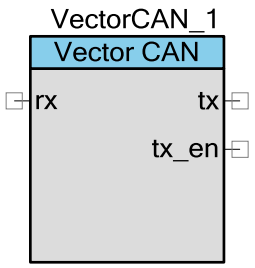
Vector CAN
1.0

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 和使能）
- 由 Vector 提供并支持驱动器

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 驱动器。

何时使用 Vector CAN

在需要针对由 Vector 提供的 PSoC 3 进行 CAN 驱动器集成时，使用 Vector CAN 组件。

输入/输出连接

本节介绍 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 说明中列出的情况下，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

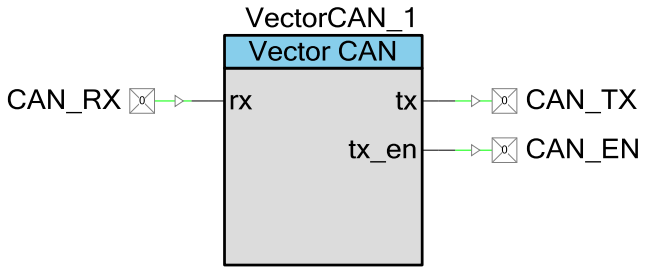
CAN 总线传输信号（连接到外部收发器的 CAN TX 总线）。

tx_en — 输出*

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

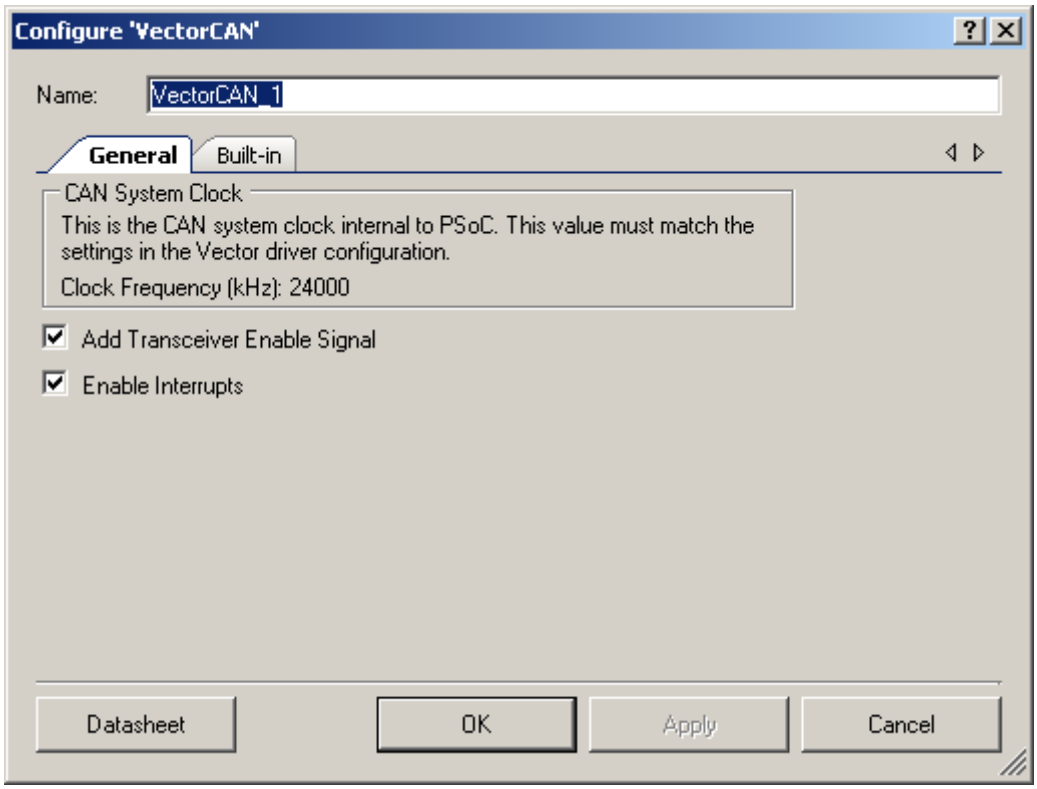
原理图宏信息

组件目录中的默认 Vector CAN 是使用带默认设置的 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。引脚组件也使用默认设置，除了输入引脚组件中的 Input Synchronized（输入同步）设为假。



元件参数

将一个 Vector CAN 组件拖放到您的设计上，并双击以打开 **Configure**（配置）对话框。此对话框有一个 **General**（一般）选项卡，可指导您通过设置 Vector CAN 组件的流程。



General（一般）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**（启用）。

Enable Interrupts（使能中断）

启用/禁用 CAN 中的全局中断。默认设置为 **Enable**（启用）。如果针对轮询模式配置驱动程序文件。

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选定的 BUS_CLK 的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。

放置

Vector CAN 组件放置于固定功能模块中。

资源

模式	数字模块					API Memory（API 存储器）（字节）		Pins（引脚）（每个外部 I/O）
	Datapaths（数据路径）	Macro cells（宏单元）	Status Registers（状态寄存器）	Control Registers（控制寄存器）	Counter7（计数器 7）	Flash（闪存）	RAM	
禁用 tx_en 信号*	不可用	不可用	不可用	不可用	不可用	734	18	2
启用 tx_en 信号*	不可用	不可用	不可用	不可用	不可用	734	18	3

*CAN 组件使用芯片中的专用 CAN 硬件模块。

应用程序编程接口

应用程序编程接口 (API) 子程序允许您使用软件配置组件。下表列出了每个函数的接口，并进行了说明。以下各节将更详细地介绍每个函数。

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、变量和常量符号的前缀。出于可读性考虑，下表中使用的实例名称为“Vector_CAN”。

函数	说明
Vector_CAN_Start()	使用 Vector_CAN_Init() 和 Vector_CAN_Enable() 函数初始化和启用 Vector CAN 组件。
Vector_CAN_Stop()	禁用 Vector CAN 组件。
Vector_CAN_GlobalIntEnable()	从 CAN 内核中启用全局中断。
Vector_CAN_GlobalIntDisable()	从 CAN 内核中禁用全局中断。
Vector_CAN_Sleep()	准备组件以进入睡眠状态。
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。

函数	说明
Vector_CAN_Enable()	启用 Vector CAN 组件。
Vector_CAN_SaveConfig()	保存组件配置。
Vector_CAN_RestoreConfig()	恢复组件配置。

全局变量

变量	说明
Vector_CAN_initVar	Vector_CAN_initVar 指示 Vector CAN 是否已初始化。变量将初始化为 0，并在第一次调用 Vector_CAN_Start() 时设置为 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() 函数之前调用 Vector_CAN_Init() 函数。

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，调用 Vector_CAN_Init() 函数，然后调用 Vector_CAN_Enable() 函数。

参数： None（无）

Return Value（返回值）： 指示寄存器是否已写入和验证。

Side Effects（副作用）： None（无）

uint8 Vector_CAN_Stop(void)

说明： 禁用 Vector CAN 组件。

参数： None（无）

Return Value（返回值）： 指示寄存器是否已写入和验证。

Side Effects（副作用）： None（无）

uint8 Vector_CAN_GlobalIntEnable(void)

说明： 此函数从 CAN 内核中启用全局中断。

参数： None（无）

Return Value（返回值）： 指示寄存器是否已写入和验证。

Side Effects（副作用）： None（无）

uint8 Vector_CAN_GlobalIntDisable(void)

说明： 此函数从 CAN 内核中禁用全局中断。

参数： None（无）

Return Value（返回值）： 指示寄存器是否已写入和验证。

Side Effects（副作用）： None（无）

void Vector_CAN_Sleep(void)

说明： 这是准备组件睡眠的首选子程序。Vector_CAN_Sleep() 子程序保存当前组件的状态。然后它调用 Vector_CAN_SaveConfig() 函数，并调用 Vector_CAN_Stop() 以保存硬件配置。

在调用 CyPmSleep() 或 CyPmHibernate() 函数之前调用 Vector_CAN_Sleep() 函数。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

void Vector_CAN_Wakeup(void)

说明： 该函数是将组件恢复到调用 Vector_CAN_Sleep() 时状态的首选子程序。Vector_CAN_Wakeup() 函数调用 Vector_CAN_RestoreConfig() 函数，以恢复配置。如果组件在调用 Vector_CAN_Sleep() 函数前已启用，则 Vector_CAN_Wakeup() 函数也将重新启用组件。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： 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“配置”对话框设置来初始化或恢复组件。无需调用 Vector_CAN_Init()，因为 Vector_CAN_Start() 子程序会调用该函数并是开始组件操作的首选方法。此函数利用由 Vector CAN 配置工具生成的中断服务子程序 CanIsr_0() 设置 CAN 中断。

参数： None（无）

Return Value（返回值）： None（无）

Side Effects（副作用）： None（无）

uint8 Vector_CAN_Enable(void)

说明： 激活硬件并开始执行组件操作。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： None（无）

Return Value（返回值）： 指示寄存器是否已写入和验证。

Side Effects（副作用）： None（无）

void Vector_CAN_SaveConfig(void)

说明：	此函数会保存组件配置和非保留寄存器。它还保存 Configure（配置）对话框中定义的或通过相应 API 修改的当前组件参数值。此函数由 Vector_CAN_Sleep() 函数调用。
参数：	None（无）
Return Value（返回值）：	None（无）
Side Effects（副作用）：	None（无）

void Vector_CAN_RestoreConfig(void)

说明：	此函数会恢复组件配置和非保留寄存器。它还将组件参数值恢复为在调用 Vector_CAN_Sleep() 函数之前的值。
参数：	None（无）
Return Value（返回值）：	None（无）
Side Effects（副作用）：	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。

固件源代码示例

PSoC Creator 在“查找示例项目”对话框中提供了大量包括原理图和代码示例的示例项目。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开 Start Page（开始页）或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（滤波器选项）可缩小可选项目的列表。

有关更多信息，请参见 PSoC Creator 帮助中的“Find Example Project（查找示例项目）”主题。

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() 函数利用由 Vector CAN 配置工具生成的中断服务子程序 CanIsr_0() 设置 CAN 中断。

功能描述

有关完整的 CAN 内核说明，请参见 *PSoC 3 和 PSoC 5 技术参考手册*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。

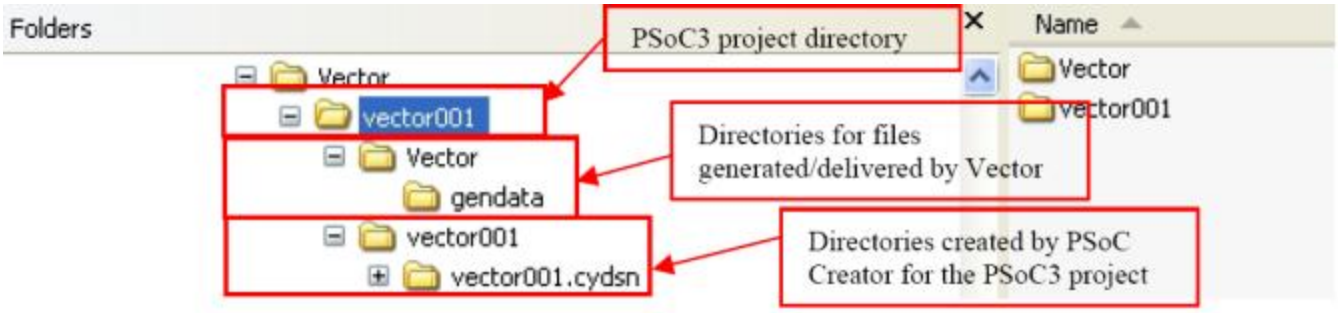
用 Vector GENy 工具创建项目。

1. 针对 CAN 驱动程序文件创建空白 PSoC 项目和目录。

在使用 Vector GENy 工具生成 CAN 驱动程序文件之前，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 1 中所示。

图 1. PSoC 项目目录



2. 为 PSoC 应用生成 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：

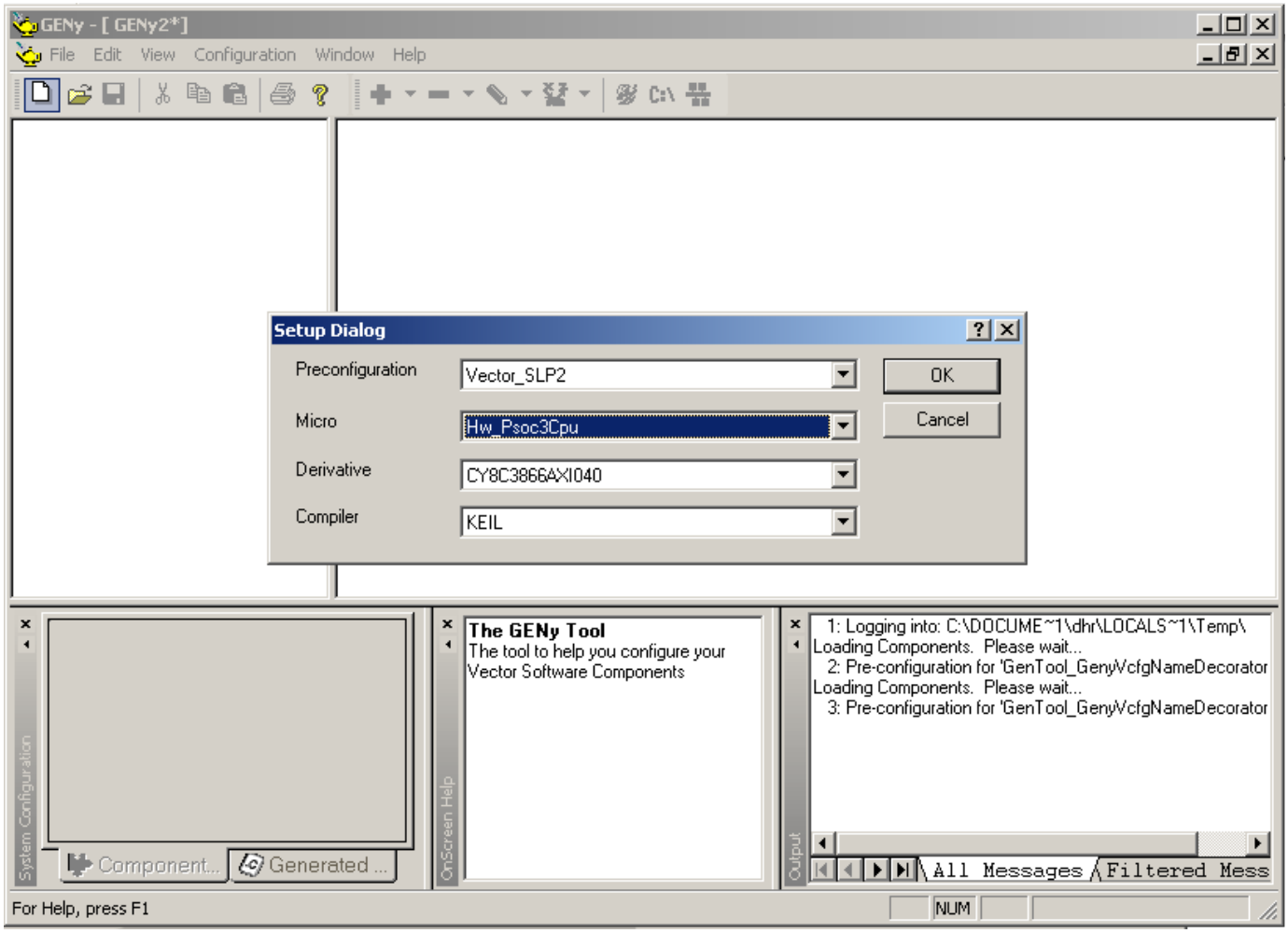
- CAN 应用消息数据库
- Vector GENy 中的配置

加载数据库之后，您可配置 CAN 驱动程序生成工具 (GENy) 以生成驱动程序，以便处理 Vector CAN 消息。

3. 打开 Vector GENy 工具并创建新配置

1. 选择 Start（开始）> All Programs（所有程序）> Vector GENy 1.4 > GENy，启动 Vector GENy。
2. 单击 **New**（新建）按钮（请参见图 2）。

图 2. 单击“New”（新建）按钮后的 Vector GENy 工具



1. 单击 **New**（新建）按钮时显示的 **Setup Dialog**（设置对话框）除了显示的选项之外没有其他选项，所以单击 **OK**（确定）。

4. 生成 CAN 驱动程序文件之前设置配置

加载数据库之后，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 消息。
3. 选择总线时序和消息接收过滤器。
4. 选择轮询模式或中断模式的选项。
5. 选择放置生成的文件的目标目录。

6. 选择 Tx 和 Rx 消息之后，选择和配置接受过滤器和总线时序，如下图中所示。下图显示这些步骤。

图 3. 接受过滤器和总线时序

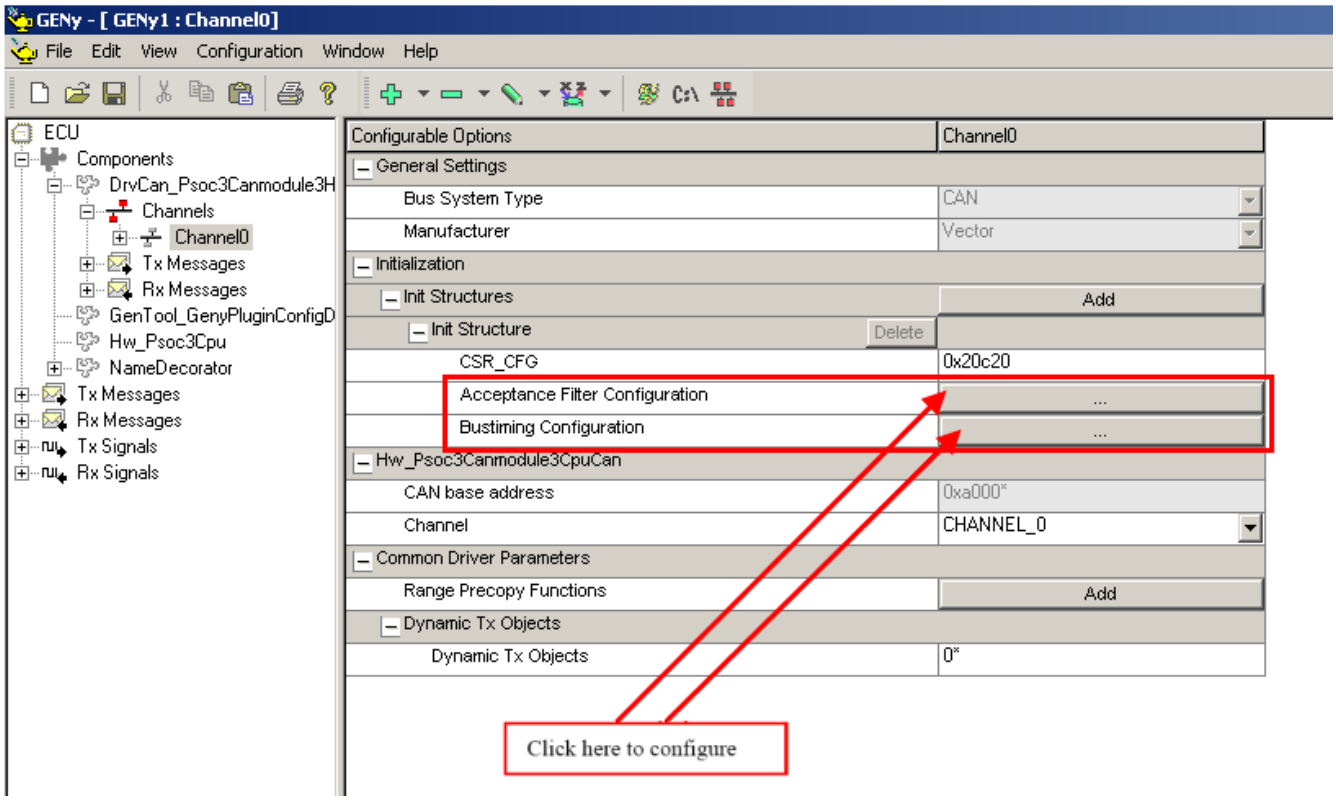
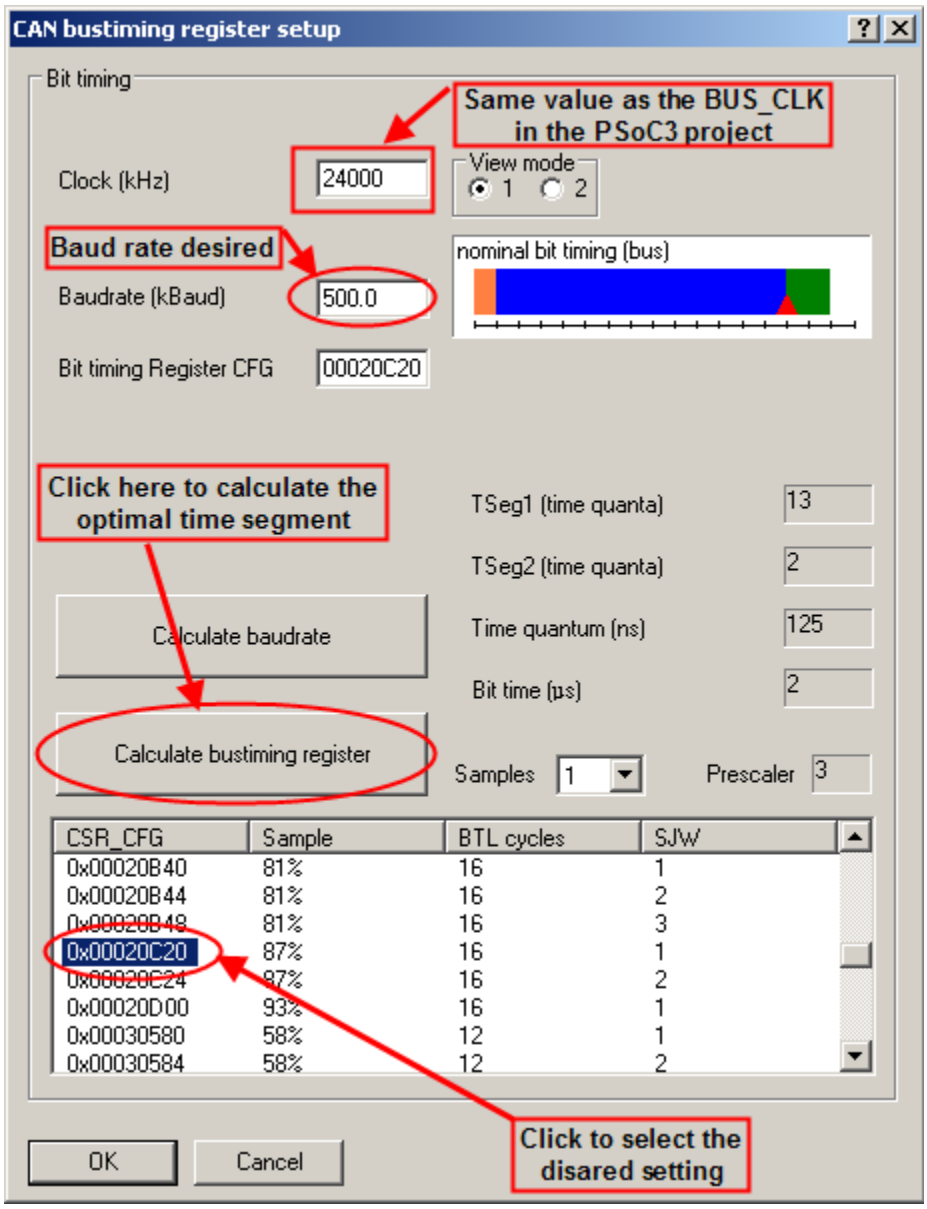


图 4. 总线时序设置



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

选择轮询模式或中断模式时，注意在轮询模式中，应用程序必须调用函数 CanTask() 以处理待定事件或消息。在中断模式中，事件或消息由中断服务子程序 CanIsr_0() 处理。

设置完接受过滤器和总线时序以及邮箱轮询/中断模式之后，选择 CAN 驱动程序文件目录。

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

图 5. 选择轮询模式或中断模式

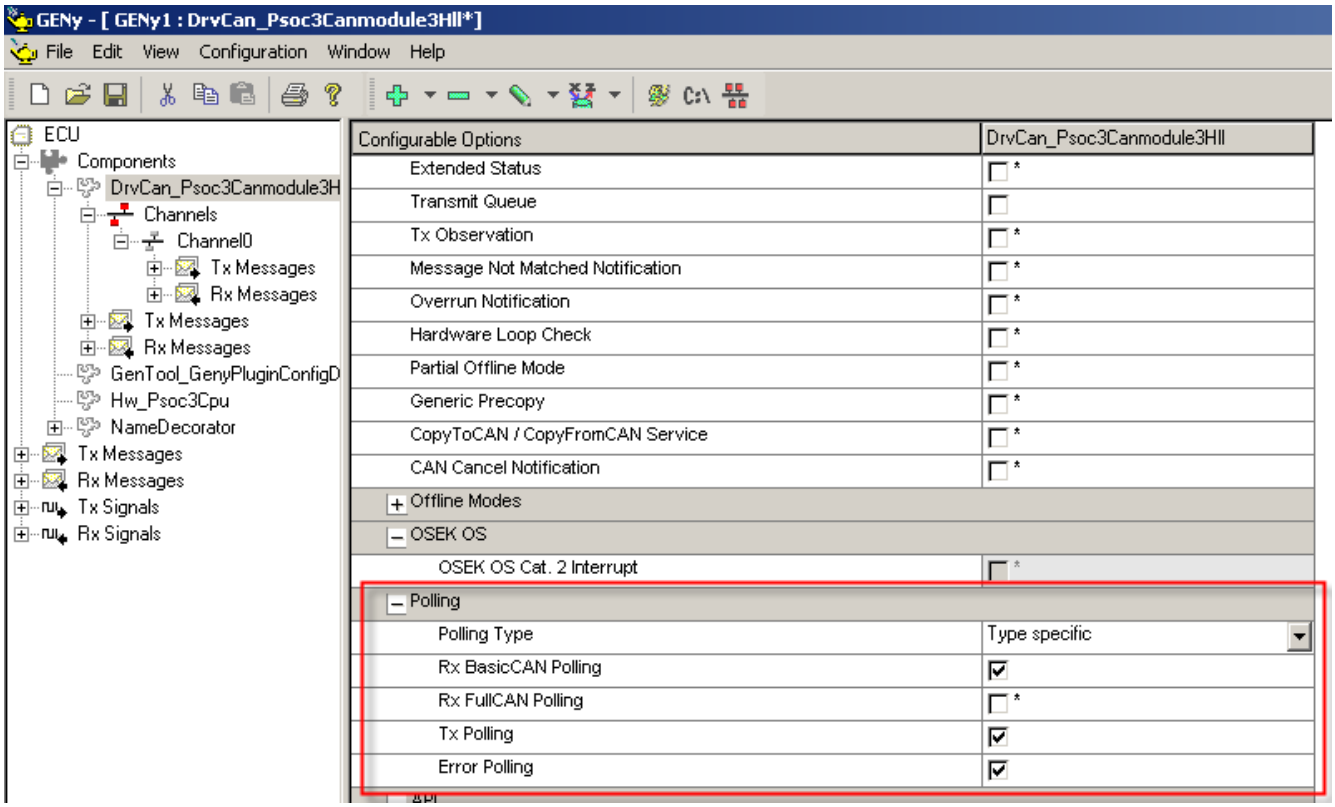


图 6. 选择目标目录

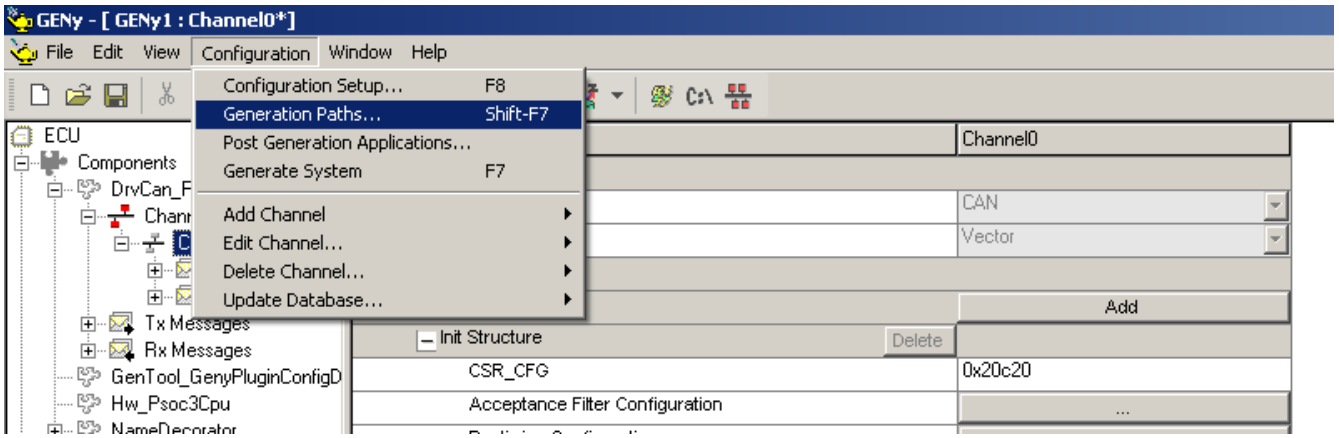
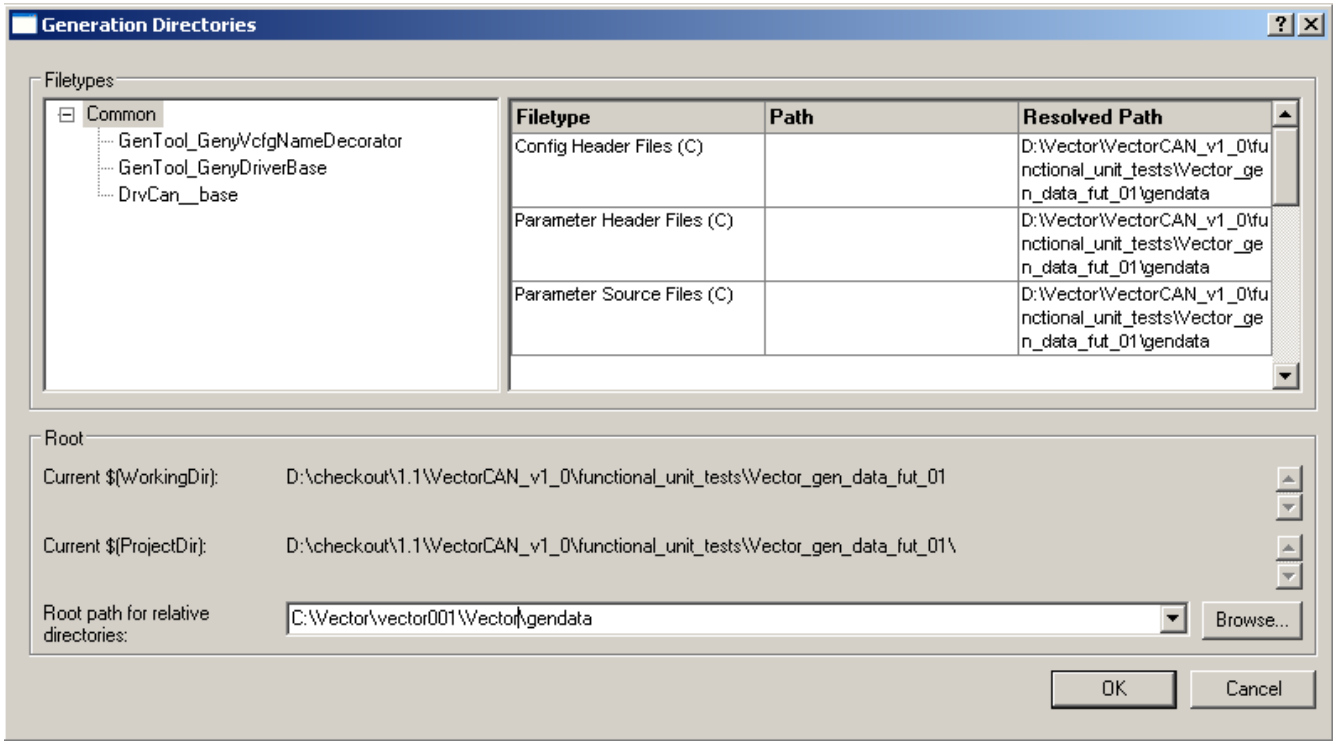


图 7. 选择 Root Path for the Relative Directories（相对目录的根路径）



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。

5. 生成 CAN 驱动程序文件

此时，工具已准备好生成 PSoC 驱动程序文件，文件将被放置在 PSoC 项目中创建的选定目录中。

您也应在 PSoC 项目目录中保存配置文件，以便在需要修改和生成了新文件时，所有项目都位于同一项目目录中。

每次按下 **Generate System**（生成系统）按钮就会生成这些文件，但为方便起见，也有一些通用文件（未更改）应复制到同一 PSoC 项目目录下。

下图说明了移至 PSoC Creator 中以构建项目之前的最终步骤。

图 8. 单击“Generate System”（生成系统）以在 PSoC 项目子目录中生成文件

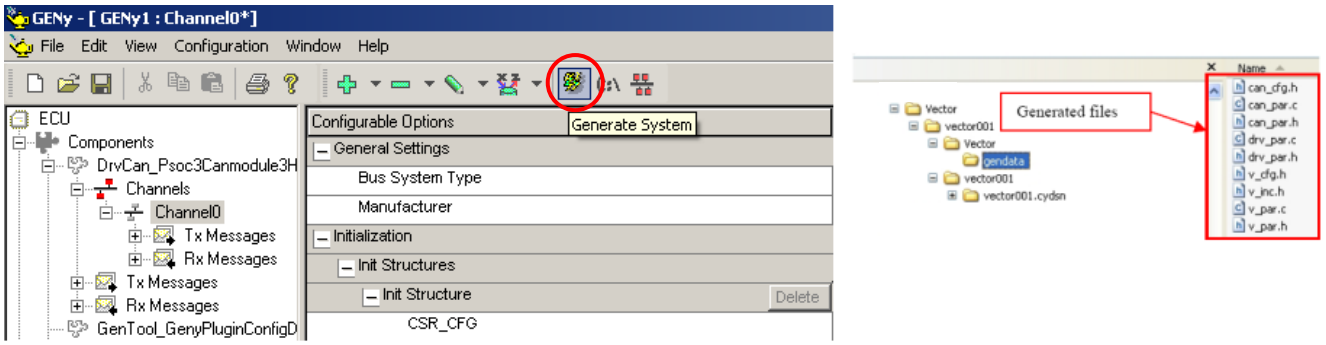
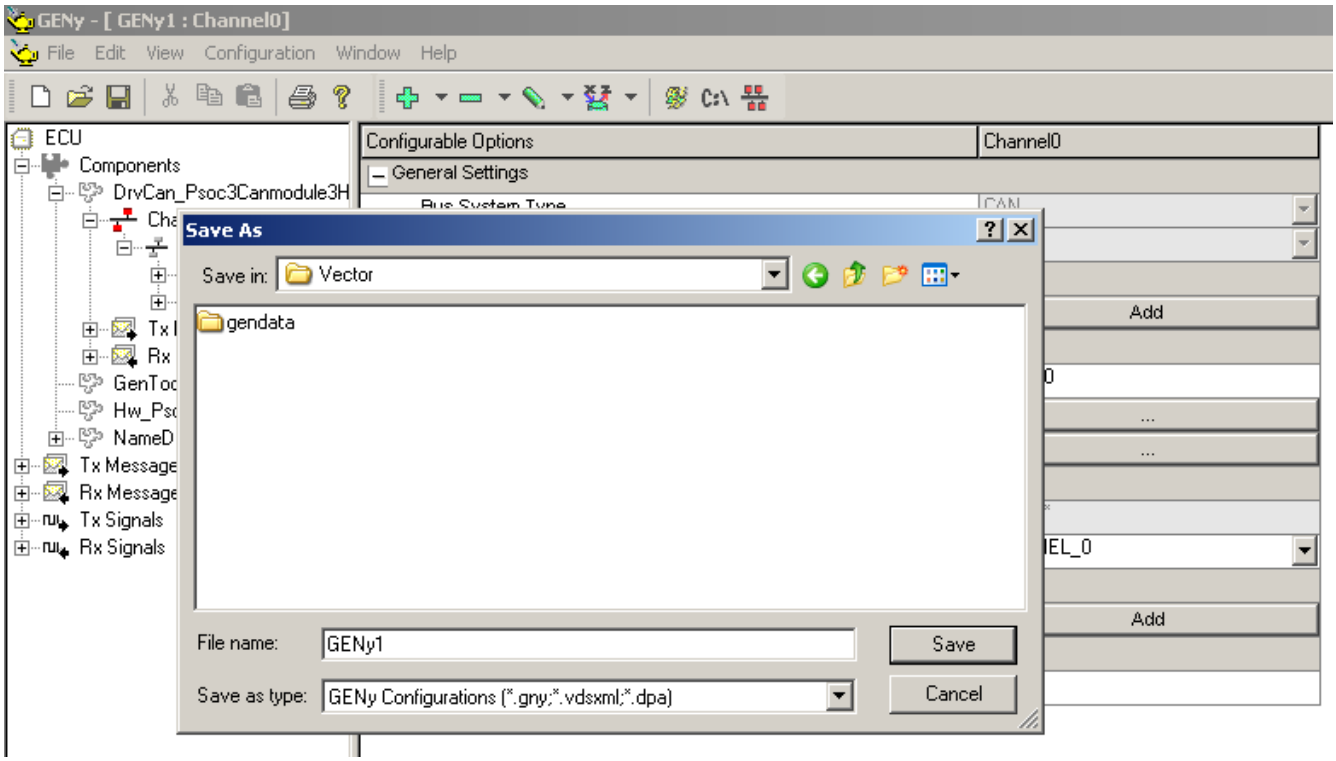


图 9. 在 PSoC 项目子目录中保存 Vector GENy 配置

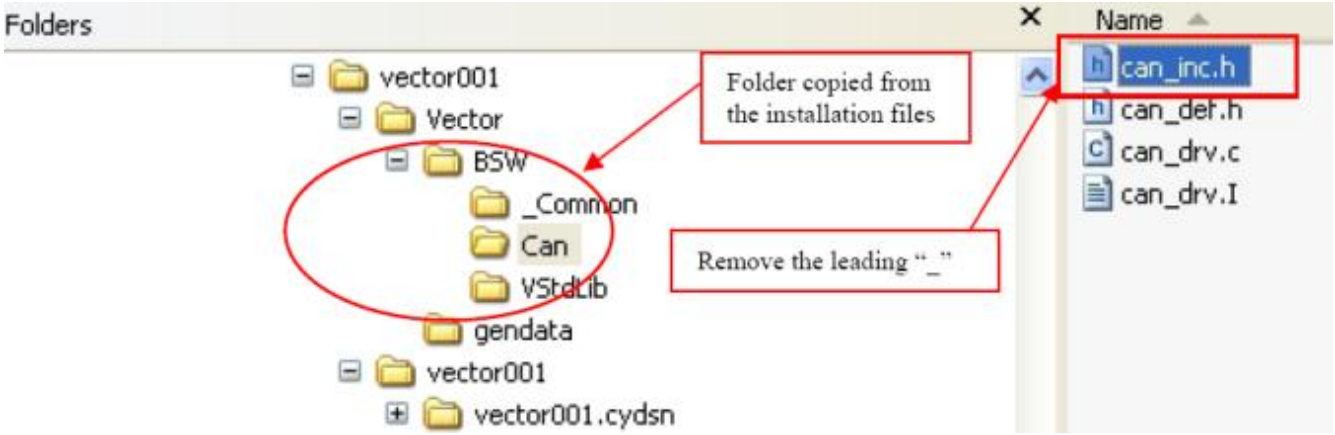


在安装过程中，如果接受了默认目录，通用文件位于：

`C:\Vector\CBD0810092_D00_Psoc3\BSW`

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

图 10. CAN 驱动程序通用目录复制到 PSoC 项目文件夹中



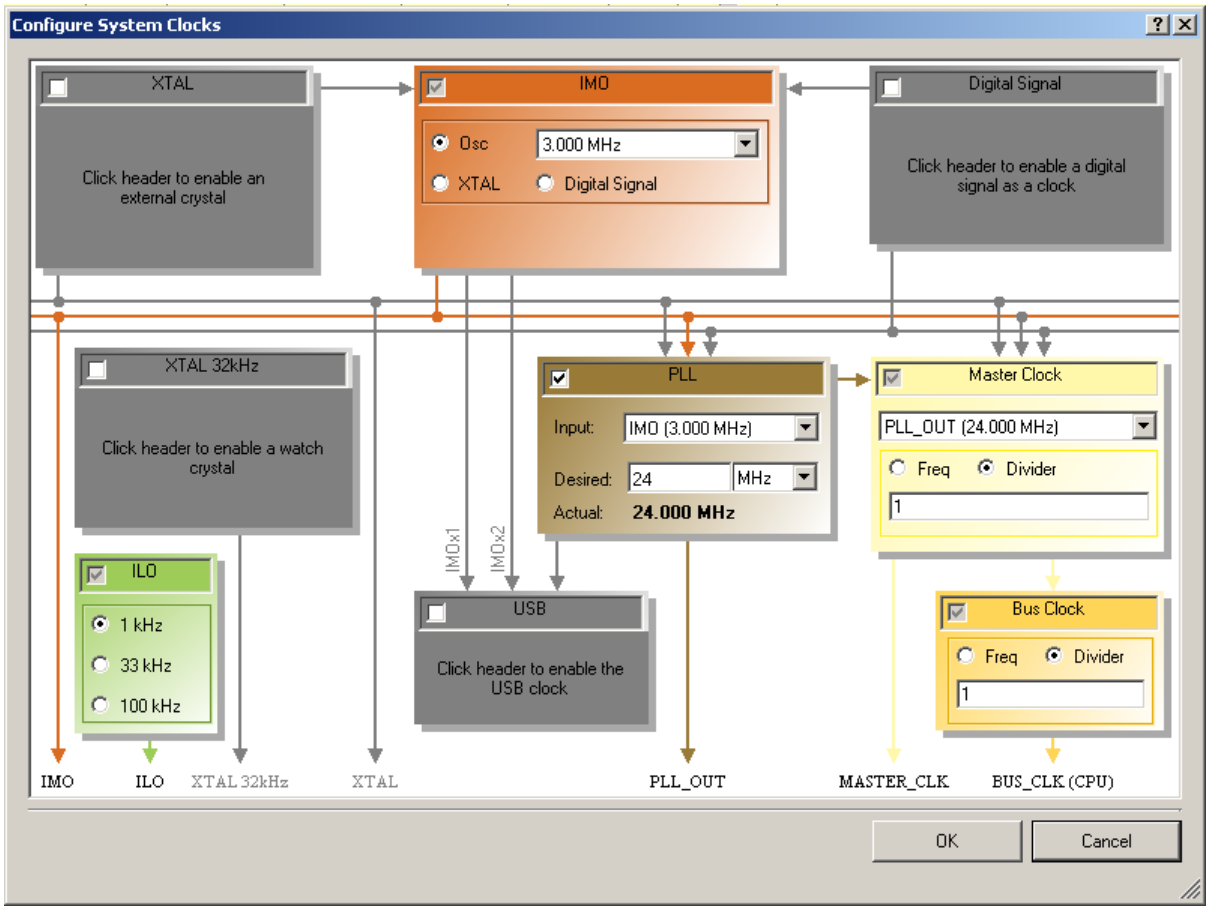
现在您可关闭 Vector GENy 工具。已进行一切所需设置，以处理 PSoC 项目。

6. 创建 PSoC 项目

1. 将 Vector CAN 组件置于原理图中，并打开配置对话框以自定义组件选项，从而启用中断并使用 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* 中写入应用程序。

设置时钟树

图 11. 时钟设置



放置引脚

在图 12 中所示示例中，为方便起见，引脚与 CAN/LIN EBK (CY8CKIT-017) 配合使用。您可使用任何引脚。

图 12. 引脚选择

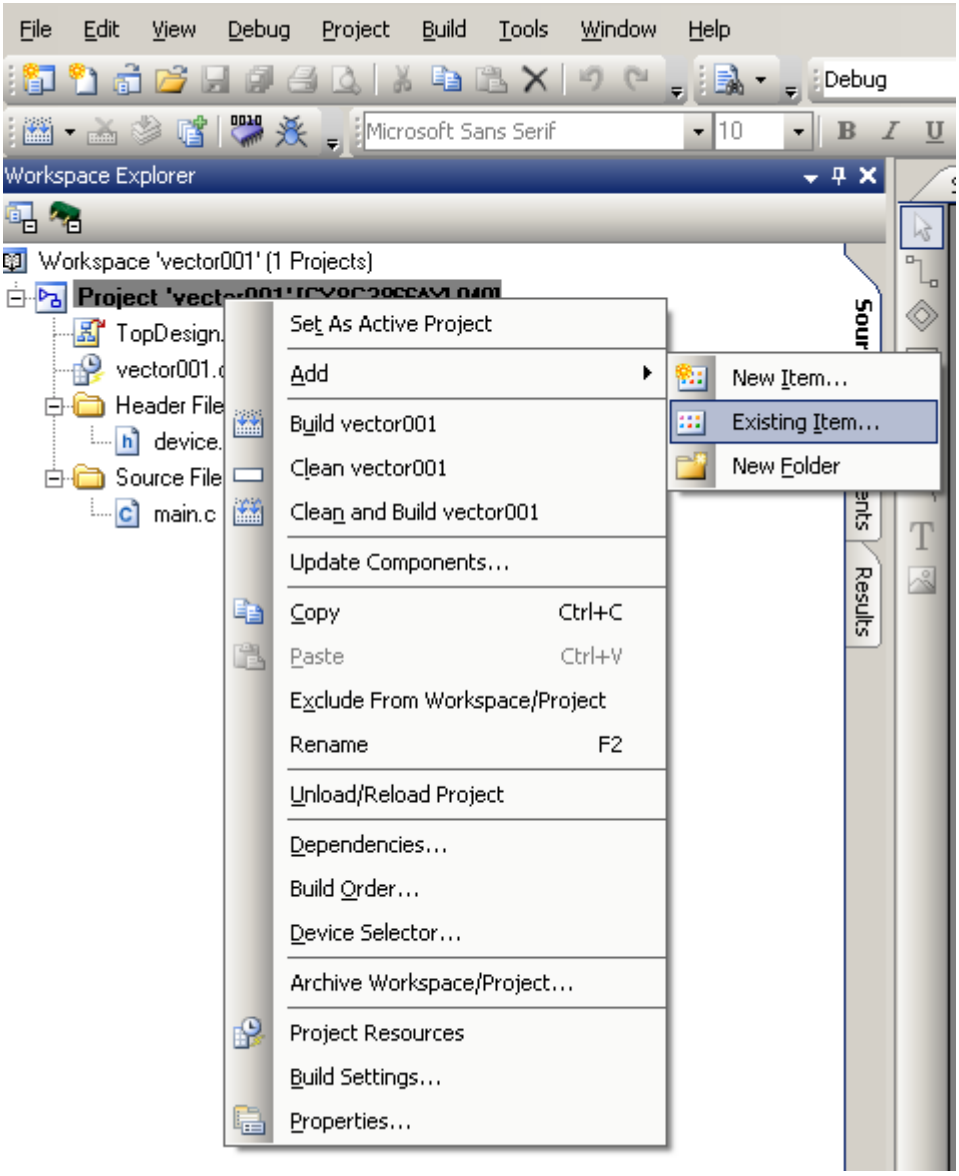
Alias	Name	Pin	Lock
	Pin_tx_en	P3[1]	☑
	Pin_tx	P3[3]	☑
	Pin_rx	P3[2]	☑

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。

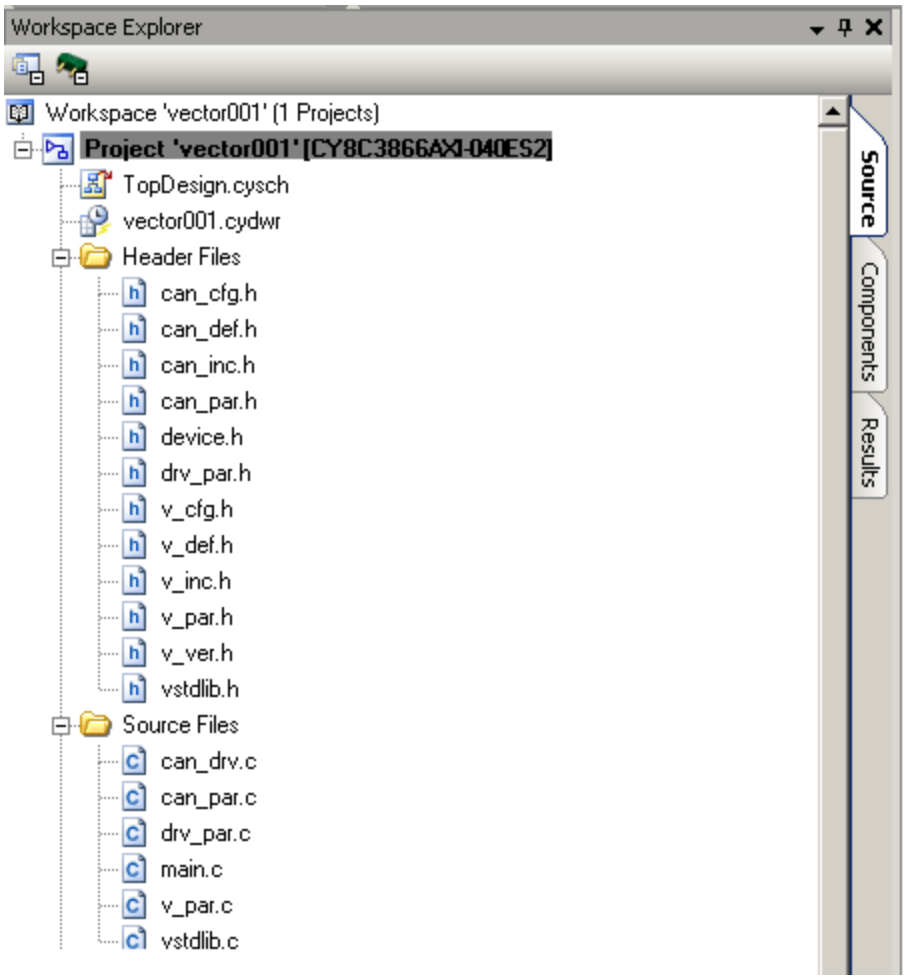
要导入文件，在 PSoC Creator 工作区浏览器中右键单击 **Header Files**（头文件）和 **Source Files**（源文件），并选择菜单以添加现有项目。重复此流程，直至导入了所有 Vector 文件。

图 13. 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 14 中所示。

图 14. 导入后的文件列表



更改“**Build settings…**”（构建设置...）以包括 **CAN** 驱动程序目录

图 15. 添加包括目录

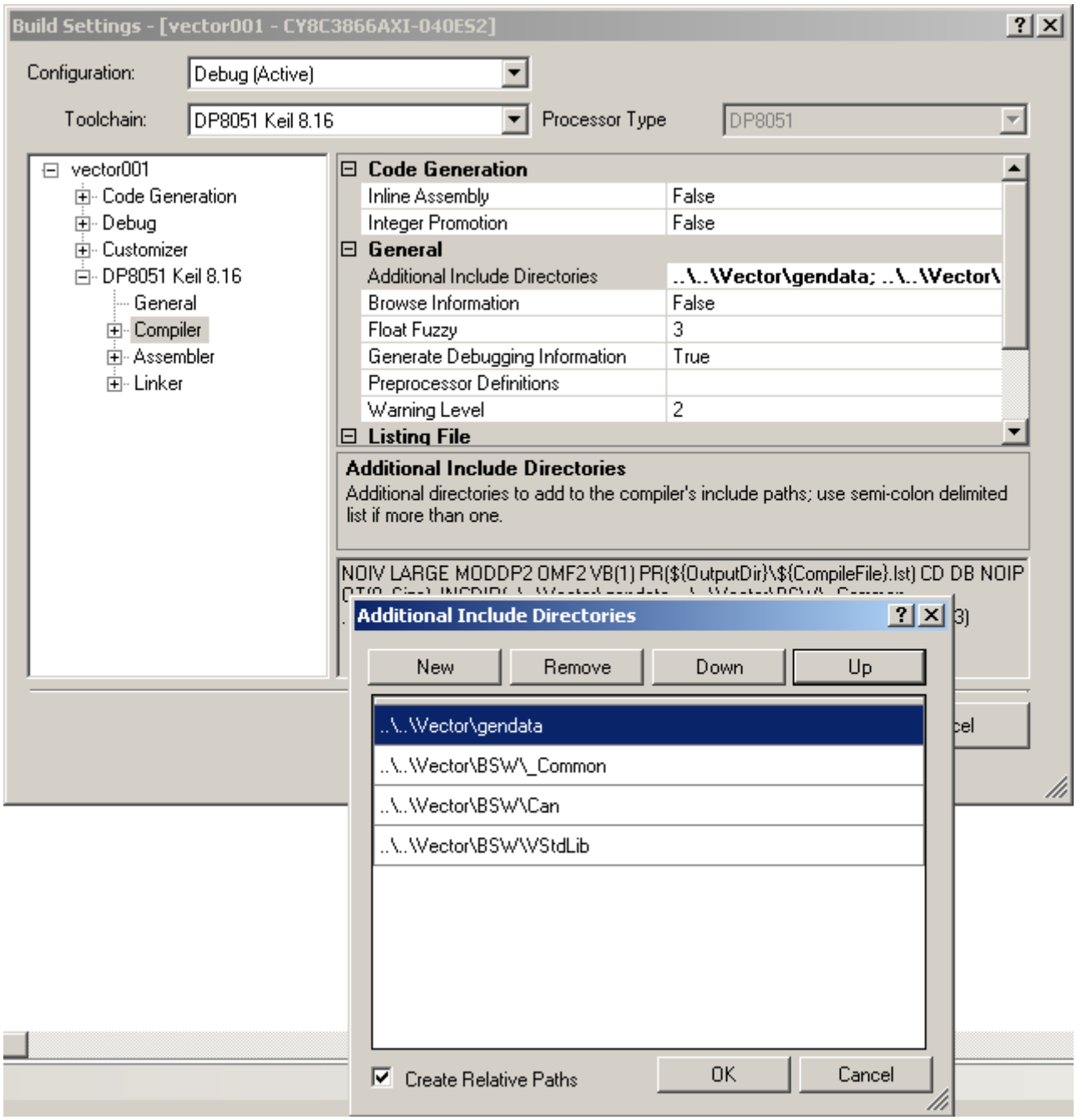
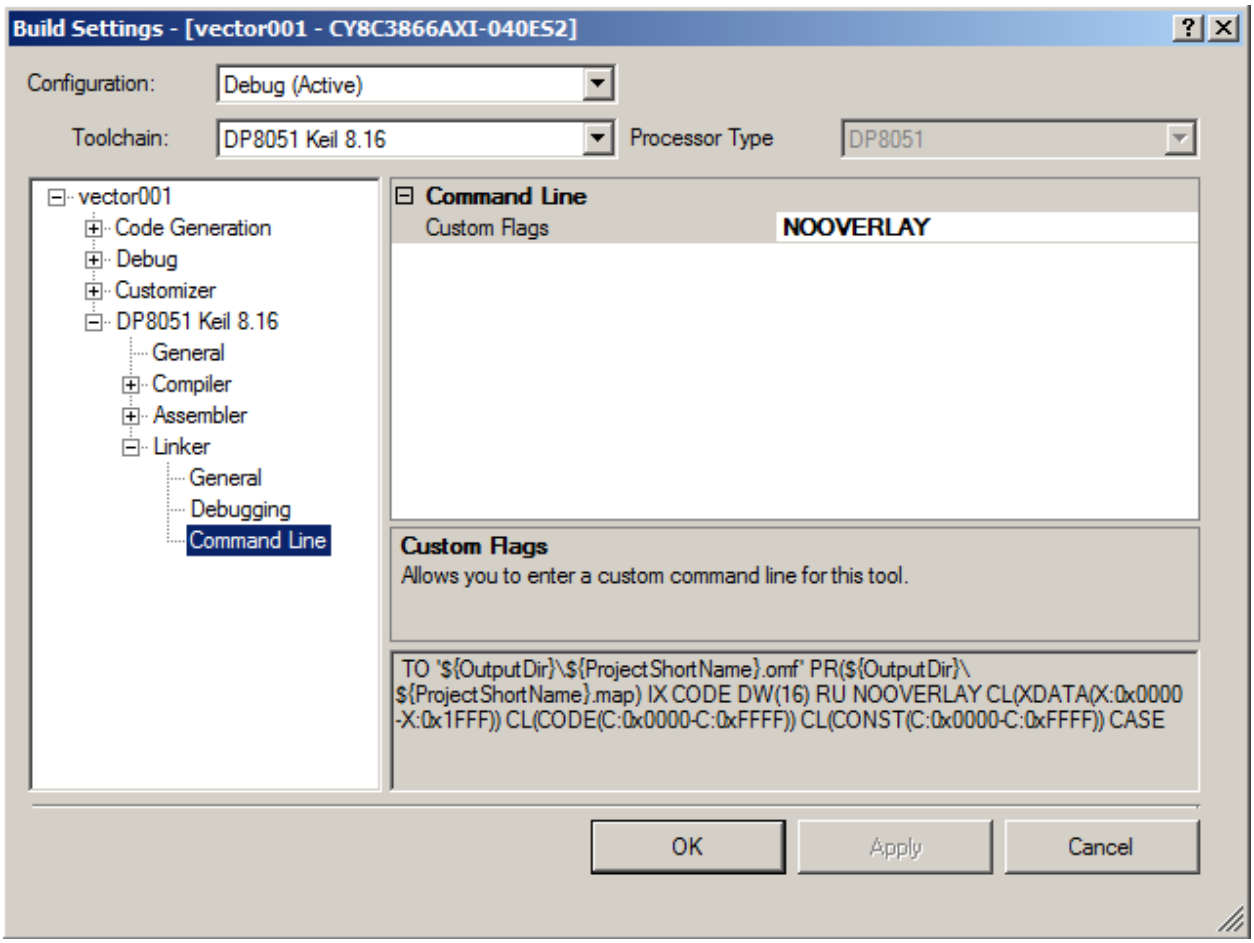


图 16. 设置 NOOVERLAY 选项



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问题。有关利用 Keil 编译器处理函数指针的更多信息，请参见应用笔记：*C51 中的函数指针* (www.keil.com/appnotes/docs/apnt_129.asp)。

初始化流程主要要求您执行以下操作：

- 在 main.c 中包括驱动程序的 *v_inc.h* 文件。
- 必要时，启用全局中断。
- 调用 Vector_CAN_Start() 函数。
- 调用 CanInitPowerOn() 函数（由 Vector GENy 工具生成）。
- 从 Vector CAN 中使用 API 写入由 Vector GENy 工具生成的必要功能。

直流和交流电气特性

下面的值表示了预计性能，它们基于初始特性数据。

CAN 直流规范

参数	说明	条件	最小值	典型值	最大值	单位
	模块电流消耗	500 kbps	--	--	285	µA
		1 Mbps	--	--	330	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	最小 8-MHz 时钟	--	--	1	Mbit

组件更改

版本 1.0 为 Vector CAN 组件的首次发行版。

© 赛普拉斯半导体公司，2012。此处所包含的信息可能会随时更改，恕不另行通知。除赛普拉斯产品的内嵌电路之外，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，否则赛普拉斯产品不保证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。

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。此处引用的所有其他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。除上述指定的用途之外，未经赛普拉斯的明确书面许可，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的赛普拉斯软件许可协议限制。