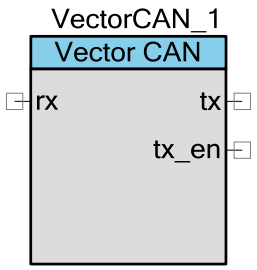
Vector CAN
1.0

特長



- CAN2.0 A/B プロトコル、ISO 11898-1 準拠
- 最大 1 Mbps @ 8 MHz (BUS_CLK) のプログラム可能ビット レート
- 外部トランシーバに対し 2 線または 3 線のインタフェースをサポート(Tx、Rx、および Tx イネーブル)
- Vector 社によって供給・サポートされるドライバ

概要説明

Vector CANbedded 環境は、車載アプリケーションで基本的な通信と診断要件をカバーするいくつかのソースコード コンポーネントで構成されます。

Vector CANbedded ソフトウェア スイートは、顧客固有であり、その動作はアプリケーションと OEM に応じて異なります。Vector CANbedded スイートのこのコンポーネントは、特定の OEM アプリケーションのタイプに関係なく、一般的に CANbedded 構造をサポートするために記述されています。

PSoC3 用に開発された Vector CAN コンポーネントは、Vector 認定の CAN ドライバの統合が簡単に行えます。

Vector CANが使用される背景

Vector CAN コンポーネントは、Vector によって供給された PSoC 3 用 CAN ドライバを統合する必要がある場合に使用されます。

入出力接続

このセクションでは、Vector CAN コンポーネントの様々な入力および出力の接続について説明します。I/O リストのアスタリスク (*) は、I/O が、その I/O の説明でリストされている条件において、シンボルに隠れている可能性があることを示します。

rx – Input(入力)

CAN バスは信号を受信します (外部トランシーバの CAN RX バスに接続されます)。

tx – Output(出力)

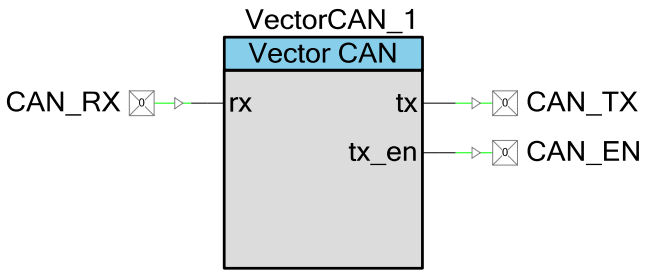
CAN バスは信号を送信します (外部トランシーバの CAN TX バスに接続されます)。

tx_en – Output(出力)*

外部トランシーバのイネーブル信号。この出力は、**Add Transceiver Enable Signal** オプションが **Configure** ダイアログで選択された時に表示されます。

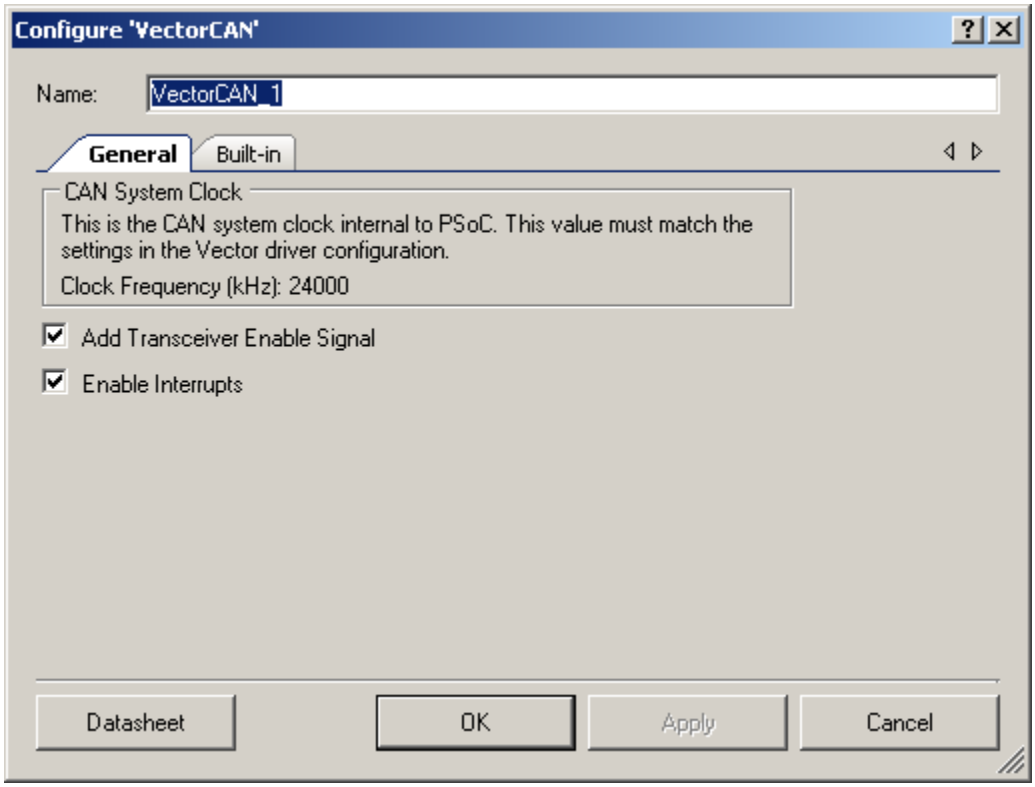
回路図マクロ情報

コンポーネントカタログにあるデフォルトの Vector CAN は、デフォルト設定の Vector CAN コンポーネントを使用した回路図マクロです。Vector CAN コンポーネントは入力および出力ピン コンポーネントに接続されます。また、このピン コンポーネントは、同期される入力が入力ピン コンポーネントで false に設定されている場合を除き、デフォルト設定で構成されます。



コンポーネント・パラメータ

Vector CAN コンポーネントを設計上にドラッグし、ダブルクリックして **Configure** ダイアログを開きます。このダイアログには **General** タブがあり、Vector CAN コンポーネントを設定するプロセスが説明されています。



General タブには、次のパラメータがあります。

Add Transceiver Enable Signal

外部 CAN トランシーバの tx_en 信号の使用を有効/無効にします。デフォルト設定は **Enable (有効)** です。

Enable Interrupts

CAN からのグローバル割り込みを有効/無効にします。デフォルト設定は **Enable (有効)** です。ドライバ ファイルがポーリング モードに設定されている場合は、無効にする必要があります (これを行わないと、コンパイル エラーが表示されます)。

Clock Selection

Vector CAN コンポーネントは BUS_CLK クロック信号に接続されます。最大 1 Mbps の標準 CAN ボーレートをすべてサポートするには、最低 8 MHz が必要です。PSoC 3 プロジェクトの Design-wide resources で選択される BUS_CLK の値は、Vector CAN ドライバのバス タイミング設定で選択された値と同じでなければなりません。

Placement

Vector CAN コンポーネントは、CAN 固定機能ブロックに配置されます。

Resources

Mode	Digital Blocks					API Memory (Bytes)		Pins (per External I/O)
	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	
With tx_en signal disabled *	N/A	N/A	N/A	N/A	N/A	734	18	2
With tx_en signal enabled *	N/A	N/A	N/A	N/A	N/A	734	18	3

*CAN コンポーネントは、専用の CAN ハードウェアブロックを使用します。

アプリケーション プログラミング インタフェース

アプリケーション・プログラミング・インターフェース (API) ルーチンにより、ソフトウェアを使用してコンポーネントを設定できます。次の表は、各関数へのインターフェースとその説明を示しています。続くセクションでは、各関数について詳しく説明します。

デフォルトでは、PSoC Creator は “Vector_CAN_1” という名前のインスタンスを、与えられた設計のコンポーネントの最初のインスタンスに割り当てます。インスタンス名は、識別子の構文ルールに従った固有の値に変更できます。インスタンス名は、すべてのグローバル関数名、変数名、定数名のプリフィックスになります。読み易くするため、次の表で使用されているインスタンス名は “Vector_CAN” としています。

Function名	説明
Vector_CAN_Start()	Vector_CAN_Init() および Vector_CAN_Enable() 関数を使用して、Vector CAN コンポーネントを初期化し、有効にします。
Vector_CAN_Stop()	Vector CAN コンポーネントを無効にします。
Vector_CAN_GlobalIntEnable()	CAN コアからのグローバル割り込みを有効にします。
Vector_CAN_GlobalIntDisable()	CAN コアからのグローバル割り込みを無効にします。
Vector_CAN_Sleep()	コンポーネントのスリープの準備をします。
Vector_CAN_Wakeup()	コンポーネントを Vector_CAN_Sleep() が呼び出された時の状態に戻します。

Function名	説明
Vector_CAN_Init()	コンポーネント カスタマイザの設定に基づいて、Vector CAN コンポーネントを初期化します。Vector CAN 構成ツールで生成された割り込みサービス ルーチン CanIsr_0() を使用して CAN 割り込みをセットアップします。
Vector_CAN_Enable()	Vector CAN コンポーネントを有効にします。
Vector_CAN_SaveConfig()	コンポーネントの設定を保存します。
Vector_CAN_RestoreConfig()	コンポーネントの設定を復元します。

グローバル変数

変数	説明
Vector_CAN_initVar	Vector_CAN_initVar は Vector CAN が初期化されているかどうかを示します。変数は、0 に初期化され、Vector_CAN_Start() 初めてが呼び出されると 1 に設定されます。これにより、コンポーネントは Vector_CAN_Start() ルーチンへの最初の呼び出し後、再初期化せずに再起動できます。 このコンポーネントの再初期化が必要な場合は、Vector_CAN_Start() または Vector_CAN_Enable() 関数の前に、Vector_CAN_Init() 関数を呼び出すことができます。

uint8 Vector_CAN_Start(void)

説明: これは、コンポーネントの動作を開始する際に推奨される方法です。Vector_CAN_Start() は initVar 変数を設定し、Vector_CAN_Init() 関数を呼び出し、続いて Vector_CAN_Enable() 関数を呼び出します。

パラメータ: なし

戻り値: レジスタが書き込まれ確認されたかどうかの表示

サイドエフェクト: なし

uint8 Vector_CAN_Stop(void)

説明: Vector CAN コンポーネントを無効にします。

パラメータ: なし

戻り値: レジスタが書き込まれ確認されたかどうかの表示

サイドエフェクト: なし

uint8 Vector_CAN_GlobalIntEnable(void)

説明: この関数は、CAN コアからのグローバル割り込みを有効にします。

パラメータ: なし

戻り値: レジスタが書き込まれ確認されたかどうかの表示

サイドエフェクト: なし

uint8 Vector_CAN_GlobalIntDisable(void)

説明: この関数は、CAN コアからのグローバル割り込みを無効にします。

パラメータ: なし

戻り値: レジスタが書き込まれ確認されたかどうかの表示

サイドエフェクト: なし

void Vector_CAN_Sleep(void)

説明: これは、コンポーネントのスリープを準備するのに推奨されるルーチンです。Vector_CAN_Sleep() ルーチンは、現在のコンポーネントの状態を保存します。続いて、Vector_CAN_SaveConfig() 関数を呼び出し、Vector_CAN_Stop() を呼び出して、ハードウェア構成を保存します。

CyPmSleep() または CyPmHibernate() 関数を呼び出す前に、Vector_CAN_Sleep() 関数を呼び出してください。

パラメータ: なし

戻り値: なし

サイドエフェクト: なし

void Vector_CAN_Wakeup(void)

説明: これは、コンポーネントを Vector_CAN_Sleep() が呼び出されたときの状態に復元する場合に推奨されるルーチンです。Vector_CAN_Wakeup() 関数は、Vector_CAN_RestoreConfig() 関数を呼び出して設定を復元します。Vector_CAN_Sleep() 関数が呼び出される前にコンポーネントが有効になった場合は、Vector_CAN_Wakeup() 関数もコンポーネントを再度有効にします。

パラメータ: なし

戻り値: なし

サイドエフェクト: 最初に Vector_CAN_Sleep() または Vector_CAN_SaveConfig() 関数を呼び出すことなく Vector_CAN_Wakeup() 関数を呼び出すと、予期しない動作を起こす場合があります。

void Vector_CAN_Init (void)

説明: カスタマイザの [Configure] (設定) ダイアログの設定に従って、コンポーネントを初期化または復元します。Vector_CAN_Start() ルーチンが この関数を呼び出すので、Vector_CAN_Init() を呼び出す必要はありません。これはコンポーネントの動作を開始する際に推奨される方法です。この関数は、Vector CAN 構成ツールで生成された割り込みサービス ルーチン CanIsr_0() を使用して CAN 割り込みをセットアップします。

パラメータ: なし

戻り値: なし

サイドエフェクト: なし

uint8 Vector_CAN_Enable(void)

説明: ハードウェアの使用を開始し、コンポーネントの動作を開始します。Vector_CAN_Start() ルーチンが この関数を呼び出すので、Vector_CAN_Enable() を呼び出す必要はありません。これはコンポーネントの動作を開始する際に推奨される方法です。

パラメータ: なし

戻り値: レジスタが書き込まれ確認されたかどうかの表示

サイドエフェクト: なし

void Vector_CAN_SaveConfig(void)

説明: この関数は、コンポーネントの設定と保持されないレジスタを保存します。この関数は、[Configure] ダイアログで定義されている、または該当する API で変更される、現在のコンポーネント・パラメータ値も保存します。この関数は、Vector_CAN_Sleep() 関数に呼び出されます。

パラメータ: なし

戻り値: なし

サイドエフェクト: なし

void Vector_CAN_RestoreConfig(void)

説明: この関数は、コンポーネントの設定と非保持レジスタを復元します。この関数はまた、コンポーネントのパラメータを Vector_CAN_Sleep() 関数を呼び出す前の状態に復元します。

パラメータ: なし

戻り値: なし

サイドエフェクト: Vector_CAN_Sleep() または Vector_CAN_SaveConfig() 関数を呼び出さずに、この関数を呼び出すと、予期しない動作を起こす場合があります。

ソースコードのサンプル

PSoC Creator は、Find Example Project ダイアログに数多くのサンプルプロジェクトを提供しており、そこには回路図およびコード例が含まれています。コンポーネント固有の例を見るには、[Component Catalog] または回路図に置いたコンポーネント インスタンスからダイアログを開きます。一般例については、[Start Page] または **[File]** メニューからダイアログを開きます。必要に応じてダイアログにある **Filter Options** を使用し、選択できるプロジェクトのリストを絞り込みます。

詳しくは、PSoC Creator ヘルプの「Find Example Project」を参照してください。

割り込みサービスルーチン

Vector ドライバは CAN 割り込みを使用するので、それにアクセスできます。Vector_CAN_Init() 関数は、Vector CAN 構成ツールで生成された割り込みサービス ルーチン CanIsr_0() を使用して CAN 割り込みをセットアップします。

機能の説明

CAN コアの詳細な説明については、*PSoC 3 および PSoC 5 テクニカル リファレンス マニュアル* の Controller Area Network (CAN: コントローラ エリア ネットワーク) の章を参照してください。

Vector GENy ツールの詳細な説明については、Vector GENy ツールのマニュアルを参照してください。

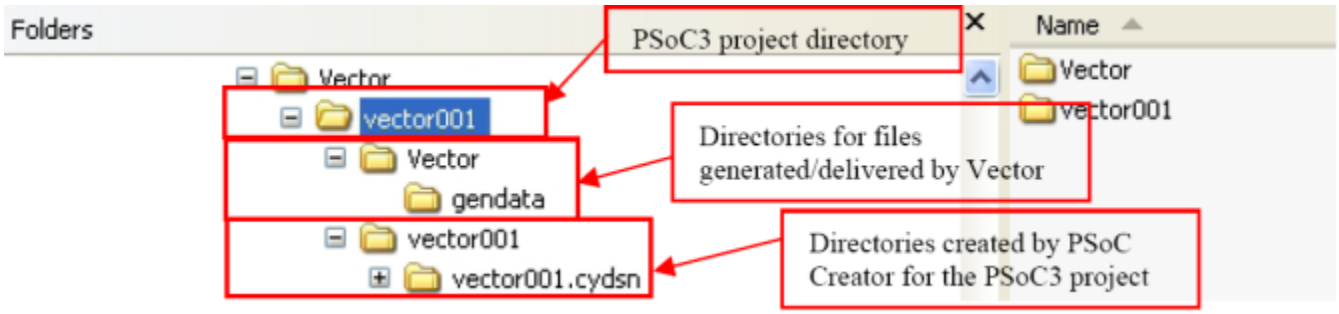
Vector GENy ツールによるプロジェクトの作成

1. CAN ドライバ ファイルの空白 PSoC プロジェクトとディレクトリの作成

CAN ドライバ ファイルの生成に Vector GENy ツールを使い始める前に、Vector GENy ツールがドライバ ファイルを直接 PSoC プロジェクト ディレクトリに配置できるように、PSoC プロジェクトを作成する必要があります。PSoC プロジェクトは、差し当たり空の状態になります。

図 2 に示すように、PSoC プロジェクト ディレクトリに、CAN ドライバ生成ファイルのディレクトリを作成します。

図 2. PSoC プロジェクト ディレクトリ



2. PSoC アプリケーションの Vector CAN ドライバ ファイルを生成する

Vector GENy ツールは、以下に基づき PSoC の CAN ドライバ ファイルを生成します:

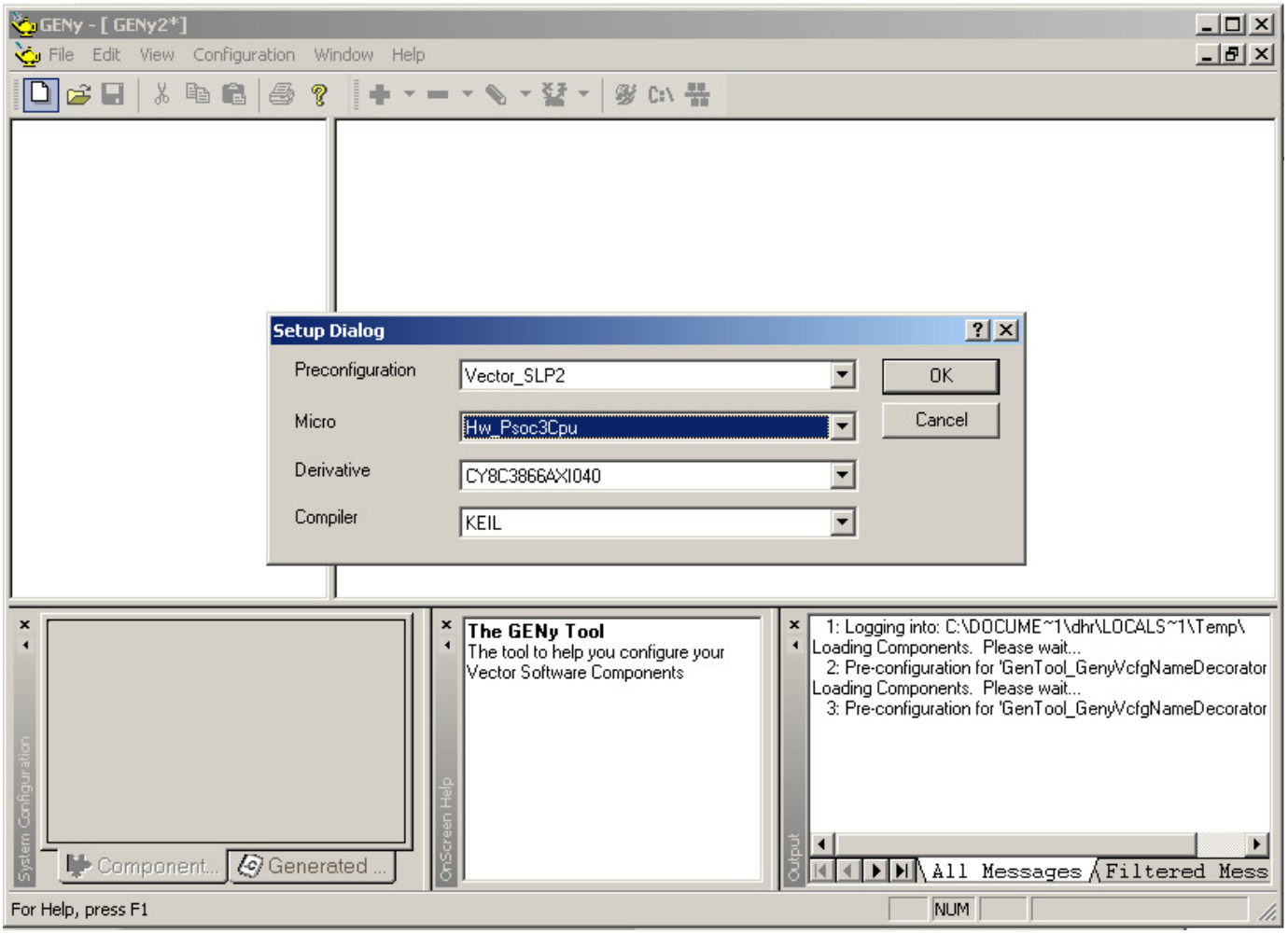
- CAN アプリケーション メッセージ データベース
- Vector GENy での設定

データベースをロードすると、you can configure the CAN ドライバ生成ツール (GENy) を設定して、Vector CAN メッセージを処理するドライバを生成できます。

3. Vector GENy ツールを開き、新規構成を作成する

1. スタート > すべてのプログラム > Vector GENy 1.4 > GENy の順に選択して Vector GENy ツールを起動します。
2. **New** ボタンをクリックします (図 3 を参照)。

図 3.「New」ボタンをクリックした後の Vector GENy ツール



Setup ダイアログが New ボタンのクリックにより表示されます。これには他にオプションがないので、[OK] をクリックします。

4. 構成をセットアップしてから CAN ドライバ ファイルを生成する

データベースをロードしたら、以下により PSoC ドライバを構成します (これは「開始」のための設定です):

1. PSoC CAN モジュール コンポーネントを選択します。
2. ドライバが使用する Tx および Rx メッセージを選択します。
3. バス タイミングとメッセージ受け入れフィルタを選択します。
4. ポーリングまたは割り込みモードのオプションを選択します。
5. 生成されたファイルを入れるディレクトリを選択します。

6. Tx および Rx メッセージを選択したら、下図に示すように、受け入れフィルタとバス タイミングを選択して設定します。

以下の図に、これらの手順を示します。

図 4. 受け入れフィルタとバス タイミング

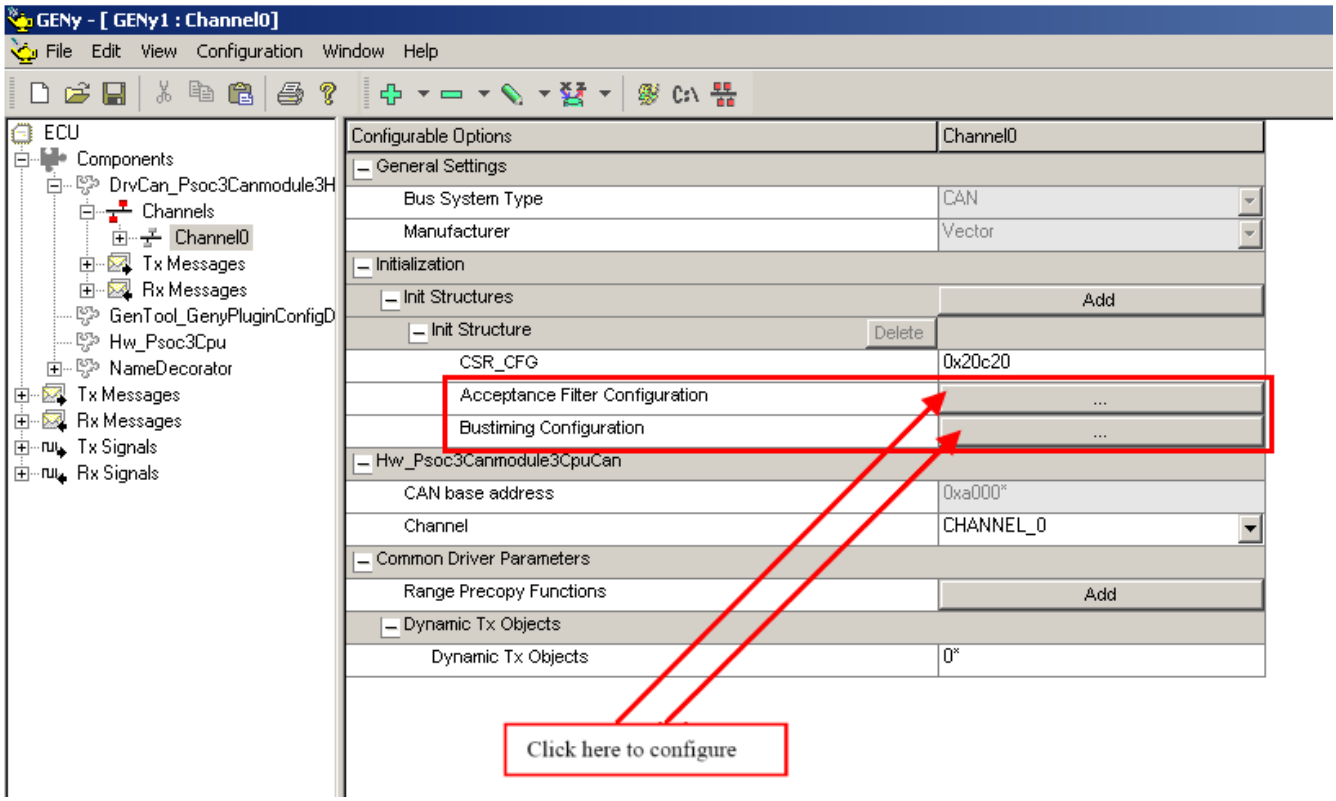
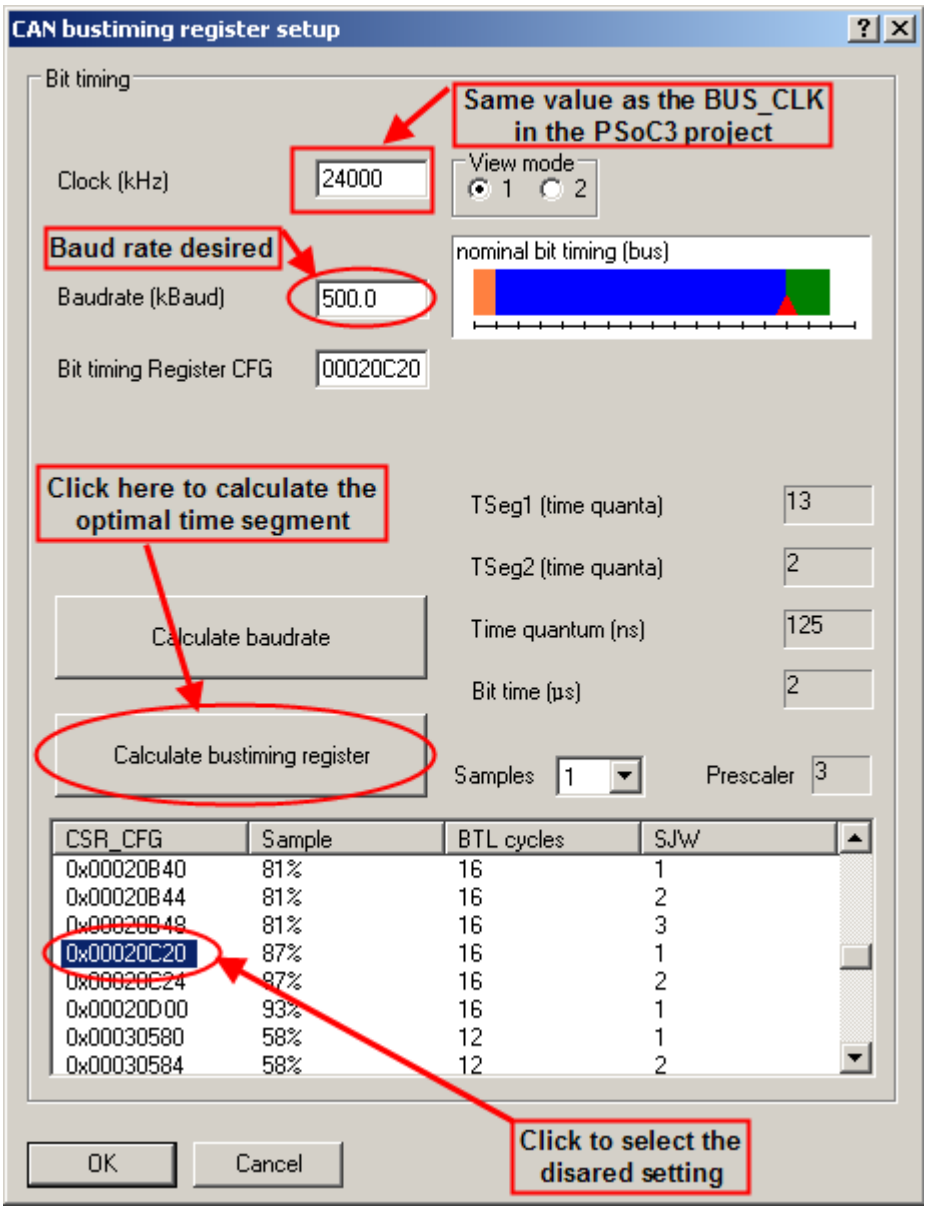


図 5. バス タイミング設定



CAN bustiming (CAN バスタイミング) ウィンドウで選択されるクロックは、PSoC 3 プロジェクトの BUS_CLK とおなじでなければなりません。

ポーリングまたは割り込みモードを選択する場合は、ポーリング モードでは、アプリケーションは保留イベントまたはメッセージを処理するために CanTask() 関数を呼び出す必要があります。割り込みモードでは、イベントまたはメッセージは、割り込みサービス ルーチン CanIsr_0() によって処理されます。

受け入れフィルタとバス タイミングおよびメールボックス ポーリング/割り込みモードをセットアップしたら、CAN ドライバ ファイル ディレクトリを選択します。

これらのディレクトリを PSoC プロジェクト ディレクトリ内に作成し (以前に説明したように)、プロジェクトに関連するものがすべて 1 つの場所に配置されるようにします。

図 6. ポーリングまたは割り込みモードの選択

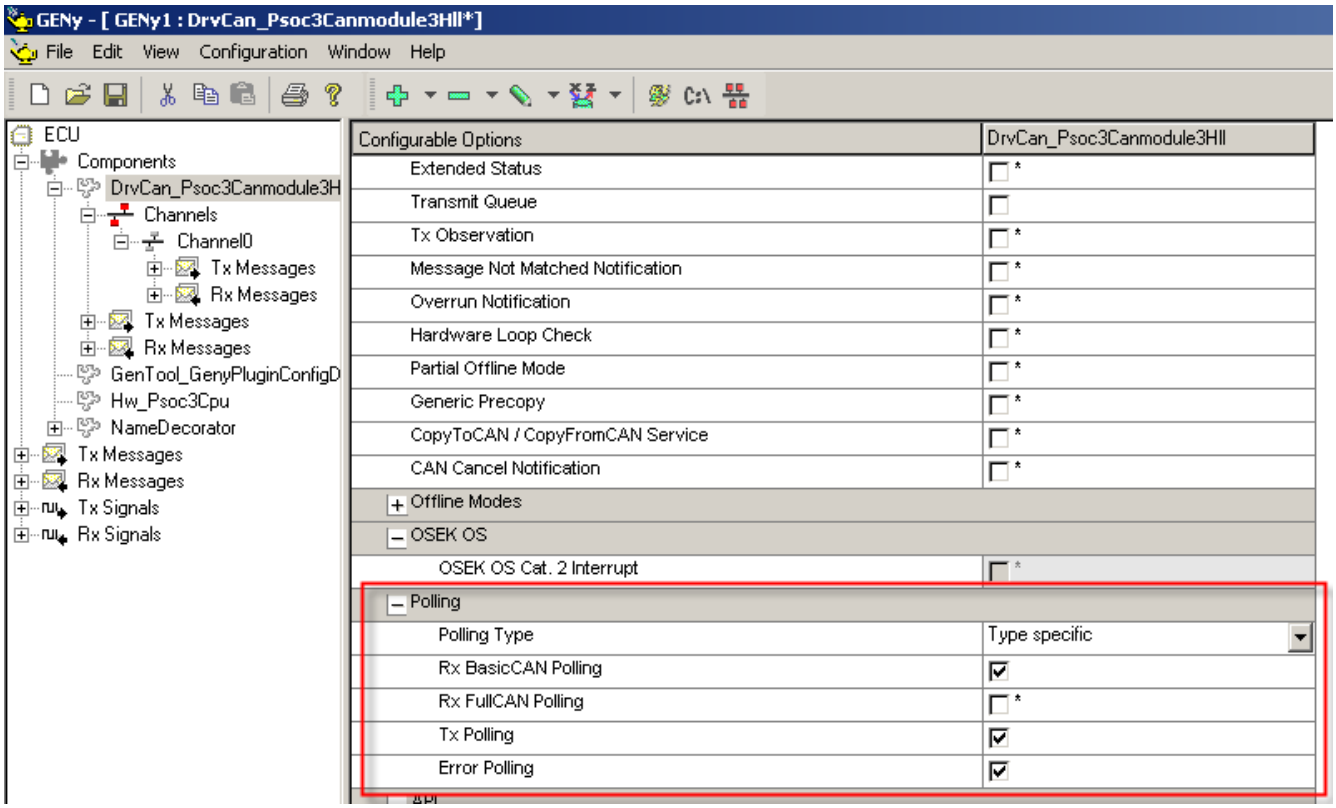


図 7. 保存先ディレクトリの選択

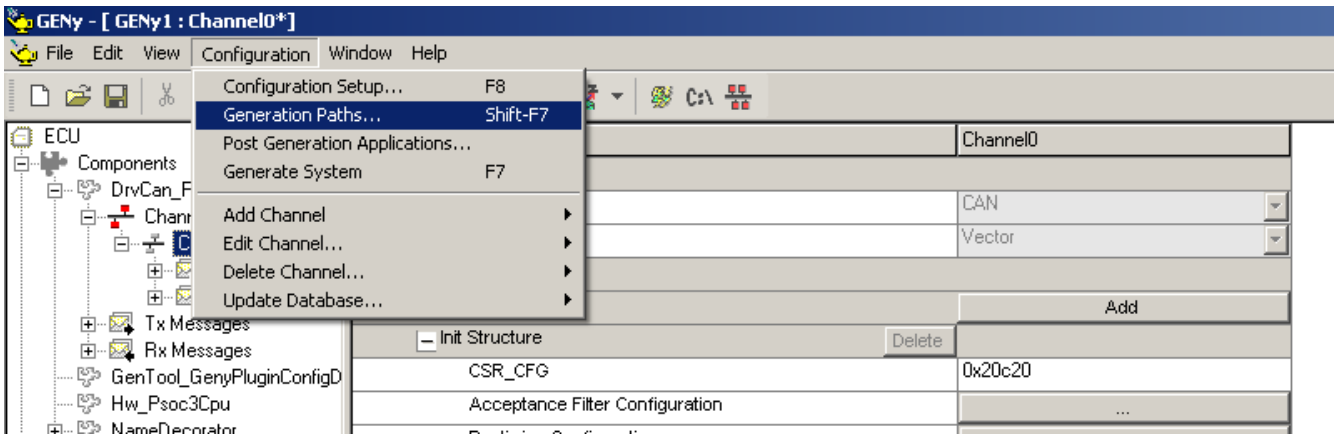
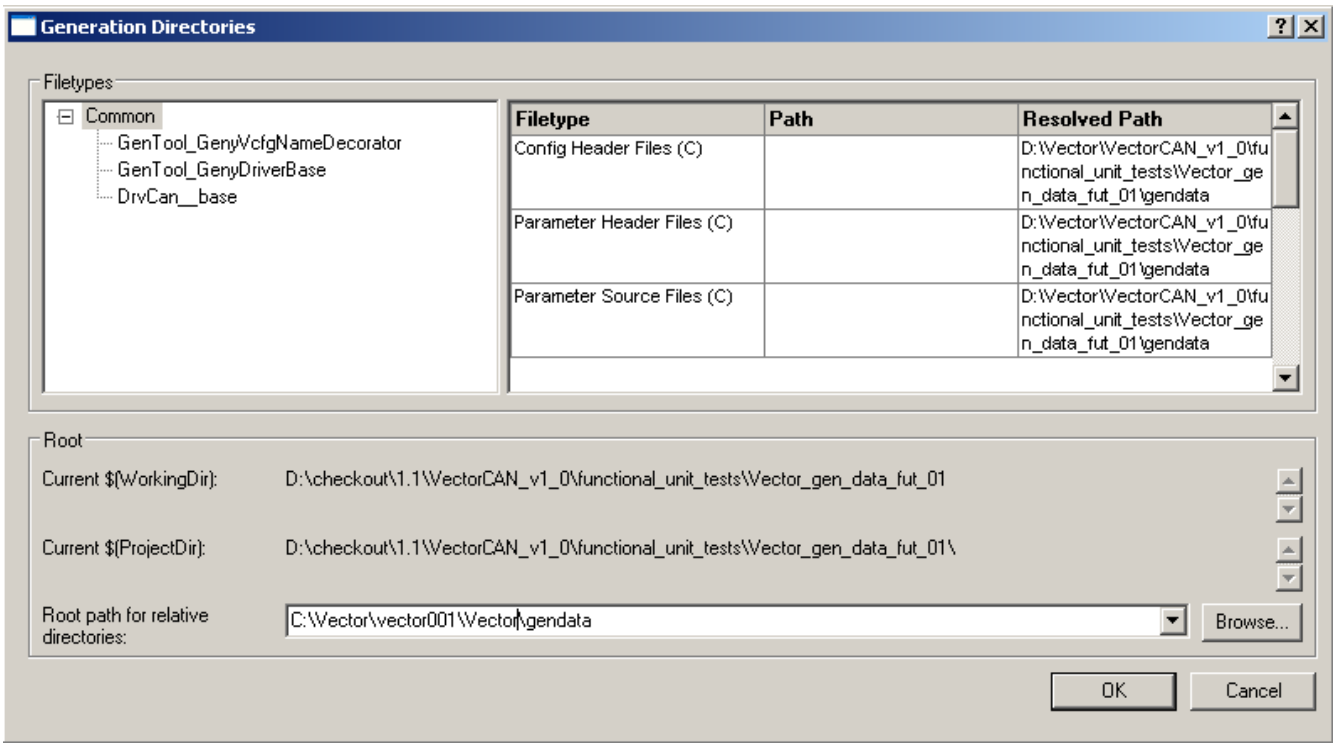


図 8. 相対ディレクトリのルートパスの選択



Generation Directories ダイアログ ウィンドウで、**Root path for relative directories (相対ディレクトリのルートパス)** を PSoC プロジェクト ディレクトリに作成されたサブディレクトリに設定します。

5. CAN ドライバ ファイルの作成

この時点で、ツールは PSoC ドライバ ファイルを生成する準備が整います。これは、PSoC プロジェクト内に作成され、選択したディレクトリに配置されます。.

また、設定ファイルを PSoC プロジェクト ディレクトリに保存し、変更が必要になり、新しいファイルが生成されたときに、すべてが同じプロジェクトディレクトリ内にあるようにします。

これらのファイルは、**Generate System** ボタンが押されるごとに生成されますが、一部の共通ファイル (変わらない) もあります。これは、同じ PSoC プロジェクト ディレクトリにコピーする必要があります。

次の図は、プロジェクトをビルドするために PSoC Creator に移動する前の最後の手順について説明しています。

図 9. “Generate System”をクリックして、PSoC プロジェクト サブディレクトリにファイルを生成します

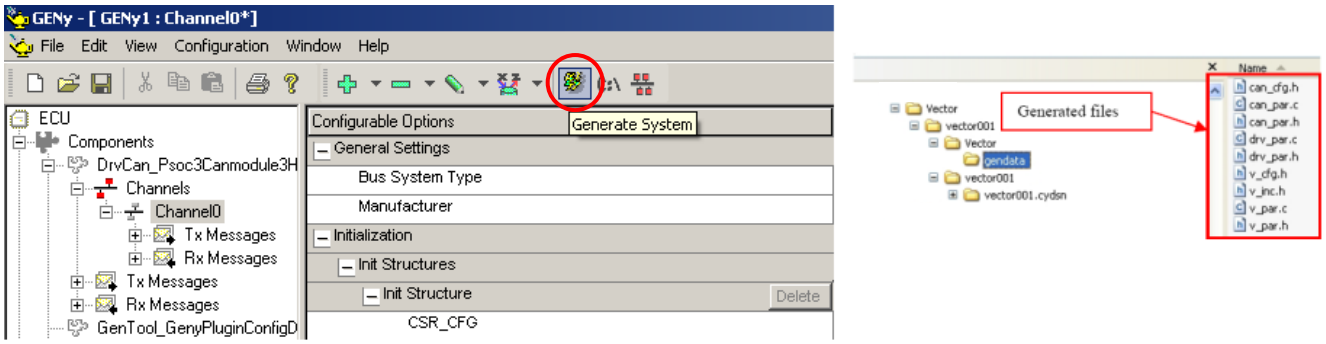
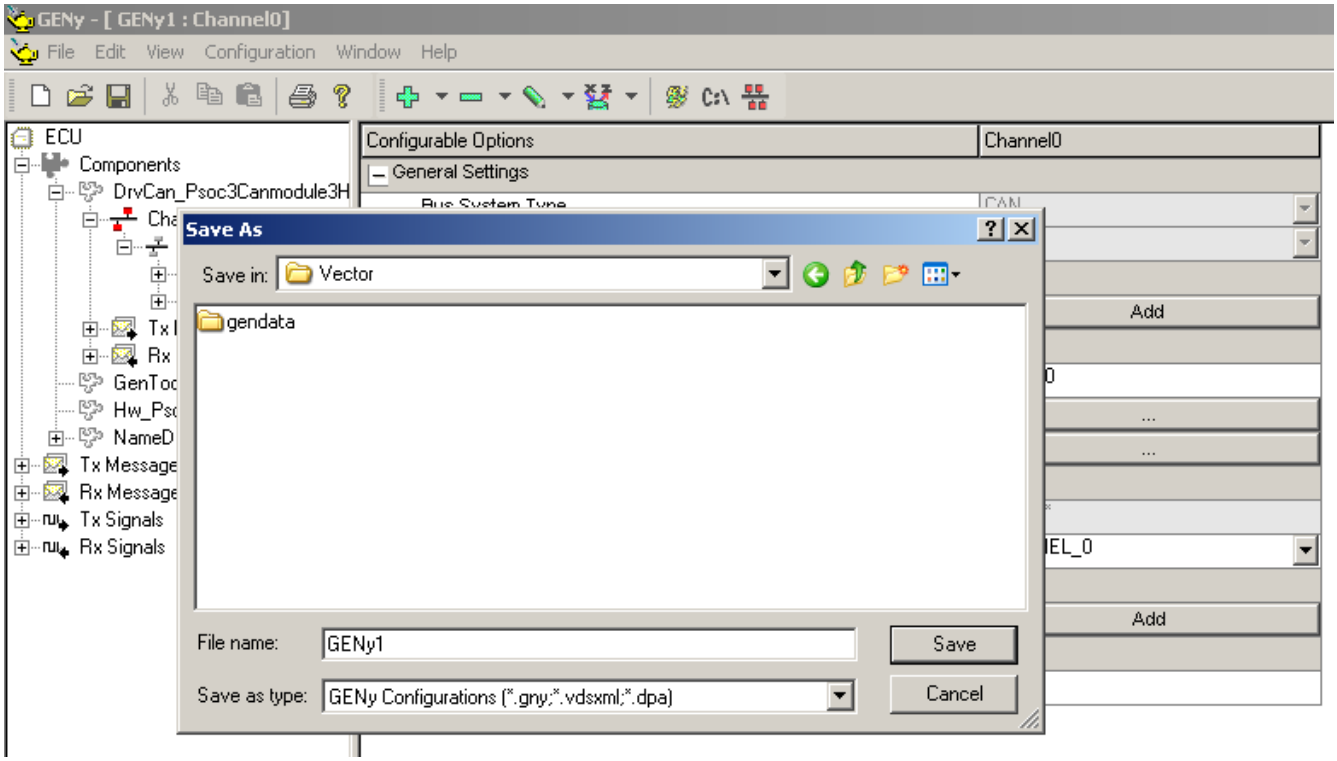


図 10. Vector GENy Configuration を PSoC プロジェクト サブディレクトリに保存する

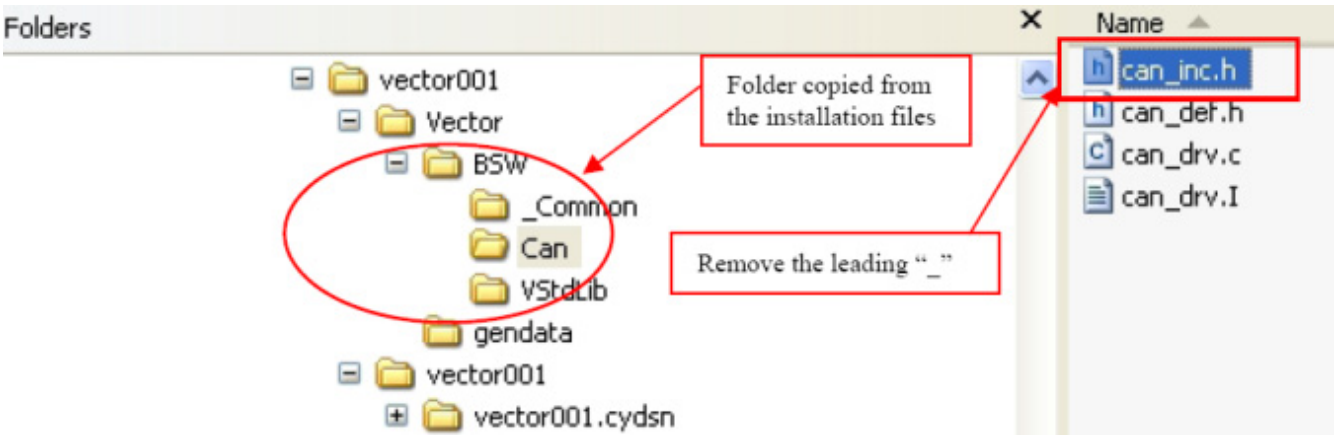


インストール時に、デフォルトのディレクトリが承認された場合は、共通ファイルは次の場所に配置されます:

`C:\Vector\CBD0810092_D00_Psoc3\BSW`

BSW ディレクトリ全体を PSoC プロジェクト フォルダにコピーして、サブディレクトリ内の *_can_inc.h* のファイル名を *can_inc.h* に変更 (語頭の “_” を削除) する必要があります。

図 11. PSoC プロジェクト フォルダにコピーされた CAN ドライバ共通ディレクトリ



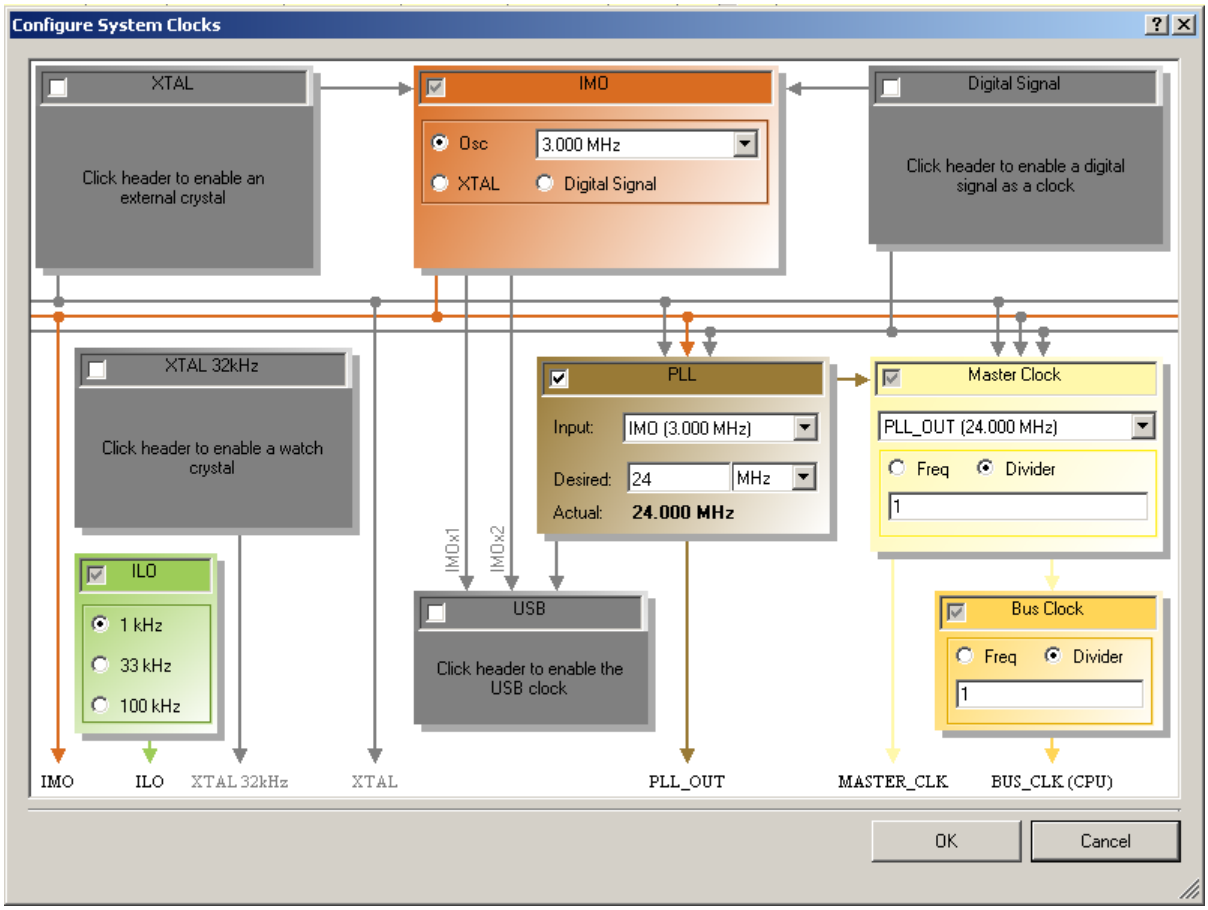
これで、Vector GENy ツールを閉じることができます。PSoC プロジェクトを続行するために、すべてがセットアップされました。

6. PSoC プロジェクトの作成

1. Vector CAN コンポーネントを回路図に配置し、Configure ダイアログを開いてコンポーネント オプションをカスタマイズして、割り込みを有効にし、tx_en 出力を使用します。
2. クロック ツリーをセットアップします。
3. ピンを配置します。
4. Vector CAN ドライバ ファイルをインポートします。
5. “Build settings… (ビルド設定)” を変更して、CAN ドライバ ディレクトリを含め、NOOVERLAY オプションを設定します。
6. アプリケーションを *main.c* に書き込みます。

クロック ツリーのセットアップ

図 12. クロック セットアップ



ピンの配置

図 13 に示す例では、ピンは CAN/LIN EBK (CY8CKIT-017) で使用されるように配置されています。任意のピンを使用できます。

図 13. ピン選択

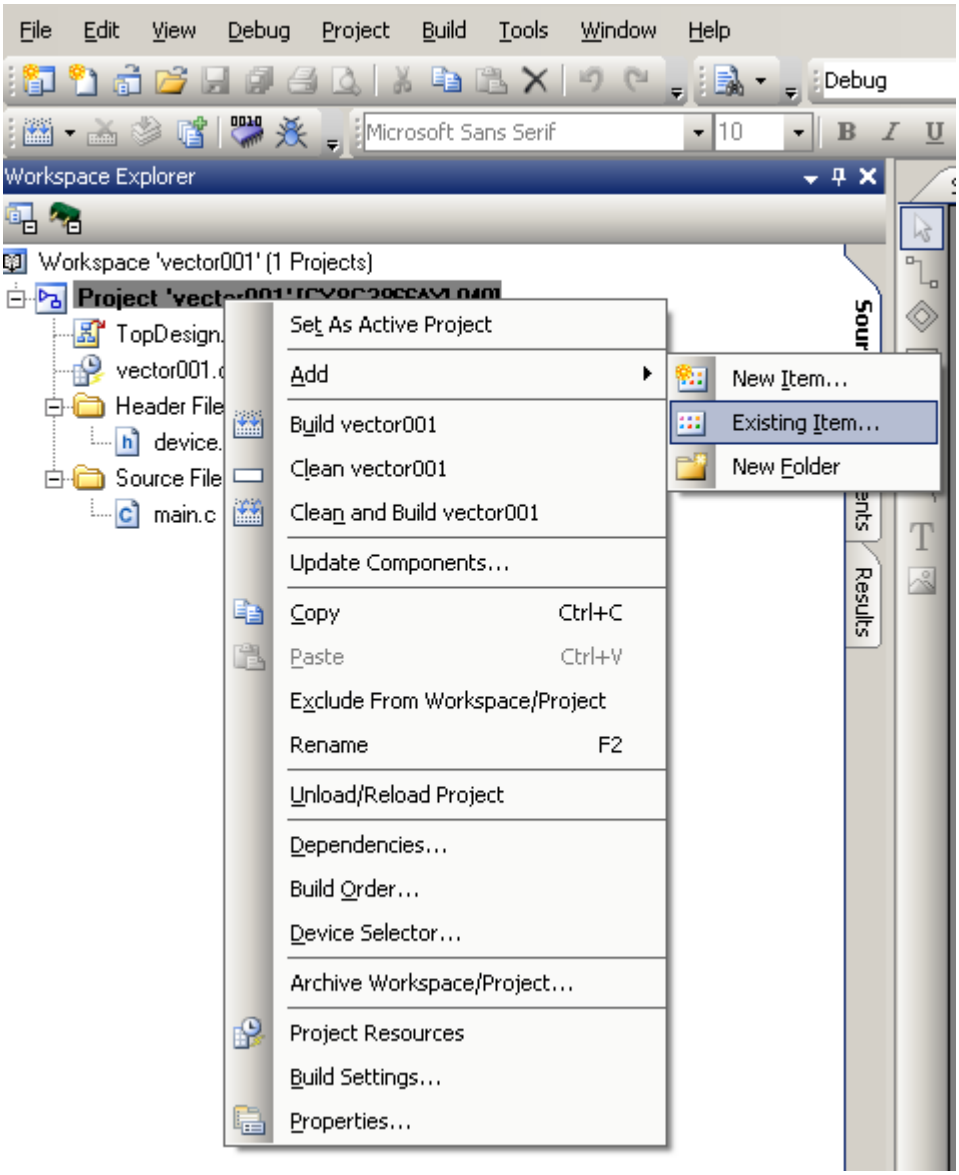
Alias	Name	Pin	Lock
	Pin_tx_en	P3[1]	☑
	Pin_tx	P3[3]	☑
	Pin_rx	P3[2]	☑

Vector CAN ドライバ ファイルのインポート

ヘッダーファイルとソースファイルの両方を “gendata” ディレクトリと “BSW” の下のすべてのフォルダからインポートする必要があります。

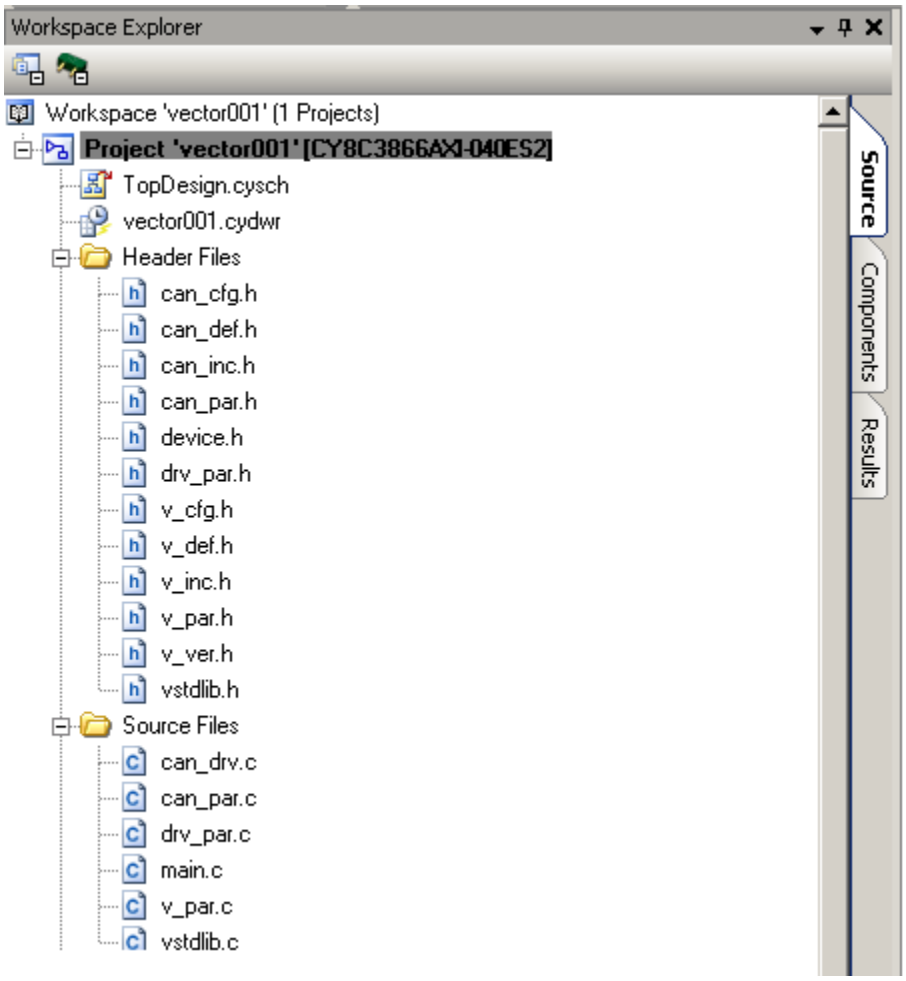
ファイルをインポートするには、PSoC Creator ワークスペース エクスプローラで **Header Files** と **Source Files** を右クリックし、既存のアイテムを追加するメニューを選択します。すべての Vector ファイルがインポートされるまでプロセスを繰り返します。

図 14. 既存のヘッダーファイルとソースファイルのインポート



インポート手順が完了すると、ワークスペース エクスプローラは図 15 のような外観になります。

図 15. インポート後のファイル リスト



“Build settings…” を変更して、CAN ドライバ ディレクトリを含める

図 16. インクルード ディレクトリの追加

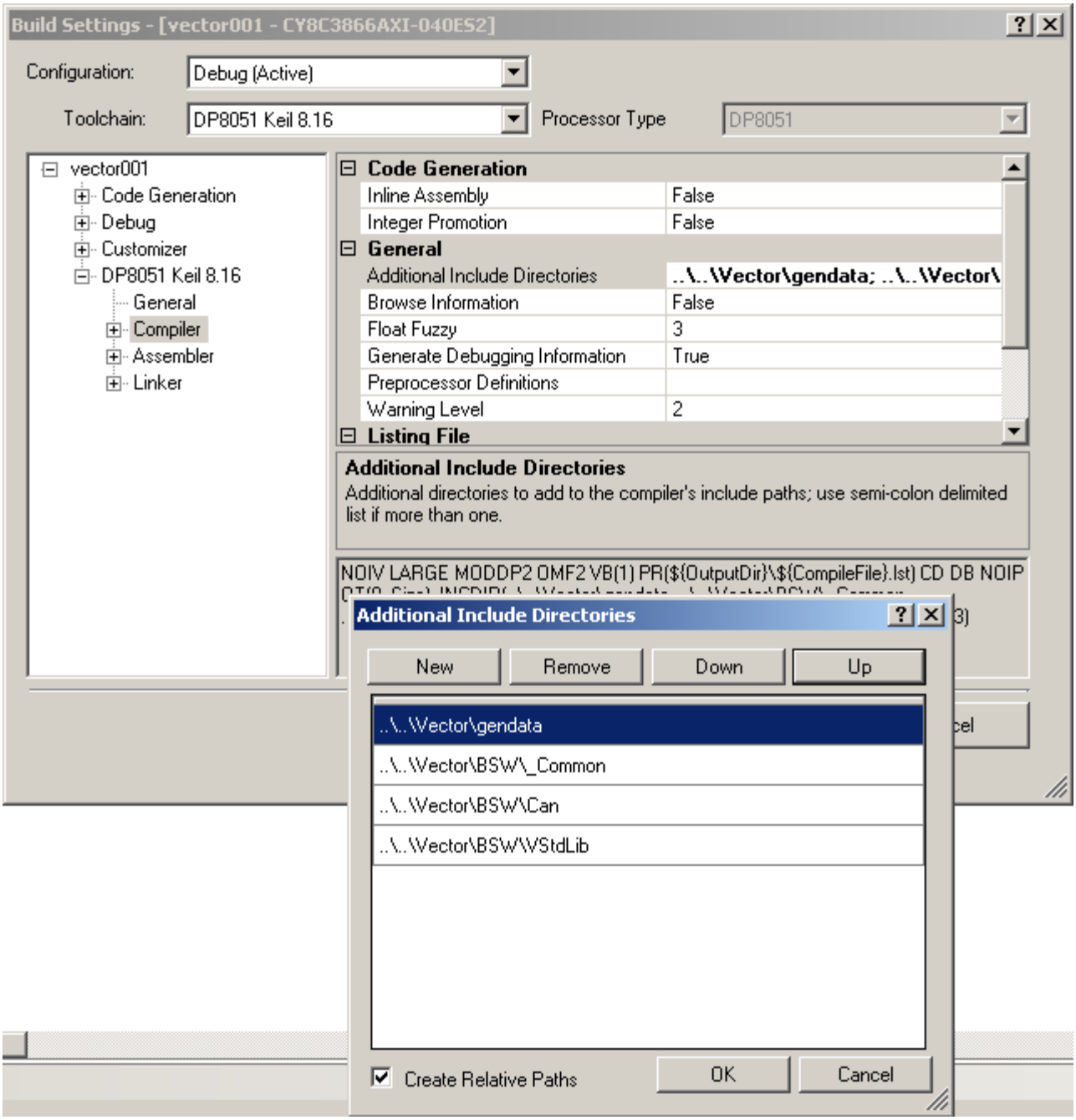
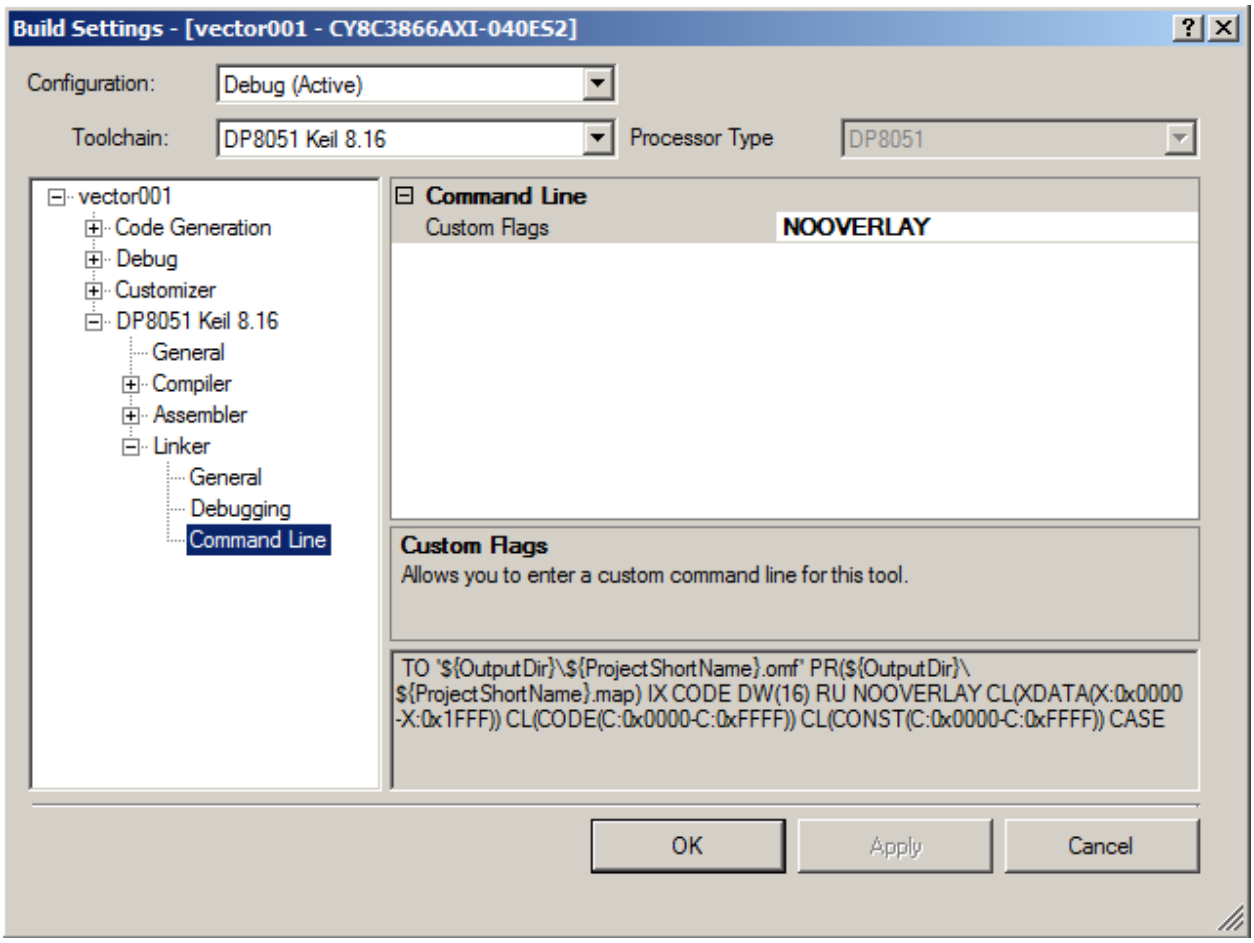


図 17. NOOVERLAY オプションの設定



Vector CAN Driver APIs は関数ポインタを使用します。PSoC 3 の Keil コンパイラは、関数呼び出し解析を行って、関数の変数と引数をどのようにオーバーレイできるか調べます。コンパイラが呼び出し構造を十分に解析できないことを、関数ポインタが表している場合は、NOOVERLAY オプションが選択され、関数ポインタの使用のために問題が発生するのを防ぎます。Keil コンパイラで関数ポインタを操作するための詳しい情報は、次のアプリケーション ノートで入手できます: *Function Pointers in C51* (www.keil.com/appnotes/docs/apnt_129.asp)。

main では、初期化プロセスは以下を行う必要があります:

- ドライバの *v_inc.h* ファイルを main.c に含めます。
- 必要に応じて、グローバル割り込みを有効にします。
- Vector_CAN_Start() 関数を呼び出します。
- CanInitPowerOn() 関数 (Vector GENy ツールによって生成される) を呼び出します。

- Vector CAN の API と Vector GENy ツールによって生成された関数を使用して、必要な機能を記述します。

DC 電気的特性と AC 電気的特性

以下の値は、期待される性能を示しており、初期特性データを基にしています。

CANのDC仕様

パラメータ	説明	条件	Min	Typ	Max	単位
	ブロックの電流消費	500 kbps	--	--	285	µA
		1 Mbps	--	--	330	µA

CANのAC仕様

パラメータ	説明	条件	Min	Typ	Max	単位
	ビット レート	最低 8-MHz クロック	--	--	1	Mbit

コンポーネントの変更

バージョン 1.0 は Vector CAN コンポーネントの最初のリリースです。

Copyright © 2005-2012 Cypress Semiconductor Corporation 本文書に記載される情報は、予告なく変更される場合があります。Cypress Semiconductor Corporation は、サイプレス製品に組み込まれた回路以外のいかなる回路を使用することに対しても一切の責任を負いません。特許又はその他の権限下で、ライセンスを譲渡又は暗示することもありません。サイプレス製品は、サイプレスとの書面による合意に基づくものでない限り、医療、生命維持、救命、重要な管理、又は安全の用途のために仕様することを保証するものではなく、また使用することを意図したものでもありません。さらにサイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことを合理的に予想される、生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

PSoC Designer™及び Programmable System-on-Chip™は、Cypress Semiconductor Corp.の商標、PSoC®は同社の登録商標です。本文書で言及するその他全ての商標又は登録商標は各社の所有物です。

全てのソースコード(ソフトウェア及び/又はファームウェア)は Cypress Semiconductor Corporation(以下「サイプレス」)が所有し、全世界(米国及びその他の国)の特許権保護、米国の著作権法並びに国際協定の条項により保護され、かつそれらに従います。サイプレスが本書面によるライセンシーに付与するライセンスは、個人的、非独占的かつ譲渡不能のライセンスであって、適用される契約で指定されたサイプレスの集積回路と併用されるライセンシーの製品のみをサポートするカスタムソフトウェア及び/又はカスタムファームウェアを作成する目的に限って、サイプレスのソースコードの派生著作物を複製、使用、変更、そして作成するためのライセンス、並びにサイプレスのソースコード及び派生著作物をコンパイルするためのライセンスです。上記で指定された場合を除き、サイプレスの書面による明示的な許可なくして本ソースコードを複製、変更、変換、コンパイル、又は表示することは全て禁止されます。

免責条項:サイプレスは、明示的又は黙示的を問わず、本資料に関するいかなる種類の保証も行いません。これには、商品性又は特定目的への適合性の黙示的な保証が含まれますが、これに限定されません。サイプレスは、本文書に記載される資料に対して今後予告なく変更を加える権利を留保します。サイプレスは、本文書に記載されるいかなる製品又は回路を適用又は使用したことによって生ずるいかなる責任も負いません。サイプレスは、誤動作や故障によって使用者に重大な傷害をもたらすことが合理的に予想される生命維持システムの重要なコンポーネンツとしてサイプレス製品を使用することを許可していません。生命維持システムの用途にサイプレス製品を供することは、製造者がそのような使用におけるあらゆるリスクを負うことを意味し、その結果サイプレスはあらゆる責任を免除されることを意味します。

ソフトウェアの使用は、適用されるサイプレスソフトウェアライセンス契約によって制限され、かつ制約される場合があります。