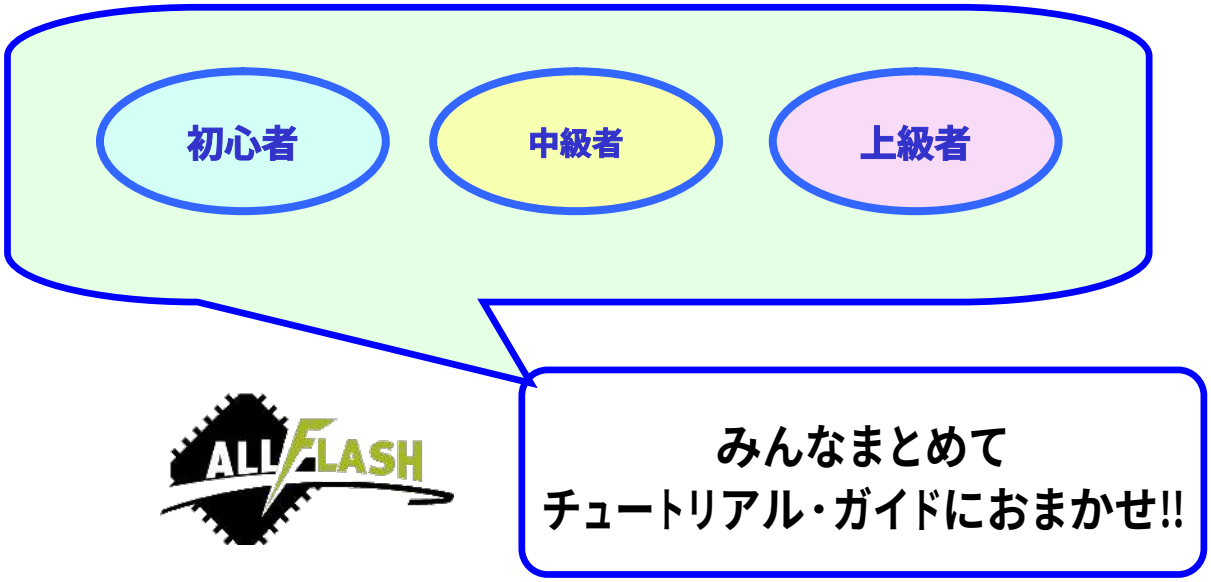
お客様各位

ZUD-CD-07-0012 1/110
2007年2月16日
NECエレクトロニクス株式会社
第四システム事業本部
汎用マイコンシステム事業部
開発ツールグループ
チームマネージャー 安藤 喜成
(担当：鈴木 康之)
CP(K), O

78K0R/KG3ターゲット・ボード
QB-78K0RKG3-TB
チュートリアル・ガイド

ごあいさつ

QB-78K0RKG3-TBをお買い求めいただき、誠にありがとうございます。
　本製品(QB-78K0RKG3-TB)は、NECエレクトロニクス社製のプログラミング機能付きオンチップ･デバッグ･エミュレータQB-MINI2(以降MINICUBE2)を使用して、マイコンを実際に試すためのターゲット･ボードです。
　チュートリアル･ガイドでは、本製品とMINICUBE2を使用した開発環境をわかりやすく説明します。サンプル･プログラム、ターゲット･システム回路例を用いて説明していますので、熟読して頂ければマイコン開発の基本が習得できます。本製品と共に活用して下さい。



　本製品に関する最新情報、必要な無償版開発ツール、およびサンプル･プログラムは、弊社webサイトにて提供しています。

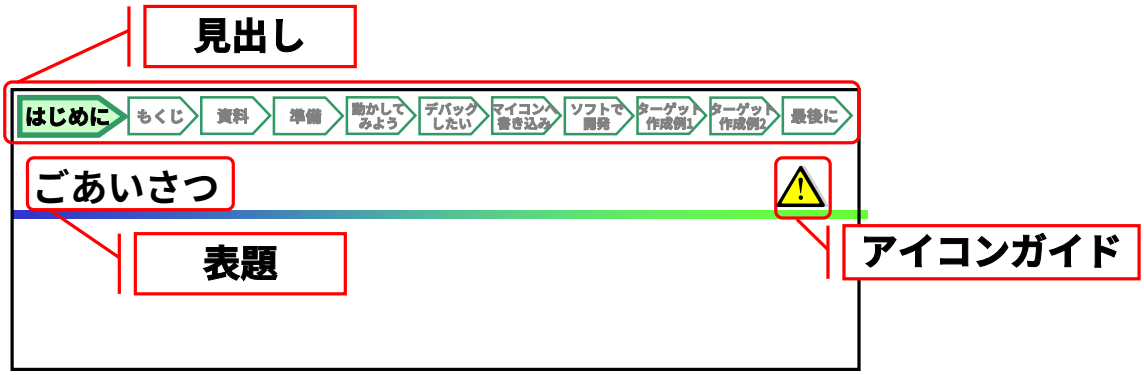
[MINICUBE2関連ツール]
http://www.necel.com/micro/ja/development/asia/minicube2/minicube2.html

[無償版開発ツール]
http://www.necel.com/micro/freesoft/

[サンプル･プログラム]
http://www.necel.com/micro/ja/development/asia/minicube2/minicube2_opt.html

本書の見方(説明)

本書では、わかりやすくするため下記の構成になっています。



[見出し]
本書の章に相当します。太い緑の線で囲まれたところは、現在の章を表します。また、薄い緑の線で囲まれたところは、他の章を表していますので本書全体の構成が、どのページでも一目でわかるようになっています。

[表題]
章の中で伝えたいことをいくつかの項目に分けています。本書中では「xxページを参照して下さい」という表現はしません。例えば「[はじめに]の[本書の見方]を参照して下さい」という表現になります。

[アイコンガイド]
アイコンガイドで、そのページ内容を一目で分かるようにアイコンで表現しています。後で読めばいいのか、必ず読む必要があるのか、このアイコンガイドを参考にして下さい。アイコンガイドには、下記の種類があります。

各章の学習する時間の目安を示します。単位は分です。


注意する内容です。

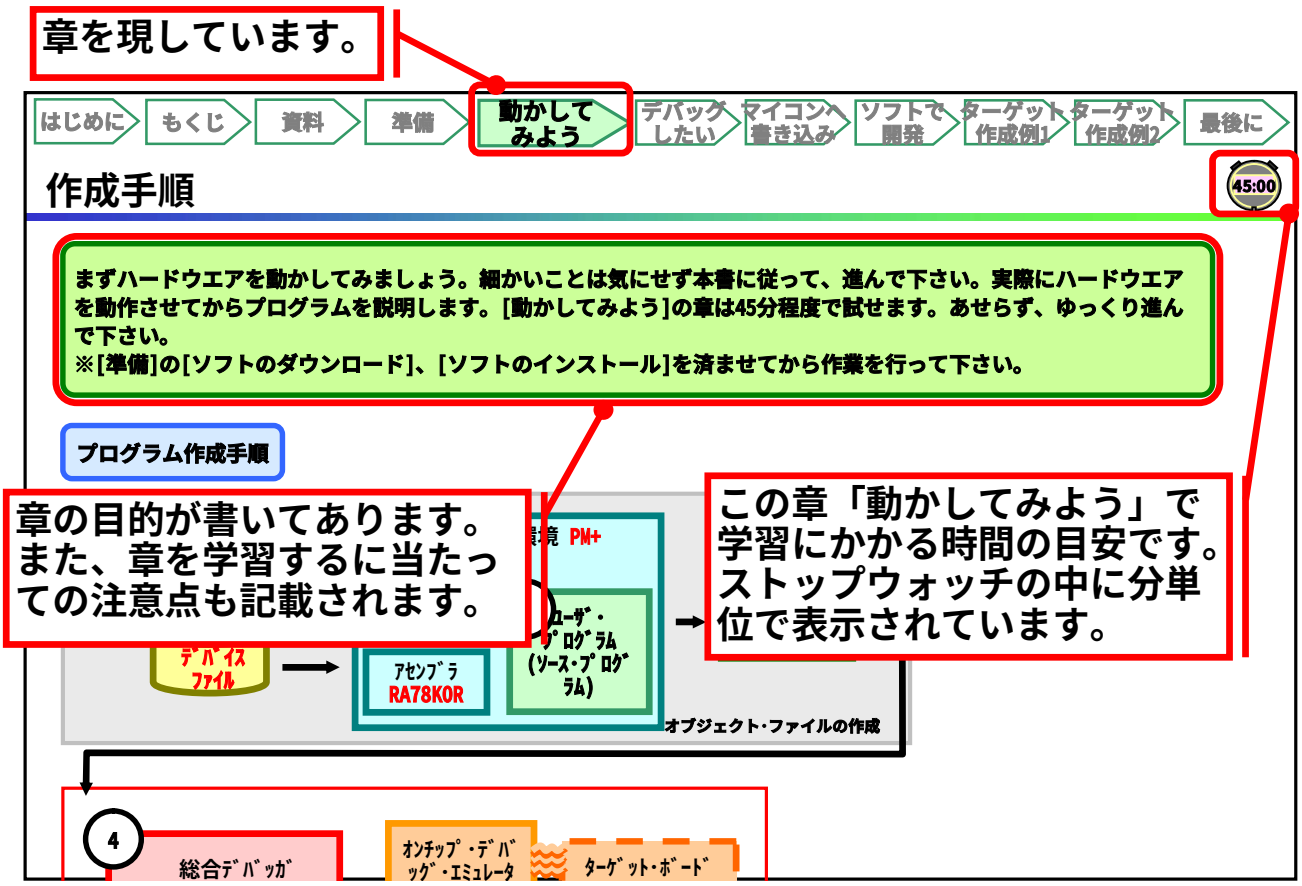
必ず読む必要のある内容です。


資料として使う内容です。


コラム･ワンポイントなど、関連する事についての内容です。

本書の見方(例)

前ページ[本書の見方(説明)]に沿った例を示します。



目次の見方について説明します。

対象とする読者によって色分けしてあります。

- 初心者向けゾーンでは、まず本製品QB-78K0RKG3-TBを動作させることを目的としています。また、後半ではサンプル･プログラムの内容を説明しています。
- 中級者向けゾーンでは、開発の基本を理解することを目的としています。デバッグ方法の基本と、高度なデバッグについて説明しています。
- 上級者向けゾーンでは、応用例としてドットマトリクスLEDを使った電光掲示板を作成し、更なる開発の手がかりを示しています。

ワンポイント

ワンポイントについて
そのページに関係する追加情報を載せています。また、深い内容についてはヒントになる語句について記しています。このワンポイントにも注目して下さい。

目次

目次

QB-78K0RKG3-TBの特徴

78K0R/KG3ターゲット・ボード(QB-78K0RKG3-TB)の特徴

●78K0R/KG3(μPD78F1166GF)搭載
●メイン・クロック20MHz(発振子を搭載)で高速動作可能(2.7V～5.5V供給時)
●フラッシュメモリ:256KB、RAM:12KBを内蔵
●最大で88本のI/Oポートを装備(N-chオープン・ドレイン4本)
●プログラミング、オンチップ・デバッグに両対応(TOOL0,TOOL1端子使用)
●LED2個、SW1個を搭載しており簡単なテストが可能
●ユニバーサル・エリア(2.54mmピッチ)を搭載
●マイコンの端子を周辺ボード・コネクタに配置した高拡張性
●鉛(Pb)フリー対応品

78K0R/KG3ターゲット・ボード(QB-78K0RKG3-TB)のハードウエア仕様

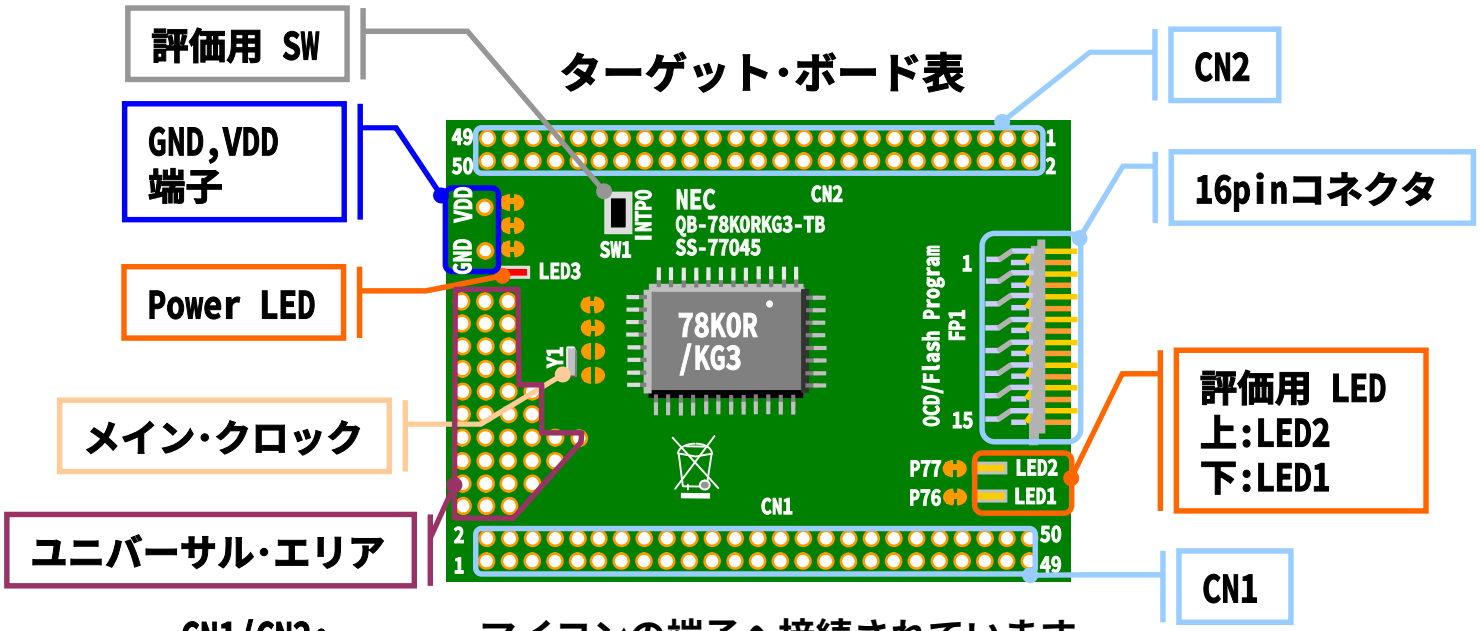
CPU μPD78F1166GF	メイン・クロック 動作周波数	20MHz(ボード上に搭載)
搭載部品	CN1,CN2:周辺ボードコネクタ(2.54mmピッチ) 50pinソケットx2(パットのみ)	
	FP1:16pinコネクタ(MINICUBE2接続用)	
	PowerLED:LED赤x1(LED3)	
	評価用LED:LED黄x2(LED1はP76,LED2はP77へ接続)	
	評価用SW:SW1(INTP0へ接続)	
	Y1:20MHz発振子(X1,X2へ接続)	
動作電圧	2.7V～5.5V(Y1:20MHz発振子使用時)	

ワンポイント

マイコンについて

ここで使うマイコンとはマイクロコントローラ(マイクロコンピュータ)の意味です。現在のマイコンはROM,RAM,I/Oだけでなく A/D,D/A,UART,I2C,LIN,CAN,LCD制御,USB,DMAなど様々な機能をもったマイコンもあります。また、性能も数MIPS～数百MIPSまで揃っています。

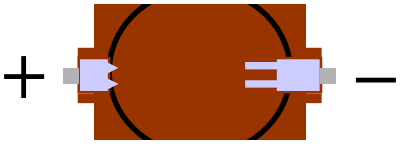
部品配置図

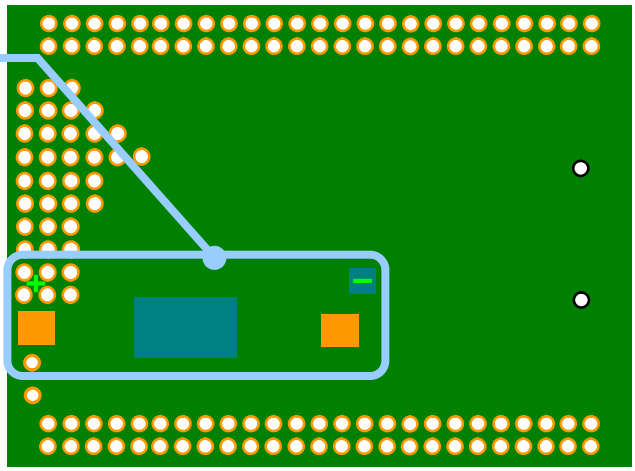


CN1/CN2: マイコンの端子へ接続されています
LED3(PowerLED)：電源が入った時に赤色に発光します
評価用LED1: ポート76(P76)がLOWで黄色に発光します
評価用LED2: ポート77(P77)がLOWで黄色に発光します
評価用SW1： INTP0に接続されています
FP1(16pinｺﾈｸﾀ)：オンチップ･デバッグや書き込み時に使用します
MINICUBE2(別売)を接続します
Y1(ﾒｲﾝｸﾛｯｸ): 20MHz発振子を搭載しています
ﾕﾆﾊﾞｰｻﾙ･ｴﾘｱ： ユーザーが部品を載せられるエリアです
GND,VDD端子: ターゲット･ボードへ電源供給する場合の端子です

ターゲット･ボード裏

ボタン電池ホルダパッド
ここにCR2032などの電池ホルダを載せます。





・基板上のパターン  について
パターンをカットすることで、その回路はオープンとなります。 
再度ショートさせたい場合は半田ショートさせて下さい。 
P76,P77を使用する場合はLEDの左隣のショートパッドをパターンカットして下さい。

コネクタ情報(CN1)

ピン番号	接続先CPU端子	備考	ピン番号	接続先CPU端子	備考
1	P60/SCL0		2	P61/SDA0	
3	P62		4	P63	
5	P31/TI03/TO03/INTP4		6	P64/$\overline{RD}$	
7	P65/$\overline{WR0}$		8	P66/$\overline{WR1}$	
9	P67/ASTB		10	P77/EX23/KR7/INTP11	LED2へ接続
11	P76/EX22/KR6/INTP10	LED1へ接続	12	P75/EX21/KR5/INTP9	
13	P74/EX20/KR4/INTP8		14	P73/EX19/KR3	
15	P72/EX18/KR2		16	P71/EX17/KR1	
17	P70/EX16/KR0		18	P06/$\overline{WAIT}$	
19	P05/CLKOUT		20	GND	
21	P80/EX0		22	P81/EX1	
23	P82/EX2		24	P83/EX3	
25	P84/EX4		26	P85/EX5	
27	P86/EX6		28	P87/EX7	
29	P30/INTP3/RTC1HZ		30	EVDD	
31	P50/EX8		32	P51/EX9	
33	P52/EX10		34	P53/EX11	
35	P54/EX12		36	P55/EX13	
37	P56/EX14		38	P57/EX15	
39	P17/EX31/TI02/TO02		40	P16/EX30/TI01/TO01/INTP5	
41	P15/EX29/RTCDIV/RTCCL		42	P14/EX28/RXD3	
43	P13/EX27/TXD3		44	P12/EX26/SO00/TXD0	
45	P11/EX25/SI00/RXD0		46	P10/EX24/$\overline{SCK00}$	
47	AVREF1		48	P110/AN00	
49	P111/AN01		50	AVREF0	

(※詳細は付録の回路図を参照して下さい)

コネクタ情報(CN2)

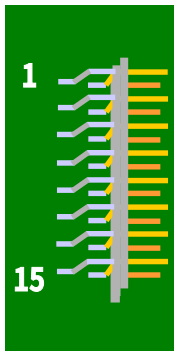
ピン番号	接続先CPU端子	備考	ピン番号	接続先CPU端子	備考
1	GND		2	P157/ANI15	
3	P156/ANI14		4	P155/ANI13	
5	P154/ANI12		6	P153/ANI11	
7	P152/ANI10		8	P151/ANI9	
9	P150/ANI8		10	P27/ANI7	
11	P26/ANI6		12	P25/ANI5	
13	P24/ANI4		14	P23/ANI3	
15	P22/ANI2		16	P21/ANI1	
17	P20/ANI0		18	P130	
19	P131/TI06/TO06		20	P04/$\overline{\text{SCK10}}$/SCL1	
21	P03/SI10/RXD1/SDA1		22	P02/SO10/TXD1	
23	P01/TO00		24	P00/TI00	
25	P145/TI07/TO07		26	P144/SO20/TXD2	
27	P143/SI20/RXD2/SDA2		28	P142/$\overline{\text{SCK20}}$/SCL2	
29	P141/PCLBUZ1/INTP7		30	P140/PCLBUZ0/INTP6	
31	P120/INTP0/EXLVI	SW1にも接続	32	P47/INTP2	
33	P46/INTP1/TI05/TO05		34	P45/SO01	
35	P44/SI01		36	P43/$\overline{\text{SCK01}}$	
37	P42/TI04/TO04		38	P41/TOOL1	16pinｺﾈｸﾀ16へ接続
39	P40/TOOL0	16pinｺﾈｸﾀ3,5へ接続	40	T_RESET	16pinｺﾈｸﾀ15へ接続
41	P124/XT2	10KΩﾌﾟﾙﾀﾞｳﾝ	42	P123/XT1	10KΩﾌﾟﾙﾀﾞｳﾝ
43	FLMD0		44	P122/X2/EXCLK	ﾃﾞﾌｫﾙﾄはﾏｲｺﾝ未接続
45	P121/X1	ﾃﾞﾌｫﾙﾄはﾏｲｺﾝ未接続	46	NC	
47	GND		48	GND	
49	VDD		50	EVDD	

(※詳細は付録の回路図を参照して下さい)

コネクタ情報(16pinヘッダ)

16pinヘッダピンアサイン

ピン番号	信号	ピン番号	信号
1	GND	2	RESET_OUT
3	TOOL0	4	VDD
5	TOOL0	6	R.F.U.
7	R.F.U.	8	R.F.U.
9	R.F.U.	10	R.F.U.
11	-----	12	R.F.U.
13	R.F.U.	14	FLMD0
15	RESET_IN注	16	TOOL1

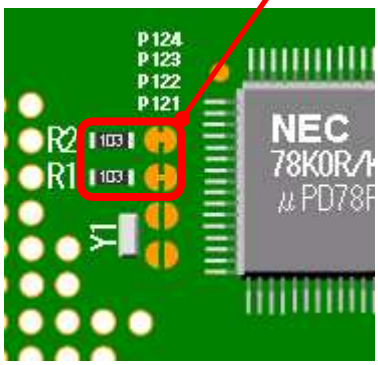


注:プログラミングのみ行う場合はR.F.U.です。

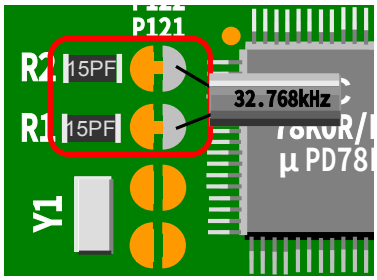
※R.F.U.は予約端子のためターゲット･ボード側でオープンになっています。

サブクロックの取り付け

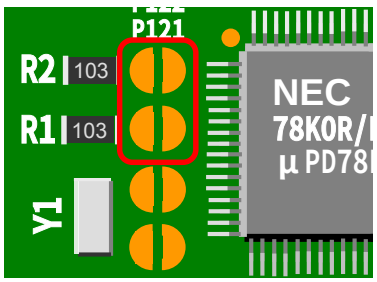
QB-78K0RKG3-TBには、サブクロックが搭載されていません。サブクロックを搭載する方法を説明します。

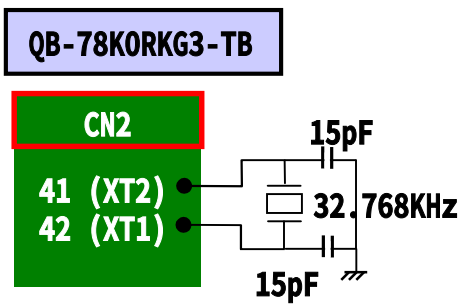


R1とR2を取り外します。替わりにコンデンサ(12～15PF)をつけます。そして、下図のようにサブシステム･クロックを接続して下さい。



実装が難しい場合は、  をパターンカットします。パターンカットすると、R1とR2のプルダウン抵抗が無効になります。CN2の41(XT2)、42(XT1)へサブシステム･クロック(32.768kHz)を接続して下さい。その際、コンデンサも取り付けてください。

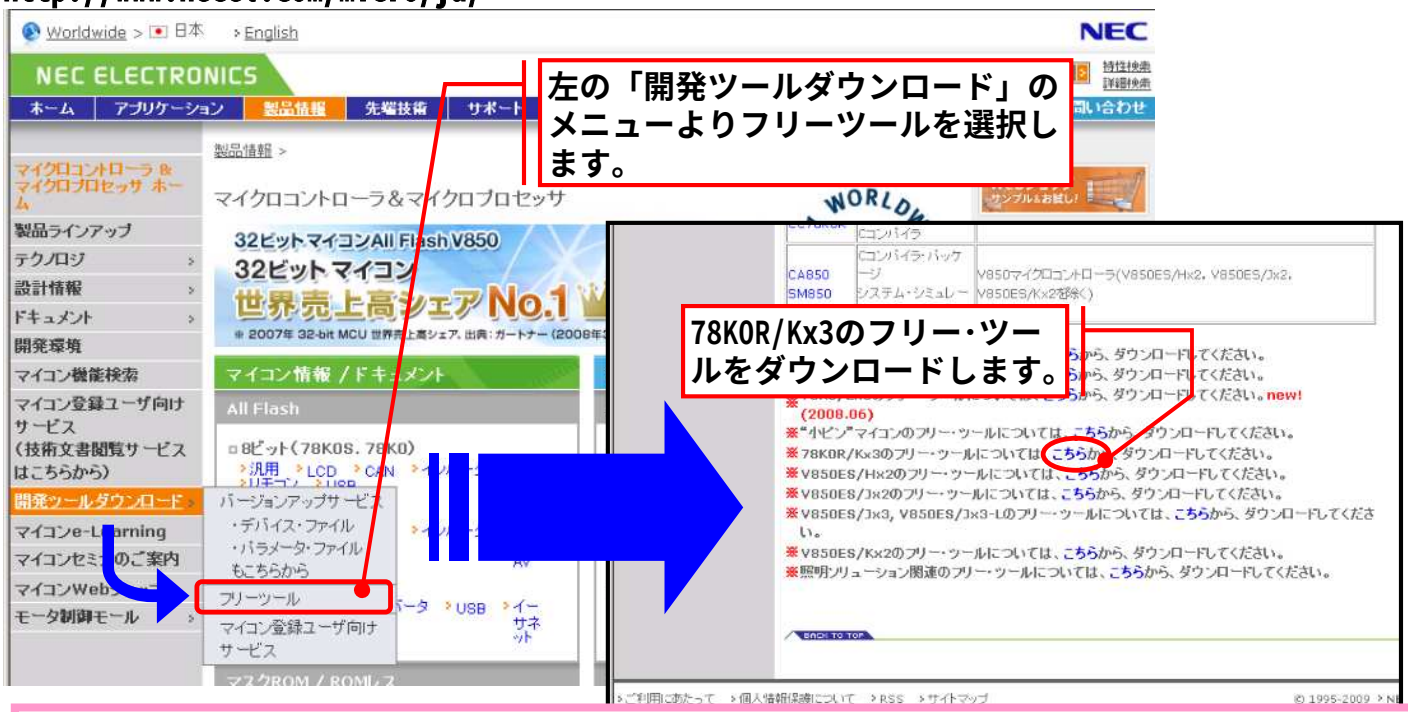




ソフトウエアのダウンロード

本書では、下記のソフトウエアをインストールする必要があります。
・[MINICUBE2関連ツール]
・[フリーツール]
[MINICUBE2関連ツール]MINICUBE2添付のセットアップ･マニュアルに従ってインストールして下さい。

ソフトウエアをダウンロードします。下記URLへアクセスして下さい。
http://www.necel.com/micro/ja/



フリーツールをダウンロードする時にユーザー登録します(メールアドレスの登録)。登録したメールアドレスにProduct IDが送られます。このProduct IDは開発ツールソフトウエアのインストールに必要ですので、忘れないようにメモして下さい。

ダウンロードするソフトウエア一覧

- **RA78K0R**
  統合開発環境PM+と78K0R用アセンブラを含めたパッケージです。
- **CC78K0R**
  78K0R用Cコンパイラです。
- **SM+ for 78K0R/Kx3（準備中の場合があります）**
  78K0R用のソフトウエア･シミュレータ(SM+)です。ハードウエアを必要としない、パソコン上で78K0Rマイコンのデバッグ/テストができます。
- **Applilet2 for 78K0R/Kx3**
  マイコンの初期プログラムを自動生成するツールです。これで作成したプロジェクトは統合開発環境PM+へ読み込み可能です。
- **78K0R/KG3用デバイス･ファイル(DF781188)**
  マイコンの品種ごと、または同系列品種のグループごとに用意された、機種依存情報を持つバイナリ･ファイルです。コンパイラやアセンブラで使います。
- **78K0R/KG3用パラメータ･ファイル(PRM78F1188)**
  マイコンへプログラミングを行うときに必要なファイルです。

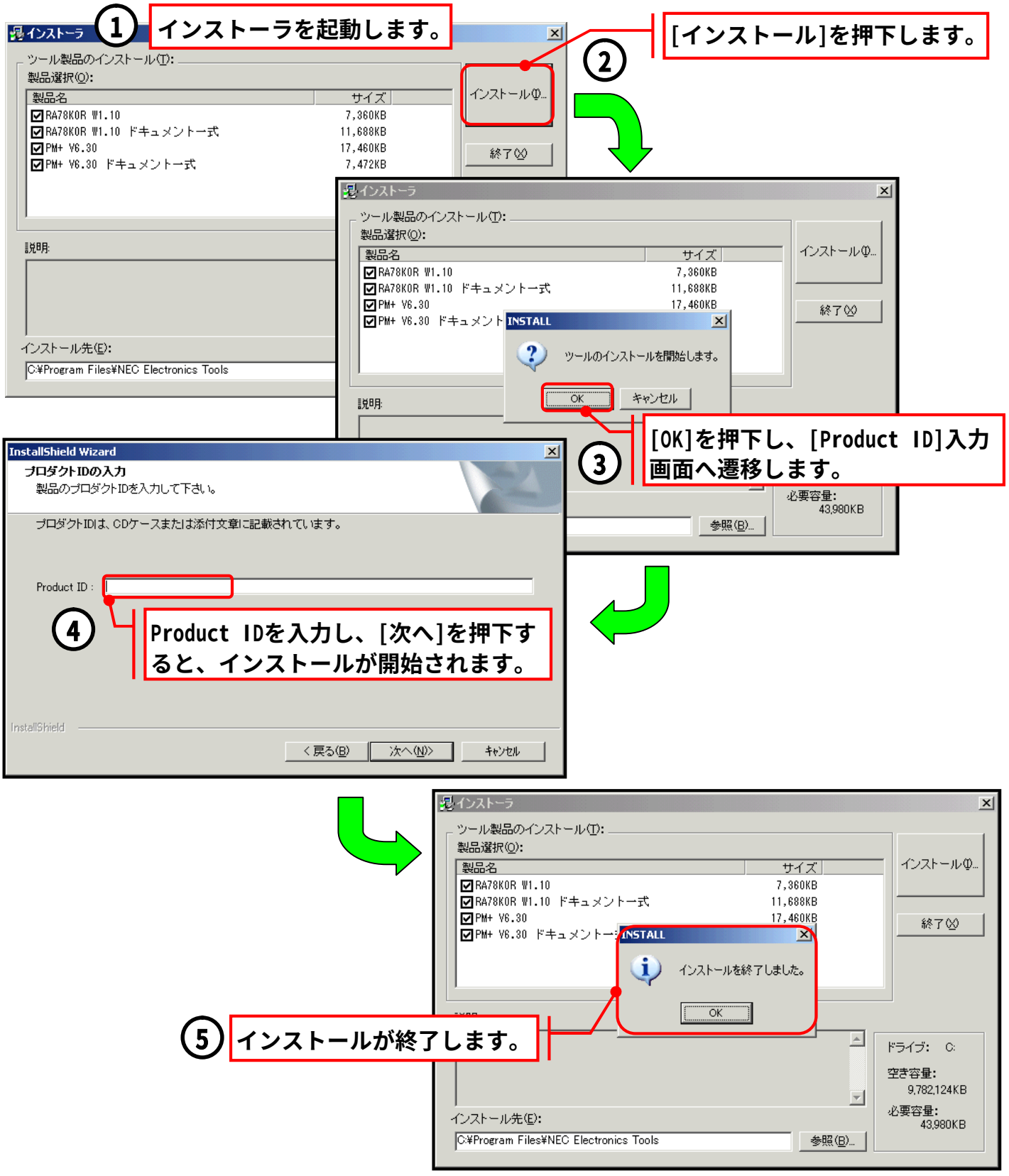
※フリーツールの場合、作成可能なオブジェクトのサイズは64KBの制限があります。

ソフトウエアのインストール

ソフトウエアをインストールします

RA78K0Rのインストール

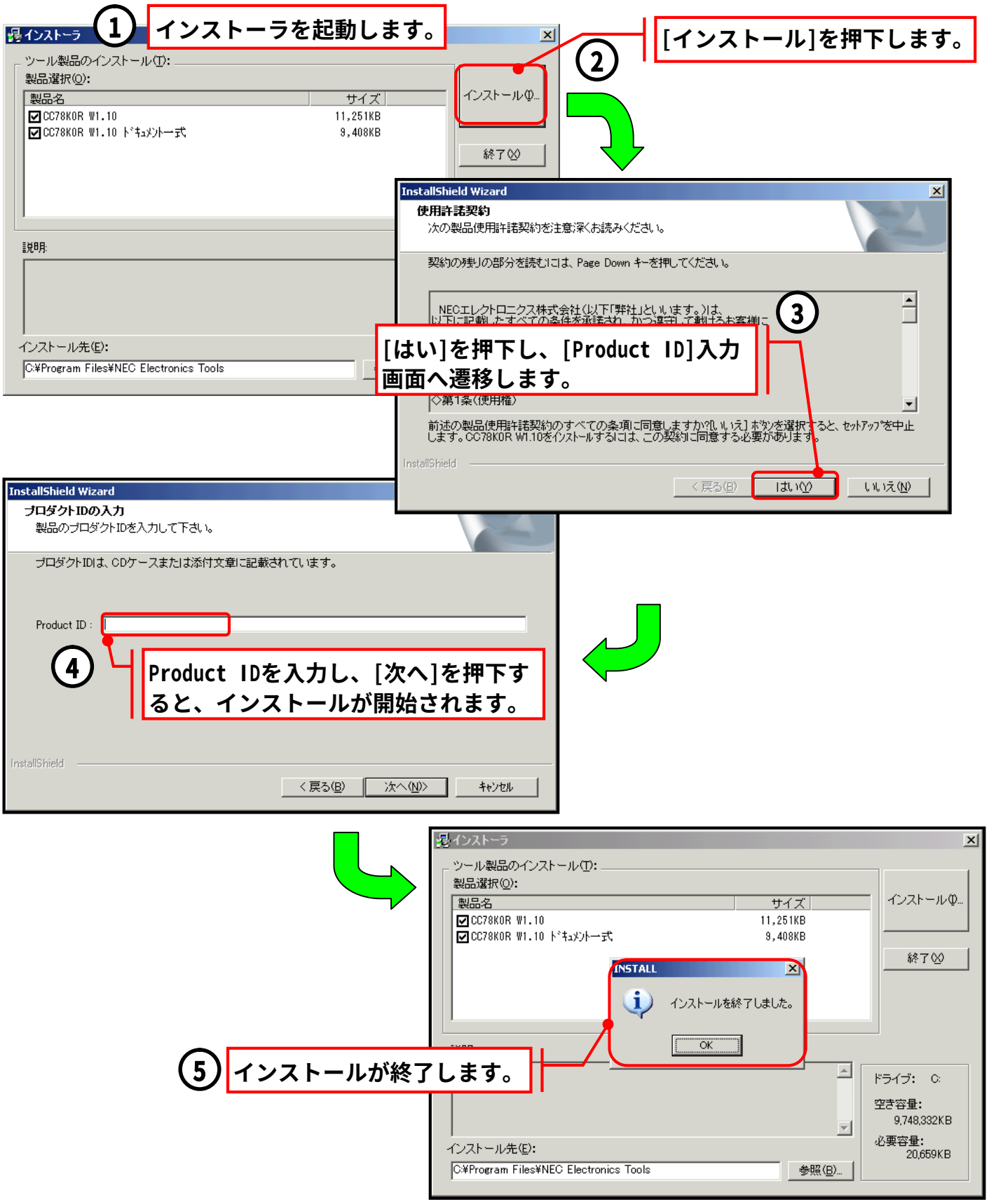
ダウンロードしたファイル[RA78K0R_w110_j.exe]をダブル･クリックし、インストーラを起動します。
[インストール]を押下し、ツールのインストールを開始します。



ソフトウエアのインストール

CC78K0Rのインストール

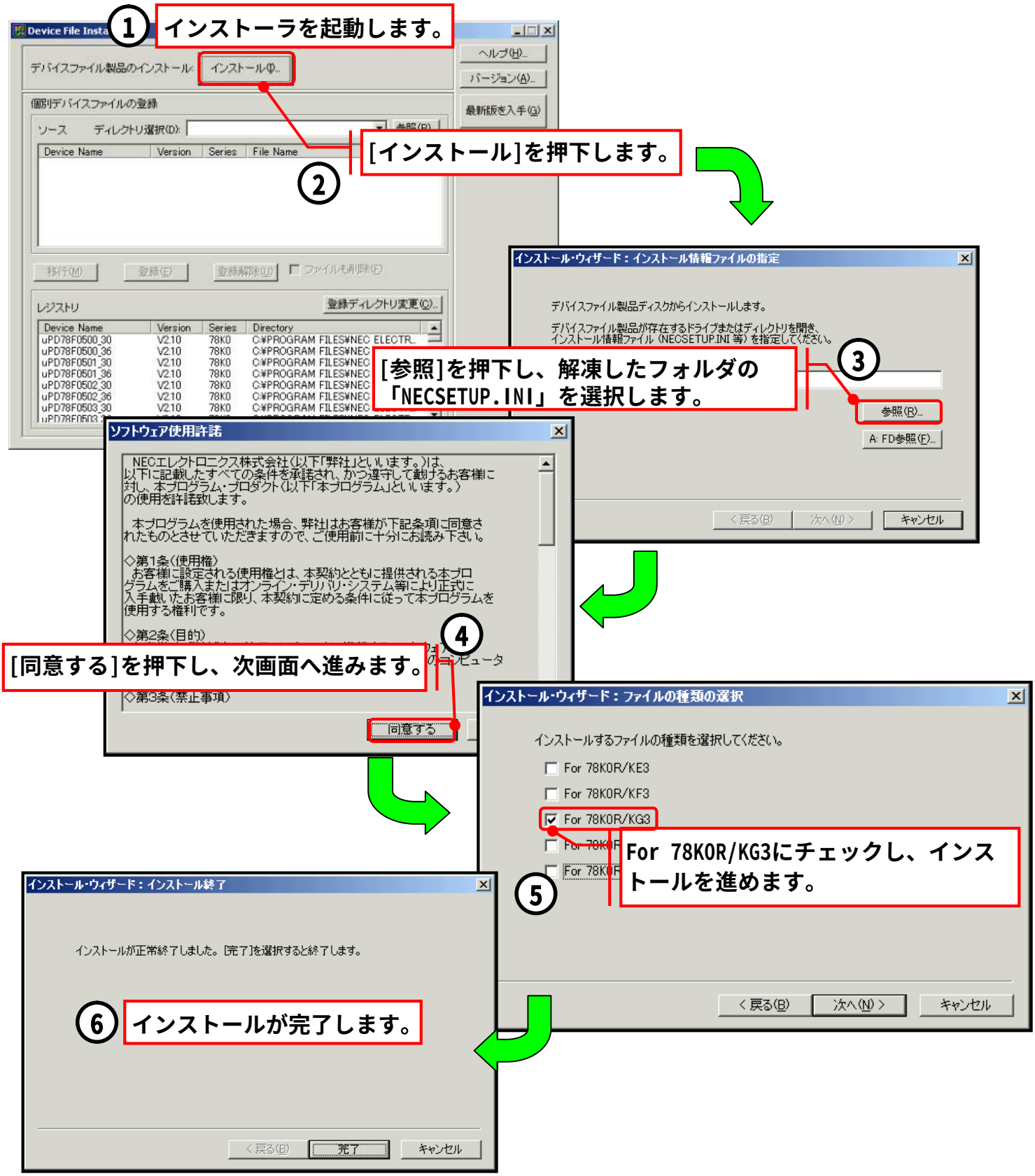
ダウンロードしたファイル[CC78K0R_w110_j.exe]をダブル･クリックし、インストーラを起動します。
[インストール]を押下し、ツールのインストールを開始します。



ソフトウエアのインストール

78K0R/KG3用デバイス・ファイル(DF781188) のインストール

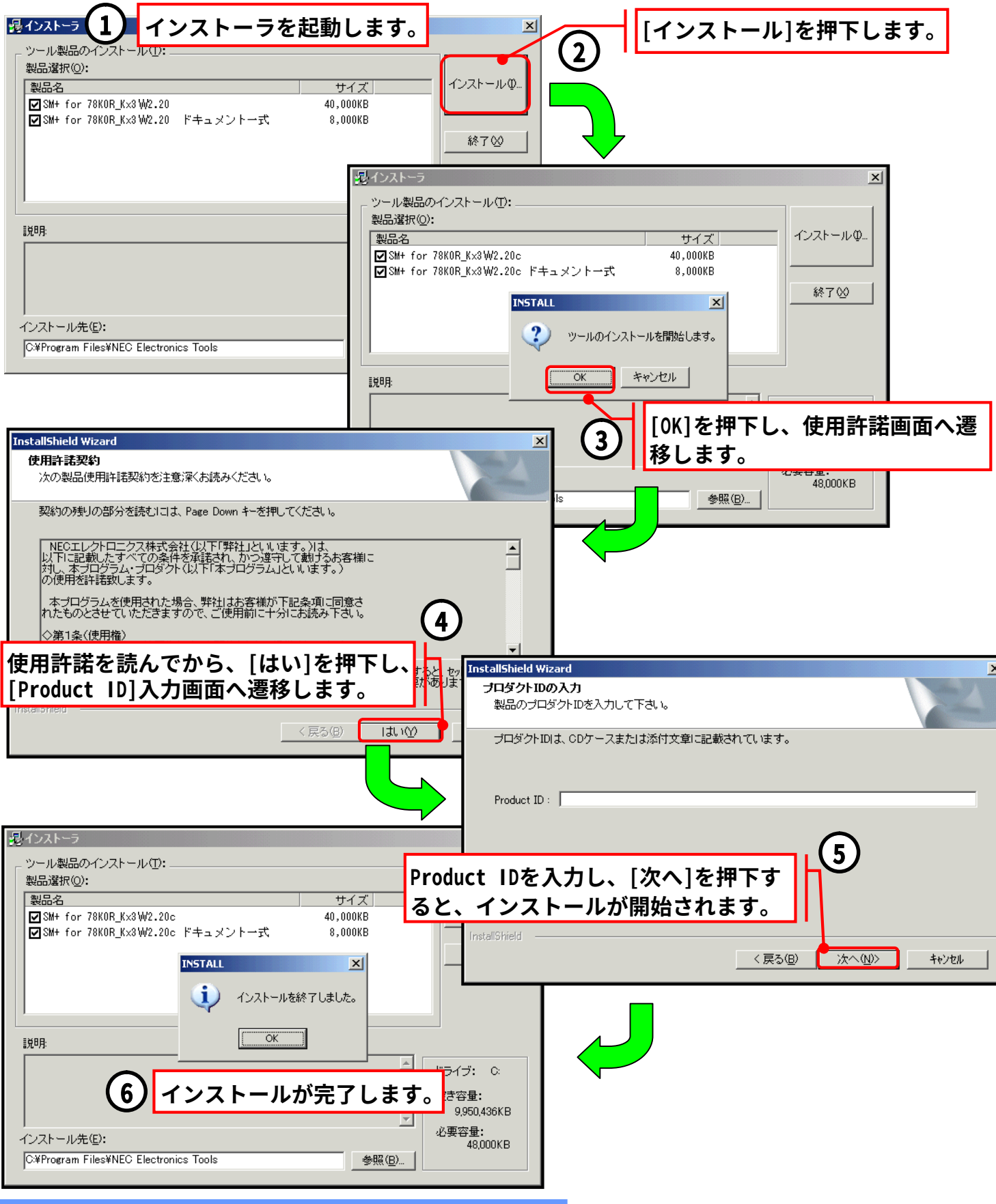
[スタート] → [プログラム(P)] → [NEC Electronics Tools] → [デバイスファイル インストーラ] を起動します。[インストール]を押下して、インストール情報ファイルを指定します。指定するフォルダは「df781188_v300.exe」を解凍したフォルダで、「NECSETUP.INI」を選択します。



ソフトウエアのインストール

SM+ for 78K0R/Kx3のインストール

ダウンロードしたファイル[sm+for78k0r_kx3_w2x0_j.exe]をダブル･クリックし、インストーラを起動します。[インストール]を押下し、ツールのインストールを開始します。

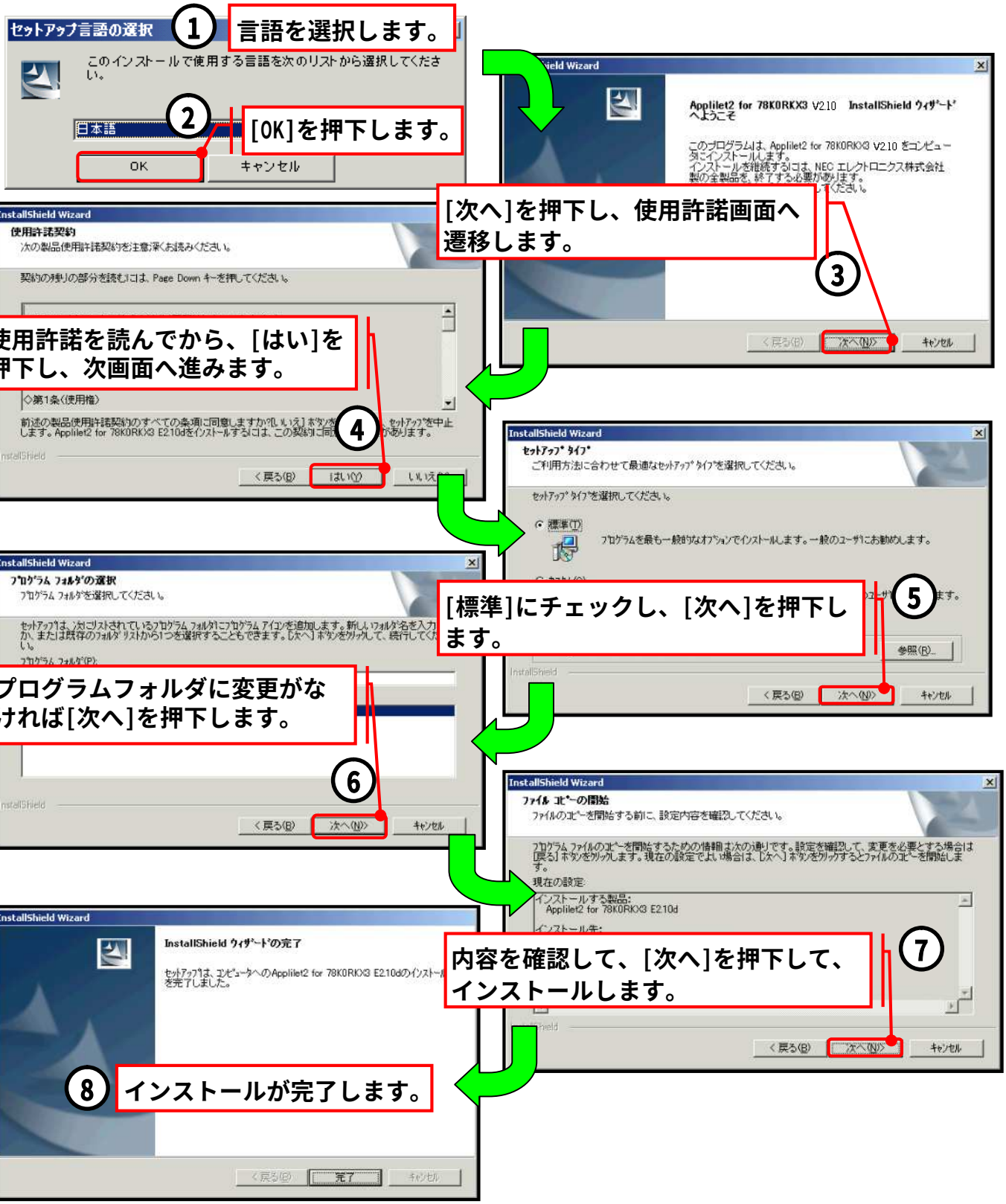


ソフトウエアのインストール

Applilet2 for 78K0R/Kx3のインストール

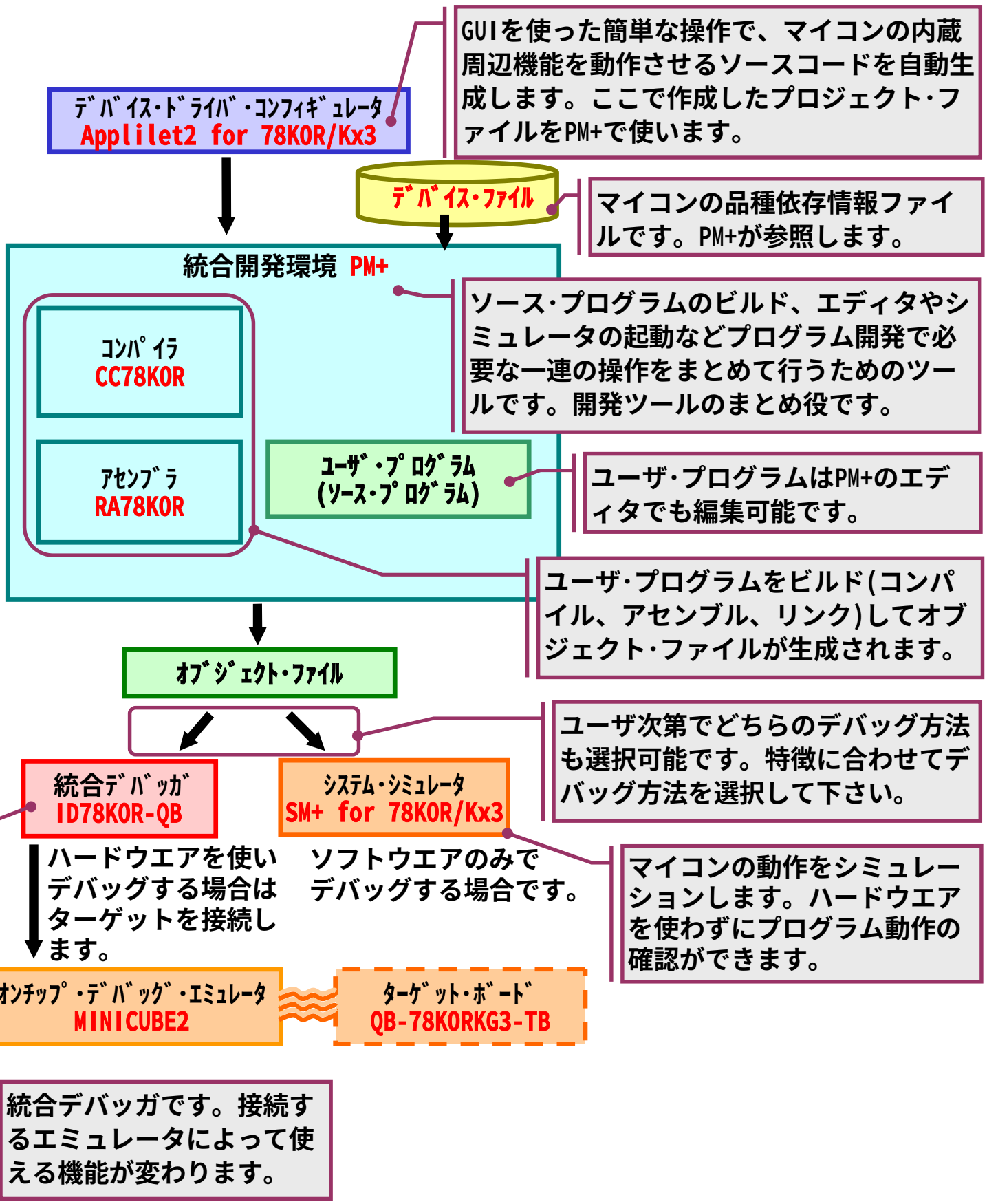
Applilet2 for 78K0R/Kx3は、マイクロソフトの「.NET Framework Version 2.0」をインストールする必要があります。インストールするソフトウエアは、「.NET Framework Version 2.0」ランタイム版で構いません。

ダウンロードしたファイルを解凍し、[setup.exe]をダブル･クリックし、セットアップする言語[日本語]を選択して、インストールを開始します。



開発環境について

NECエレクトニクスの開発ツール環境の全体図を示します。



進化したApplilet2

Appliletの機能が進化して、Applilet2になりました。Appliletは下記の特徴があります。
- **GUIでソースを自動生成**
- **共通のAPIを提供**
- **資源の競合の自動チェック**
- **クロックの変更による、ボーレートや周波数の自動計算**

Applilet2では、下記の新機能が加わりました。
- **ソースコードのガード機能**
- **API名を自由に変更可能**
- **設定した内容のドキュメント化**

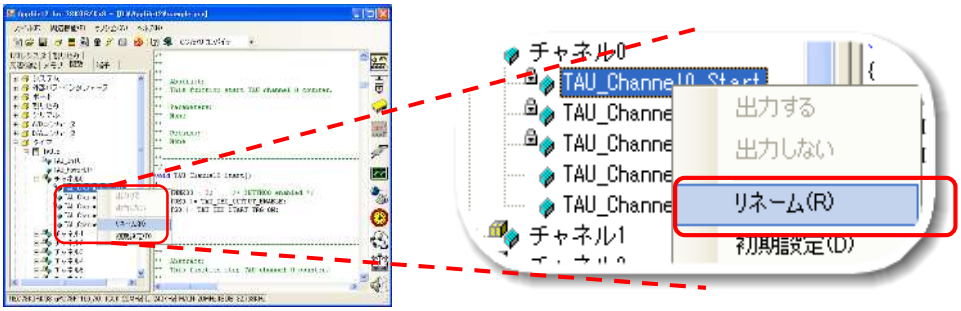
ソースコードのガード機能

プログラムを作成する時、今までのAppliletでは [xxxx_user.c]のファイルを修正することを前提としていました。しかし、Appliletでコード生成を行うと[xxxx_user.c]のファイルフを上書きするか、そのまま残すかありませんでした。Applilet2では、下記のコメント間にプログラムを書けば、どのファイルでも上書きされる心配はありません。例えばタイマ値、割り込みを増やしてもAPIは増えますが、今までに編集したコード部分も残ります。いわゆるマージ機能が追加されました。

```
void function ( void )
{
/* Start user code. Do not edit comment generated here */


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

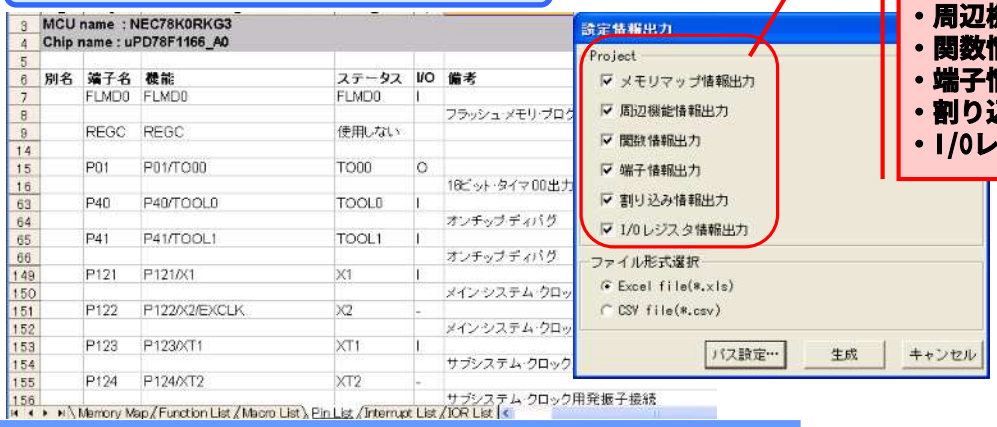
このコメントに挟まれた場所にプログラムを書けば、Applilet2のコード生成で上書きされることはありません。以前に書いたコードは、そのまま残ります。

API名を自由に変更可能



Applilet2で提供するAPIの名前を自由に変更可能です。

設定した内容のドキュメント化

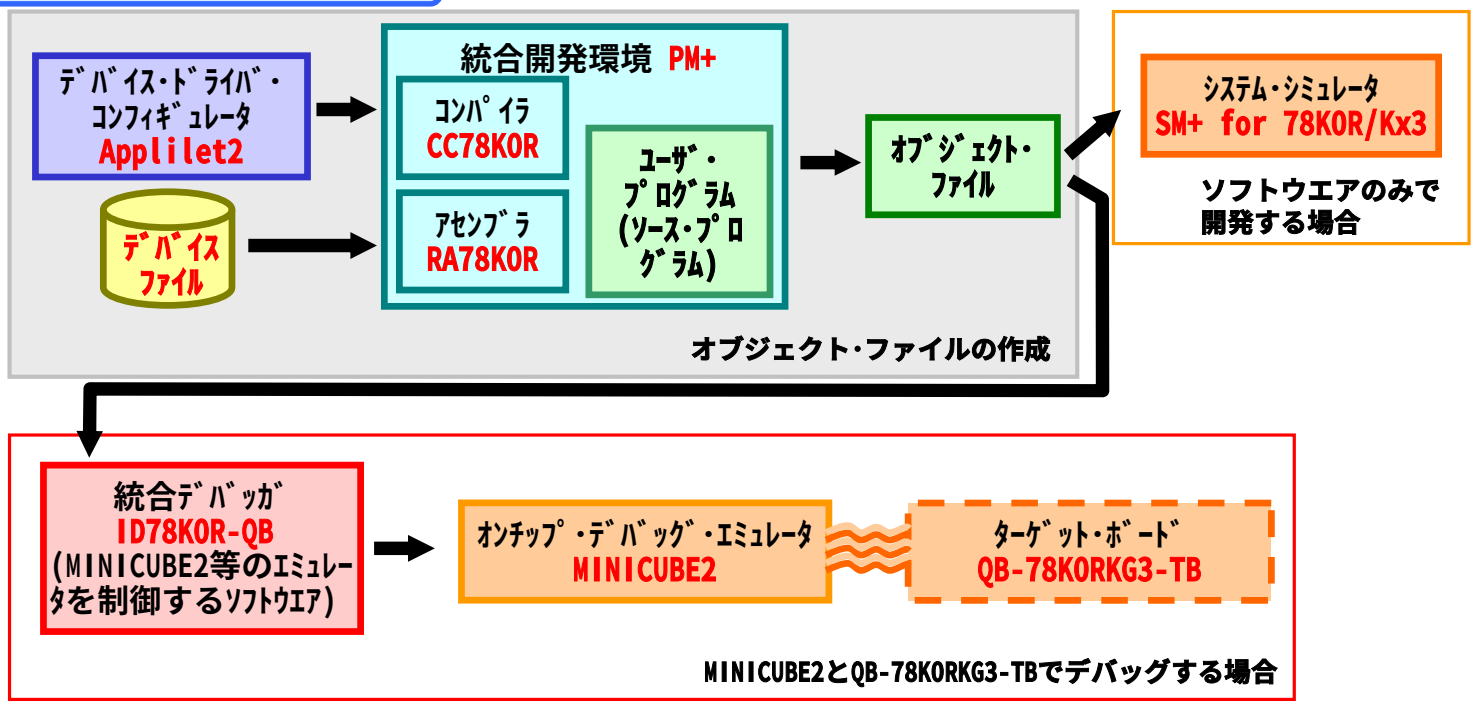


下記の情報をEXCELファイルへ出力します
- **メモリマップ**
- **周辺機能情報**
- **関数情報**
- **端子情報**
- **割り込み情報**
- **I/Oレジスタ情報**

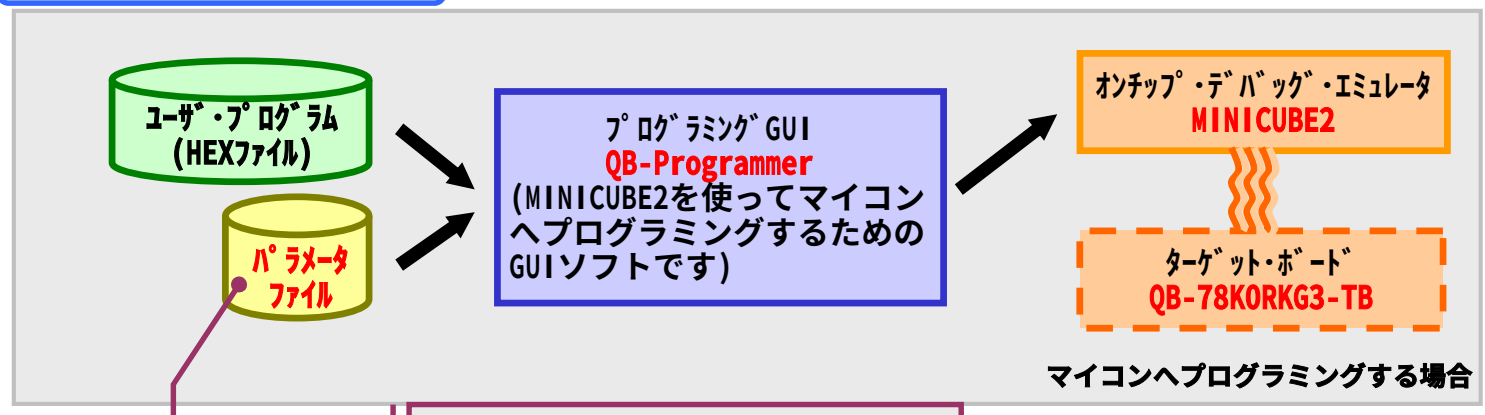
システム構成図

プログラムの開発方法としてソフトウエアのみで行う場合と、ハードウエアを使う場合の2通りの開発方法が選択できます。
・ソフトウエアのみで開発する場合(デバッグもPC上で行います)
・MINICUBE2とターゲット・システムを使う場合(デバッグ時にはマイコンを使います)
完成したプログラムはHEXファイルにしてマイコンへ書き込みます。
・マイコンへプログラミングする

プログラム開発の流れ



プログラミングの流れ



フラッシュ･メモリの書き換え時に使用する品種依存情報ファイルです。

ワンポイント

プログラミングについて
通常プログラミングと言えばプログラムを作成することを示しますが、もう1つの意味があります。
半導体デバイス(マイコン、各種ROMなど)へ書き込みを行う場合も「プログラミング」と呼びます。

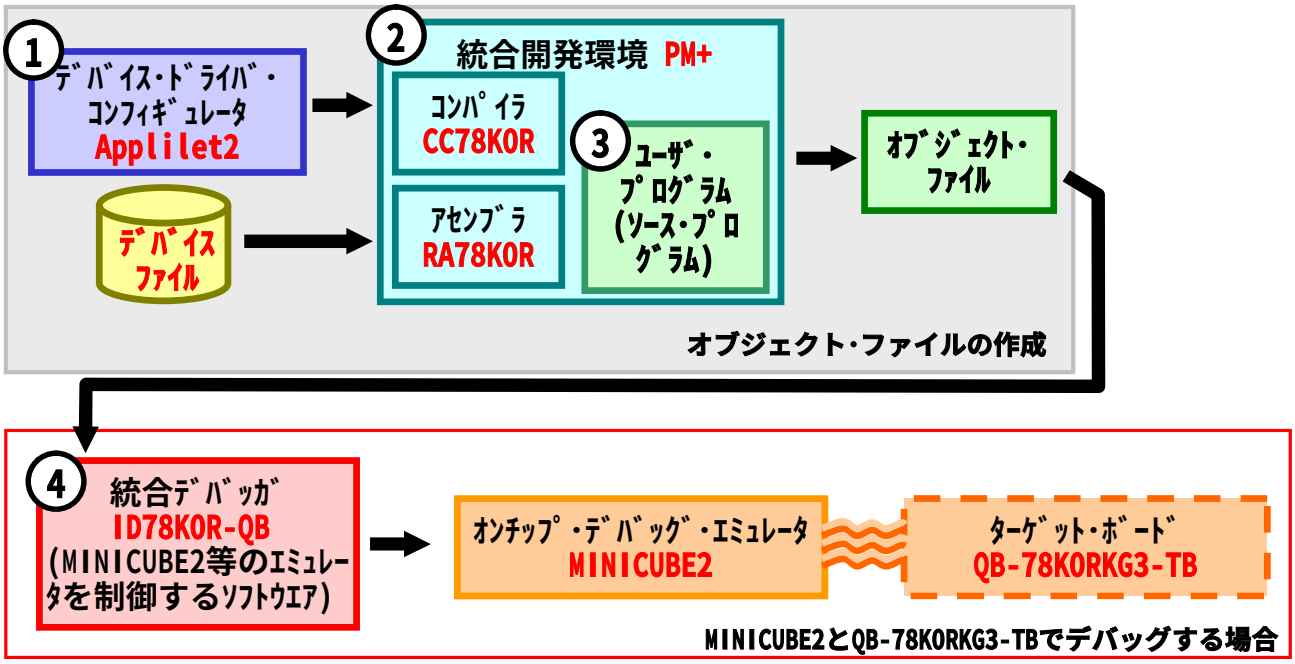
作成手順

45:00

まずハードウエアを動かしてみましょう。細かいことは気にせず本書に従って、進んで下さい。実際にハードウエアを動作させてからプログラムを説明します。[動かしてみよう]-プログラムの実行までは45分程度で試すことができます。あせらず、ゆっくり進んで下さい。

※[準備]の[ソフトのダウンロード]、[ソフトのインストール]を済ませてから作業を行って下さい。

プログラム作成手順



① Applilet2を使ってマイコンの設定を行います。 GUIを使った簡単な操作で、マイコンの内蔵周辺機能を動作させるソースコードとプロジェクトファイルが自動生成されます。

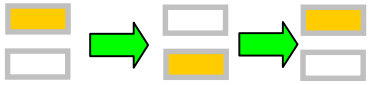
② ①で生成したプロジェクトを読み込みます。PM+はソースコードのビルド、エディタや統合デバッガの起動などプログラム開発で必要な一連の操作をまとめて行うためのツールです。

③ ユーザ･プログラムを作成します。作成したプログラムをコンパイルしてオブジェクト･ファイルを作ります。

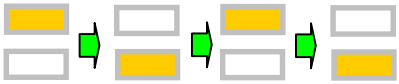
④ ③で作成したオブジェクト･ファイルを元にデバッグします。MINICUBE2とQB-78K0RKG3-TBを使います。実際にマイコンを動作させます。

ワンポイント

ここで作成するサンプル･プログラムはSWを1つ、LEDを2つ使います。交互に点滅を繰り返すLEDをSWの押下によって点滅のスピードを変化させる簡単なプログラムです。


LEDが交互に点滅しています


SWを押下すると

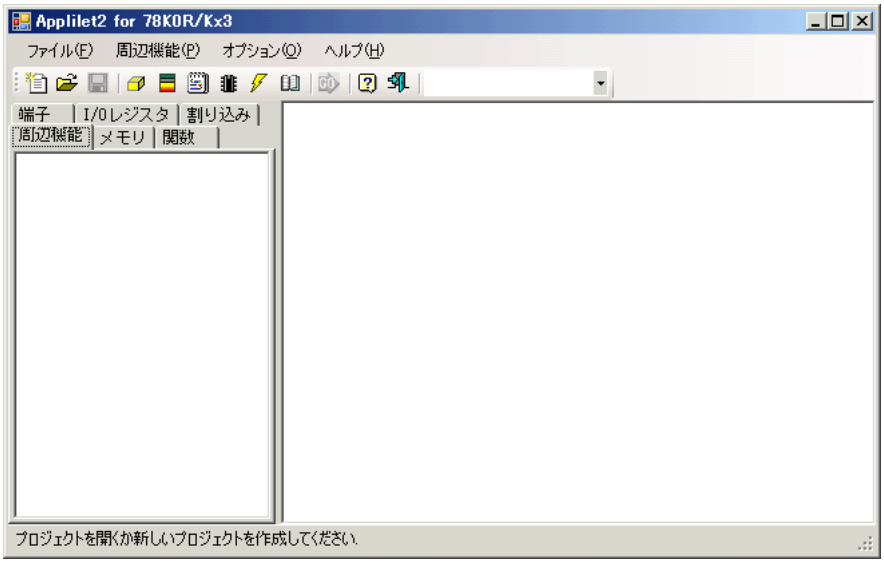

LEDの点滅速度が変化します

Applilet2でプロジェクト作成

① Applilet2を使い統合開発環境 PM+で読み込み可能なプロジェクトファイルを作成します。

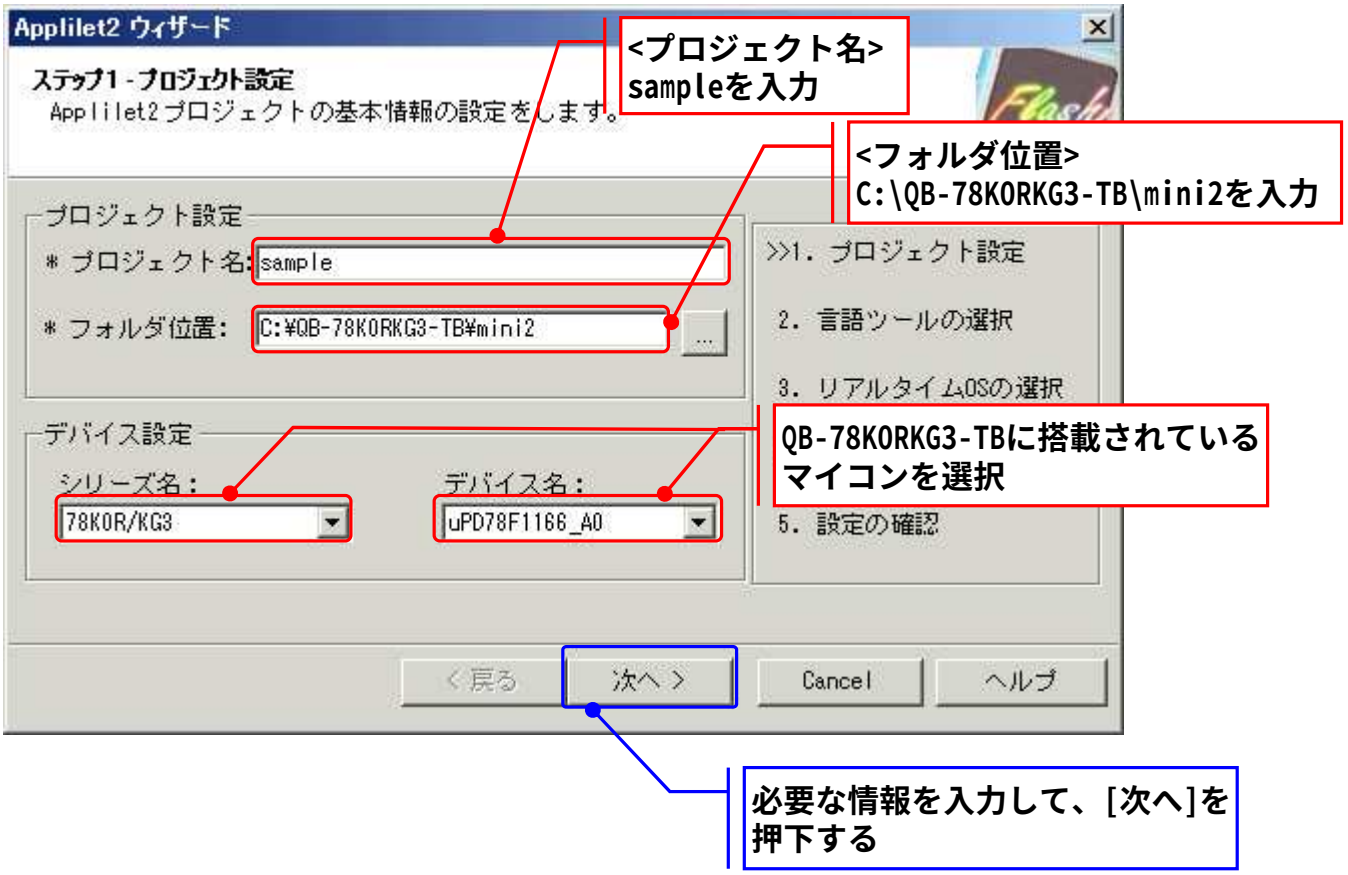
a. Applilet2を起動します。

[スタート] → [プログラム(P)] → [NEC Electronics Tools] →
[Applilet2 for 78K0RKX3] → [Vx.xx] → [Applilet2 for 78K0RKX3 Vx.xx]



b. Applilet2のプロジェクトファイルを新規に作成します。

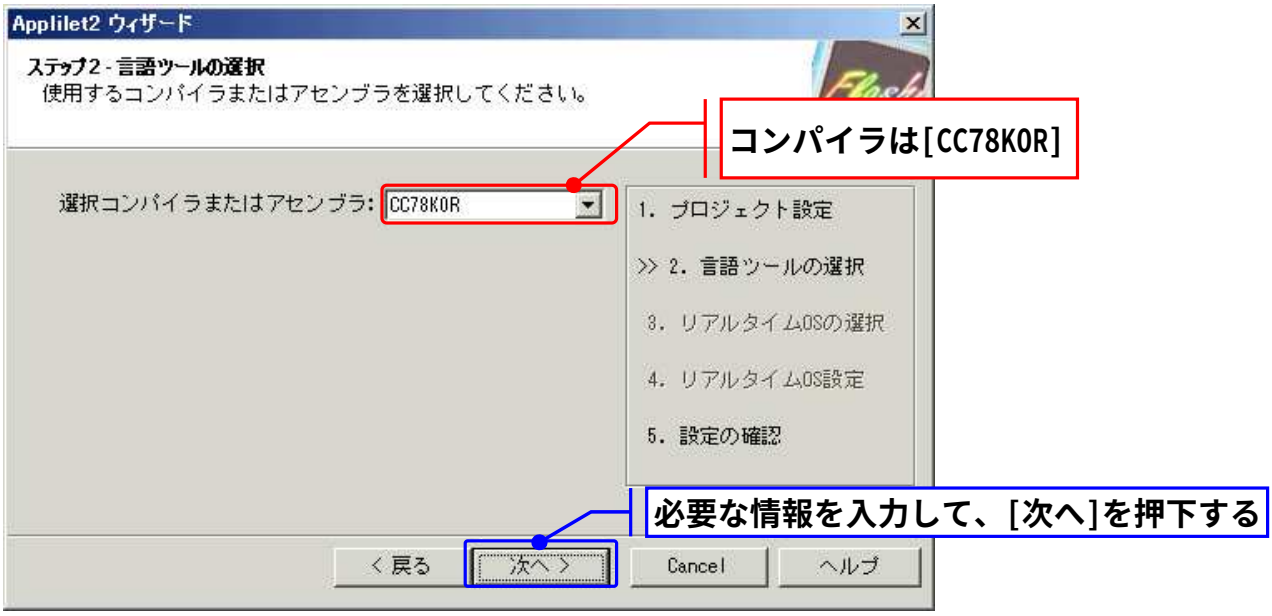
メニュー・バーの[ファイル(F)]→[新規作成(N)...]を選択します。
「Applilet2ウィザード」ダイアログで、必要な情報を設定して下さい。



Applilet2でプロジェクト作成

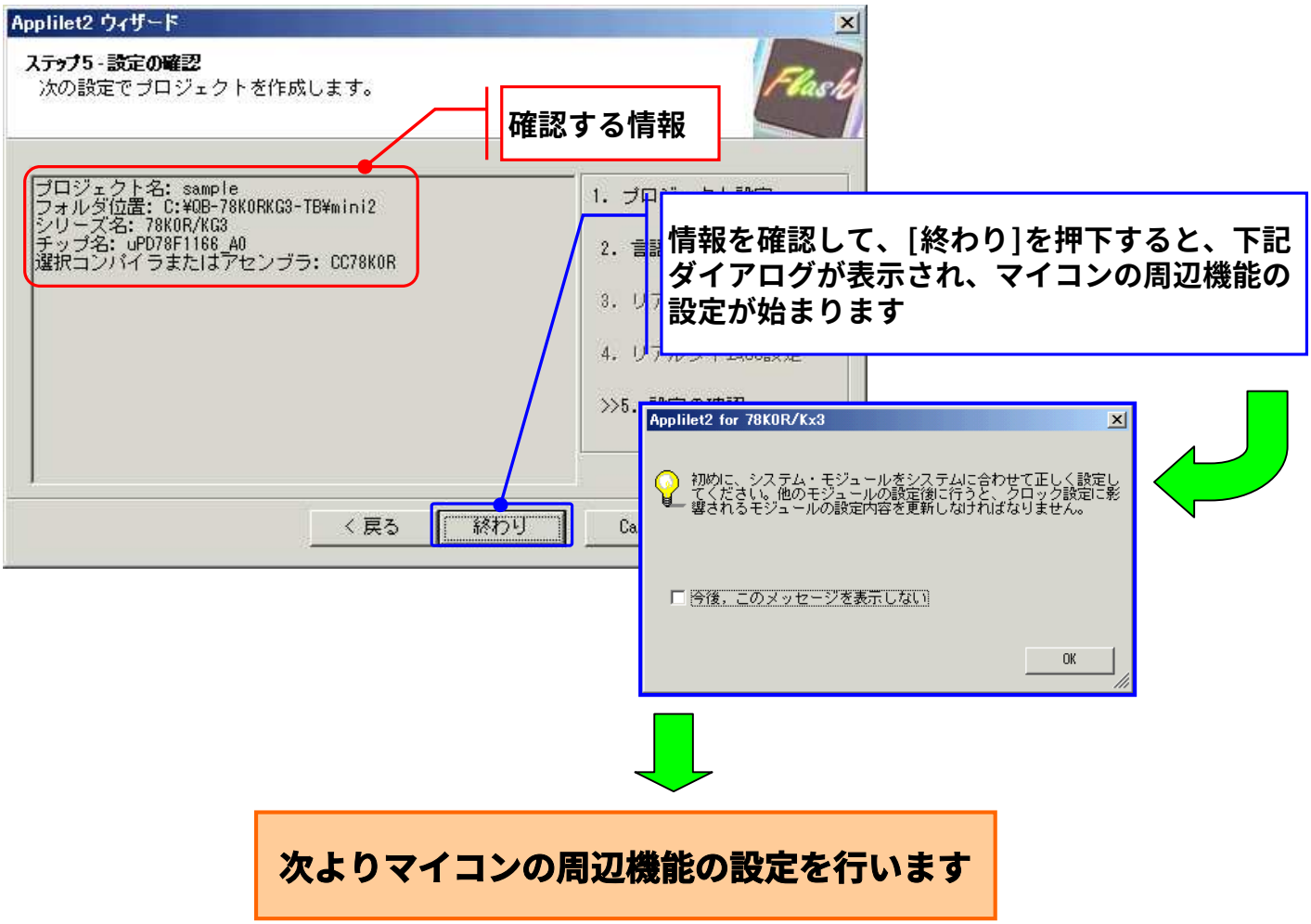
c. 言語ツールを選択します。

[CC78K0R]が表示されていることを確認して[次へ]を押下します。



d. Applilet2の設定を確認します。

下図の画面と同じになっているか確認して下さい。

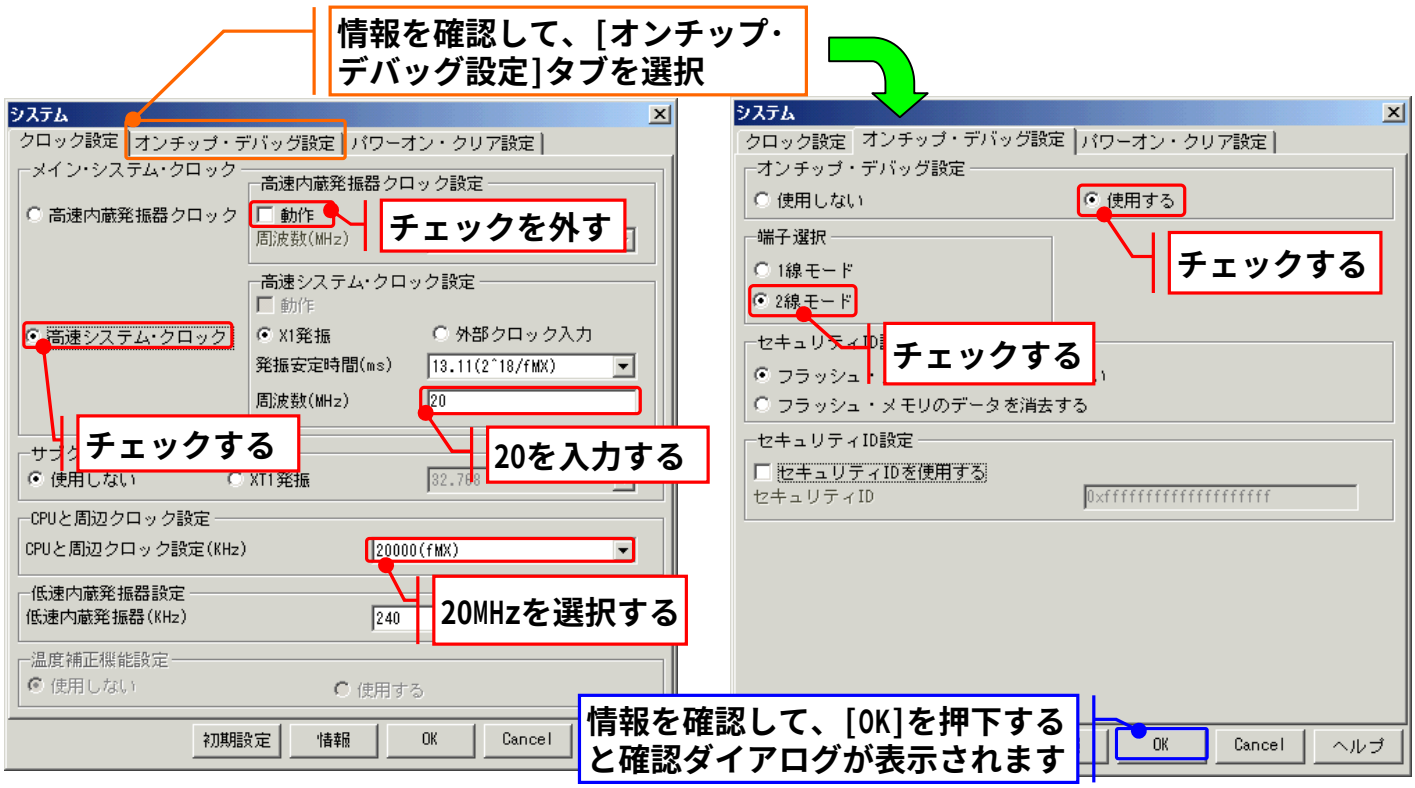


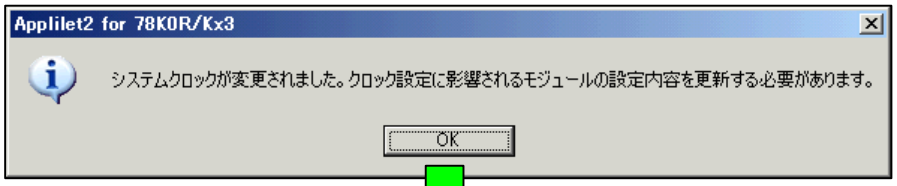
次よりマイコンの周辺機能の設定を行います

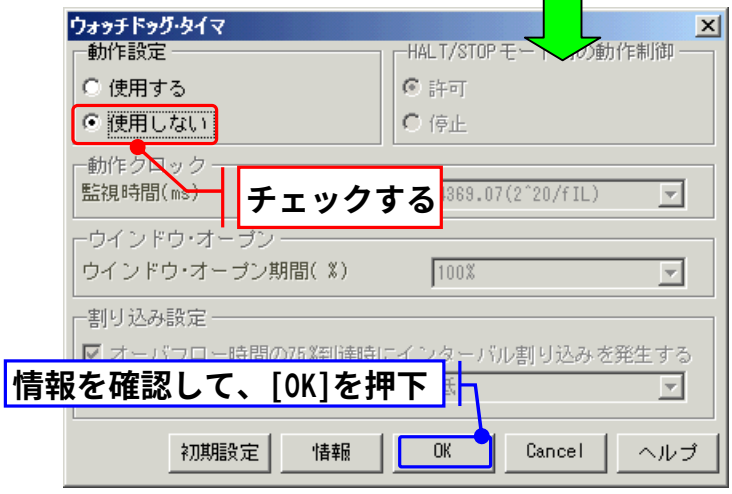
周辺機能設定[システム]

e. [システム]を設定します。

[システム]ダイアログは、クロック等マイコンの基本を設定します。







ワンポイント

ウォッチドッグ・タイマについて

ウォッチドッグタイマとはプログラムの暴走を検出するための機構です。プログラム暴走と検出された場合は内部リセット信号が発生されマイコンはリセットされます。より信頼性の高いプログラムにするためにはウォッチドッグタイマを使用します。

ワンポイント

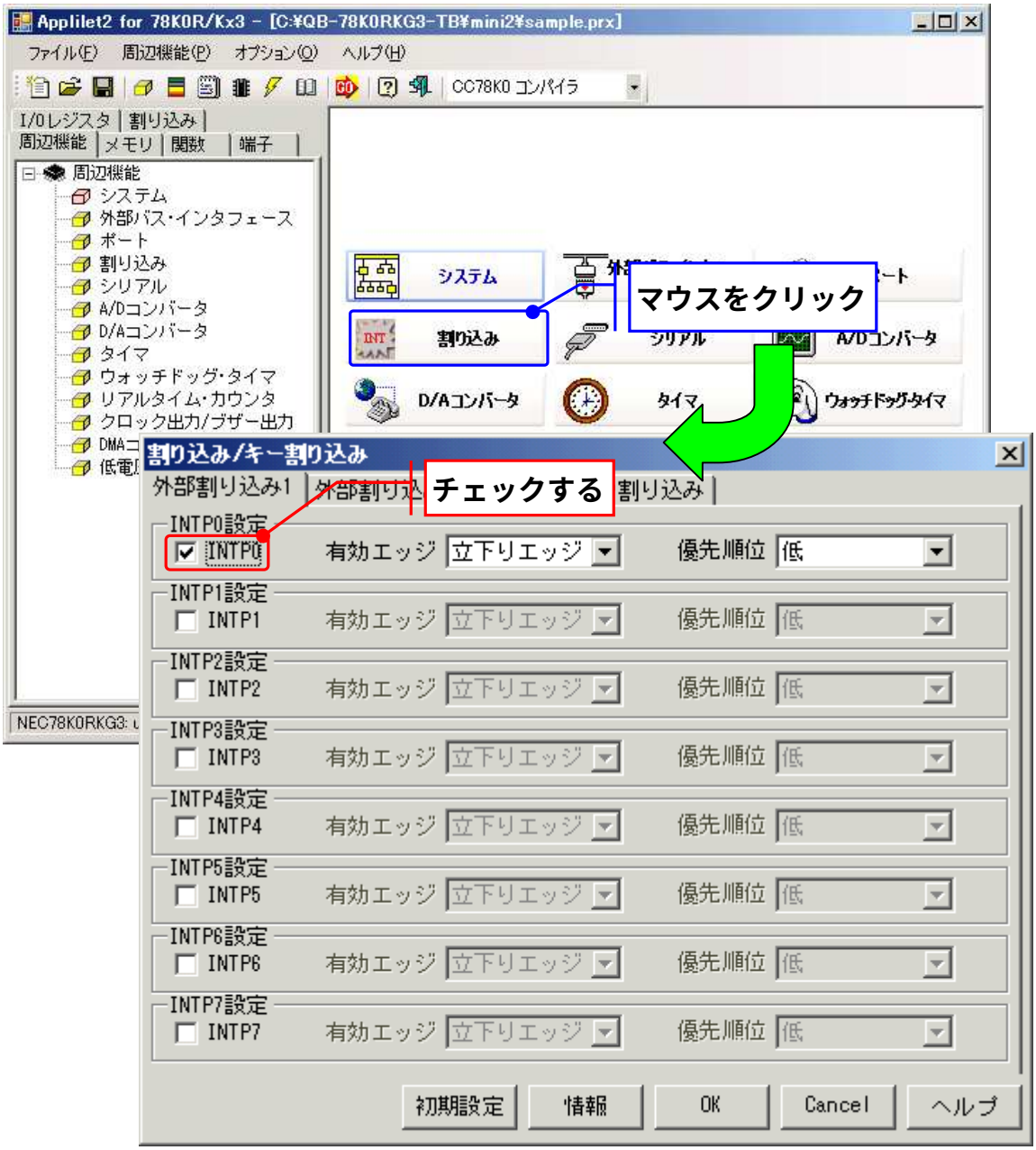
システムについて

メイン・システム・クロック、サブクロックの設定を行います。ここで指定したクロック値は、タイマ・モジュールのコンペア・ レジスタ値の計算やシリアル・モジュールのボーレートに影響を与えます。後から動作クロックを変更した場合には、シリアルのボーレートの値等を確認して下さい。

周辺機能設定[割り込み]

f. [割り込み]を設定します。

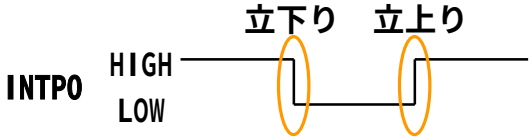
[割り込み]ダイアログは、外部割り込みを設定します。



ワンポイント

外部割り込みについて

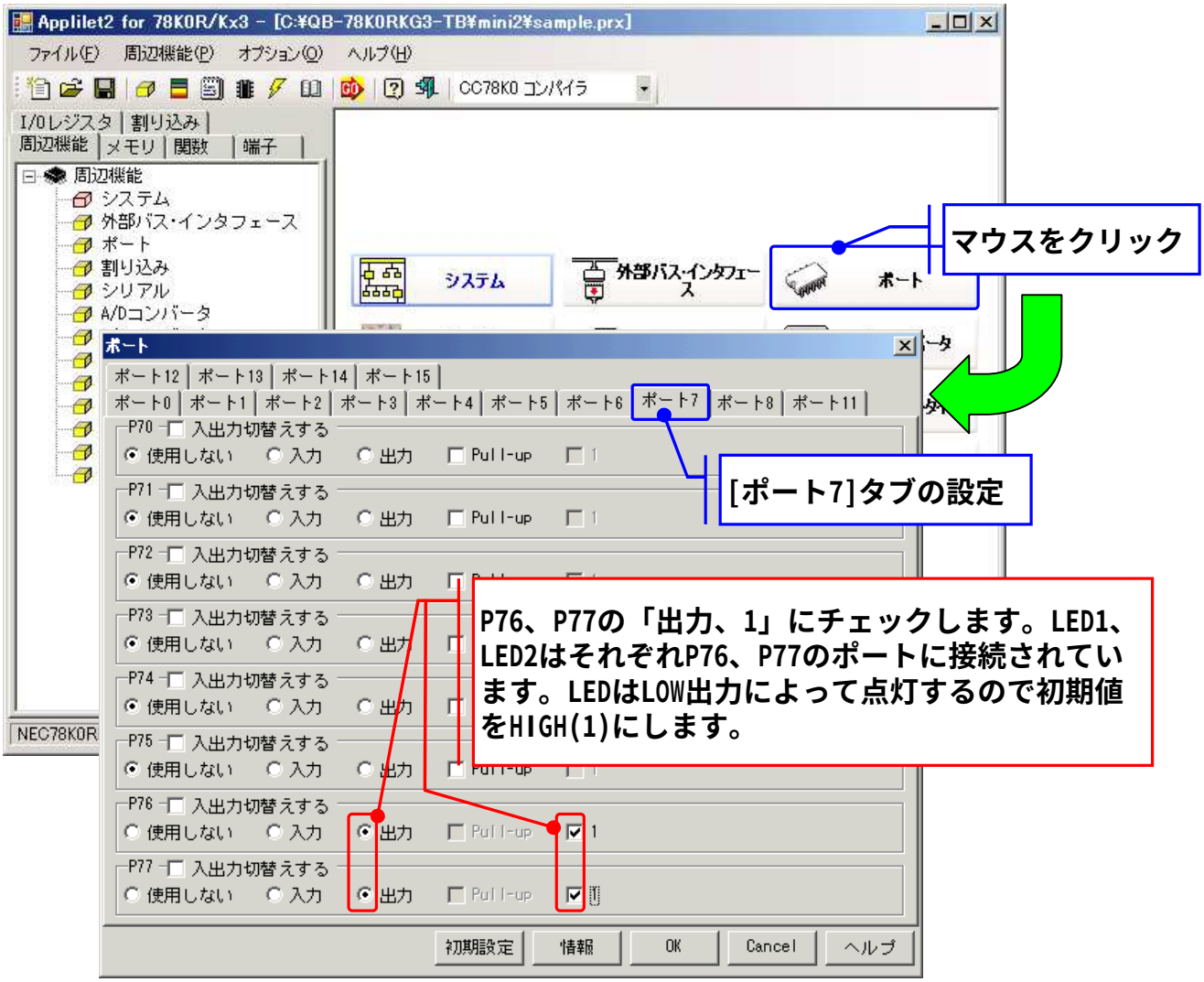
外部端子割り込み、キー割り込みの設定を行います。サンプルプログラムではINTP0の外部端子にSWが接続されています。有効エッジの**立下がりエッジ**とは、信号が1から0へ変化するときに有効とする設定です。ターゲット・ボードのSWを押した時が立ち下がり、押してからSWを離した時が立ち上がりになります。「立下がり」の逆が「立上がり」です。

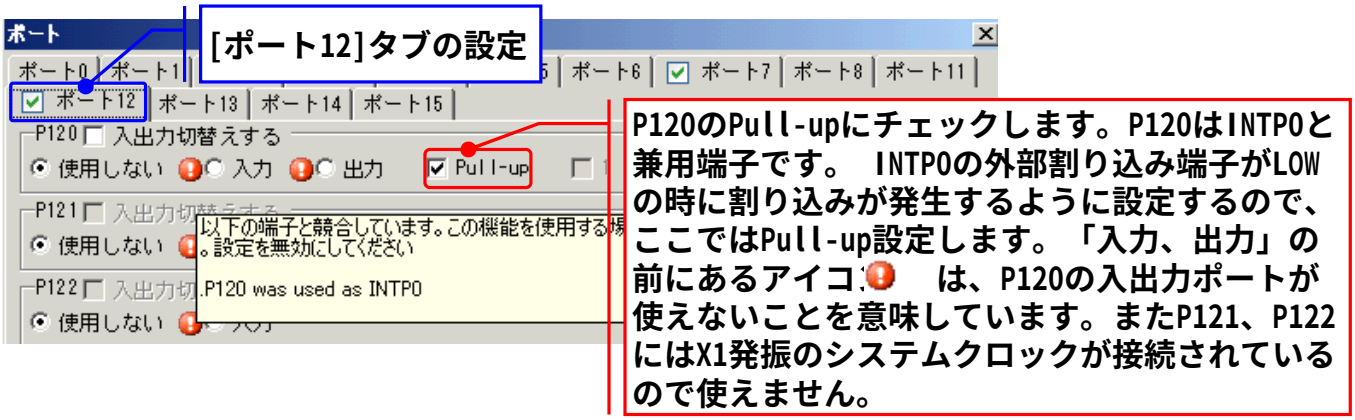


周辺機能設定[ポート]

g. [ポート]を設定します。

[ポート]ダイアログは、ポートの入出力を設定します。





ワンポイント

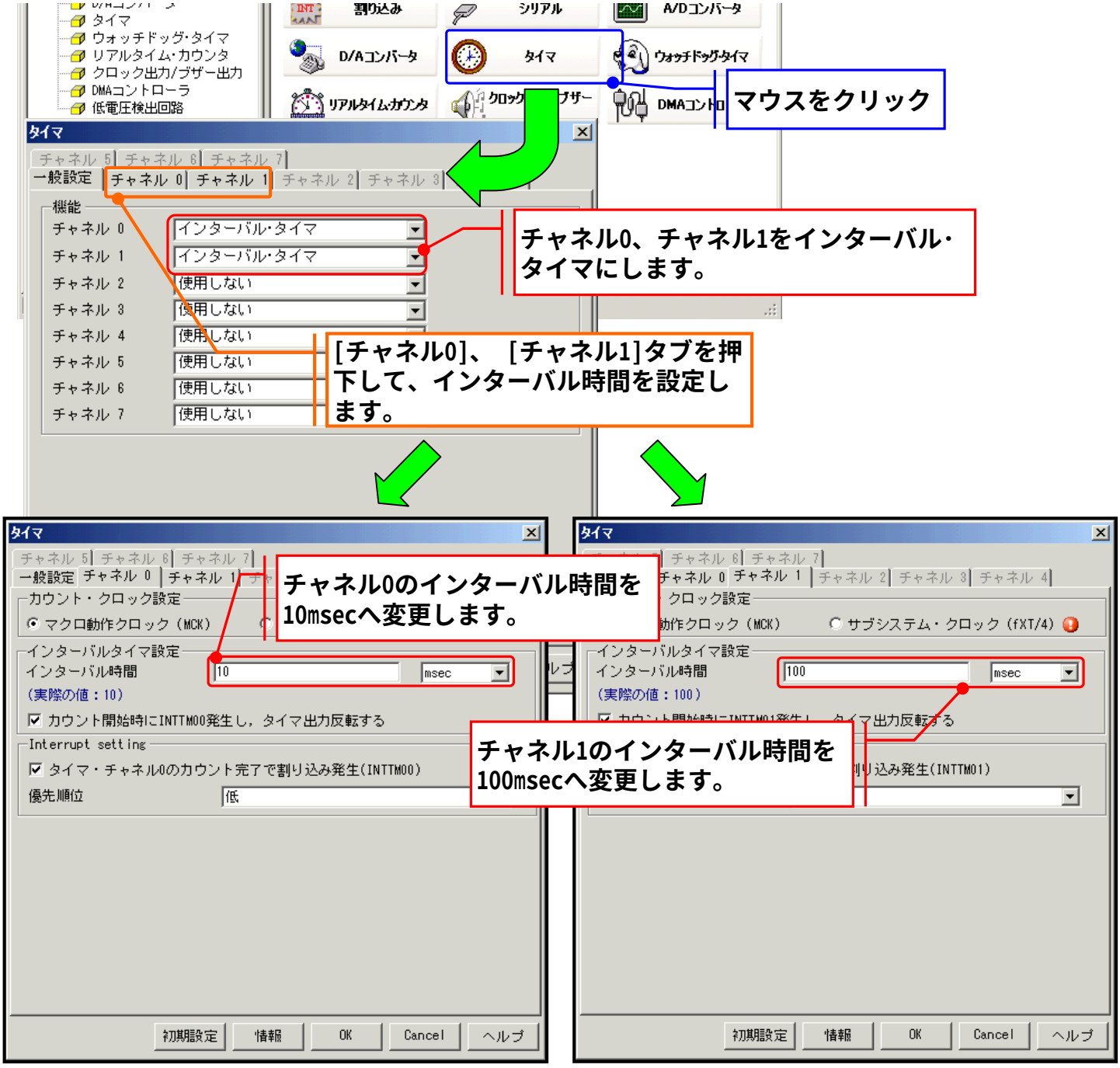
ポートについて

各ポートの設定を行います。各ポートは入力/出力、内蔵プルアップ抵抗(Pull-up)、初期値の設定が可能です。ポートは他の周辺I/Oと兼用端子になっている場合が殆どです。

周辺機能設定[タイマ]

h. [タイマ]を設定します。

[タイマ]ダイアログで、タイマ割り込みを設定します。



ワンポイント

タイマについて

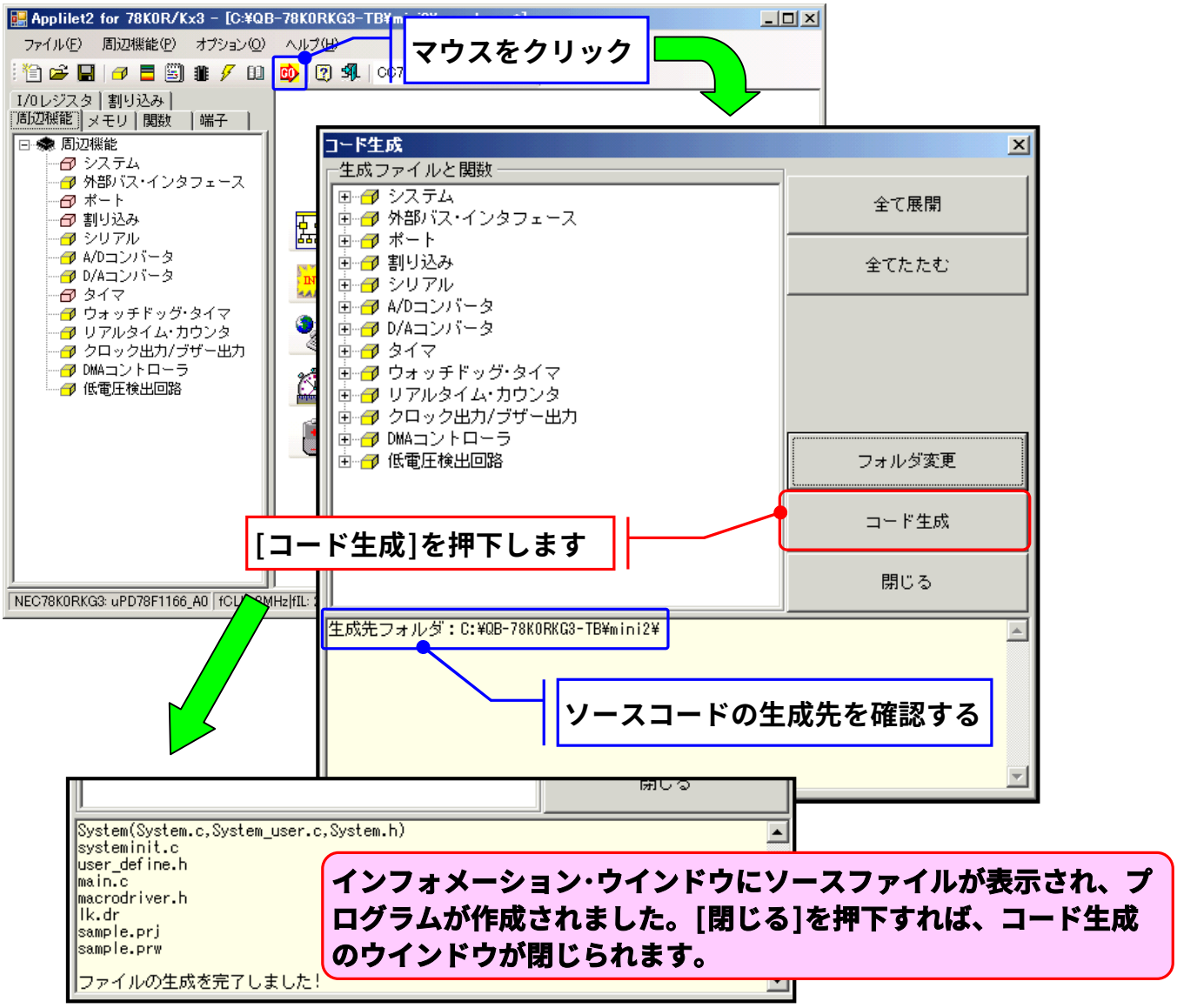
タイマにはインターバル･タイマの他にも様々な機能があります。

外部イベント･カウンタ（外部から入力される信号のパルス数を測定できます）、**方形波出力**（任意の周波数の方形波出力が可能です）、**PPG出力**（周波数と出力パルス幅を任意に設定できる矩形波を出力できます）、**ワンショット・パルス出力**（出力パルス幅を任意に設定できるワンショット・パルスを出力できます）、**パルス幅測定**（外部から入力される信号のパルス幅を測定できます）、**PWM出力**などがあります。

コード生成

i. コード生成します。

コード生成アイコン を押下してプログラムを生成します。



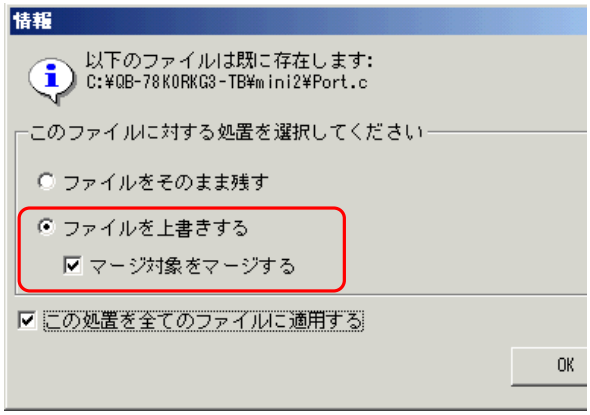
ワンポイント

コード生成について

すでにソースファイルが存在していた場合は、右図のダイアログが表示されます。「ファイルを上書きする」、「マージ対象をマージする」にチェックすれば、下記のコメントに、はさまれたソースコードは上書きされずにマージされます。

```
/* Start user code. Do not edit comment generated here */
        ユーザのソースプログラム
/* End user code adding. Do not edit comment generated here */
```

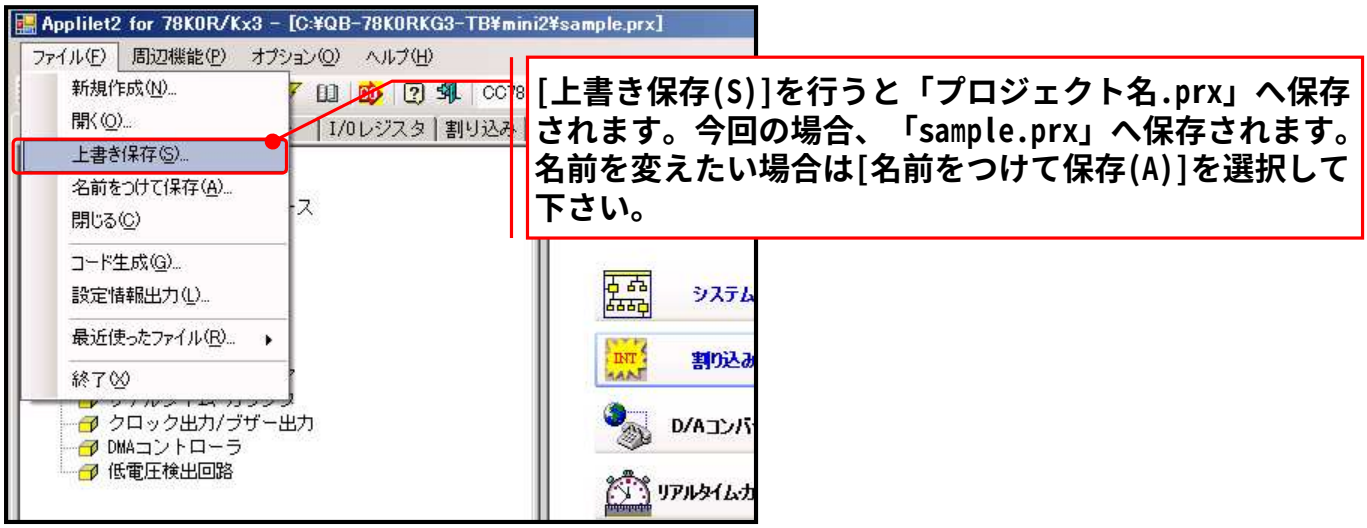
これはApplilet2の新機能「ソースコードのガード機能」です。



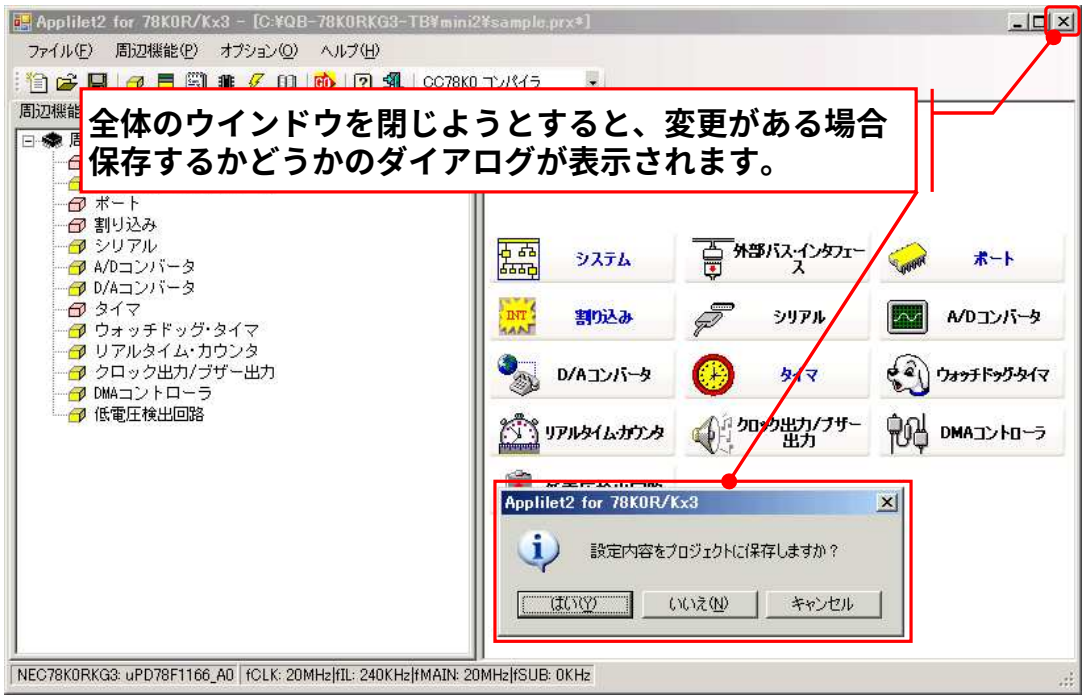
プロジェクトの保存

j. プロジェクトの保存

メニューより ファイル(F) -> 上書き保存(S)...でプロジェクトを保存します。



また、Applilet2のウインドウ自体を閉じようとすると、設定に変更がある場合プロジェクトを保存するかダイアログが表示されます。



PM+を起動する

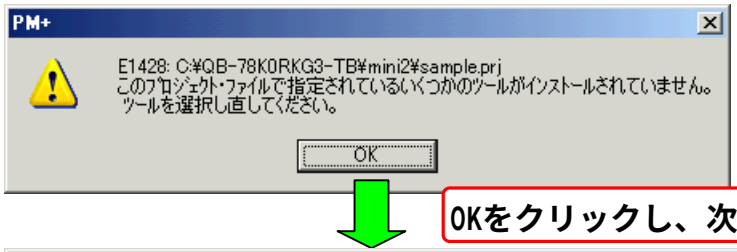
② 統合開発環境 PM+でプロジェクトファイルを読み込み環境の設定を行います。

k. PM+を起動します。

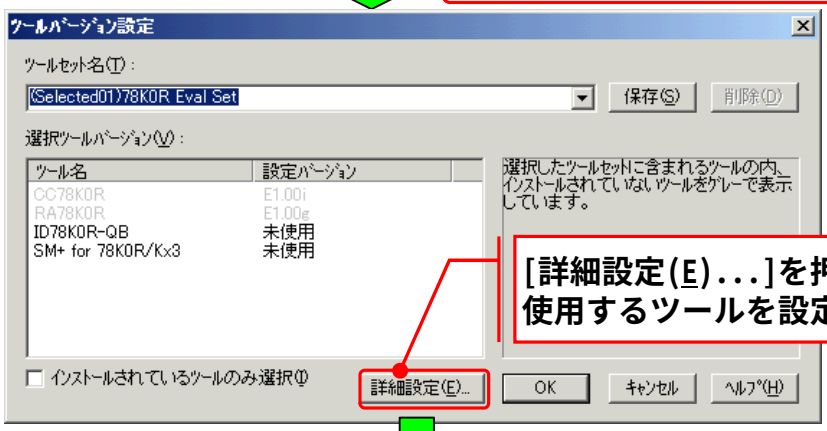
[スタート] → [プログラム(P)] → [NEC Electronics Tools] → [PM+ Vx.xx]

l.ワークスペース(sample.prw)を開きます。

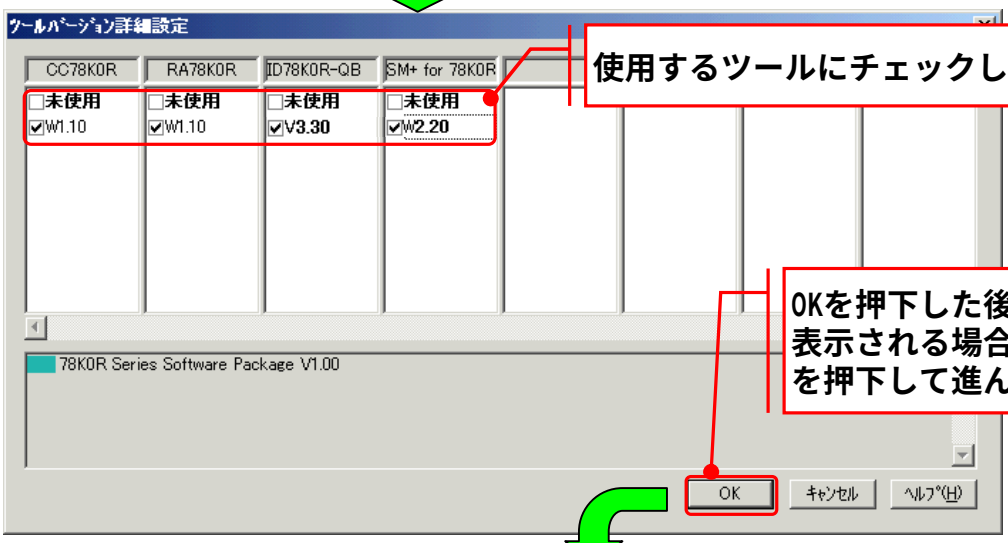
メニューの[ファイル(F)] → [ワークスペースを開く(W)...] を選択し、
”c:\QB-78K0RKG3-TB\mini2\sample.prw ” を指定して、[開く(O)] ボタンを押して下さい。
インストールされているツールのバージョンが異なる場合、下記のダイアログが表示されます。



OKをクリックし、次へ進みます

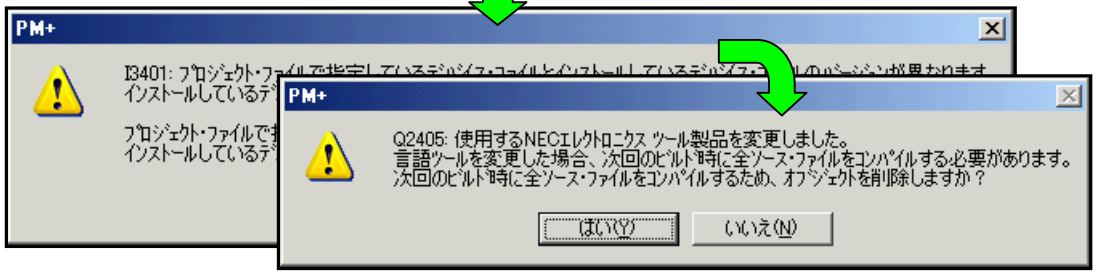


[詳細設定(E)...]を押下し、
使用するツールを設定します。



使用するツールにチェックします。

OKを押下した後に、下記のダイアログが
表示される場合もありますが、[はい(Y)]
を押下して進んで下さい。

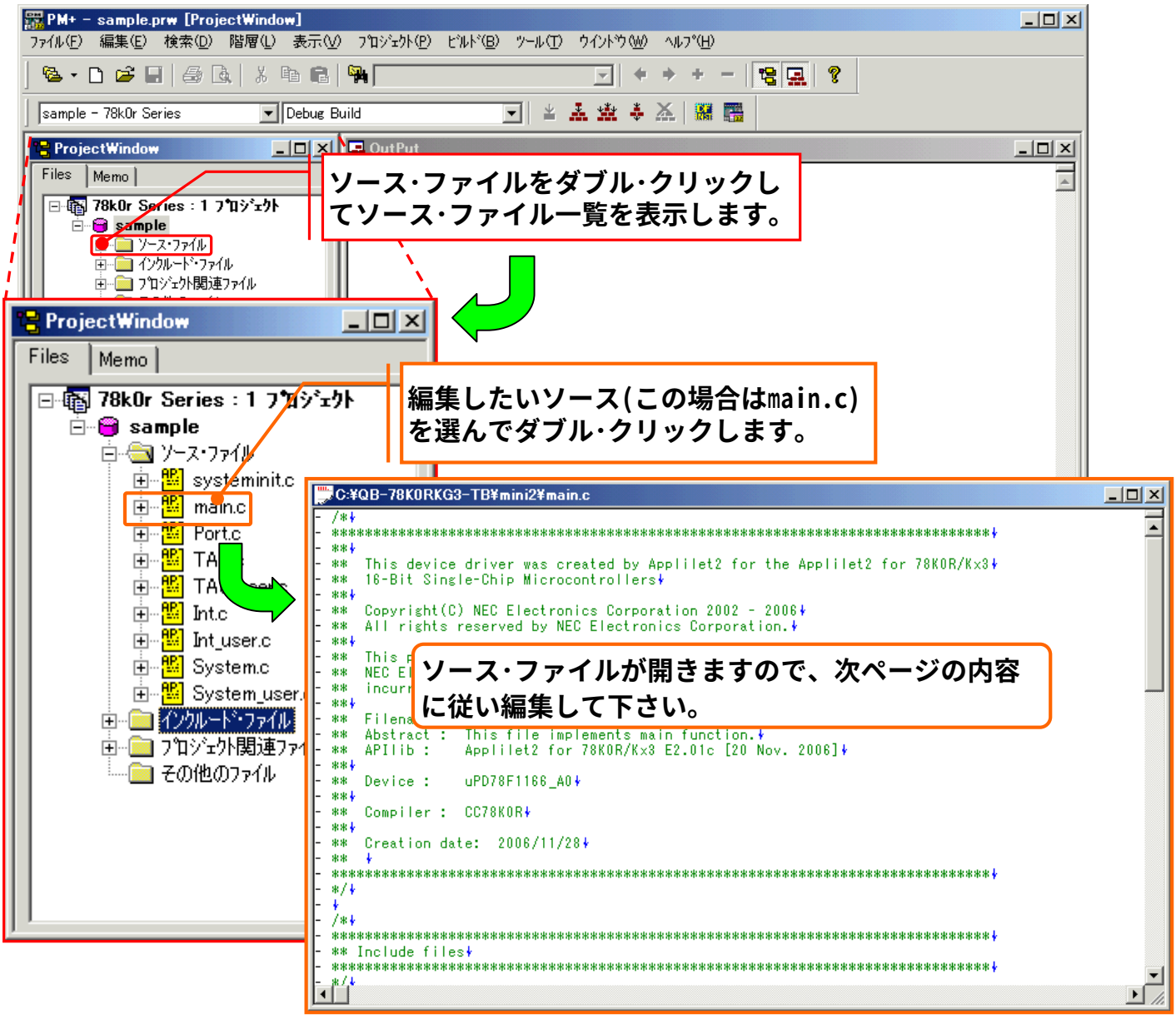


プログラムの編集

③ ユーザ･プログラムを作成します。

m. 編集するソースファイルを開きます。

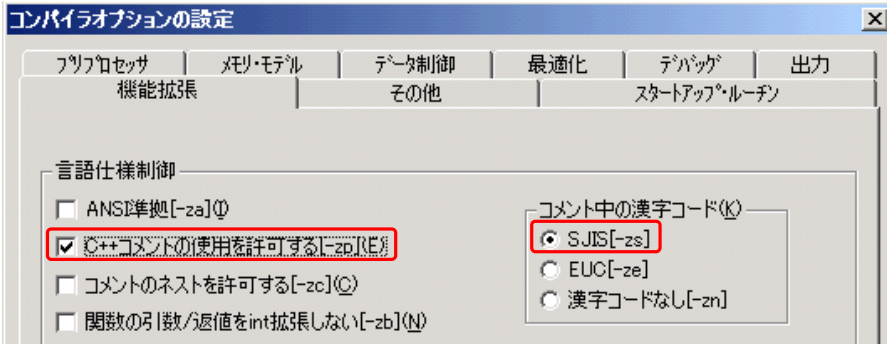
[ProjectWindow] の ソース･ファイルをダブル･クリック → 編集したいソースをダブル･クリックします。次ページの内容に従って、編集して下さい。



ワンポイント

ソースの記述について

ソースファイル中にコメントを漢字で記述することができます。またコメントの記述として // を使用が可能です。メニューの[ツール(T)]→[コンパイラオプションの設定(C)] を選択します。機能拡張タブを選択し、右図の通りに設定して下さい。



プログラムの編集

n. 各ソースを編集します。

main.c

```
void  main( void )
{
    /* Start user code. Do not edit comment generated here */
    g_interval_sw = 0;
    g_interval_count = 1;

    TAU_Channel0_Start();
    TAU_Channel1_Start();

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

コードを追加して下さい

Int_user.c

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

UCHAR g_interval_sw = 0;
UCHAR g_onetime_sw = 0;

/* End user code for global definition. 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 */

    if ( g_onetime_sw == 0 )
    {
        g_onetime_sw = 2;
        g_interval_sw++;
        g_interval_sw = g_interval_sw & 7;
    }

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

コードを追加してく下さい

コードを追加して下さい

プログラムの編集

TAU_user.c

```
/***********************************************************************************
**  Global define
***********************************************************************************
*/
/* Start user code for global definition. Do not edit comment generated here */
UINT g_interval_count;
UCHAR g_counter_data[ 8 ] = { 2, 4, 7, 15, 31, 47, 63, 127 };

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

/*-----------------------------------------------------------------------------
**  Abstract:   This function is INTTM00 interrupt service routine.
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTTM00( void )
{
    /* Start user code. Do not edit comment generated here */
    UCHAR inreg, outreg;
    if ( g_interval_count == 0 )
    {
        g_interval_count = g_counter_data[ g_interval_sw & 7 ];
        inreg = P7.6;
        outreg = inreg ^ 1;
        P7.6 = outreg;
        P7.7 = inreg;
    }
    g_interval_count--;
    /* End user code. Do not edit comment generated here */
}

/*-----------------------------------------------------------------------------
**  Abstract:    This function is INTTM01 interrupt service routine.
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTTM01( void )
{
    /* Start user code. Do not edit comment generated here */
    if ( g_onetime_sw != 0 )
    {
        g_onetime_sw--;
    }
    /* End user code. Do not edit comment generated here */
}
```

コードを追加して下さい

コードを追加して下さい

コードを追加して下さい

user_define.h

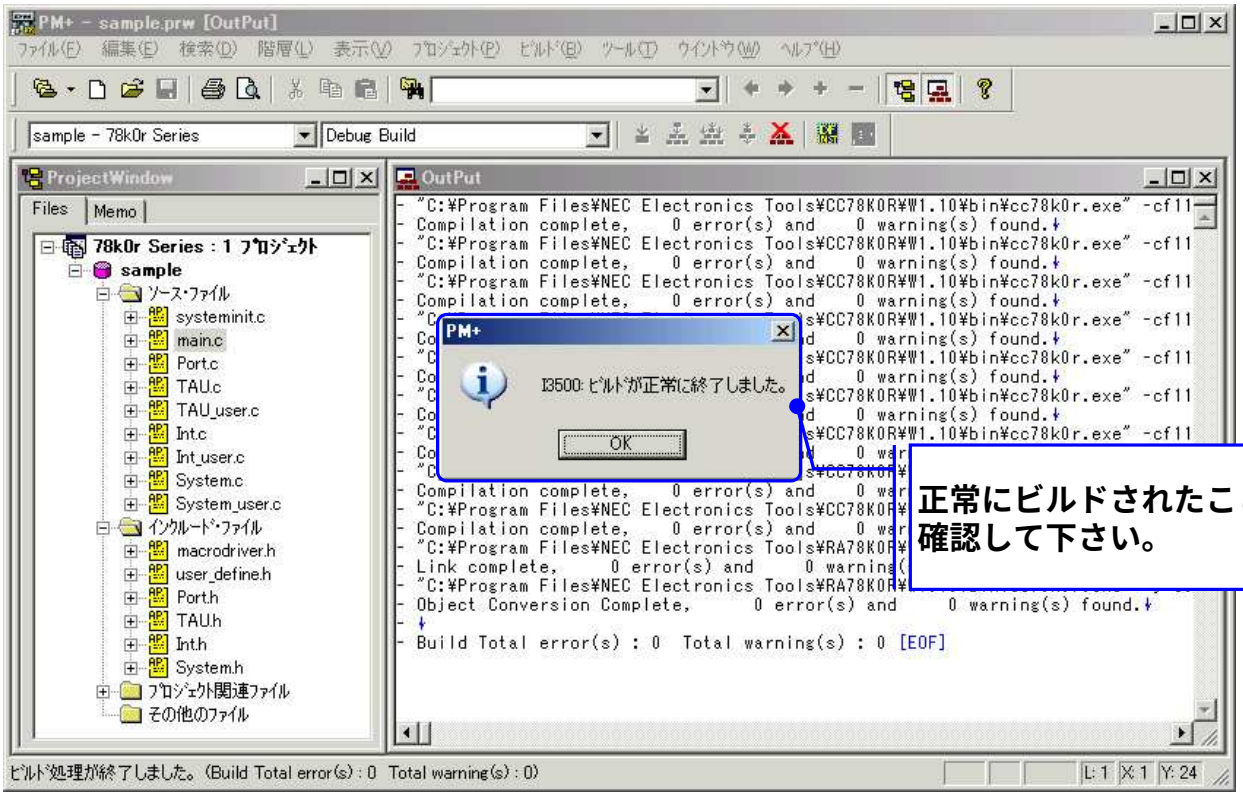
```
/*
***********************************************************************************
**  Macro define
***********************************************************************************
*/

extern UCHAR g_interval_sw;
extern UCHAR g_onetime_sw;
extern UINT  g_interval_count;
```

コードを追加して下さい

プログラムのビルド

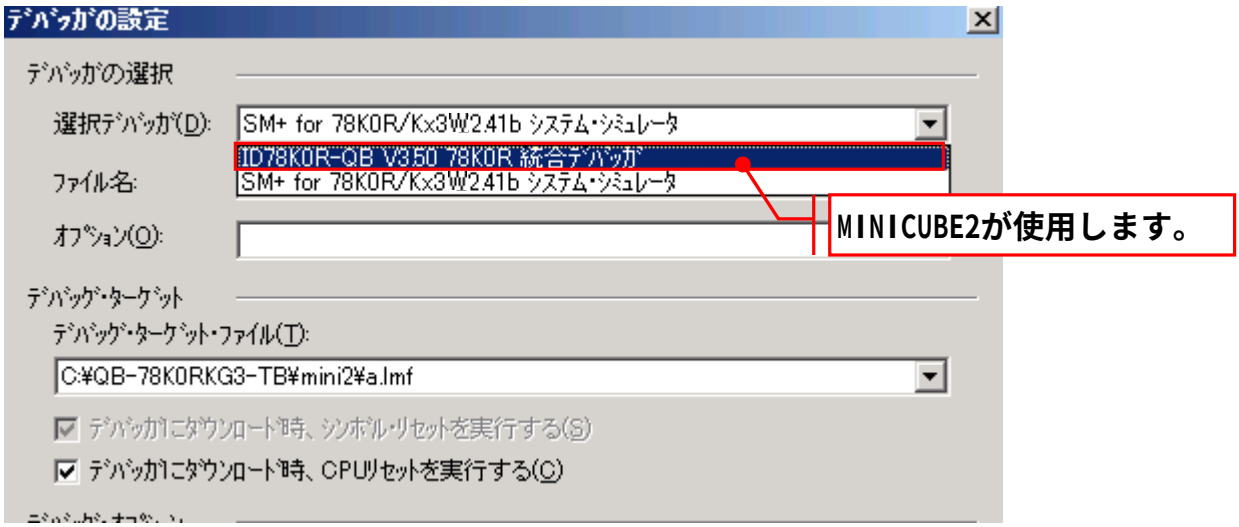
o. プログラムをビルドします。

メニューの[ビルド(B)] → [ビルド(B)] を選択するか、「F7」キーを押下して下さい。
プログラムがコンパイルされオブジェクト・ファイルが作成されます。



p. デバッガを設定します。

メニューの[ツール(T)] → [デバッガの設定(D)...] で「デバッガの設定」ダイアログを開き「ID78K0R-QB Vx.xx」を選択して下さい。V3.50以上を使用してください。



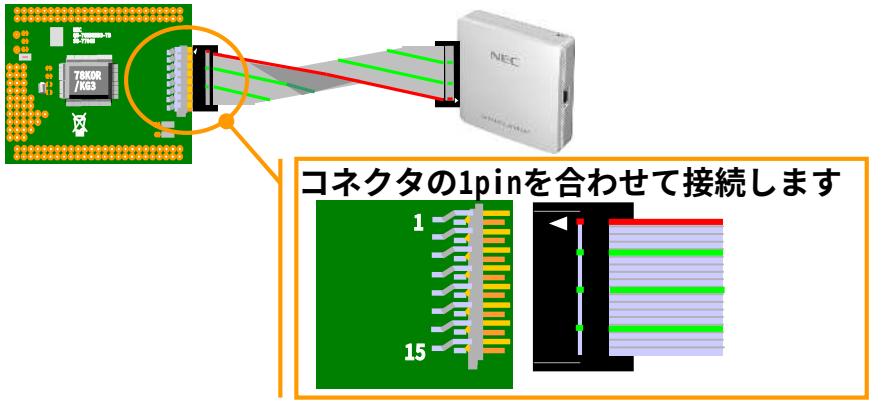
MINICUBE2の接続

④ 実際にQB-78K0RKG3-TB上でプログラムを実行します。

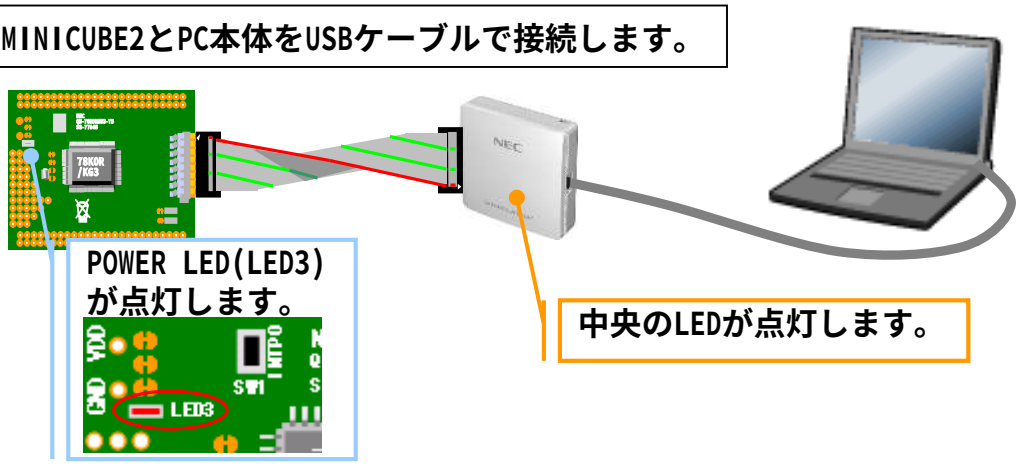
q. PC本体、MINICUBE2、 QB-78K0RKG3-TBを接続します。
接続する順番に注意して接続します。

1.MINICUBE2のスイッチを設定します。

2.QB-78K0RKG3-TBと接続します。



3.MINICUBE2とPC本体をUSBケーブルで接続します。



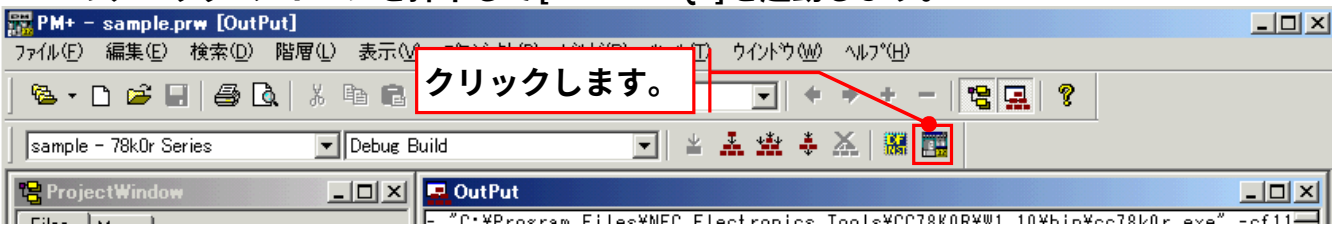
ワンポイント

電源選択SWについて
ターゲット用電源を切り替えます。切り替えは5V供給、3V供給、ターゲット電源使用の3種類です。78K0Rは20MHzの動作電圧が2.7～5.5Vなので5V、3V供給両方で動作します。小さい回路ならMINICUBE2からの電源供給で大丈夫ですが、100mAを超えるような(モータを使うなど)回路は必ずターゲット･システム側で電源を用意して下さい。

デバッガの起動

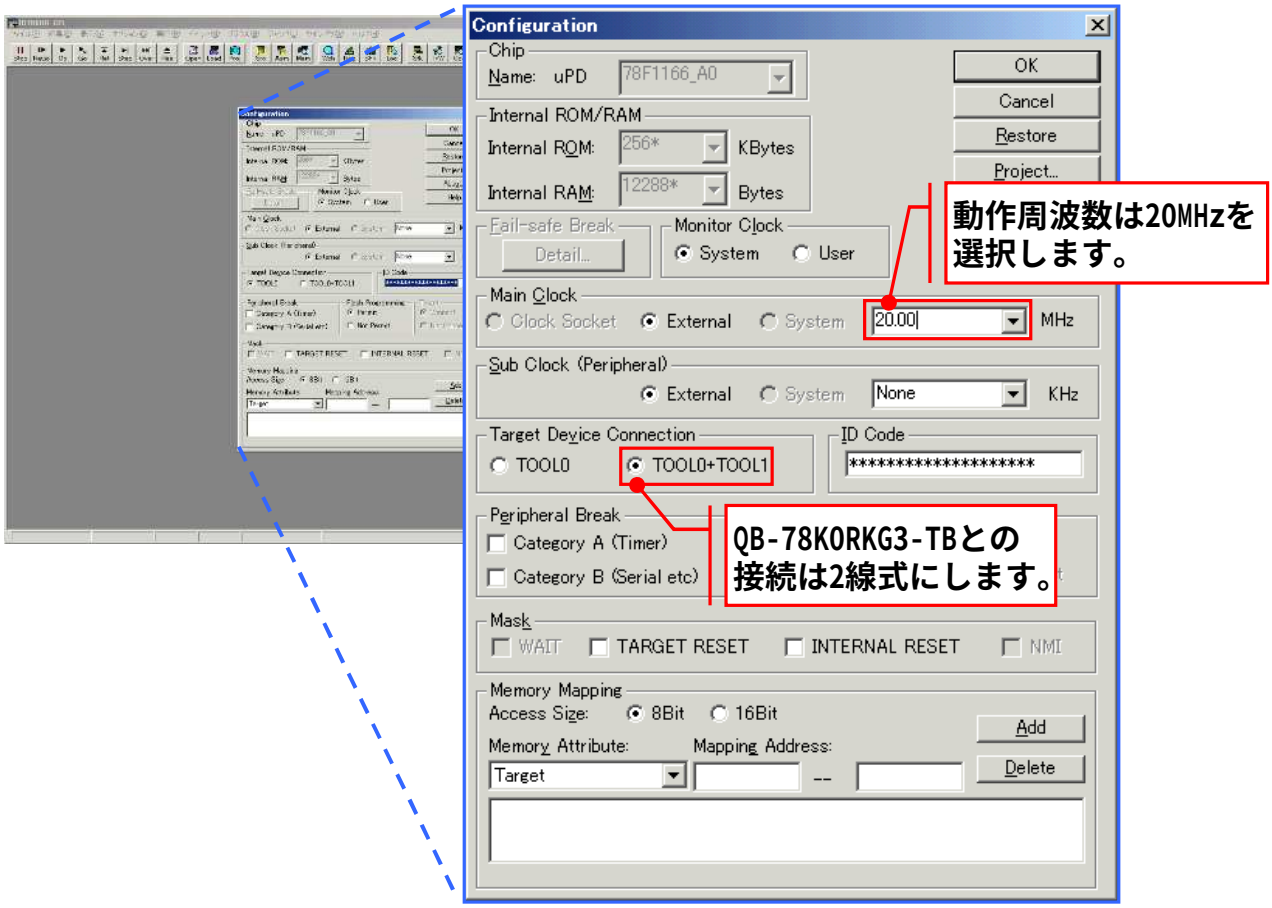
r. デバッガを起動します。

PM+のデバッグ・アイコンを押下して[ID78K0R-QB]を起動します。



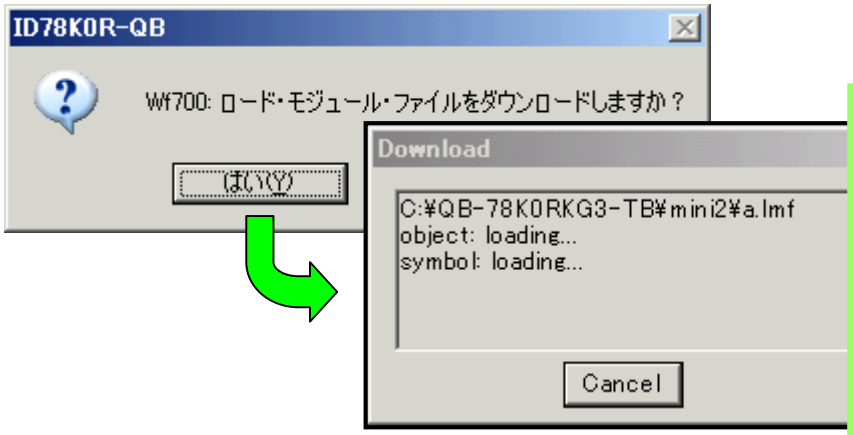
s. デバッガの設定を行います。

デバッガが起動するのでターゲットとの接続を設定し、[OK]を押下します。



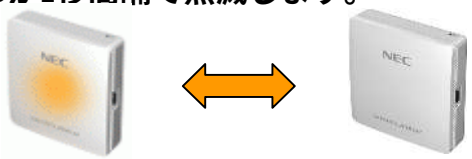
t. プログラムをダウンロードします。

下記ダイアログが表示されるので[はい(Y)]を押下して、プログラムをダウンロードします。



ワンポイント

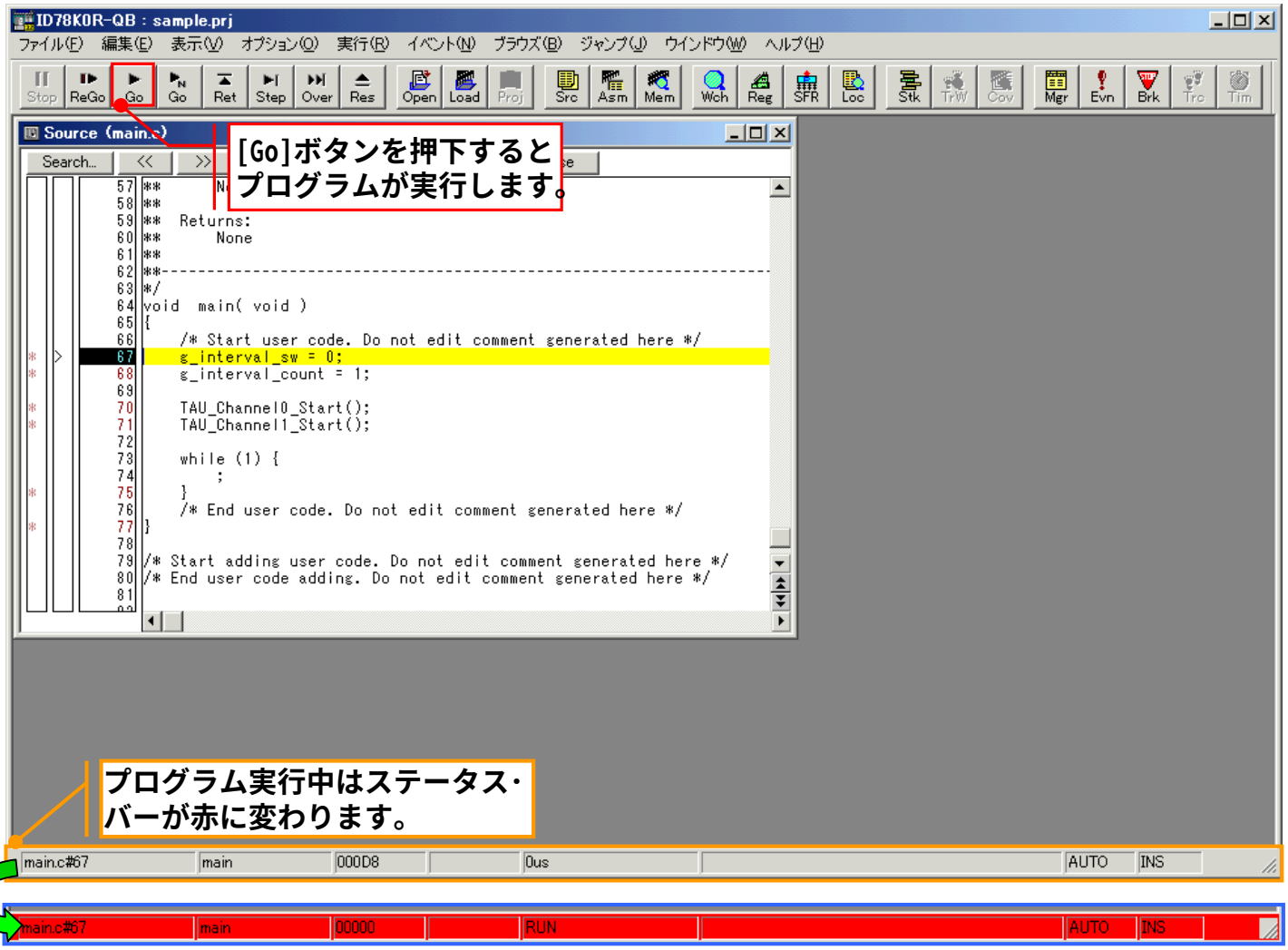
ダウンロードの時間は2～3秒ほどです。
また、ダウンロード中はMINICUBE2のLEDが1秒間隔で点滅します。

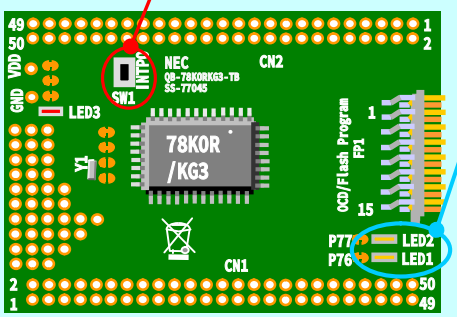


プログラムの実行

u. プログラムを実行します。

プログラムがロードされ、ソースファイルが表示されたら、プログラムを実行します。





SWを押下するとLEDの点滅速度が変化します。

LEDが点滅します。最初は高速に点滅するため両方点灯しているように見えますがSWを押すと点滅がゆっくりになります。

ターゲット・ボード上の動作

プログラムは動いたでしょうか？
思ったより簡単にマイコンを動かすことができたと思います。次ページよりサンプル・プログラムについて説明します。プログラムの動作を理解して、自由にプログラムを改造して下さい。更にマイコンへの理解が深まるはずです。

プログラムの説明

マイコンについて基本的な知識は弊社のWEBページ「マイコンe-Learning」で学習できます。
下記URLよりアクセスして下さい。
http://www.necel.com/micro/ja/eLearning/
これよりサンプル･プログラムについて説明します。学習予定時間は30分です。

1. サンプル･プログラムの構成

大きく分けて[メイン]、[タイマ割り込み]、[外部割り込み]の3つに分類できます。

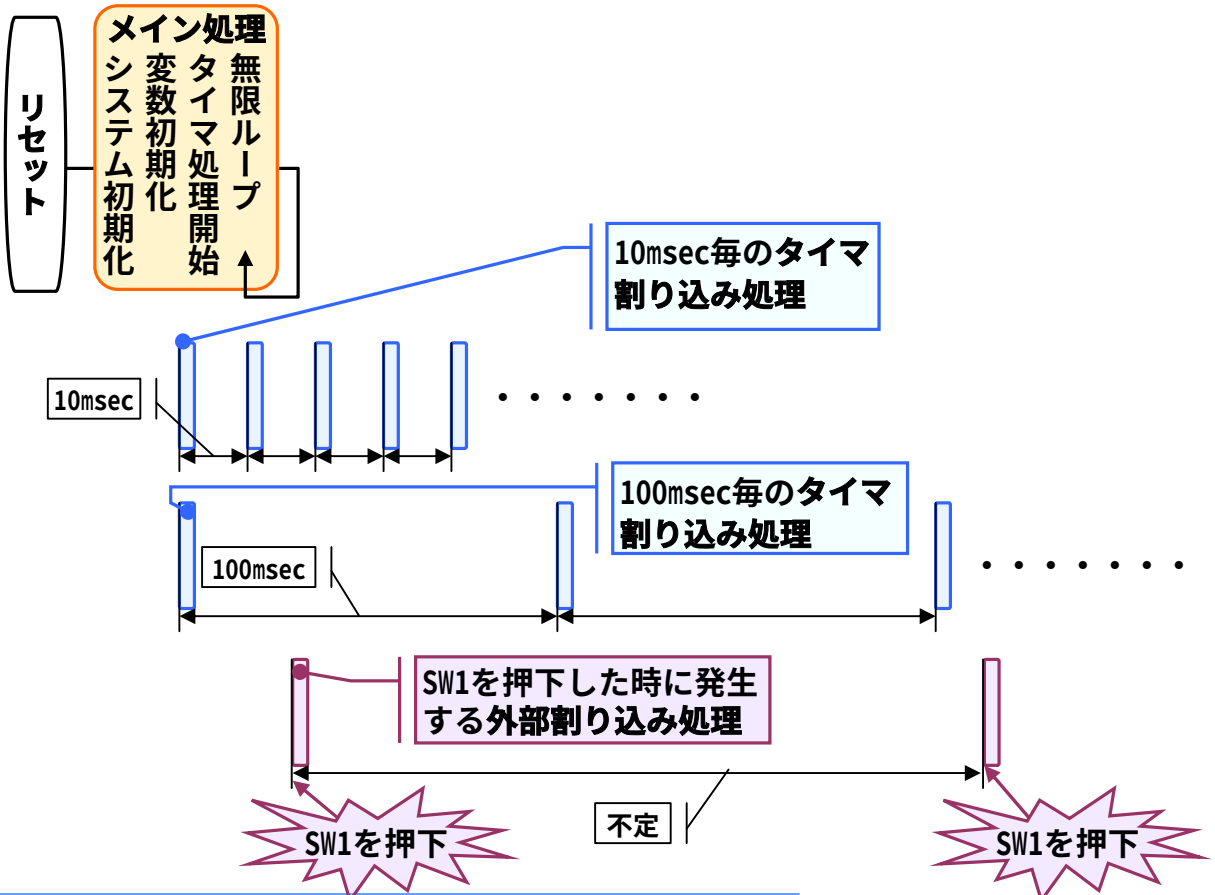
メイン処理:
マイコンがリセットされてから動作する処理です。変数初期化、タイマの開始を行った後は無限ループになります。

タイマ割り込み処理:
10msec毎と100msec毎に呼ばれる2つの処理があります。10msecの割り込み処理はLEDの点灯処理、100msecの割り込み処理はチャタリング防止処理です。

外部割り込み処理:
QB-78K0RKG3-TB上のSW1が押された時に呼ばれる割り込み処理です。回路にチャタリング防止が施されていないのでソフトウエアでチャタリング防止を行います。

2. プログラムの時間的な流れ

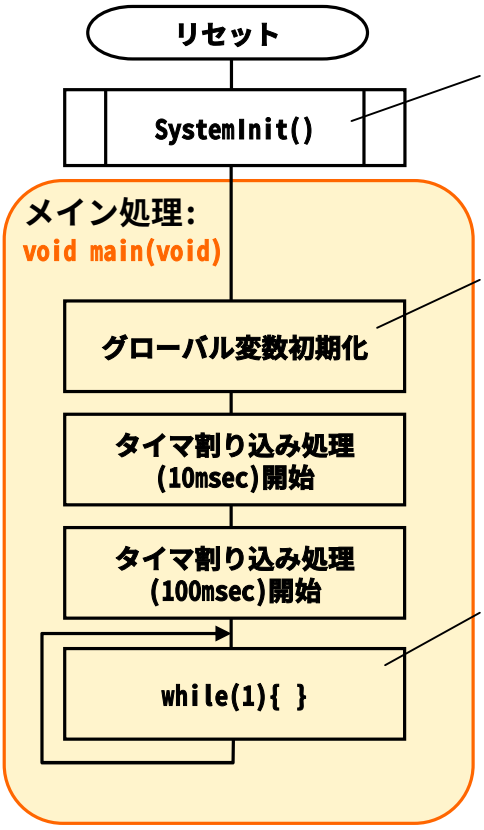
[メイン]は初期化とタイマ処理開始だけの処理です。プログラムは[タイマ割り込み]、[外部割り込み]で動作します。



プログラムの説明

3．処理の説明

[メイン]、[タイマ割り込み]、[外部割り込み]、それぞれの処理について説明します。



ハードウエアを初期化します。クロック、ポート、外部割り込み、タイマについてマイコンを設定します。 この処理はApplilet2が自動的に生成します。

グローバル変数 g_interval_sw, g_interval_count を初期化します。

処理はタイマ割り込み、外部割り込みで行われるのでメインは無限ループします。

全体的な処理の説明

10msec毎に呼ばれる関数内でg_interval_countを-1して、それが0になった場合にLEDを点滅させる処理をしています。最初はg_interval_countが1なので2回に1回LEDが点滅します。するとP76,P77に接続されたLEDが20msec毎に交互に点滅します。はじめは20msec毎ですが、外部SWを押下することによって点滅速度が遅くなります。点滅速度は 20msec，40msec，80msec，160msec，320msec，480msec，640msec，1280msecを繰り返します。

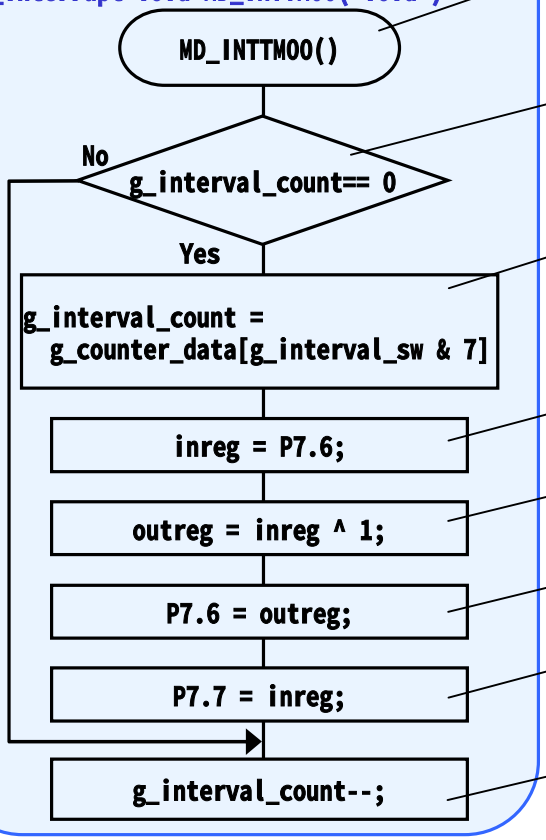
グローバル変数の説明

g_interval_sw:	外部SWを押下すると変化する。0～7の値になる。
g_onetime_sw:	この値が0の時に外部割り込み処理を有効とします。
g_interval_count:	10msec毎に -1されるカウンタ。0の時にLED点灯を変更します。
g_counter_data[8]:	点灯速度を決めるデータ。8段階のスピードを設定します。

プログラムの説明

タイマ割り込み処理(10msec毎):

```
__interrupt void MD_INTTM00( void )
```



10msec毎に発生するインターバル･タイマです。
10msec毎に、この関数が呼ばれます。

g_interval_countをチェックします。

次の点灯時間を決めます。点灯時間は外部割り込みSW(g_interval_sw)の値を元にg_counter_dataから参照します。

内部変数 inregにLED1(ポート7のbit6接続)の値を読み込みます。

内部変数 outregにinregを反転(xor)させた値を設定します。

LED1(ポート7のbit6接続)に outregの値を入れます。

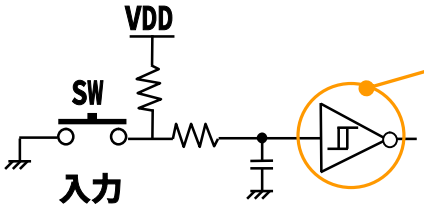
LED2(ポート7のbit7接続)に inregの値を入れます。

g_interval_countの値を減らします。

ワンポイント

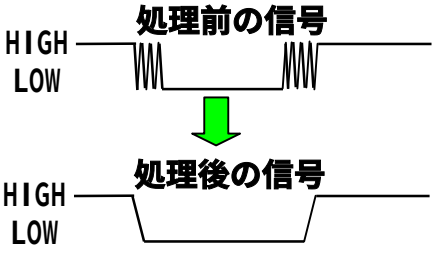
チャタリングについて

SWを押した時、実際には機械的な接点では瞬間的にON<->OFFを繰り返します。これを「チャタリング」と言います。QB-78K0RKG3-TBでは、ハードウエア回路にはチャタリング防止を行っていないのでチャタリング防止をソフトウエアで行う必要があります。ハードウエアで、チャタリング防止する方法としてシュミットトリガインバータ(74HC14など)を使うのが一般的です。この方法はノイズ除去にも使われます。



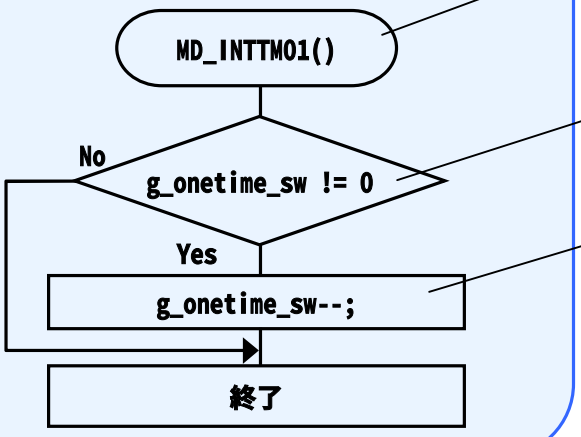
74HC14などのシュミットトリガインバータ

出力



プログラムの説明

タイマ割り込み処理(100msec毎):
__interrupt void MD_INTTM01(void)

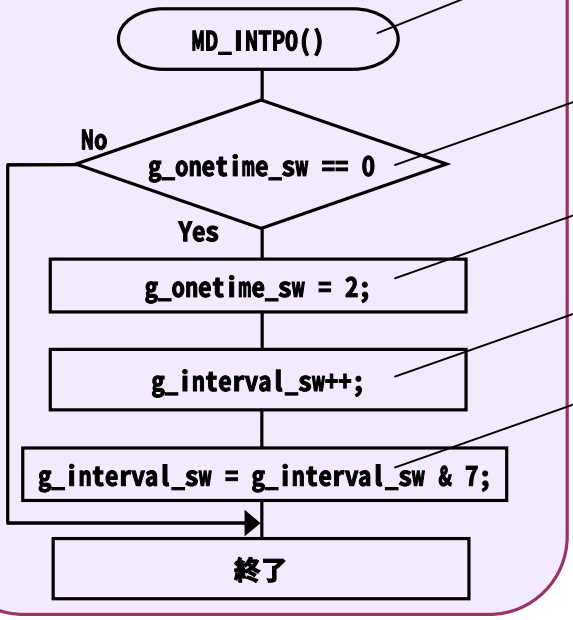


100msec毎に発生するインターバル･タイマです。一度押したSWを200msec以内は無視するための処理です。

g_onetime_swをチェックします。

g_onetime_swの値を減らします。

外部割り込み処理:
__interrupt void MD_INTP0(void)



SWが押下された時に呼ばれます。

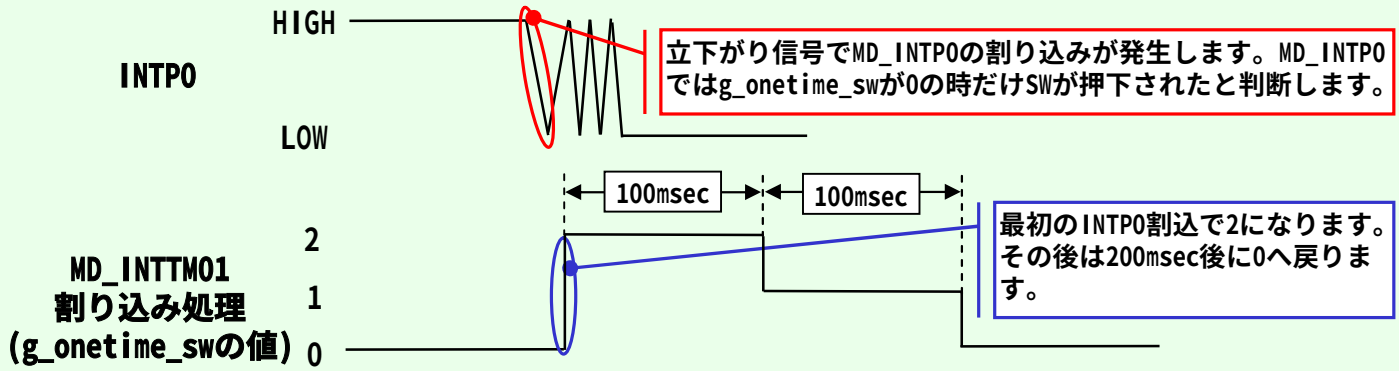
g_onetime_swをチェックします。

g_onetime_swに値をセットします。

g_interval_swを増やします。LEDの点灯時間を参照する時に使用します。

LED点灯時間のデータは8個なので参照する値を調整します。

外部SWが押下されると g_onetime_sw に2が代入されます。g_onetime_swは100msec毎に-1されます。1度外部SWが押下されるとg_onetime_swが0になるまで200msecかかるわけです。MD_INTP0()の処理は、このg_onetime_swが0でないとLED点滅のタイミングを変更しません。ゆえにチャタリング発生し短い時間に何度MD_INTP0()が呼ばれても200msec以上の時間を空けないとLED点滅タイミングは変更されません。



立下がり信号でMD_INTP0の割り込みが発生します。MD_INTP0ではg_onetime_swが0の時だけSWが押下されたと判断します。

最初のINTP0割込で2になります。その後は200msec後に0へ戻ります。

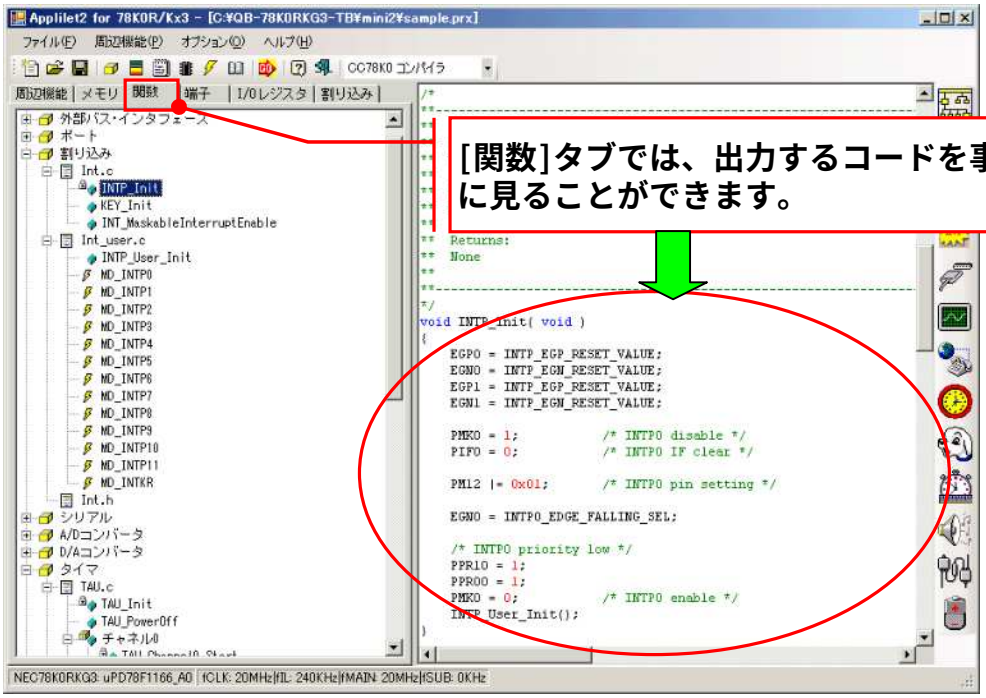
プログラムの説明

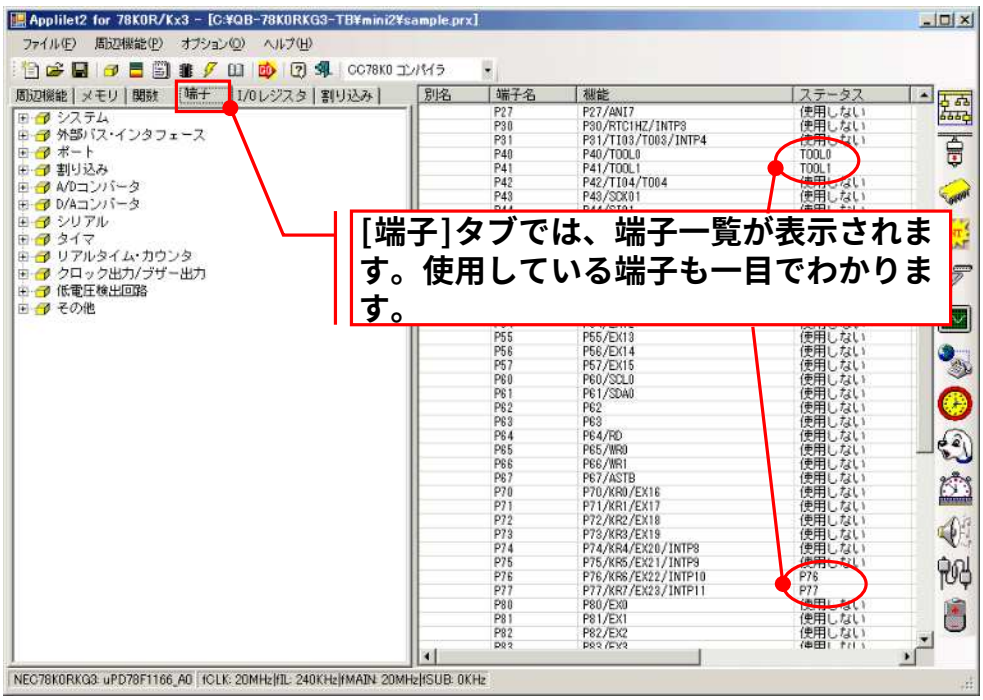
プログラムの説明は以上です。追加するコードの小さい事に驚いたのではないでしょうか？デバイスドライバ･コンフィギュレータ「Applilet2」を使えば、誰でも手軽にマイコンの開発ができる事でしょう。Appliletの情報については、下記URLよりアクセスして下さい。
http://www.necel.com/micro/ja/development/asia/applilet2/

ワンポイント

Applilet2の機能ついて

タブ切り替えを行うことで、機能が変わります。詳細は[ヘルプ(H)] → [ユーザーズ･ガイド(G)...]を参照して下さい。





プログラムの作り方

ここでは、プログラムを作成する時の諸注意を挙げます。ここにあげる内容が全てではありませんが、プログラムを作る上での参考にして下さい。また、有名なコーディング規約として、自動車関連ソフトウェアの業界団体(MISRA)が策定した組込みC言語用ガイドライン「MISRA-C」があります。参考にして下さい。

設計時に考えること

[マイコン資源の見積]

入出力ポートの使い方、タイマ割り込み、などハードウエア資源とソフトウエアの資源について充分に見積って下さい。例えばA/D変換を行うタイミングとして100us毎で処理するのか10ms毎でも充分なのか、処理方法を考えて下さい。

[ソース･ファイルの構成]

Applilet2が必ず生成するソース一覧

systeminit.c	マイコン初期化
System.c / h	マイコン初期化
System_user.c	ユーザー追加用初期化関数
macrodriver.h	Applilet2用定数定義
user_define.h	ユーザー追加用定義ファイル
main.c	メイン関数

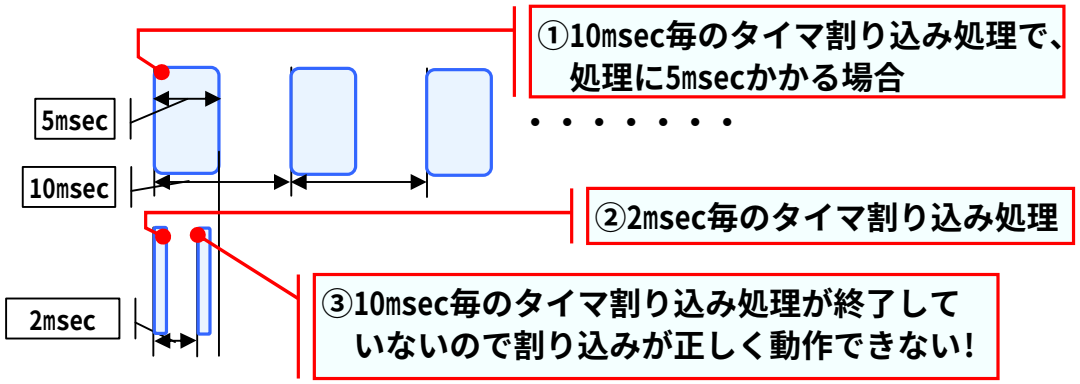
その他にマイコン周辺機能を使う場合（ポート、外部割り込み、タイマを追加した場合）

Port.c /h	入出力ポート定義
Int.c / h	外部割り込み
Int_user.c	ユーザー追加用外部割り込みハンドラ
TAU.c / h	タイマ処理
TAU_user.c	ユーザー追加用タイマ割り込みハンドラ

※ Applilet2が必要なソースを生成しますので、基本的には xxxx_user.c にプログラムを追加すれば大丈夫です。

[割り込みハンドラ内では処理速度を優先に考える]

割り込みハンドラ処理中は、さらに優先度の高い割り込み処理が入らない限り、他の処理は停止してしまいます。割り込みハンドラは、処理速度を優先させて下さい。そのため処理としては、割り込み要因を解除し、割り込み処理自体はmainから制御します。mainへ処理を渡すために割り込みハンドラ内ではグローバル変数などのフラグを立てて、それをmainで処理するのが望ましいです。割り込み処理を考える時は、**割り込みの優先度**にも注意して設計して下さい。



プログラムの作り方

コーディング時に考えること

[名前の付け方を考慮する]

define、マクロ、変数などの命名をきちんと定義すべきです。
例えば
- define、マクロ名は全て大文字にする。
- グローバル変数には[g_]、グローバルポインタ変数には[gp_]を接頭語としてつける。

また enum，構造体の定義も解るように命名定義しましょう。
命名も重要なコーディングと考えて下さい。

[関数の作り方で注意したい事]

・入力と出力は1箇所にする。
なぜ、入力と出力は1箇所にするのか。例えば、こんな関数があったとします。

```
void function( UCHAR para )
{
    size_t *size;
    size = malloc(100);
    if ( size == NULL )
        return;
    if ( para == 0 )
        return;
        :
        :
    free( size );
}
```

上の行のreturnは、正しく動作するがここでのreturnは領域確保した*sizeを開放していないのでメモリリークが発生する。

```
void function( UCHAR para )
{
    size_t *size;
    size = malloc(100);
    if ( size != NULL )
    {
        if ( para != 0 )
        {
            :
            :
        }
        free( size );
    }
}
```

階層は深くなりますが、メモリリークを防ぐ意識の書き方になります。

他の利点としてデバッグ時は、入口と出口だけチェックだけでよいのでデバッグ効率があがります。

・多くても1つの関数を100行程度にする。
例えば、車の構成を考えて下さい。車は小さい部品で構成されているとは思いませんか？小さい部品一つ一つがテストされ組み上がっています。プログラムも同様です。全体で1万行のプログラムに1000行を超えるような関数があるとすれば、かなりアンバランスです。関数一つにしても可読性、移植性を考えるべきです。

・ハードウエアのアクセスは最小限にする。
下記の例は、一見問題無いように思われますが、潜在的なバグがあります。

```
void function()
{
    UCHAR a = 0;
    if ( P4.0 == 0 )
    {
        a = 1;
    }
    else if ( P4.0 == 1 )
    {
        a = 2;
    }
}
```

このif命令を実行している時点ではP4.0=1の値で次のeles if命令を実行する時にはP4.0=0の値の場合、どのケースにも該当しないでa=0の場合がある。

```
void function()
{
    UCHAR a = 0, port = P4.0;
    if ( port == 0 )
    {
        a = 1;
    }
    else
    {
        a = 2;
    }
}
```

ハードウエアのアクセスを最小にし、更にロジック的にありえないelse if命令を消去しました。

プログラムの作り方

コーディング時に考えること

[ソースを見やすくするために]

・直値を書かない。

数値をソース中に書く場合に、直接な値を書かないようにします。
値を直接書くと修正する箇所が複数発生するため、修正ミスが発生する確率が高くなります。

```
void function( void )
{
    UCHAR lcd_max;

    lcd_max = 8;
            :
            :
```

直値を書くと、この値を使った部分のソースを全て編集しなければならなくなります。

```
#include “user_define.h”

void function( void )
{
    UCHAR lcd_max;

    lcd_max = LCDMAX;
            :
            :
```

```
#define LCDMAX 8
```

“user_define.h”に定義する。

LCDMAXの値が変わっても”user_define.h”だけ修正すればよい。

・関数の引数宣言について

通常、関数引数を扱う場合、引数で値を返す場合はポインタ渡しを行います。関数に値を渡すだけの場合には、ポインタ渡しを使用をしないようにします。このように、明確に書き方を分けておけば、関数の宣言だけで引数の意味の判断が可能です。

```
void set_function( UCHAR *para )
{
    UCHAR data = *para;
            :
            :
}

void get_function( UCHAR *para )
{
    UCHAR data;
            :
            :
    *para = data;
}
```

```
void set_function( UCHAR para )
{
    UCHAR data = para;
            :
            :
}

void get_function( UCHAR *para )
{
    UCHAR data;
            :
            :
    *para = data;
}
```

関数で値を取得する時だけパラメータにポインタ渡しを行います。明確に使い分けることで関数宣言を見るだけで、その関数は値を必要とするのか、値を取得するのかが分かります。

プログラムの作り方

マイコン開発特有の注意点

[マイコン資源関係]

・固定データにはconstを書く。
単に定義したデータはRAMに確保される場合があります。マイコンのRAM領域はROMに比べて少ないので変更しない参照するだけのデータには、必ずconstをつけるようにします。

```
const UCHAR g_font_numeric[] =
{ /* font data 0 - 9 */
  0x00, 0x0E, 0x11, 0x13, 0x15, 0x19, 0x11, 0x0E    /* 0 */
, 0x00, 0x04, 0x0C, 0x04, 0x04, 0x04, 0x04, 0x0E    /* 1 */
, 0x00, 0x0E, 0x11, 0x01, 0x02, 0x04, 0x08, 0x1F    /* 2 */
, 0x00, 0x0E, 0x11, 0x01, 0x06, 0x01, 0x11, 0x0E    /* 3 */
, 0x00, 0x02, 0x06, 0x0A, 0x12, 0x1F, 0x02, 0x02    /* 4 */
, 0x00, 0x1F, 0x10, 0x10, 0x1E, 0x01, 0x01, 0x1E    /* 5 */
, 0x00, 0x0F, 0x10, 0x10, 0x1E, 0x11, 0x11, 0x0E    /* 6 */
, 0x00, 0x1F, 0x01, 0x02, 0x04, 0x04, 0x04, 0x04    /* 7 */
, 0x00, 0x0E, 0x11, 0x11, 0x1E, 0x11, 0x11, 0x0E    /* 8 */
, 0x00, 0x0E, 0x11, 0x11, 0x0F, 0x01, 0x01, 0x0E    /* 9 */
};
```

この様なフォントデータは、変更しないのでconst宣言してROM領域に置く。

・レジスタから値を取得する際はvolatileをつける。
volatile(ボラタイル)とは最適化しないという宣言です。
例えば、レジスタなどハードウエアからの情報を取得してグローバル変数などに代入する場合にvolatileをつけます。

Applilet2でシリアルを使用した時のソースコード「Serial.c」

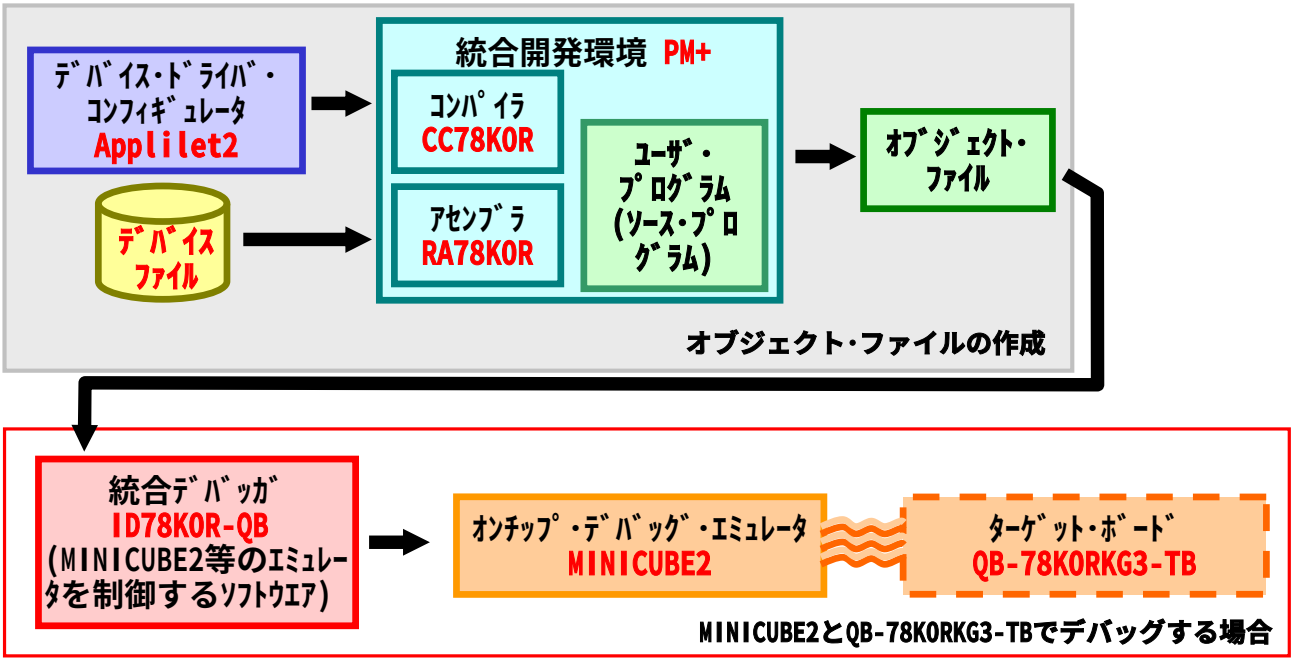
```
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "Serial.h"
/* Start user code for include definition. Do not edit comment generated here */
/* End user code for include definition. Do not edit comment generated here */

/*
***********************************************************************************
**  Global define
***********************************************************************************
*/
volatile UCHAR*      gpUart0TxAddress;                  /* uart0 transmit buffer address */
volatile USHORT      gUart0TxCnt;                       /* uart0 transmit data number */
volatile UCHAR*      gpUart0RxAddress;                  /* uart0 receive buffer address */
volatile USHORT      gUart0RxCnt;                       /* uart0 receive data number */
volatile USHORT      gUart0RxLen;                       /* uart0 receive data length */
/* Start user code for global definition. Do not edit comment generated here */
/* End user code for global definition. Do not edit comment generated here */
```

基本的なプログラムの作り方については以上です。弊社WEBページにはサンプルプログラムも掲載しています。下記URLも参考にして下さい。
http://www.necel.com/micro/ja/designsupports/sampleprogram/

デバッグの基本

[デバッグしたい]の章では、ハードウエアを使ってデバッグする時に必要な基本項目を学習します。
[動かしてみよう]の章で使用したシステム構成(MINICUBE2 + QB-78K0RKG3-TB)で統合デバッガ(ID78K0R-QB)の使い方を学習します。学習時間は30分です。

この章でのデバッグ時のシステム構成



次ページより下記の項目順に説明します

- デバッガの基本画面
- ブレークポイントの設定
- ステップ実行
- 変数表示
- メモリ表示
- レジスタ表示

ワンポイント

統合デバッガ(ID78K0R-QB)について
ここで説明するデバッガの機能は、オンチップ･デバッグ･エミュレータ(MINICUBE2)を使った場合です。
もっと高機能なハードウエア･エミュレータ「IECUBE」を使えば豊富なデバッグ機能(トレース機能、イベント間時間測定、カバレッジ機能、疑似エミュレーション機能)があります。
「IECUBE」詳細については下記URLにアクセスして下さい。
http://www.necel.com/micro/ja/development/asia/78k0rkx3/iecube.html

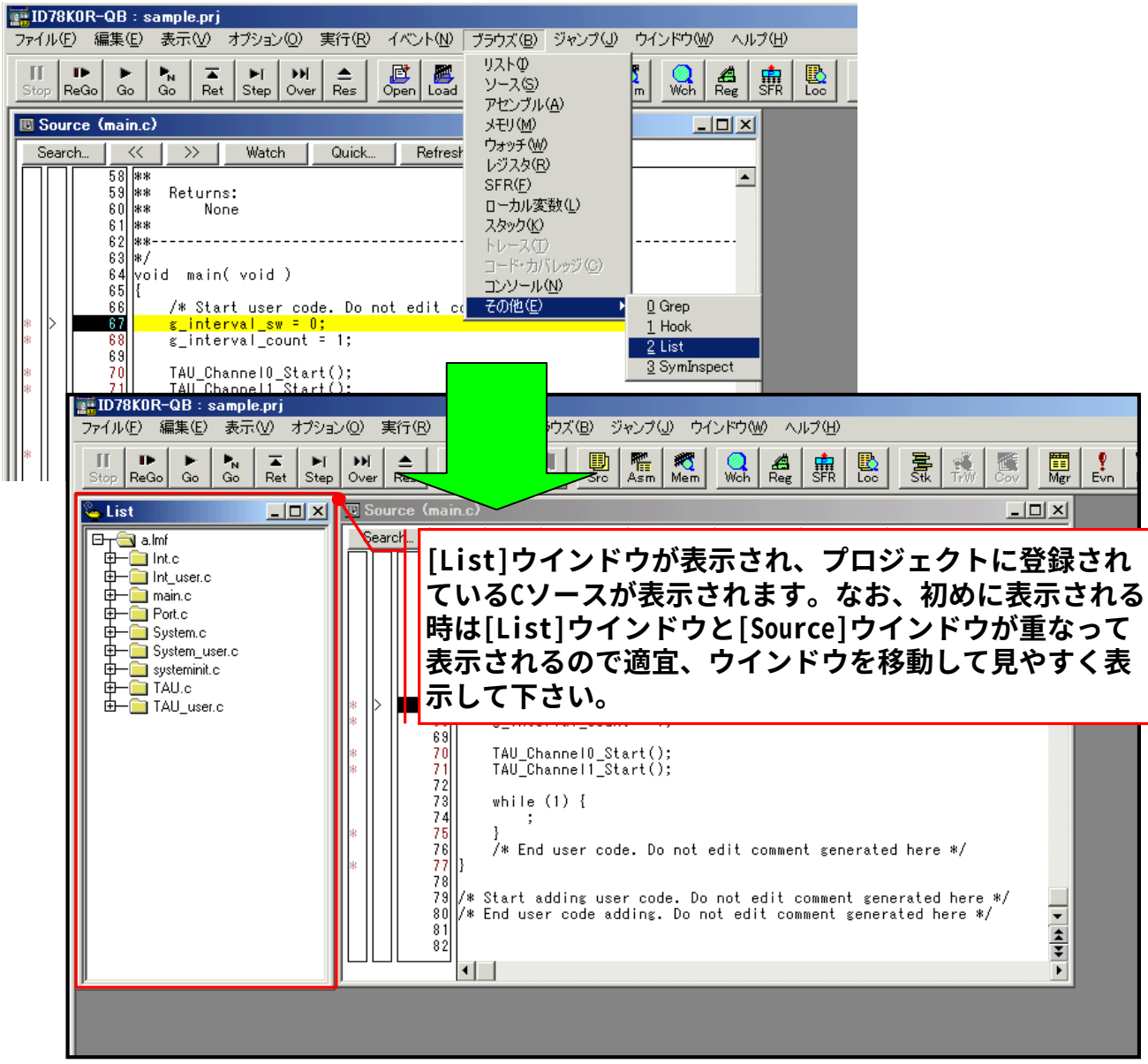
デバッガの基本画面

a. デバッガを起動します。

[動かしてみよう] の章 [デバッガの起動] を参考にして下さい。2回目以降に起動する場合、プロジェクトが保存されていれば、自動的にオブジェクト･ファイルがダウンロードされます。

b. ファイル一覧を表示します。

メニューの[ブラウズ(B)] → [その他(E)] → [2 LIST]を選択して下さい。プロジェクトに登録されているファイル一覧が表示されます。



ワンポイント

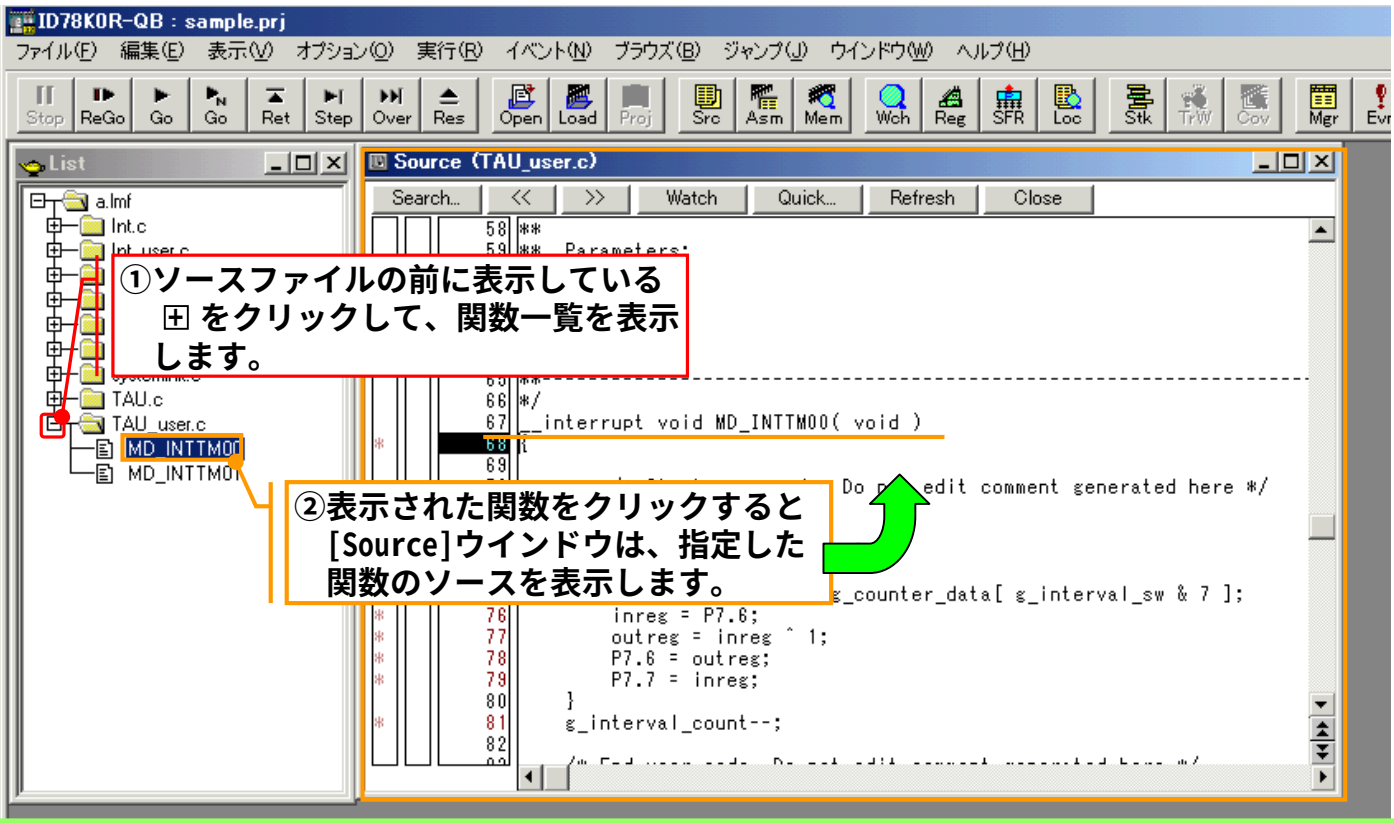
デバッガ(ID78K0R-QB)の環境ファイルについて

設定したデバッガの環境は、「プロジェクト名.pri」ファイルに保存されます。サンプル･プログラムのプロジェクト名は[sample]なので、「sample.pri」へ保存されます。デバッガの環境を複数持ちたい時などは、この「プロジェクト名.pri」を名前を変更して保存して下さい。また、2回目以降のデバッガの起動は「.pri」ファイルを使用するので、プログラムのダウンロードも自動的に行われます。

デバッガの基本画面

c. ソース上の、ある関数に移動したい時。

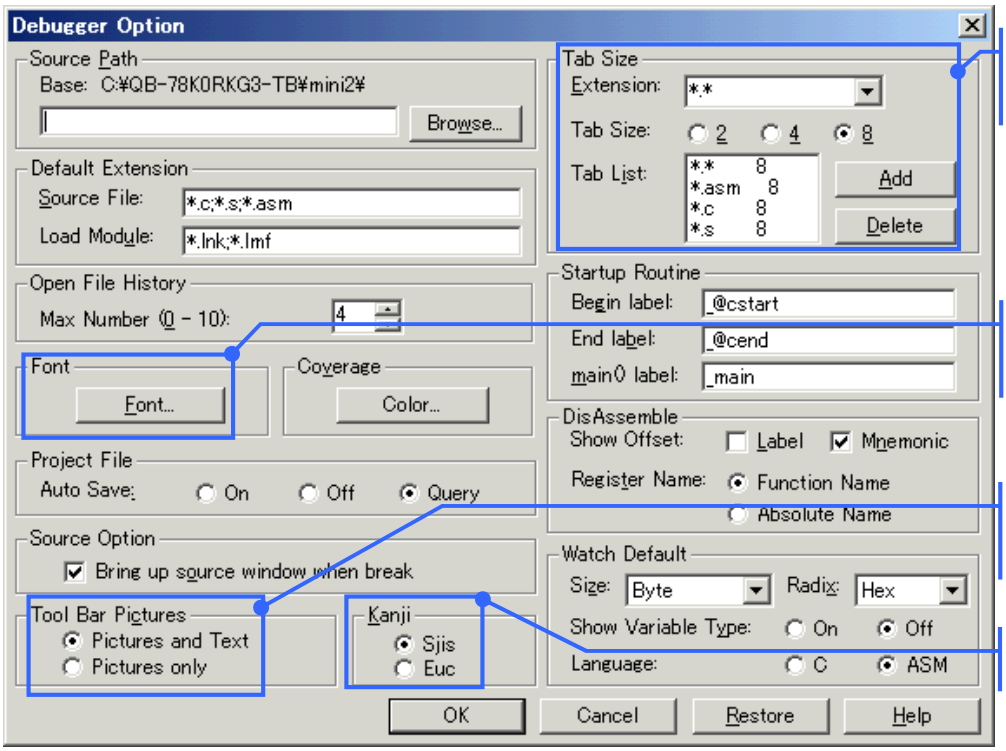
[List]ウインドウのソースファイルをクリックして関数一覧を表示します。関数が表示されたらその関数をクリックすると、[Source]ウインドウの表示が変わります。



ワンポイント

デバッガ(ID78K0R-QB)のオプションについて

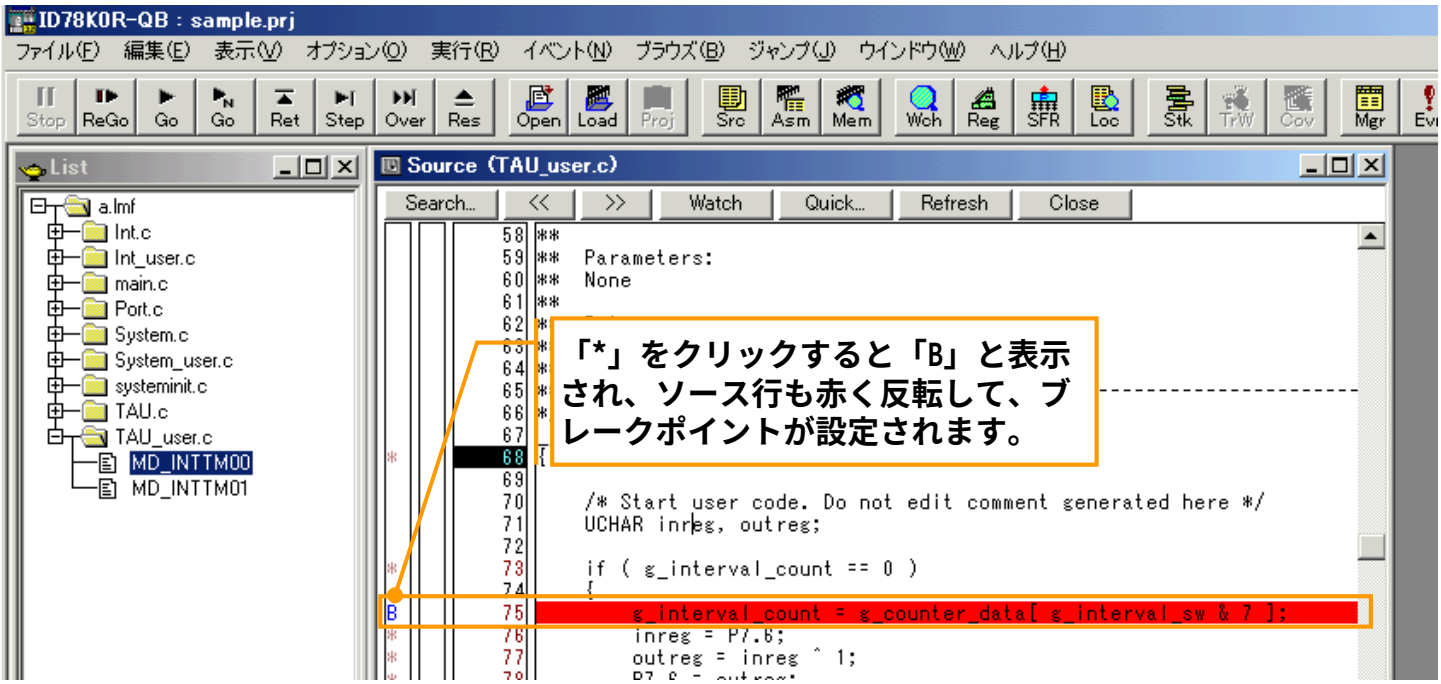
メニューの[オプション(O)] → [デバッガ･オプション(G)...] で表示の変更ができます。[Debugger Option]ダイアログが表示されますので、各種設定して下さい。



ブレークポイントの設定

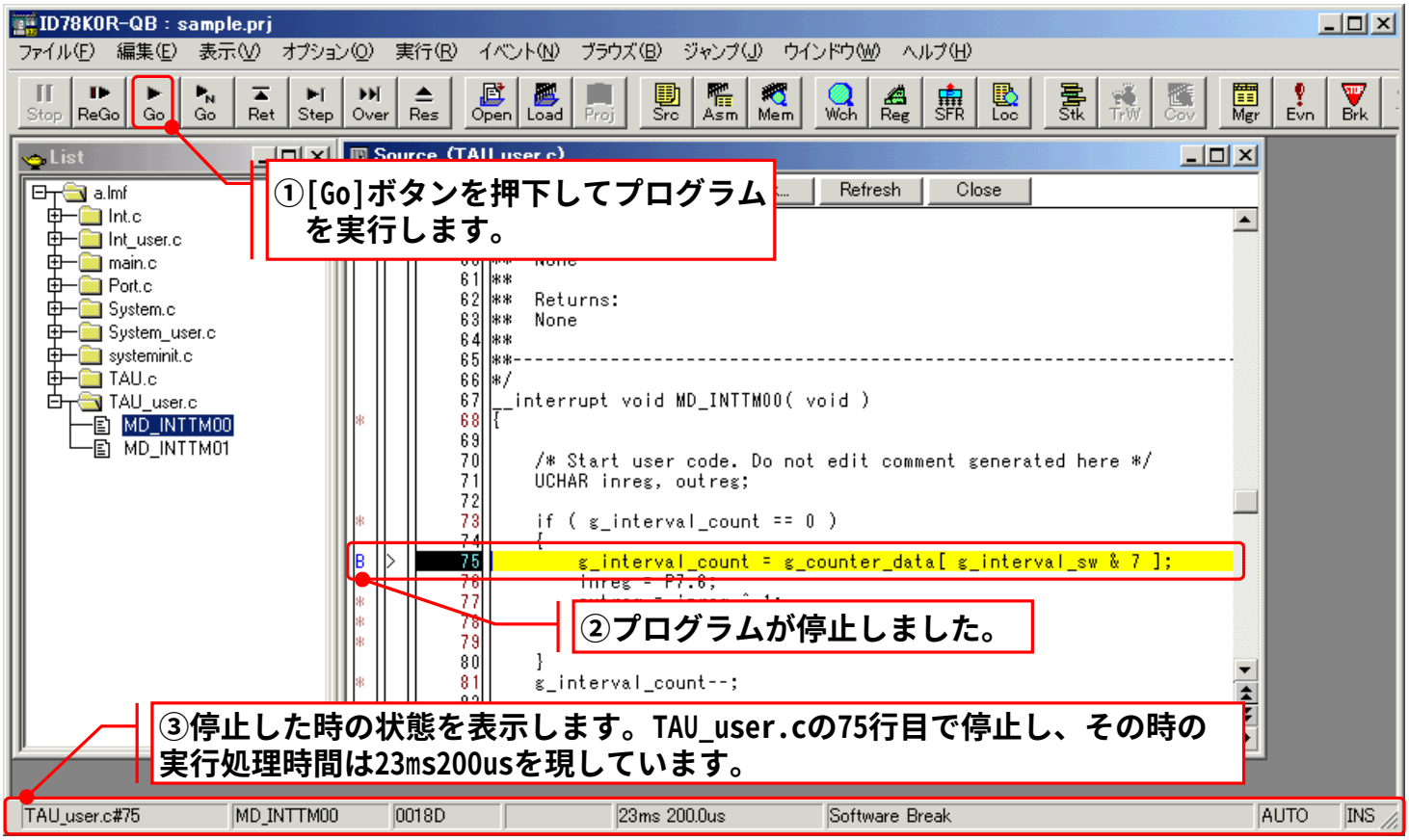
d. ブレークポイントを設定したい画面を表示します。

ソースを表示させ、行番号前の「*」をクリックします。すると、クリックした行が赤に反転表示され、ブレークポイントが設定されます。



e. 実行してブレークするか確認します。

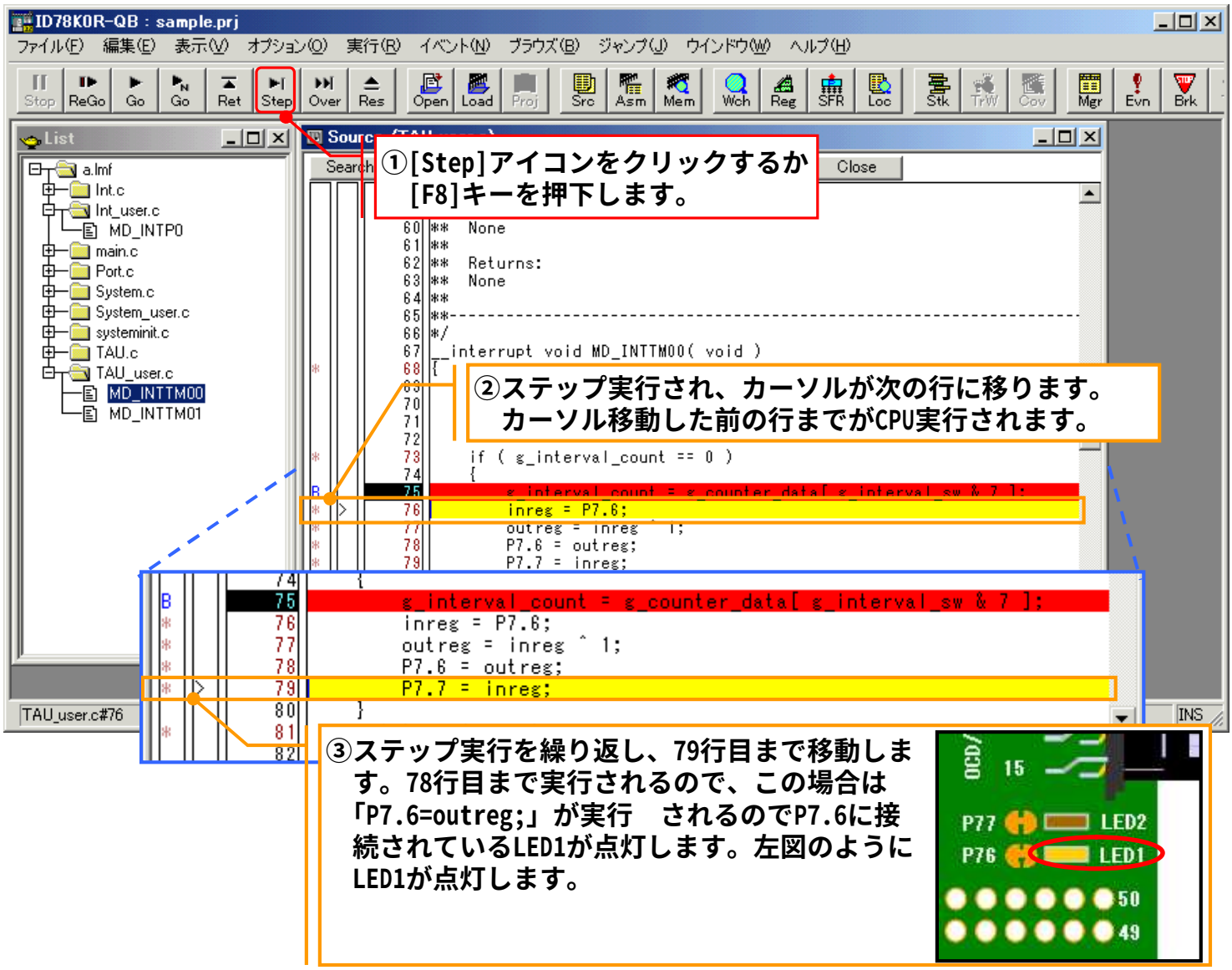
[Go]ボタンを押下して実行し、プログラムが停止することを確認します。



ステップ実行

f. ブレークした場所からステップ実行します。

停止している時に、[F8]キーを押下または、[Step]アイコンをクリックしてステップ実行します。
ステップ実行して 79行目までカーソルを移動して下さい。



ワンポイント

ステップ実行の種類について

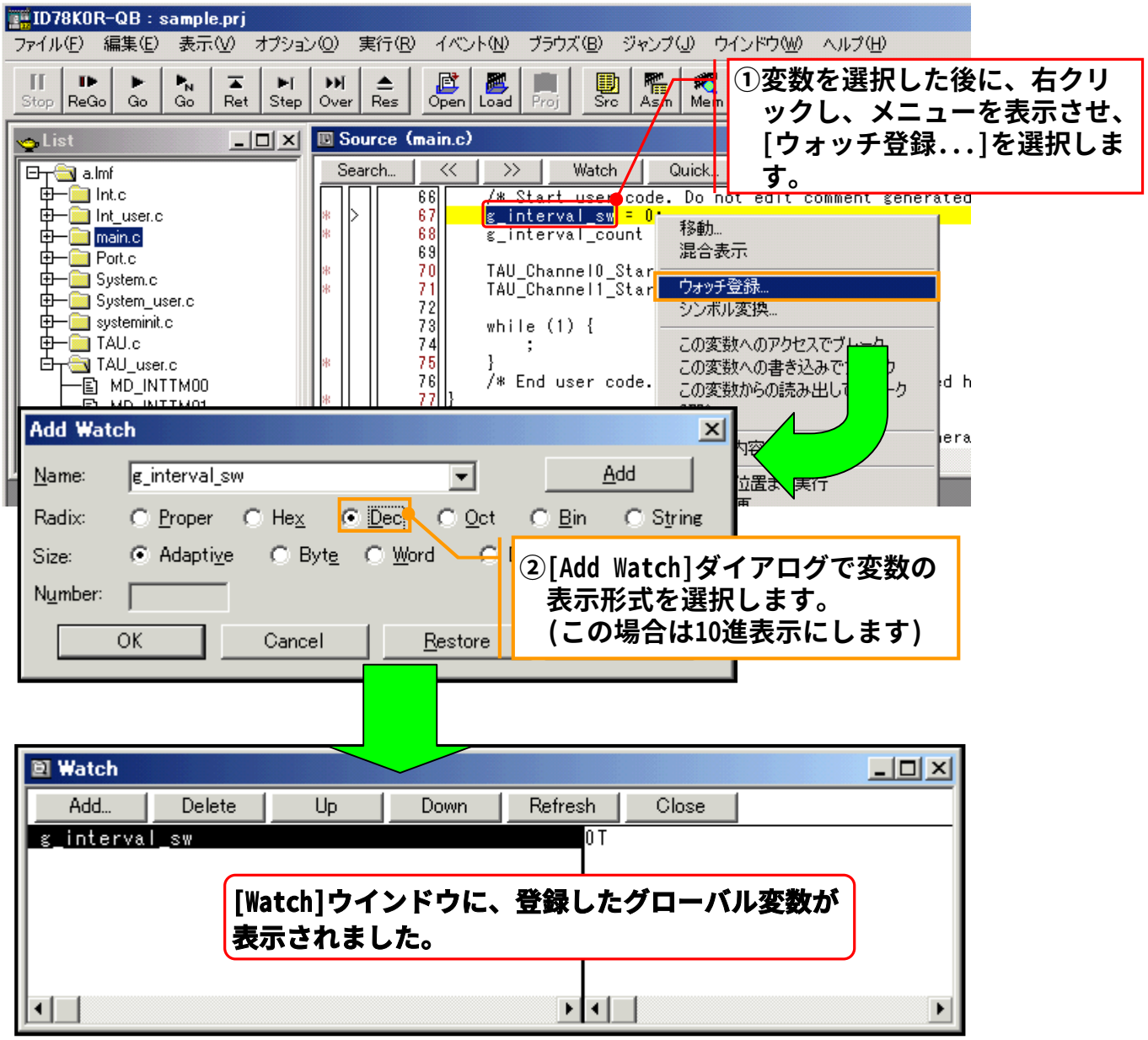
ステップ実行には色々な種類があります。[ステップ･イン(F8)]，[ネクスト･オーバー(F10)]，[リターン･アウト(F7)]など他にも[カーソル位置まで実行]，[カーソル位置から実行]もあります。

- [ステップ･イン(F8)]
  関数呼び出しがあった場合、その呼び出し関数内もステップ実行します。全てのシーケンスを確認したい時に使用します。
- [ネクスト･オーバー(F10)]
  関数呼び出しがあった場合、その呼び出し関数は一度に処理されます。ブレークしたソース内のみをデバッグしたい場合に便利です。
- [リターン･アウト(F7)]
  ブレークした関数内をリターンするまで一度に処理します。

変数表示

g. グローバル変数の表示。

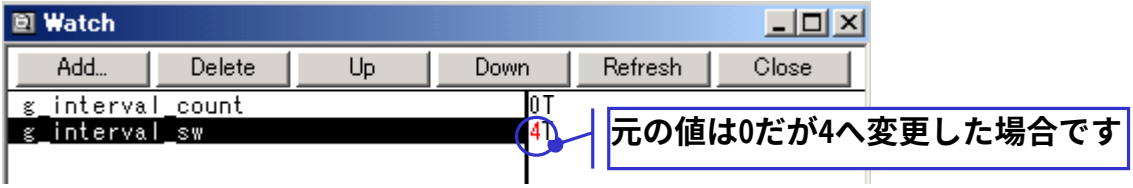
グローバル変数を表示したい場合は、[Source]ウインドウより表示させたい変数を選択して、ウォッチ登録します。[Add Watch]ダイアログより表示形式を選択すればOKです。



ワンポイント

Watchウインドウに表示している変数の書き換えについて

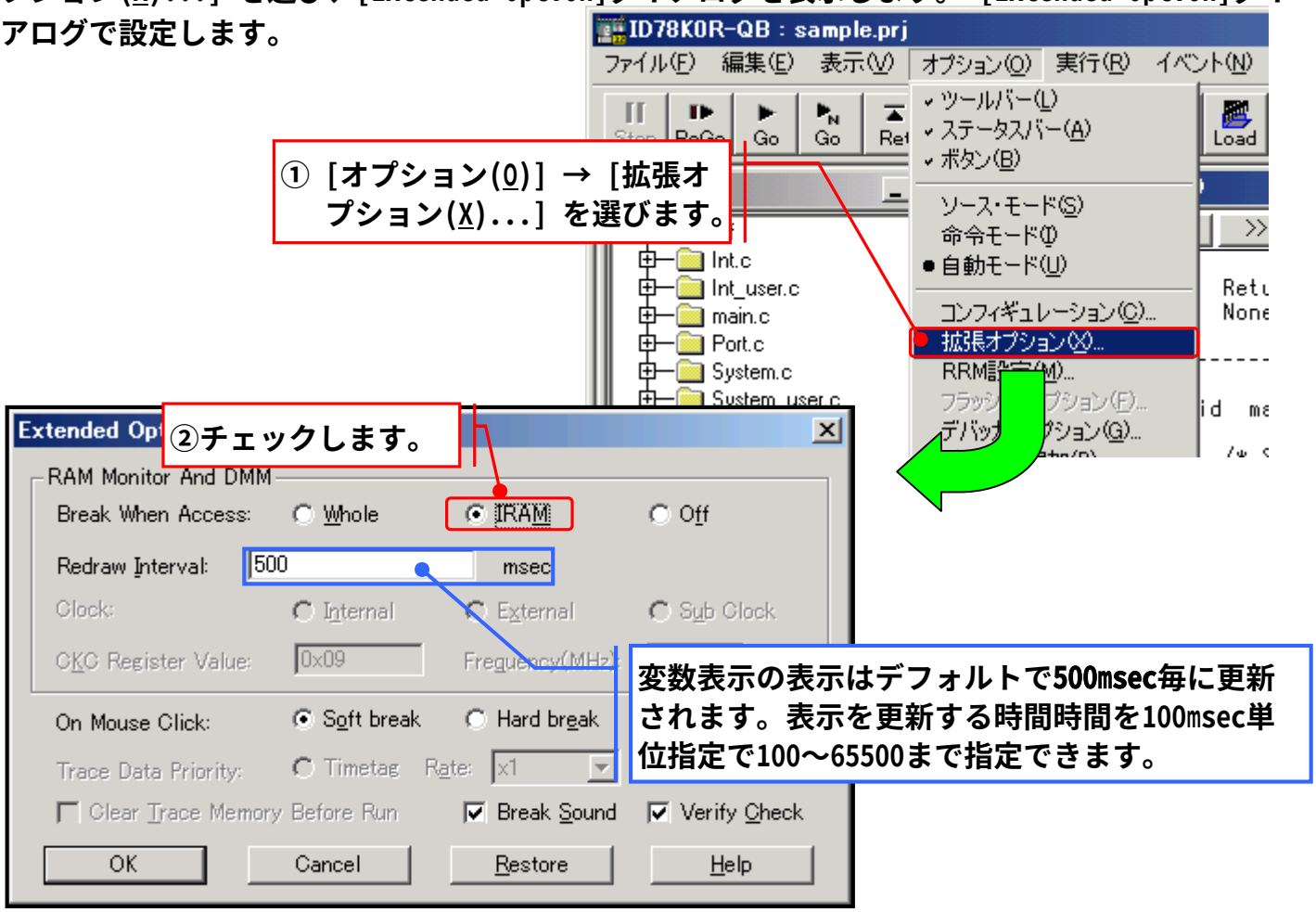
プログラム実行中にメニューの[実行(R)] → [ストップ(S)]またはStopボタン でプログラムを停止させた時Watchウインドウに表示されている変数の値を変更することができます。



変数表示

h. プログラム実行中のグローバル変数の表示方法。(V3.50より変更されました)

プログラム実行中にグローバル変数を表示したい場合、メニューの[オプション(O)] → [拡張オプション(X)...] を選び、[Extended Option]ダイアログを表示します。 [Extended Option]ダイアログで設定します。



ワンポイント

Whole,IRAMの違いについて

[Whole]

内部RAM、汎用レジスタ、SFRを対象としてプログラム実行中に表示させます。
ここにチェックした場合、メモリを読み出す範囲が広く、オープンしているウインドウが多い場合、ユーザ・プログラムを停止させる時間が長くなります。

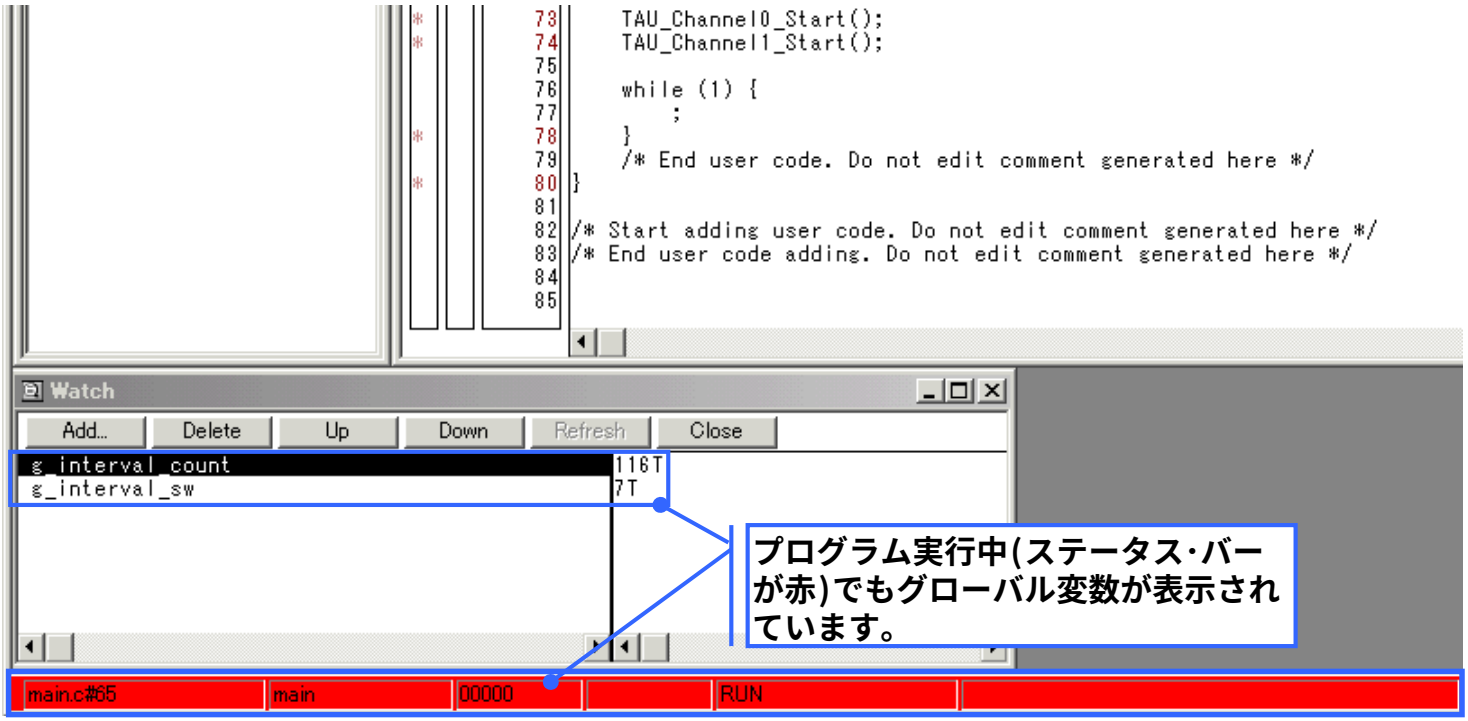
[IRAM]

内部RAM 領域及びSFR 領域を対象とします。

※Whole,IRAMのいずれかにチェックするということはプログラム実行中に一瞬プログラムを停止させます。その停止している間にレジスタ値などを読み出しを行います。プログラム実行中にメモリ、レジスタ値を表示するということは、プログラムが一瞬(数バイトの表示で数十μ秒)停止することを意識してください。

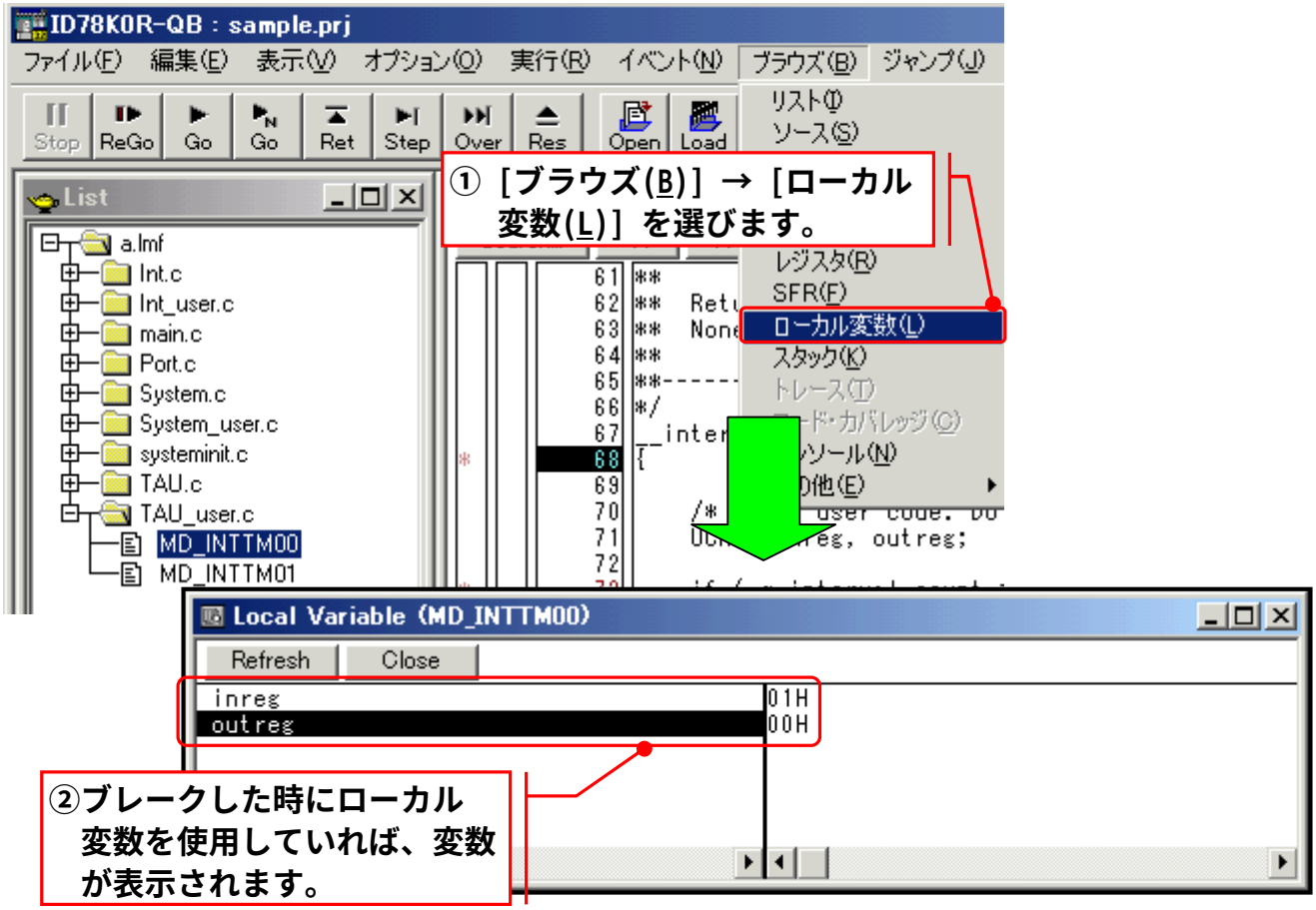
変数表示

i．プログラム実行中のグローバル変数表示画面。



j．ローカル変数の表示方法。

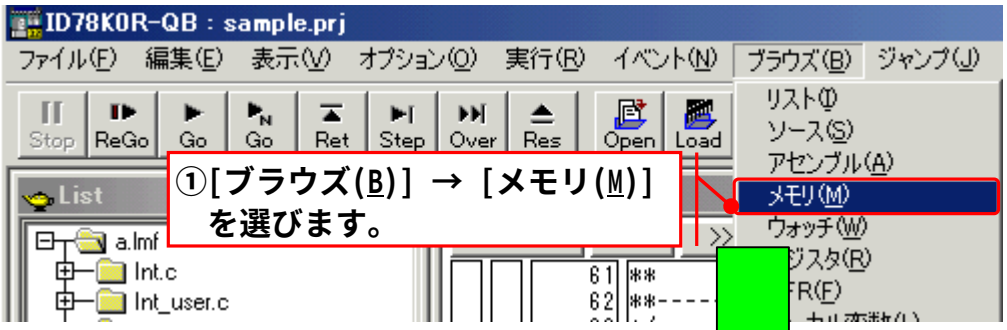
メニューの[ブラウズ(B)] → [ローカル変数(L)] を選ぶと、[Local Variable]ウインドウが表示されます。ローカル変数は実行中には表示されません。ブレークした時のみ表示します。

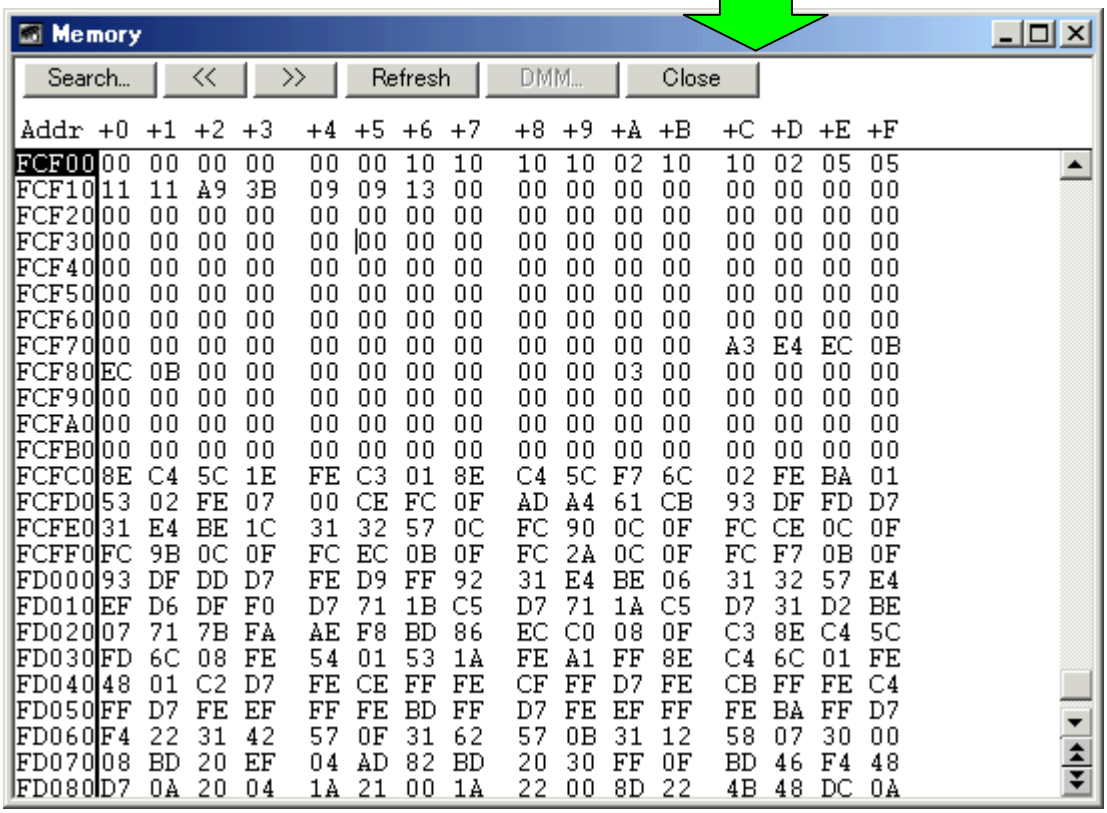


メモリ表示

k. メモリ表示方法。

メニューの[ブラウズ(B)] → [メモリ(M)] を選ぶと、[Memory]ウインドウが表示されます。プログラム実行中は通常、メモリ内容を表示できません。しかし、拡張オプションの設定を行うことにより、プログラム実行中でもメモリ内容を表示できます。



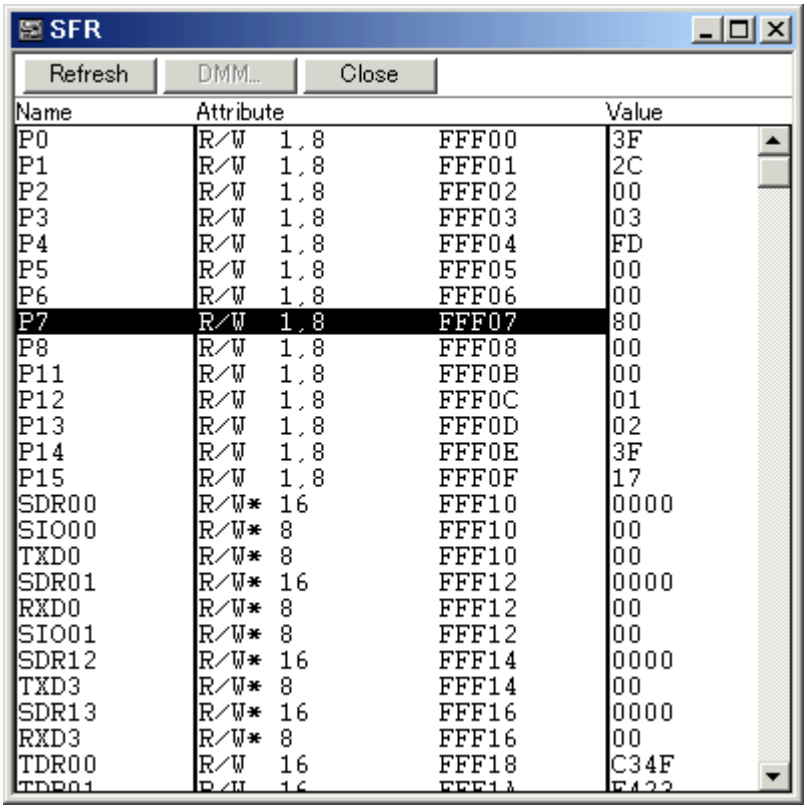


拡張オプション[IRAM]にチェックを入れるだけで プログラム実行中にメモリ表示されます。

SFR表示

1. SFR表示方法。

メニューの[ブラウズ(B)] → [SFR(F)] を選ぶと、[SFR]ウインドウが表示されます。



Name	Attribute		Value
P0	R/W 1,8	FFF00	3F
P1	R/W 1,8	FFF01	2C
P2	R/W 1,8	FFF02	00
P3	R/W 1,8	FFF03	03
P4	R/W 1,8	FFF04	FD
P5	R/W 1,8	FFF05	00
P6	R/W 1,8	FFF06	00
P7	R/W 1,8	FFF07	80
P8	R/W 1,8	FFF08	00
P11	R/W 1,8	FFF0B	00
P12	R/W 1,8	FFF0C	01
P13	R/W 1,8	FFF0D	02
P14	R/W 1,8	FFF0E	3F
P15	R/W 1,8	FFF0F	17
SDR00	R/W* 16	FFF10	0000
SIO00	R/W* 8	FFF10	00
TXD0	R/W* 8	FFF10	00
SDR01	R/W* 16	FFF12	0000
RXD0	R/W* 8	FFF12	00
SIO01	R/W* 8	FFF12	00
SDR12	R/W* 16	FFF14	0000
TXD3	R/W* 8	FFF14	00
SDR13	R/W* 16	FFF16	0000
RXD3	R/W* 8	FFF16	00
TDR00	R/W 16	FFF18	C34F

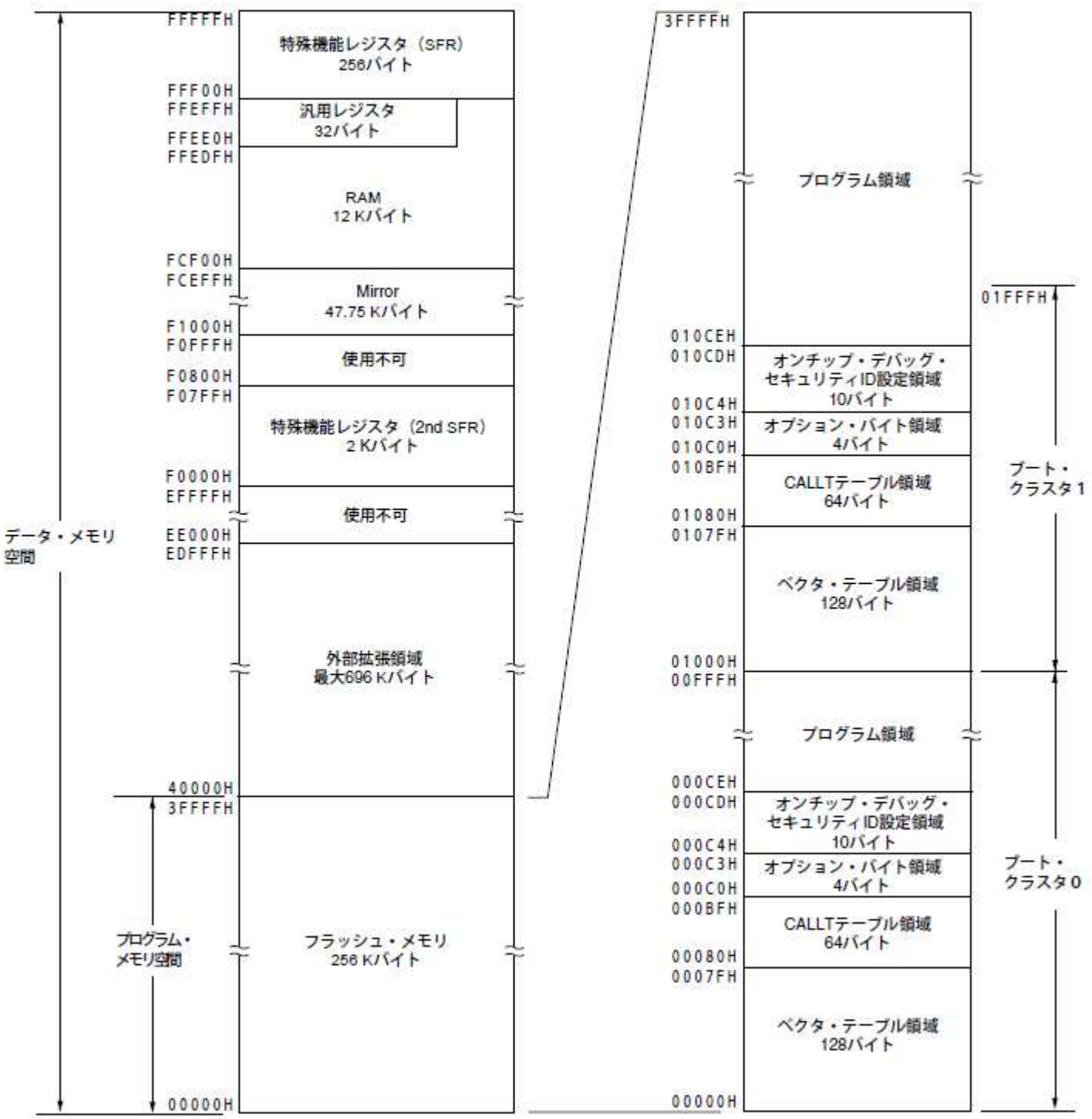
ワンポイント

SFRについて

特殊機能レジスタの意味です。レジスタ類はメモリ空間に割り当てられています。FFFF00h～FFFFFFhの256バイトが特殊機能レジスタ(SFR)と呼ばれます。

メモリマップ

m. QB-78K0RKG3-TBに搭載されているマイコン(μPD78F1166)のメモリマップです。



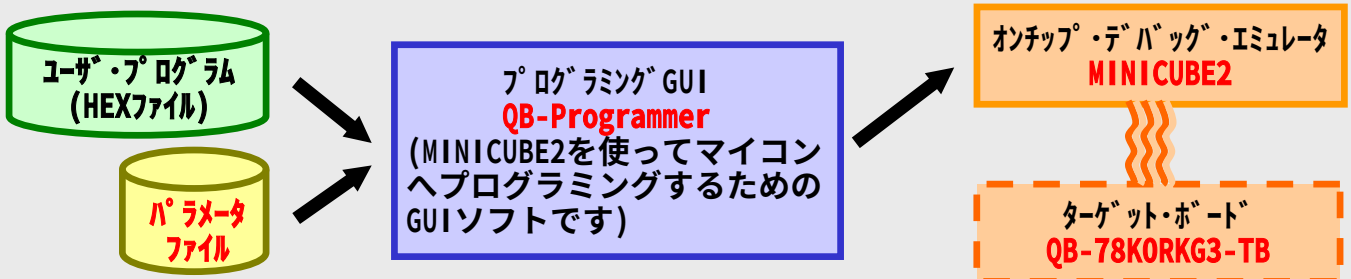
「デバッグしたい」の章は以上です。ここに記したデバッグ手法は基本に過ぎません。プログラムが複雑になるほど多くの問題が発生します。多くの例では、スタックの問題が挙げられます。マイコンはRAMの容量が限られています。配列などを多く使うと知らない内にプログラムが使うスタック領域が足りなくなる問題が発生します。プログラムを作る際はマイコンのメモリマップも意識する必要があります。
次章より「マイコンへ書き込み」を説明します。マイコンへプログラミング(書き込み)する場合は、次章を読んで下さい。

プログラミングについて

15:00

[マイコンへ書き込み]の章では、作成したプログラムをマイコンへプログラミング(書き込み)します。一度プログラミングすれば電源を供給するだけでプログラムが実行されます。学習時間は15分です。

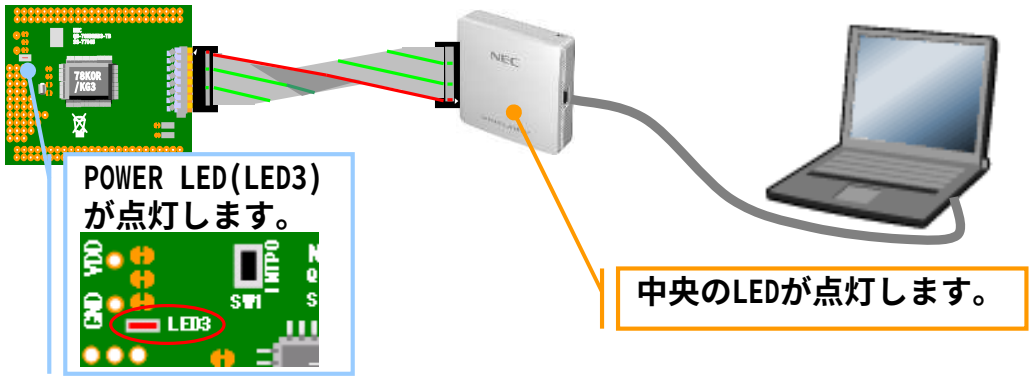
プログラミング方法



マイコンへプログラミングする場合

MINICUBE2との接続

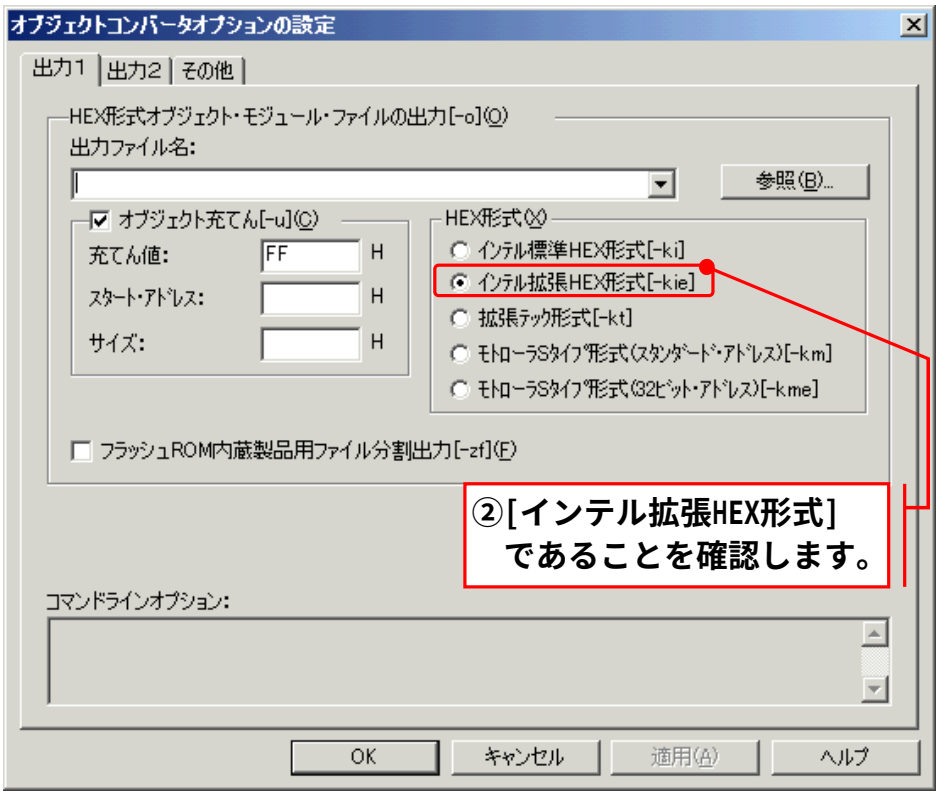
[動かしてみよう]の章、[MINICUBE2の接続]と同様にデバッガを使用するときのように接続します。



HEXファイル作成

a. HEXファイル作成する設定を確認します。

PM+のメニューより[ツール(T)] → [オブジェクトコンバータオプションの設定(O)...] を選んで下さい。[オブジェクトコンバータオプションの設定]ウインドウが開きます。



b. ビルドを行い、HEXファイルを作成します。

PM+のメニューより[ビルド(B)] → [ビルド(B)...] を選んでビルドを行って下さい。
詳細は[動かしてみよう]の章[プログラムのビルド]を参考にして下さい。

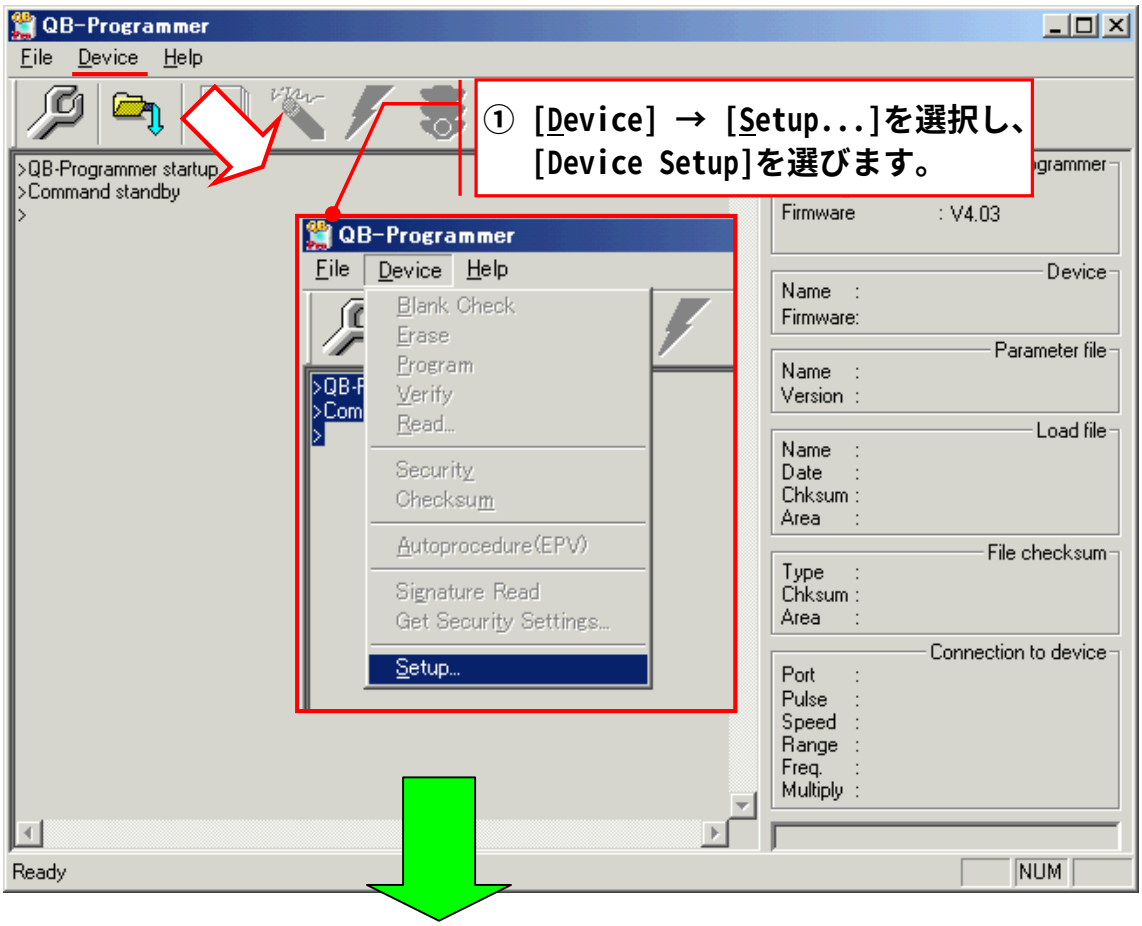
ワンポイント

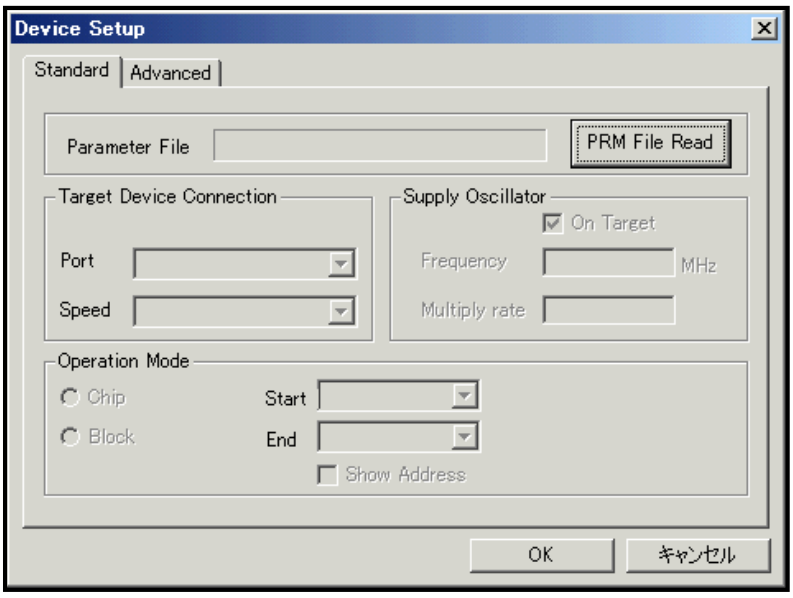
HEX形式について

「インテル拡張HEX形式」ではアドレス0xFFFFFまでをサポートしています。「インテル標準HEX形式」では64KByteのアドレス0xFFFFまでしかサポートしていませんので、1MByteのアドレス空間を持つ78K0Rマイコンでは通常使用しません。MINICUBE2以外でプログラミングする場合には「モトローラSタイプ形式」を使う場合があります。

QBP(QB-Programmer)の起動

c. マイコンへプログラミングするソフトウエア(QBP)を起動します。
[スタート] → [プログラム(P)] → [NEC Electronics Tools] → [QBP] → [Vx.xx] → [QBP Vx.xx QB-Programmer]でQBPを起動します。起動後に[Device] → [Setup...]を選択し、[Device Setup]ウインドウを開きます。

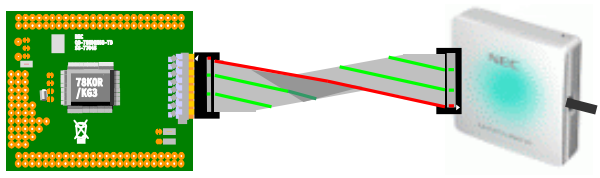




ワンポイント

QBP起動時について

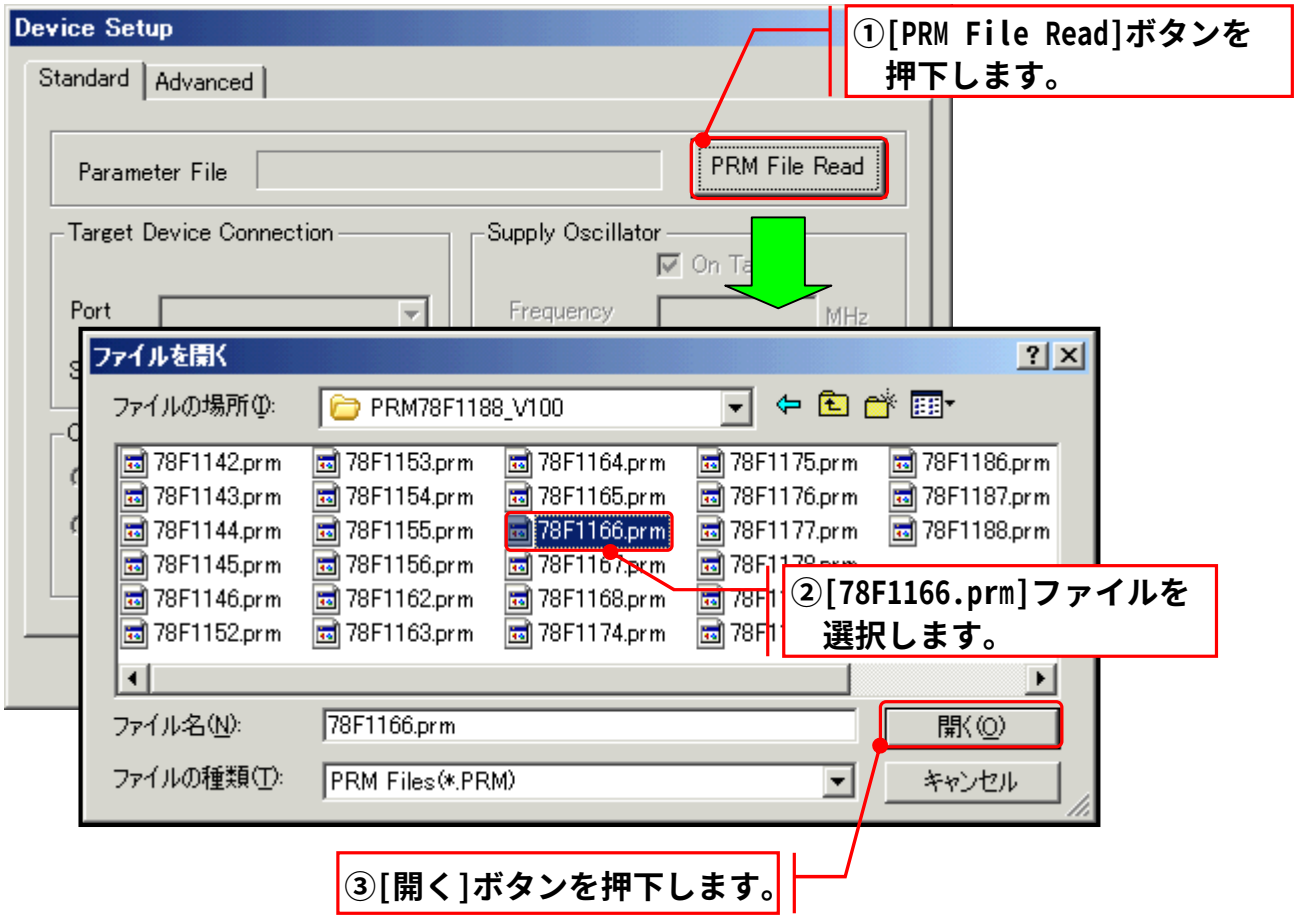
QBPを起動すると、MINICUBE2の中央LEDが緑に変わります。これはプログラムモードに入った事を示します。



パラメータファイルの読み込み

d. パラメータファイルを読み込みます。

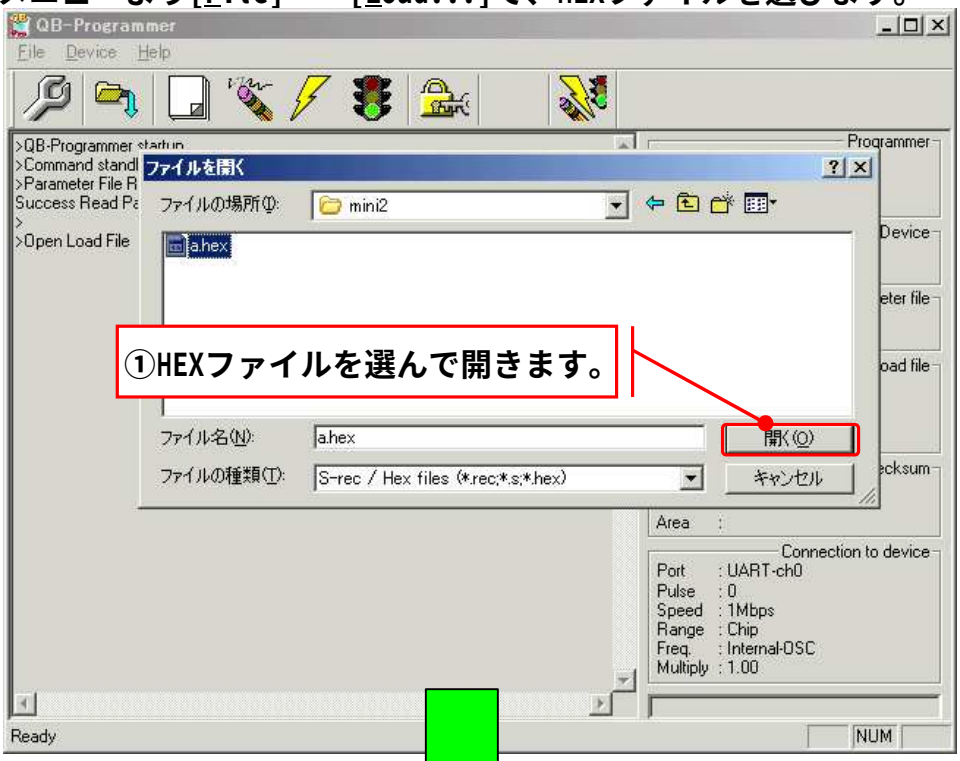
[Device Setup]ウインドウより[PRM File Read]ボタンを押下してパラメータファイルを読んで下さい。

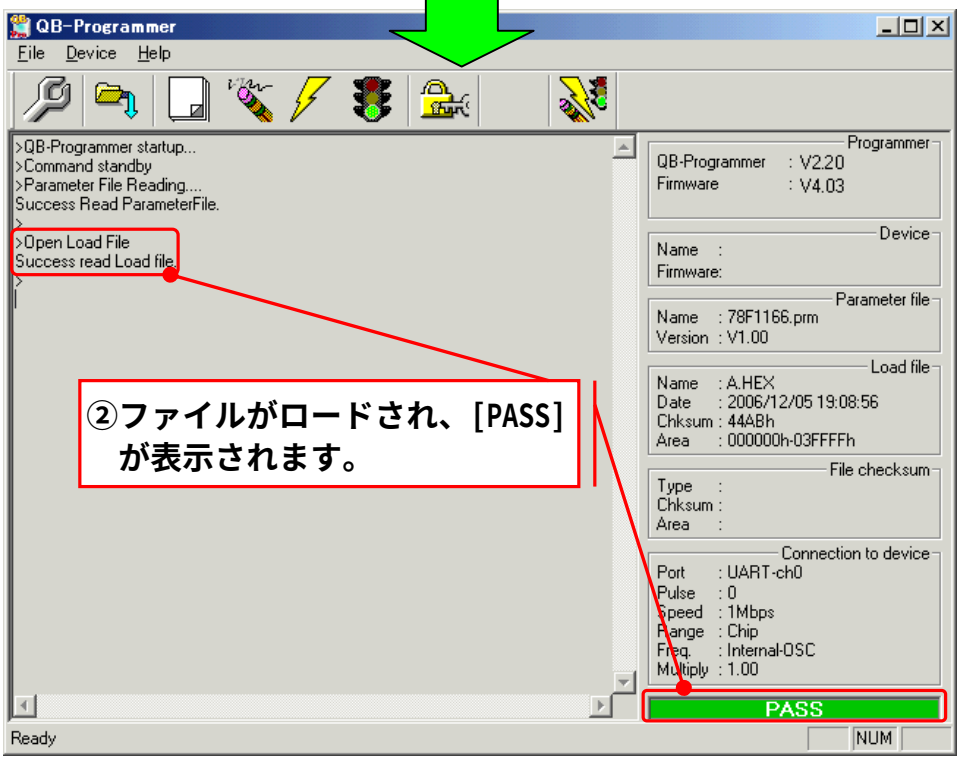


HEXファイルの読み込み

e. HEXファイルを読み込みます。

メニューより[File] → [Load...]で、HEXファイルを選びます。





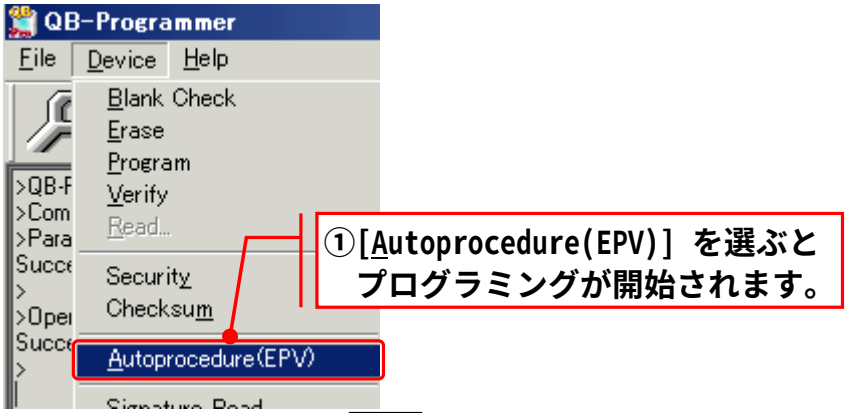
ワンポイント

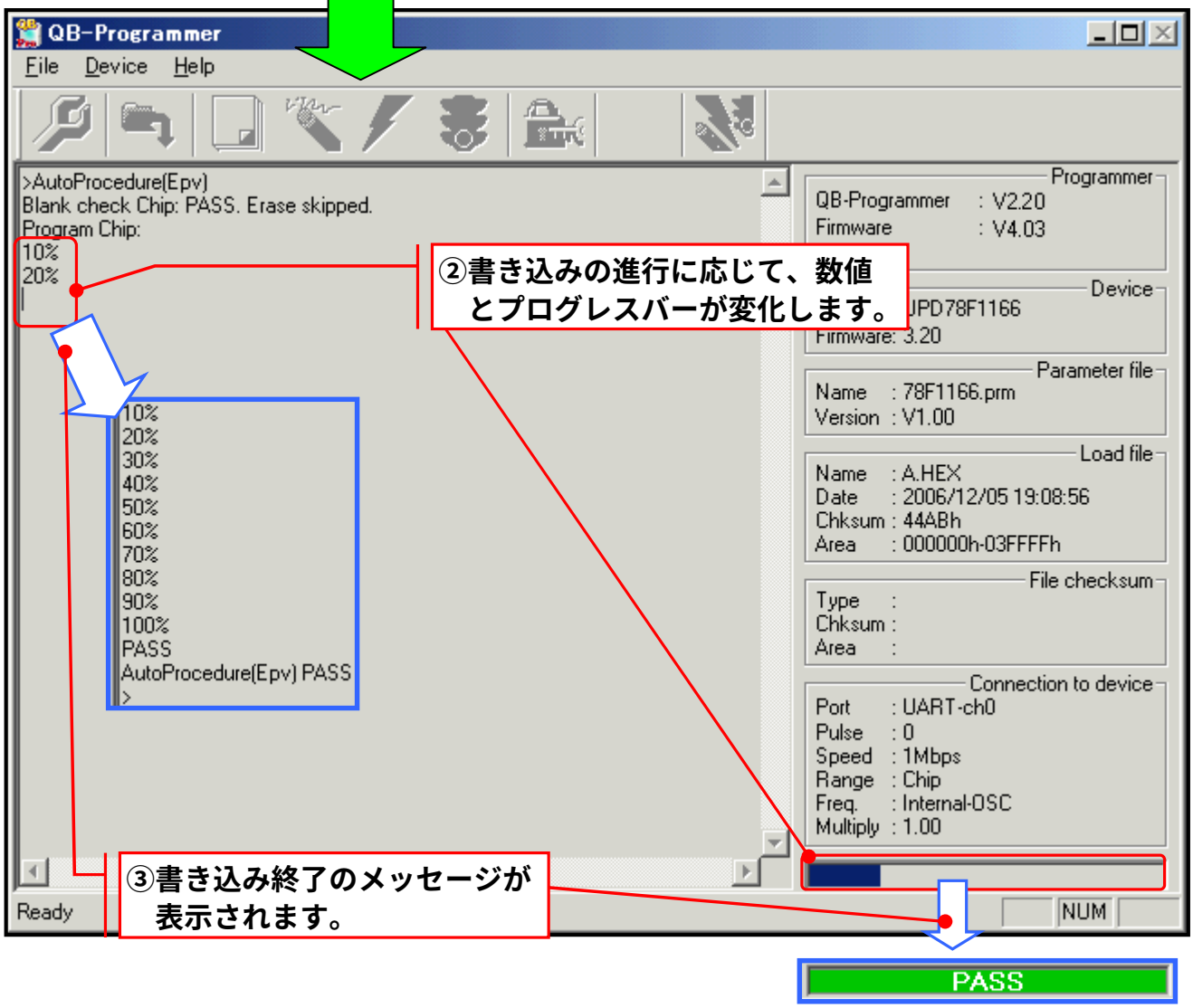
QBP(QB-Programmer)の起動について

2回目以降にQBPを起動した場合は、以前の設定が反映されます。そのためパラメータファイルの設定、プログラムのダウンロードなど再設定する必要がありません。

マイコンへプログラミング

e. マイコンへプログラミングします。
メニューより[File] → [Autoprocedure(EPV)]を選ぶとプログラミングされます。

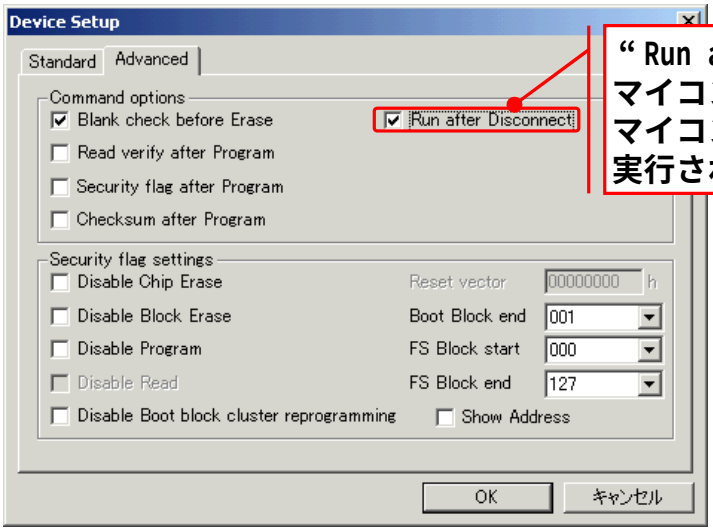




動作確認

f. マイコンの動作確認します。

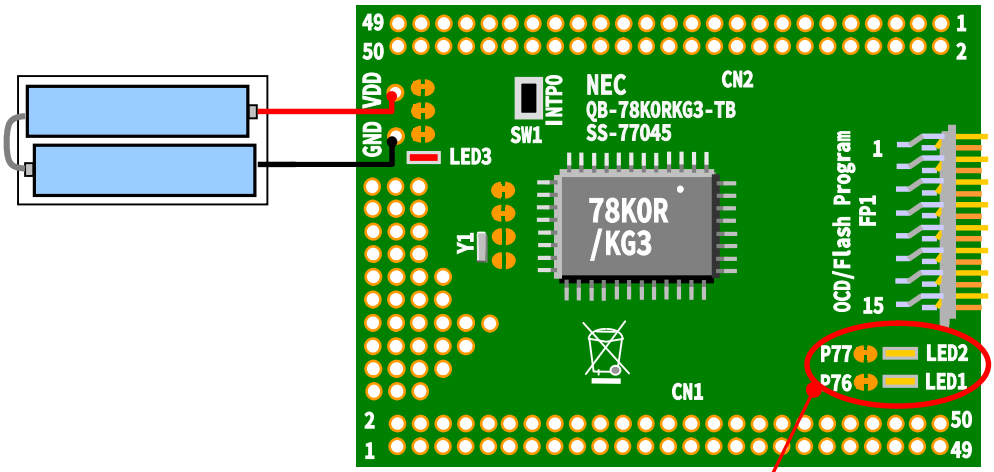
[Device] → [Setup...]を選択し、[Device Setup]ウインドウを開きます。[Device Setup]ウインドウより[Acvanced]タブを選択し、”Run after Disconnect”へチェックして下さい。



“Run after Disconnect” へチェックすると、「e. マイコンへプログラミングします。」の処理を行い、マイコンの書き込みが終了した後に、プログラムが実行されます。

また、下記の方法でも動作確認できます。

MINICUBE2とUSBケーブルを外し、16pinケーブルも外してQB-78K0RKG3-TBだけにします。
VDD，GNDに乾電池2本を接続し、動作することを確認します。



電池を接続するとLEDが点滅開始します。
SW1の押下でLEDの点滅タイミングも変化します。

「マイコンへ書き込み」の章は以上です。現在までの章「動かしてみよう」、「デバッグしたい」とあわせて、マイコンの開発一通りを学びました。次章ではプログラムの実行、デバッグをソフトウエアのみで実現する「システム･シミュレータ」について説明します。

システム･シミュレータ(SM+)とは

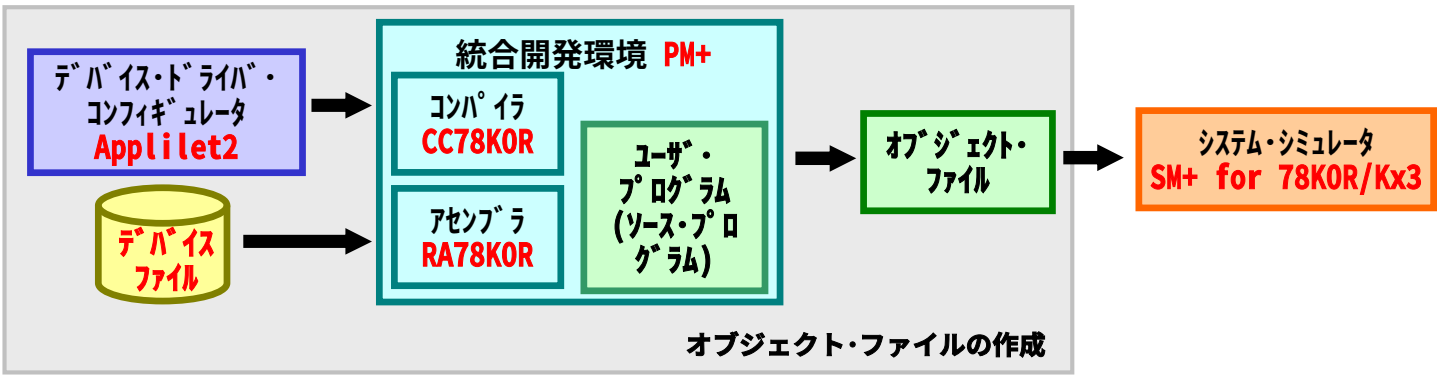
30:00

[ソフトで開発]の章では、作成したプログラムの実行、デバッグをソフトウエアで実現する「システム･シミュレータ(以下SM+)」について説明します。学習時間は30分です。

デバッグ可能なハードウエア環境があるのに、なぜSM+を使う必要があるのか？と思われるでしょう。SM+には、以下の特徴があります。

- **ターゲットを必要としない評価の実現**
  ハードウエアを必要としないので、実際のデバッグ環境を構築せずに先行してソフトウエアのみでデバッグが可能です。
- **さまざまなシミュレーション機能の実現**
  スイッチ、LEDなどの入出力パネルをソフトウエアで構築可能です。また、マイコンの入出力波形をタイミング･チャート･ウインドウ(ロジック･アナライザのイメージ)で計測可能です。
- **制限のないデバッグ**
  ブレークポイントの数やカバレッジ機能などオンチップ･デバッグでは実現できない豊富なデバッグ手法が使えます。

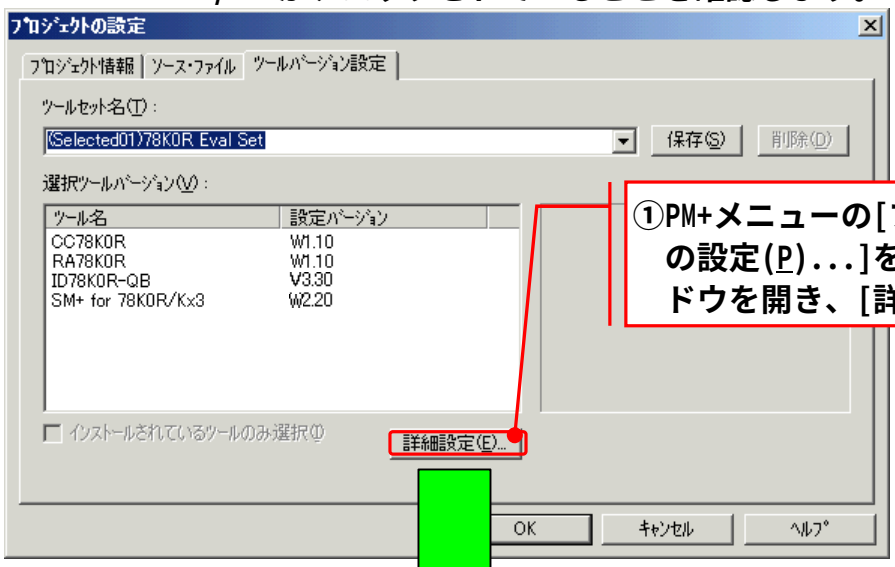
プログラム開発環境



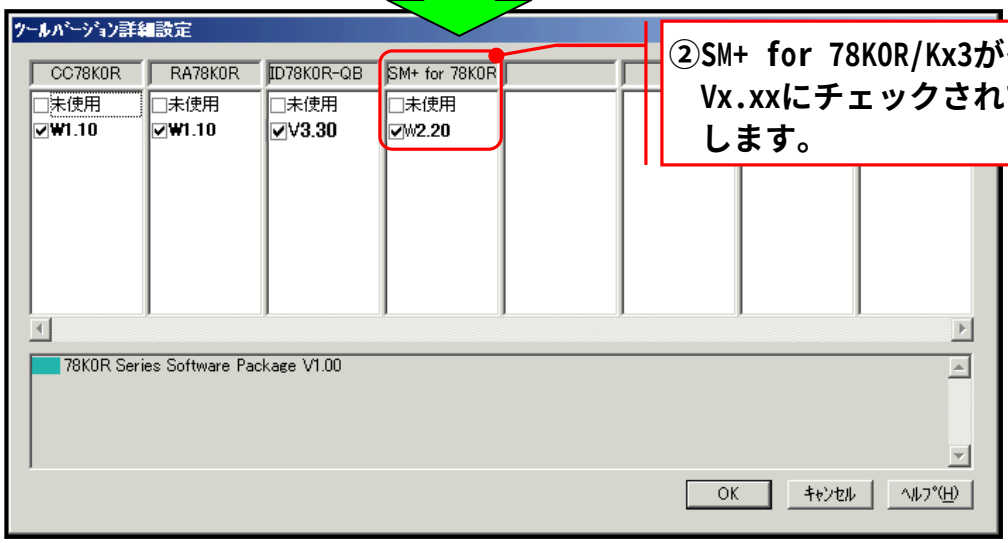
環境の確認

a. PM+の環境を確認します。

PM+メニューの[プロジェクト(P)] → [プロジェクトの設定(P)...]を選び、[プロジェクトの設定]ウインドウを開き、[詳細設定]ボタンを押下します。[ツールバージョン詳細設定]ウインドウでSM+ for 78K0R/Kx3がチェックされていることを確認します。



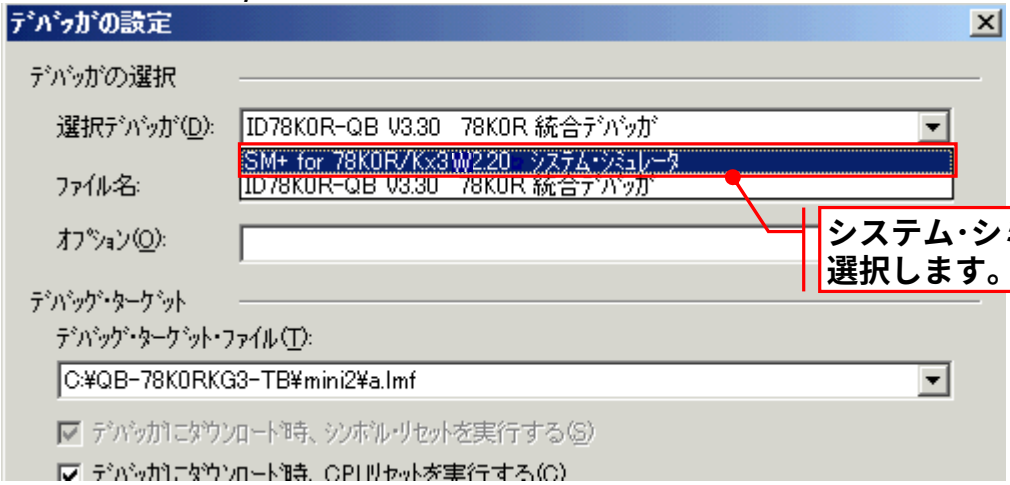
①PM+メニューの[プロジェクト(P)] → [プロジェクトの設定(P)...]を選び、[プロジェクトの設定]ウインドウを開き、[詳細設定]ボタンを押下します。



②SM+ for 78K0R/Kx3がインストールされ、Vx.xxにチェックされていることを確認します。

b. PM+で使用するデバッガを設定します。

メニューの[ツール(T)] → [デバッガの設定(D)...] で「デバッガの設定」ダイアログを開き「SM+ for 78K0R/Kx3 Vx.xxシステム・シミュレータ」を選択して下さい。

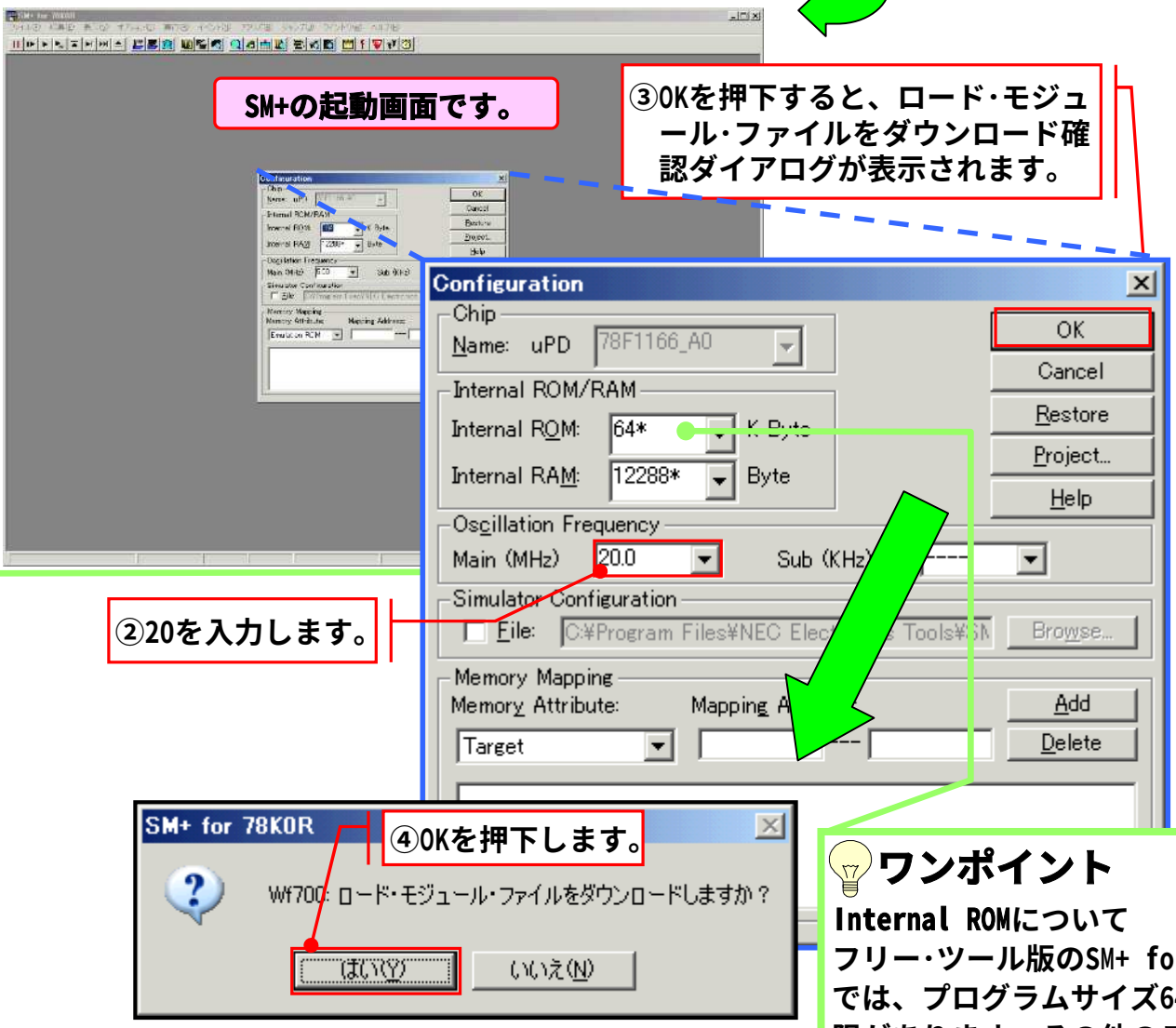


システム・シミュレータを選択します。

SM+の起動

c. SM+を起動します。

PM+のデバッグ･アイコンを押下して[SM+]を起動します。プログラムのダウンロードまで行って下さい。



ワンポイント

Internal ROMについて

フリー･ツール版のSM+ for 78K0R/Kx3では、プログラムサイズ64KBまでの制限があります。その他のデバッグ機能については制限ありません。

ワンポイント

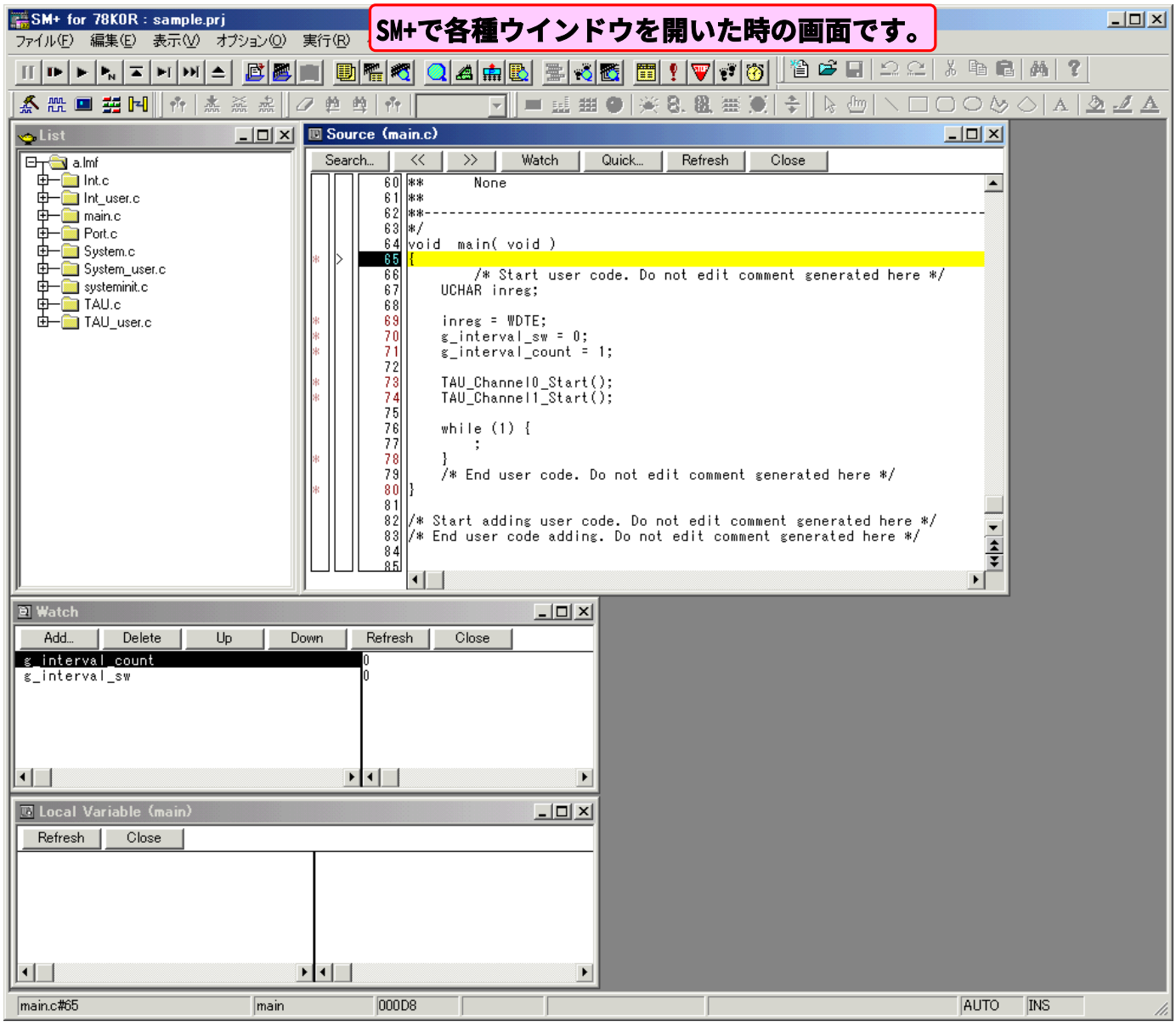
Oscillation Frequencyについて

Oscillation Frequencyの値はApplilet2の高速システム･クロック設定の周波数(MHz)と同じ値を設定して下さい。この値はシリアルのボーレート、タイマの動作などに影響しますので正しい値を設定して下さい。

SM+の基本画面

d. SM+の基本画面を設定します。

[デバッグしたい] の章 [デバッガの基本画面] を参考に[List], [Watch], [Local Variable]のウインドウを開いて下さい。ただし、[デバッグしたい] の章[変数表示]で行っている [オプション(O)] → [拡張オプション(X)...]の[Use MINICUBE Extended Function]にチェックする必要はありません。



ワンポイント

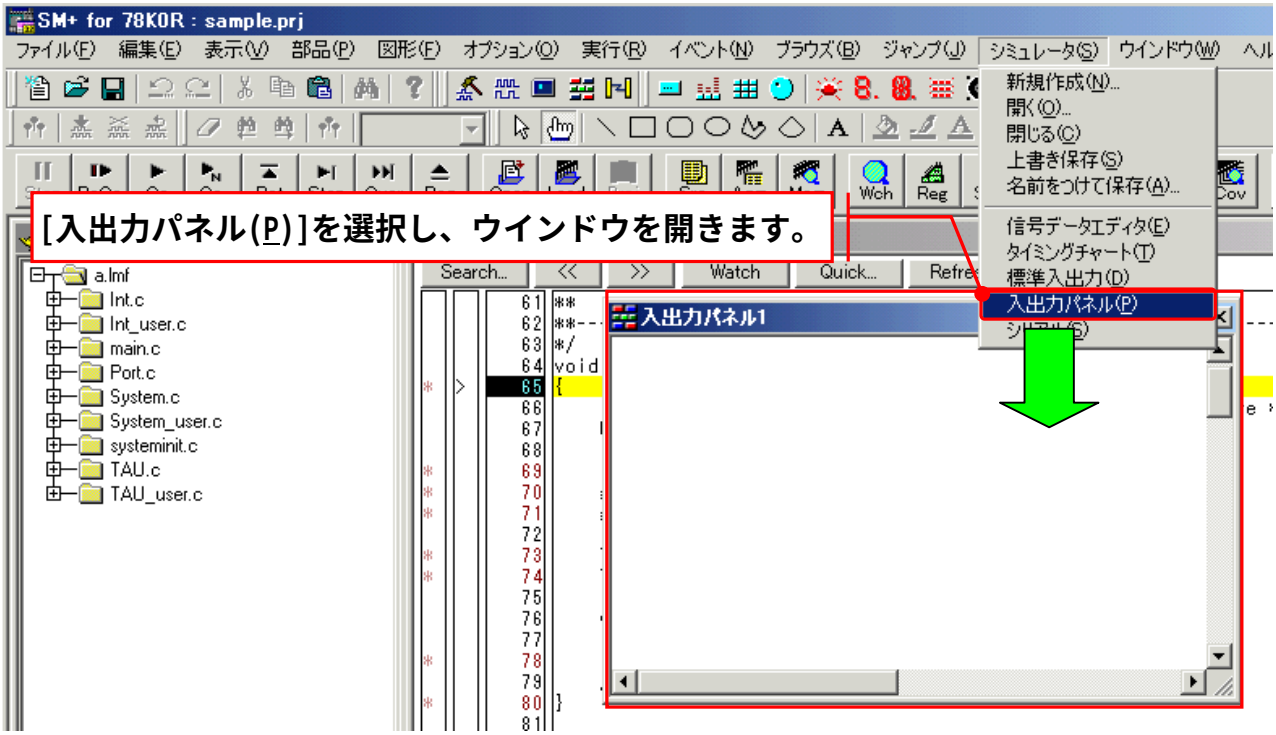
基本的な使い方について

システム･シミュレータ(SM+)も統合デバッガ(ID78K0R-QB)も使い方としては同じです。ブレークポイントの設定、Watchウインドウの設定方法も同じです。しかし、SM+にはさまざまなシミュレーション機能があります。次ページよりシミュレーション機能を説明します。

入出力パネルの設定

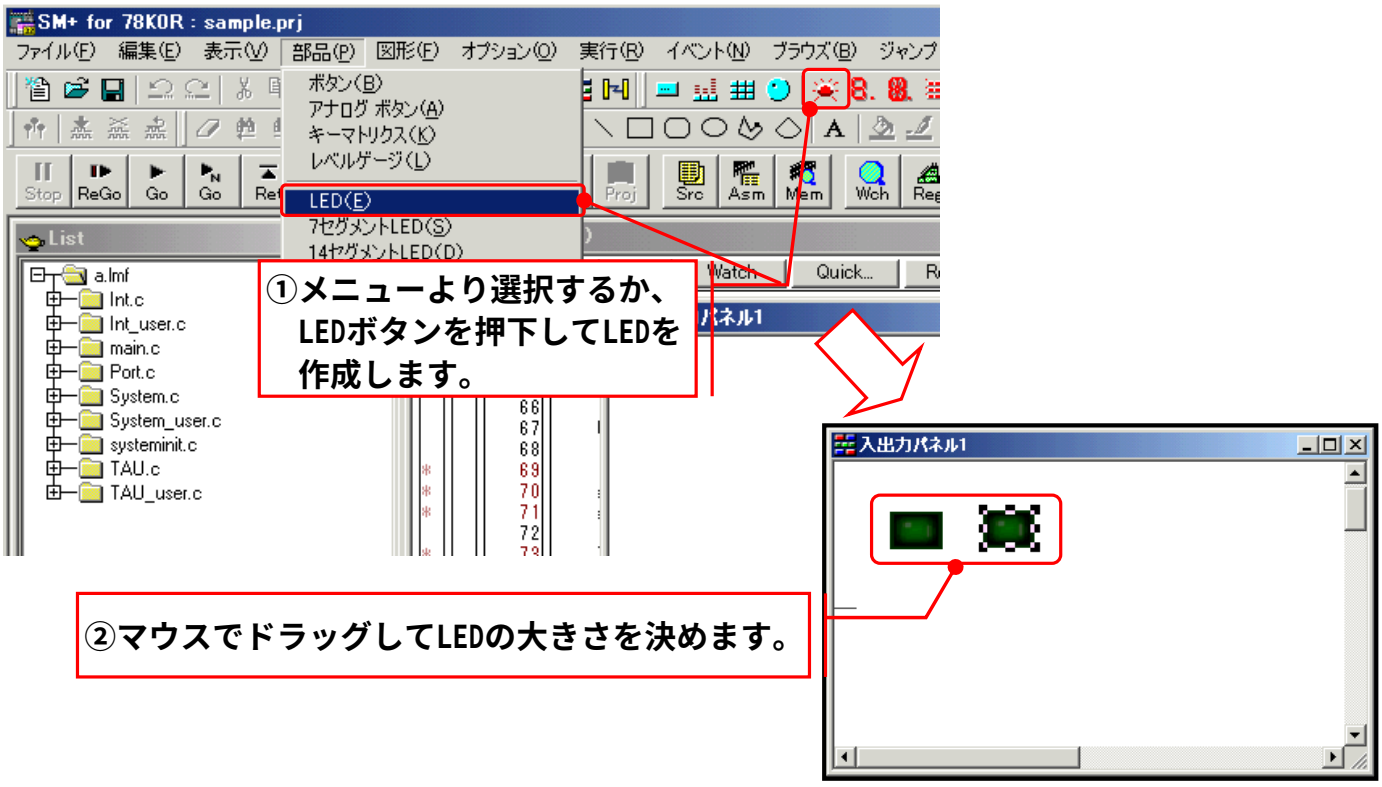
e. SWやLEDを作る為のウインドウを開きます。

SM+のメニュー［シミュレータ(S)］→［入出力パネル(P)］を選び［入出力パネル］ウインドウを開きます。



f. LEDを作成する。

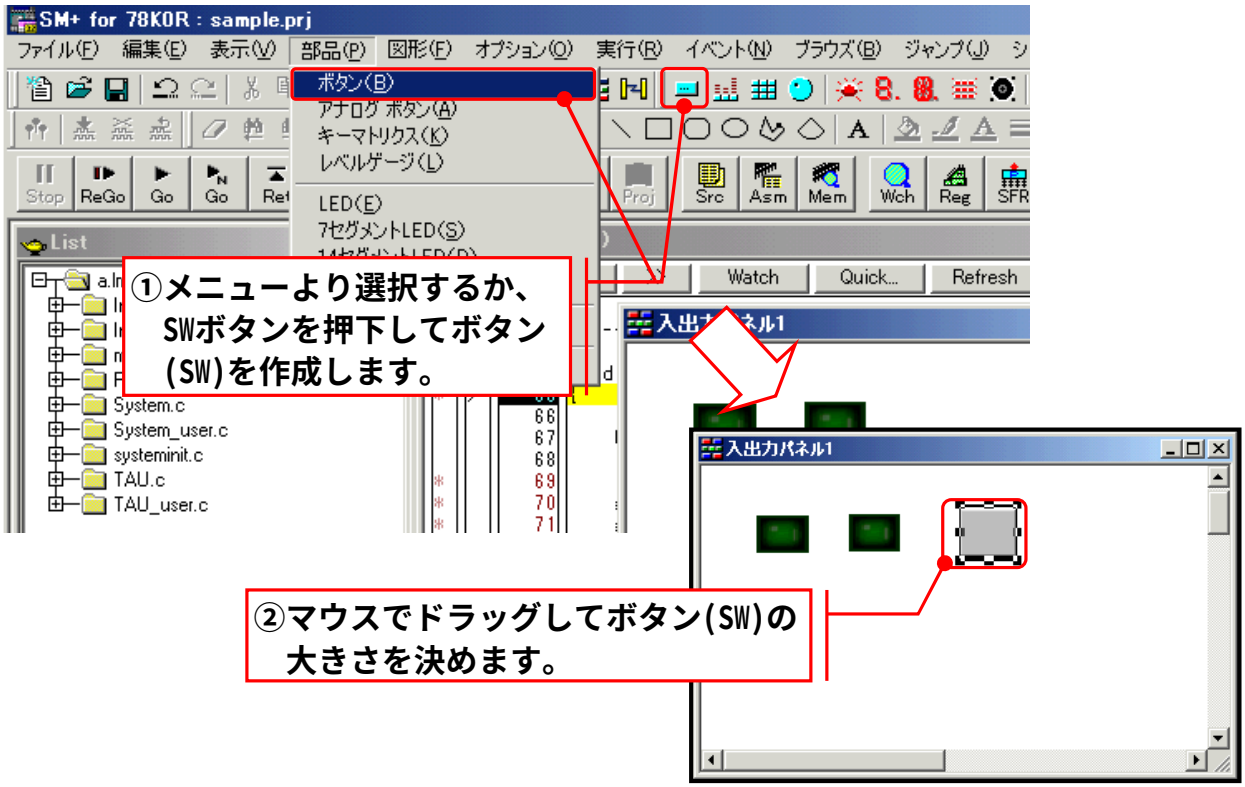
「入出力パネル１」をアクティブにした状態でLED作成アイコン  を押すか［部品(P)］→［LED(E)］を選択して、「入出力パネル１」に任意の大きさのLEDを2個、貼り付けます



入出力パネルの設定

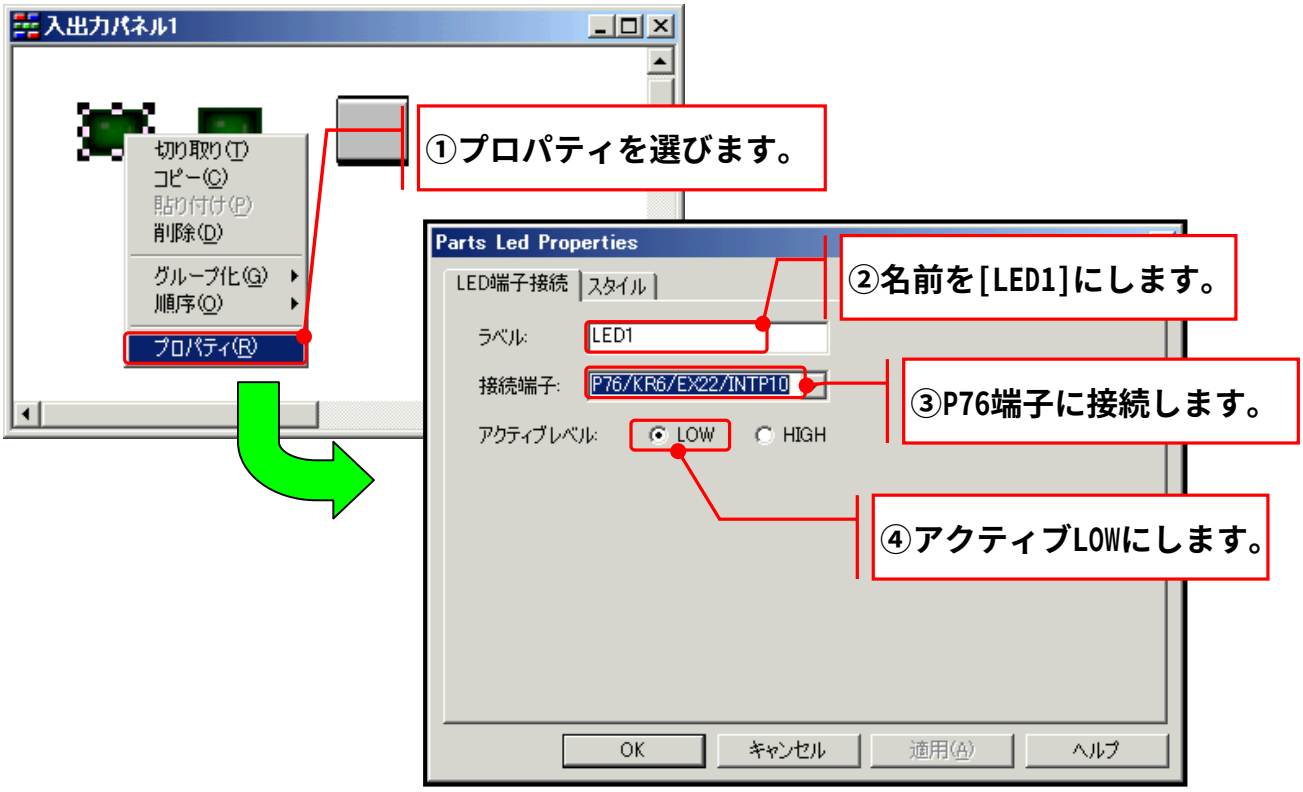
g. SW(ボタン)を作成します。

LEDを作成するのと同様にボタン作成アイコン  を押下するか[部品(P)]→[ボタン(B)]を選択して、「入出力パネル１」に任意の大きさのSWを貼り付けます



h. 作成したLEDに端子を割り当てます。

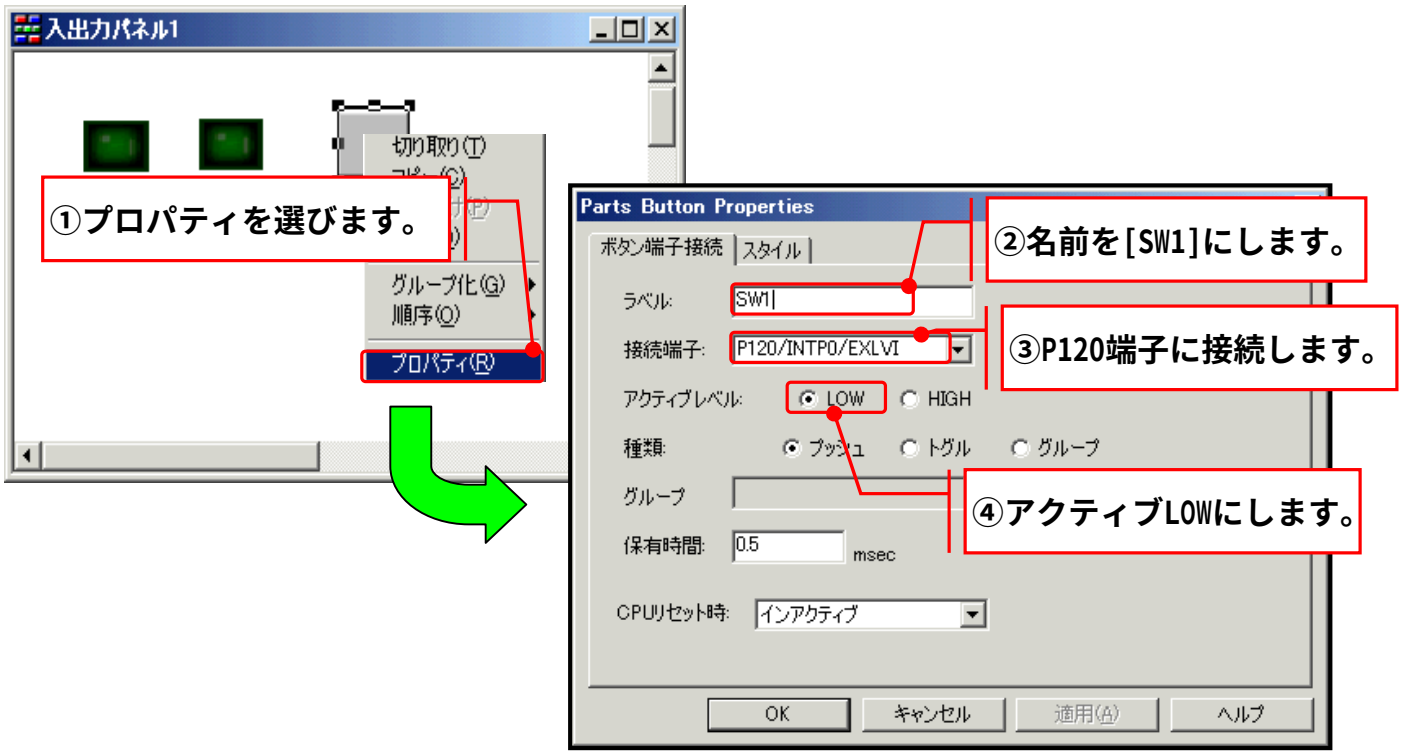
LED部品の上で右クリック→[プロパティ(R)]を選択して表示される[Parts Led Properties]ダイアログで、接続する端子を設定します。同様に2つ目のLEDはラベル[LED2]、接続端子[P77]、アクティブレベル[LOW]に設定します



入出力パネルの設定

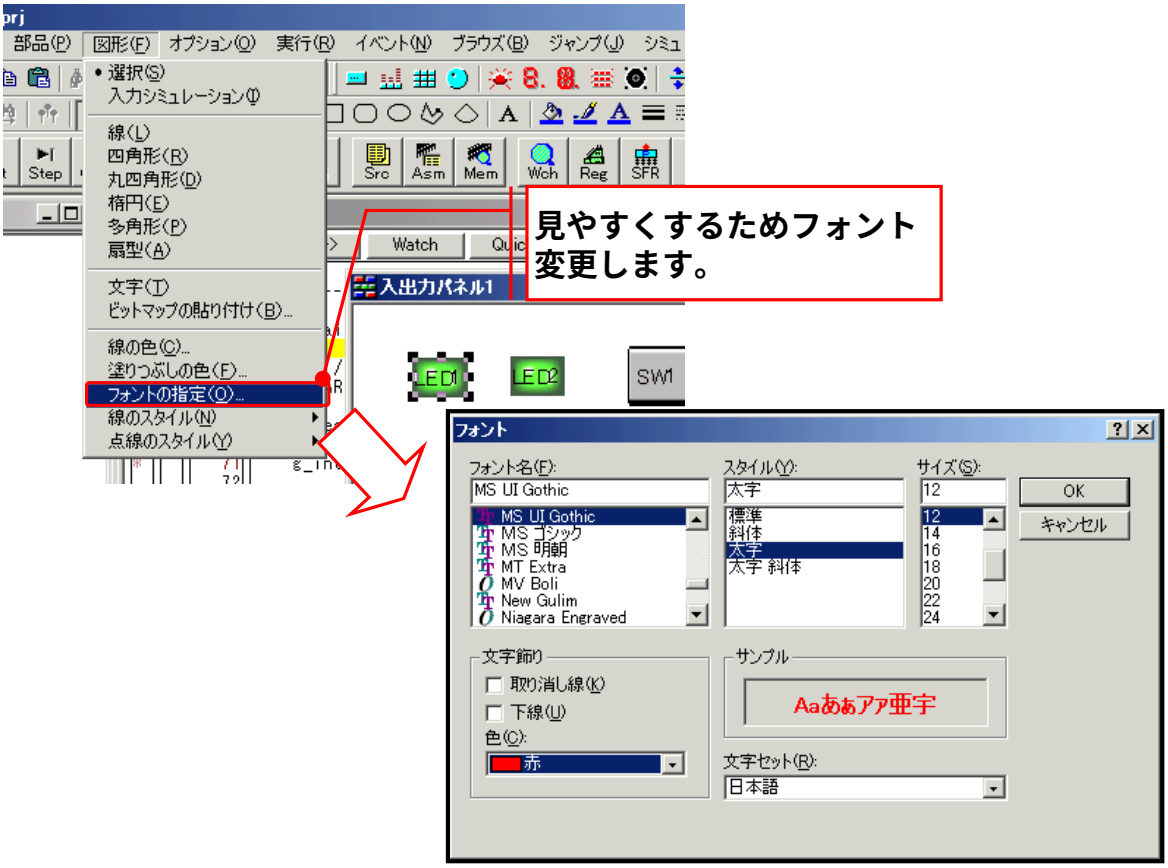
i. 作成したSWに端子を割り当てます。

LED部品の上で右クリック→[プロパティ(R)]を選択して表示される[Parts Button Properties]ダイアログで、接続する端子を設定します。



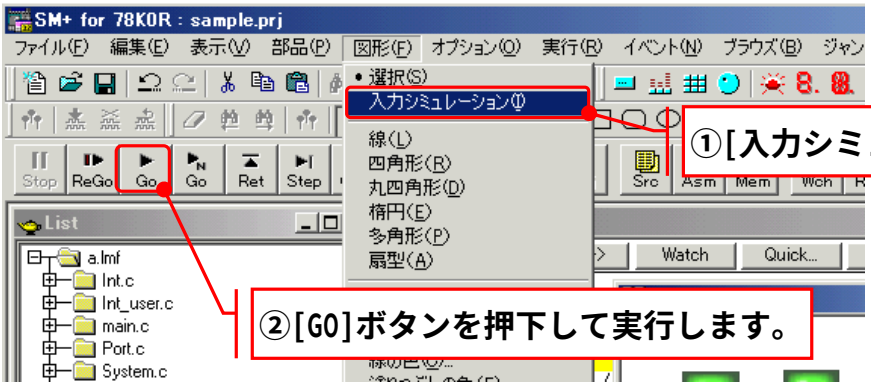
j. フォントを変更します。

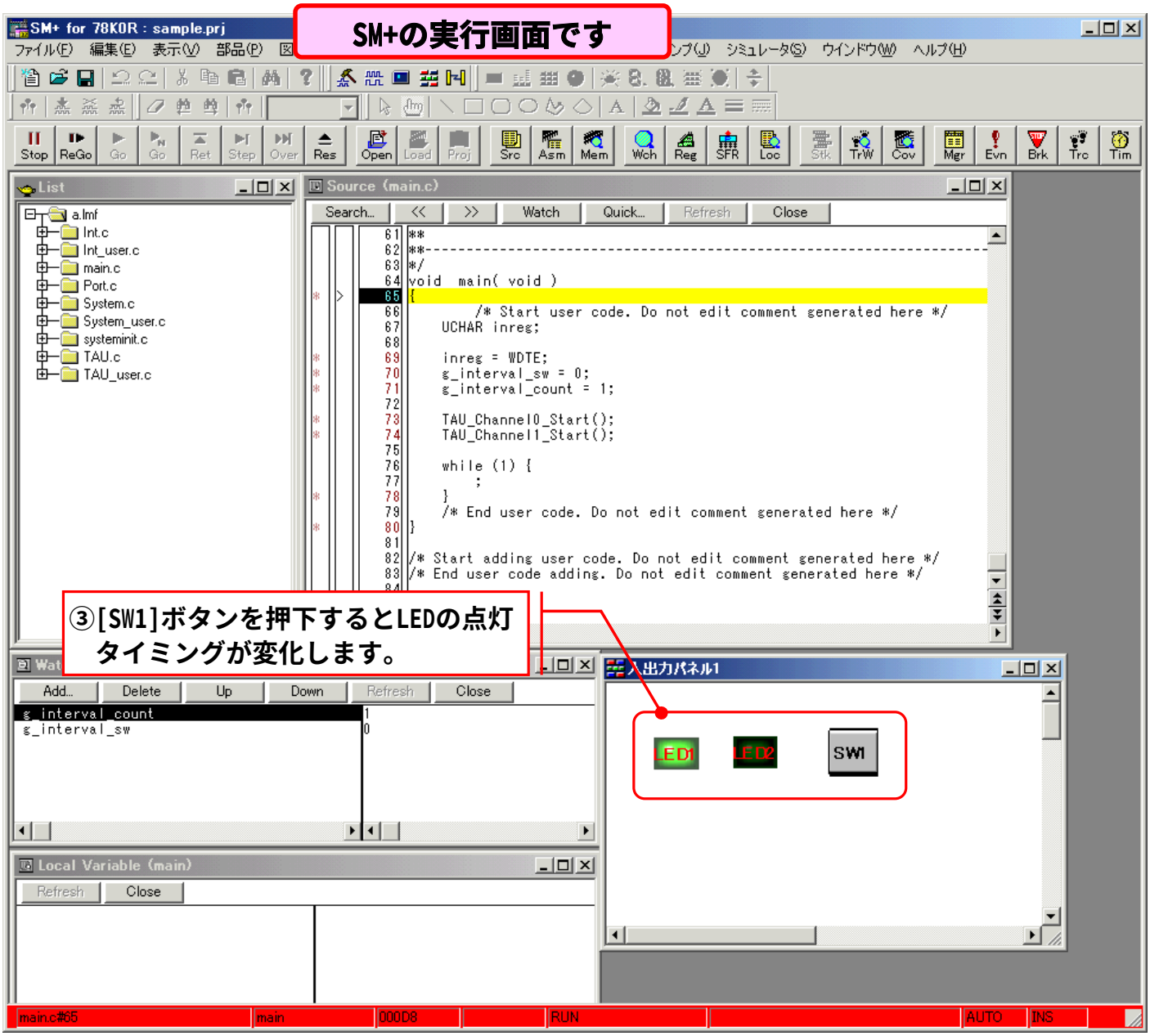
SM+のメニュー [図形(F)] → [フォントの指定(O)...]でフォント変更が可能です。



実行画面

k. 入出力パネルを[入力シミュレーション]へ設定して実行します。
入出力パネル編集時は[選択]、プログラムを実行するときは[入力シミュレーション]にして下さい。

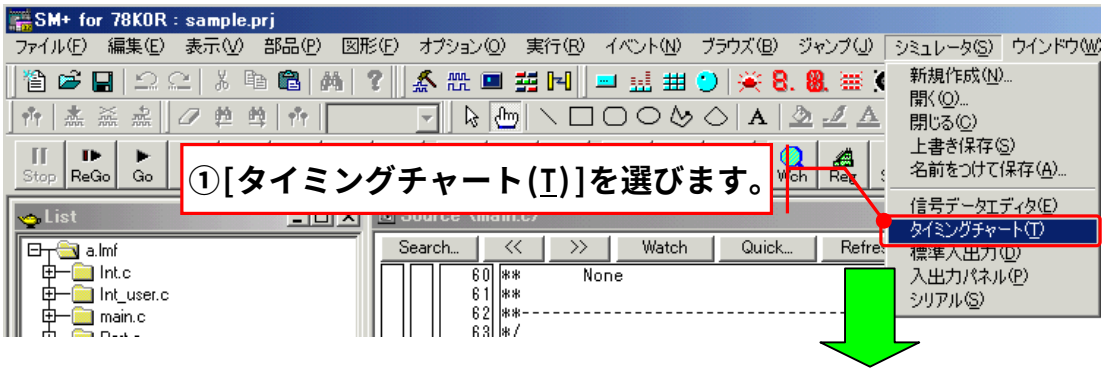


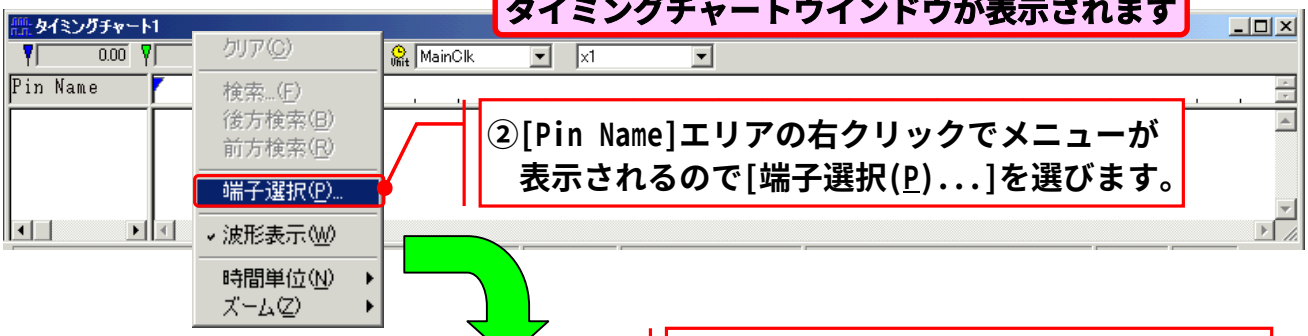


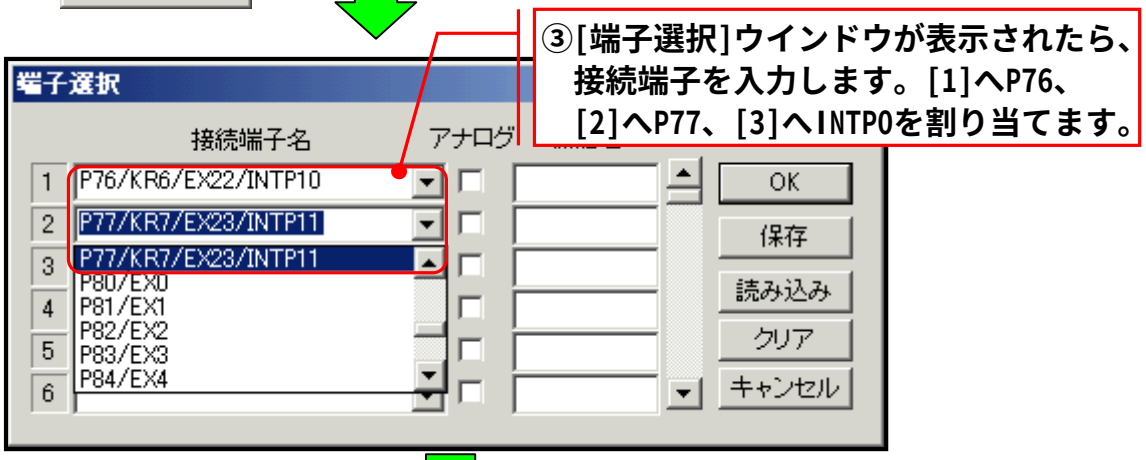
SM+でのデバッグ

1. タイミングチャートを設定します。

SM+では信号のタイミングチャートを表示する機能を持っています。

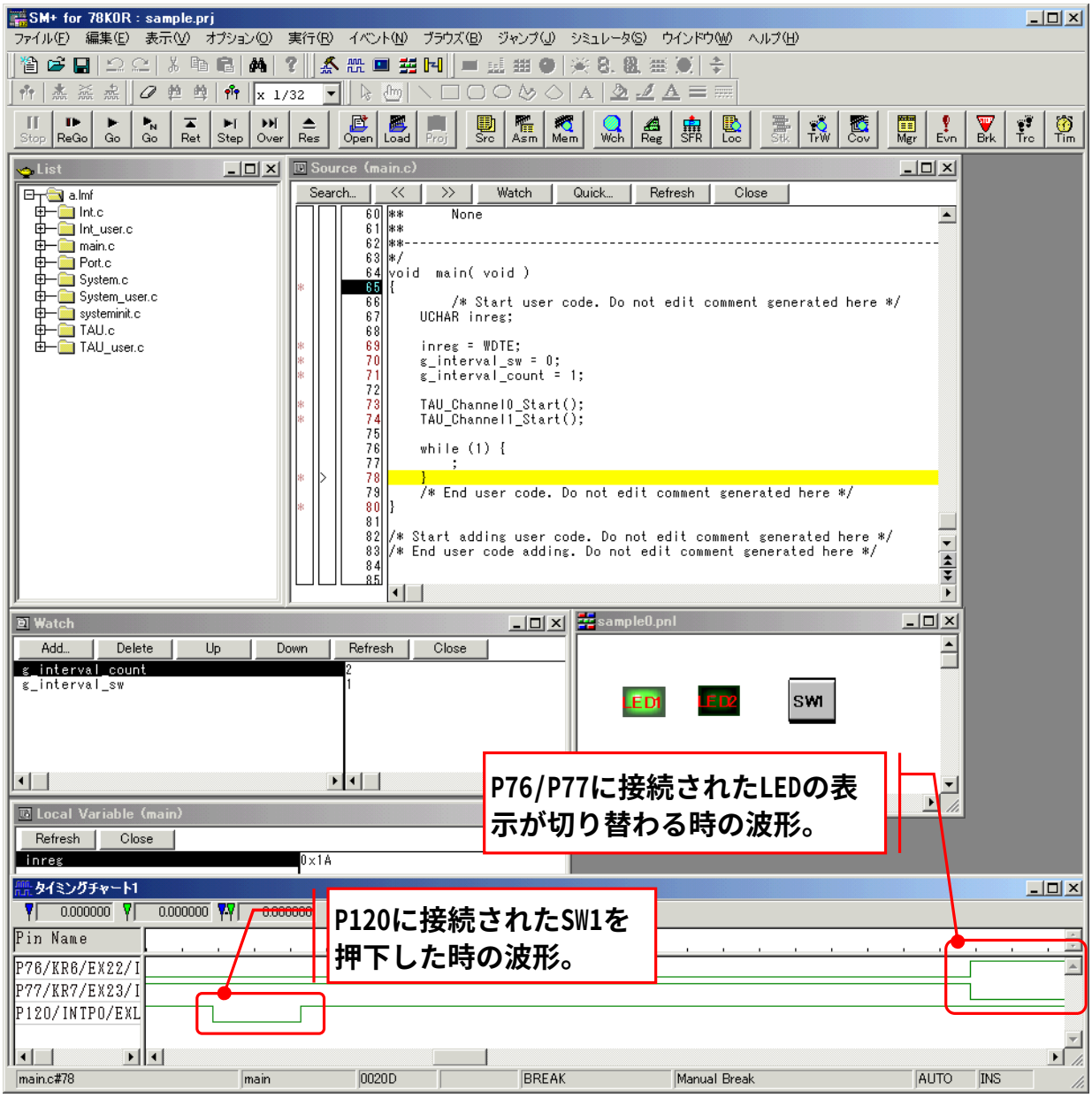






SM+でのデバッグ

m．タイミングチャートを表示した実行画面。



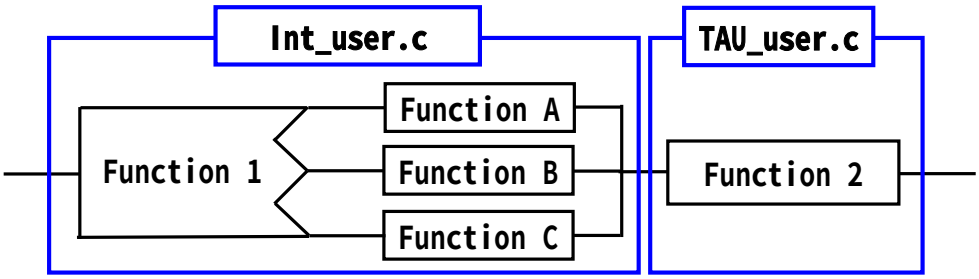
システム･シミュレータ(SM+)の基本的な使い方については、以上です。次章では「システム･シミュレータ」のもう少し高度なデバッグ方法について説明します。

更に高度なデバッグ機能

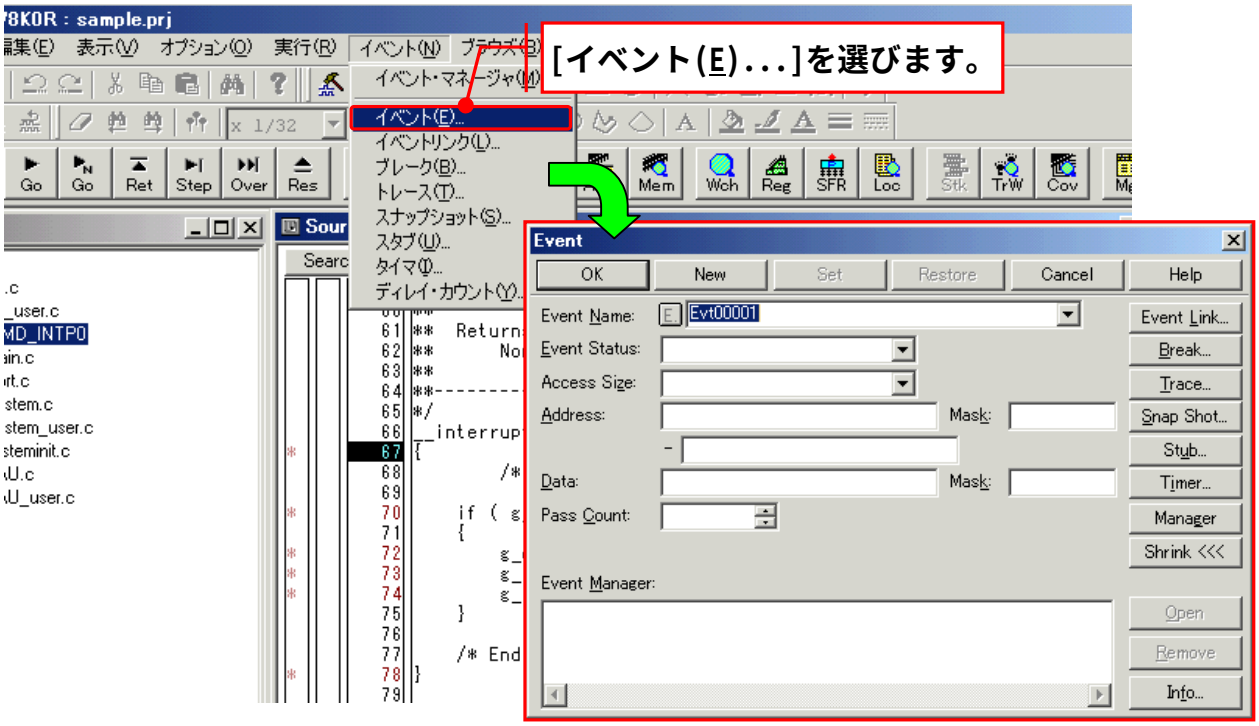
SM+では更に高度なデバッグ機能が多々あり、全てを伝えるのは難しいのですが、是非知って欲しい機能があります。それは「イベント･リンク」です。
「イベント」とは、「アドレス0x1000 番地をフェッチした」、「アドレス0x2000 番地にデータを書き込んだ」などのデバッグにおけるターゲット･システムの特定の状態を指しています。デバッガ(SM+、ID78K0R-QB)では、このようなイベントをブレーク、トレース等の各デバッグ機能のアクション･トリガとして利用しています。ブレークポイントで設定した”B”は**ブレーク･イベント**なのです。イベントの種類は他に**「トレース･イベント」、「タイマ･イベント」、「スタブ･イベント」、「スナップショット･イベント」**があります。
ここで説明するのは、デバッグ用にいくつかのイベントを定義し、ある条件下によってブレークする場合を説明します。

n. [Event]ダイアログを開く。

「イベント･リンク」で任意のイベントを通過した時のみブレークすることができます。この機能はプログラム･シーケンスの確認にとても有効です。
下記の例を説明します。プログラムは左から流れてきています。Function 1 で、ある値の判断を行い処理を分岐して Function A～C を実行します。最後に Function 2 を行うフローです。



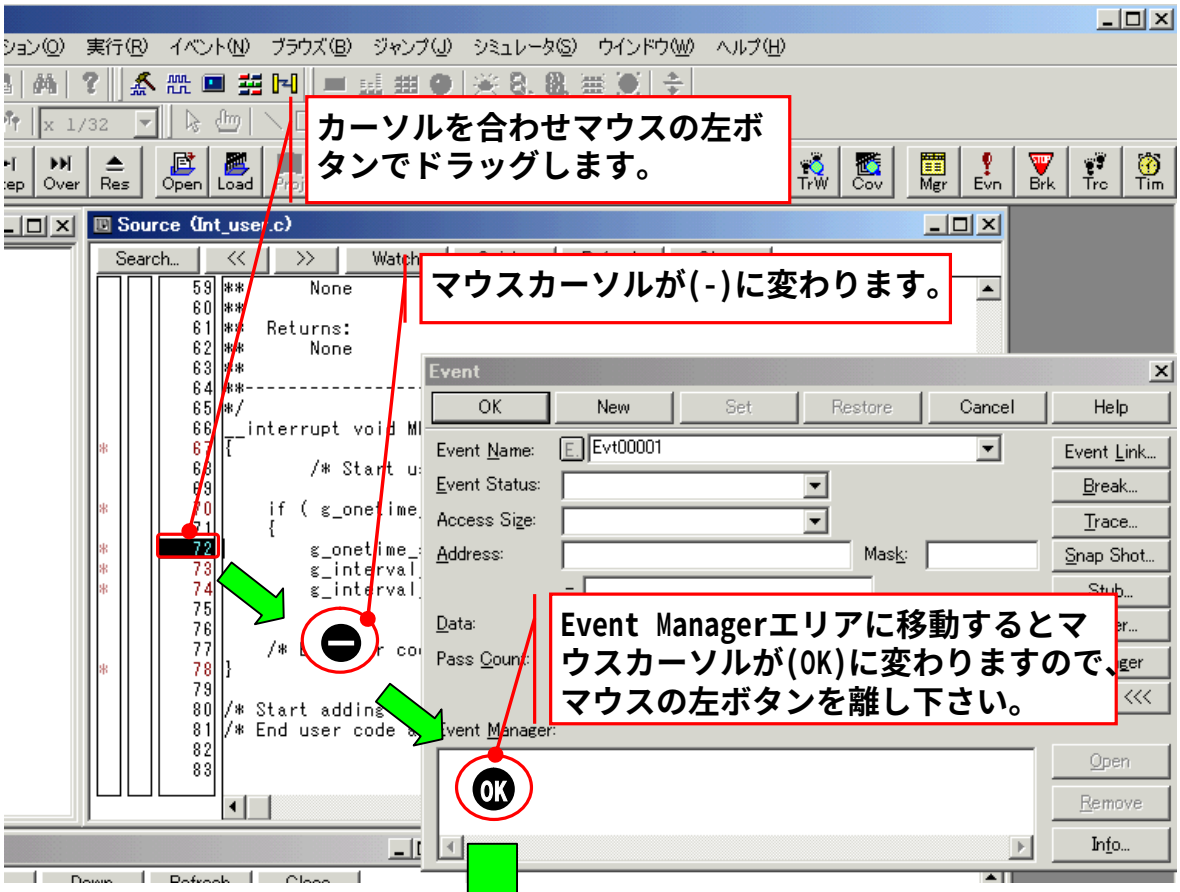
Function 2 でプログラムを止める場合、Function A → Function 2を通った場合だけプログラムを停止させることが可能です。実際にサンプル･プログラムで試してみます。まず、SM+上で[イベント]ダイアログを表示させます。

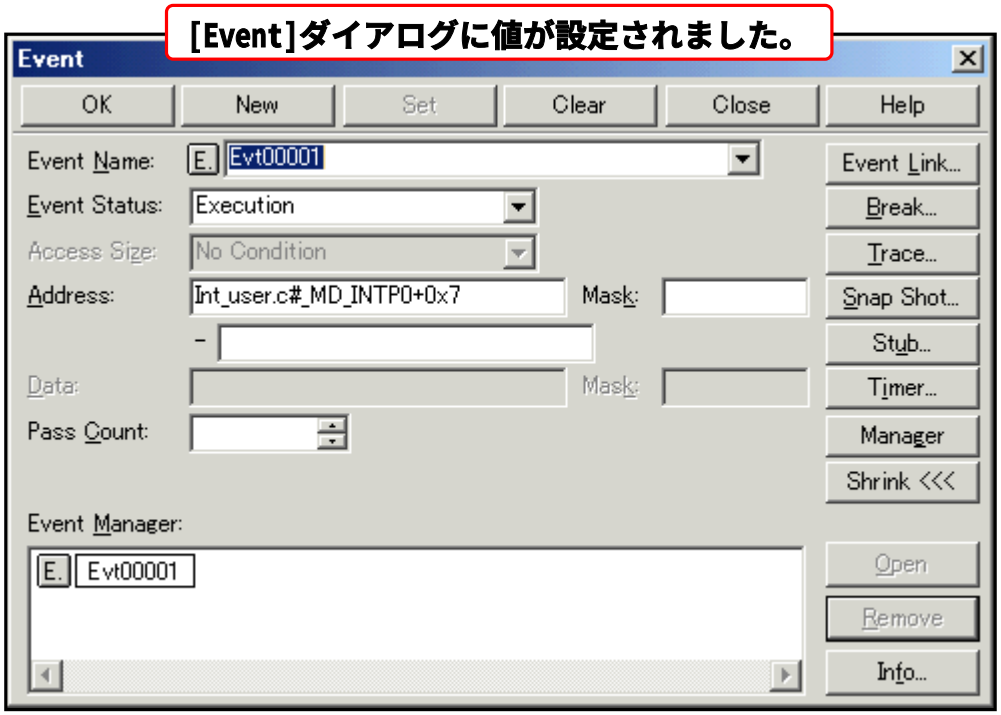


更に高度なデバッグ機能

o. イベントを設定します。

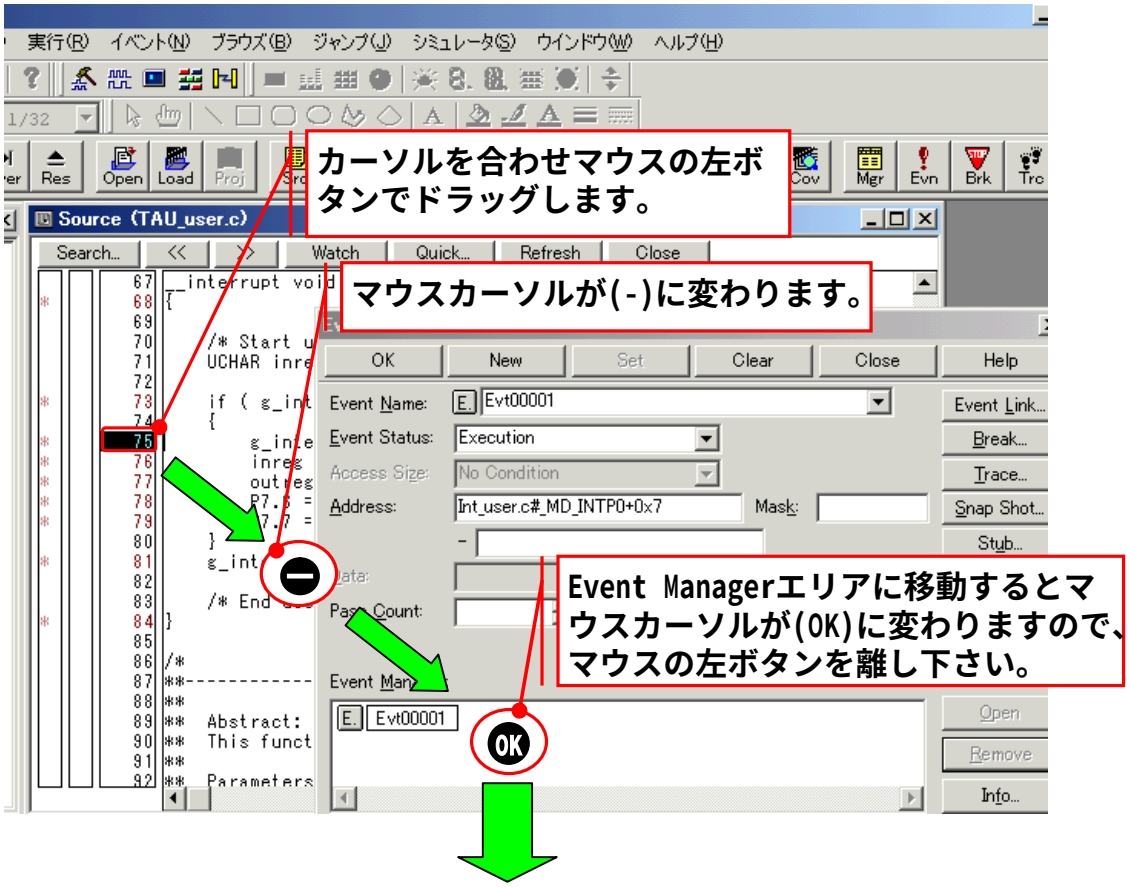
Int_user.cの72行目にカーソルを合わせ、左ボタンを押したままマウスを移動します。[Event]ダイアログのEvent Managerのエリアにカーソルを合わせて左ボタンを離して下さい。

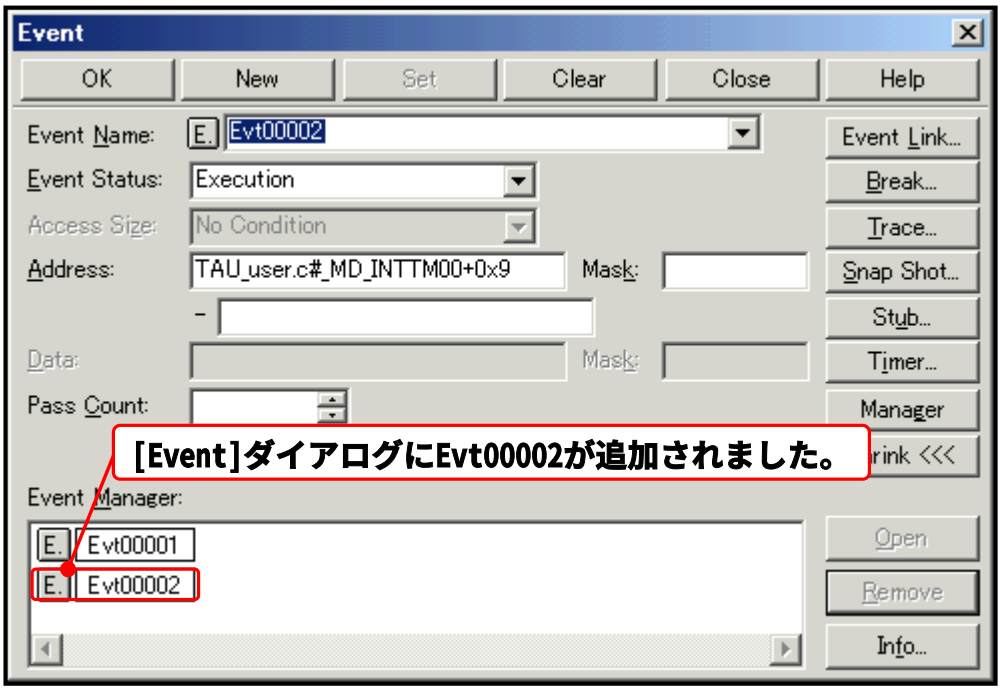




更に高度なデバッグ機能

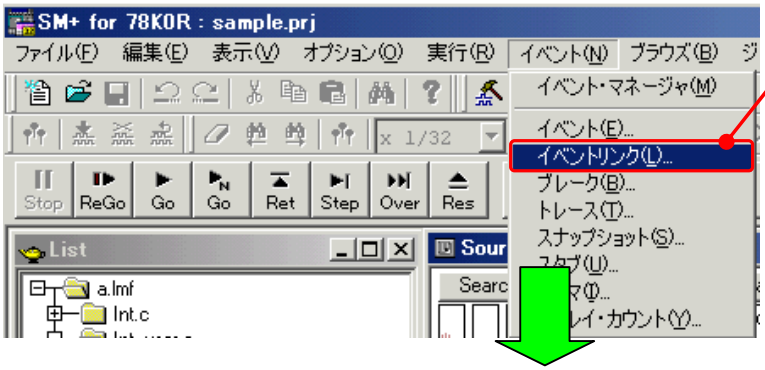
p. 次のイベントを設定します。
同様にTAU_user.cの75行目にカーソルを合わせて、イベントを作成します。



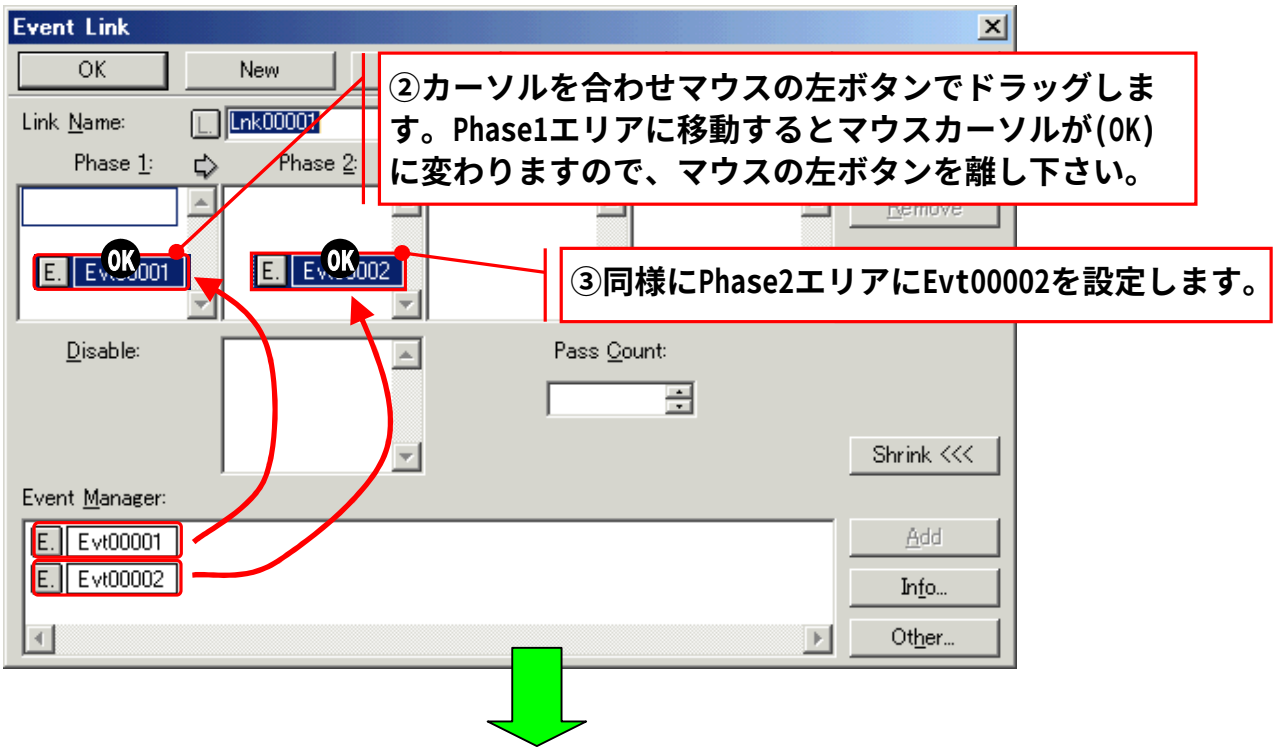


更に高度なデバッグ機能

q. イベント・リンクを設定します。
メニュー[イベント(N)] → [イベントリンク(L)...]で、[Event Link]ダイアログを表示して下さい。

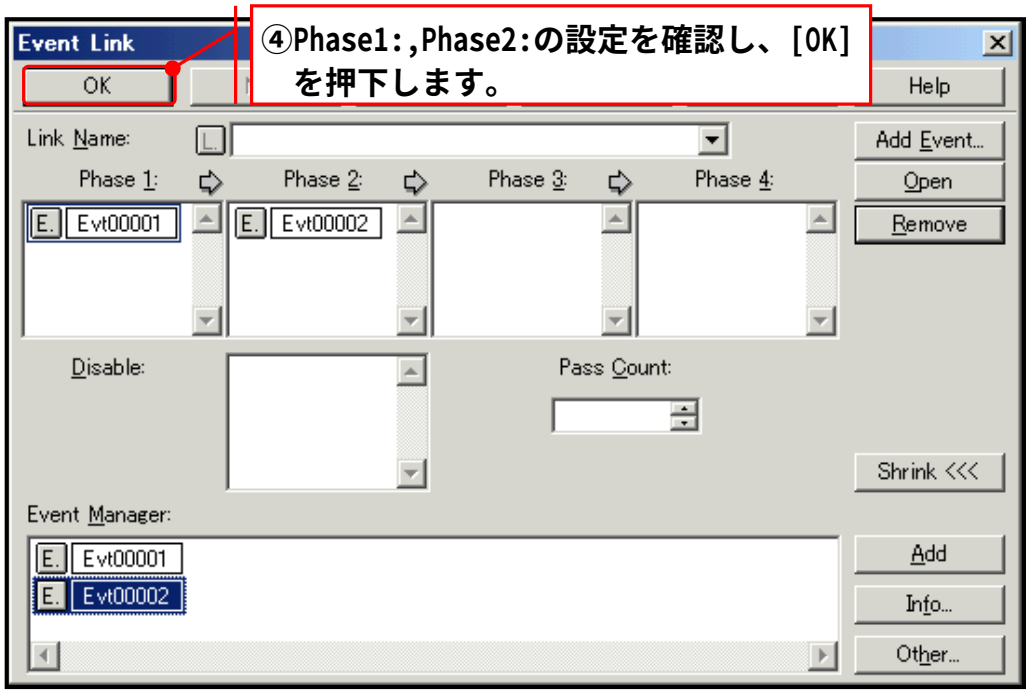


①[Event Link]ダイアログを開きます。



②カーソルを合わせマウスの左ボタンでドラッグします。Phase1エリアに移動するとマウスカーソルが(OK)に変わりますので、マウスの左ボタンを離し下さい。

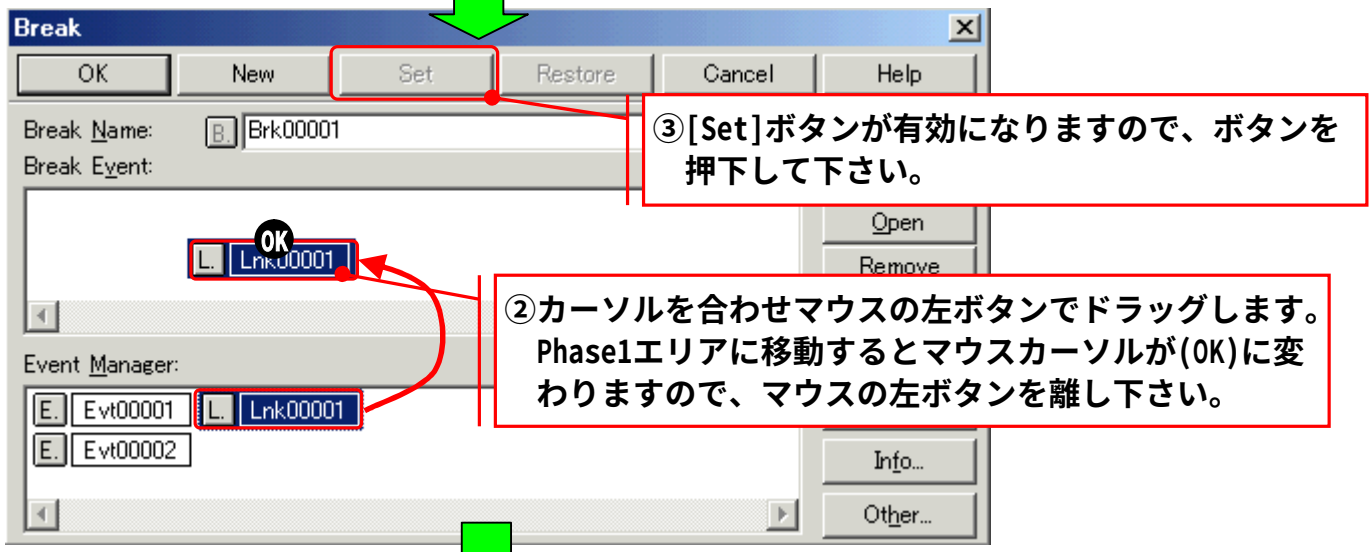
③同様にPhase2エリアにEvt00002を設定します。

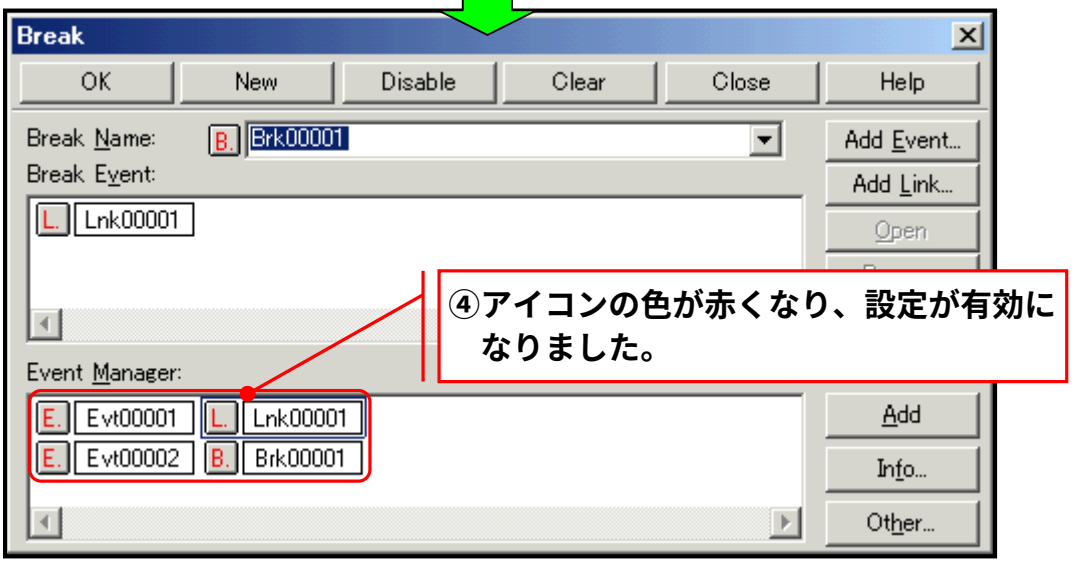


④Phase1:,Phase2:の設定を確認し、[OK]を押下します。

更に高度なデバッグ機能

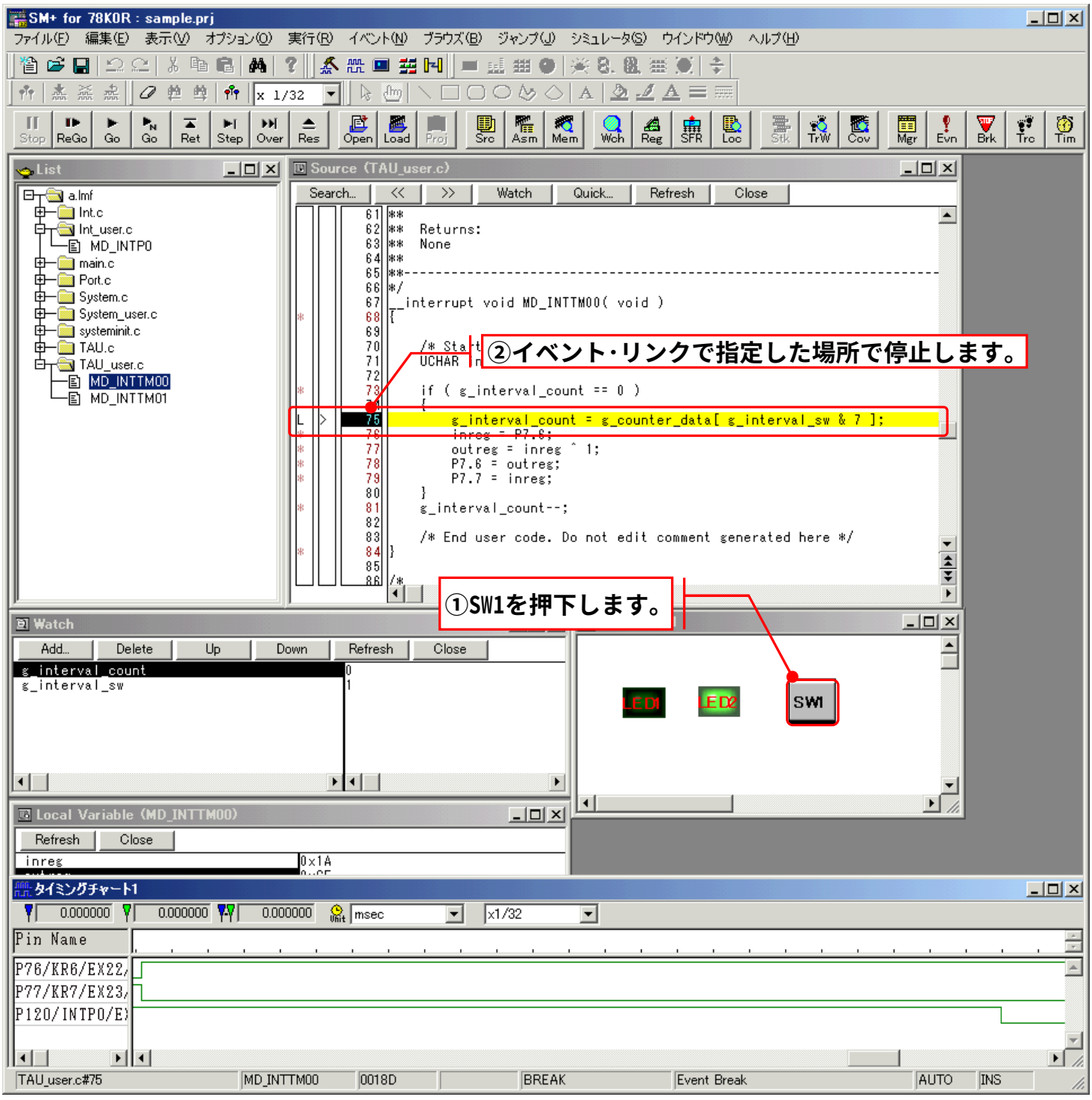
r. ブレークポイントを設定します。
メニュー[イベント(N)] → [ブレーク(B)...]で、[Break]ダイアログを表示して下さい。



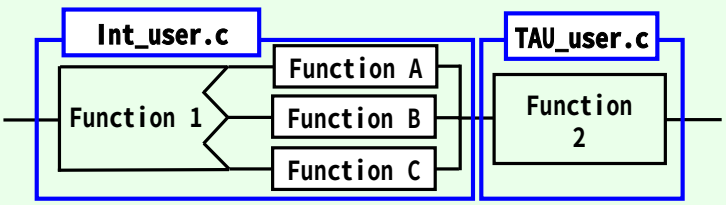


更に高度なデバッグ機能

s．実行して入出力パネルのSW1を押下します。
プログラムを実行してSW1を押下するとTAU_user.cの75行目で停止します。



以上のように、Int_user.c(Function1)を通り、TAU_user.c(Function2)で指定したイベントリンクで停止しました。単にTAU_user.c(Function2) にブレーク･ポイントを設定すると、毎回100ms毎に停止してしまうのですが、イベントリンクを使うことによってSW1を押した時だけ停止させることができました。

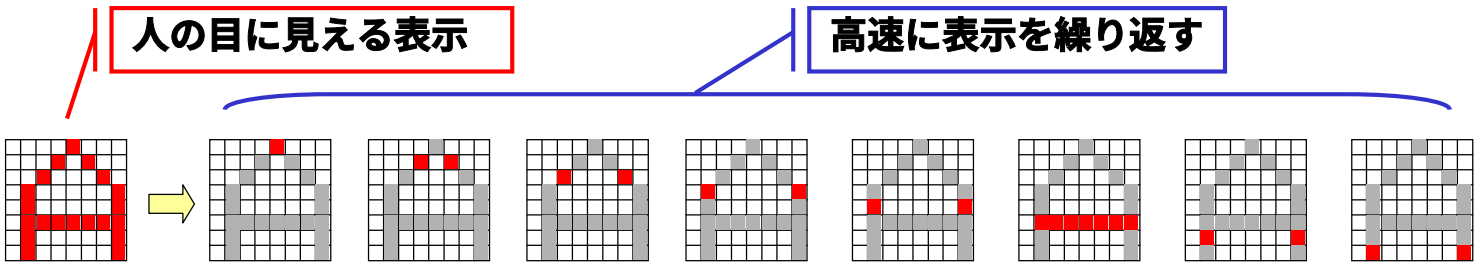


電光掲示板を作る

この章ではターゲット･システム作成例1を紹介します。ドットマトリクスLEDを使用した電光掲示板を作成します。マイコンに表示機能とSWがあればゲーム作成などにも応用可能です。ターゲット･システム作成例2では、電光掲示板を応用した簡単なゲームを作成しています。

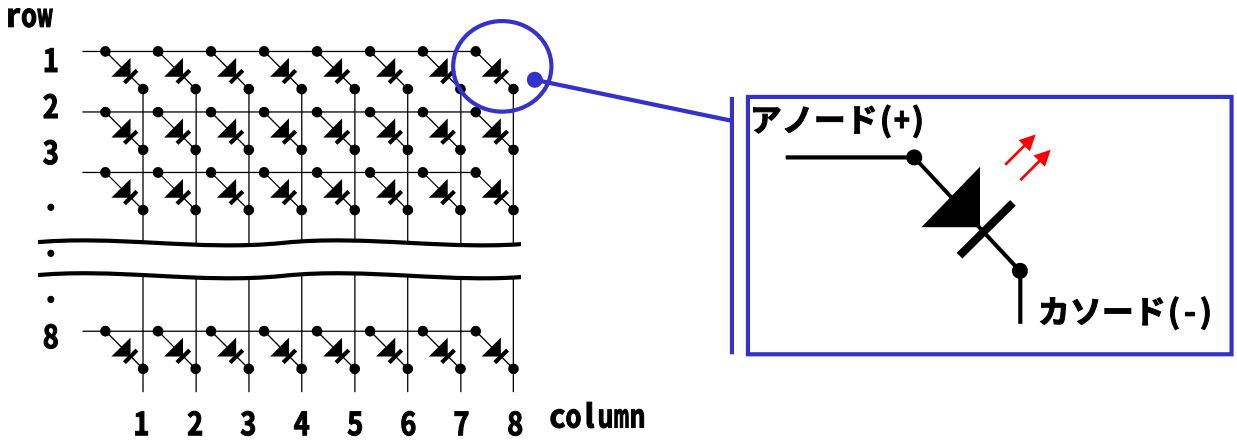
ターゲット･ボードを使用した回路を作成します。ここでは8x8のドットマトリクスLEDをダイナミック点灯させます。ダイナミック点灯とは表示を分割して(今回は8個のLED)行う方法です。表示を高速に順次繰り返す事によって人の目には全てが表示されているように見えます。
ダイナミック点灯には下記の特徴があります。
利点:常時点灯していないので消費電力が少ない。ポート制御が少なく済む。
欠点:スタティック(常時)点灯に比べて表示が暗い。

ダイナミック点灯の原理



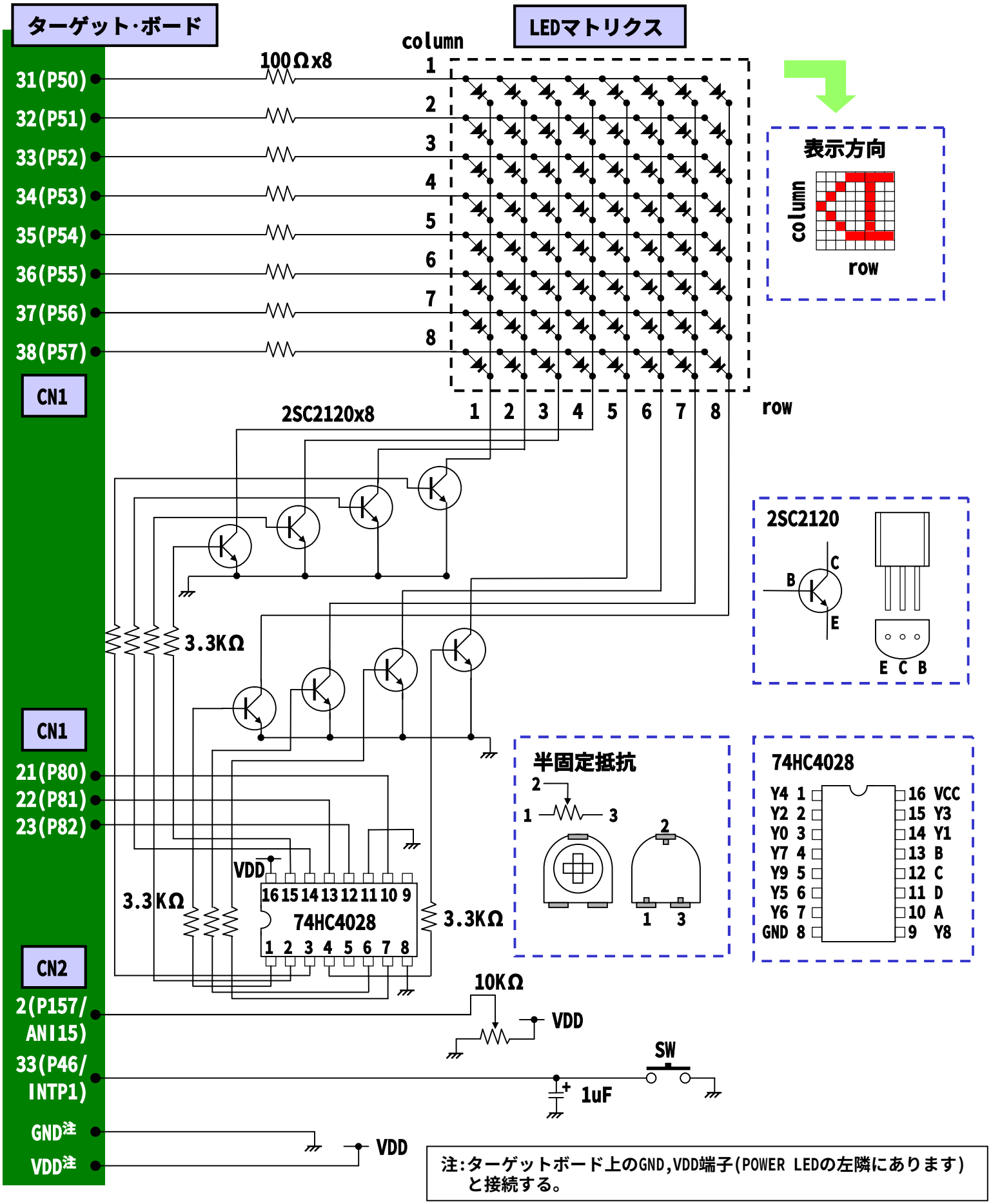
ワンポイント

ドットマトリクスLEDについて
ドットマトリクスLEDとはLEDがマトリクス(matrix)状に並んだ表示器です。8x8ならLEDが行(column) 8個、列(row) 8個が格子状に64個並んでいます。



例えば、上記の青丸内のLEDを点灯させるにはrow1に+(プラス)、column8に-(GND)を接続すれば点灯します。LEDを1つ点灯させるには通常10mA～20mA必要です(LEDの色、LEDの特性によって異なります)。これを+5Vで供給するには抵抗150Ω～300Ωを接続します。マトリクスLEDは1列に8個接続されていますので80mA～160mAが必要です。これを+5Vで供給するには抵抗37.5Ω～75Ωを接続します。しかし、マイコンの1ポートに80mA～160mAを流せませんので何らかの回路(トランジスタ制御など)が必要になります。

回路図



注:ターゲットボード上のGND,VDD端子(POWER LEDの左隣にあります)と接続する。

回路説明

ドットマトリクスLEDは8個のLEDを点灯させるため、マイコンで直接ドライブできませんのでトランジスタを使用します。
74HC4028はBCDコードを10進化します。ポート80～82からの出力を74HC4028のABCで受け、その結果をY0～Y7へ変換します。ドットマトリクスLEDのrowは0～7の値ですのでY8,Y9は不要です。そのため74HC4028のD入力はGNDに接します。
更にA/Dコンバータ(ANI15)を使用します。半固定抵抗の抵抗値によって電光掲示板の流れる速度を調整できるようにします。
INTP1に接続しているSWは表示切り替え用です。SWにはチャタリング防止回路を入れてあります。(内蔵のプルアップ抵抗を使い、コンデンサのみを使用した簡易なチャタリング防止です。)
電源はMINICUBE2より供給します。ただし供給可能な電流は100mAなので、使用する部品によって100mAを超える場合は「ターゲット作成例1」の[動作概要]のコラムを参照して外部電源を利用して下さい。

ワンポイント

シンク電流、ソース電流について
78K0R/KG3のポートはソース電流で10mA(端子合計80mA)、シンク電流で30mA(端子合計200mA)の容量があります(ポート番号によって異なります)。

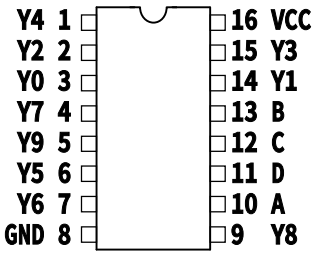
シンク電流(I_{OL})

ソース電流(I_{OH})

74HC4028について
BCD TO DECIMAL DECODERです。 下図のABCDの入力に応じてY0～Y9がH(ハイレベル)になります

BCD入力				DECIMAL出力									
A	B	C	D	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
L	L	L	L	H	L	L	L	L	L	L	L	L	L
H	L	L	L	L	H	L	L	L	L	L	L	L	L
L	H	L	L	L	L	H	L	L	L	L	L	L	L
H	H	L	L	L	L	L	H	L	L	L	L	L	L
L	L	H	L	L	L	L	L	H	L	L	L	L	L
H	L	H	L	L	L	L	L	L	H	L	L	L	L
L	H	H	L	L	L	L	L	L	L	H	L	L	L
H	H	H	L	L	L	L	L	L	L	L	H	L	L
L	L	L	H	L	L	L	L	L	L	L	L	H	L
H	L	L	H	L	L	L	L	L	L	L	L	L	H

74HC4028



プログラム設計

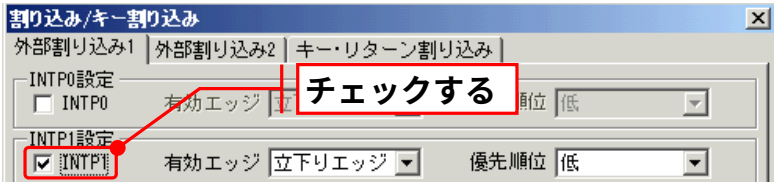
a. Applilet2を使いプログラムを設計します。

[動かしてみよう]の[Applilet2でプロジェクト作成]を参照して、プロジェクトを作成して下さい。ここではプロジェクト名を[extend]とします。[動かしてみよう]の[周辺機能設定(システム)]と同様に設定して下さい。下記の設定になります。

- 高速システム･クロックX1発振20MHzを使用
- オンチップ･デバッグ2線モード
- ウォッチドッグ･タイマは使用しない

b. [割り込み]を設定します。

[割り込み]ダイアログは、外部割り込みを設定します。

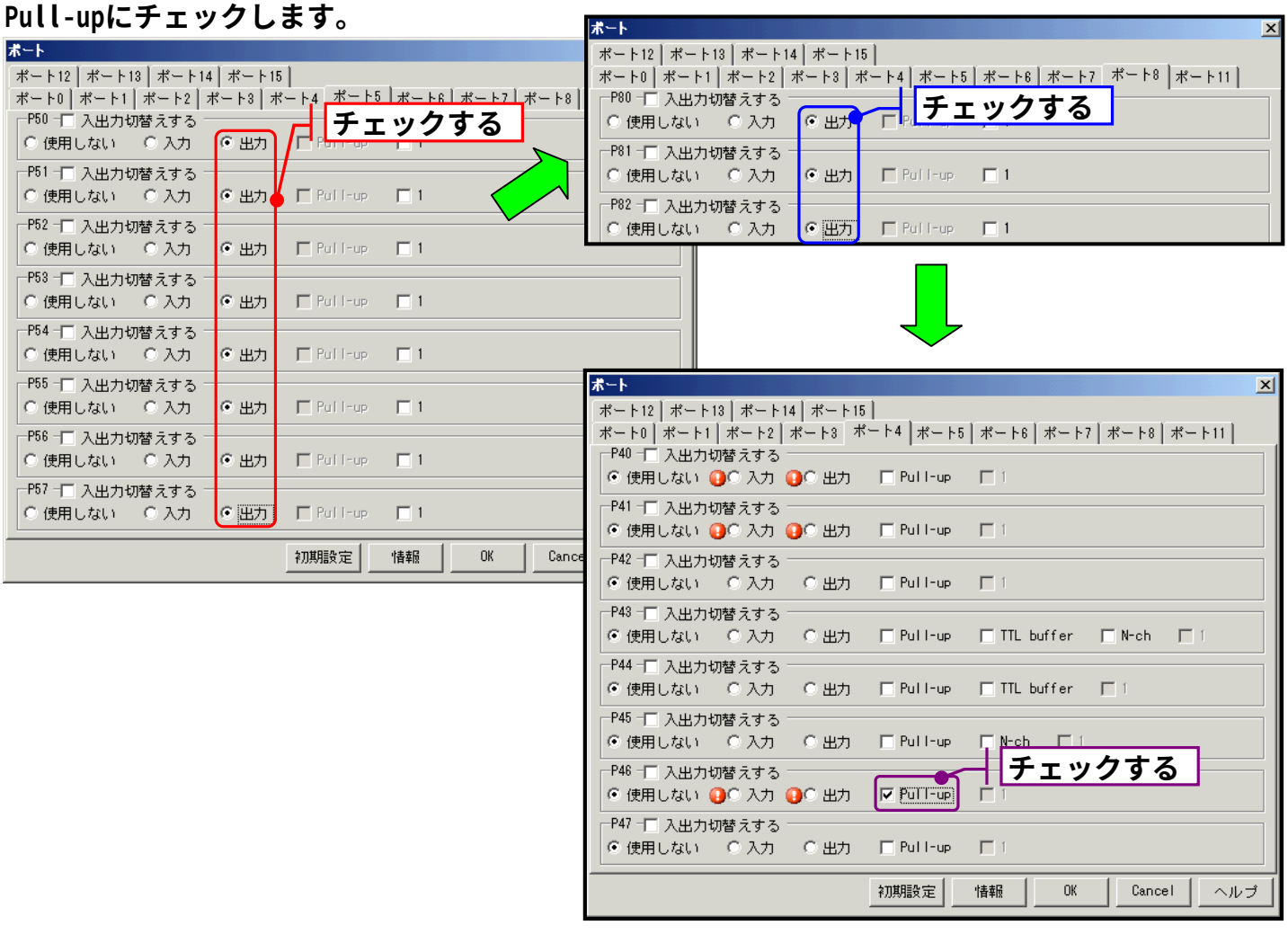


ワンポイント

INTP1兼用端子のP46の設定も必要です。INTP1を立ち下がりエッジで使用するので、信号をHigh、P46をPull-upしておく必要があります。抵抗を端子に接続してもよいのですが、マイコンの機能を利用して回路を省略しています。

c. [ポート]を設定します。

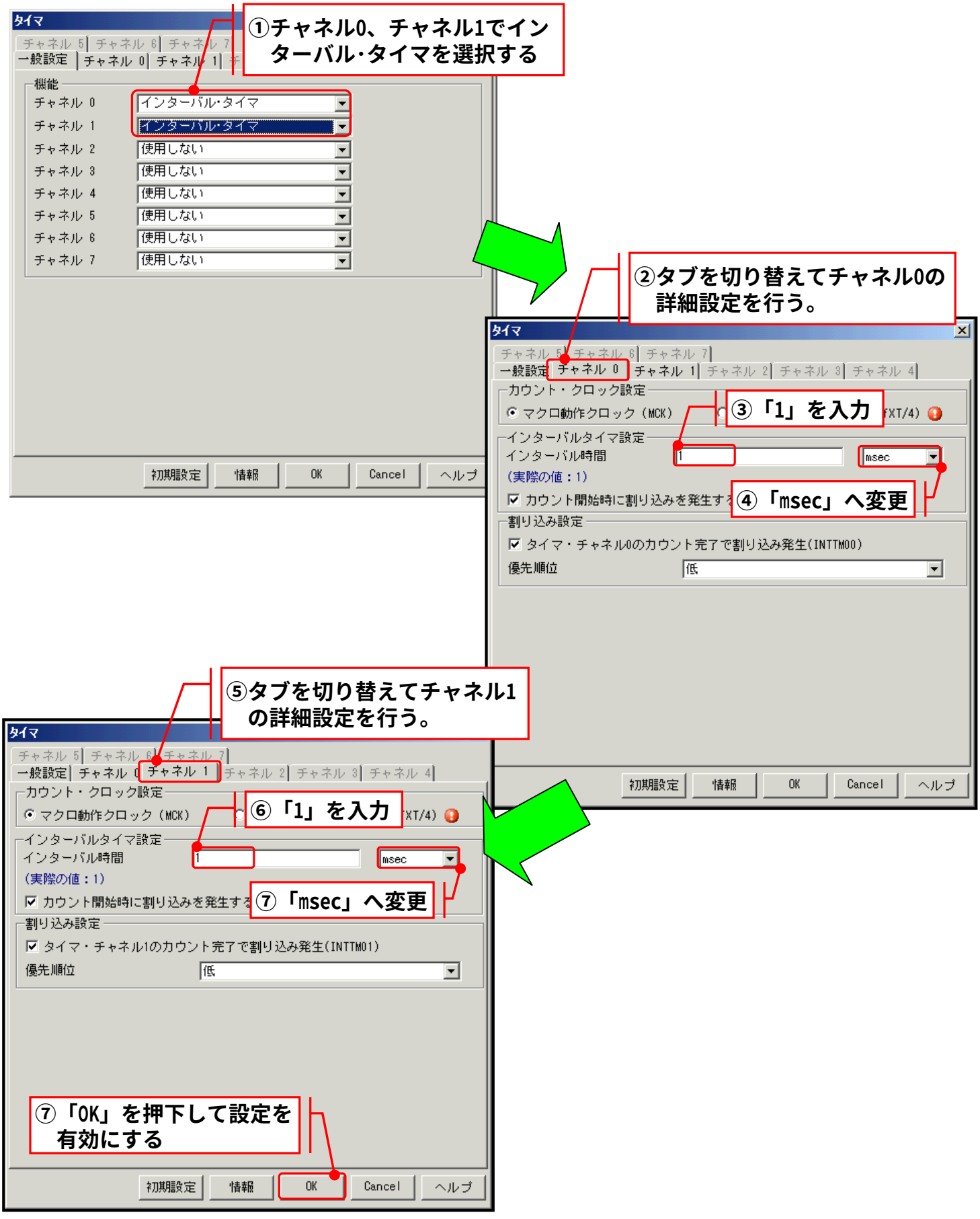
ポート5[P50～P57]の出力にチェック、ポート8[P80～P82]の出力にチェック、ポート4[P46]のPull-upにチェックします。



プログラム設計

d. [タイマ]を設定します。

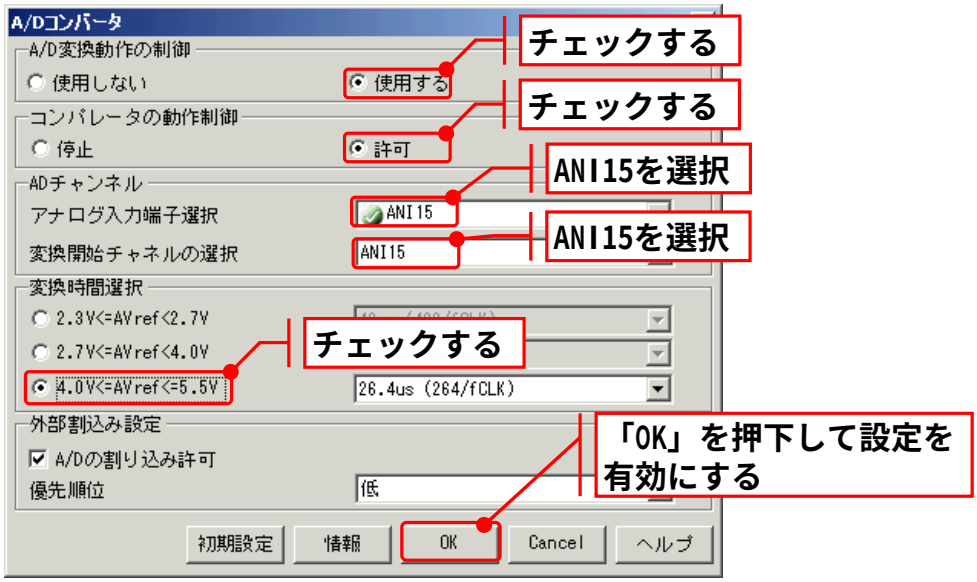
チャネル0、チャネル1にインターバル･タイマ1msを設定します。



プログラム設計

e. [A/Dコンバータ]を設定します。

A/Dを使用する、コンパレータの動作制御を許可、アナログ入力端子選択ANI15、変換開始チャネルの選択をANI15、変換時間選択を 4.0V<=AVref<=5.5Vにチェックします。



f. コード生成します。

[動かしてみよう]の[コード生成]を参考にコードを生成を行い、プログラムの編集をします。プログラムの編集については、[動かしてみよう]の[プログラムの編集]を参考にして下さい。

編集するファイルはApplilet2生成プログラム中の下記に示すファイルです。

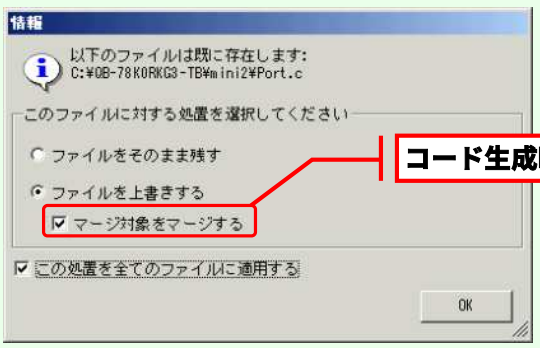
- user_define.h　ユーザが定義する定数などを書く
- main.c　ユーザのメイン関数を含む
- Int_user.c　ユーザが書く割り込み処理
- TAU_user.c　ユーザが書くタイマ割り込み処理
- Ad_user.c　ユーザが書くA/D変換終了の割り込み処理

プログラムを作成する時は、下記コメント間にプログラムを書いて下さい。

```
void function ( void )
{
/* Start user code. Do not edit comment generated here */


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

この場所にプログラムを書きます。Applilet2のコード生成で上書きされることはありません。

Applilet2でコード生成する際に、既存のコードに「マージ」することができます。上記以外の場所にプログラムを書くと、「ファイルを上書きする」にチェックしてもマージされず、コードが消去されます。



プログラムリスト

g. プログラムを編集して下さい。

下記に示す青字のコードを追加して下さい

user_define.h

リスト省略

```
#ifndef _MD_USER_DEF_
#define _MD_USER_DEF_
/*
************************************************
**  Macro define
************************************************
*/
/* Start user code for macro definition. Do not edit comment generated here */

#include "string.h"
#include "ctype.h"

#define D_MLED_CNT   2  /* Matrix LED number   */
#define D_MLED_ROW   8  /* Matrix LED row      */
#define D_FONT_WIDTH 6  /* font width          */
#define D_MAXTEXT   34  /* Max display of text */

/* End user code for macro definition. Do not edit comment generated here */
#endif
```

main.c(リスト1)

リスト省略

```
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "Port.h"
#include "TAU.h"
#include "Int.h"
#include "Ad.h"
#include "System.h"
/* Start user code for include definition. Do not edit comment generated here */

void disp_putfont( UCHAR font_ );
void initialize_text( void );

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

/*
***********************************************************************************
**  Global define
***********************************************************************************
*/

/* Start user code for global definition. Do not edit comment generated here */

const UCHAR g_font_numeric[] =
{ /* font data 0 - 9 */
  0x00, 0x0E, 0x11, 0x13, 0x15, 0x19, 0x11, 0x0E    /* 0 */
, 0x00, 0x04, 0x0C, 0x04, 0x04, 0x04, 0x04, 0x0E    /* 1 */
, 0x00, 0x0E, 0x11, 0x01, 0x0E, 0x10, 0x10, 0x1F    /* 2 */
, 0x00, 0x0E, 0x11, 0x01, 0x0E, 0x01, 0x11, 0x0E    /* 3 */
, 0x00, 0x02, 0x06, 0x0A, 0x12, 0x1F, 0x02, 0x02    /* 4 */
, 0x00, 0x1F, 0x10, 0x10, 0x1E, 0x01, 0x01, 0x1E    /* 5 */
, 0x00, 0x0F, 0x10, 0x10, 0x1E, 0x11, 0x11, 0x0E    /* 6 */
, 0x00, 0x1F, 0x01, 0x02, 0x04, 0x04, 0x04, 0x04    /* 7 */
, 0x00, 0x0E, 0x11, 0x11, 0x1E, 0x11, 0x11, 0x0E    /* 8 */
, 0x00, 0x0E, 0x11, 0x11, 0x0F, 0x01, 0x01, 0x0E    /* 9 */
};
```

プログラムリスト

main.c(リスト2)

```
const UCHAR g_font_alphabet[] =
{ /* font data A - Z */
  0x00, 0x04, 0x0A, 0x0A, 0x11, 0x1F, 0x11, 0x11    /* A */
, 0x00, 0x1E, 0x11, 0x11, 0x1E, 0x11, 0x11, 0x1E    /* B */
, 0x00, 0x0E, 0x11, 0x10, 0x10, 0x10, 0x11, 0x0E    /* C */
, 0x00, 0x1C, 0x12, 0x11, 0x11, 0x11, 0x12, 0x1C    /* D */
, 0x00, 0x1F, 0x10, 0x10, 0x1E, 0x10, 0x10, 0x1F    /* E */
, 0x00, 0x1F, 0x10, 0x10, 0x1E, 0x10, 0x10, 0x10    /* F */
, 0x00, 0x0E, 0x11, 0x10, 0x17, 0x11, 0x11, 0x0E    /* G */
, 0x00, 0x11, 0x11, 0x11, 0x1F, 0x11, 0x11, 0x11    /* H */
, 0x00, 0x0E, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0E    /* I */
, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0x12, 0x0C    /* J */
, 0x00, 0x11, 0x12, 0x14, 0x18, 0x14, 0x12, 0x11    /* K */
, 0x00, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x1F    /* L */
, 0x00, 0x11, 0x11, 0x1B, 0x15, 0x11, 0x11, 0x11    /* M */
, 0x00, 0x11, 0x11, 0x19, 0x15, 0x13, 0x11, 0x11    /* N */
, 0x00, 0x0E, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0E    /* O */
, 0x00, 0x1E, 0x11, 0x11, 0x1E, 0x10, 0x10, 0x10    /* P */
, 0x00, 0x0E, 0x11, 0x11, 0x11, 0x15, 0x13, 0x0F    /* Q */
, 0x00, 0x1E, 0x11, 0x11, 0x1E, 0x14, 0x12, 0x11    /* R */
, 0x00, 0x0E, 0x11, 0x10, 0x0E, 0x01, 0x11, 0x0E    /* S */
, 0x00, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04    /* T */
, 0x00, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0E    /* U */
, 0x00, 0x11, 0x11, 0x11, 0x11, 0x0A, 0x0A, 0x04    /* V */
, 0x00, 0x11, 0x15, 0x15, 0x15, 0x15, 0x15, 0x0A    /* W */
, 0x00, 0x11, 0x11, 0x0A, 0x04, 0x0A, 0x11, 0x11    /* X */
, 0x00, 0x11, 0x11, 0x0A, 0x04, 0x04, 0x04, 0x04    /* Y */
, 0x00, 0x1F, 0x01, 0x02, 0x04, 0x08, 0x10, 0x1F    /* Z */
};

const UCHAR g_textdata[][ D_MAXTEXT ] =
{   /* text data */
  "THIS IS 78KORKG3 TARGET BOARD    "
, "DISPLAY CHANGES IF SW IS PUSHED  "
, "THIS IS PROGRAM SAMPLE OF NECEL  "
, "abcdefghijklmnopqrstuvwxyz       "
, "0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6"
, 0
};

/* Interval timer counter  */
UINT g_timer0_counter = 0;
UINT g_timer1_counter = 0;

/* control for text */
USHORT g_text_speed = 0;
UCHAR g_text_sw = 0;
UCHAR g_text_cnt = 0;
UINT g_text_scrollcnt = 0;

/* g_matrix_ram[0] is real vram. [1] is buffer vram. */
UCHAR g_matrix_ram[ D_MLED_CNT ][ D_MLED_ROW ];

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void initialize_value( void )
{

    memset( g_matrix_ram, 0x00, sizeof( g_matrix_ram ));
    g_timer0_counter = 0;
    g_timer1_counter = 0;
    g_text_sw = 0;
}
```

プログラムリスト

main.c(リスト3)

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void initialize_text( void )
{
    UCHAR font;

    memset( g_matrix_ram, 0x00, sizeof( g_matrix_ram ));
    g_text_scrollcnt = 0;
    g_text_cnt = 0;

    font = g_textdata[ g_text_sw ][ 0 ];
    disp_putfont( font );
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* font_ is displayed to virtual vram.                                         */
void disp_putfont( UCHAR font_ )
{
    int i;
    UCHAR cnvf, fnt;

    cnvf = (UCHAR)(toupper( font_ ));
    if ( cnvf>='A' && cnvf<='Z' )
    {   /* Alphabet font */
        cnvf -='A';
        for ( i=0; i<D_MLED_ROW; i++ )
        {
            fnt = g_font_alphabet[ cnvf*8 + i ];
            g_matrix_ram[ 1 ][ i ] = (fnt << 2);
        }
    }
    else if ( cnvf>='0' && cnvf<='9' )
    {   /* Numeric font */
        cnvf -='0';
        for ( i=0; i<D_MLED_ROW; i++ )
        {
            fnt = g_font_numeric[ cnvf*8 + i ];
            g_matrix_ram[ 1 ][ i ] = (fnt << 2);
        }
    }
    else
    {   /* Null font */
        for ( i=0; i<D_MLED_ROW; i++ )
        {
            g_matrix_ram[ 1 ][ i ] = 0;
        }
    }
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 1 dot is scrolled.                                                          */
void disp_move1dot(void )
{
    int i;
    UCHAR vtmp, vtmp2;
     for ( i=0; i<D_MLED_ROW; i++ )
    {
        vtmp = ( g_matrix_ram[ 0 ][ i ] & 0x7f) << 1;
        vtmp2 = ( g_matrix_ram[ 1 ][ i ] & 0x80) >> 7;
        g_matrix_ram[ 0 ][ i ] = vtmp + vtmp2;
        vtmp2 = ( g_matrix_ram[ 1 ][ i ] & 0x7f) << 1;
        g_matrix_ram[ 1 ][ i ] = vtmp2;
    }
}

/* End user code for global definition. Do not edit comment generated here */
```

プログラムリスト

main.c(リスト4)

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

    /* Target-Board LED off */
    P7.6 = 1;
    P7.7 = 1;

    initialize_value();

    initialize_text();

    /* start timer */
    TAU_Channel0_Start();
    TAU_Channel1_Start();

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

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

プログラムリスト

Int_user.c

```
                        リスト省略
#pragma interrupt INTP1 MD_INTP1
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "Int.h"
/* Start user code for include definition. Do not edit comment generated here */

extern void initialize_text( void );

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

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

extern UCHAR g_text_sw;
extern UCHAR g_matrix_ram[ D_MLED_CNT ][ D_MLED_ROW ];
extern const UCHAR g_textdata[][ D_MAXTEXT ];

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

/*
**-----------------------------------------------------------------------------
**
**  Abstract:
**                      This function is INTP1 interrupt service routine.
**
**  Parameters:
**                      None
**
**  Returns:
**                      None
**
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTP1( void )
{
    /* Start user code. Do not edit comment generated here */

    g_text_sw++;
    if (  g_textdata[ g_text_sw ][ 0 ] == 0 )
    {
        g_text_sw = 0;
    }

    memset( g_matrix_ram, 0x00, sizeof( g_matrix_ram ));
    initialize_text();

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

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

プログラムリスト

TAU_user.c(リスト1)

```
                        リスト省略
#pragma interrupt INTTM00 MD_INTTM00
#pragma interrupt INTTM01 MD_INTTM01
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "TAU.h"
/* Start user code for include definition. Do not edit comment generated here */
#include "ad.h"

extern void disp_putfont( UCHAR font_ );
extern void disp_move1dot( void );
/* End user code for include definition. Do not edit comment generated here */

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

extern UINT   g_timer0_counter;
extern UINT   g_timer1_counter;
extern UCHAR  g_text_sw;
extern UCHAR  g_text_cnt;
extern UINT   g_text_scrollcnt;
extern USHORT g_text_speed;
extern UCHAR  g_matrix_ram[ D_MLED_CNT ][ D_MLED_ROW ];
extern const UCHAR g_textdata[][ D_MAXTEXT ];
UCHAR g_line;

/* End user code for global definition. 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 */

    UCHAR row, line;

    /* Matrix LED line disp */
    row = (UCHAR)(g_timer0_counter) & 7;
    /* display column */
    P5 = 0;
    line = g_matrix_ram[ 0 ][ row ];
    g_line = line;
    P5 = line;
   /* display row */
    P8.0 = ( row & 1 );
    P8.1 = (( row & 2 ) >> 1);
    P8.2 = (( row & 4 ) >> 2);

    g_timer0_counter++;

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

プログラムリスト

TAU_user.c(リスト2)

```
/*
**-----------------------------------------------------------------------------
**
**  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 */

    UCHAR font;

    AD_Start();

    g_timer1_counter++;
    if ( g_timer1_counter > g_text_speed )
    {
        if ( ( g_text_scrollcnt % D_FONT_WIDTH ) == 0 )
        {   /* next font */
            font = g_textdata[ g_text_sw ][ g_text_cnt ];
            disp_putfont( font );

            g_text_cnt++;
            if ( g_text_cnt >= D_MAXTEXT )
                g_text_cnt = 0;
        }

        /* scroll */
        disp_move1dot();

        g_text_scrollcnt++;
        g_timer1_counter = 0;
    }

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

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

プログラムリスト

Ad_user.c

リスト省略

```
#pragma interrupt INTAD MD_INTAD
/*
************************************************************************
** Include files
************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "Ad.h"
/* Start user code for include definition. Do not edit comment generated here */
/* End user code for include definition. Do not edit comment generated here */

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

extern USHORT g_text_speed;

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


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

    AD_Stop();

    AD_Read( &adval );
    g_text_speed = adval/2 + 1;
    if ( g_text_speed > 500 )
    {
        g_text_speed = adval;
    }
    /* End user code. Do not edit comment generated here */
}

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

プログラム説明

h. プログラムで使用している関数について説明します。

main.c

```
const UCHAR g_font_numeric[] /* font data 0 - 9 */
```
マトリックスLEDに表示する数字フォントデータです

```
const UCHAR g_font_alphabet[] /* font data A - Z */
```
マトリックスLEDに表示する英字(大文字のみ)フォントデータです

```
const UCHAR g_textdata[][D_MAXTEXT] /* text data */
```
マトリックスLEDに表示するテキストデータです

```
UINT g_timer0_counter = 0;
UINT g_timer1_counter = 0;
```
タイマ割り込み時に+1されるカウンタです

```
UCHAR g_text_sw = 0;
```
表示するテキストを選択します

```
UCHAR g_text_cnt = 0;
```
表示されているテキストが何文字目かを示します

```
UINT  g_text_scrollcnt = 0;
```
表示されている文字が何ドット移動したかを示します

```
USHORT g_text_speed = 0;
```
スクロールするスピードを示します

```
UCHAR g_matrix_ram[ D_MLED_CNT ][ D_MLED_ROW ];
```
マトリックスLEDに表示するデータです。8x16ドット分ありますが実際に表示されるのは8x8ドットです

```
void initialize_value( void )
void initialize_text( void )
```
変数初期化を行う関数です

```
void disp_putfont( UCHAR font_ )
```
パラメータ font_(asciiコード)で指定された文字をマトリックスLEDの非表示領域へ出力します

```
void disp_move1dot(void )
```
マトリックスLEDの表示を1ドット移動(スクロール)します

TAU_user.c

```
__interrupt void MD_INTTM00()
```
1msec毎に呼ばれる関数です。
マトリックスLEDにg_matrix_ram よりデータを読み込み1ライン表示します

```
__interrupt void MD_INTTM01()
```
1msec毎に呼ばれる関数です。下記の処理を行います。
・A/D変換を開始します
・移動のチェックを行い、1文字分のスクロールが終了したら disp_putfont を呼び出し、指定された文字をマトリックスLEDの非表示領域へ出力します。
そうでなければ disp_move1dot を呼び出し、1ドット分スクロールします。

Ad_user.c

```
__interrupt void MD_INTAD( void )
```
A/D変換終了時に呼ばれる関数です。
A/D値を読み込み g_text_speed へ反映させます

Int_user.c

```
__interrupt void MD_INTP1( void )
```
外部割り込みINTP1に呼ばれる関数です。
SWが押下されたら g_text_sw を+1し、表示するテキストを変更します。

user_define.h

```
#define D_MLED_CNT   2  /* Matrix LED number   */
```
マトリックスLEDの数を定義します。実際は1つですが仮想的に2つ持っていることにしています。

```
#define D_MLED_ROW   8  /* Matrix LED row      */
```
マトリックスLEDのROWの数

```
#define D_FONT_WIDTH 6  /* font width          */
```
1文字のフォントの幅

```
#define D_MAXTEXT   34  /* Max display of text */
```
表示するメッセージの長さ

動作概要

g.全体の処理

プログラムの動作として、下記の5つに分けられます

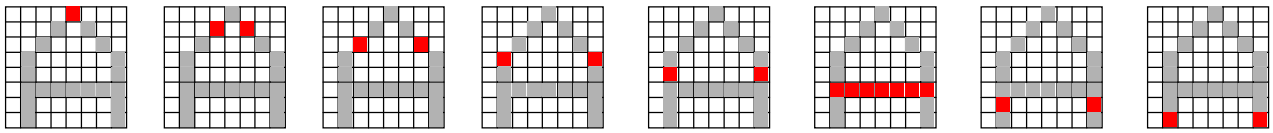
1.メイン処理
2.タイマ00(インターバル･タイマとして使用)
3.タイマ01(インターバル･タイマとして使用)
4.A/Dコンバータ完了割り込み
5.INTP1外部割り込み

1.メイン処理 void main(void)

変数/マトリックスLEDの初期化、タイマ00/01を開始します。
処理としては無限ループとなります。

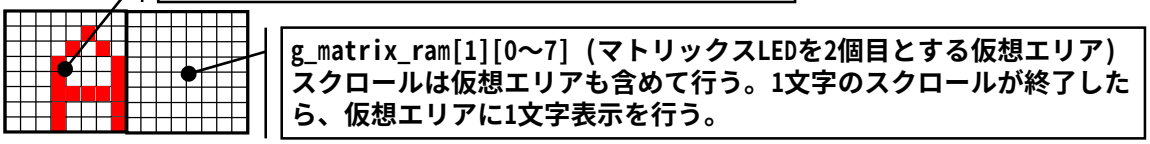
2.タイマ00 __interrupt void MD_INTTM00( )

1msec毎に起動します。g_matrix_ram[0][0～7]の1byteをマトリックスLEDの1ラインへ出力します。
呼ばれる毎に1ライン表示を行います。(下図の点灯が1msec毎に行われます)

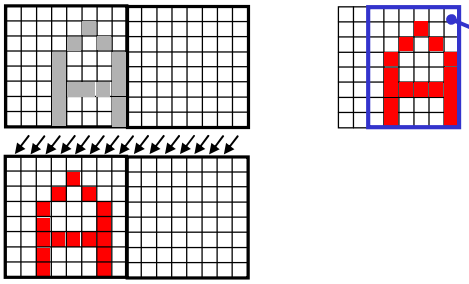


3.タイマ01 __interrupt void MD_INTTM01( )

1msec毎に起動します。
A/D変換の開始を行います。A/D変換が完了するとA/D変換終了割り込みが発生します。
A/D変換の結果は g_text_speed へ反映されるので、それを元にスクロールスピードを決めます
タイマ01は呼ばれる毎に g_timer1_counter を+1していますので、g_text_speedとg_timer1_counter の値を比較してスクロールを行うか行わないかを判断しています。
スクロールを行う場合は2通りの処理があります。
a.マトリックスLEDへ表示されているデータを1ドット左へ移動する
b.1文字のスクロールが終了したので新たな文字を表示する

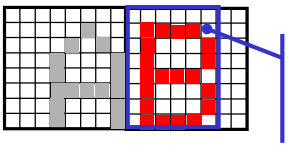


a.マトリックスLEDへ表示されているデータを1ドット左へ移動する



1フォントは縦8x横6ドットで構成されています。a.の処理は1文字のスクロール処理(6ドット左へ移動)のうち5ドット左移動までの分を行います。
1文字のスクロール処理開始時は b.1文字のスクロールが終了したので新たな文字を表示する の処理を行います。

b.1文字のスクロールが終了したので新たな文字を表示する



1フォントは縦8x横6ドットで構成されています。1文字のスクロール開始時にはマトリックスLEDを2個目とする仮想エリアへ1文字出力します。

動作概要

4.A/D変換終了割り込み　__interrupt void MD_INTAD(void)

タイマ01で開始されたA/D変換が終了時に呼ばれます。
A/D変換の中止を行い、変換結果を読み込みます。
結果は g_text_speed へ代入されます。g_text_speed の値は 0 以外を想定していますので、念の為変換結果に +1 を行います。表示するスピードが調整しやすいよう変換結果を変えています。

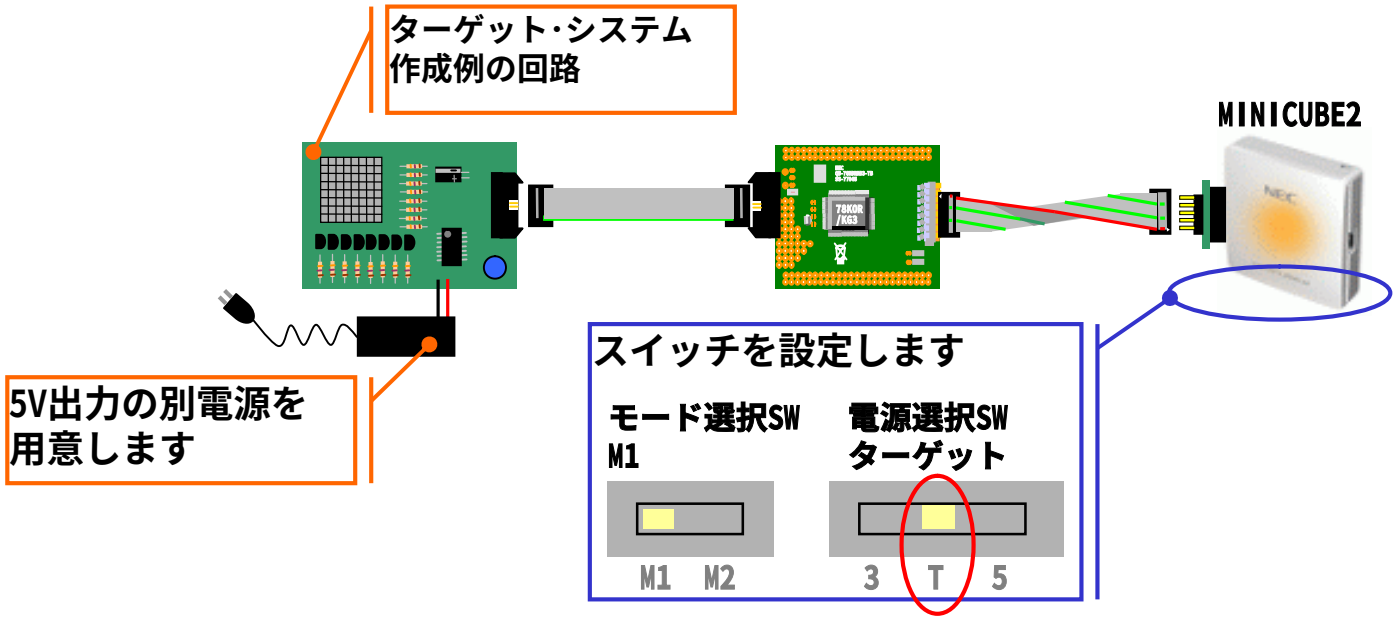
5.INTP1外部割り込み __interrupt void MD_INTP1(void)

INTP1端子がLOWになった場合(SWが押下された場合)に呼ばれます。
スクロールするテキストを変えます。表示させるテキストは1文が32文字で構成されています。
g_textdata[][D_MAXTEXT]の内容を書き換えればマトリックスLEDへ表示させる事ができますが、表示するデータは必ず32byteで構成して下さい。

ワンポイント

電源供給について

MINICUBE2より電源供給する場合5V供給で最大定格100mAです。ターゲット･システム作成例の回路で使用する部品によっては100mAを超える場合があります。その場合は5V電源を別に用意して下さい。ターゲット･システム側の電源を使用する場合はMINICUBE2の「電源選択SW」を「T」に設定して下さい。



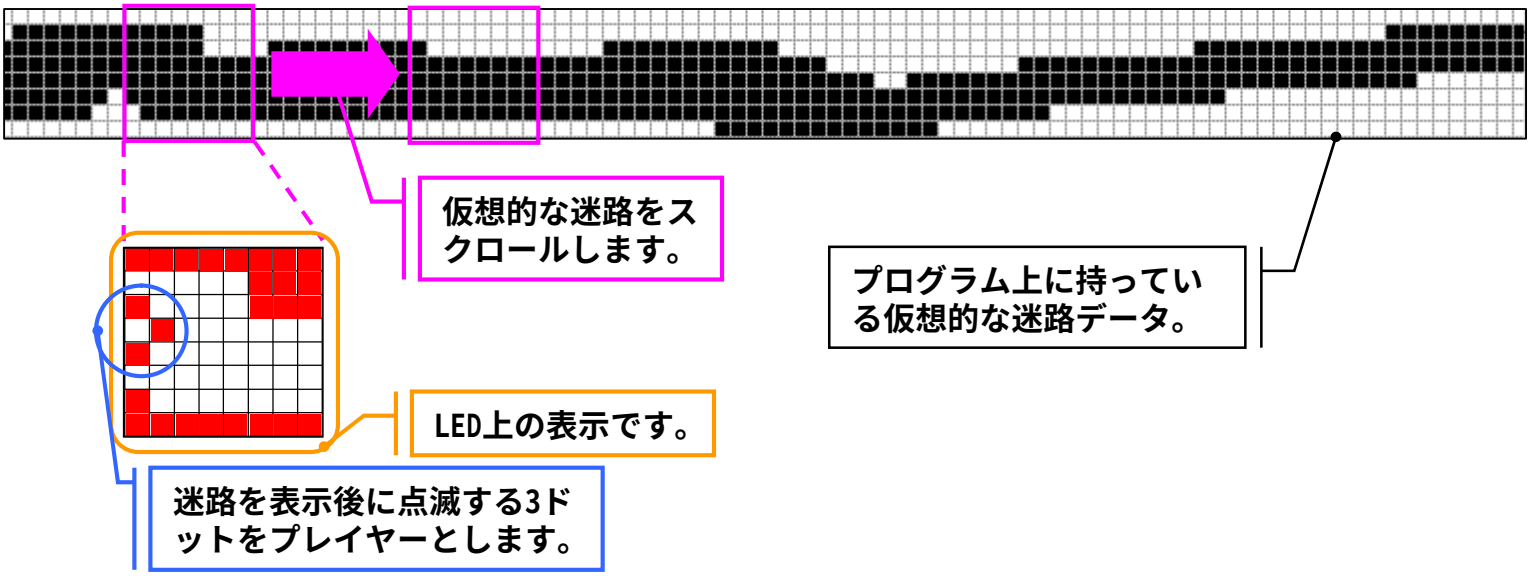
下記の順番で接続を行って下さい。
1.MINICUBE2のSWを設定する
3.16pinターゲット･ケーブルをターゲット･ボードへ接続する
4.ターゲット･システム回路とターゲット･ボードを接続する
5. MINICUBE2とPC本体を接続する(MINICUBE2のLEDが白点滅します)
6.ターゲット･システム回路へ電源を供給する(MINICUBE2のLEDが白点灯します)

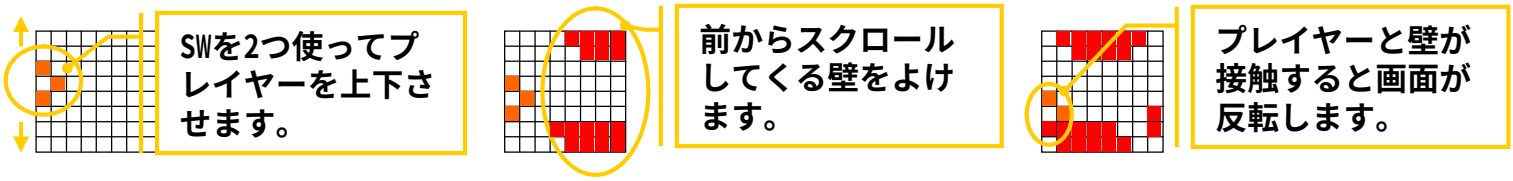
ゲームを作る

この章ではターゲット･システム作成例2を紹介します。ターゲット作成例1(電光掲示板)の回路を流用し、SWを1つ追加しました。電光掲示板では文字をスクロールさせましたが、今度は文字ではなく迷路をスクロールさせます。ただそれだけではゲーム性に乏しいのでプレイヤーを操作して障害物をよけるゲームにします。ハードが同じでもソフトウエアによって、いろいろ応用が広がることを体験して下さい。

ゲーム概要

プログラム中に仮想的な迷路データを作ります。白い部分を壁としてドットマトリクスLEDへ表示します。プレイヤーをLEDの左側に3ドットで表現しています。このスクロールする迷路をプレイヤーを操作して避けて先に進むというゲームです。

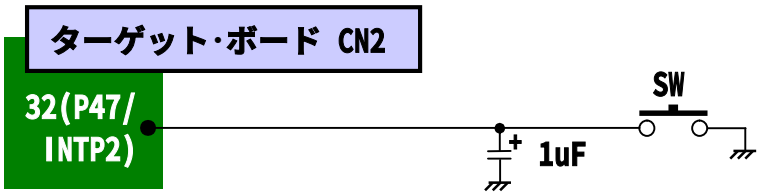




壁のスクロール速度はA/Dに接続した半固定抵抗で調整可能です。これで、ゲームの難易度を調整します。プレイヤーと壁は同じ色(LED1色のみ)で表示しますが、プレイヤーは点滅するので壁と判別可能です。

追加回路

ターゲット作成例1の回路にSWを一つ追加します。



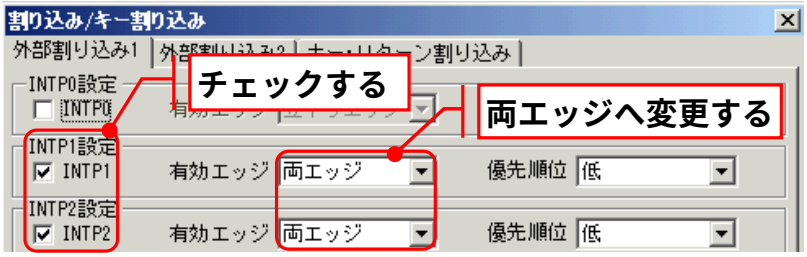
プログラム設計

a. Applilet2を使いプログラムを設計します。

[ターゲット作成例1]の[プログラム設計]を参照して同じ設定をして下さい。ここでは、異なる設定部分のみを説明します。

b. [割り込み]を設定します。

INTP1、INTP2の設定をします。

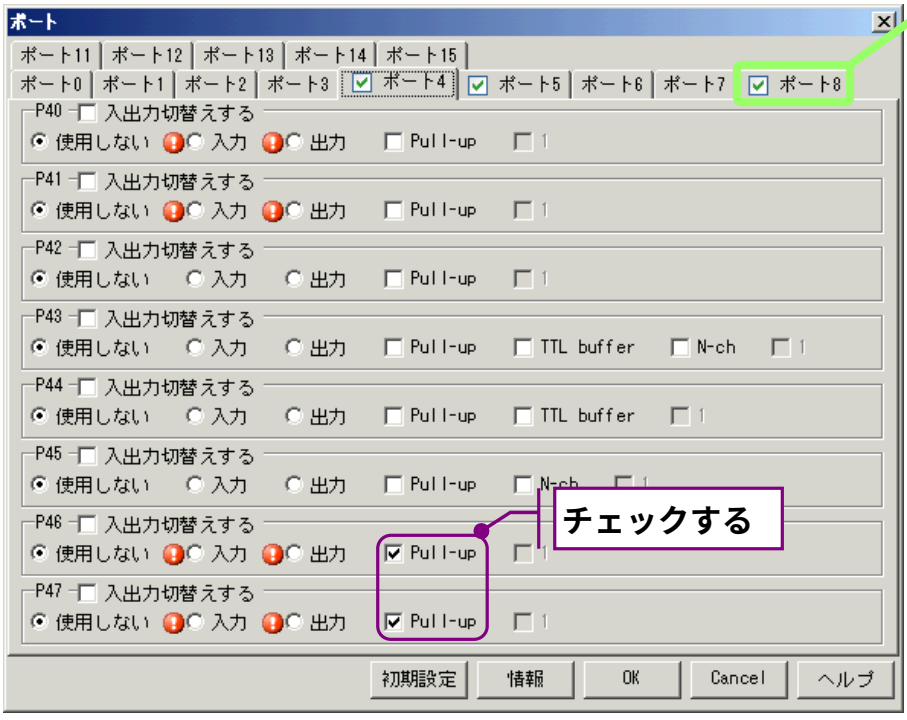


ワンポイント

INTP1兼用端子のP46の設定、INTP2兼用端子のP47の設定も必要です。

c. [ポート]を設定します。

ポート5[P50～P57]、ポート8[P80～P82]以外に設定の追加があります。ポート4[P46、P47]のPull-upをチェックします。



ワンポイント

設定を行っているポートにはチェックアイコン☑ が表示されます。

プログラムリスト

d. プログラムを編集して下さい。
下記に示す青字のコードを追加して下さい

user_define.h

```
            リスト省略
#ifndef _MD_USER_DEF_
#define _MD_USER_DEF_
/*
************************************************
**  Macro define
************************************************
*/
/* Start user code for macro definition. Do not edit comment generated here */

#include "string.h"
#include "ctype.h"

#define D_MLED_CNT   2  /* Matrix LED number   */
#define D_MLED_ROW   8  /* Matrix LED row      */
#define D_FONT_WIDTH 8  /* font width          */

/* End user code for macro definition. Do not edit comment generated here */
#endif
```

main.c(リスト1)

```
            リスト省略
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "Port.h"
#include "TAU.h"
#include "Int.h"
#include "Ad.h"
#include "System.h"
/* Start user code for include definition. Do not edit comment generated here */
/* End user code for include definition. Do not edit comment generated here */

/*
***********************************************************************************
**  Global define
***********************************************************************************
*/

/* Start user code for global definition. Do not edit comment generated here */

const UCHAR g_maze_data[] =
{
                0x08,0xab,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xe0,0x00,0x00,
  0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xf0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x07,0xf8,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x00,0x00,0x02,0x02,0x00,0xff,
  0x00,0x03,0xff,0xf8,0x00,0xff,0xfc,0x00,0x0f,0xfc,0x0f,0xf0,0x00,0x00,0x1f,0xff,
  0xf8,0x00,0x7f,0xc0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

                0x00,0x00,0x08,0x0e,0x06,0x3e,0x30,0x83,0xe0,0x1e,0x1f,0x80,0x70,
  0x0e,0x00,0x1f,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xf8,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x3f,0xff,0xfe,0x00,0x00,0x00,0xc0,0x00,0x06,0x00,0xc0,0x00,
  0x00,0x00,0x00,0x60,0x0c,0x00,0x03,0xe0,0x03,0xe0,0x00,0xe0,0x00,0x00,0x00,0x00,
  0x07,0xe0,0x00,0x00,0x3c,0x00,0xc0,0x00,0x20,0x20,0x10,0x00,0x02,0x02,0x00,0xff,
  0x00,0x03,0xff,0xf8,0x00,0x7f,0xfc,0x00,0x0f,0xfc,0x03,0xe0,0x03,0xff,0xff,0xfe,
  0x00,0x00,0x07,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x40,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00,0x40,0x00,0x00,0x41,
```

プログラムリスト

main.c(リスト2)

```
                0x00,0x00,0x00,0x00,0x00,0x0c,0x00,0x00,0x00,0x04,0x0f,0x00,0x20,
  0x04,0x00,0x1e,0x00,0x7f,0xf0,0x01,0xff,0xff,0xff,0x80,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xc0,0x01,0x86,0x00,0xc0,0x60,
  0x0c,0x03,0x00,0x60,0x0c,0x00,0x03,0xe0,0x07,0xe0,0x00,0xc0,0x07,0xe0,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x20,0x10,0x02,0x02,0x02,0x00,0x30,
  0x00,0x00,0xff,0xf0,0x00,0x1f,0xf8,0x00,0x07,0xfc,0x00,0x00,0xff,0xff,0xff,0xe0,
  0x00,0x00,0x00,0x00,0x00,0x0f,0x80,0x00,0xe0,0x00,0x40,0x00,0x00,0x00,0x40,0x04,
  0x00,0x00,0x04,0x00,0x00,0x10,0x08,0x00,0x04,0x10,0x02,0x04,0x00,0x88,0x00,0x00,

                0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x06,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x3f,0xfc,0x00,0x00,0x00,0x00,0x03,0xff,0xff,
  0xf0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x80,0x00,0x01,0x80,0x00,0x00,0x60,
  0x0c,0x03,0x00,0x03,0x00,0x00,0x03,0xe0,0x03,0xc0,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x3f,0x80,0x00,0x00,0x20,0x22,0x21,0x11,0x02,0x00,0x02,0x00,0x00,
  0x00,0x00,0x1f,0xe1,0xfe,0x07,0xf8,0x00,0x03,0xf8,0x00,0x00,0x1f,0xf0,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x06,0x00,0x00,0x00,0x00,0x00,0x80,0x00,0x40,0x00,0x80,
  0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x11,0x01,0x01,0x00,0x00,0x10,0x00,0x04,0x04,

                0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x06,0x00,0x00,0x00,0x01,0xff,0xff,0xff,0xff,
  0xf8,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x80,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x06,0x03,0x00,0x07,0x80,0x00,0x00,0x00,0xe0,0x00,0x00,0x00,0x00,0x7f,
  0xe0,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x02,0x21,0x01,0x00,0x20,0x00,0x00,0x00,
  0x00,0x00,0x03,0xc1,0xff,0x01,0xf0,0x00,0x03,0xf0,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x02,0x00,0x00,
  0x00,0x02,0x00,0x04,0x00,0x01,0x00,0x00,0x40,0x00,0x08,0x00,0x00,0x00,0x40,0x00,

                0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x02,
  0x00,0x04,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1f,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x00,0x18,0x0c,0x03,
  0x00,0x30,0x06,0x03,0x00,0x07,0x80,0x00,0xc0,0x00,0xe0,0x01,0x00,0x07,0xe0,0x00,
  0x00,0x01,0xfc,0x00,0x00,0xfc,0x04,0x02,0x02,0x01,0x01,0x10,0x20,0x40,0x18,0x00,
  0x0f,0x80,0x00,0x01,0xff,0x00,0x00,0x1f,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0xf8,0x00,0x03,0xf0,0x00,0xc2,0x00,0x00,0x00,0x00,0x00,0x00,
  0x84,0x00,0x00,0x80,0x00,0x00,0x00,0x40,0x00,0x10,0x00,0x01,0x00,0x00,0x00,0x20,

                0x00,0x00,0x40,0x80,0xc0,0x61,0x8e,0x38,0x0f,0x00,0x00,0x1c,0x07,
  0x00,0x0e,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0xff,0xf0,0x1f,0x80,0x00,0x00,0x00,0x30,0x18,0x18,0x0c,0x03,
  0x00,0x30,0x60,0x00,0x03,0x07,0x80,0x00,0xe0,0x00,0xe0,0x01,0xc0,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x04,0x02,0x02,0x01,0x00,0x10,0x20,0x40,0x7f,0x00,
  0x0f,0xe0,0x00,0x03,0xff,0x00,0x00,0x7f,0x00,0x00,0x0f,0xc0,0x00,0x00,0x00,0x3f,
  0xff,0xfc,0x00,0x00,0xff,0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x00,
  0x00,0x00,0x10,0x00,0x10,0x00,0x00,0x01,0x02,0x00,0x40,0x40,0x02,0x08,0x00,0x00,

                0x12,0xaf,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0xff,0xff,0xe0,0x00,0x7f,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
  0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xfc,0x00,0x00,0x18,0x00,0x00,0x03,
  0x00,0x00,0x60,0x00,0x03,0x00,0x00,0x00,0xe0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x02,0x01,0x00,0x00,0x00,0x40,0xff,0x80,
  0x1f,0xe0,0x00,0x03,0xff,0x00,0x00,0xff,0x00,0x00,0x1f,0xe0,0x00,0x00,0xff,0xff,
  0xff,0xfe,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x04,0x02

};


/* Interval timer counter  */
UINT g_timer0_counter = 0;
UINT g_timer1_counter = 0;
UINT g_bump_counter;

/* control for maze */
USHORT g_maze_scrollcnt = 0;
USHORT g_maze_speed = 0;

/* INTP1, INTP2 status */
UCHAR g_intp1_sw;
UCHAR g_intp2_sw;
```

プログラムリスト

main.c(リスト3)

```
/* g_matrix_ram[0] is real vram. [1] is buffer vram. */
UCHAR g_matrix_ram[ D_MLED_CNT ][ D_MLED_ROW ];
UCHAR g_matrix_realram[ D_MLED_ROW ];
UCHAR g_plane_location;
BOOL  g_plane_bump;

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void initialize_value( void )
{

    memset( g_matrix_ram, 0x00, sizeof( g_matrix_ram ));
    g_timer0_counter = 0;
    g_timer1_counter = 0;
    g_plane_location = 2;
    g_plane_bump = FALSE;
    g_bump_counter = 3000;
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* font_ is displayed to virtual vram.                                   */
void disp_putmaze( USHORT cnt_ )
{
    int i;
    UCHAR fnt;

    for ( i=0; i<D_MLED_ROW; i++ )
    {
        fnt = g_maze_data[ i*125 + (cnt_/8) ];
        g_matrix_ram[ 1 ][ i ] = fnt;
    }
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 1 dot is scrolled.                                                    */
void disp_move1dot(void )
{
    int i;
    UCHAR vtmp, vtmp2;

    for ( i=0; i<D_MLED_ROW; i++ )
    {
        vtmp = ( g_matrix_ram[ 0 ][ i ] & 0x7f) << 1;
        vtmp2 = ( g_matrix_ram[ 1 ][ i ] & 0x80) >> 7;
        g_matrix_ram[ 0 ][ i ] = vtmp + vtmp2;
        vtmp2 = ( g_matrix_ram[ 1 ][ i ] & 0x7f) << 1;
        g_matrix_ram[ 1 ][ i ] = vtmp2;
    }
}

/* End user code for global definition. Do not edit comment generated here */
```

プログラムリスト

main.c(リスト4)

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

    initialize_value();

    /* TB-board LED off */
    P7.6 = 1;
    P7.7 = 1;

    /* start timer */
    TAU_Channel0_Start();
    TAU_Channel1_Start();

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

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

Ad_user.c

リスト省略

```
/* Start user code for include definition. Do not edit comment generated here */
extern USHORT g_maze_speed;
/* End user code for include definition. Do not edit comment generated here */

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

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

    AD_Stop();
    AD_Read( &adval );
    g_maze_speed = adval/2 + 1;
    if ( g_maze_speed > 500 )
    {
        g_maze_speed = adval;
    }
    /* End user code. Do not edit comment generated here */
}
```

プログラムリスト

Int_user.c

リスト省略

```
#pragma interrupt INTP1 MD_INTP1
#pragma interrupt INTP2 MD_INTP2
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "Int.h"
/* Start user code for include definition. Do not edit comment generated here */
/* End user code for include definition. Do not edit comment generated here */

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

extern UCHAR g_intp1_sw;
extern UCHAR g_intp2_sw;

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

/*
**-----------------------------------------------------------------------------
**  Abstract:
**                      This function is INTP1 interrupt service routine.
**  Parameters:
**                      None
**  Returns:
**                      None
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTP1( void )
{
    /* Start user code. Do not edit comment generated here */

    g_intp1_sw = P4.6 ^ 1;

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

/*
**-----------------------------------------------------------------------------
**  Abstract:
**                      This function is INTP2 interrupt service routine.
**  Parameters:
**                      None
**  Returns:
**                      None
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTP2( void )
{
    /* Start user code. Do not edit comment generated here */

    g_intp2_sw = P4.7 ^ 1;

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

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

プログラムリスト

TAU_user.c(リスト1)

リスト省略

```
#pragma interrupt INTTM00 MD_INTTM00
#pragma interrupt INTTM01 MD_INTTM01
/*
***********************************************************************************
** Include files
***********************************************************************************
*/
#include "macrodriver.h"
#include "user_define.h"
#include "TAU.h"
/* Start user code for include definition. Do not edit comment generated here */
#include "AD.h"

extern void initialize_value( void );
extern void disp_putmaze( USHORT cnt_ );
extern void disp_move1dot( void );

extern UINT   g_timer0_counter;
extern UINT   g_timer1_counter;
extern UINT   g_bump_counter;
extern USHORT g_maze_scrollcnt;
extern USHORT g_maze_speed;
extern UCHAR  g_matrix_ram[ D_MLED_CNT ][ D_MLED_ROW ];
extern UCHAR  g_matrix_realram[ D_MLED_ROW ];
extern UCHAR  g_intp1_sw;
extern UCHAR  g_intp2_sw;
extern UCHAR  g_plane_location;
extern BOOL   g_plane_bump;
/* End user code for include definition. Do not edit comment generated here */

/*
***********************************************************************************
**  Global define
***********************************************************************************
*/
/* Start user code for global definition. Do not edit comment generated here */
/* End user code for global definition. 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 */
    UCHAR row, line, pdat;

    /* Matrix LED line disp */
    row = (UCHAR)(g_timer0_counter) & 7;

    /* display column */
    P5 = 0;
```

プログラムリスト

TAU_user.c(リスト2)

```
    if ( g_timer0_counter & 0x40 )
    {
        if ( row == g_plane_location )
        {
            pdat = 0x80;
        }
        else if ( row == ( g_plane_location + 1 ) )
        {
            pdat = 0x40;
        }
        else if ( row == ( g_plane_location + 2 ) )
        {
            pdat = 0x80;
        }
        else
        {
            pdat = 0;
        }

        if ( g_matrix_realram[ row ] &  pdat )
        {   /* bump judgment */
            g_plane_bump = TRUE;
        }
        else
        {
            g_matrix_realram[ row ] |= pdat;
        }
    }
    line = g_matrix_realram[ row ];
    P5 = line;
    /* display row */
    P8.0 = ( row & 1 );
    P8.1 = (( row & 2 ) >> 1);
    P8.2 = (( row & 4 ) >> 2);
    g_timer0_counter++;
    /* 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 */
    UCHAR dat;
    UINT i;

    AD_Start();

    g_timer1_counter++;
```

プログラムリスト

TAU_user.c(リスト3)

```
    if ( g_plane_bump )
    {
        for ( i=0; i<D_MLED_ROW; i++ )
        {
            dat = g_matrix_ram[ 0 ][ i ];
            if ( g_timer1_counter & 0x40 )
            {
                dat = dat ^ 0xff;
                g_bump_counter--;
            }
            g_matrix_realram[ i ] = dat;
        }
        if ( g_bump_counter <= 0 )
        {
            initialize_value();
        }
    }
    else
    {
        if ( g_timer1_counter > g_maze_speed )
        {
            if ( ( g_maze_scrollcnt % D_FONT_WIDTH ) == 0 )
            {   /* next maze */
                disp_putmaze( g_maze_scrollcnt );
            }

            /* scroll */
            disp_move1dot();

            g_maze_scrollcnt++;
            if ( g_maze_scrollcnt >=1000 )
            {
                g_maze_scrollcnt = 0;
            }
            g_timer1_counter = 0;

            /* plane postion */
            if ( g_intp2_sw )
            {
                if ( g_plane_location > 0 )
                {
                    g_plane_location--;
                }
            }
            if ( g_intp1_sw )
            {
                if ( g_plane_location < 5 )
                {
                    g_plane_location++;
                }
            }
        }

        for ( i=0; i<D_MLED_ROW; i++ )
        {
            g_matrix_realram[ i ] = g_matrix_ram[ 0 ][ i ];
        }
    }
    /* End user code. Do not edit comment generated here */
}

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

プログラム説明

e. プログラムで使用している関数について説明します。

main.c

UINT g_timer0_counter = 0;
UINT g_timer1_counter = 0;
タイマ割り込み時に+1されるカウンタです

UINT g_bump_counter = 0;
衝突した時に点滅する時間です

UCHAR g_text_sw = 0;
表示するテキストを選択します

UINT g_maze_scrollcnt = 0;
表示されている壁が何ドット移動したかを示します

USHORT g_maze_speed = 0;
壁のスクロール速度を示します

UCHAR g_matrix_ram[D_MLED_CNT][D_MLED_ROW];
UCHAR g_matrix_realram[D_MLED_ROW];
マトリックスLEDに表示するデータです。8x16ドット分ありますが実際に表示されるのは8x8ドットです

UCHAR g_plane_location;
プレイヤーの位置を示します

BOOL g_plane_bump;
プレイヤーと壁が衝突したかを示します

void initialize_value(void)
変数初期化を行う関数です

void disp_putmaze(USHORT cnt_)
パラメータ cnt_ で指定された壁をマトリックスLEDの非表示領域へ出力します

void disp_move1dot(void)
マトリックスLEDの表示を1ドット移動(スクロール)します

Ad_user.c

__interrupt void MD_INTAD(void)
A/D変換終了時に呼ばれる関数です。
A/D値を読み込み g_maze_speed へ反映させます

Int_user.c

__interrupt void MD_INTP1(void)
外部割り込みINTP1に呼ばれる関数です。
SWの値を g_intp1_sw へ反映します

__interrupt void MD_INTP2(void)
外部割り込みINTP2に呼ばれる関数です。
SWの値を g_intp2_sw へ反映します

TAU_user.c

__interrupt void MD_INTTM00()
1msec毎に呼ばれる関数です。下記の処理を行います
・プレイヤーの表示
・プレイヤーと壁の衝突判定
・マトリックスLEDにg_matrix_realram よりデータを読み込み1ライン表示

__interrupt void MD_INTTM01()
1msec毎に呼ばれる関数です。下記の処理を行います。
・A/D変換を開始します
・プレイヤーと壁が衝突していたら衝突処理します
・スクロールのチェックを行い、1画面分のスクロールが終了したら disp_putmaze を呼び出し、指定された壁をマトリックスLEDの非表示領域へ出力します。
そうでなければ disp_move1dot を呼び出し、1ドット分スクロールします。
・プレイヤーの移動処理します
・マトリックスLEDの非表示領域より実際の表示領域 g_matrix_realram へデータ転送します

user_define.h

#define D_MLED_CNT   2 /* Matrix LED number   */
マトリックスLEDの数を定義します。実際は1つですが仮想的に2つ持っていることにしています。

#define D_MLED_ROW   8 /* Matrix LED row   */
マトリックスLEDのROWの数

#define D_FONT_WIDTH 8 /* font width   */
1文字のフォントの幅。これは壁データになります

迷路のデータ作成方法

f．プログラム内で定義している迷路データについて説明します。

const UCHAR g_maze_data[] データ作成方法

main.cのconst UCHAR g_maze_data[] ですが簡単に作成できる方法があります。Windowsのペイントツールを使います。まず画像を新規作成します。

・大きさを1000x8
・色は黒
・ファイルの種類はモノクロビットマップ

次にメニューより[表示(V)]→[拡大(Z)]→[拡大率の指定(U)]で800%を指定します。また、[表示(V)]→[拡大(Z)]→[グリッドを表示(G)]を指定します。画面は下図のようになります

白を描画すると、それが壁のデータになります。後は保存したbmpファイル(バイナリファイル)をソースへ変換して下さい。bmpファイルのフォーマット説明します。

最初の59バイトはbmpヘッダ
3バイト(ヘッダ)+125バイト (1000ドット分)が 1ライン分のデータです。

x 8

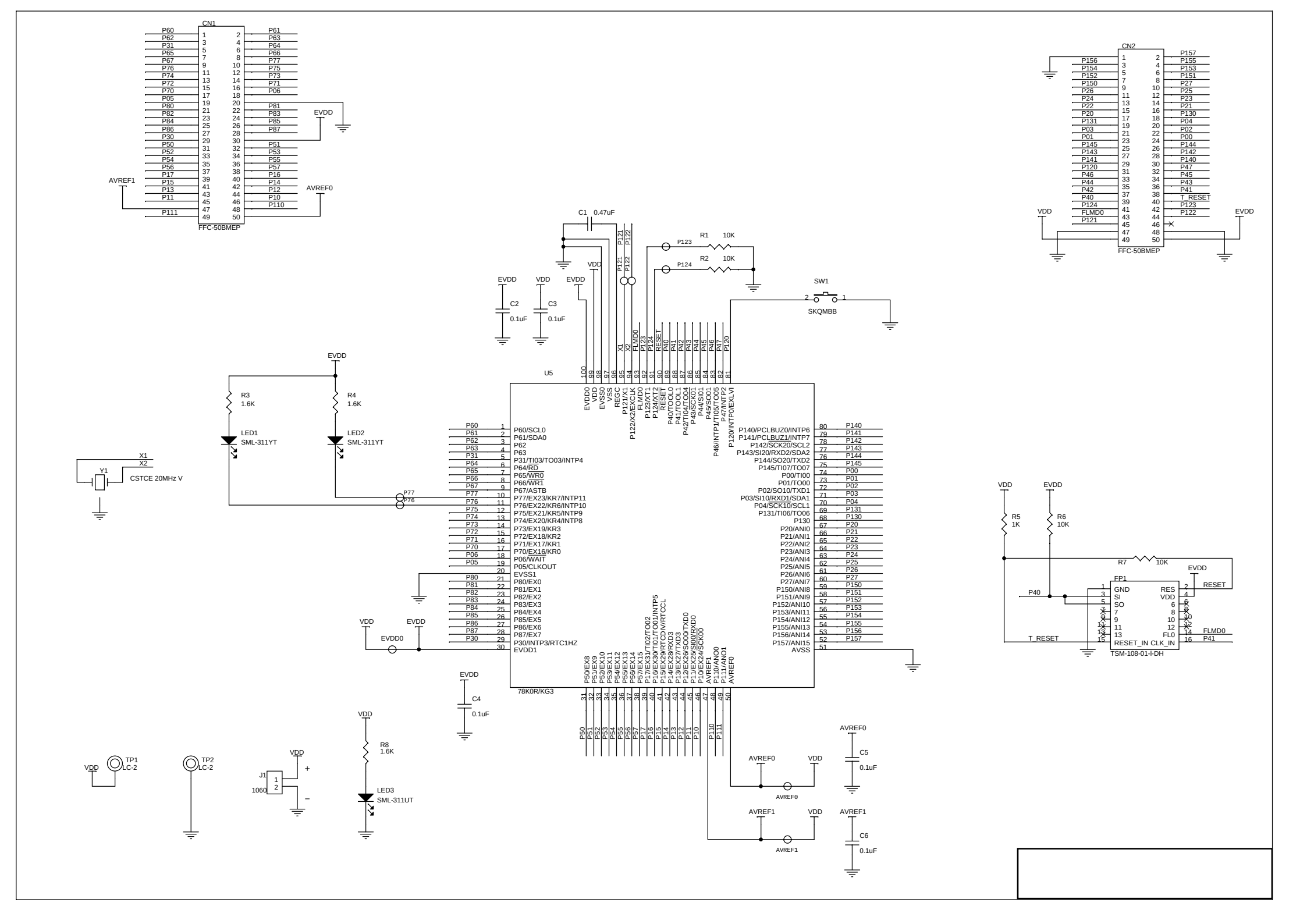
1ライン分データの125バイト(1000ドット)を変換すれば実データになります。ラインデータは逆になっています画面最下ラインから上に向かってのデータになっていますので、ソースへ変換後順序を入れ替えて下さい。

最後に

　ターゲット･システム作成例は以上です。最後の作成例はゲーム性を持たせホビーの要素を取り入れました。このゲームには音がないので、タイマをPWM出力させたポートへ圧電スピーカを接続しても面白いと思います。圧電スピーカは小型で非常に薄く、高能率、低消費電力ですのでアンプを通さず、そのままポートに接続しても充分音がでます。

　昨今の電機製品の殆どにマイコンが使われています。特に子供向けのゲームなど1000円前後で買えるものは1チップマイコンにスピーカー、LEDをつけただけです。それが音声出力、各種制御まで行っています。これだけでもマイコンの可能性がわかるのではないでしょうか。このチュートリアル･ガイドがマイコンを始めるきっかけになれば幸いです。

　この作成例を元に電光掲示板の大きさを大きくして表示可能なドット数を増やしたり、赤外線送受信を加えて対戦方式のゲームを作成など、これを応用して是非チャレンジして下さい。


CN1
FFC-50BMEP
CN2
FFC-50BMEP
U5
78K0R/KG3
C1 0.47uF
C2 0.1uF
C3 0.1uF
C4 0.1uF
C5 0.1uF
C6 0.1uF
R1 10K
R2 10K
R3 1.6K
R4 1.6K
R5 1K
R6 10K
R7 10K
R8 1.6K
SW1
SKQMBB
LED1 SML-311YT
LED2 SML-311YT
LED3 SML-311UT
X1
X2
Y1
CSTCE 20MHz V
FP1
TSM-108-01-I-DH
TP1 LC-2
TP2 LC-2
J1
EVDD
VDD
AVREF0
AVREF1
RESET
T_RESET
FLMD0