Applilet3共通 スタートアップガイド

デバイス・ドライバ・コンフィギュレータ（Applilet3）

マイコンを**動かしたい？**

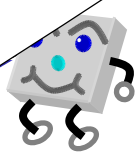
まず**「マニュアル」を引き出し**にしまって下さい

そんなツールです

ルネサス エレクトロニクス株式会社
MCU事業本部 ソフトウェア統括 MCUツール技術部

2010/10/29 Rev1.00

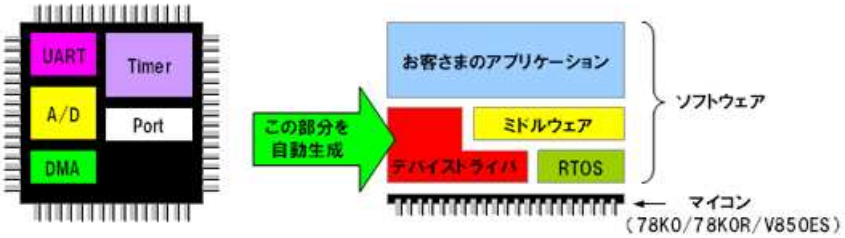
ZUD－CD－10－0180（原紙承認済み）



もくじ

Appliletの情報

Appliletのダウンロード方法、情報は下記URLに掲載しています。
http://www2.renesas.com/micro/ja/development/asia/applilet3/



掲載情報

紹介資料

Appliletの概要と使い方、目的、サポートデバイスが理解できます。
・Applilet3共通　紹介資料（パンフレット）
・Applilet3共通　スタートアップガイド（この資料）

ユーザーズマニュアル操作編

Applilet3のインストール、ウインドウの説明など、Applilet3の共通操作を示したユーザーズマニュアルです。マイクロコントローラの区別なく、Applilet3をお使いになる全ての方に、読んで頂きたいマニュアルです。
・Applilet3共通　操作編ユーザーズマニュアル

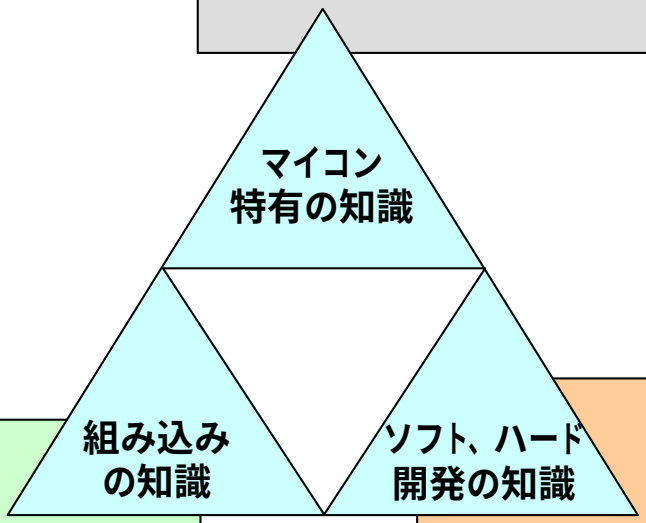
ユーザーズマニュアルAPI操作編

78K0、78K0R、V850はそれぞれ8bit、16bit、32bitのマイクロコントローラです。それぞれのマイクロコントローラに対応したAPI(Application User Interface)を説明しているマニュアルです。
・Applilet3 78K0マイクロコントローラ　APIリファレンス編
・Applilet3 78K0Rマイクロコントローラ APIリファレンス編
・Applilet3 V850マイクロコントローラ　APIリファレンス編

マイコンシステム開発に必要なこと

マイコンシステムを
開発するのに
必要なことは？

・デバイスの学習
・デバイスの使い方,クロックの設定、周辺機能の初期化、設定にかかわる計算式
・消費電流、消費電力、端子の耐圧、寄生容量
・リセット処理、フラッシュプログラミング、書き込み処理、デバッグ環境の構築



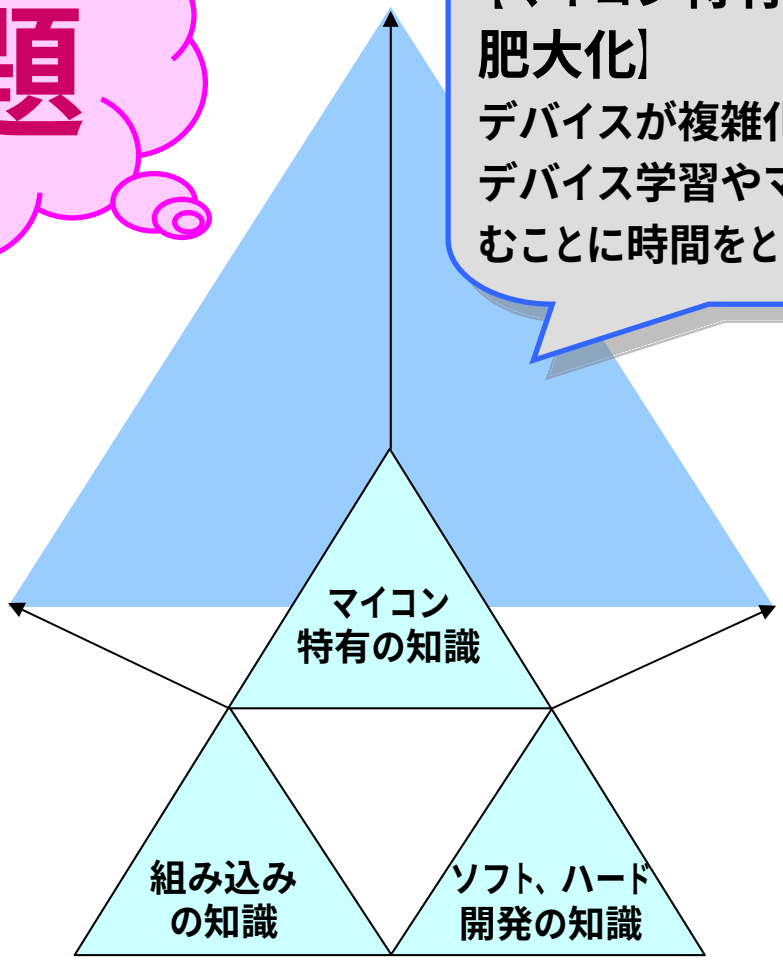
・割り込み制御
・リアルタイム制御
・電源ON～電源OFFまでに処理することが理解できる
・ソフトウェア、ハードウェアの切り分け
・UART、I2C、CSI（SPI）、DMAなどマイコンの周辺機能に関する知識（通信規格、概念）

・ISO9126（ソフトウェア品質に関する国際規格）
・開発手法（CMMI、アジャイル、XP、ウォーターフォール、その他）
・回路の基礎知識
・各種測定器の使い方（オシロスコープ、ロジックアナライザ、通信系のジェネレータ、アナライザ）
・リアルタイムOS（T-Kernel、ITRON、Toppers、Linuxなど）

マイコンシステム開発に必要なこと

問題

【マイコン特有の知識が肥大化】
デバイスが複雑化し、ユーザは、デバイス学習やマニュアルを読むことに時間をとられてしまう。



目的

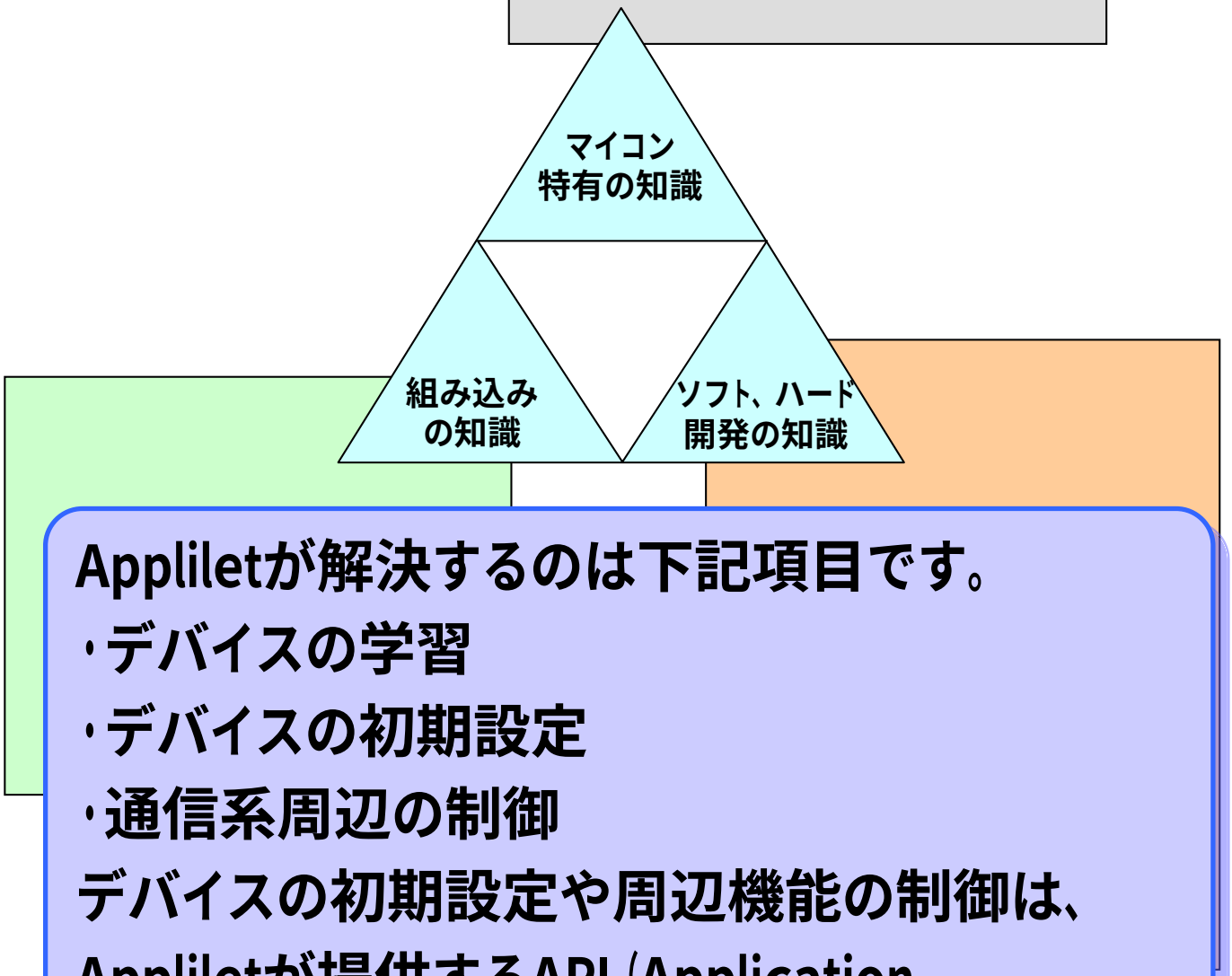
ユーザは組み込みの知識、開発の知識を求めていてる。しかし、マイコン特有の知識は求めていない。開発効率の向上、開発知識の蓄積が本来の目的。

Applilet 導入のメリット

解決

・デバイスの学習
・デバイスの使い方,クロックの設定、周辺機能の初期化、設定にかかわる計算式

・消費電流、消費電力、端子の耐圧、寄生容量
・リセット処理、フラッシュプログラミング、書き込み処理、デバッグ環境の構築



Appliletが解決するのは下記項目です。
・デバイスの学習
・デバイスの初期設定
・通信系周辺の制御
デバイスの初期設定や周辺機能の制御は、Appliletが提供するAPI（Application Programming Interface）を使います。

Q. Appliletは初心者用のツールでしょ？

A. いいえ、上級者でもお使いください。**Appliletはマイコン特有の学習を省くためのツール**です。

Q. CubeSuiteの「コード生成」と何が違うの？

A. 機能は同じです。**Appliletは、PM+、IAR、Multi (V850のみ) のプロジェクトを生成**します。

Q. マイコンの初期化だけを行うの？

A. いいえ、初期化だけではありません。**割り込みハンドラや、通信系の周辺機能を制御するAPIを提供**します。

Q. 全てのマイコンをサポートしていないのはどうして？

A. 周辺機能はマイコンによって、それぞれ異なります。また、パッケージの種類も豊富にあります。ツールとして安心してお使い頂くためリリース可能と判断したものを順次お出しします。

Q. Appliletがあれば、プログラムなんて簡単にできるよね？

A. Appliletは万能ではありません。**組み込み技術の知識、マイコンの基礎知識は必要**です。この資料で、知っておいて欲しい基礎知識を説明します。

Q. もちろん、Appliletが生成するソースはコード保証するよね？

A. いいえ、コード保証はできません。なぜなら、お客様のコードが追加されることを前提としたツールです。検証については、**お客様のコードと共にシステム検証を充分**に行ってください。


マイコンの基礎
これだけは知っておきたい
チャタリングの原理を
理解して、原因を解決できる。
チャタリングの原理を理解していても
最適な解決方法がわかりますか？例え
ると、ハードで解決するのか、ソフトで解決
するのか。それにかかるコストは？一つの
事を処理するにしても多方面から検討
できる。ソフトとハードの
問題の切り分け
が判断可能。
マイコンの電源をONに
してから、main()が実行される
シーケンス（流れ）を理解している。
main()が実行されるまでにマイコンは
様々な処理を行います。割り込みハンドラの
設定や各周辺機能のハードウェア初期化
スタックポインタの設定、グローバル変数
の初期化など多くの処理を行っています。
これらの処理を理解している必要
があります。
ソフトウェアの基礎
ISO9126を知っている。
完全には理解していなくても
ソフトウェアで重要なことは何か
どうすればよいのかを知っている。
具体的な解決方法は経験しながら
学ぶものです。問題を認識して
いるのと、していないのでは
学習に差がでます。
回路図を書けない
が読むことはできる。
ソフトウェア技術者でも
最低限の知識が必要です。
ハードを理解するために
回路図を読めるように
なってください
マイコンのUM
を読める。実は読んでも
理解できない技術者がいま
す。完全に理解できなくても
読めばわかるように
しましょう。
プルアップ、プルダウン
の意味や特性を知っている。
外部割り込み、ポートの入出力
に必要になるのはプルアップと
プルダウンの設定です。マイ
コンの内部でもプルアップ
設定があります
割り込みハンドラの意味を
理解しており、割り込みタイミングや
ハンドラ内ですべき事を完全に把握している。
組み込みプログラムは、イベントドリブンで動作
させます。割り込みが発生していないときプログラム
は何もせず待機状態にさせ、CPUはHalt状態にして
電力消費を行わないようにします。このイベントドリ
ブン方式は、一般的な方法です。OSを使う場合でも
用いられます。初心者に見られるプログラムの
書き方に割り込みハンドラへ処理を入れ込み
過ぎてタイマの周期を超えている場合
をよく見受けます。
バグが発生しても
最終的には解決できる。
どんなデバッグ方法でも構いません。
求められるのは、自己解決能力です。
このような技術力がこれから必要に
なってきます。なぜなら、新技術や新
規格が絶えず使われるからです。
それらの技術は、全て把握できる
ものではありません。その都度に
対応できる技術が必要
です

AppliletとISO9126の関係

【移植性】
Applilet**はマイコンの初期化コードや物理層部分を担当します。他マイコンへ移植を行う場合、**AppliletのAPI**を入れ替えるだけで簡単に移行できます。**

【保守性】
ユーザは自分の書いたコードを管理します。マイコンに関する部分は全て Applilet**が担当するので、保守しやすくなります。**

ISO9126
ソフトウェア品質に関する国際規格

【機能性】
現在、UART、CSI(SPI)、I2C**などの通信系をサポートしています。更に**USB、Ethernet**など機能を強化する予定です。**

【効率性】
ユーザのコードに大きく依存します。ただし、Applilet**を使うことで開発効率は格段に上がります。**

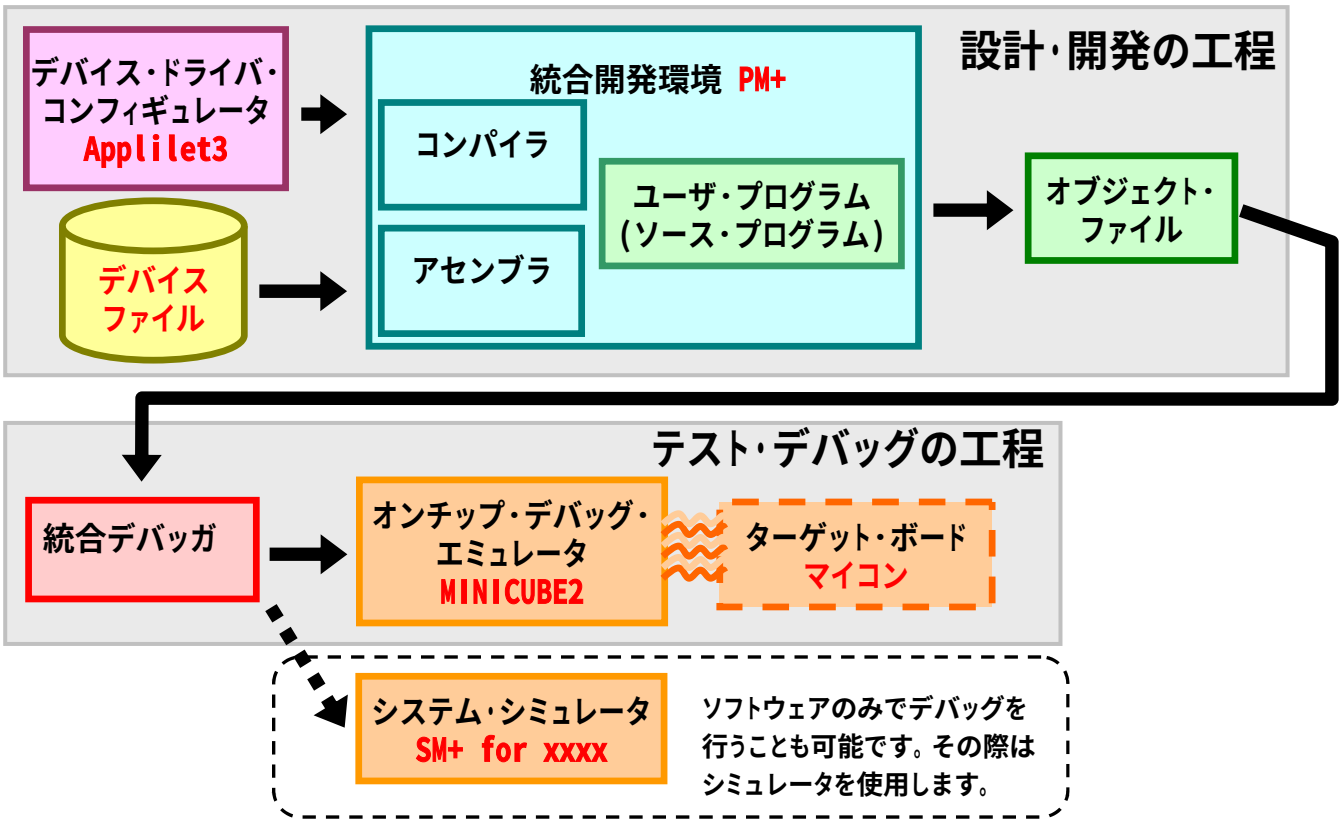
【使用性】
Applilet**が提供するAPIにはルールがあります。そのため、**Applilet**を使うことで、ソースも見やすくなります。GUIで設定できるので、簡単に使えるツールになっています。**

【信頼性】
Applilet**が提供するAPIは、充分評価されています。ただし、ユーザ作成部分を含めての評価方法の確立が必要と考えています。**

= Appliletが対象とするISO9126の項目

= Appliletが機能強化したいISO9126の項目

Appliletを使ったときの開発工程



用語

Applilet3
マイコンのシリーズ品種毎に存在します。サポートマイコンについては、WEB掲載の「Applilet3共通　紹介資料」を参照してください。

デバイスファイル
マイコンの品種依存情報ファイルです。PM+が参照します。

コンパイラ/アセンブラ
ソースプログラムをオブジェクトファイルへ生成します。マイコンによって使用するツールが異なります。
・78K0マイコン　CC78K0、CA78K0
・78K0Rマイコン　CC78K0R、CA78K0R
・V8500マイコン　CA850

統合デバッガ
接続するエミュレータによって使える機能が変わります。下記はMINICUBE2を使用した場合に対応するデバッガです。また、MINICUBE2はモニタ方式でデバッグするのでユーザ資源を使います。Appliletは、そのユーザ資源の設定を自動的に行い、効率よくデバッグ可能です。
・78K0マイコン　ID78K0-QB
・78K0Rマイコン　ID78K0R-QB
・V8500マイコン　ID850QB

Appliletがサポートする周辺機能の一覧

[SYSTEM]
CPU、周辺の動作クロック、オンチップデバッグの設定

[PORT]
ポートの入出力、プルアップ、ダウン、初期値の設定

[割り込み]
キー割り込み、外部(INTP)割り込みの設定

[タイマ]
インターバル、方形波、外部イベントカウンタ、ワンショットパルス、PPG、パルス幅測定、PWM、キャリアジェネレータの設定

[A/D変換]
入力端子、コンパレータ、変換時間、変換方式の設定

78K0

[ウォッチドッグ]
オーバーフロー、ウィンドウオープン時間の設定

[RTC]※1
アラーム、リアルタイムカウンタ動作、定周期の設定

[シリアル]
UART、CSI(SPI)、I2C、DALI※1、データ長、転送レート、転送方向、パリティ、スレーブ/マスタ、自局アドレスの設定

[オペアンプ]※1
PGA動作、AMP動作、コンパレータの設定

[クロック出力]
出力クロックの設定

[低電圧検出]
LVI動作、検出電圧、検出時動作の設定

※1 マクロ搭載品のみ対応

[SYSTEM]
CPU、周辺の動作クロック、オンチップデバッグの設定

[PORT]
ポートの入出力、プルアップ、ダウン、初期値の設定

[割り込み]
キー割り込み、外部(INTP)割り込みの設定

[タイマ]
インターバル、方形波、外部イベントカウンタ、ワンショットパルス、PPG、パルス幅測定、PWM、キャリアジェネレータの設定

[A/D変換]
入力端子、コンパレータ、変換時間、動作モードの設定

78K0R

[ウォッチドッグ]
オーバーフロー、ウィンドウオープン時間の設定

[RTC]
アラーム、リアルタイムカウンタ動作、定周期の設定

[シリアル]
UART、CSI(SPI)、I2C、データ長、転送レート、転送方向、パリティ、スレーブ/マスタ、自局アドレスの設定

[コンパレータ]※2
PGアンプ動作、コンパレータの設定

[クロック出力]
出力クロック、ブザーの設定

[低電圧検出]
LVI動作、検出電圧、検出時動作の設定

[DMA]
転送方向、データサイズ、起動要因、転送回数の設定

[D/A変換]
リアルタイム、モード、変換値、基準電圧の設定

[外部バス]※2
モード、アクセス幅、外部ウェイト、メモリ領域の設定

[LCDコントローラ]※3
駆動電圧回路、表示モード、基準電圧、表示データ領域、セグメント出力端子、クロックソースの設定

※2 マクロ搭載品のみ対応
※3 Applilet3 for 78K0R/Lx3のみ対応

Appliletがサポートする周辺機能の一覧

[SYSTEM]
CPU、周辺の動作、サブクロック、PLL、SSCG、クロックアウト、スタンバイ、ウォッチドッグタイマ，オンチップデバッグの設定

[割り込み]
キー割り込み、外部(INTP)割り込みの設定

[タイマ]
インターバル、外部イベントカウンタ、トリガパルス、ワンショットパルス、パルス幅測定、PWM、フリーランニングの設定

[A/D変換]
入力端子、コンパレータ、変換時間、動作モードの設定

V850

[外部バス]
オーバーフロー、ウィンドウオープン時間の設定

[RTC]※1
アラーム、リアルタイムカウンタ動作、定周期の設定

[シリアル]
UART、CSI(SPI)、I2C、データ長、転送レート、転送方向、パリティ、スレーブ/マスタ、自局アドレスの設定

[PORT]
ポートの入出力、プルアップ、ダウン、初期値の設定

[リアルタイム出力]
動作、信号の有効エッジの設定

[低電圧検出]
LVI動作、検出電圧、検出時動作の設定

[DMA]
転送方向、データサイズ、起動要因、転送回数の設定

[D/A変換]
リアルタイム、モード、変換値、基準電圧の設定

[時計タイマ]
動作クロック、インターバルの設定

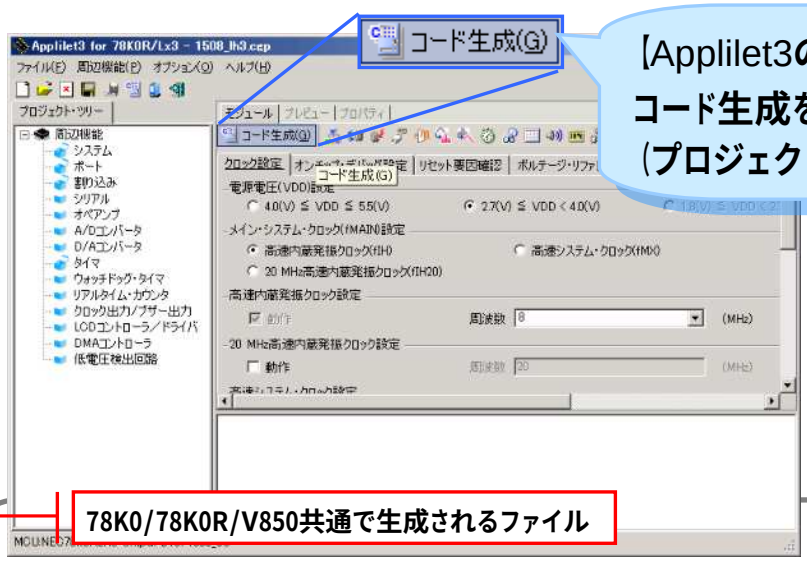
[CAN]※2
CANドライバとのインタフェースの設定

[モータ]※2
波形パターン、書き込み、ハイインピ-ダンスの設定

※1 マクロ搭載品のみ対応
※2 Applilet2 for V850ES/Fx3のみ対応

Appliletを使って、サポートできないマイコンの機能があります。
・USBホスト、USBファンクション、LIN、イーサネット
・フラッシュ・セルフ・プログラミング、EEPROMエミュレーション
・インバータ制御機能(78K0R/Ix3)
以上の機能につきましては、アプリケーションノートなどを参考にしてください。

Appliletが生成するファイル



【Applilet3の画面】
コード生成をクリックし、ファイルを生成する。
(プロジェクト名が「sample」の場合)

【必ず生成されるファイル】

CG_main.c	main() 関数を含むメインのプログラム
CG_systeminit.c	スタートアップからcallされる hdwinit() を含む、ハードウェア初期化プログラム
CG_system.c / h	クロック初期化関数を含む。上記の hdwinit() からcallされる
CG_system_user.c	ユーザが追加する初期化処理 (空ファイル)
CG_macrodriver.h	Appliletが共通で使用する定義を含む。生成する c ソースに incilude される
CG_userdefine.h	ユーザが共通で使用する定義を含む。生成する c ソースに include される
CG_lk.dr (78K0/R)	リンクディレクティブファイル。78K0と78K0Rで使用する
CG_option.asm (78K0)	オプションバイトの定義ファイル。78K0で使用する
CG_lk.dir (V850)	リンクディレクティブファイル。デフォルトの設定を定義する
CG_inttab.s (V850)	割り込みベクタの定義ファイル
CG_start.s (V850)	V850の場合のスタートアップルーチン
sample.prj	統合開発環境 PM+ が使うファイル
sample.prw	統合開発環境 PM+ が使うファイル
sample.cgp	Applilet3のプロジェクトファイル

※マイコンの種類によって、生成するソースは若干異なります。その違いはコードの違いではなく、スタートアップ処理の違いによるものです。
※ファイル先頭の「CG_」 コード生成(Code Generate)の略です。「CG_」とついたソースファイルがコード生成されたファイルを示しています。

【周辺機能を使った場合に生成されるファイル】

CG_timer.c / h	タイマを使った場合に生成される。タイマ初期化、タイマ開始、タイマ停止を含む
CG_timer_user.c	割り込みハンドラ関数が定義される。ユーザは関数内にハンドラ処理を追加する
CG_port.c / h	ポートを使った場合に生成される。ポート初期化を含む
CG_port_user.c	通常は空のファイル。ユーザがポート処理を追加したい場合に書く
CG_int.c / h	外部割込みを使った場合に生成される。外部割込みの有効/無効処理を含む
CG_int_user.c	割り込みハンドラ関数が定義される。ユーザは関数内にハンドラ処理を追加する
CG_周辺機能.c / h **CG_周辺機能_user.c**	上記は、タイマ、ポート、外部割込みを使った場合の例。周辺機能を使うと左記のようなファイルが生成される。CG_周辺機能.c では、初期化、開始、停止、その他周辺機能を使うためのAPIが用意される。CG_周辺機能_user.c では、割り込みハンドラ処理が行われる。通常、ユーザは割り込みハンドラ関数内に処理を追加すればよい。

Appliletが提供するAPI (Application Programming Interface)

提供するAPI、関数の一覧は下記ファイルを参照してください。
・Applilet3 78K0マイクロコントローラ APIリファレンス編
・Applilet3 78K0Rマイクロコントローラ APIリファレンス編
・Applilet3 V850マイクロコントローラ APIリファレンス編
ここでは、マイコンのポート、タイマ、外部割り込みの周辺機能を使う場合にプログラムでAPIをどう利用するのか解説します。ほんの一例ですが、API使い方の基本を学習してください。

マイコン周辺機能別の基本的な使い方

【ポート】

API名	
PORT_Init()	ポート初期化は、スタートアップルーチンからcallされるのでユーザは意識しない

【タイマ】(インターバル・タイマの場合)

タイマ周辺名_init()	タイマの初期化。スタートアップルーチンからcallされる。「タイマ周辺名」はマイコンによって変わる。
タイマ周辺名_start()	タイマ割り込みを開始する。
タイマ周辺名_stop()	タイマ割り込みを停止する。
タイマ周辺名_ChangeCondition()	デューディ比を変更する。

使い方の例

```
main()
{
    タイマ周辺名_start();          // タイマ割り込みを開始する。
    while (1) { ; }                // 設定したタイマ周期で下記のハンドラが呼ばれる。
}

_interrupt void MD_タイマ周辺名(void)          // 割り込みハンドラ
{
   // ここにユーザソースコードを記述する。
}
```

【外部割り込み】(INTPxを使う場合)

INTP_init()	外部割り込みの初期化。スタートアップルーチンからcallされる。
INTPx_Enable()	外部割り込みを許可する。
INTPx_Disable()	外部割り込みを禁止する。

使い方の例

```
main()
{
    INTPx_Enablet();               // 外部割り込みを許可する。
    while (1) { ; }                // 外部割り込みが発生したら、下記のハンドラが呼ばれる。
}

_interrupt void MD_INTPx(void)  // 割り込みハンドラ
{
   // ここにユーザソースコードを記述する。
}
```

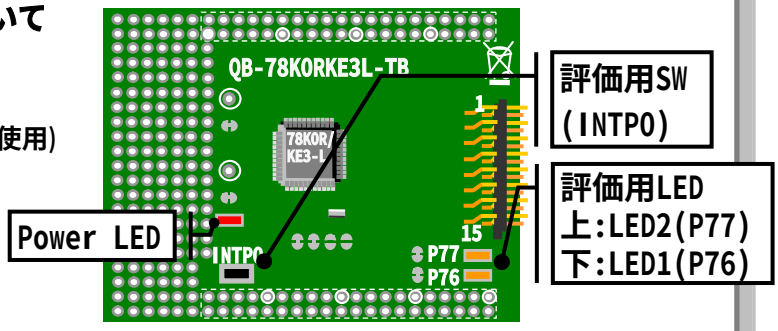
使用例：MINICUBE2＋CPUボードで実行

MINICUBE2、QB-78K0RKE3L-TBを用いて実際にプログラムを実行するまでを解説します。
ここではQB-78K0RKE3L-TBを使用していますが、他のCPUボードを使用しても基本操作に違いはありません。

サンプルの概要

QB-78K0RKE3L-TB (CPUボード) について

・78K0R/KE3-L(μPD78F1009GB)搭載
・メイン・クロック20MHz(発振子を搭載)
・オンチップ・デバッグに対応(TOOL0,TOOL1端子使用)
・評価用LED x 2、評価用SW を搭載
・MINICUBE2と専用ケーブルで接続
・MINICUBE2のSW設定 (SW1：M1、SW2：3)

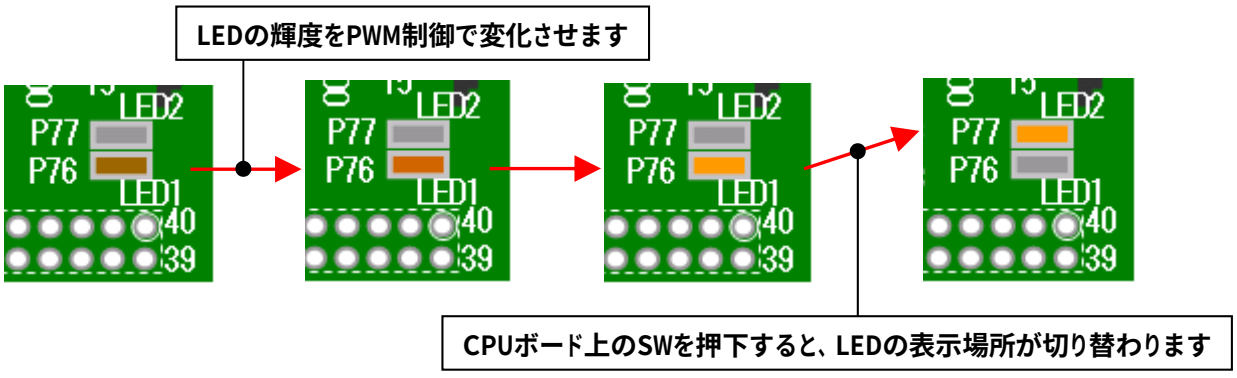


使用するマイコン周辺機能

・ポート(LED制御用)
・タイマ(PWM出力を行う)
・外部入力(ボード上のSW入力に使用)

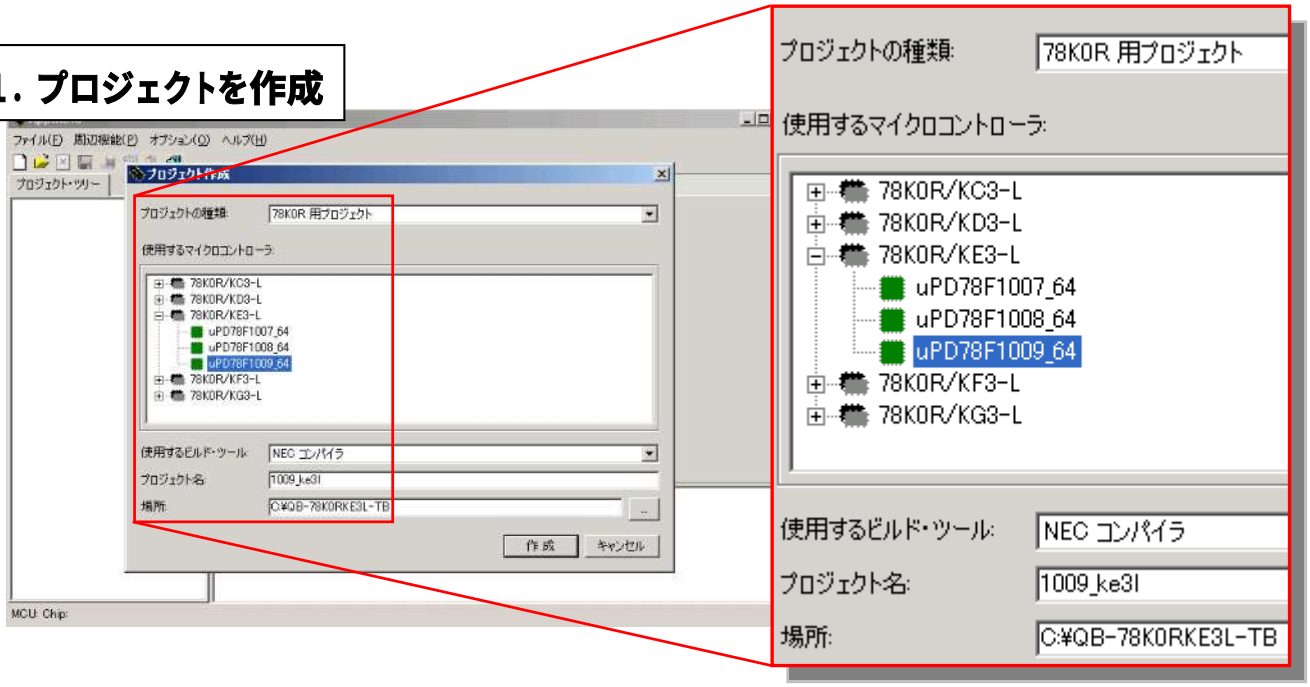
作成するプログラム

・CPUボード上のLEDをPWM(タイマ)を使って明るさを変える。また、CPUボード上のSWを押下すると点灯するLEDが替わる。
・LEDの輝度は100msecごとにデューティ比1%づつ変化する。デューティー比は1～99%まで変化する。

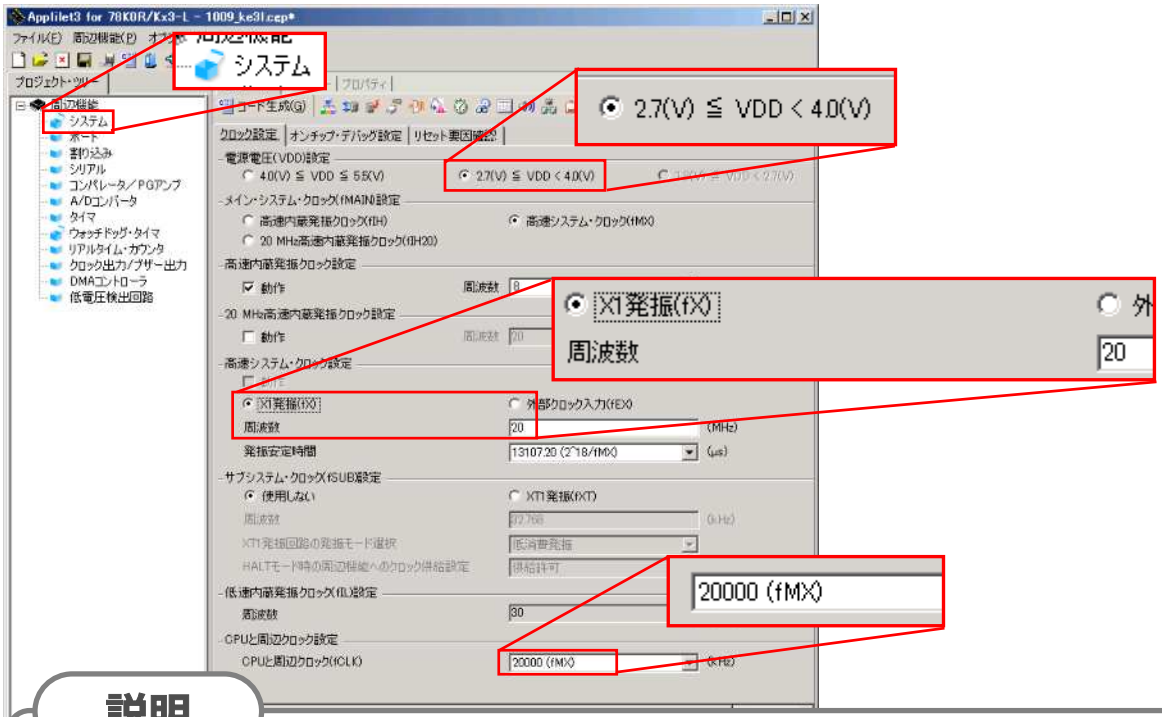


使用例：Appliletを起動する

1. プロジェクトを作成



2. システムの設定



説明

1. プロジェクトを作成

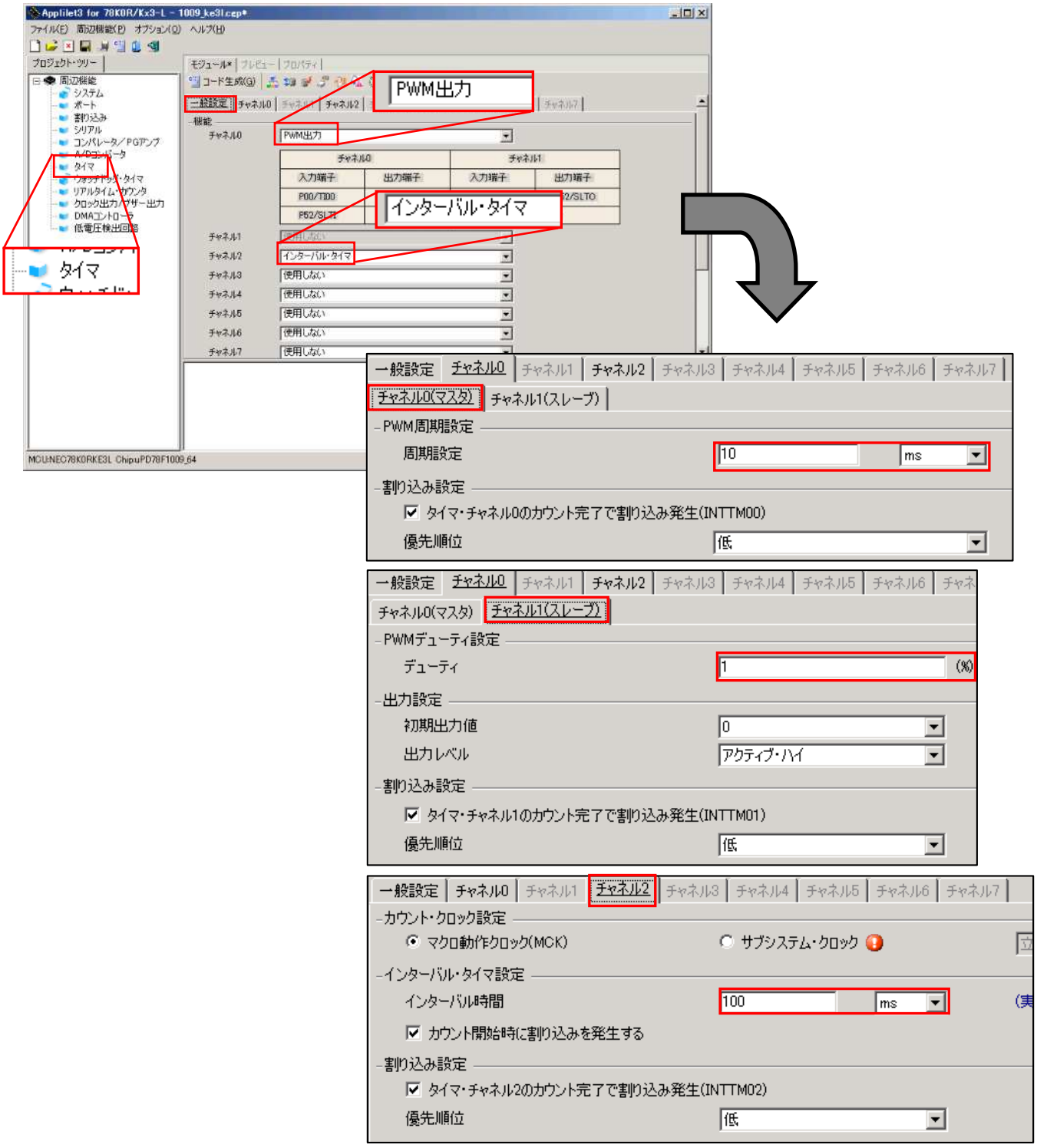
マイコンのシリーズ品種毎に存在します。サポートマイコンについては、「ZUD-CD-10-0179 Applilet3紹介資料」を参照してください。

2. システムの設定

システムとはマイコンの基本設定のこと。CPUクロック、オンチップデバッグの有無、動作電圧(動作電圧は設定できる周波数に影響する)を設定します。

使用例：Appliletで周辺機能を設定する

3．タイマの設定



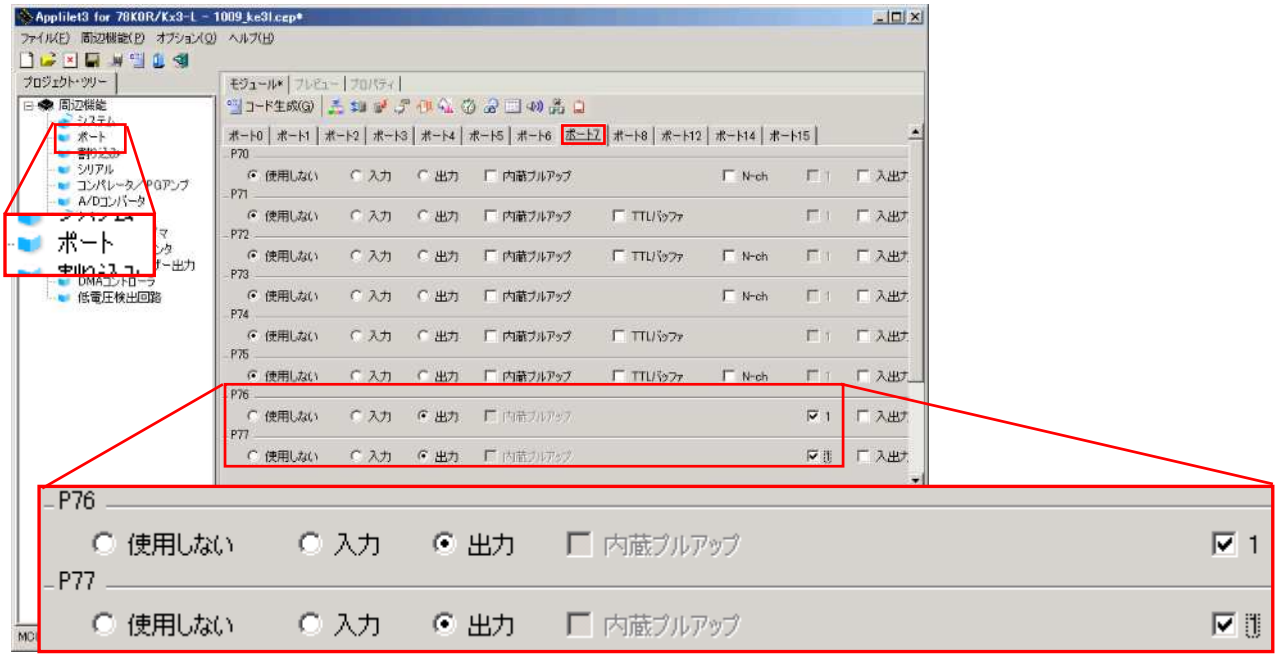
説明

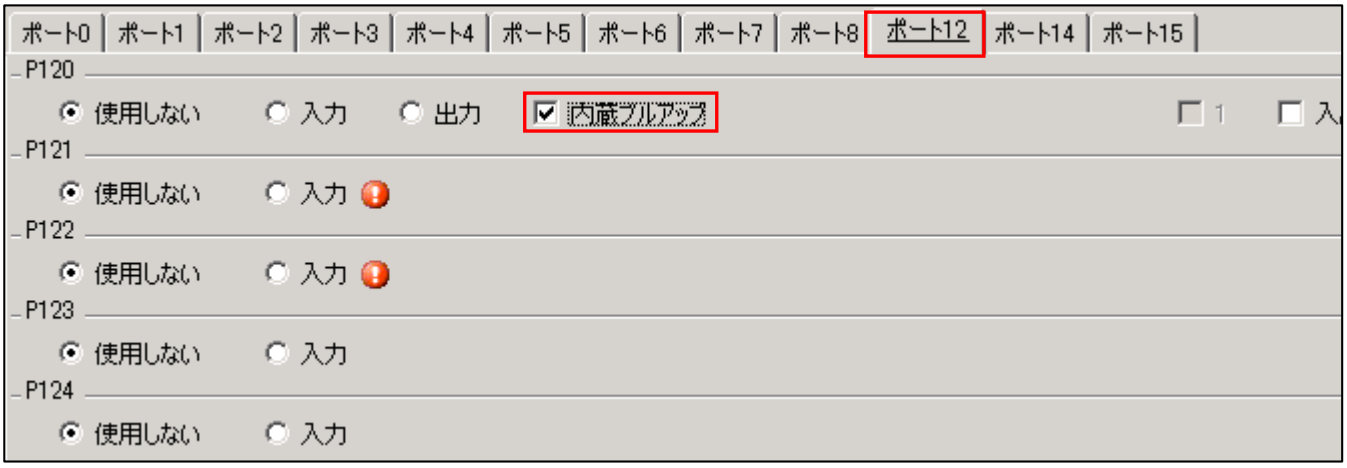
3．タイマの設定

チャネル0をPWM、10ms、デューティ 1%にします。また、チャネル2を100msのインターバル・タイマへ設定します。

使用例：Appliletで周辺機能を設定する

4．ポートの設定





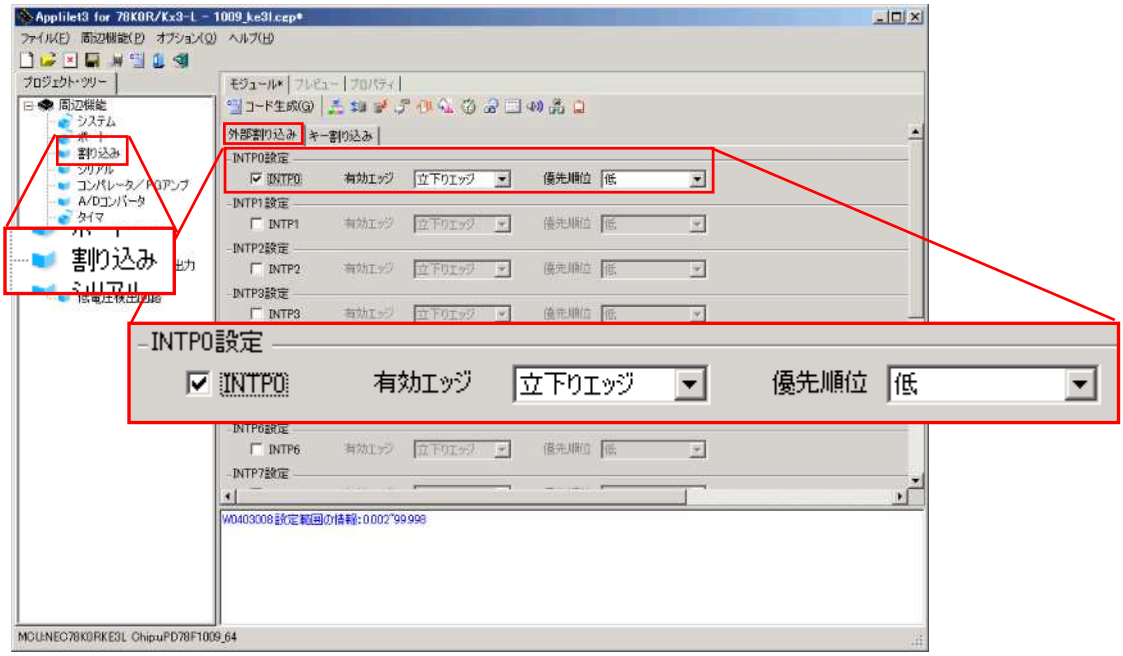
説明

4．ポートの設定

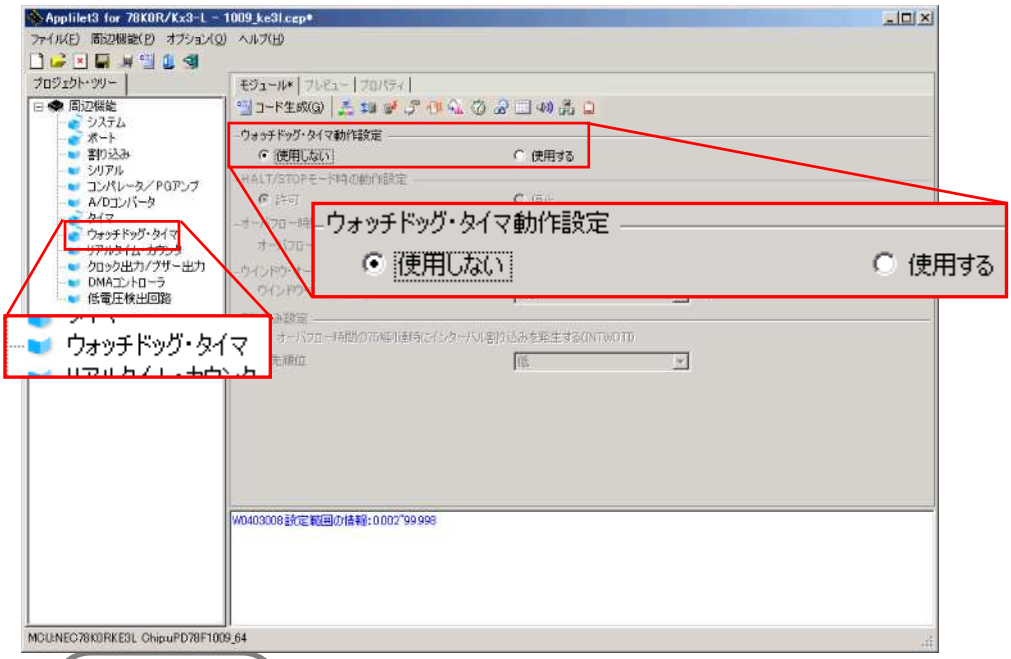
P76、P77を出力にします。またP120はINTP0で使用するので内蔵プルアップにチェックします。

使用例：Appliletで周辺機能を設定する

5．外部割り込みの設定



6．ウォッチドッグ・タイマの設定



説明

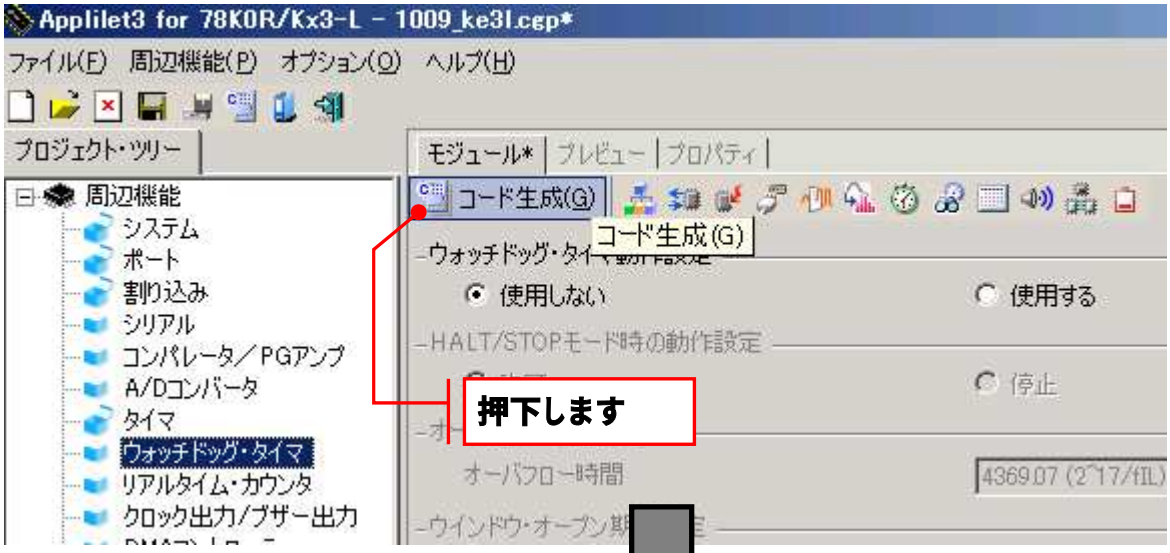
5．外部割り込みの設定

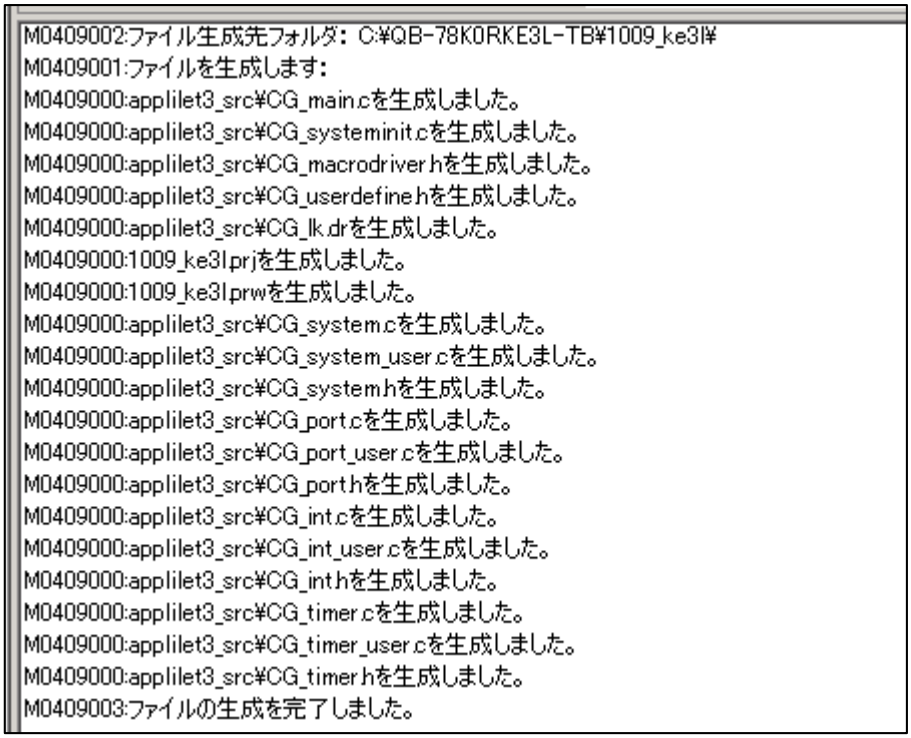
INTP0を有効エッジ 立ち下がりで設定します。

6．ウォッチドッグ・タイマの設定

ウォッチドッグ・タイマを使用しないに設定します。

使用例：プロジェクト（ソースコード）を生成する



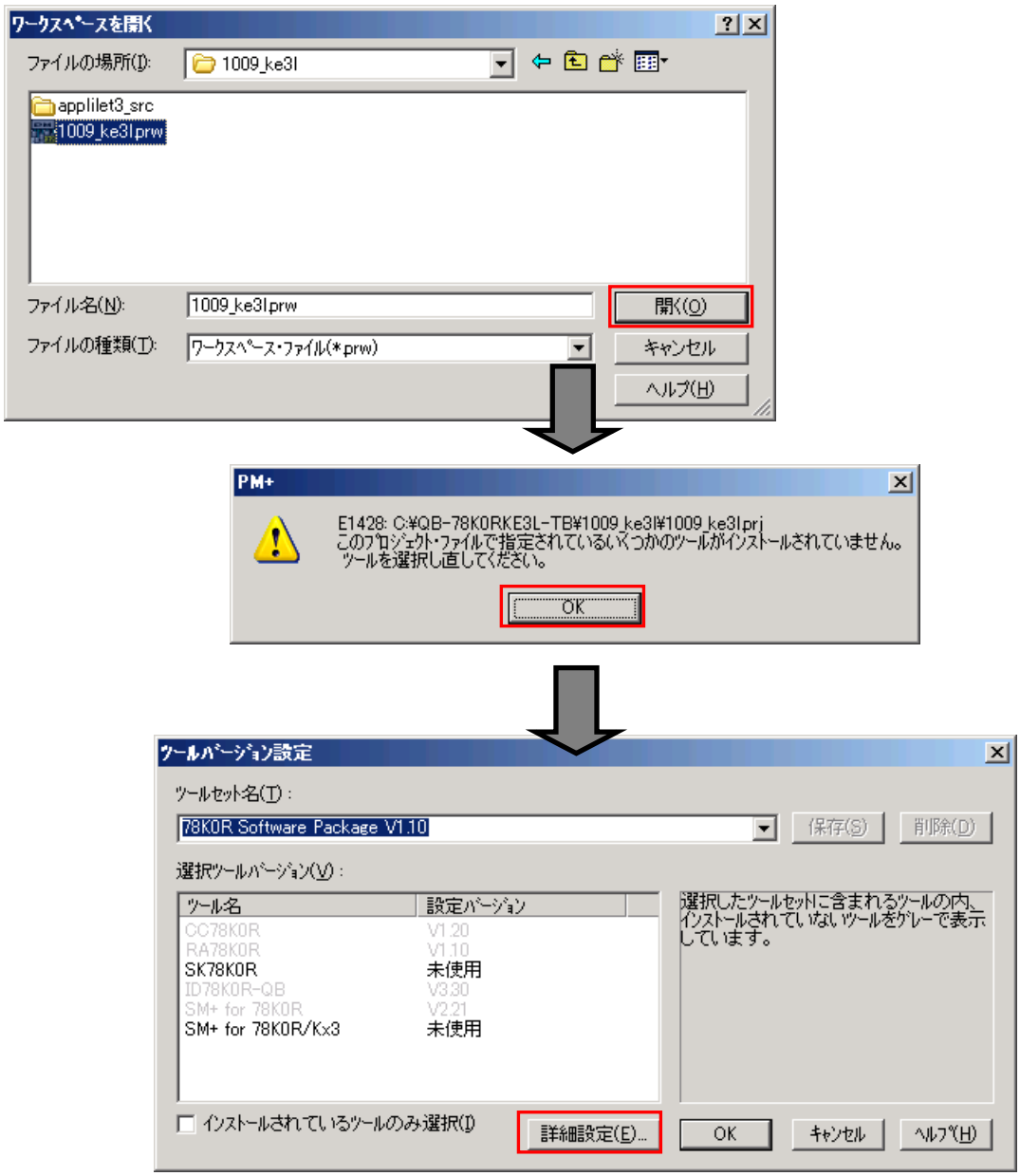


説明

コード生成を押下

ウインドウに生成したファイル一覧が表示されます。

使用例：PM+でプロジェクトを開く



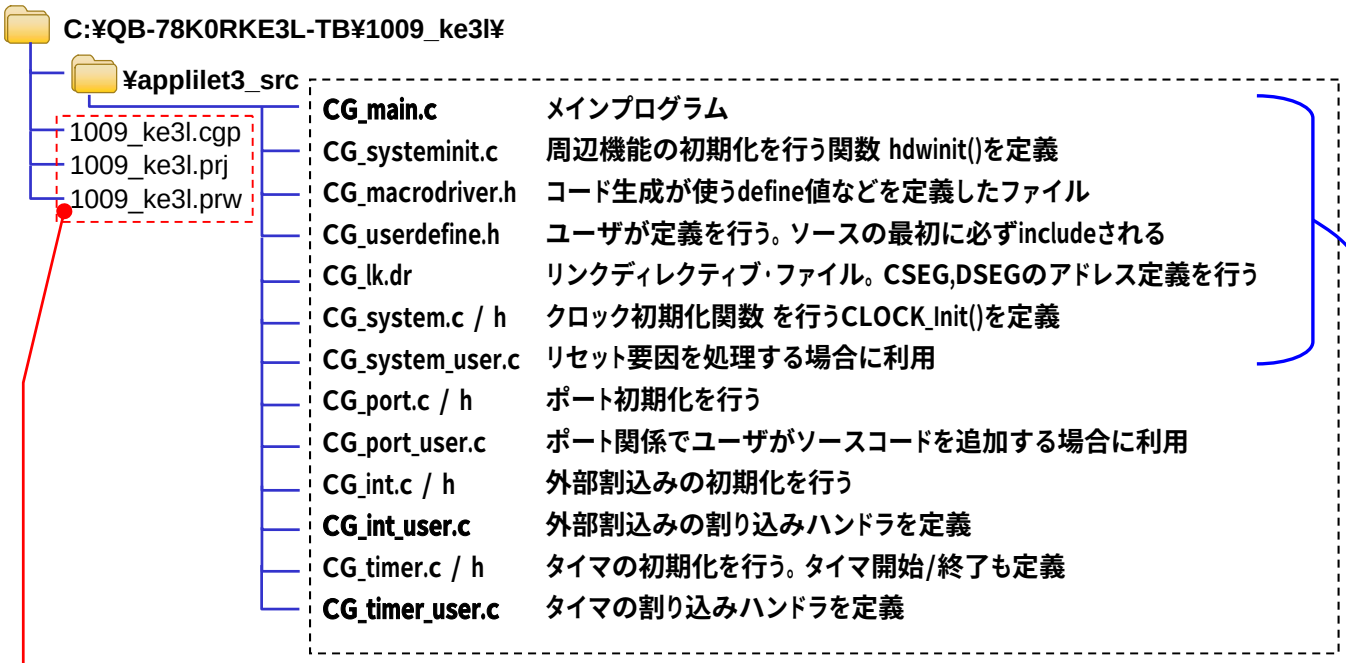
説明

PM+上でワークスペースを開く

上記のようにワーニングが表示された場合、ツールバージョンの詳細を設定しなおしてください。

使用例：生成されたソースファイルを編集する

Appliletが生成するファイルは、下記のルールに沿っています。また、ユーザが編集するファイルを決めることによって、コーディングルールを守りやすくなる利点があります。サンプルで設定した場合のファイル一覧は下記の通りになります。



1009_ke3l.cgp はApplilet3で使用されるファイル。その他は統合環境PM+で使われるプロジェクトファイル。PM+を起動後には、次のファイルも作成される
(1009_ke3l.sdb、1009_ke3l.fni、1009_ke3l.prk)

コード生成の作成ルール

必ず生成するのがカッコで示したファイル。マイコンに必要なクロック設定、各種定義などが含まれる。
その他のソースファイルは、マイコンの周辺機能を使うたびに生成される。
例えば、ポート機能を使うと CG_port.c , CG_port.h, CG_port_user.c の3種類が作成され、外部割り込みなら、CG_int.c, CG_int.h, CG_int_user.c が作成される。それらは下記のルールに従っている。

CG_周辺機能.c / CG_周辺機能.h → Appliletが生成するAPI関数を定義する。ユーザはAPI関数を利用するだけで基本的には編集しない。

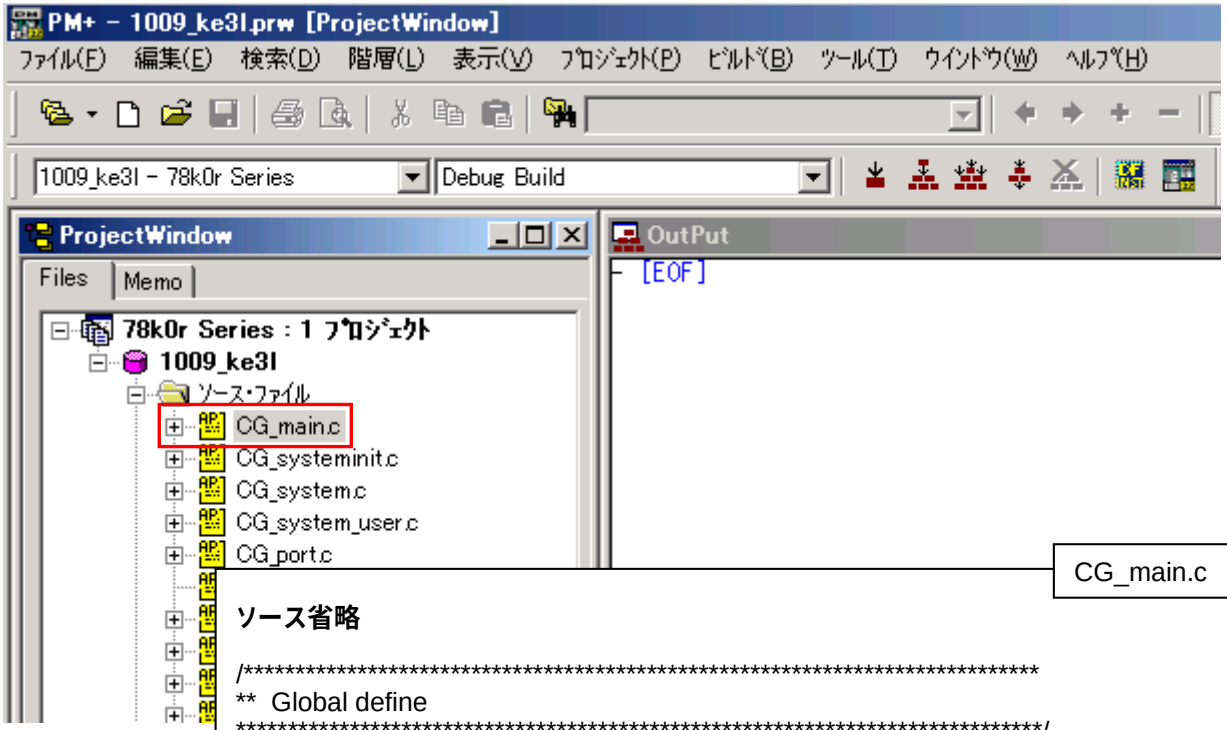
CG_周辺機能_user.c → 割り込みハンドラ関数を定義する。ユーザは割り込みハンドラのみを編集すればよい。但し、CG_port_user.cはハンドラ処理がないので、特に編集する必要がない。

説明

PM+上で編集するファイル

CG_main.c、CG_int_user.c、CG_timer_user.c が編集対象のファイルです。CG_main.cはメインプログラム、CG_int_user.cは外部割り込みハンドラ、CG_timer_user.cはタイマハンドラが発生したときの処理を書きます。

使用例：生成されたソースファイルを編集する



CG_main.c

ソース省略

```
/***********************************************************************************
**  Global define
***********************************************************************************/
/* Start user code for global. Do not edit comment generated here */

UINT gCh2Counter;
UCHAR gLedChange;

/* End user code. Do not edit comment generated here */

/*-----------------------------------------------------------------------------
**  Abstract:       This function implements main function.
**  Parameters:  None
**  Returns:        None
**-----------------------------------------------------------------------------*/
void main(void)
{
    /* Start user code. Do not edit comment generated here */
    gCh2Counter = 0;
    gLedChange = 0;

    INTP0_Enable();
    TAU0_Channel0_Start();
    TAU0_Channel2_Start();

    while (1U)
    {
                    ;
    }
    /* End user code. Do not edit comment generated here */
}
```

gCh2Counter : LEDのデューディー比をあらわす
gLedChange : LEDの点灯位置をあらわす

外部割り込みを有効にする。
チャネル0のPWMタイマ（10msec）とチャネル2のインターバルタイマを開始する。

説明

PM+上でソースファイルを編集する

CG_main.cを編集してください。**赤字**がソースを追加した部分です。

使用例：生成されたソースファイルを編集する

CG_int_user.c

ソース省略

```
/***********************************************************************************
**  Global define
***********************************************************************************/
/* Start user code for global. Do not edit comment generated here */

extern UCHAR gLedChange;

/* End user code. Do not edit comment generated here */

/*
**-----------------------------------------------------------------------------
**  Abstract:      This function is INTP0 interrupt service routine.
**  Parameters:  None
**  Returns:       None
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTP0(void)
{
    /* Start user code. Do not edit comment generated here */

    gLedChange ^=1;
    P7.6 = 1;
    P7.7 = 1;

    /* End user code. Do not edit c
}
```

LEDの点灯位置をSW押下ごとに変更する。

LEDの点灯位置を変更した場合に、チラツキが生じないように一度LEDを消灯する。

説明

PM+上でソースファイルを編集する

CG_int_user.cを編集してください。**赤字**がソースを追加した部分です。

使用例：生成されたソースファイルを編集する

CG_timer_user.c

```
ソース省略
/***********************************************************************************
**  Global define
***********************************************************************************/
/* Start user code for global. Do not edit comment generated here */
extern UINT gCh2Counter;
extern UCHAR gLedChange;
/* End user code. Do not edit comment generated here */

/*---------------------------------------------------------------------------
**  Abstract:      This function is INTTM00 interrupt service routine.
**  Parameters:  None
**  Returns:       None
**---------------------------------------------------------------------------*/
__interrupt void MD_INTTM00(void)
{
    /* Start user code. Do not edit comment generated here */
    if ( gLedChange == 0 )
        P7.6 = 0;
    else
        P7.7 = 0;
    /* End user code. Do not edit comment generated here */
}
/*---------------------------------------------------------------------------
**  Abstract:      This function is INTTM01 interrupt service routine.
**  Parameters:  None
**  Returns:       None
**---------------------------------------------------------------------------*/
__interrupt void MD_INTTM01(void)
{
    /* Start user code. Do not edit comment generated here */
    if ( gLedChange == 0 )
        P7.6 = 1;
    else
        P7.7 = 1;
    /* End user code. Do not edit comment generated here */
}
/*---------------------------------------------------------------------------
**  Abstract:      This function is INTTM02 interrupt service routine.
**  Parameters:  None
**  Returns:       None
**---------------------------------------------------------------------------*/
__interrupt void MD_INTTM02(void)
{
    /* Start user code. Do not edit comment generated here */
    UCHAR ratio;

    gCh2Counter++;
    gCh2Counter %= 100;
    ratio = (UCHAR)gCh2Counter;

    TAU0_Channel1_ChangeDuty( ratio );
    /* End user code. Do not edit comment gener
}
```

10msecのPWM割り込みハンドラ。LED点灯を処理する。

LEDの点灯位置を判断して、該当するLEDを点灯する。

10msecのPWM割り込みハンドラ。LED消灯を処理する。

LEDの消灯位置を判断して、該当するLEDを消灯する。

100msecのインターバル割り込みハンドラ。LED輝度を処理する。

ratioに計算したデューディー比を代入する 。

10msecPWMのデューディー比を変更する。

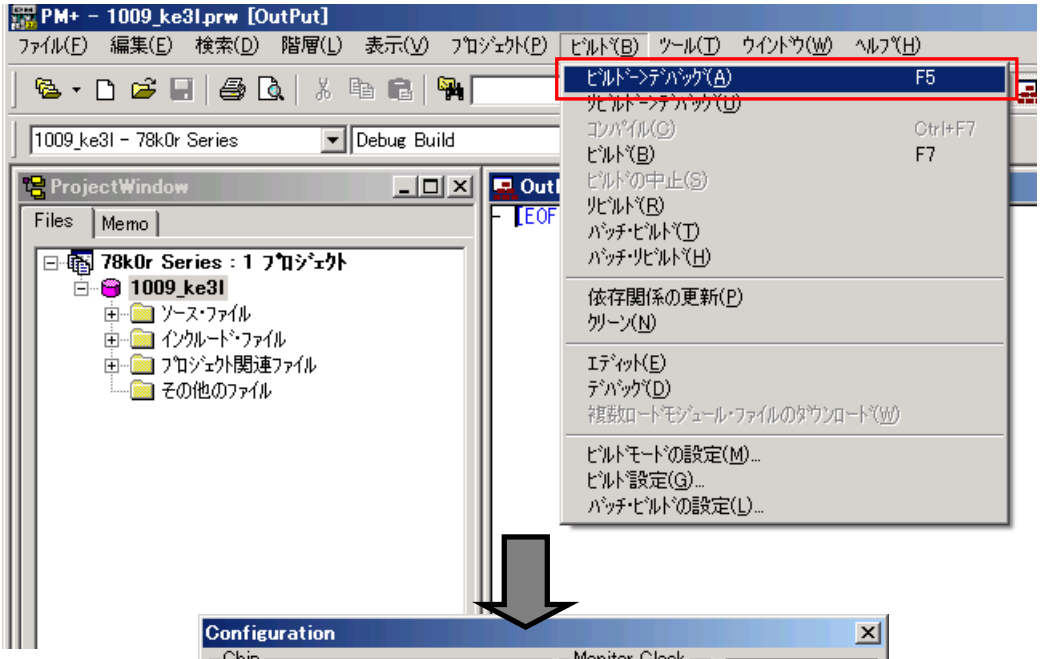
説明

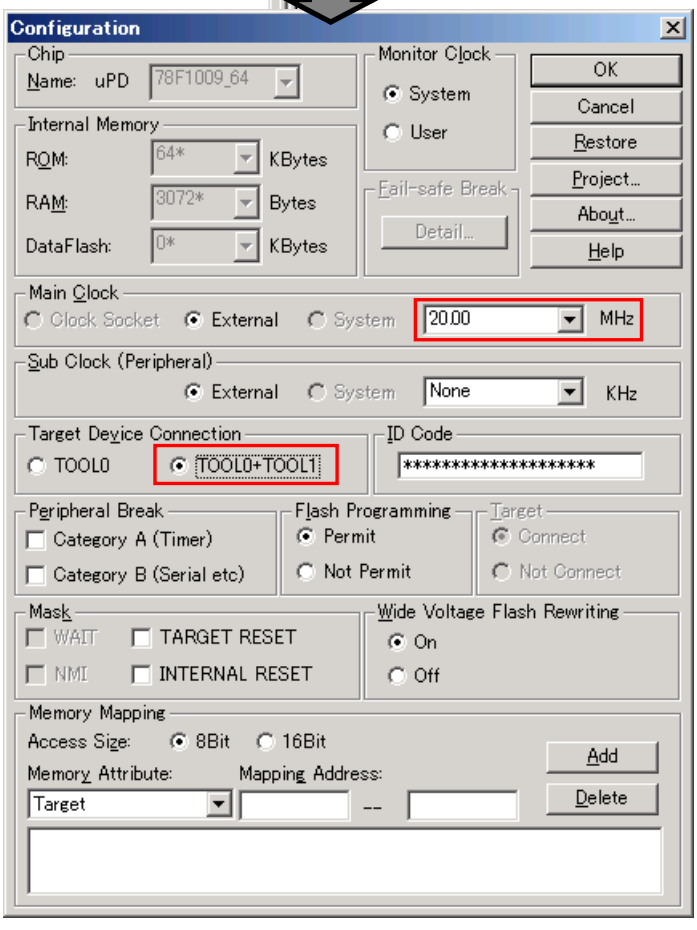
PM+上でソースファイルを編集する

CG_timer_user.cを編集してください。**赤字**がソースを追加した部分です。ソース編集はこれで、終わりです。

使用例：ビルドを行い、実行する

PM+よりビルド->デバッグを選択します。すると、ソースからオブジェクトが生成され、自動的にターゲットのマイコンへダウンロードされます。デバッガの画面では設定に注意してください。ダウンロード後にプログラムを実行させます。





最後に

Appliletの便利さがご理解頂けたでしょうか？ 初心者のためのツールではありません。
開発を短縮するためのツールなのです。Appliletはこれからも進化します。
情報も充実させていく予定です。
http://www2.renesas.com/micro/ja/development/asia/applilet3/
ご期待ください。

CubeSuiteのコード生成とApplilet3の関係について

新統合開発環境のCubeSuiteにも自動的にソースを作成する機能があります。それなのにApplilet3が必要なのですか？
このような疑問もあるかと思います。Applilet3には次の狙いがあります
・既存（PM+）ユーザのサポート
・IAR、Multiユーザのサポート
・フリーツールとして提供
全てのお客様の利便性を考え、Applilet3を提供します。