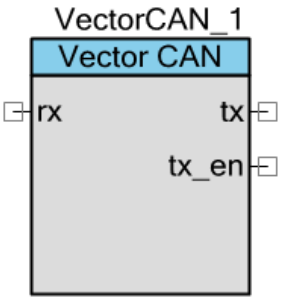
Vector CAN
1.10

特性



- CAN2.0 A/B 协议实现，符合 ISO 11898-1 标准
- 高达 1 Mbps @ 8 MHz (BUS_CLK)的可编程比特率
- 连接外部收发器的两线或三线接口（Tx、Rx 和 Tx 使能）
- 由 Vector 提供并支持的驱动程序

概述

Vector CANbedded 环境包含大量自适应源代码组件，这些组件包括汽车应用中的基本通信和诊断要求。

Vector CANbedded 软件套装是客户特定的套装，其操作会因应用和 OEM 而有所不同。写入此 Vector CANbedded 套装组件，通常可支持 CANbedded 结构，无论特定 OEM 应用的风格如何。

使用针对 PSoC3 开发的 Vector CAN 组件，可轻松集成经 Vector 认证的 CAN 驱动程序。

何时使用 Vector CAN

在需要针对由 Vector 提供的 PSoC 3 进行 CAN 驱动程序集成时，使用 Vector CAN 组件。

输入/输出连接

本节介绍了 Vector CAN 组件的各种输入和输出连接。I/O 列表中的星号（*）表示，在 I/O 说明部分所罗列的各项中，该 I/O 可能不可见。

rx — 输入

CAN 总线接收信号（连接到外部收发器的 CAN RX 总线）。

tx — 输出

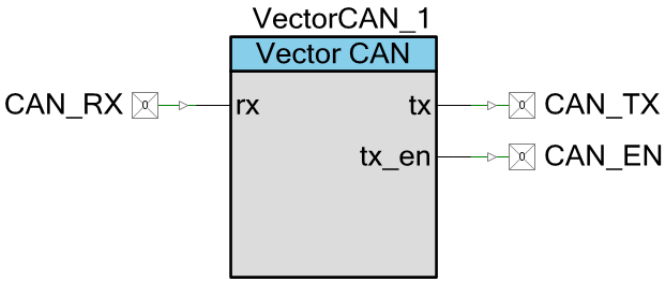
CAN 总线传输信号（连接到外部收发器的 CAN TX 总线）。

tx_en — 输出*

外部收发器使能信号。当在 **Configure** 对话框中选择 **Add Transceiver Enable Signal**（添加收发器使能信号）选项时，将显示此输出。

原理图宏的信息

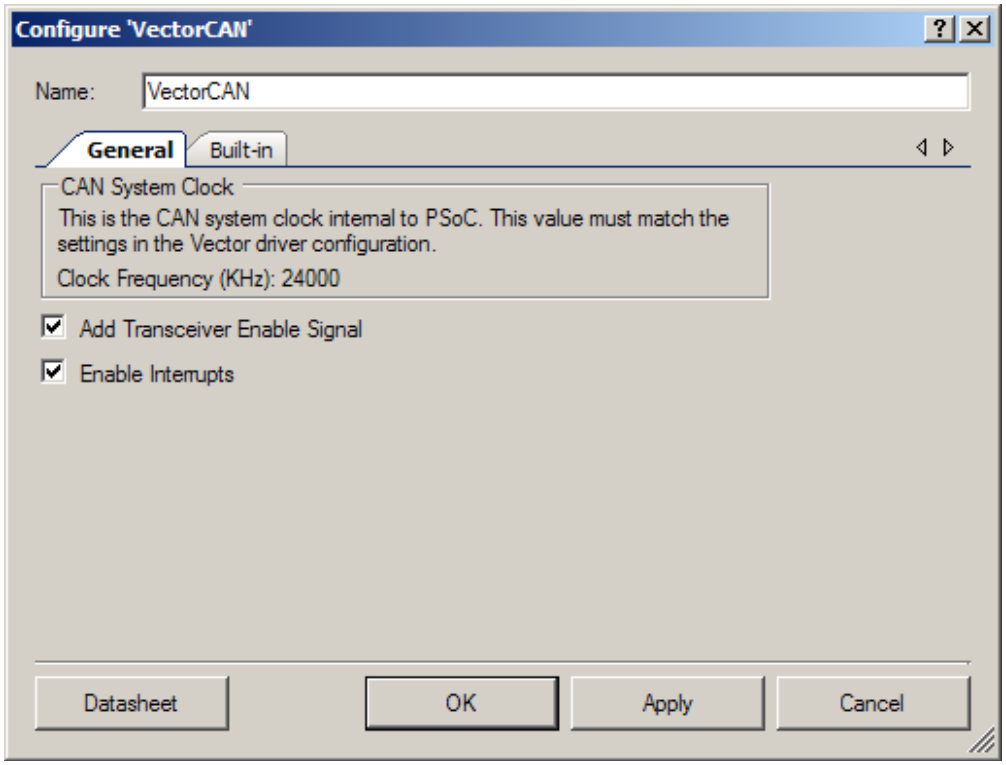
组件目录中的默认 Vector CAN 是一个使用带默认设置的 Vector CAN 组件的原理图宏。Vector CAN 组件连接到输入和输出引脚组件。除了输入引脚组件中的“Input Synchronized”（输入同步）被设为假以外，引脚组件也设置为默认设置。



组件参数

将一个 Vector CAN 组拖入您的设计中，并双击以打开 **Configure** 对话框。此对话框有一个 **General**（通用）选项卡，可指导您设置 Vector CAN 组件的流程。

图 1. 配置 Vector CAN 对话框



General（常规）选项卡提供以下参数。

Add Transceiver Enable Signal（添加收发器使能信号）

启用/禁用针对外部 CAN 收发器的‘tx_en’信号的使用。默认设置为 **Enable**（使能）。

Enable Interrupts（使能中断）

使能/禁用 CAN 中的全局中断。默认设置为 **Enable**（使能）。如果针对轮询模式配置驱动程序文件，则应禁用此参数（否则，编译过程会出现错误）。

时钟选择

Vector CAN组件连接到BUS_CLK时钟信号。最小值必须为8 MHz，以支持所有高达1 Mbps的标准CAN波特率。PSoC 3项目设计范围资源中所选的BUS_CLK值必须与针对总线时序的Vector CAN驱动程序配置中选定的值相同。

应用编程接口

通过应用编程接口（API）子程序，您可以使用软件对组件进行配置。下表列出并说明了每个函数的接口。以下各节将对每个函数加以说明。

默认情况下，PSoC Creator 将实例名称“Vector_CAN_1”分配给指定设计中组件的第一个实例。您可以将该实例重新命名为符合标识符语法规则的任意唯一值。实例名称会成为每个全局函数名称、变量和符号常量的前缀。为了便于阅读，下表中使用的实例名称为“Vector_CAN”。

函数

函数	说明
Vector_CAN_Start()	使用Vector_CAN_Init()和Vector_CAN_Enable()函数初始化并使能Vector CAN组件。
Vector_CAN_Stop()	禁用Vector CAN组件。
Vector_CAN_GlobalIntEnable()	从CAN内核中使能全局中断。
Vector_CAN_GlobalIntDisable()	从CAN内核中禁用全局中断。
Vector_CAN_Sleep()	为组件进入睡眠状态做准备。
Vector_CAN_Wakeup()	将组件恢复到调用Vector_CAN_Sleep()时的状态。
Vector_CAN_Init()	基于组件定制器中的设置对Vector CAN组件进行初始化。利用由Vector CAN配置工具生成的中断服务子程序CanIsr_0()设置CAN中断。
Vector_CAN_Enable()	使能Vector CAN组件。
Vector_CAN_SaveConfig()	保存组件配置。
Vector_CAN_RestoreConfig()	恢复组件配置。

uint8 Vector_CAN_Start(void)

说明： 这是开始执行组件操作的首选方法。Vector_CAN_Start()设置initVar变量，并依次调用Vector_CAN_Init()以及Vector_CAN_Enable()函数。

参数： 无

返回值： 指示寄存器是否已被写入及验证。

值	说明
CYRET_SUCCESS	函数已成功完成。
Vector_CAN_FAIL	函数执行失败。

其他影响： 无

uint8 Vector_CAN_Stop(void)

说明： 禁用Vector CAN组件。

参数： 无

返回值： 指示寄存器是否已被写入及验证。

值	说明
CYRET_SUCCESS	函数已成功完成。
Vector_CAN_FAIL	函数执行失败。

其他影响： 无

uint8 Vector_CAN_GlobalIntEnable(void)

说明： 此函数从CAN内核中使能全局中断。

参数： 无

返回值： 指示寄存器是否已被写入及验证。

值	说明
CYRET_SUCCESS	函数已成功完成。
Vector_CAN_FAIL	函数执行失败。

其他影响： 无

uint8 Vector_CAN_GlobalIntDisable(void)

说明： 此函数从CAN内核中禁用全局中断。

参数： 无

返回值： 指示寄存器是否已被写入及验证。

值	说明
CYRET_SUCCESS	函数已成功完成。
Vector_CAN_FAIL	函数执行失败。

其他影响： 无

void Vector_CAN_Sleep(void)

说明： 这是使组件准备进入睡眠模式时的首选子程序。Vector_CAN_Sleep()子程序保存当前组件的状态。然后它依次调用Vector_CAN_SaveConfig()和Vector_CAN_Stop()函数，以保存硬件配置。

在调用CyPmSleep()或CyPmHibernate()函数前，先调用Vector_CAN_Sleep()函数。

参数： 无

返回值： 无

其他影响： 无

void Vector_CAN_Wakeup(void)

说明： 该函数是将组件恢复到调用Vector_CAN_Sleep()时的状态的首选子程序。Vector_CAN_Wakeup()函数将调用Vector_CAN_RestoreConfig()函数，以恢复配置。如果组件在调用Vector_CAN_Sleep()函数前已使能，那么Vector_CAN_Wakeup()函数也将重新使能组件。

参数： 无

返回值： 无

其他影响： 调用Vector_CAN_Wakeup()函数前未调用Vector_CAN_Sleep()或Vector_CAN_SaveConfig()函数可能会产生意外行为。

void Vector_CAN_Init (void)

说明： 根据自定义程序“Configure”对话框设置，初始化或恢复组件。无需调用Vector_CAN_Init()，因为Vector_CAN_Start()子程序会调用该函数，这是开始组件操作的首选方法。此函数将利用Vector CAN配置工具所生成的中断服务子程序CanIsr_0()设置CAN中断。

参数： 无

返回值： 指示寄存器是否已被写入及验证。

值	说明
CYRET_SUCCESS	函数已成功完成。
Vector_CAN_FAIL	函数执行失败。

其他影响： 无

uint8 Vector_CAN_Enable(void)

说明： 激活硬件，开始组件操作。无需调用Vector_CAN_Enable()，因为Vector_CAN_Start()子程序会调用该函数，这是开始组件操作的首选方法。

参数： 无

返回值： 指示寄存器是否已被写入及验证。

值	说明
CYRET_SUCCESS	函数已成功完成。
Vector_CAN_FAIL	函数执行失败。

其他影响： 无

void Vector_CAN_SaveConfig(void)

说明： 此函数保存组件配置。它将保存非保留寄存器。此函数还将保存当前“Configure”对话框中所定义的或通过相应API修改的组件参数值。此函数将由Vector_CAN_Sleep()函数调用。

参数： 无

返回值： 无

其他影响： 无

void Vector_CAN_RestoreConfig(void)

说明： 此函数恢复组件配置。这将恢复非保留寄存器。该函数还会将组件参数值恢复为调用Vector_CAN_Sleep()函数之前的值。

参数： 无

返回值： 无

其他影响： 调用该函数前未调用Vector_CAN_Sleep()或Vector_CAN_SaveConfig()函数可能会产生意外行为。

全局变量

变量	说明
Vector_CAN_initVar	Vector_CAN_initVar指示Vector CAN是否已被初始化。变量将初始化为0，并在第一次调用Vector_CAN_Start()时被设置为1。这样，第一次调用Vector_CAN_Start()子程序后，组件不用重新初始化即可重启。如果需要重新对组件进行初始化，则可在调用Vector_CAN_Start()或Vector_CAN_Enable()函数前先调用Vector_CAN_Init()函数。

示例固件源代码

在“Find Example Project”对话框中，PSoC Creator 提供了大量的示例项目，包括原理图和示例代码。要获取组件特定的示例，请打开器件目录中的对话框或原理图中的器件实例。要查看通用示例，请打开“Start Page”或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

更多信息，请参考《PSoC Creator 帮助》中主题为“查找示例项目”一节的内容。

MISRA 合规性

本节介绍了MISRA-C:2004合规性和本器件的偏差情况。定义了下面两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 — 仅适用于该组件的偏差

本节提供了有关组件特定偏差的信息。*《系统参考指南》*的“MISRA 合规性”章节中介绍了项目偏差以及有关 MISRA 合规性验证环境的信息。

Vector CAN 组件没有任何特定偏差。

该组件具有下面的嵌入式组件：中断和时钟组件。MISRA 合规性与特定偏差的相关信息，请参见相应组件数据手册。

API 的内存使用量

根据编译器、器件、所使用的 API 数量以及组件的配置不同，组件的内存使用量也不一样。下表提供了在某种器件配置中所有 API 占用存储器的大小。

数据是在将编译器设置为 Release 模式并将优化等级设置为 Size 的情况下测得的。对于特定的设计，通过分析编译器生成的映射文件后可以确定存储器的使用情况。

配置	PSoC 3（Keil_PK51）	
	闪存 字节	SRAM 字节
默认值	728	18

中断服务子程序

Vector 驱动程序使用 CAN 中断，允许您访问它。Vector_CAN_Init()函数利用 Vector CAN 配置工具所生成的中断服务子程序 CanIsr_0()设置 CAN 中断。

功能说明

有关完整的 CAN 内核说明，请参见 *《术参考手册》* 中的“控制器区域网络（CAN）”章节。

有关 Vector GENy 工具的完整说明，请参见 Vector GENy 工具文档。

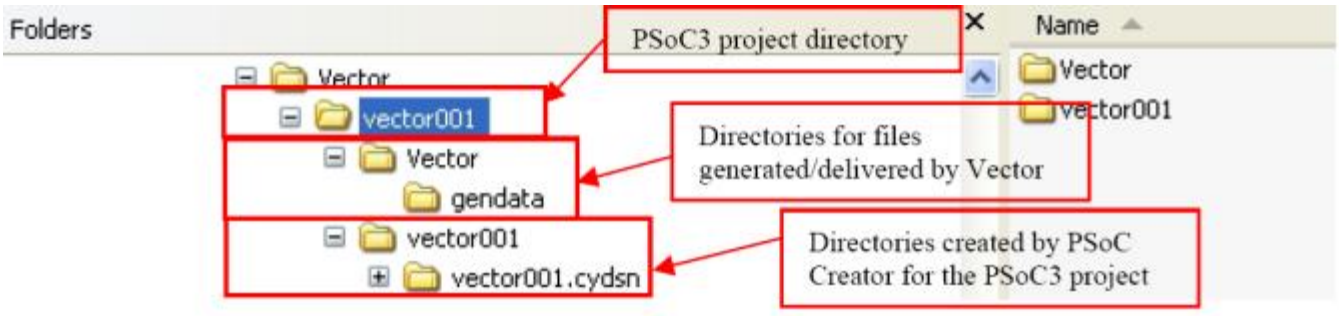
使用 Vector GENy 工具创建项目。

创建 CAN 驱动程序文件的空白 PSoC 项目和目录。

在使用 Vector GENy 工具生成 CAN 驱动程序文件前，应创建一个 PSoC 项目，以便 Vector GENy 工具可将驱动程序文件直接放在 PSoC 项目目录中。PSoC 项目此时将为空。

在 PSoC 项目目录中，为 CAN 驱动程序生成的文件创建一个目录，如图 2 所示。

图 2. PSoC 项目目录



生成 PSoC 应用的 Vector CAN 驱动程序文件

Vector GENy 工具基于以下项目为 PSoC 生成 CAN 驱动程序文件：

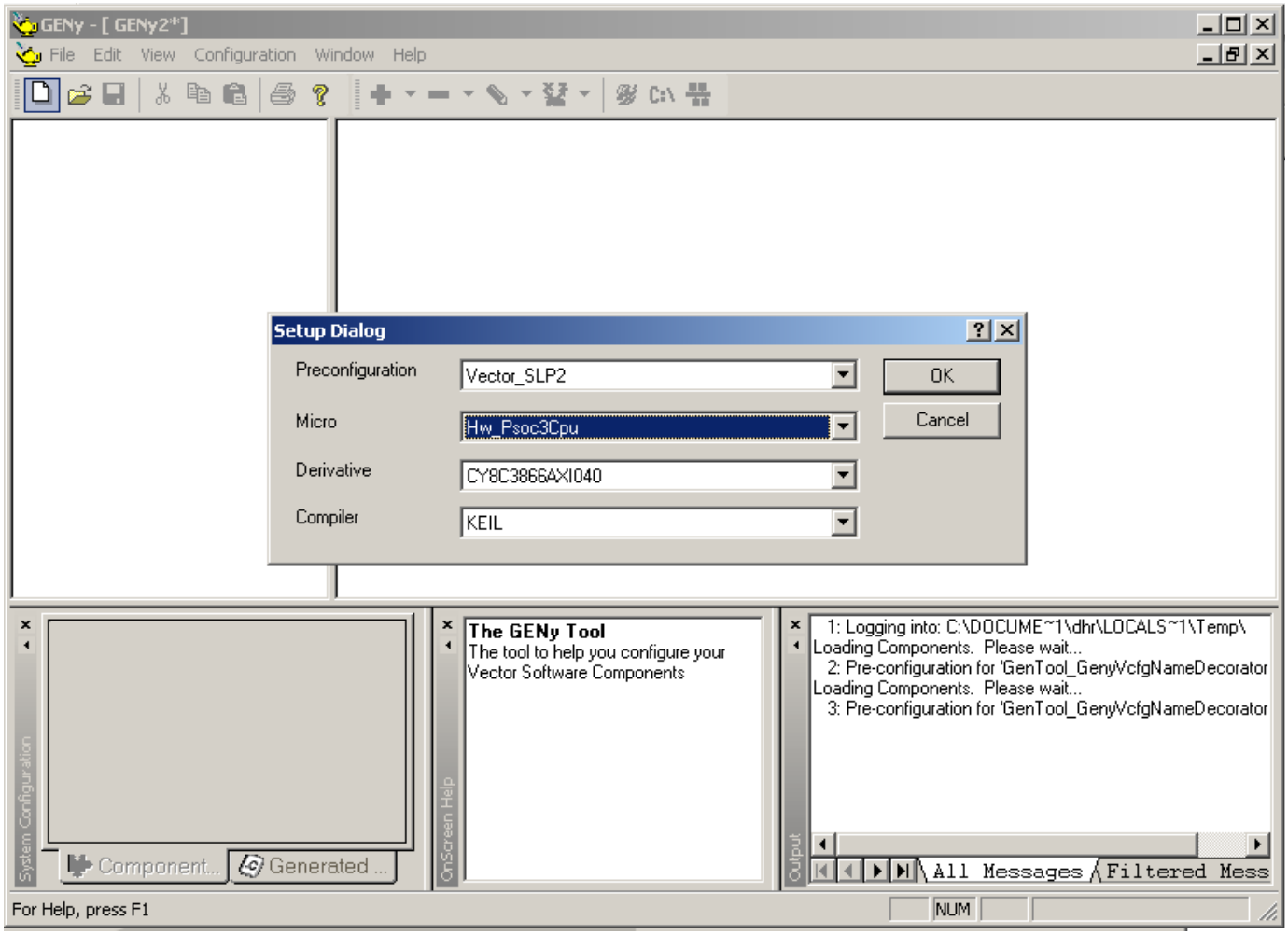
- CAN 应用信息数据库
- Vector GENy 工具中的配置

加载数据库后，您可配置 CAN 驱动程序生成工具（GENy）以生成驱动程序，从而处理 VectorCAN 信息。

打开 Vector GENy 工具，并创建新配置

1. 依次选择 Start > All Programs > Vector GENy 1.4 > GENy，启动 Vector GENy。
2. 单击“**New**”（新建）按键（请参见图 3）。

图 3. 单击“New”按键后的 Vector GENy 工具



1. 单击“**New**”按键时显示的 **Setup Dialog**（设置对话框）除了所显示的选项以外没有其他选项，所以单击 **OK**。

生成 CAN 驱动程序文件前设置配置

加载数据库后，通过以下步骤配置 PSoC 驱动程序（这只是“入门”配置）：

1. 选择 PSoC CAN 模块组件。
2. 选择驱动程序将使用的 Tx 和 Rx 信息。
3. 选择总线时序和信息接受滤波器。
4. 选择轮询模式或中断模式的选项。
5. 选择用于放置所生成文件的目标目录。

6. 选择 Tx 和 Rx 信息后，选择并配置接受滤波器和总线时序，如下图所示。

下图显示了这些步骤。

图 4. 接受滤波器和总线时序

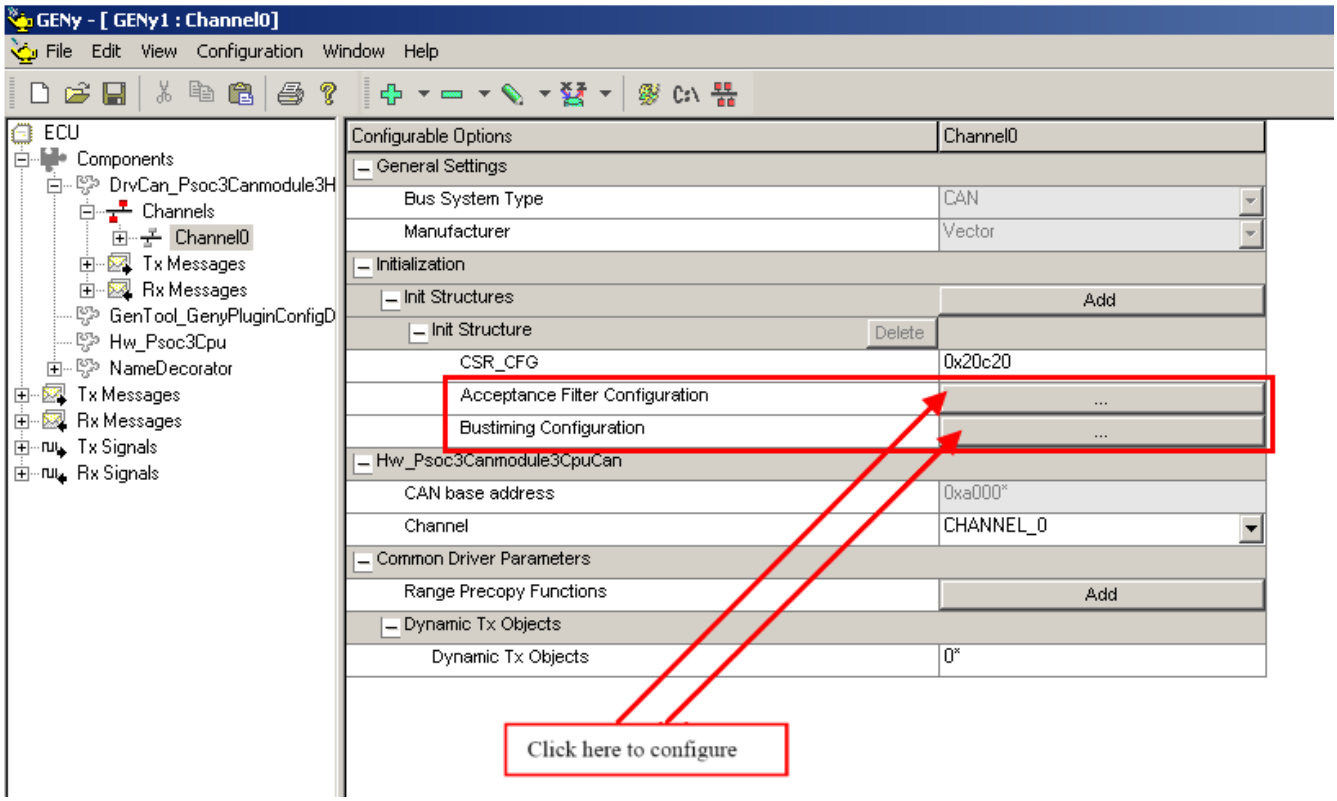
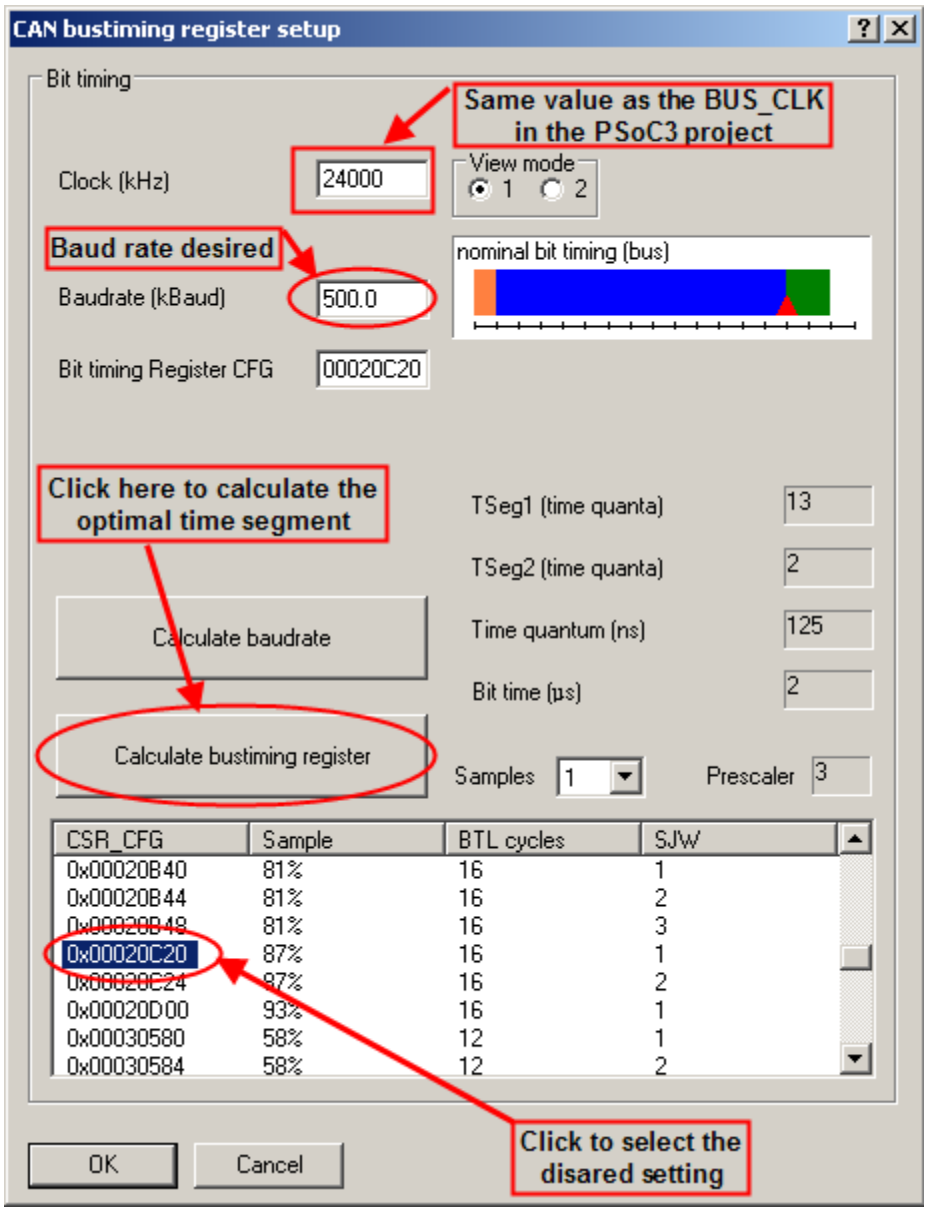


图 5. 总线时序设置



CAN bustiming（CAN 总线时序）窗口中选定的时钟频率必须与 PSoC 3 项目中的 BUS_CLK 频率相同。

选择轮询模式或中断模式时，请注意在轮询模式下，应用必须调用 CanTask()函数，以处理挂起事件或信息。在中断模式下，事件或信息由中断服务子程序 CanIsr_0()处理。

设置完接受滤波器、总线时序以及邮箱轮询/中断模式后，选择 CAN 驱动程序文件目录。

在 PSoC 项目目录中创建这些目录（如上面所述），使与此项目相关的所有项目都处于同一位置。

图 6. 选择轮询模式或中断模式

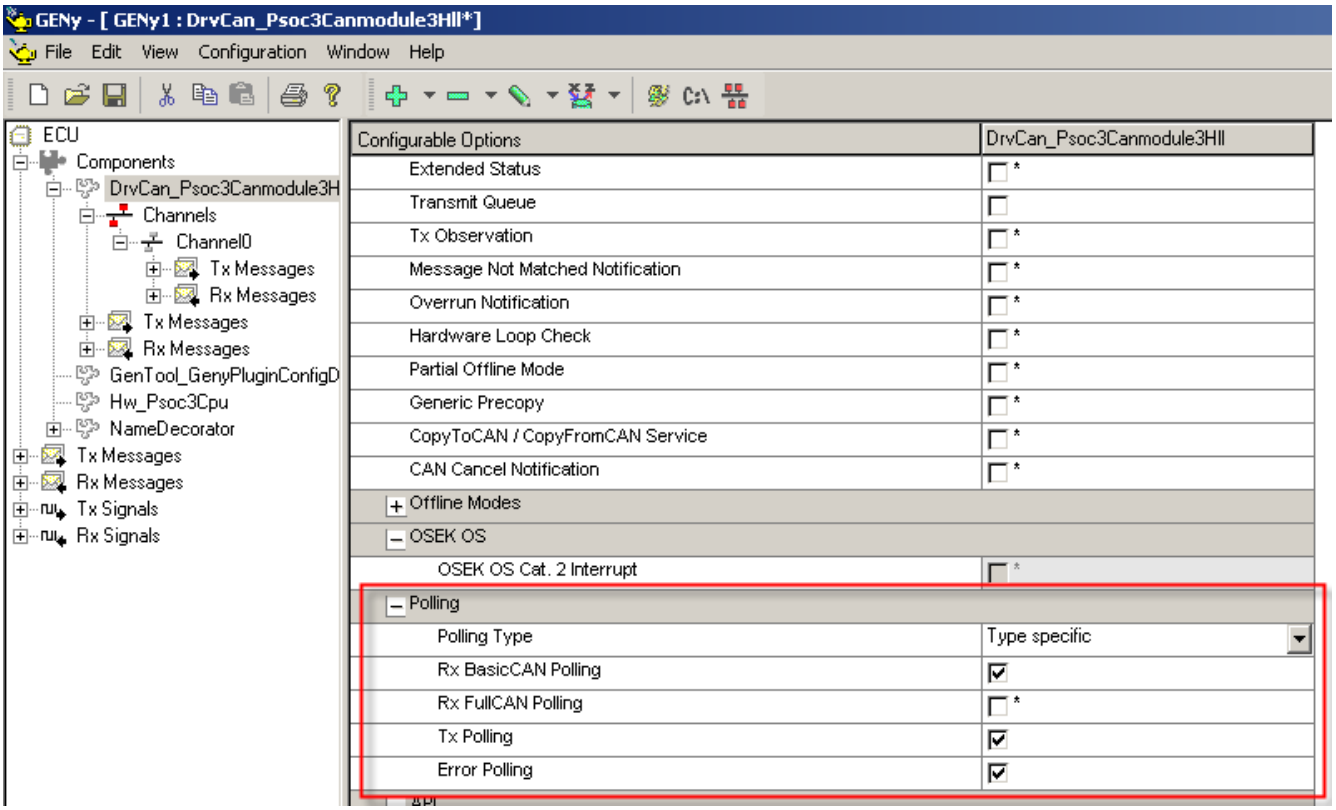


图 7. 选择目标目录

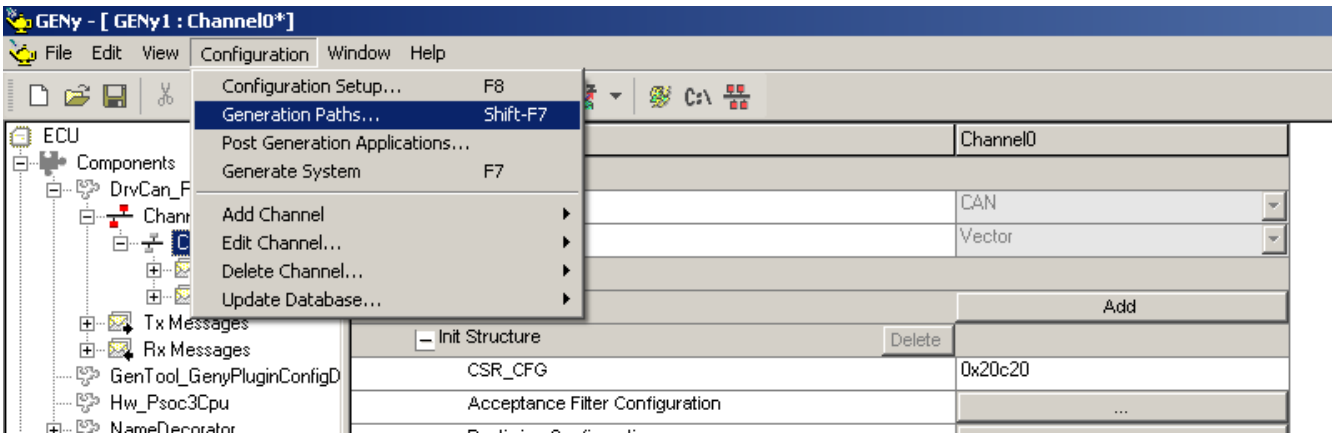
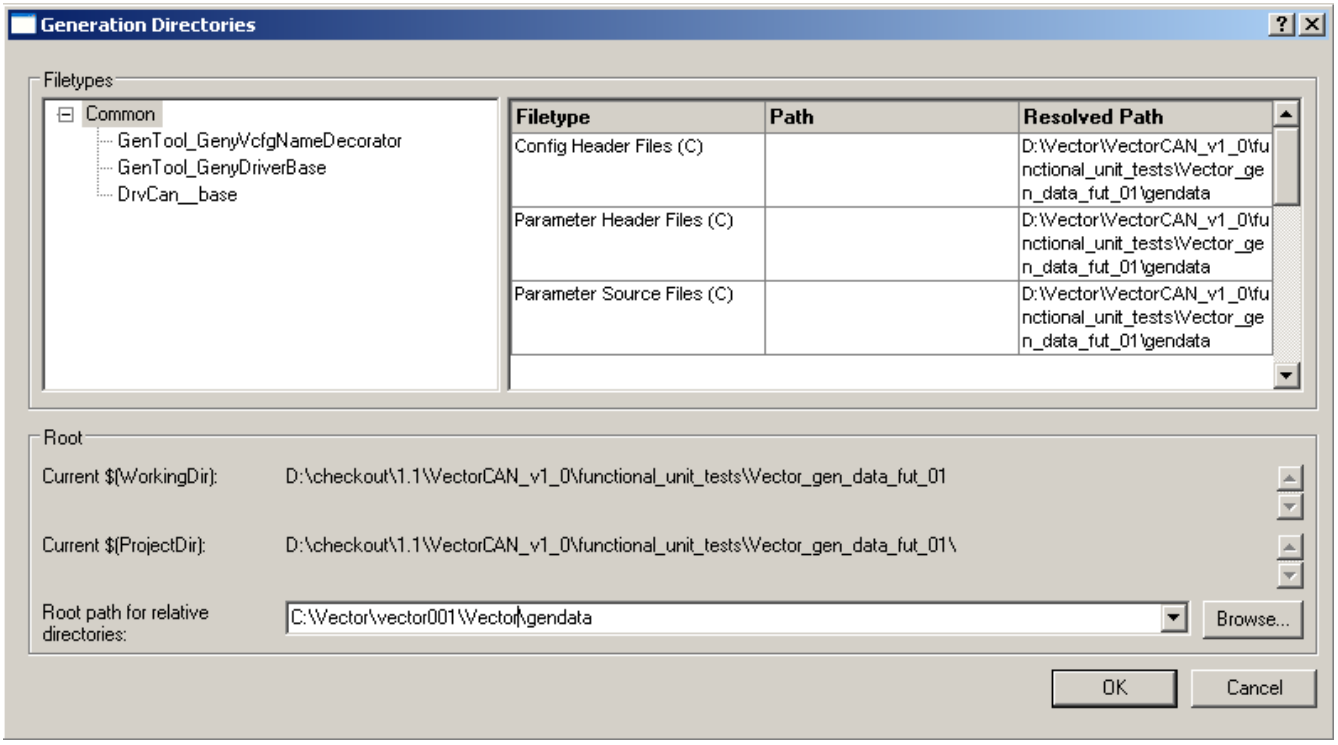


图 8. 选择“Root Path for the Relative Directories”（相对目录的根路径）



在 **Generation Directories**（生成目录）对话框窗口中，将 **Root path for relative directories**（相对目录的根路径）设为在 PSoC 项目目录中创建的子目录。

生成 CAN 驱动程序文件

此时，该工具已对生成 PSoC 驱动程序文件做好准备，这些文件将被放置在 PSoC 项目中所创建的选定目录中。

您也应该将配置文件保存在 PSoC 项目目录中，以便在需要修改并生成新文件时，所有项目都位于同一项目目录中。

每当按下 **Generate System**（生成系统）按键时都会生成这些文件，但为方便起见，也应该将一些（未更改的）通用文件复制到同一 PSoC 项目目录下。

下图说明了移至 PSoC Creator 中构建项目之前的最终步骤。

图 9. 单击“Generate System”以在 PSoC 项目子目录中生成文件

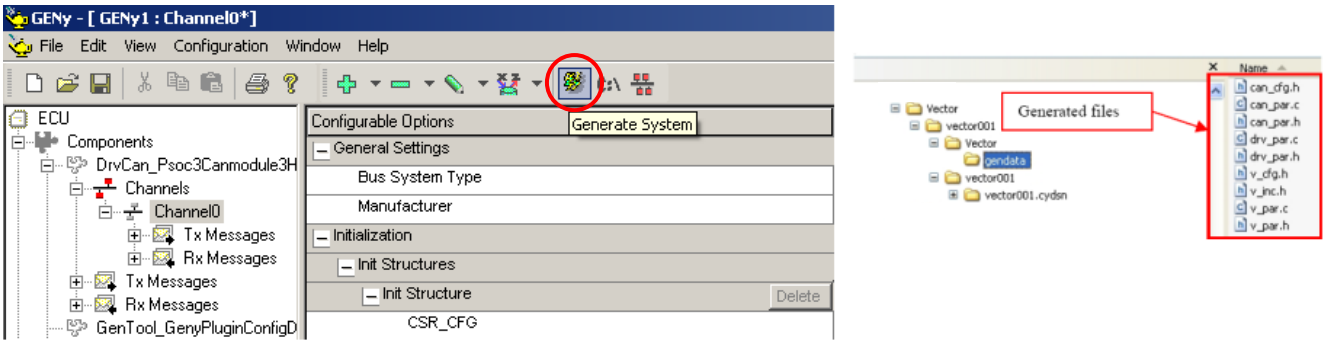
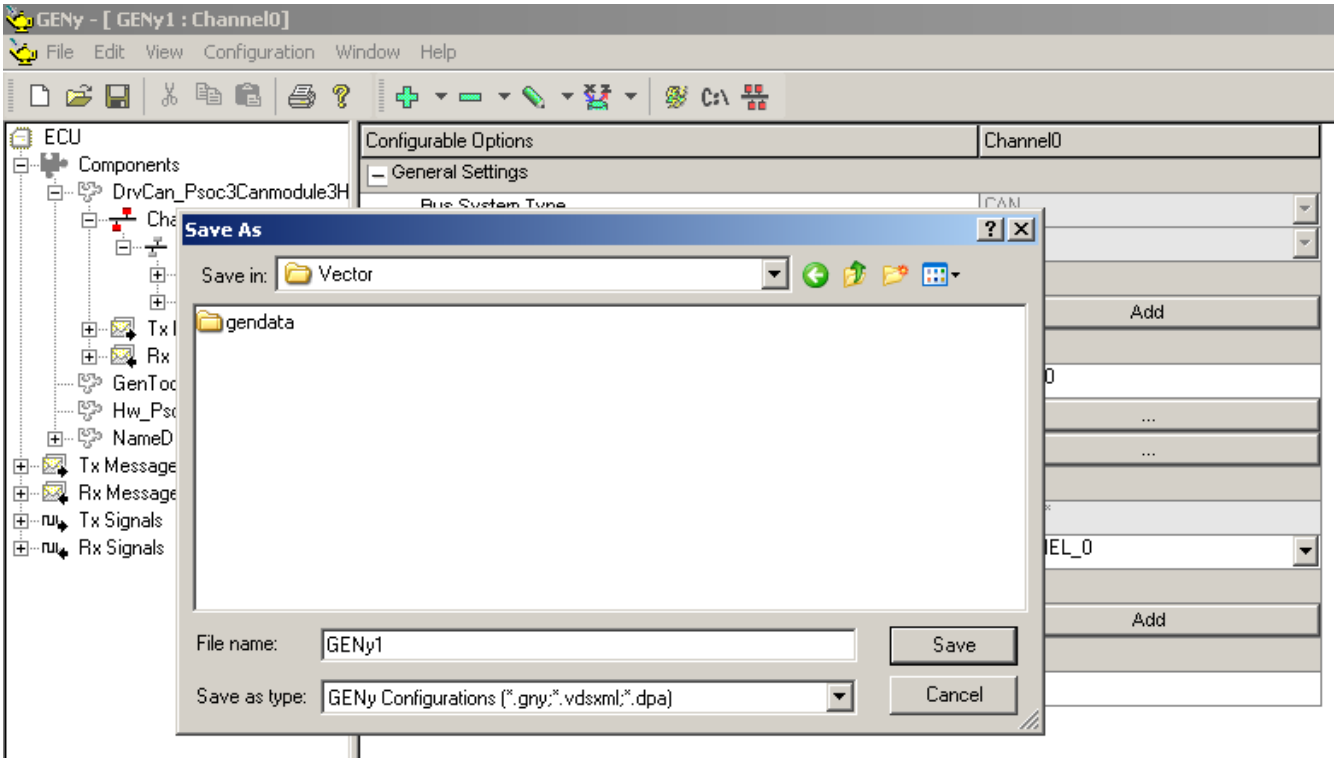


图 10. 在 PSoC 项目子目录中保存 Vector GENy 配置

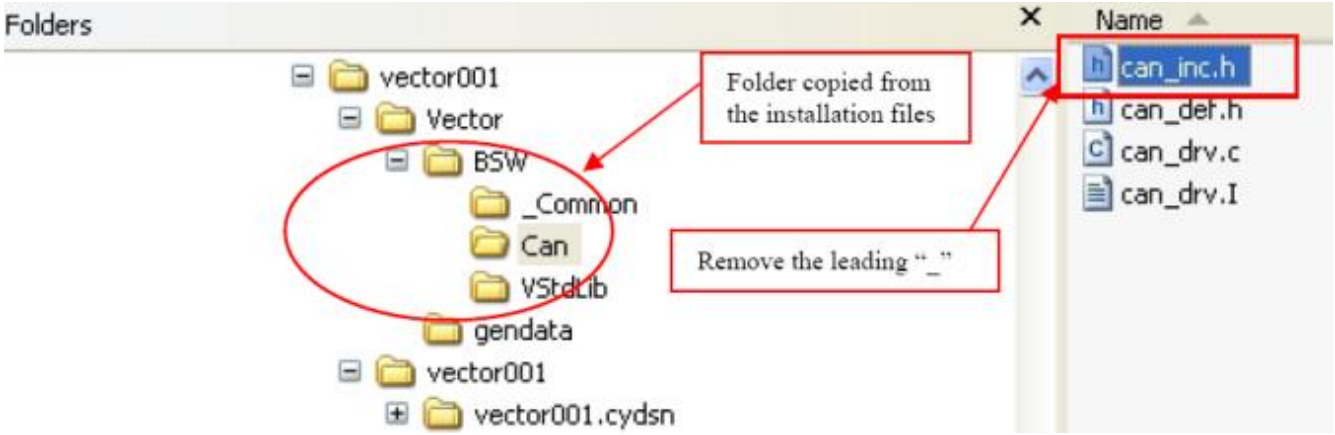


在安装过程中，如果接受了默认目录，通用文件位于：

`C:\Vector\CBD0810092_D00_Psoc3\BSW`

应将整个 BSW 目录复制到 PSoC 项目文件夹中，CAN 子目录中的*_can_inc.h* 文件名应更改为 *can_inc.h*（即移除前导“_”）。

图 11. CAN 驱动程序通用目录复制到 PSoC 项目文件夹中



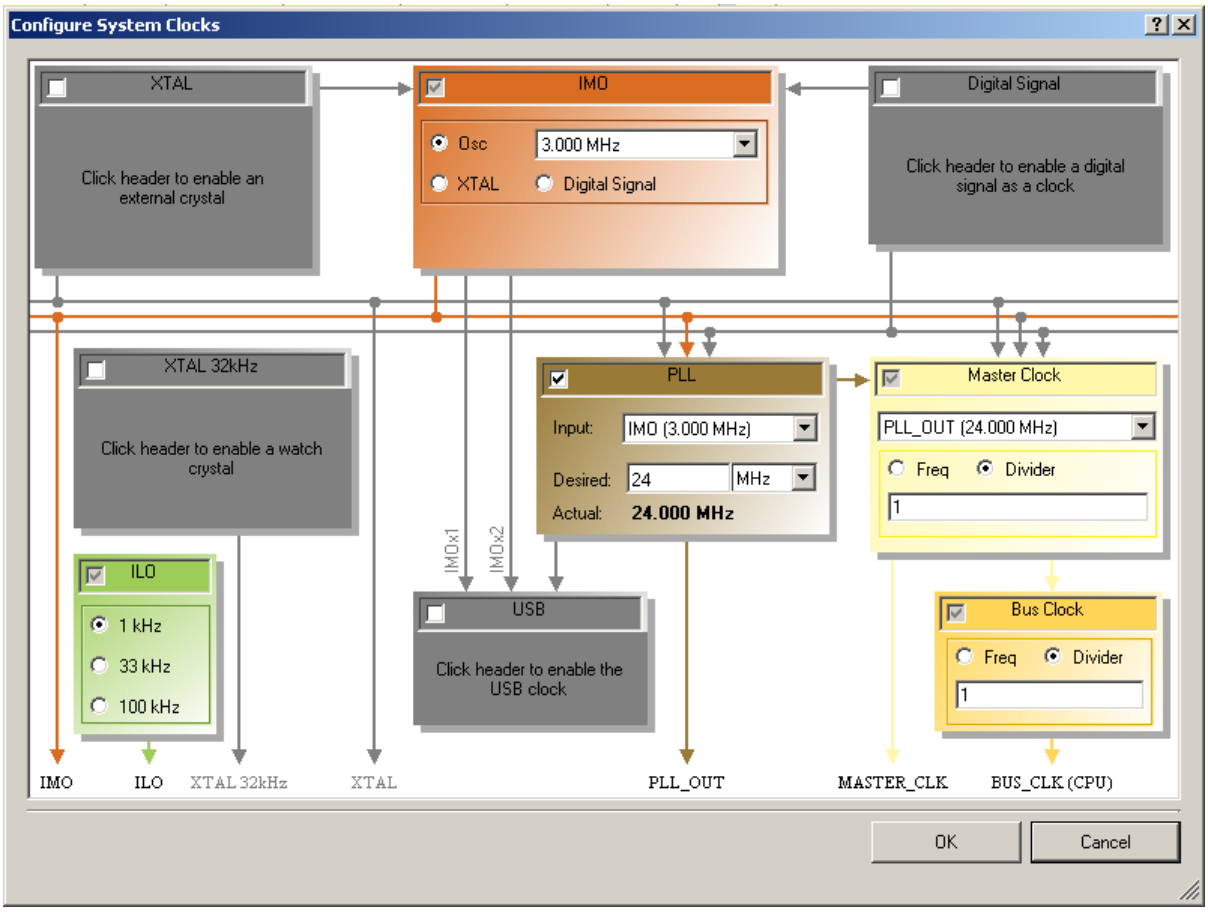
现在您可关闭 Vector GENy 工具。已进行一切所需设置，以处理 PSoC 项目。

创建 PSoC 项目

1. 将 Vector CAN 组件置于原理图中，并打开 Configure 对话框以自定义组件选项，从而使能中断并使用 tx_en 输出。
2. 设置时钟树。
3. 放置引脚。
4. 导入 Vector CAN 驱动程序文件。
5. 更改“Build settings…”（构建设置…）以包括 CAN 驱动程序目录并设置 NOOVERLAY 选项。
6. 在 *main.c* 中编写应用程序。

设置时钟树

图 12. 时钟设置



放置引脚

在图 13 中所示的示例中，为方便起见，引脚与 CAN/LIN EBK （CY8CKIT-017）应配合使用。您可使用任何引脚。

图 13. 引脚选择

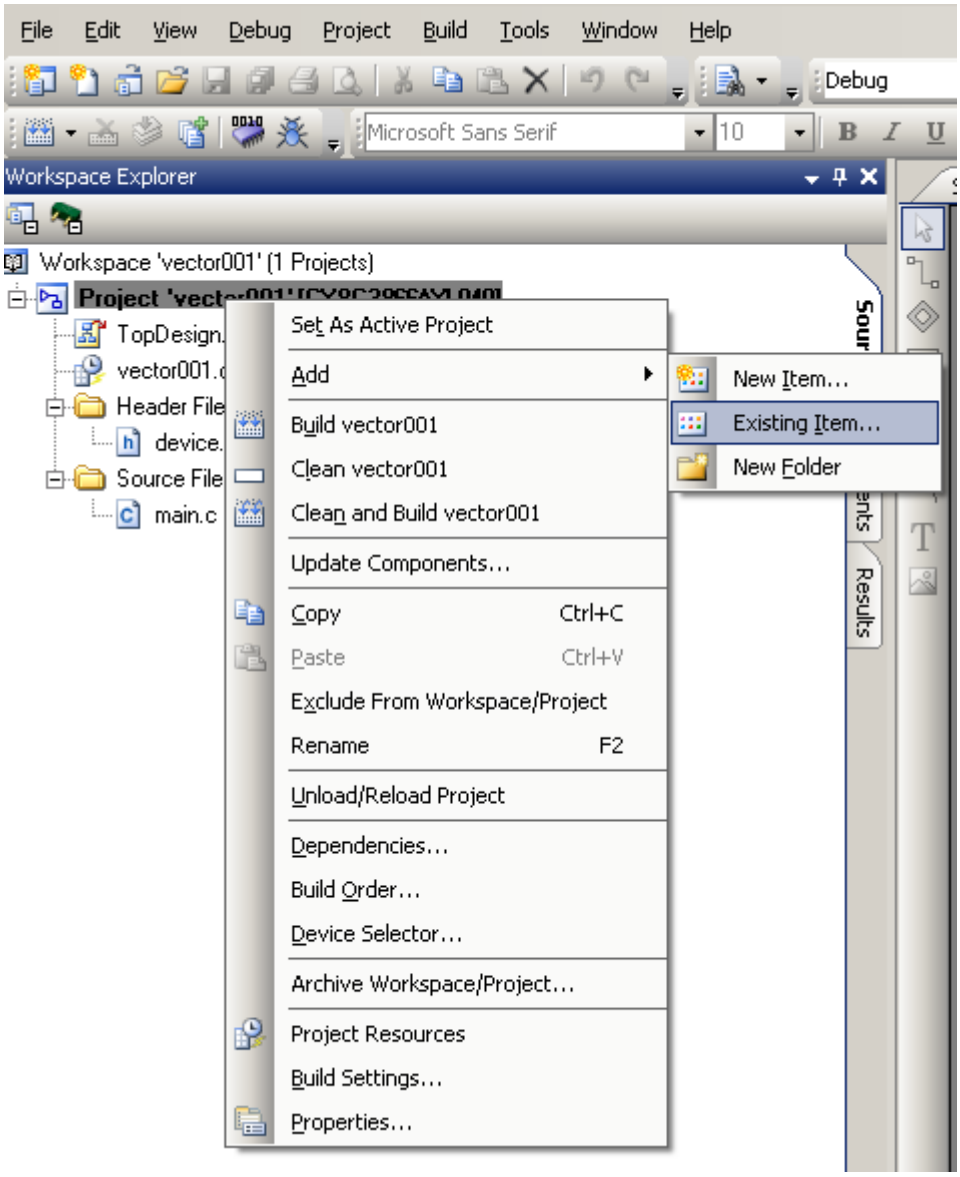
Alias	Name	Pin	Lock
	Pin_tx_en	P3[1]	☑
	Pin_tx	P3[3]	☑
	Pin_rx	P3[2]	☑

导入 Vector CAN 驱动程序文件

必须从“gendata”目录中及“BSW”目录下的所有文件夹中导入头文件和源文件。

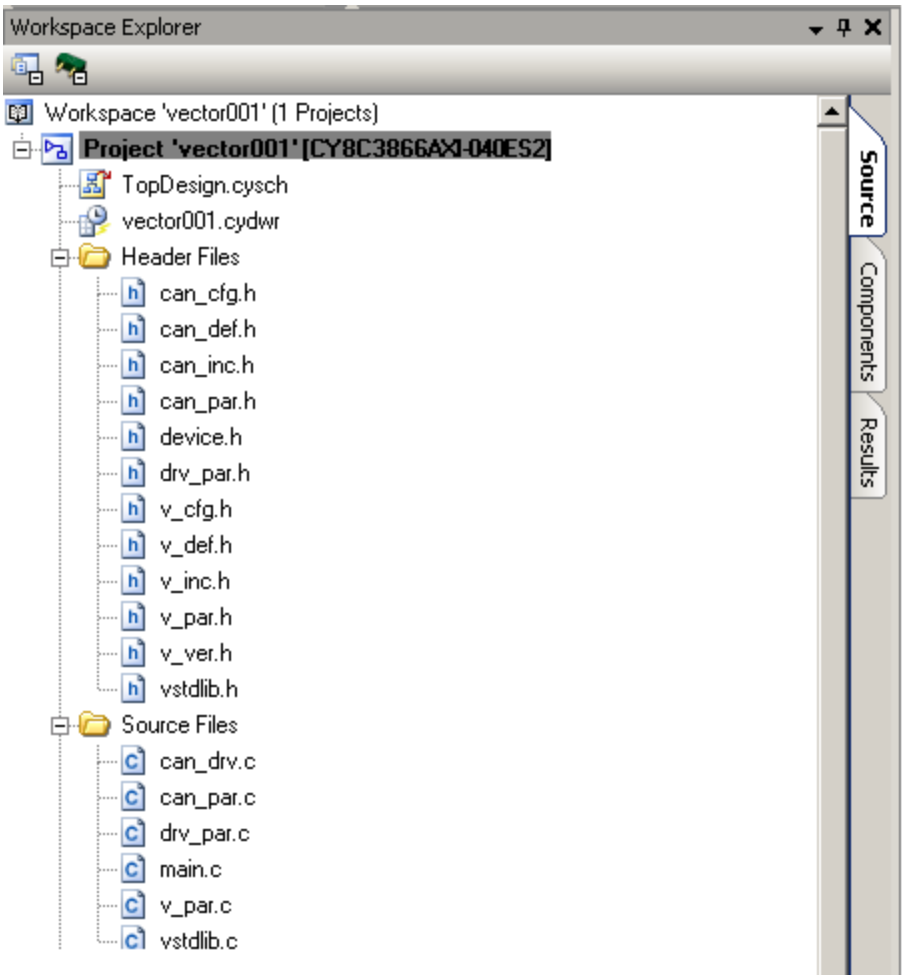
要导入文件，在 PSoC Creator 工作区浏览器中右键单击 **Header Files**（头文件）和 **Source Files**（源文件），并选择菜单以添加现有的项目。重复此流程，直至导入了所有 Vector 文件为止。

图 14. 导入现有头文件和源文件



完成导入步骤后，工作区浏览器应如图 15 所示。

图 15. 导入后的文件列表



更改“Build settings…”以包含 CAN 驱动程序目录

图 16. 添加包含目录

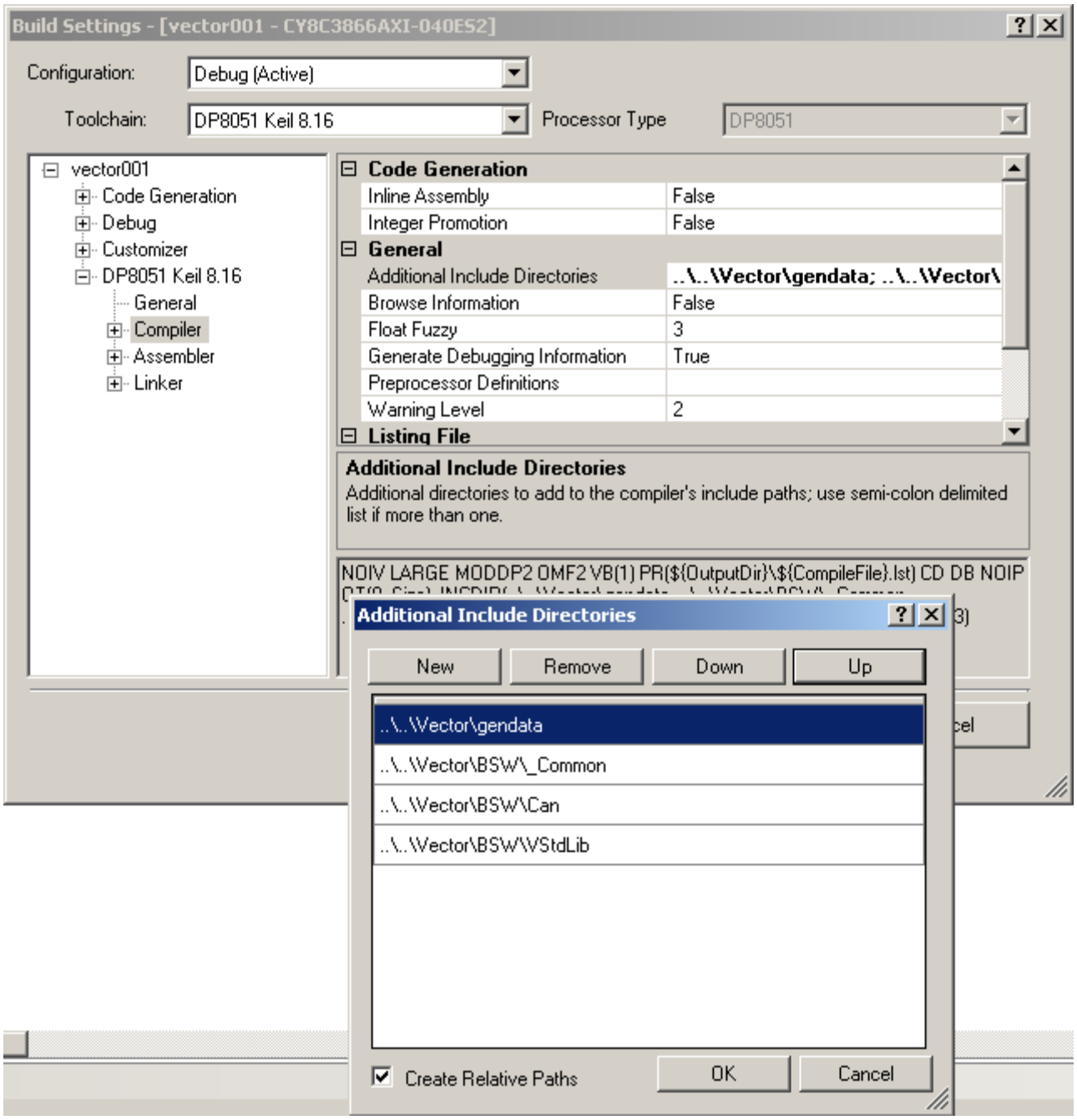
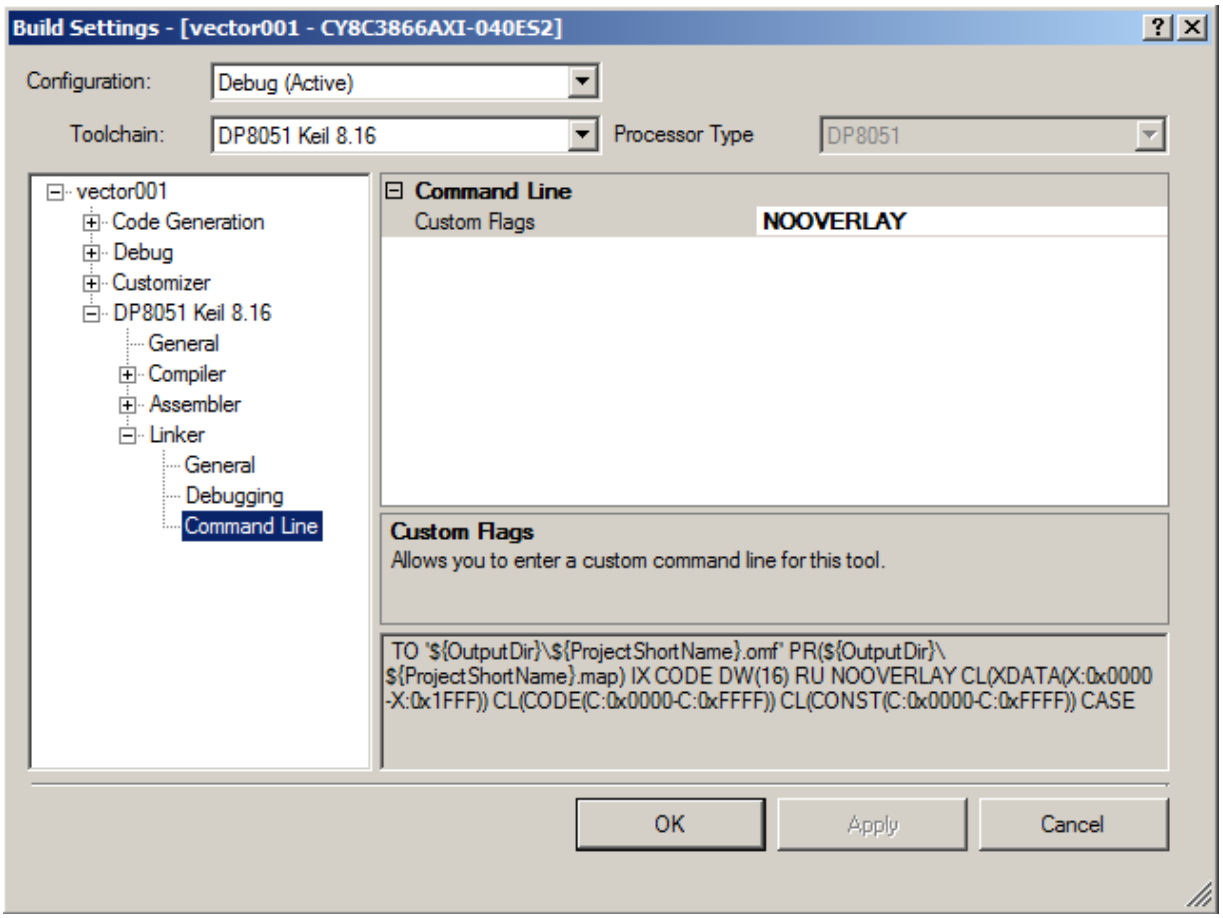


图 17. 设置 NOOVERLAY 选项



Vector CAN 驱动程序 API 使用函数指针。用于 PSoC 3 的 Keil 编译器执行函数调用分析，以确定它覆盖函数变量和参数的方式。当函数指针存在时，编译器无法充分分析调用结构，所以选择了 NOOVERLAY 选项以避免因使用函数指针而产生的问题。更多有关利用 Keil 编译器处理函数指针的信息，请参见应用手册：C51 中的函数指针（www.keil.com/appnotes/docs/apnt_129.asp）。

主要初始化流程要求您执行以下操作：

- main.c 中包含驱动程序的 *v_inc.h* 文件。
- 必要时，使能全局中断。
- 调用 Vector_CAN_Start()函数。
- 调用由 Vector GENy 工具所生成的 CanInitPowerOn()函数。
- 使用由 Vector GENy 工具所生成的 API，在 Vector CAN 中编写必要的功能。

资源

Vector CAN 组件使用芯片中的专用 CAN 硬件模块。

直流和交流电气特性

除非另有说明，否则这些规范的适用条件是：-40 °C ≤ TA ≤ 85 °C 且 TJ ≤ 100 °C。除非另有说明，否则这些规范的适用范围为 1.71 V 到 5.5 V。

CAN 直流规范

参数	说明	条件	最小值	典型值	最大值	单位
IDD	模块电流消耗		_	_	200	µA

CAN 交流规范

参数	说明	条件	最小值	典型值	最大值	单位
	比特率	时钟的最低频率为8 MHz	_	_	1	Mbit

组件勘误表

本节列出了组件的已知问题。

赛普拉斯ID	组件版本	问题	解决方案
191257	v1.10	在没有修正PSoC Creator 3.0 SP1中的版本编号时进行更改这组件。更多有关信息，请参见基础知识文章 KBA94159（网页地址：www.cypress.com/go/kba94159）。	解决方案并非必要的。不会对设计产生影响。

组件更改

本节列出了该组件各版本中的主要更改内容。

版本	更新内容	更改原因/影响
1.10.c	编辑数据手册并将其添加到组件勘误章节。	文档的组件被更改，但设计不受任何影响。
1.10.b	更新了数据表。	移除了已停产的PSoC 5器件的参考内容。
1.10.a	修复了Configure对话框中的总线时钟的可见性问题，使总线时钟频率大于32 MHz。	
1.10	已添加了以下函数的返回值说明：Vector_CAN_Start()、Vector_CAN_Stop()、Vector_CAN_Init()、Vector_CAN_Enable()、Vector_CAN_GlobalIntEnable()以及Vector_CAN_GlobalIntDisable()等函数	
	已添加了MISRA合规性章节。	该组件没有任何特定偏差。
1.0.b	更新了直流和交流电气特性章节。	信息不完整。
	对数据手册进行了少量编辑。	提高可读性。
1.0.a	更新了网上发布的修订版本编号。	无变化。

© 赛普拉斯半导体公司。2012-2015. 此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品内嵌的电路外，赛普拉斯半导体公司不对任何其他电路的使用承担任何责任。也不根据专利或其他权利以明示或暗示的方式授予任何许可。除非与赛普拉斯签订明确的书面协议，否则赛普拉斯不保证产品能够用于或适用于医疗、生命支持、救生、关键控制或安全应用领域。此外，对于可能发生运转异常和故障并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

PSoC®是赛普拉斯半导体公司的注册商标，PSoC Creator™和 Programmable System-on-Chip™是赛普拉斯半导体公司的商标。该处引用的所有其它商标或注册商标归其各自所有者所有。

所有源代码（软件和/或固件）均归赛普拉斯半导体公司（赛普拉斯）所有，并受全球专利法规（美国和美国以外的专利法规）、美国版权法以及国际条约规定的保护和约束。赛普拉斯据此向获许可者授予适用于个人的、非独占性、不可转让的许可，用以复制、使用、修改、创建赛普拉斯源代码的派生作品、编译赛普拉斯源代码和派生作品，并且其目的只能是创建自定义软件和/或固件，以支持获许可者仅将其获得的产品依照适用协议规定的方式与赛普拉斯集成电路配合使用。除上述指定的用途外，未经赛普拉斯明确的书面许可，不得对此类源代码进行任何复制、修改、转换、编译或演示。

免责声明：赛普拉斯不针对此材料提供任何类型的明示或暗示保证，包括（但不仅限于）针对特定用途的适销性和适用性的暗示保证。赛普拉斯保留在不做出通知的情况下对此处所述材料进行更改的权利。赛普拉斯不对此处所述之任何产品或电路的应用或使用承担任何责任。对于合理预计可能发生运转异常和故障，并对用户造成严重伤害的生命支持系统，赛普拉斯不授权将其产品用作此类系统的关键组件。若将赛普拉斯产品用于生命支持系统中，则表示制造商将承担因此类使用而招致的所有风险，并确保赛普拉斯免于因此而受到任何指控。

产品使用可能受适用于赛普拉斯软件许可协议的限制。