CMSIS-Pack 文檔

<https://blog.csdn.net/sunflowerfsw/article/details/46491905>

CMSIS-Pack描述了一個軟件組件，設備參數和評估板板級支持之間交互機制。基於xml的包描述文件(PDSC)描述一個軟件的內容包(文件集合),包括：

•源代碼，頭文件和軟件庫

•文檔和源代碼模板

•設備以及啟動代碼和編程算法參數

•示例項目

完整文件與PDSC文件的打包收集在ZIP格式的軟件包。所述PDSC文件被設計為軟件開發環境，並描述了用戶和設備相關的上下文中這樣的軟件包提供的文件。軟件包可用於多種用途（見下文）。

安裝軟件包後，所有附帶的軟件組件提供給開發工具。軟件組件的源模塊，頭和配置文件，以及庫的集合。包含軟件組件包也可以包括示例項目和用戶代碼模板。

**軟件包使用案例**

上圖顯示了以下使用情況的軟件包：

•器件系列包（DFP）：包含CMSIS系統/啟動文件，驅動程序和閃存算法微控制器器件系列。

•CMSIS軟件包：包含由ARM提供的通用CMSIS組件（CORE，DSP庫和RTOS實施）。

•中間件包：包含屬於中間件軟件組件（如源代碼或庫）。

•板級支持包（BSP）：包含文檔，圖表，以及一定的開發板的驅動程序。

•內部軟件包：通常包含可公司或工程集團內的分佈式軟件組件。

注意

軟件包可以解決多個用例在同一時間

**以下各節給你的基礎知識的詳細信息**：

•創建軟件包解釋需要創建一個軟件包，一般的步驟。

•包裝用的軟件組件提供了在PDSC文件的內容的詳細信息。

•包與設備支持的描述器件系列包（指定聯絡點）的額外要求。

•包帶板支持涵蓋了創建板級支持包（BSP的）。

•組為例說明了軟件包的例子，它是CMSIS包更詳細的一部分。

•發布包顯示發布包的選項。

•工具，用於創建包迎合的是創建一個包時有用的程序。

•包裝說明（\*.PDSC）格式描述了可以在一個PDSC文件中使用的所有XML元素。

•配置向導註解可用於創建GUI的像在開發工具的配置文件的元素。

有關CMSIS-Pack文件存在於以下ARM CMSIS::目錄：

|  |  |
| --- | --- |
| **文件夾** | **內容** |
| Utilities | 實用工具用於創建包 |
| Pack\Example | 參考軟件包 |
| Pack\Tutorials | 教程創建包 |

**創建軟件包**

該CMSIS-Pack格式用於實現一個軟件包，旨在為可擴展性為未來的需求。它提供了一個管理流程，支持工具的獨立分佈：

•針對特定設備，設備支持工具鏈和軟件應用：

•關於處理器和它的功能的信息。

•C和匯編文件，設備啟動和訪問內存映射外設寄存器。

•參數，技術信息，以及關於器件系列的數據表和具體的設備。

•設備描述和可用的外設。

•內部和外部的RAM和ROM的地址范圍的內存佈局。

•編程設備flash算法。

•調試和跟蹤配置以及系統視圖描述文件的存儲設備特定的顯示映射外設寄存器。

•在原型階段快速開發的板級支持：

•關於開發板和它的功能的信息。

•參數，技術資料以及有關的電路板，安裝的微控制器和外圍設備的數據表。

•用於板載外圍設備驅動程序

•軟件組件簡化了軟件的再使用和第三方軟件的管理。它們包括：

•源模塊，頭文件和配置文件，以及庫的集合。

•軟件，包括功能和API的文檔。

•應用程序編程接口中指定的一個或多個API，用於軟件組件的另一個軟件包的一部分。

•示例項目可以作為類似應用的基準和起點。它們包括：

•完全定義在指定的構建和調試環境中工作的開箱即用的應用項目。

•關於這個例子的目的，如何構建，配置和執行應用程序的文檔。

•用戶代碼模板有助於瞭解一個軟件組件的基本概念，並更好地向用戶提供一個良好的起點，實現了他的申請。

**創建一個包的步驟**

下圖顯示了需要創建一個軟件包的基本步驟：

1.產生輸入：創建將被包被傳遞的文件。

2.組織文件：使用目錄來分隔包的內容。

3.創建 PDSC文件：將基於XML的文件PDSC可以用任何文本編輯器創建。編輯器可以在包的早期開發階段通過驗證XML代碼對綱要文件來幫助找到可能的bug。

4.Generate PACK：任何壓縮工具支持ZIP格式可以用來創建一個包文件。 PackCheck.exe對包的內容驗證有用。

以下頁面將展示如何從頭開始創建軟件包。軟件組件包頁面將以僅含有PDSC文件和一個軟件組件的簡單包開始，然後將詳細解釋包內的更多元件。設備支持包頁面加上一個DFP有的需求信息，板級支持包將詳細闡述特定BSP內容。

**包裝規范**

屬於某個軟件包的所有文件壓縮成使用標准ZIP文件格式\*.PACK文件。每個軟件包的內容在包描述（\*.PDSC）文件，是每個包的一部分描述。請參閱包裝說明（\*.PDSC）格式的更多信息。

軟件包都必須有一個唯一的文件名下面這個命名約定：

<vendor>.<name>.<version>.pack.

其中：

•<vendor>是軟件包的供應商或銷售商的名稱。

•<name>是軟件包的名稱。這是供應商的責任，以確保獨特的包名。

•<version>指定軟件包的版本號。

•.pack是文件擴展名識別軟件包

<vendor>, <name>, and <version> 在 PDSC 文件中聲明。

PDSC文件需要駐留在包文件的頂層，用它作為對所有文件引用的根目錄。

**包教程**

在ARM ::CMSIS包包含了下面幾頁解釋教程。在\ CMSIS\包\教程目錄中的以下三個ZIP文件可供選擇：

|  |  |
| --- | --- |
| **ZIP 文件** | **文檔** |
| Pack\_with\_Software\_Components.zip | 軟件組件包 |
| Pack\_with\_Device\_Support.zip | 器件支持包 |
| Pack\_with\_Board\_Support.zip | 板級支持包 |

**軟件組件包**

這部分是一個教程，介紹了如何創建一個軟件包。最初，創建軟件包用一個簡單的PDSC文件，該文件只包含一個軟件組件。然後這個軟件包擴展到顯示各種功能的描述軟件組件。

**准備工作**

1. 在電腦上創建一個工作目錄，例如C：\TEMP\working。

2. 在ARM::CMSIS包裡進入目錄 \CMSIS\Pack\Tutorials安裝使用。請咨詢您的開發工具的文檔上的包安裝目錄結構的詳細信息。在一個標准的μVision安裝，你會發現它下面的C：\Keil\_v5\ARM\Pcak\ARM\CMSIS\Version。

3. 打開Pack\_with\_Software\_Components.zip文件.

4. 復制ZIP文件中的01\_Create\_PDSC目錄的內容到你的工作目錄。

5. 確保文件/目錄沒有寫保護（刪除只讀標志）。

6. 在ARM:: CMSIS Pack安裝目錄CMSIS\：Utilities將列文件復制到您的工作目錄：

◦ PackChk.exe

◦ PACK.xsd

**創建PDSC文件**

1. 用編輯器打開vendor.packname.pdsc文件並改變它像這樣：

<?xml version="1.0"encoding="UTF-8"?>

<package schemaVersion="1.2"xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"xs:noNamespaceSchemaLocation="PACK.xsd">

<vendor>MyVendor</vendor>

<name>MyPack</name>

<description>InternalSoftware Pack</description>

<url></url>

<supportContact></supportContact>

<license>Docs\license.txt</license>

注意：在此所有代碼的實施例和下面的各節可以在01\_Create\_PDSC目錄snippets.xml文件中找到

一個PDSC文件以XML版本和編碼信息開始。在<package>部分聲明架構文件，用於驗證PDSC文件(因此復制PACK.xsd文件到工作目錄)的XSD文件的版本：

◦ **<name>**指定包的名稱

◦ **<description>**描述了包中的幾句話

◦ **<vendor>**規定了包的供應商的名稱

◦ **<url>**定義包的下載URL（可能為空）

注意：在本教程中，未指定的URL。這意味著包將被標記的包安裝程序為脫機，需要手動更新。要解決這個問題，就可以指定文件“下載”使用文件URI位置。對於Windows用戶來說，這看起來像下面這樣：

<url>file:///c:/temp/working</url>

◦ **<supportContact>**對包的具體問題/問題可以提供一個電子郵件地址或網頁URL

◦ **<license>**包的安裝過程中會顯示一個可選許可文件。如果許可協議不被接受，安裝開發工具將中止。

1. 增加發布信息和關鍵字：

<releases>

<release version="1.0.0"date="2015-03-09">

Initial version

</release>

</releases>

<keywords>

<keyword>MyVendor</keyword>

<keyword>My SoftwareComponent</keyword>

</keywords>

**<releases>**部分信息將用於：

◦ 決定包的版本（需要的包文件的文件名）

◦ 發布信息顯示

**<keywords>**可以賦予搜索引擎更好的可視性

1. 添加<components>部分有以下幾點：

<components>

<component Cclass="MyClass" Cgroup="MyGroup" Csub="MySubGroup" Cversion="1.0.0">

<description>MySWComp</description>

<files>

<file category="doc"name="Docs\MySWComp.htm"/>

<file category="header"name="MySWComp\header\_mylib.h"/>

<file category="header" name="MySWComp\config\_mylib.h" attr="config" version="1.0.0"/>

<file category="source"name="MySWComp\mylib\_one.c"/>

<file category="source"name="MySWComp\mylib\_two.c"/>

</files>

</component>

</components>

在**<components>**的更多信息將在SoftwareComponents部分給出。保存文件並關閉編輯器。在接下來的部分中，一個包的生成將進行詳細說明。

**生成一個包**

1. 重命名文件vendor.pack\_name.pdsc為MyVendor.MyPack.pdsc。
2. 從您的工作目錄在文本編輯器重打開批處理文件gen\_pack.bat。檢查下面的代碼行反映您的設置：

“C：\ Program Files\7-Zip\ 7z.exe”一％PACKNAME％Files-tzip

如果你沒有7-Zip在你的電腦上安裝，您可以指定任何其他ZIP工具，支持命令行調用。在這種情況下，你必須相應地調整命令行參數。

1. 保存gen\_pack.bat，您可以通過雙擊或使用該建議看PackChk.exe的輸出命令行執行它（在文件的位置打開一個命令行窗口，然後輸入gen\_pack.bat ）。該批處理文件將：

◦ 檢查PDSC文件的可用性

◦ 復制PDSC文件到文件目錄

◦ 在文件目錄運行PackChk.exe

◦ 在工作目錄創建一個包文件。它會被稱為MyVendor.MyPack.1.0.0.pack。

1. 通過雙擊它來注冊包。取決於所使用的開發軟件，包內容的顯示可能有所不同：

**軟件組件**

一個組件列出了屬於它和相關的項目中的文件。組件本身或每個單獨的文件涉及到condition必須解析是真的;如果是假，則該組件或文件是不適用在給定的上下文中。

每個軟件組件必須有用於識別組件的以下屬性：

• 組件類（Cclass）：例子是CMSIS，設備，文件系統

• 組件組（CGROUP）：例子是CMSIS：RTOS，設備：啟動，文件系統：CORE

• 組件版本（Cversion）：該軟件組件的版本號。

此外，一個軟件組件可以具有附加屬性：

• 組件子組（Csub）：例如 CMSIS:RTOS:MyRTOS,

Device:Driver USBD:Full-speed

• 組件變形（Cvariant）：軟件組件的變形。

• 組件供應商（Cvendor）：該軟件組件的供應商。

所述Cclass，Cgroup，Csub的，Cvariant和Cversion屬性與由包指定識別部件供應商一起使用。組件供應商必須確保組合Cclass，Cgroup，Csub的和Cversion是獨一無二的，不被多個組件使用。

下面的Cclass名具有特殊的意義：

• 板級支持通常包含用於外圍設備板特定的驅動程序。

• CMSIS包含一般CMSIS組件，如CORE，DSP擴展，RTOS（如Keil的RTX）

• 設備包含目標設備的啟動文件

• CMSIS驅動包含為目標設備准備的符合外設驅動程序的CMSIS驅動程序。這些將列舉在各自CGROUP API下面的條目（例如以太網（API））

• 文件系統，圖形，網絡和USB包含中間件，以支持這些組件

其他Cclass的名稱，可以自由分配到軟件組件。

萬一多個相互依賴的元件（屬於同一類）形成解決方案的一部分，這些可以被分組在一個所謂的bundle。

使用下面的語法引用軟件組件和API：

<Vendor>::<ComponentClass>:<Component Group>:<Sub-Group>

例如：

• :: CMSIS：RTOS（API） - CMSIS-RTOS API。

• ARM ::CMSIS：DSP - CMSIS-DSP庫。

• ::File System：Drive：NOR - 文件系統的NOR閃存驅動器。

下面是在開發工具的軟件組件的顯示示例:

在開發工具的軟件組件的顯示

**軟件組件文件**

每個軟件組件包括一定數量的文件。每個文件至少具有以下屬性：

• name：文件路徑，文件名，文件擴展以路徑/ 名.擴展格式。該文件路徑是相對於包的根目錄下。

• categry：定義文件的目的。如表文件類別列出選擇預定義的值。

每個文件可以選擇具有以下屬性：

• attr：定義文件的特殊使用和處理。選擇作為表文件屬性定義的一個預定值。

• condition：輸入的條件標識符（屬性ID）。該元素用於如果條件解析為真。

• select：簡要說明文件和目的。當attr設置為模板或接口，則需要select屬性。當一個組件的多個模板文件具有相同的select字符串，它們將被視為一個單個可選擇模板。這樣一來，多個模板或接口文件可以捆綁。

• src：路徑信息。指定的路徑是相對於包描述文件（PDSC）。

• version：文件的具體版本信息。這是特別用於復制到項目工作區文件。前一個文件被復制，版本檢查避免了不必要的復制行為。如果文件不具有一個版本，則該組件版本被使用。

**條件**

一個條件描述依賴關系：

• 特定設備

• 一定處理器

• 工具屬性

• 其它組件的存在

條件用於定義 AND/OR規則，使部件有條件，因此，在某些情況下只提供，例如用於特定設備或處理器。條件也用於表達的軟件組件之間的依賴性。

每個條件有一個id，是一個在PDSC文件的范圍內是唯一的。一個id可以在組件的API，例子中，文件和其它條件的條件屬性來引用。在接受，要求，或拒絕元素設置的所有屬性必須解析為true才能成為真正的元素。一個條件為真時：

• 至少有一個接受的元素是真實的，

• 所有必需的元素是真實的，

• 沒有否認元素是真實的。

如果在處理期間條件解析為假，各元件將被忽略。

讓我們修改從創建PDSC文件部分的示例中加入了CMSIS-RTOS和特定的Cortex-M級核心庫的必要條件：

• 當使用 Cortex-M0 and Cortex-M0+processor需要mylib\_cm0.lib

• 當使用Cortex-M3 processor 需要mylib\_cm3.lib

• 當使用 Cortex-M4 or Cortex-M7processor需要 mylib\_cm4.lib

復制Pack\_with\_Software\_Components.zip文件02\_Conditions目錄的內容到你的工作環境中。

1. 加個**<conditions>**部分如下所示：

<conditions>

<conditionid="ARM Compiler">

<requireTcompiler="ARMCC"/>

</condition>

<conditionid="CM0">

<description>Cortex-M0based device</description>

<requirecondition="ARM Compiler"/>

<acceptDcore="Cortex-M0"/>

<acceptDcore="Cortex-M0+"/>

</condition>

<conditionid="CM3">

<description>Cortex-M3based device</description>

<requirecondition="ARM Compiler"/>

<acceptDcore="Cortex-M3"/>

</condition>

<conditionid="CM4\_CM7">

<description>Cortex-M4based device</description>

<requirecondition="ARM Compiler"/>

<acceptDcore="Cortex-M4"/>

< acceptDcore="Cortex-M7"/>

</condition>

<condition id="CMSIS Core with RTOS">

<description>CMSISCore with RTOS for Cortex-M processor</description>

<acceptcondition="CM0"/>

<acceptcondition="CM3"/>

<acceptcondition="CM4\_CM7"/>

<requireCclass="CMSIS" Cgroup="CORE"/>

<requireCclass="CMSIS" Cgroup="RTOS"/>

</condition>

</conditions>

1. 通過添加條件和增加組件的版本號改變已經存在的組件的第一行：

<componentCclass="MyClass"Cgroup="MyGroup"Csub="MySubGroup" Cversion="1.0.1" condition="CMSISCore with RTOS">

1. 下面的代碼添加到現有的組件：

<file category="library" condition="CM0" name="MySWComp\Lib\mylib\_cm0.lib"/>

<file category="library" condition="CM3" name="MySWComp\Lib\mylib\_cm3.lib"/>

<file category="library" condition="CM4\_CM7" name="MySWComp\Lib\mylib\_cm4.lib"/>

需要注意的是庫文件應始終對一個特定的C / C++編譯器的有相應的條件。要做到這一點，每個處理器都有條件做為ARM C / C++編譯器的附加要求。

1. 在PDSC文件的頭部添加新的版本號，使得一個包用一個新的版本號將被創建：

<release version="1.0.1">

Conditions added

</release>

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具，並觀察這些差異1.0.0版本。

顯示有條件增加文件的Cortex-M3的設備開發工具

注意

PackChk.exe將會顯示M362的警告。

\*\*\* WARNING M362: (Line 82)

Not all Component Dependencies for'Cclass=MyClass, Cgroup=MyGroup, Csub=MySubGroup, Cversion=1.0.1, Capiversion='can be resolved.

RTE Model reports: No component is installed.

這是因為它無法解析，並檢查依賴性成分不屬於此包的一部分（在這種情況下，CMSIS組件）。您可以放心地忽略此警告或ARM補充：CMSIS PDSC的檢查過程，從而使依賴可以正確檢測。

**變體**

軟件組件可以具有變體，例如：

• 調試版本或發布版本沒有診斷輸出

• 長/短文件名

• 快速/慢速模式

變體是相互排斥的（僅一種變體可以一次被選擇）。變體是組件ID的可選部分。該變種說明是一個簡短的字符串（如發布，調試）。

**注意**

版本管理依賴於變體，以保持版本之間不變。

下面的示例介紹了兩個變體一個新的組件的包：調試和發布。復制Pack\_with\_Software\_Components.zip文件03\_Variants目錄的內容到您的工作環境文件目錄的。

1. 添加以下行的組件部分的PDSC文件：

<component Cclass="MyVariant" Cgroup="MyGroup" Cvariant="Release" Cversion="1.0.2" condition="CMSISCore with RTOS">

<description>Releaseversion of MyVariant</description>

<RTE\_Components\_h>

<!-- the following contentgoes into file 'RTE\_Components.h' -->

#define RTE\_MyVariant\_Release/\* MyVariant Release Version \*/

</RTE\_Components\_h>

<files>

<file category="doc"name="Docs\MySWComp.htm"/>

<file category="header"name="MySWComp\header\_mylib.h"/>

<file category="header"name="MySWComp\config\_mylib.h" attr="config" version="1.0.0"/>

<file category="source"name="MySWComp\mylib\_one.c"/>

<file category="source"name="MySWComp\mylib\_two.c"/>

<file category="library" condition="CM0" name="MySWComp\Lib\mylib\_cm0.lib"/>

<file category="library" condition="CM3" name="MySWComp\Lib\mylib\_cm3.lib"/>

<file category="library" condition="CM4\_CM7" name="MySWComp\Lib\mylib\_cm4.lib"/>

</files>

</component>

<component Cclass="MyVariant" Cgroup="MyGroup" Cvariant="Debug" Cversion="1.0.2" condition="CMSISCore with RTOS">

<description>Debugversion of MyVariant</description>

<RTE\_Components\_h>

<!-- the following contentgoes into file 'RTE\_Components.h' -->

#define RTE\_MyVariant\_Debug /\* MyVariant Debug Version \*/

</RTE\_Components\_h>

<files>

<file category="doc"name="Docs\MySWComp.htm"/>

<file category="header"name="MySWComp\header\_mylib.h"/>

<file category="header"name="MySWComp\debug\_config\_mylib.h" attr="config" version="1.0.0"/>

<file category="source"name="MySWComp\debug\_mylib\_one.c"/>

<file category="source"name="MySWComp\debug\_mylib\_two.c"/>

<file category="library" condition="CM0" name="MySWComp\Lib\debug\_mylib\_cm0.lib"/>

<file category="library" condition="CM3" name="MySWComp\Lib\debug\_mylib\_cm3.lib"/>

<file category="library" condition="CM4\_CM7" name="MySWComp\Lib\debug\_mylib\_cm4.lib"/>

</files>

</component>

1. 添加一個新的修訂，以反映變化中的一個新生成的包：

<release version="1.0.2">

Variants introduced

</release>

3 .最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具，並觀察這些差異1.0.1版本。

MyVariant在開發工具中的顯示

**RTE\_Components.h**

有時，軟件組件需要知道其它組件。根據另一組件的可用性或配置，某些設置可能是必需的。一個明顯的頭文件可以使組件知道對方：RTE Components.h。將一條C代碼添加到這個頭文件，只需下面一樣就可以添加到您的組件：

<RTE\_Components\_h>

#define RTE\_MyVariant\_Debug /\* MyVariant Debug Version \*/

</RTE\_Components\_h>

**＃define**可以由一個項目的其他部件中進行檢查。

**注意**

**RTE components.h**文件需要包括在你的源文件的某處。

**束**

一個束基本上是在Cclass級別的變體。它規定了相互依賴的組件的集合屬性類，版本和可選組和供應商。一個包內的組件繼承了包的屬性設置，不能再次設置這些屬性。束確保跨越多個互通組件屬性的一致性，從不同的解決方案Cclass內限制組件的混搭。除組件外，一個束有強制性內容描述和DOC（用於文檔）。

束的一個例子出現在創建BSP束，其中所述束是為一定的開發平台用來提供電路板支持的文件。

**實例**

一些軟件組件允許多個實例。如果一個以上的外設可以被連接的話，這是有用的。

1. 添加實例的最大數目的信息是容易的。改變的第一個組件（:: MyClass的：mygroup的：MySub）如下：

<componentCclass="MyClass"Cgroup="MyGroup"Csub="MySubGroup" Cversion="1.0.3" condition="CMSISCore with RTOS" maxInstances="3">

1. 添加一個新版本號

<release version="1.0.3">

Maximum number of instances specified

</release>

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具，並觀察這些差異1.0.2版本。當選擇一定數量的組件的實例的，使得開發工具來復制所需要的配置文件多次到項目。這確保了每個組件實例可以單獨配置。

**API接口**

API是一個軟件組件，它只是定義了一個C / C++應用程序編程接口（API）的一種特殊形式。一個API不包含實際執行（通常由源代碼或庫文件提供），並且不能在一個開發工具選擇。一個例子是CMSIS-RTOS API，它被指定為CMSIS的一部分。但是，實際的實時操作系統的實現是由不同的廠商提供的。一個API包含名稱，簡要說明以及一個或多個的頭文件，以及含有該API的詳細規範文檔。

復制Pack\_with\_Software\_Components.zip文件目錄內的05\_APIs目錄的文件到您的工作環境中：

1. 按照下面添加一個<apis>

<api Cclass="Device"Cgroup="MyAPI" exclusive="0">

<description>API forMyAPI</description>

<files>

<file category="doc"name="Docs\API\MyAPI.htm"/>

<file category="header"name="API\Include\MyAPI.h"/>

</files>

</api>

1. 添加一個新的版本號

<release version="1.0.4">

MyAPI added

</release>

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具，並觀察這些差異1.0.3版本。最有可能的，你會看到沒有因為API是不可選擇的，因而沒有在工具顯示。

**用戶代碼模板**

用戶代碼模板提供了一個快速啟動實施的應用程序。要添加用戶的代碼模板，添加源文件的屬性在<組件>ATTR=“模板”。選擇屬性是用來識別模板文件。具有相同屬性的選擇模板文件將同時添加到項目中。

復制Pack\_with\_Software\_Components.zip文件的06\_User\_Code\_Templates目錄的內容到工作環境中：

1. 將以下代碼添加到變體（debug 和 release）::MyVariant:MyGroupcomponent:

<file category="source"name="MySWComp\Templates\mylib\_template1.c" attr="template" select="EasyTemplate"/>

<file category="source"name="MySWComp\Templates\mylib\_template2.c" attr="template" select="ComplexTemplate"/>

<file category="source"name="MySWComp\Templates\mylib\_template3.c" attr="template" select="ComplexTemplate"/>

1. 添加一個新的版本號：

<release version="1.0.5">

User Code Templates added

</release>

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具，並觀察這些差異1.0.4版本。

用戶代碼模板在開發工具中的顯示

**示例工程**

示例工程有助於更好瞭解一定的MCU或開發板。一個例子，它的每個單獨的文件條件必須解析真實的條件;如果是假的，示例或文件將被忽略。板級元素被用來使用的電路板廠商引用到一個或多個板的描述和命名的例子是針對。每個例子可以指定使用Cclass，CGROUP，Csub的和Cversion相關組件的屬性。在<項目>元素包含要加載的支持的開發工具和項目文件的名稱。

復制Pack\_with\_Software\_Components.zip文件在您的工作環境中的文件目錄的07\_Example\_Projects目錄的內容：

1. 如下所示添加一個<examples>部分

<examples>

<example name="MyDevBoard Example" doc="Abstract.txt" folder="MyPackExample">

<description>CMSIS-RTOSbased example</description>

<board name="MyDevBoard" vendor="MyVendor"/>

<project>

<environment name="uv"load="MyPackExample.uvprojx"/>

</project>

<attributes>

<component Cclass="CMSIS"Cgroup="CORE"/>

<component Cclass="Device"Cgroup="Startup"/>

<component Cclass="CMSIS"Cgroup="RTOS"/>

<component Cclass="MyVariant" Cgroup="MyGroup"/>

</attributes>

</example>

<examples>

1. 添加一個新的版本號

<release version="1.0.6">

Example project added

</release>

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具，並觀察這些差異1.0.5版本。

示例工程在開發工具中的顯示

示例工程依賴於一個存在的開發板（代碼需要在真實的硬件上運行）。在上面的代碼中，你會發現，被指定的開發板稱為“MyDevBoard”。板級支持包說明如何板卡集成到一個包。

**器件支持包**

一個包含**<devices>**元素的軟件包稱作設備家族包（DFP）。一個DFP可以包含額外的軟元件或者工程示例或者其任意組合。典型的DFP內容有：

 解釋設備或者設備系列功能的設備屬性。

 配置設備的軟元件和基本設備驅動程序，典型的有：

 需要用來設置C運行時庫，器件時鐘和存儲器接口的CMSIS兼容的系統和啟動文件。

 提供給中間件棧使用的物理器件外設的軟件例行程序對應的外設驅動程序接口

 程序員視角的用來描述器件外設的一個或多個系統視圖描述文件。驅動程序可遵守CMSIS-Driver標准。

 擦除和下載代碼到片上閃存用的Flash編程算法。

 顯示器件及其外設的用法的工程示例。

 可以用來作為應用開發起點的用戶代碼模板。

該節是一個說明如何創建DFP的教程。起初，在DFP中只有描述器件的PDSC（Package description——包描述）文件。然後向該DFP中擴充添加SVD（System View Description——系統視圖描述）文件、Flash算法和器件相關的如系統和HAL（Hardware Abstraction Layer——硬件抽象層）的軟元件文件。示例項目和代碼模板可作為描述節添加到軟元件包中。

**DFP使用案例**

為支持新的器件，器件系列包（DFP）可以作為開發工具的擴展由芯片供應商提供。相對於某一器件系列，DFP使芯片供應商能獨立的分發器件支持工具。

DFP也能被用來提供顯示相關網站的信息。一個例子是[www.keil.com/dd2/](http://www.keil.com/dd2/) 上的新設備數據庫：從DFP中提取網站上的設備信息

**創建一個DFP的步驟**

**基本器件系列包**

在下面的章節中，將為設備供應商Myvendor提供的稱為MVCM3的一個虛構的器件系列創建DFP。器件系列由被分成兩個子系列的四個成員組成。所述MVCM3系列的規格如下：

准備工作

1. 在您的PC上創建一個工作目錄，例如C:\temp\working\_dfp。

2. 進入可用的ARM::CMSIS包安裝目錄的\CMSIS\Pack\Tutorials。請查閱您的開發工具的文檔來獲取包安裝目錄結構的詳細信息。在μVision中，它在目錄C:\Keil\ARM\Pack\ARM\CMSIS\version下。

3. 打開文件Pack\_with\_Device\_Support.zip。

4. 將該ZIP文件中的01\_Basic\_Pack目錄復制到您的工作目錄。

5. 確保文件或目錄沒有被寫保護（刪除只讀標志）。

6. 從可用的ARM::CMSIS包安裝目錄的\CMSIS\Utilities下復制如下文件到您的工作目錄中：

o PackChk.exe

o PACK.xsd

o SVDConv.exe

7. 用一個編輯器打開文件**MyVendor.MVCM3.pdsc**。

**代碼示例**

1. 取消PDSC文件中對<devices>部分的注釋，並添加以下內容：

<familyDfamily="MVCM3 Series" Dvendor="Generic:5">

<processorDcore="Cortex-M3" DcoreVersion="r2p1" Dfpu="0"Dmpu="0" Dendian="Little-endian"/>

<description>

The MVCM3 device familycontains an ARM Cortex-M3 processor, running up to 100 MHz with a versatile setof on-chip peripherals.

</description>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Subfamily 'MVCM3100' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<subFamilyDsubFamily="MVCM3100">

<processorDclock="50000000"/>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Device 'MVCM3110' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<deviceDname="MVCM3110">

<memoryid="IROM1" start="0x00000000" size="0x4000"startup="1" default="1"/>

<memoryid="IRAM1" start="0x20000000" size="0x0800"default="1"/>

</device>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Device 'MVCM3120' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<deviceDname="MVCM3120">

<memoryid="IROM1" start="0x00000000" size="0x8000"startup="1" default="1"/>

<memoryid="IRAM1" start="0x20000000" size="0x1000"default="1"/>

</device>

</subFamily>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Subfamily 'MVCM3200' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<subFamilyDsubFamily="MVCM3200">

<processorDclock="100000000"/>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Device 'MVCM3250' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<deviceDname="MVCM3250">

<memoryid="IROM1" start="0x00000000" size="0x4000"startup="1" default="1"/>

<memoryid="IRAM1" start="0x20000000" size="0x0800"default="1"/>

</device>

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Device 'MVCM3260' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<device Dname="MVCM3260">

<memoryid="IROM1" start="0x00000000" size="0x8000"startup="1" default="1"/>

<memoryid="IRAM1" start="0x20000000" size="0x1000"default="1"/>

</device>

</subFamily>

</family>

注意

Dvendor ID不能自由選擇。此處設置的ID可以在PACK.xsd中找到（可用的ARM::CMSIS包安裝目錄的CMSIS\Utilities中）。

本節以及下面各節的所有的代碼示例可以在**01\_Basic\_Pack** 目錄下的snippets.xml文件中找到。

2. 保存PDSC文件並用gen\_pack.bat腳本生成包文件。參見Generate a Pack以瞭解進一步的細節。然後安裝這個包到您的開發工具中。

在開發工具中顯示微控制器器件

系統和啟動文件

**CMSIS-CORE**定義了如下要在嵌入式應用中使用的文件：

 **startup\_<device>.s**包含復位處理程序和異常向量。其在復位之後執行，然後調用SystemInit，且可能包含用戶應用程序的堆棧配置。

 **system\_<device>.c**和**system\_<device>.h**包含通用的系統配置信息（如時鐘和總線設置）。

 **<device.h>** 提供了可訪問的處理器核心和所有外設。該文件應該由SVD文件通過SVDConv.exe產生，以此來確保頭文件和調試器顯示的一致性。

與用戶代碼有關的系統和啟動文件

**注意**

CMSIS-CORE闡明了系統和啟動文件的結構以及其創建方式。

復制Pack\_with\_Device\_Support.zip中的02\_System\_and\_Startup目錄到您的工作環境中的文件目錄中。

1. 取消PDSC文件中對<conditions>部分的注釋，並添加以下內容（該conditions部分提供了該步驟的詳細信息）：

<conditionid="MVCM3 CMSIS-CORE">

<!-- conditionsselecting Devices -->

<description>MyVendorMVCM3 Series devices and CMSIS-CORE</description>

<requireCclass="CMSIS" Cgroup="CORE"/>

<requireDvendor="Generic:5" Dname="MVCM3\*"/>

</condition>

<conditionid="Startup ARM">

<description>Startupassembler file for ARMCC</description>

<requireTcompiler="ARMCC"/>

</condition>

<conditionid="Startup GCC">

<description>Startupassembler file for GCC</description>

<require Tcompiler="GCC"/>

</condition>

<conditionid="Startup IAR">

<description>Startupassembler file for IAR</description>

<requireTcompiler="IAR"/>

</condition>

注意

基於匯編的startup\_<device>.s文件是和工具有關的。因此，對於各工具供應商，單獨的條件是必需的。

2. 取消PDSC文件中對<components>部分的注釋，並添加以下內容（該components部分提供了該步驟的詳細信息）：

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

<description>SystemStartup for MyVendor MVCM3 Series</description>

<files>

<!-- include folder-->

<filecategory="include" name="Device\Include\"/>

<filecategory="source"name="Device\Source\ARM\startup\_MVCM3xxx.s" attr="config"condition="Startup ARM"/>

<filecategory="source"name="Device\Source\GCC\startup\_MVCM3xxx.s" attr="config"condition="Startup GCC"/>

<filecategory="source"name="Device\Source\IAR\startup\_MVCM3xxx.s" attr="config"condition="Startup IAR"/>

<filecategory="source" name="Device\Source\system\_MVCM3xxx.c"attr="config" />

</files>

</component>

3. 添加一個新的版本號：

<releaseversion="1.0.1">

Startup files included

</release>

4. 最後，保存PDSC文件並用gen\_pack.bat腳本生成包文件。然後安裝這個包到您的開發工具中並創建一個新的工程。選擇軟元件::CMSIS:CORE和::Device:Startup給該工程：

啟動和系統文件添加到了工程中

**系統視圖描述文件**

**CMSIS-SVD**用來書面地描述包含基於ARM Cortex-M處理器的微控制器系統，尤其是外設寄存器的內存映射的程序員視圖。系統視圖中包含描述的詳細程度足以與由芯片廠商公佈的器件參考手冊中的描述媲美。信息范圍從上層的外設功能描述一直到底層內存映射寄存器中單個位域的定義和功能。由芯片供應商對CMSIS-SVD文件進行制定和維護。將基於XML的SVD文件的輸入到SVDConv.exe來生成依賴於調試視圖和器件頭文件的工具。

復制Pack\_with\_Device\_Support.zip中的03\_SVD\_File目錄到您的工作環境中的文件目錄中。

1. 用一個編輯器打開Files\SVD目錄中的MVCM3xxx.svd文件並按如下進行更改：

<deviceschemaVersion="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>ARM32-bit Cortex-M3 Microcontroller based device, CPU clock up to 100MHz.</description>

2. 在您的工作目錄中打開一個命令行窗口並輸入：

C:\temp\working\_dfp>SVDConv.exeFiles\SVD\MVCM3xxx.svd --generate=header –-fields=macro

3. 您應該能看到類似於這樣的一些SVDConv輸出：

CMSIS-SVD SVD ConsistencyChecker / Header File Generator V2.82g

Copyright (C) 2010 - 2014ARM 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 SVDSchema V1.1

Found 0 Errors and 0Warnings

Return Code: 0 (OK)

將生成的頭文件MVCM3xxx.h移動到Device\Include目錄中。

1. 向其PDSC文件的系列級區域添加如下兩行代碼：

<compileheader="Device\Include\MVCM3xxx.h"/>

<debugsvd="SVD\MVCM3xxx.svd"/>

5. 添加一個新的版本號：

<releaseversion="1.0.2">

SVD and header fileincluded

</release>

6. 最後，保存PDSC文件並用gen\_pack.bat腳本生成包文件。然後安裝這個包到您的開發工具中並創建

工程中的SVD文件

**注意**

更多關於CMSIS-SVD的信息，請訪問www.keil.com/cmsis/svd

**Flash編程算法**

Flash編程算法用於擦除或下載應用程序到Flash器件。個DFP通常包含預定義的Flash算法來對它支持的器件編程。算法功能頁面詳盡地闡述了該機制。

對於MVCM3系列的器件，需要創建兩種flash算法。MVCM3110/250擁有16kB的Flash,而MVCM3120/260擁有32kB的Flash存儲。

復制Pack\_with\_Device\_Support.zip中的04\_Flash\_Programming目錄到您的工作環境中的文件目錄中。

1. 重命名工程文件NewDevice.uvproj（在目錄\_Template\_Flash中）來表示新的Flash ROM器件名稱，例如MVCM3XXX\_16.uvproj。

2. 用uVision打開該工程。選好目標(Cortex-M)器件。

注意

MDK-Lite不支持創建Flash編程算法。

3. 打開會話框Project - Options for Target -Output並且更改Name of Executable域的內容來表示當前器件，這裡用MCVM3XXX\_16。

'Cortex-M'目標選項

4. 事實上，現在您就可以開始調整文件FlashPrg.c中的編程算法。現在只更改文件FlashPrg.c中的器件參數（器件名稱，器件大小以及扇區大小）：

struct FlashDevice constFlashDevice = {

FLASH\_DRV\_VERS, // DriverVersion, do not modify!

"MVCM3110/250Flash", // Device Name

ONCHIP, // Device Type

0x00000000, // Device StartAddress

0x00004000, // Device Sizein Bytes (16kB)

1024, // Programming PageSize

0, // Reserved, must be 0

0xFF, // Initial Content ofErased Memory

100, // Program PageTimeout 100 mSec

3000, // Erase SectorTimeout 3000 mSec

// Specify Size and Addressof Sectors

0x002000, 0x000000, //Sector Size 8kB (2 Sectors)

SECTOR\_END

};

5. 用Project - Build Target來生成新的Flash編程算法。算法會在當前目錄下的\_Template\_Flash目錄中被創建。

6. 復制輸出文件..\MVCM3XXX\_16.FLM到一個新的稱為Files\Flash的子目錄下。添加這行代碼到MVCM3110/250的device部分：

algorithmname="Flash\MVCM3XXX\_16.FLM" start="0x00000000"size="0x4000" default="1"/>

7. 為MVCM3120/260器件創建一個MVCM3XXX\_32.flm文件。

8. 添加這行代碼到MVCM3120/260的device部分：

algorithmname="Flash\MVCM3XXX\_32.FLM" start="0x00000000"size="0x8000" default="1"/>

9. 添加一個新的版本號：

<releaseversion="1.0.3">

Flash ProgrammingAlgorithms added

</release>

10. 最後，保存PDSC文件並用gen\_pack.bat腳本生成包文件。然後安裝這個包到您的開發工具中。根據您的開發環境，在您的工程中將會看到Flash編程算法：

Flash編程算法的顯示

**器件屬性**

為了減少冗餘，可將器件分為兩個級別的分組：

 系列：一個器件系列擁有相同的處理器的屬性。

 子系列：一個可選的子系列屬性；這裡增加或修改一個系列的特徵。

某具體器件屬性規定了：

 器件：具體半導體器件的屬性

 變種：器件變體的屬性（例如，擁有不同的封裝或溫度范圍）或者OEM器件或者集成器件的電路板。

一個器件由如下屬性來進行描述：

 描述：文本形式的器件描述

 特徵：器件的外設和特徵分類表。該列表用於在網站上顯示器件的特徵。

 手冊：關於器件以及其處理器的文檔

 處理器：嵌入到器件中的處理器和處理器的特徵

 編譯：器件支持的構建工具的常規設置

 調試配置：調試鏈接的默認配置

 調試端口：調試器進行調試連接配置時器件的調試端口的描述。

 調試：調試器進行調試連接配置時器件的特定信息，包括系統視圖描述文件。

 跟蹤：調試器在配置跟蹤時器件的特定信息。

 存儲器：器件內部和外部RAM和ROM的存儲器區域佈局

 算法：器件特定的Flash編程算法

一個器件將會同時繼承來自系列級和子系列級的技術參數。一些屬性必需是唯一的。例如<debug>屬性中SVD文件名屬性值。因此，在SVD文件中由系列級指定的屬性值可以被子系列級或者器件級的屬性值重新定義。在系列級以及子系列級中的比如描述以及特徵項等信息會被級聯在一起，然後最終確定這些器件級的信息。

接下來將介紹如何指定器件MVCM3110的器件屬性（請參閱基本器件系列包中所示的技術參數）。該系列的其它成員的器件屬性也相應地被指定。

復制Pack\_with\_Device\_Support.zip中的05\_Device\_Properties目錄到您的工作環境中的文件目錄中：

1. MVCM3 Series系列的處理器編譯描述和調試屬性已經被指定了。這裡對系列級的其它公共的手冊和特徵屬性進行指定。添加下面的代碼行到其PDSC文件的<family>部分中：

<bookname="Docs\dui0552a\_cortex\_m3\_dgug.pdf" title="Cortex-M3 GenericUser Guide"/>

<book name="Docs\MVCM3XXX\_Datasheet.pdf"title="MVCM3 Series Datasheet"/>

<bookname="Docs\MVCM3XXX\_Product\_Brief.pdf" title="MVCM3 ProductBrief"/>

<featuretype="ExtInt" n="16"/>

<featuretype="Temp" n="-40" m="105" name="ExtendedTemperature Range"/>

<featuretype="VCC" n="2.5" m="3.6"/>

<featuretype="RTC" n="32768"/>

<featuretype="WDT" n="1"/>

2. MVCM31xx子系列有些特徵是其兩個成員器件共有的。請將下面的代碼添加到MVCM3100的<subFamily>部分：

<featuretype="IOs" n="26"/>

<featuretype="I2C" n="1"/>

<featuretype="UART" n="4"/>

<featuretype="Timer" n="6" m="32"/>

<featuretype="QFP" n="32"/>

3. MVCM3110器件具有一些獨有的器件特徵。請將下面的代碼添加到MVCM3110的< device >部分：

<featuretype="PWM" n="2" m="16"/>

**器件特有的軟元件**

最後，需要將軟元件和工程示例添加到DFP中。

1. 將下面的代碼行添加到其PDSC文件的**<components>**部分：

<componentCclass="Device" Cgroup="HAL" Csub="GPIO"Cversion="1.0.0" condition="MVCM3 CMSIS-CORE">

<description>GPIO HALfor MyVendor MVCM3 Series</description>

<files>

<filecategory="header" name="Device\Include\GPIO.h"/>

<filecategory="source" name="Device\Source\GPIO.c"/>

</files>

</component>

<componentCclass="Device" Cgroup="HAL" Csub="ADC"Cversion="1.0.0" condition="MVCM3 CMSIS-CORE">

<description>ADC HALfor MyVendor MVCM3 Series</description>

<files>

<filecategory="header" name="Device\Include\ADC.h"/>

<file category="source"name="Device\Source\ADC.c"/>

</files>

</component>

<componentCclass="CMSIS Driver" Cgroup="I2C"Cversion="1.0.0" condition="MVCM3 CMSIS-CORE"maxInstances="3">

<description>I2CDriver 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>

<filecategory="source" name="Drivers\I2C\I2C\_MVCM3.c"/>

<filecategory="header" name="Drivers\I2C\I2C\_MVCM3.h"/>

</files>

</component>

<componentCclass="CMSIS Driver" Cgroup="UART"Cversion="1.0.0" condition="MVCM3 CMSIS-CORE"maxInstances="5">

<description>UARTDriver 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>

<filecategory="source" name="Drivers\UART\UART\_MVCM3.c"/>

<filecategory="header" name="Drivers\UART\UART\_MVCM3.h"/>

</files>

</component>

注意

前兩個軟元件被添加到**Device::HA**L是因為它們是器件系列特有的且未使用發布的API。I2C和UART的驅動秉承了CMSIS-Driver規范。因此，把它們添加到了CMSIS Driver器件類中。

2. 同樣創建一個工程示例。將**<examples>**部分注釋取消並添加這些：

<examplename="Dummy" doc="Abstract.txt"folder="Examples\dummy">

<description>Dummyproject</description>

<board name="MVCM3Starter Kit" vendor="MyVendor"/>

<project>

<environmentname="uv" load="dummy.uvprojx"/>

</project>

<attributes>

<category>GettingStarted</category>

</attributes>

</example>

3. 添加一個新的版本號：

<releaseversion="1.0.4">

DFP finalized

</release>

4. 最後，保存PDSC文件並用gen\_pack.bat腳本生成包文件。然後安裝這個包到您的開發工具中。

**板級支持包**

包含**<board>**元素軟件包被稱為板級支持包（BSP）。一個BSP可能包含其他軟件組件，示例項目，以及代碼模板。一個BSP的典型內容是：

• 源代碼,庫，為底層硬件和文檔准備的頭文件/配置文件（例如用戶手冊，入門指南和原理圖）。

• 示例項目，展示了開發板及其外圍設備的使用情況。

• 用戶代碼模板可以用來作為起點用於使用開發板或安裝。

這部分是一個教程，介紹了如何創建一個BSP。最初在BSP的PDSC文件只是描述一個評估板。這是BSP然後擴展到還含有軟件組件與電路板的硬件接口。示例項目和代碼模板可以像軟件組件包描述的那樣添加進來。

**BSP使用案例**

一個BSP通常由電路板廠商提供，用於擴展開發工具。在一般情況下，BSP能夠使電路板供應商來分發他們的開發平台工具，BSP還可以在用於提供信息有關的網站上顯示。其中一個例子是www.keil.com/boards2/：

**指定一個開發板**

**准備工作**

1. 安裝軟件包的Keil::ARMCortex\_DFP，因為需要這，BSP才能正常工作。

2. 創建一個工作目錄您的電腦上，例如C：\ TEMP\ working\_bsp。

3. 在ARM::CMSIS包目錄\ CMSIS\Pack\Tutorials注冊。請咨詢您的開發工具的文檔上的包安裝目錄結構的詳細信息。在一個標准的μVision安裝，你會發現它在下面的C：\Keil\_v5\ ARM\包\ ARM\ CMSIS\version

4. 打開文件Pack\_with\_Board\_Support.zip。

5. 復制ZIP文件的01\_Specify\_Dev\_Board目錄中的內容到你的工作目錄。

6. 確保文件/目錄沒有寫保護（刪除只讀標志）。

7. 從ARM::CMSIS 包注冊文件裡目錄 \CMSIS\Utilities的內容復制到您的工作目錄：

◦ PackChk.exe

◦ PACK.xsd

8. 在編輯器中打開MyVendor.MyBSP.pdsc文件。

代碼示例

1. 按照下面添加一個<boards>部分到PSDC文件：

<boards>

<board vendor="MyVendor" name="MyDevBoard" revision="V1" salesContact="sales@keil.com" orderForm="http://www.keil.com">

<description>MyVendorMyDevBoard Board Description</description>

<image small="Images\MyDevBoard\_small.png" large="Images\MyDevBoard\_large.png"/>

<book category="setup"name="Docs\MyDevBoard\MyDevBoard\_setup.pdf" title="GettingStarted"/>

<book category="schematic" name="Docs\MyDevBoard\MyDevBoard\_schematics.pdf" title="Schematics"/>

<book category="manual"name="Docs\MyDevBoard\MyDevBoard\_um.pdf" title="UserManual"/>

<mountedDevicedeviceIndex="0" Dvendor="ARM:82" Dname="ARMCM3"/>

<compatibleDevicedeviceIndex="0" Dvendor="ARM:82" Dfamily="ARMCortex M3"/>

<feature type="ODbg" n="1" name="On-board J-LINK Lite"/>

<feature type="PWR" n="5" name="USB Powered"/>

<feature type="DIO" n="8"/>

<feature type="SPI" n="1"/>

<feature type="LED" n="6" name="User LEDs"/>

<feature type="ArduinoFF" n="1"/>

<debugInterface adapter="J-Link Lite" connector="Mini-USB"/>

</board>

</boards>

注意

在此所有代碼的實施例和下面的各節可以在**01\_Specify\_Dev\_Board**目錄snippets.xml文件中找到。

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。請參閱生成包的進一步細節。之後，安裝包在你的開發工具。

開發板在開發工具中的顯示

注意

PackCheck.exe將顯示M346的信息：

\*\*\* WARNING M346:.\Files\MyVendor.MyBSP.pdsc (Line 29)

Referenced device(s) in 'MyDevBoard' notfound: 'ARMCM3'

\*\*\* WARNING M346:.\Files\MyVendor.MyBSP.pdsc (Line 29)

Referenced device(s) in 'MyDevBoard' notfound: 'ARM Cortex M3'

這是因為BSP不包含具有所提到的設備的設備部分。要解決此問題，您可以添加ARM：CMSIS PDSC文件檢查過程。

**創建一個BSP束**

一個束基本上是在Cclass級別的變體。它規定了相互依賴的組件的集合屬性類，版本和可選組和供應商。一個包內的組件繼承了包的屬性設置，不能再次設置這些屬性。束確保跨越多個互通組件屬性的一致性，從不同的解決方案Cclass內限制組件的混搭。除組件外，束有強制性內容description和doc（用於文檔）。

復制Pack\_with\_Board\_Support.zip文件裡的02\_BSP\_Bundle目錄的內容到您的工作環境中的文件目錄：

1. 如下添加<conditions>部分在PDSC（condition部分提供了有關此步驟的詳細信息）：

<condition id="ARM Cortex M3 Series">

<description>ARM Cortex M3 devicerequired</description>

<require Dfamily="ARM Cortex M3" Dvendor="ARM:82"/>

</condition>

<condition id="MyDevBoard Board Setup">

<description>Board Setup Coderequired</description>

<require condition="ARM Cortex M3 Series"/>

<require Cclass="Board Support" Cgroup="BoardSetup"/>

</condition>

1. 添加<component>部分在PDSC與以下（component部分提供了有關此步驟的詳細信息）：

<bundle Cbundle="MyDevBoardSupport" Cclass="BoardSupport" Cversion="1.0.1">

<description>MyDevBoardcustom board support package</description>

<doc>Docs\BSP\MyBSP.htm</doc>

<component Cgroup="Board Setup" condition="ARMCortex M3 Series">

<description>Customboard setup code</description>

<files>

<file category="doc"name="Docs\BSP\Setup.htm"/>

<file category="header"name="MySWComp\Include\setup.h"/>

<file category="source"name="MySWComp\Source\setup.c"/>

</files>

</component>

<component Cgroup="I/O"Csub="LED" condition="MyDevBoardBoard Setup">

<description>LED code for customboard</description>

<files>

<file category="doc"name="Docs\BSP\LED.htm"/>

<file category="header"name="MySWComp\Include\led.h"/>

<file category="header"name="MySWComp\Include\config\_led.h" attr="config" version="1.0.0"/>

<file category="source"name="MySWComp\Source\led.c"/>

</files>

</component>

<component Cgroup="I/O"Csub="GPIO" condition="MyDevBoardBoard Setup">

<description>GPIO code for customboard</description>

<files>

<file category="doc"name="Docs\BSP\GPIO.htm"/>

<file category="header"name="MySWComp\Include\gpio.h"/>

<file category="header"name="MySWComp\Include\config\_gpio.h" attr="config" version="1.0.0"/>

<file category="source"name="MySWComp\Source\gpio.c"/>

</files>

</component>

</bundle>

1. 添加一個新的版本號

<release version="1.0.1">

Board support bundle added

</release>

1. 最後，保存PDSC文件，然後使用gen\_pack.bat腳本生成包文件。之後，安裝包在你的開發工具，並觀察這些差異1.0.0版本。

在開發工具中顯示BSP束