2 kB SRAM	RAM	2 kB SRAM
Flash	16 kB	Flash	16 kB
PWM	2×16-bit		
MVCM3120特征		**MVCM3260特征**	
RAM	4 kB SRAM	RAM	4 kB SRAM
Flash	32 kB	Flash	32 kB
PWM	4×16-bit		

MVCM3 器件系列规格

准备工作

1. 在您的 PC 上创建一个工作目录，例如 **C:\temp\working_dfp**。
2. 进入可用的 **ARM::CMSIS** 包安装目录的**\CMSIS\Pack\Tutorials**。请查阅您的开发工具的文档来获取包安装目录结构的详细信息。在 µVision 中，它在目录 **C:\Keil\ARM\Pack\ARM\CMSIS\version** 下。
3. 打开文件 **Pack_with_Device_Support.zip**。
4. 将该 ZIP 文件中的 **01_Basic_Pack** 目录复制到您的工作目录。
5. 确保文件或目录没有被写保护（删除只读标志）。
6. 从可用的 **ARM::CMSIS** 包安装目录的**\CMSIS\Utilities** 下复制如下文件到您的工作目录中：
   - PackChk.exe
   - PACK.xsd
   - SVDConv.exe
7. 用一个编辑器打开文件 **MyVendor.MVCM3.pdsc**。

代码示例

1. 取消 PDSC 文件中对**<devices>**部分的注释，并添加以下内容：

```
<family Dfamily="MVCM3 Series" Dvendor="Generic:5">
<processor Dcore="Cortex-M3" DcoreVersion="r2p1" Dfpu="0" Dmpu="0"
Dendian="Little-endian"/>
<description>
The MVCM3 device family contains an ARM Cortex-M3 processor, running up
to 100 MHz with a versatile set of on-chip peripherals.
</description>
<!-- ************************ Subfamily 'MVCM3100'
*************************** -->
<subFamily DsubFamily="MVCM3100">
<processor Dclock="50000000"/>
<!-- ************************ Device 'MVCM3110'
*************************** -->
<device Dname="MVCM3110">
<memory id="IROM1" start="0x00000000" size="0x4000" startup="1"
default="1"/>
<memory id="IRAM1" start="0x20000000" size="0x0800" default="1"/>
</device>
<!-- ************************ Device 'MVCM3120'
*************************** -->
<device Dname="MVCM3120">
<memory id="IROM1" start="0x00000000" size="0x8000" startup="1"
default="1"/>
<memory id="IRAM1" start="0x20000000" size="0x1000" default="1"/>
</device>
</subFamily>
<!-- ************************ Subfamily 'MVCM3200'
*************************** -->
<subFamily DsubFamily="MVCM3200">
<processor Dclock="100000000"/>
<!-- ************************ Device 'MVCM3250'
*************************** -->
<device Dname="MVCM3250">
<memory id="IROM1" start="0x00000000" size="0x4000" startup="1"
default="1"/>
<memory id="IRAM1" start="0x20000000" size="0x0800" default="1"/>
</device>
<!-- ************************ Device 'MVCM3260'
*************************** -->
<device Dname="MVCM3260">
```

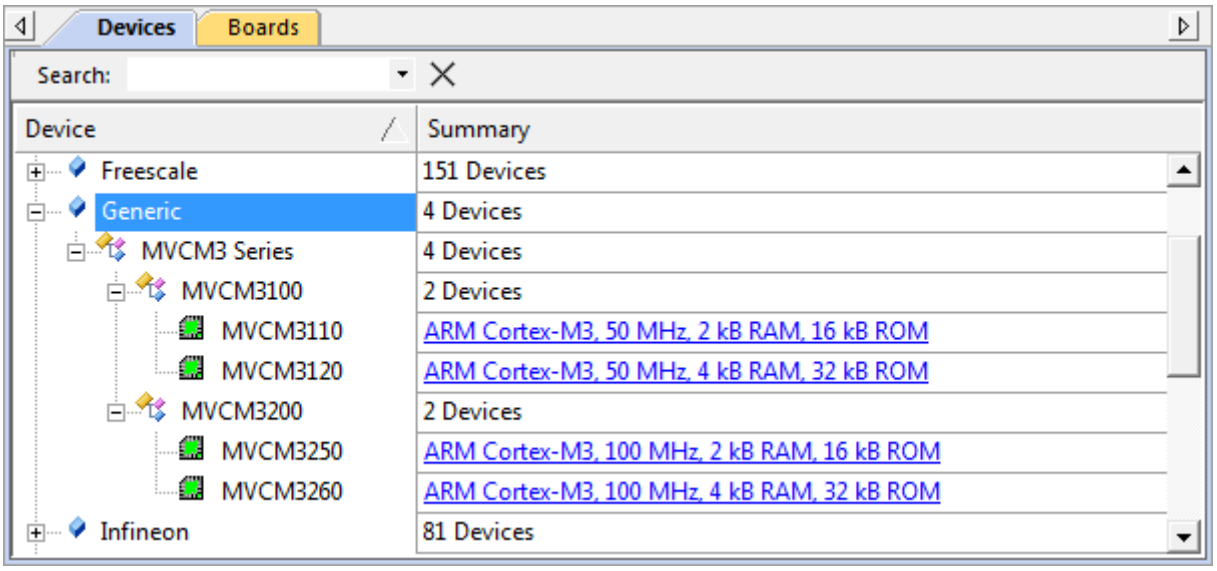
```
<memory id="IROM1" start="0x00000000" size="0x8000" startup="1"
default="1"/>
<memory id="IRAM1" start="0x20000000" size="0x1000" default="1"/>
</device>
</subFamily>
</family>
```

注意

Dvendor ID 不能自由选择。此处设置的 ID 可以在 PACK.xsd 中找到（可用的 **ARM::CMSIS** 包安装目录的 **CMSIS\Utilities** 中）。

本节以及下面各节的所有的代码示例可以在 **01_Basic_Pack** 目录下的 snippets.xml 文件中找到。

2. 保存 PDSC 文件并用 **gen_pack.bat** 脚本生成包文件。参见 **Generate a Pack** 以了解进一步的细节。然后安装这个包到您的开发工具中。

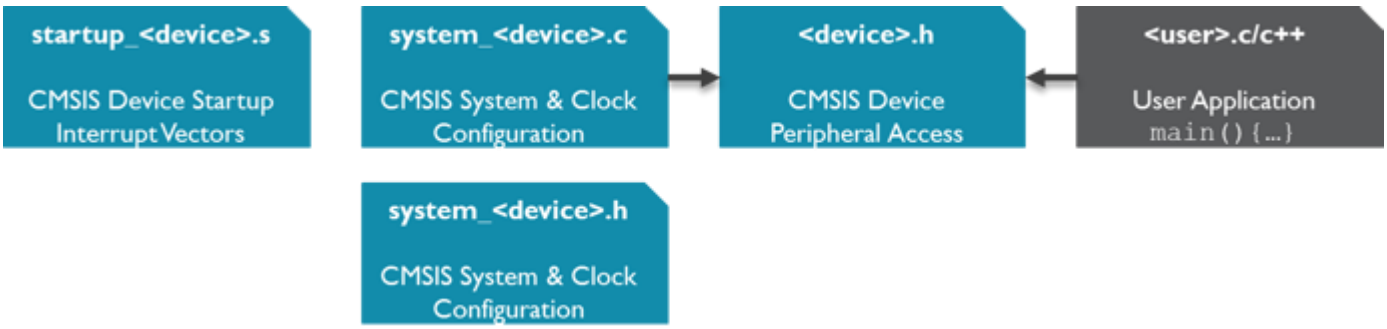


在开发工具中显示微控制器器件

系统和启动文件

CMSIS-CORE 定义了如下要在嵌入式应用中使用的文件：

- startup_<device>.s 包含复位处理程序和异常向量。其在复位之后执行，然后调用 SystemInit，且可能包含用户应用程序的堆栈配置。
- system_<device>.c 和 system_<device>.h 包含通用的系统配置信息（如时钟和总线设置）。
- <device.h> 提供了可访问的处理器核心和所有外设。该文件应该由 **SVD** 文件通过 **SVDConv.exe** 产生，以此来确保头文件和调试器显示的一致性。



与用户代码有关的系统和启动文件

> **注意**
>
> CMSIS-CORE 阐明了系统和启动文件的结构以及其创建方式。

复制 **Pack_with_Device_Support.zip** 中的 **02_System_and_Startup** 目录到您的工作环境中的**文件**目录中。

1. 取消 PDSC 文件中对<conditions>部分的注释，并添加以下内容（该 **conditions** 部分提供了该步骤的详细信息）：

```
<condition id="MVCM3 CMSIS-CORE">
<!-- conditions selecting Devices -->
<description>MyVendor MVCM3 Series devices and CMSIS-CORE</description>
<require Cclass="CMSIS" Cgroup="CORE"/>
<require Dvendor="Generic:5" Dname="MVCM3*"/>
</condition>
<condition id="Startup ARM">
<description>Startup assembler file for ARMCC</description>
<require Tcompiler="ARMCC"/>
</condition>
<condition id="Startup GCC">
<description>Startup assembler file for GCC</description>
<require Tcompiler="GCC"/>
</condition>
<condition id="Startup IAR">
<description>Startup assembler file for IAR</description>
<require Tcompiler="IAR"/>
</condition>
```

> **注意**
>
> 基于汇编的 startup_<device>.s 文件是和工具有关的。因此，对于各工具供应商，单独的条件是必需的。

2. 取消 PDSC 文件中对<components>部分的注释，并添加以下内容（该 **components** 部分提供了该步骤的详细信息）：

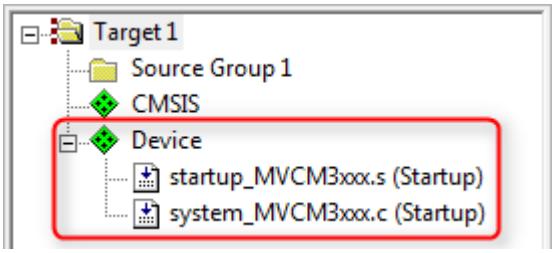
```
<component Cclass="Device" Cgroup="Startup" Cversion="1.0.0"
condition="MVCM3 CMSIS-CORE">
```

```
<description>System Startup for MyVendor MVCM3 Series</description>
<files>
<!-- include folder -->
<file category="include" name="Device\Include\"/>
<file category="source" name="Device\Source\ARM\startup_MVCM3xxx.s"
attr="config" condition="Startup ARM"/>
<file category="source" name="Device\Source\GCC\startup_MVCM3xxx.s"
attr="config" condition="Startup GCC"/>
<file category="source" name="Device\Source\IAR\startup_MVCM3xxx.s"
attr="config" condition="Startup IAR"/>
<file category="source" name="Device\Source\system_MVCM3xxx.c"
attr="config" />
</files>
</component>
```

3. 添加一个新的版本号：

```
<release version="1.0.1">
Startup files included
</release>
```

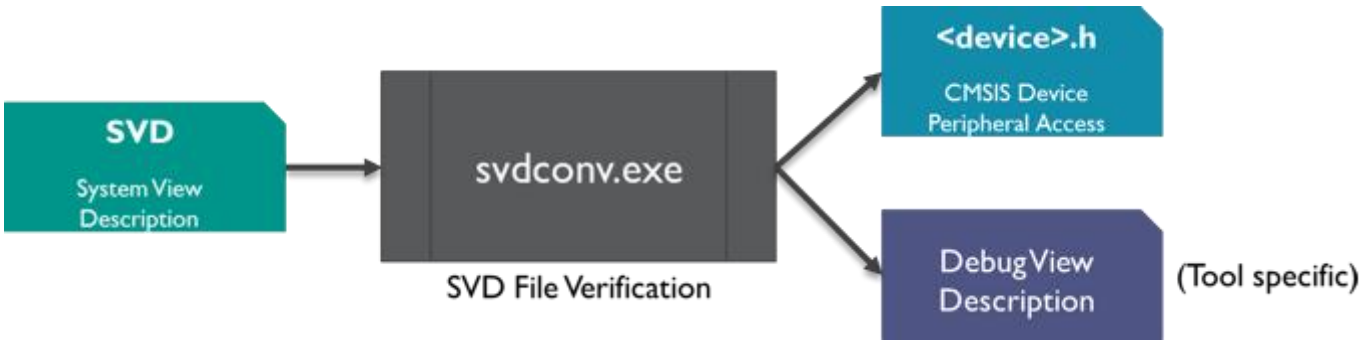
4. 最后，保存 PDSC 文件并用 **gen_pack.bat** 脚本生成包文件。然后安装这个包到您的开发工具中并创建一个新的工程。选择软元件**::CMSIS:CORE** 和**::Device:Startup** 给该工程：



启动和系统文件添加到了工程中

系统视图描述文件

CMSIS-SVD 用来书面地描述包含基于 ARM Cortex-M 处理器的微控制器系统，尤其是外设寄存器的内存映射的程序员视图。系统视图中包含描述的详细程度足以与由芯片厂商公布的器件参考手册中的描述媲美。信息范围从上层的外设功能描述一直到底层内存映射寄存器中单个位域的定义和功能。由芯片供应商对 CMSIS-SVD 文件进行制定和维护。将基于 XML 的 SVD 文件的输入到 SVDConv.exe 来生成依赖于调试视图和器件头文件的工具。



SVD 文件转换流程

复制 **Pack_with_Device_Support.zip** 中的 **03_SVD_File** 目录到您的工作环境中的**文件**目录中。

1. 用一个编辑器打开 **Files\SVD** 目录中的 MVCM3xxx.svd 文件并按如下进行更改：

```
<device schemaVersion="1.1" xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNamespaceSchemaLocation="CMSIS-SVD.xsd" >
 <vendor>MyVendor</vendor> <!-- device vendor name -->
 <vendorID>Generic</vendorID> <!-- device vendor short name -->
 <name>MVCM3xxx</name> <!-- name of part-->
 <series>MVCM3xxx</series> <!-- device series the device belongs to -->
 <version>1.2</version> <!-- version of this description, adding CMSIS-SVD 1.1 tags -->
 <description>ARM 32-bit Cortex-M3 Microcontroller based device, CPU clock up to 100 MHz.</description>
```

2. 在您的工作目录中打开一个命令行窗口并输入：

```
C:\temp\working_dfp>SVDConv.exe Files\SVD\MVCM3xxx.svd --generate=header --fields=macro
```

3. 您应该能看到类似于这样的一些 SVDConv 输出：

```
CMSIS-SVD SVD Consistency Checker / Header File Generator V2.82g
Copyright (C) 2010 - 2014 ARM Ltd and ARM Germany GmbH. All rights reserved.
Options: "Files\SVD\MVCM3xxx.svd" --generate=header --fields=macro
Reading file: "Files\SVD\MVCM3xxx.svd"

Decoding using XML SVD Schema V1.1

Found 0 Errors and 0 Warnings

Return Code: 0 (OK)
```

   将生成的头文件 **MVCM3xxx.h** 移动到 **Device\Include** 目录中。

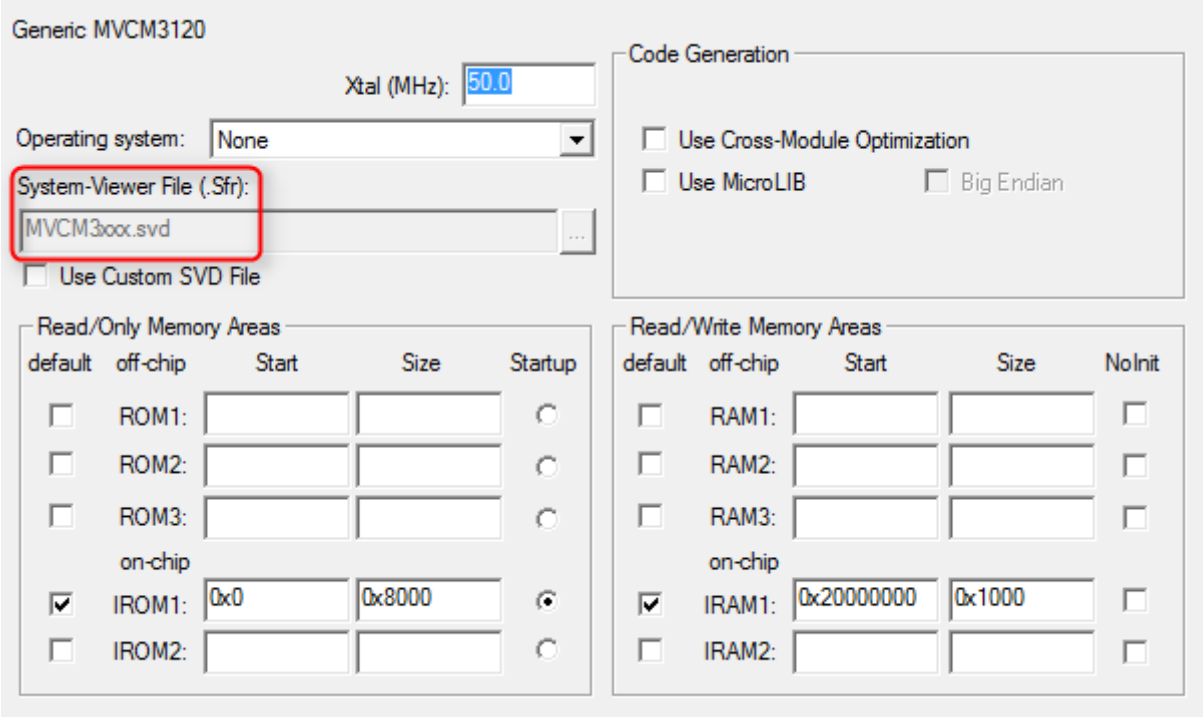
4. 向其 PDSC 文件的系列级区域添加如下两行代码：

```
<compile header="Device\Include\MVCM3xxx.h"/>
<debug svd="SVD\MVCM3xxx.svd"/>
```

5. 添加一个新的版本号：

```
<release version="1.0.2">
SVD and header file included
</release>
```

6. 最后，保存 PDSC 文件并用 **gen_pack.bat** 脚本生成包文件。然后安装这个包到您的开发工具中并创建一个新的工程。根据您的开发环境，您将看到在您工程中包含的 SVD 文件：



工程中的 SVD 文件

> **注意**
> 更多关于 CMSIS-SVD 的信息，请访问 **www.keil.com/cmsis/svd**

Flash 编程算法

Flash 编程算法用于擦除或下载应用程序到 Flash 器件。个 DFP 通常包含预定义的 Flash 算法来对它支持的器件编程。算法功能页面详尽地阐述了该机制。

对于 MVCM3 系列的器件，需要创建两种 flash 算法。MVCM3110/250 拥有 16kB 的 Flash,而 MVCM3120/260 拥有 32kB 的 Flash 存储。

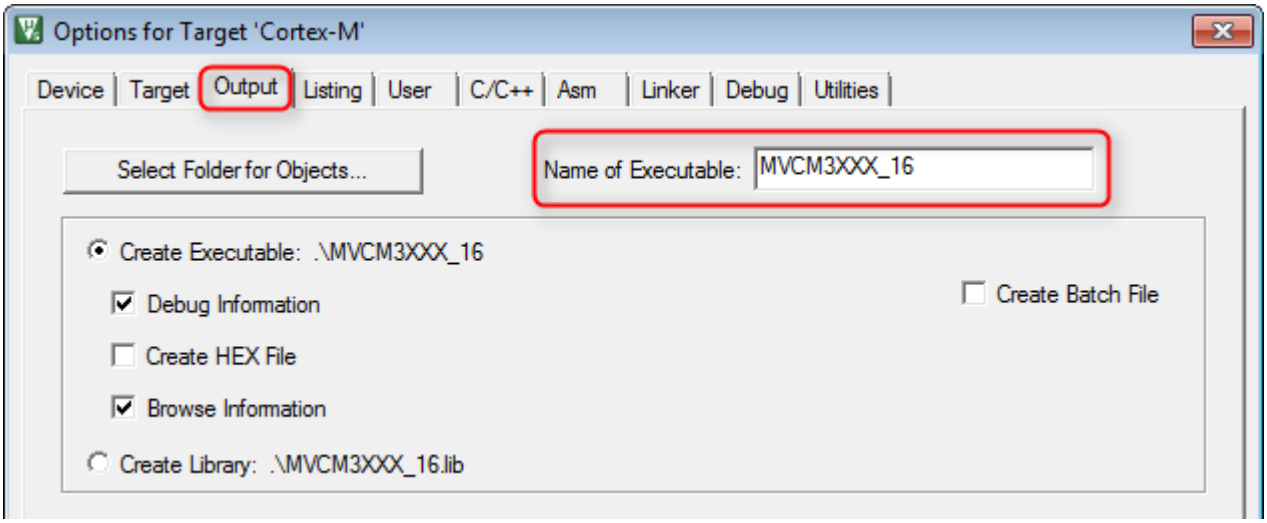
复制 **Pack_with_Device_Support.zip** 中的 **04_Flash_Programming** 目录到您的工作环境中的**文件**目录中。

1. 重命名工程文件 **NewDevice.uvproj**（在目录**_Template_Flash** 中）来表示新的 Flash ROM 器件名称，例如 MVCM3XXX_16.uvproj。

2. 用 uVision 打开该工程。选好目标(Cortex-M)器件。

> **注意**
> MDK-Lite 不支持创建 Flash 编程算法。

3. 打开会话框 **Project - Options for Target - Output** 并且更改 **Name of Executable** 域的内容来表示当前器件，这里用 MCVM3XXX_16。



'Cortex-M'目标选项

4. 事实上，现在您就可以开始调整文件 **FlashPrg.c** 中的编程算法。现在只更改文件 **FlashPrg.c** 中的器件参数（器件名称，器件大小以及扇区大小）：

```
struct FlashDevice const FlashDevice = {
    FLASH_DRV_VERS,            // Driver Version, do not modify!
    "MVCM3110/250 Flash",      // Device Name
    ONCHIP,                    // Device Type
    0x00000000,                // Device Start Address
    0x00004000,                // Device Size in Bytes (16kB)
    1024,                      // Programming Page Size
    0,                         // Reserved, must be 0
    0xFF,                      // Initial Content of Erased Memory
    100,                       // Program Page Timeout 100 mSec
    3000,                      // Erase Sector Timeout 3000 mSec
// Specify Size and Address of Sectors
    0x002000, 0x000000,        // Sector Size 8kB (2 Sectors)
    SECTOR_END
};
```

5. 用 **Project - Build Target** 来生成新的 Flash 编程算法。算法会在当前目录下的**_Template_Flash** 目录中被创建。
6. 复制输出文件**..\MVCM3XXX_16.FLM** 到一个新的称为 **Files\Flash** 的子目录下。添加这行代码到 MVCM3110/250 的 device 部分：

```
algorithm name="Flash\MVCM3XXX_16.FLM" start="0x00000000" size="0x4000"
default="1"/>
```

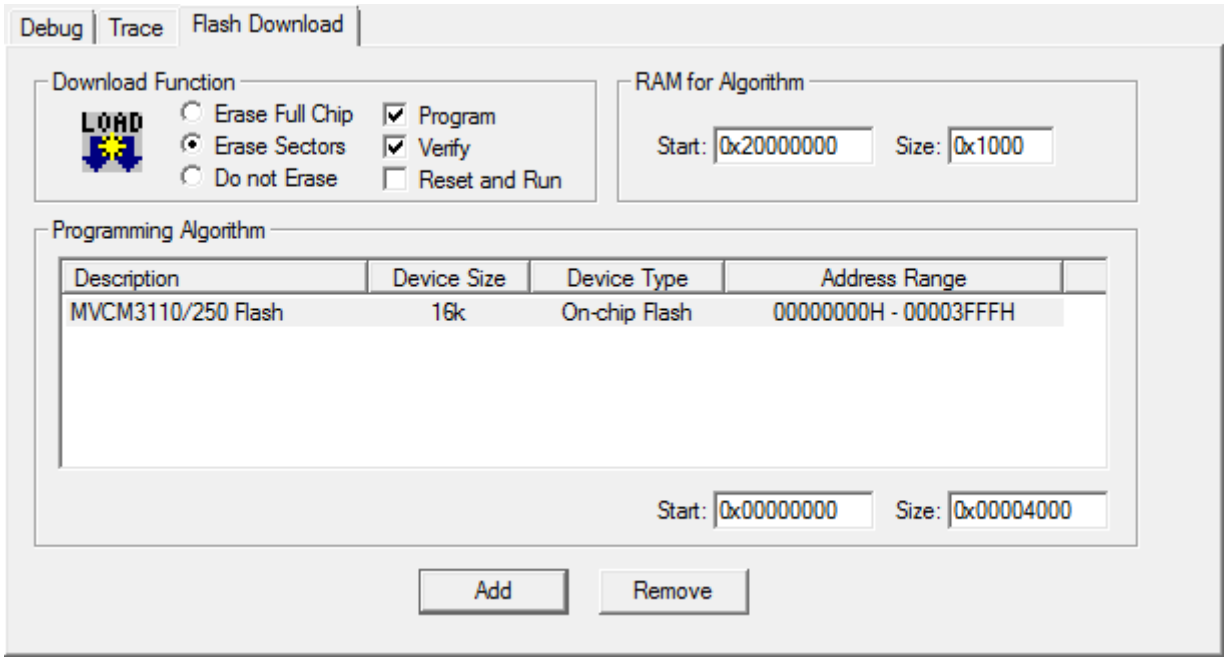
7. 为 MVCM3120/260 器件创建一个 **MVCM3XXX_32.flm** 文件。
8. 添加这行代码到 MVCM3120/260 的 device 部分：

```
algorithm name="Flash\MVCM3XXX_32.FLM" start="0x00000000" size="0x8000"
default="1"/>
```

9. 添加一个新的版本号：

```
<release version="1.0.3">
Flash Programming Algorithms added
</release>
```

10. 最后，保存 PDSC 文件并用 **gen_pack.bat** 脚本生成包文件。然后安装这个包到您的开发工具中。根据您的开发环境，在您的工程中将会看到 Flash 编程算法：



Flash 编程算法的显示

> **注意**
>
> 在 Flash 编程算法页面中提供了这一话题的更多信息。

器件属性

为了减少冗余，可将器件分为两个级别的分组：

- 系列：一个器件系列拥有相同的处理器的属性。
- 子系列：一个可选的子系列属性；这里增加或修改一个系列的特征。

某具体器件属性规定了：

- 器件：具体半导体器件的属性
- 变种：器件变体的属性（例如，拥有不同的封装或温度范围）或者 OEM 器件或者集成器件的电路板。

一个器件由如下属性来进行描述：

- 描述：文本形式的器件描述
- 特征：器件的外设和特征分类表。该列表用于在网站上显示器件的特征。
- 手册：关于器件以及其处理器的文档
- 处理器：嵌入到器件中的处理器和处理器的特征
- 编译：器件支持的构建工具的常规设置
- 调试配置：调试链接的默认配置
- 调试端口：调试器进行调试连接配置时器件的调试端口的描述。
- 调试：调试器进行调试连接配置时器件的特定信息，包括系统视图描述文件。
- 跟踪：调试器在配置跟踪时器件的特定信息。
- 存储器：器件内部和外部 RAM 和 ROM 的存储器区域布局
- 算法：器件特定的 Flash 编程算法

一个器件将会同时继承来自系列级和子系列级的技术参数。一些属性必需是唯一的。例如**<debug>**属性中 SVD 文件名属性值。因此，在 SVD 文件中由系列级指定的属性值可以被子系列级或者器件级的属性值重新定义。在系列级以及子系列级中的比如描述以及特征项等信息会被级联在一起，然后最终确定这些器件级的信息。

接下来将介绍如何指定器件 MVCM3110 的器件属性（请参阅基本器件系列包中所示的技术参数）。该系列的其它成员的器件属性也相应地被指定。

复制 **Pack_with_Device_Support.zip** 中的 **05_Device_Properties** 目录到您的工作环境中的**文件**目录中：

1. **MVCM3 Series** 系列的处理器 编译 描述和调试属性已经被指定了。这里对系列级的其它公共的手册和特征属性进行指定。添加下面的代码行到其 PDSC 文件的**<family>**部分中：

```
<book name="Docs\dui0552a_cortex_m3_dgug.pdf" title="Cortex-M3 Generic User Guide"/>
<book name="Docs\MVCM3XXX_Datasheet.pdf" title="MVCM3 Series Datasheet"/>
<book name="Docs\MVCM3XXX_Product_Brief.pdf" title="MVCM3 Product Brief"/>
<feature type="ExtInt" n="16"/>
<feature type="Temp" n="-40" m="105" name="Extended Temperature Range"/>
<feature type="VCC" n="2.5" m="3.6"/>
<feature type="RTC" n="32768"/>
<feature type="WDT" n="1"/>
```

2. **MVCM31xx** 子系列有些特征是其两个成员器件共有的。请将下面的代码添加到 MVCM3100 的**<subFamily>**部分：

```
<feature type="IOs" n="26"/>
<feature type="I2C" n="1"/>
<feature type="UART" n="4"/>
<feature type="Timer" n="6" m="32"/>
<feature type="QFP" n="32"/>
```

3. **MVCM3110** 器件具有一些独有的器件特征。请将下面的代码添加到 MVCM3110 的**< device >**部分：

```
<feature type="PWM" n="2" m="16"/>
```

器件特有的软元件

最后，需要将软元件和工程示例添加到 DFP 中。

1. 将下面的代码行添加到其 PDSC 文件的**<components>**部分：

```
<component Cclass="Device" Cgroup="HAL" Csub="GPIO" Cversion="1.0.0"
condition="MVCM3 CMSIS-CORE">
<description>GPIO HAL for MyVendor MVCM3 Series</description>
<files>
<file category="header" name="Device\Include\GPIO.h"/>
<file category="source" name="Device\Source\GPIO.c"/>
</files>
</component>
<component Cclass="Device" Cgroup="HAL" Csub="ADC" Cversion="1.0.0"
condition="MVCM3 CMSIS-CORE">
<description>ADC HAL for MyVendor MVCM3 Series</description>
<files>
<file category="header" name="Device\Include\ADC.h"/>
<file category="source" name="Device\Source\ADC.c"/>
</files>
</component>
<component Cclass="CMSIS Driver" Cgroup="I2C" Cversion="1.0.0"
condition="MVCM3 CMSIS-CORE" maxInstances="3">
<description>I2C Driver for MVCM3 Series</description>
<RTE_Components_h>
#define RTE_Drivers_I2C0 /* Driver I2C0 */
#define RTE_Drivers_I2C1
#define RTE_Drivers_I2C2
</RTE_Components_h>
<files>
<file category="source" name="Drivers\I2C\I2C_MVCM3.c"/>
<file category="header" name="Drivers\I2C\I2C_MVCM3.h"/>
</files>
</component>
<component Cclass="CMSIS Driver" Cgroup="UART" Cversion="1.0.0"
condition="MVCM3 CMSIS-CORE" maxInstances="5">
<description>UART Driver for MVCM3 Series</description>
<RTE_Components_h>
```

```
#define RTE_Drivers_UART0
#define RTE_Drivers_UART1
#define RTE_Drivers_UART2
#define RTE_Drivers_UART3
#define RTE_Drivers_UART4
</RTE_Components_h>
<files>
<file category="source" name="Drivers\UART\UART_MVCM3.c"/>
<file category="header" name="Drivers\UART\UART_MVCM3.h"/>
</files>
</component>
```

注意

前两个软元件被添加到 **Device::HAL** 是因为它们是器件系列特有的且未使用发布的 API。I2C 和 UART 的驱动秉承了 **CMSIS-Driver** 规范。因此，把它们添加到了 **CMSIS Driver** 器件类中。

2. 同样创建一个工程示例。将**<examples>**部分注释取消并添加这些：

```
<example name="Dummy" doc="Abstract.txt" folder="Examples\dummy">
<description>Dummy project</description>
<board name="MVCM3 Starter Kit" vendor="MyVendor"/>
<project>
<environment name="uv" load="dummy.uvprojx"/>
</project>
<attributes>
<category>Getting Started</category>
</attributes>
</example>
```

3. 添加一个新的版本号：

```
<release version="1.0.4">
DFP finalized
</release>
```

4. 最后，保存 PDSC 文件并用 **gen_pack.bat** 脚本生成包文件。然后安装这个包到您的开发工具中。