お客様各位

ZUD-CD-09-0036 1/116
2009年5月29日
NECエレクトロニクス株式会社
マイクロコンピュータ事業本部
汎用マイコンシステム事業部
開発ツールソリューショングループ
チームマネージャー 安藤 喜成
(担当：鈴木 康之)

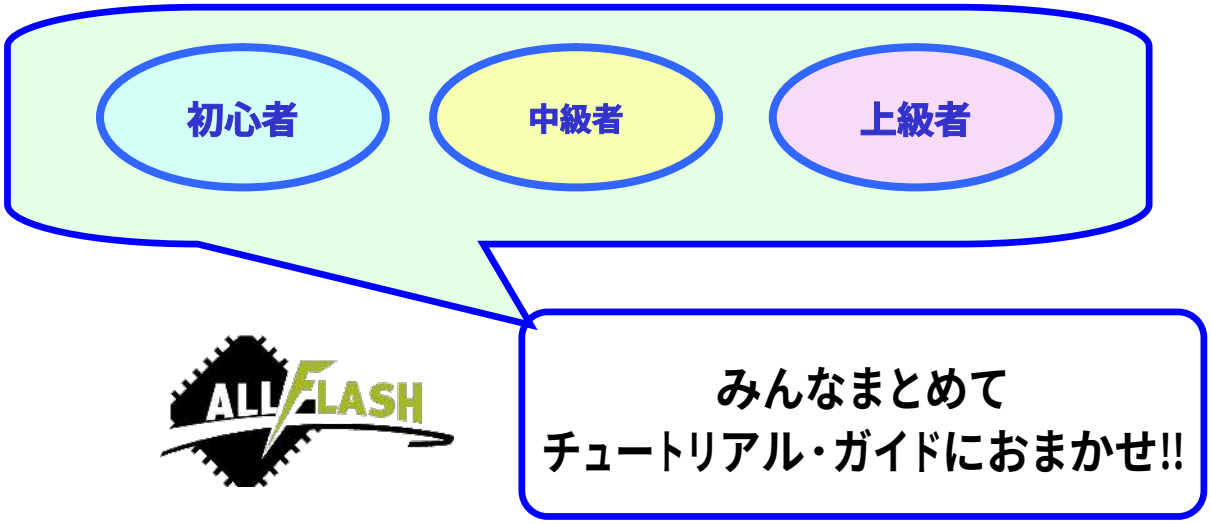
V850ES/JG3-L、V850ES/JJ3
ターゲット・ボード
QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
チュートリアル・ガイド
基礎編

ごあいさつ

QB-V850ESJG3L-TBまたはQB-V850ESJJ3-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-V850ESJG3L-TBまたはQB-V850ESJJ3-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。

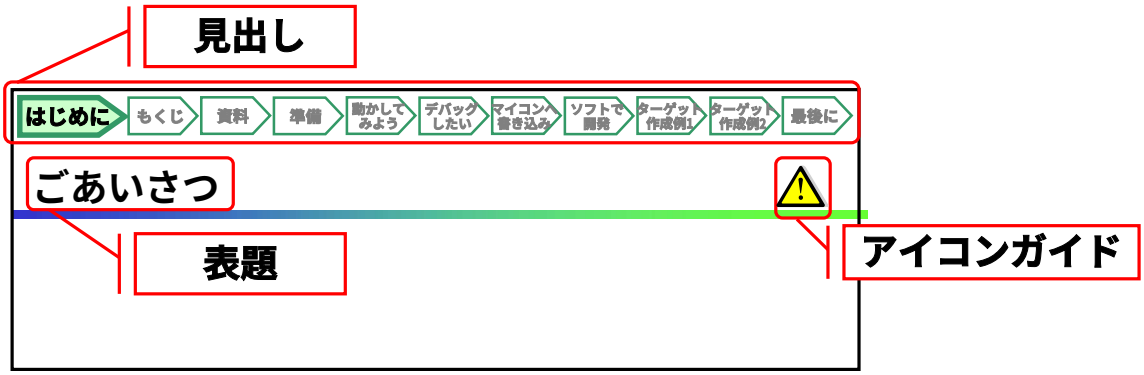
[MINICUBE2関連ツール]
http://www.necel.com/micro/ja/development/asia/minicube2/minicube2.html

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

[サンプル･プログラムとCPUボードの資料]
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。

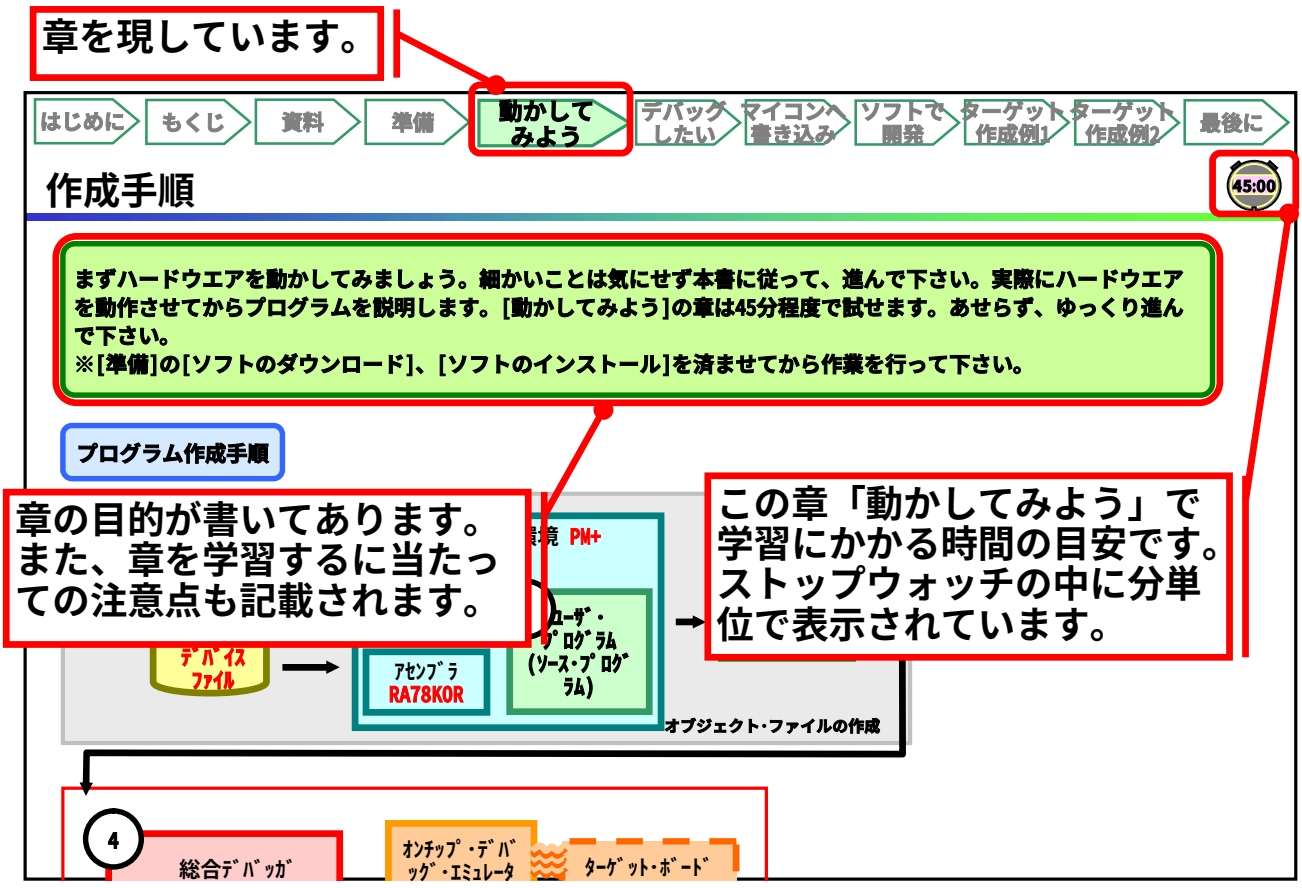
必ず読む必要のある内容です。


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-V850ESJG3L-TBの特徴

V850ES/JG3Lターゲット・ボード(QB-V850ESJG3L-TB)の特徴

●V850ES/JG3L(μPD70F3738GC)搭載
●メイン・クロック5MHz(発振子を搭載)で最大20MHzで動作可能(2.7V～3.6V供給時)
●フラッシュメモリ:256KB、RAM:16KBを内蔵
●最大で84本のI/Oポートを装備(5Vトレラント/N-chオープン・ドレーン31本)
●プログラミング、オンチップ・デバッグに両対応(SIB0,SOB0,SCKB0,PCM0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

V850ES/JG3Lターゲット・ボード(QB-V850ESJG3L-TB)のハードウエア仕様

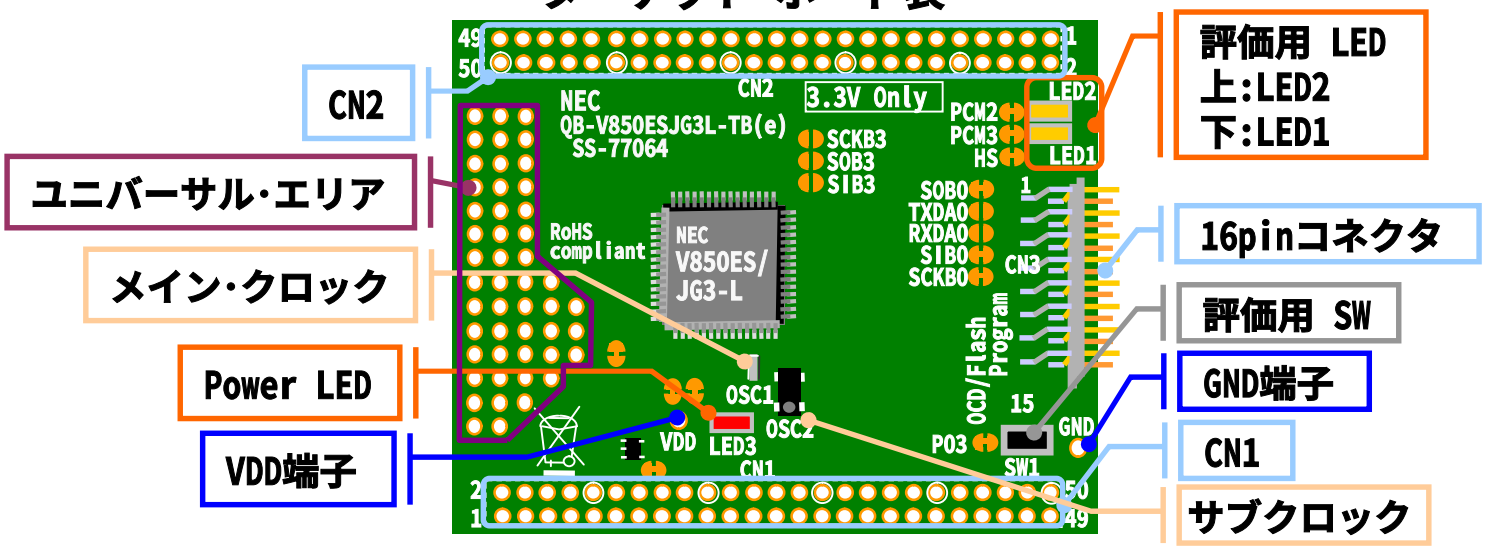
CPU μPD70F3738GC	メイン・クロック 動作周波数	最大20MHz (ボード上に発振子5MHz搭載)
	サブクロック 動作周波数	32.768KHz (ボード上に搭載)
搭載部品	CN1,CN2:周辺ボードコネクタ(2.54mmピッチ) 50pinソケットx2(パットのみ)	
	FP1:16pinコネクタ(MINICUBE2接続用)	
	PowerLED:LED赤x1(LED3)	
	評価用LED:LED黄x2(LED1はPCM3,LED2はPCM2へ接続)	
	評価用SW:SW1(INTP0へ接続)	
	OSC1:5MHz発振子(X1,X2へ接続)	
	OSC2:32.768KHz発振子(XT1,XT2へ接続)	
動作電圧	2.7V～3.6V(Y1:5MHz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。

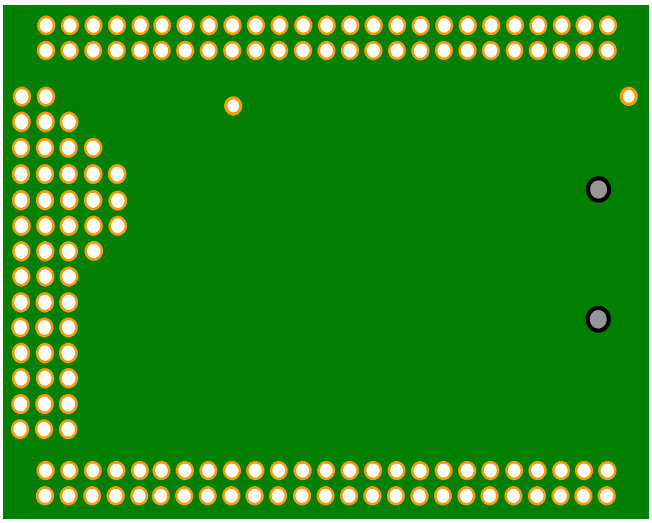
QB-V850ESJG3L-TB部品配置図

ターゲット･ボード表



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PCM3がLOWで黄色に発光します
評価用LED2: ポートPCM2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
OSC1(ﾒｲﾝｸﾛｯｸ): 5MHz発振子を搭載しています
OSC2(ｻﾌﾞｸﾛｯｸ): 32.768KHz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



・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。 
PCM3,PCM2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。

QB-V850ESJG3L-TB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	AVREF0		2	GND	
3	P10/ANO0		4	P11/ANO1	
5	AVREF1		6	PDH4/A20	
7	PDH5/A21		8	FLMD0	16pinｺﾈｸﾀ14へ接続
9	VDD		10	NC	
11	GND		12	NC	
13	NC		14	T_RESET	16pinｺﾈｸﾀ15へ接続
15	NC		16	NC	
17	P02/NMI		18	P03/INTP0/ADTRG	SW1にも接続 10KΩﾌﾟﾙｱｯﾌﾟ
19	P04/INTP1		20	P05/INTP2/$\overline{\text{DRST}}$	10KΩﾌﾟﾙﾀﾞｳﾝ
21	P06/INTP3		22	P40/SIB0/SDA01	16pinｺﾈｸﾀ5へ接続
23	P41/SOB0/SCL01	16pinｺﾈｸﾀ3へ接続	24	P42/$\overline{\text{SCKB0}}$	16pinｺﾈｸﾀ7へ接続
25	P30/TXDA0/SOB4		26	P31/RXDA0/INTP7/SIB4	
27	P32/ASCKA0/$\overline{\text{SCKB4}}$/TIP00/TOP00		28	P33/TIP01/TOP01	
29	P34/TIP10/TOP10		30	P35/TIP11/TOP11	
31	P36		32	P37	
33	GND		34	EVDD	
35	P38/TXDA2/SDA00		36	P39/RXDA2/SCL00	
37	P50/TIQ01/KR0/TOQ01/RTP00		38	P51/TIQ02/KR1/TOQ02/RTP01	
39	P52/TIQ03/KR2/TOQ03/RTP02/DDI		40	P53/SIB2/KR3/TIQ00/TOQ00/RTP03/DDO	
41	P54/SOB2/KR4/RTP04/DCK		42	P55/$\overline{\text{SCKB2}}$/KR5/RTP05/DMS	
43	P90/A0/KR6/TXDA1/SDA02		44	P91/A1/KR7/RXDA1/SCL02	
45	P92/A2/TIP41/TOP41		46	P93/A3/TIP40/TOP40	
47	P94/A4/TIP31/TOP31		48	P95/A5/TIP30/TOP30	
49	P96/A6/TIP21/TOP21		50	P97/A7/SIB1/TIP20/TOP20	

(※詳細は付録の回路図を参照して下さい)

QB-V850ESJG3L-TB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	P98/A8/SOB1		2	P99/A9/$\overline{\text{SCKB1}}$	
3	P910/A10/SIB3		4	P911/A11/SOB3	
5	P912/A12/$\overline{\text{SCKB3}}$		6	P913/A13/INTP4	
7	P914/A14/INTP5/TIP51/TOP51		8	P915/A15/INTP6/TIP50/TOP50	
9	PDH2/A18		10	PDH3/A19	
11	PCM0/$\overline{\text{WAIT}}$		12	PCM1/CLKOUT	
13	PCM2/$\overline{\text{HLDAK}}$	LED2にも接続	14	PCM3/$\overline{\text{HLDRQ}}$	LED1にも接続
15	PCT0/$\overline{\text{WR0}}$		16	PCT1/$\overline{\text{WR1}}$	
17	PCT4/$\overline{\text{RD}}$		18	PCT6/ASTB	
19	GND		20	EVDD	
21	PDL0/AD0		22	PDL1/AD1	
23	PDL2/AD2		24	PDL3/AD3	
25	PDL4/AD4		26	PDL5/AD5/FLMD1	16pinｺﾈｸﾀ12へ接続
27	PDL6/AD6		28	PDL7/AD7	
29	PDL8/AD8		30	PDL9/AD9	
31	PDL10/AD10		32	PDL11/AD11	
33	PDL12/AD12		34	PDL13/AD13	
35	PDL14/AD14		36	PDL15/AD15	
37	PDH0/A16		38	PDH1/A17	
39	P711/ANI11		40	P710/ANI10	
41	P79/ANI9		42	P78/ANI8	
43	P77/ANI7		44	P76/ANI6	
45	P75/ANI5		46	P74/ANI4	
47	P73/ANI3		48	P72/ANI2	
49	P71/ANI1		50	P70/ANI0	

(※詳細は付録の回路図を参照して下さい)

QB-V850ESJJ3-TBの特徴

V850ES/JJ3ターゲット・ボード(QB-V850ESJJ3-TB)の特徴

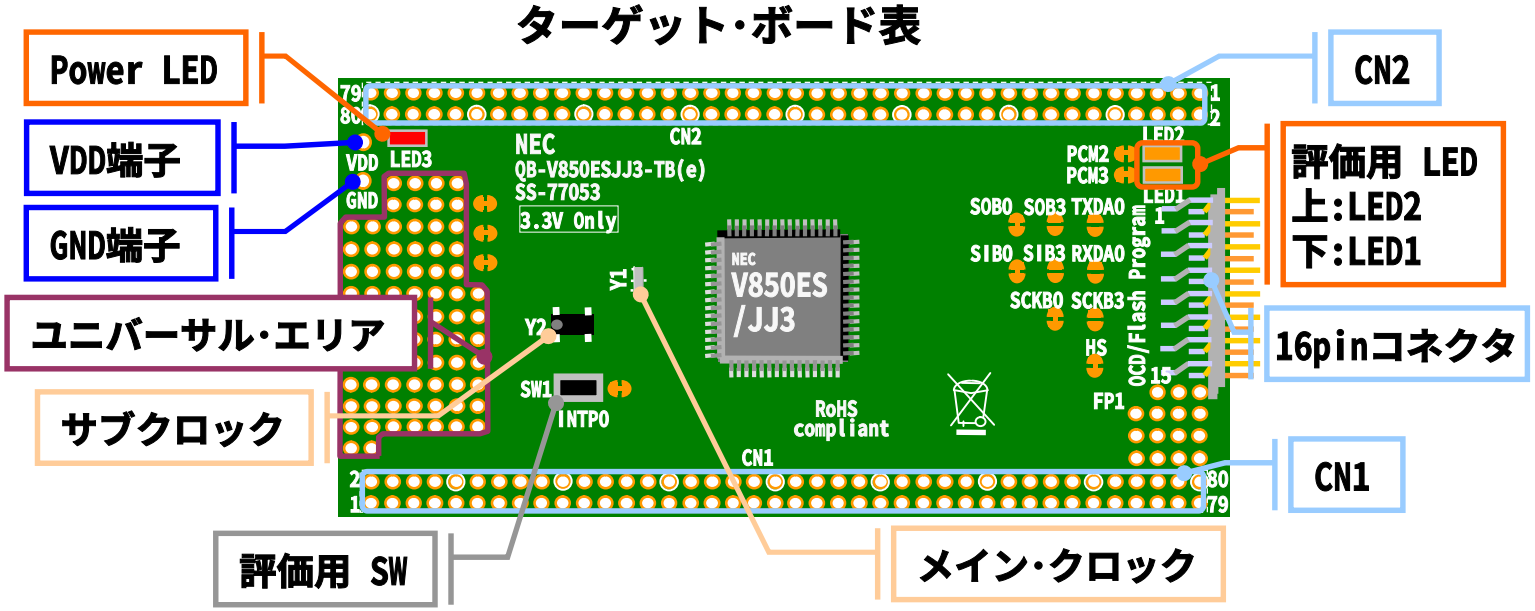
●V850ES/JJG3(μPD70F3746GJ)搭載
●メイン・クロック4MHz(発振子を搭載)で最大32MHzで動作可能(3.0V～3.6V供給時)
●フラッシュメモリ:1024KB、RAM:60KBを内蔵
●最大で128本のI/Oポートを装備(5Vトレラント/N-chオープン・ドレーン60本)
●プログラミング、オンチップ・デバッグに両対応(SIB0,SOB0,SCKB0,PCM0端子使用)
●プログラミング、オンチップ・デバッグ端子はCSI0/CSI3/UART選択可能
●LED2個、SW1個を搭載しており簡単なテストが可能
●ユニバーサル・エリア(2.54mmピッチ)を搭載
●マイ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

V850ES/JJ3ターゲット・ボード(QB-V850ESJJ3-TB)のハードウエア仕様

CPU μPD70F3746GJ	メイン・クロック 動作周波数	最大32MHz (ボード上に発振子4MHz搭載)
	サブクロック 動作周波数	32.768KHz (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) 80pin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PCM3,LED2はPCM2へ接続)	
	評価用SW:SW1(INTP0へ接続)	
	OSC1:4MHz発振子(X1,X2へ接続)	
	OSC2:32.768KHz発振子(XT1,XT2へ接続)	
動作電圧	3.0V～3.6V(Y1:4MHz発振子使用時)	

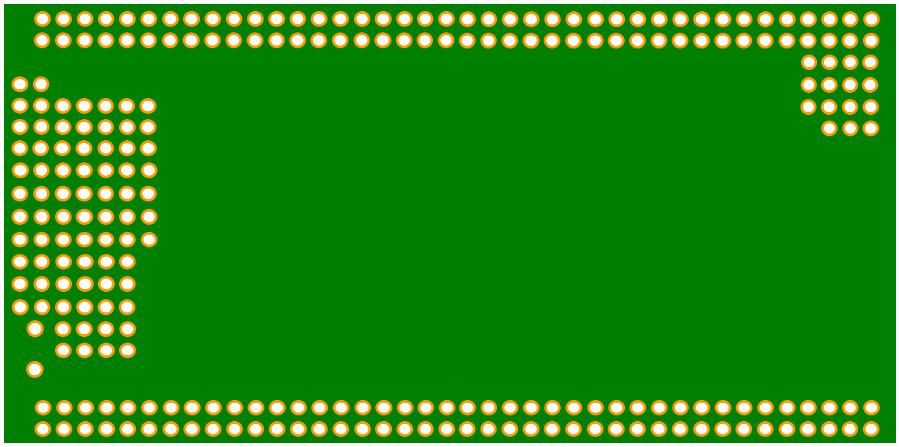
QB-V850ESJJ3-TB部品配置図

ターゲット･ボード表



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PCM3がLOWで黄色に発光します
評価用LED2: ポートPCM2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): 4MHz発振子を搭載しています
Y2(ｻﾌﾞｸﾛｯｸ): 32.768KHz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



・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。

QB-V850ESJJ3-TBコネクタ情報(CN1)

番ピ号ン	接続先　CPU端子	備考	ピン番号	接続先　CPU端子	備考
1	AVREF0		2	GND	
3	P10/ANO0		4	P11/ANO1	
5	AVREF1		6	P00/TIP61/TOP61	
7	P01/TIP60/TOP60		8	FLMD0	16pinｺﾈｸﾀ14へ接続
9	VDD		10	NC	
11	GND		12	NC	
13	NC		14	T_RESET	
15	NC		16	NC	
17	P02/NMI		18	P03/INTP0/ADTRG	SW1にも接続 10KΩﾌﾟﾙｱｯﾌﾟ
19	P04/INTP1		20	P05/INTP2/$\overline{\text{DRST}}$	10KΩﾌﾟﾙﾀﾞｳﾝ
21	P06/INTP3		22	P40/SIB0/SDA01	16pinｺﾈｸﾀ5へ接続
23	P41/SOB0/SCL01	16pinｺﾈｸﾀ3へ接続	24	P42/$\overline{\text{SCKB0}}$	16pinｺﾈｸﾀ7へ接続
25	P30/TXDA0/SOB4		26	P31/RXDA0/INTP7/SIB4	
27	P32/ASCK0/$\overline{\text{SCKB4}}$/TIP00/ TOP00		28	P33/TIP01/TOP01	
29	P34/TIP10/TOP10		30	P35/TIP11/TOP11	
31	P36		32	P37	
33	GND		34	EVDD	
35	P38/TXDA2/SDA00		36	P39/RXDA2/SCL00	
37	P50/TIQ01/KR0/TOQ01/ RTP00		38	P51/TIQ02/KR1/TOQ02/RTP01	
39	P52/TIQ03/KR2/TOQ03/ RTP02/DDI		40	P53/SIB2/KR3/TIQ00/TOQ00/ RTP03/DDO	
41	P54/SOB2/KR4/RTP04/DCK		42	P55/$\overline{\text{SCKB2}}$/KR5/RTP05/DMS	
43	P60/RTP10		44	P61/RTP11	
45	P62/RTP12		46	P63/RTP13	
47	P64/RTP14		48	P65/RTP15	
49	P66/SIB5		50	P67/SOB5	
51	P68/SCKB5		52	P69/TIP70/TOP70	
53	P610/TIP71		54	P611/TOP71	
55	P612/TIP80/TOP80		56	P613/TIP81/TOP81	
57	P614		58	P615	
59	P80/RXDA3/INTP8		60	P81/TXDA3	
61	P90/A0/KR6/TXDA1/SDA02		62	P91/A1/KR7/RXDA1/SCL02	
63	P92/A2/TIP41/TOP41		64	P93/A3/TIP40/TOP40	
65	P94/A4/TIP31/TOP31		66	P95/A5/TIP40/TOP40	
67	P96/A6/TIP21/TOP21		68	P97/A7/SIB1/TIP20/TOP20	
69	P98/A8/SOB1		70	P99/A9/$\overline{\text{SCKB1}}$	
71	P910/A10/SIB3		72	P911/A11/SOB3	
73～ 79	NC		74～ 80	NC	

(※詳細は付録の回路図を参照して下さい)

QB-V850ESJJ3-TB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	P912/A12/SCKB3		2	P913/A13/INTP4	
3	P914/A14/INTP5/TIP51/TOP51		4	P915/A15/INTP6/TIP50/TOP50	
5	PCD0		6	PCD1	
7	PCD2		8	PCD3	
9	PCS0/CS0		10	PCS1/CS1	
11	PCS2/CS2		12	PCS3/CS3	
13	PCM0/$\overline{\text{WAIT}}$	16pinｺﾈｸﾀ8へ接続	14	PCM1/CLKOUT	
15	PCM2/$\overline{\text{HLDAK}}$	LED2へ接続	16	PCM3/$\overline{\text{HLDRQ}}$	LED1へ接続
17	PCM4		18	PCM5	
19	PCS4		20	PCS5	
21	PCS6		22	PCS7	
23	PCT0/$\overline{\text{WR0}}$		24	PCT1/$\overline{\text{WR1}}$	
25	PCT2		26	PCT3	
27	PCT4/$\overline{\text{RD}}$		28	PCT5	
29	PCT6/ASTB		30	PCT7	
31	GND		32	EVDD	
33	PDL0/AD0		34	PDL1/AD1	
35	PDL2/AD2		36	PDL3/AD3	
37	PDL4/AD4		38	PDL5/AD5/FLMD1	16pinｺﾈｸﾀ12へ接続
39	PDL6/AD6		40	PDL7/AD7	
41	PDL8/AD8		42	PDL9/AD9	
43	PDL10/AD10		44	PDL11/AD11	
45	PDL12/AD12		46	PDL13/AD13	
47	PDL14/AD14		48	PDL15/AD15	
49	PDH0/AD16		50	PDH1/AD17	
51	PDH2/AD18		52	PDH3/AD19	
53	PDH4/AD20		54	PDH5/AD21	
55	PDH6/AD22		56	PDH7/AD23	
57	P715/ANI15		58	P714/ANI14	
59	P713/ANI13		60	P712/ANI12	
61	P711/ANI11		62	P710/ANI10	
63	P79/ANI9		64	P78/ANI8	
65	P77/ANI7		66	P76/ANI6	
67	P75/ANI5		68	P74/ANI4	
69	P73/ANI3		70	P72/ANI2	
71	P71/ANI1		72	P70/ANI0	
73～79	NC		74～80	NC	

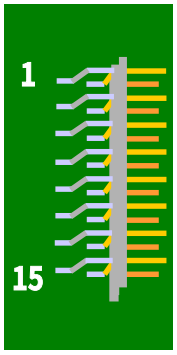
(※詳細は付録の回路図を参照して下さい)

コネクタ情報(16pinヘッダ)

※上記の情報はQB-V850ESJG3L-TB、QB-V850ESJJ3-TB共に共通です。

16pinヘッダピンアサイン(デバッグ時に扱う信号)

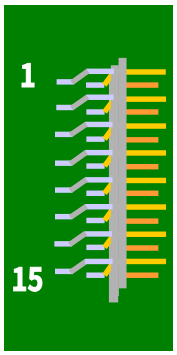
ピン番号	信号	ピン番号	信号
1	GND	2	RESET_OUT
3	SIBO	4	VDD
5	SOBO	6	R.F.U.
7	$\overline{\text{SCKBO}}$	8	H/S(PCMO)
9	R.F.U.	10	R.F.U.
11	------	12	R.F.U.
13	R.F.U.	14	FLMDO
15	RESET_IN	16	R.F.U.



※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

16pinヘッダピンアサイン(プログラミング時に扱う信号)

ピン番号	信号	ピン番号	信号
1	GND	2	RESET_OUT
3	SIBO	4	VDD
5	SOBO	6	R.F.U.
7	$\overline{\text{SCKBO}}$	8	H/S(PCMO)
9	R.F.U.	10	R.F.U.
11	------	12	R.F.U.
13	R.F.U.	14	FLMDO
15	R.F.U.	16	R.F.U.



※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

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]については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。
[フリーツール]のインストールについて、これから説明します。

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/

ダウンロードするソフトウエア一覧

- **CA850**
  統合開発環境PM+とV850ES用コンパイラ含めたパッケージです。
- **Applilet2 for V850ESJX3**
  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。
- **V850ES/Jx3，V850ES/Jx3-L用デバイス･ファイル**
  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。

※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128KBの制限があります。

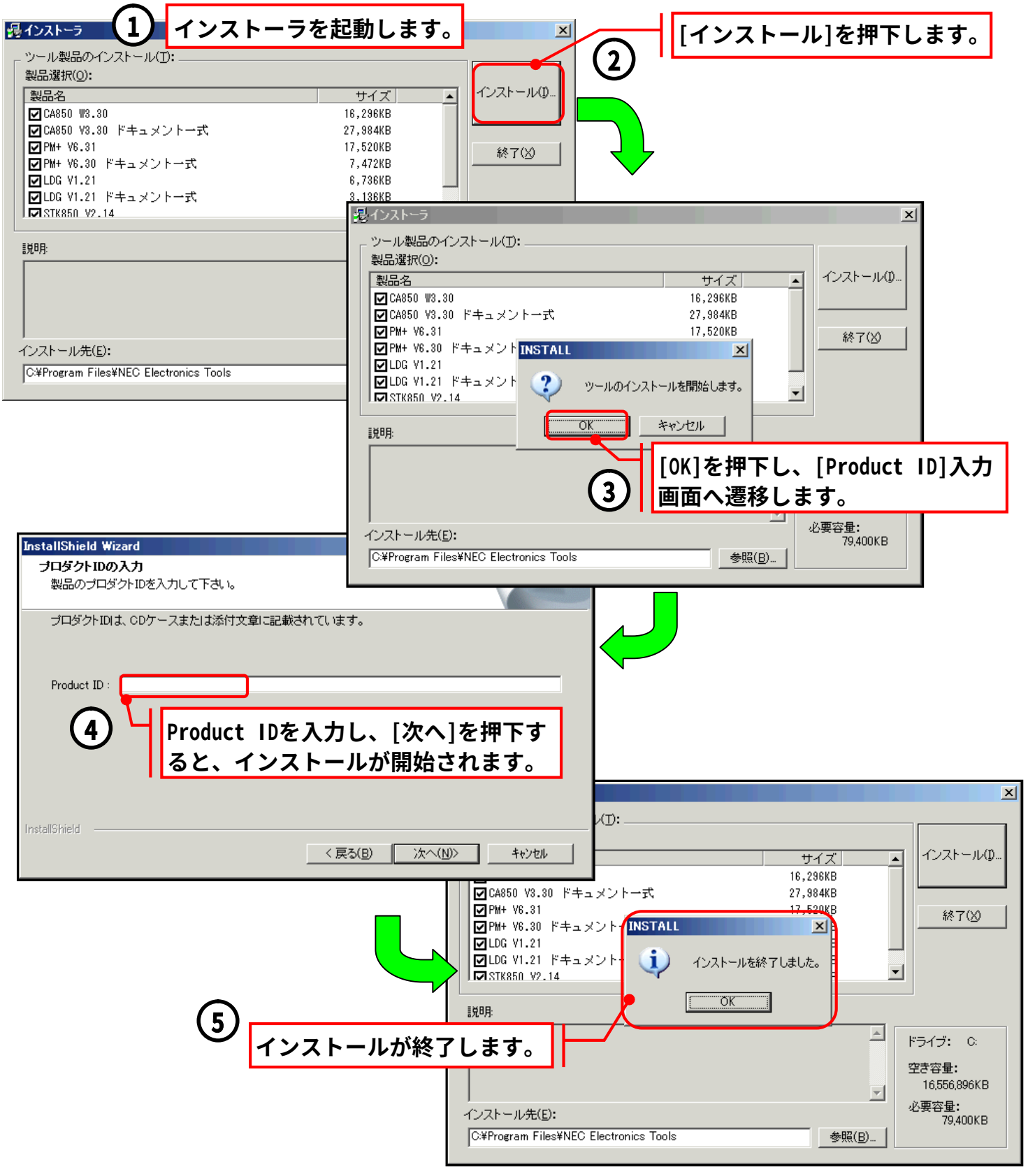
フリーツールをダウンロードする時にユーザー登録します。登録したメールアドレスにProduct IDが送られます。このProduct IDは開発ツールソフトウエアのインストールに必要ですので、忘れないようにメモして下さい。

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

CA850のインストール

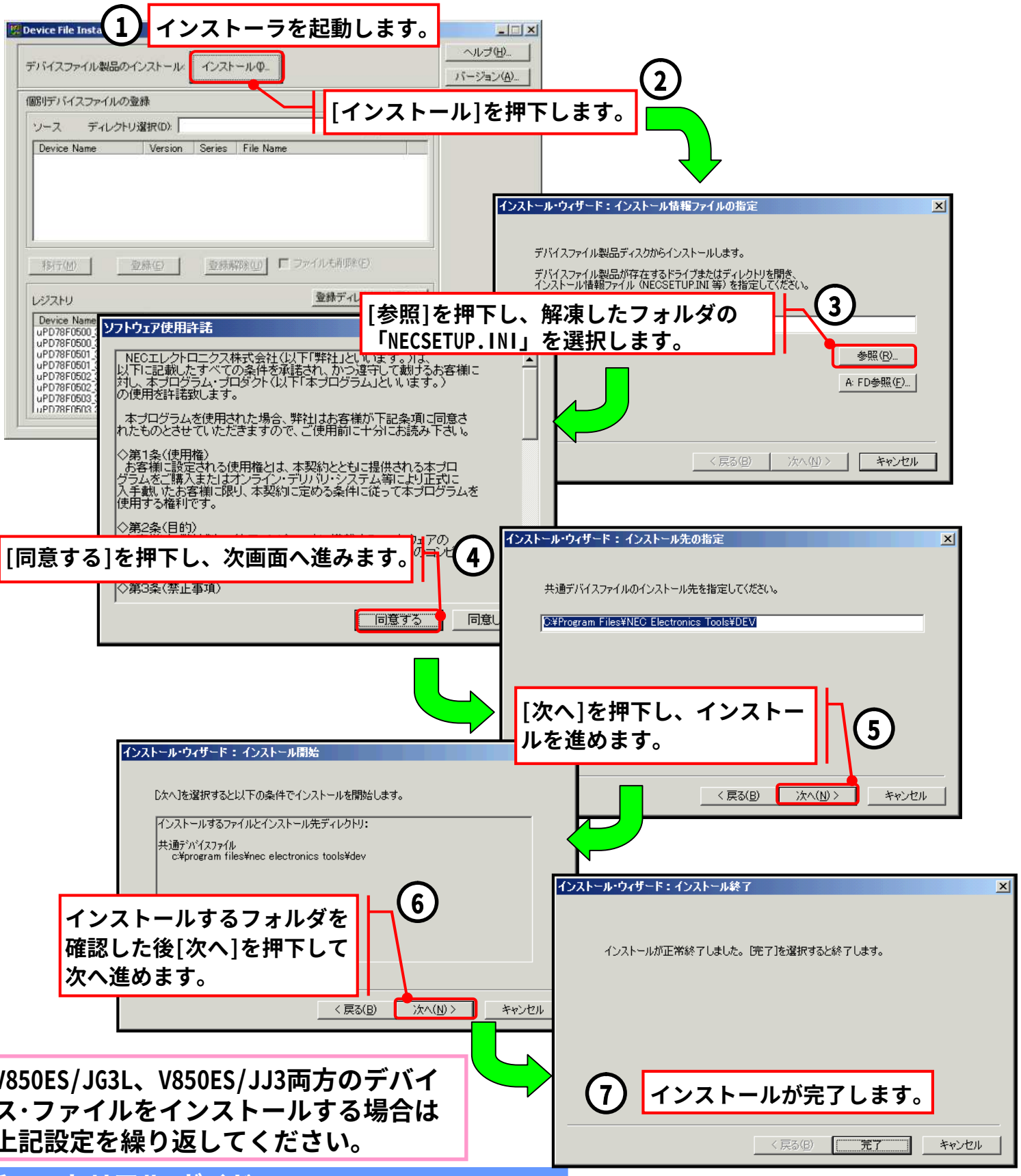
ダウンロードしたファイル[ca703000_w330_j.exe]をダブル･クリックし、インストーラを起動します。
[インストール]を押下し、ツールのインストールを開始します。



ソフトウエアのインストール

V850ES/JG3LとV850ES/JJ3用デバイス･ファイルのインストール

[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デバイスファイルを解凍したフォルダで、「NECSETUP.INI」を選択します。



V850ES/JG3L、V850ES/JJ3両方のデバイス･ファイルをインストールする場合は上記設定を繰り返してください。

ソフトウエアのインストール

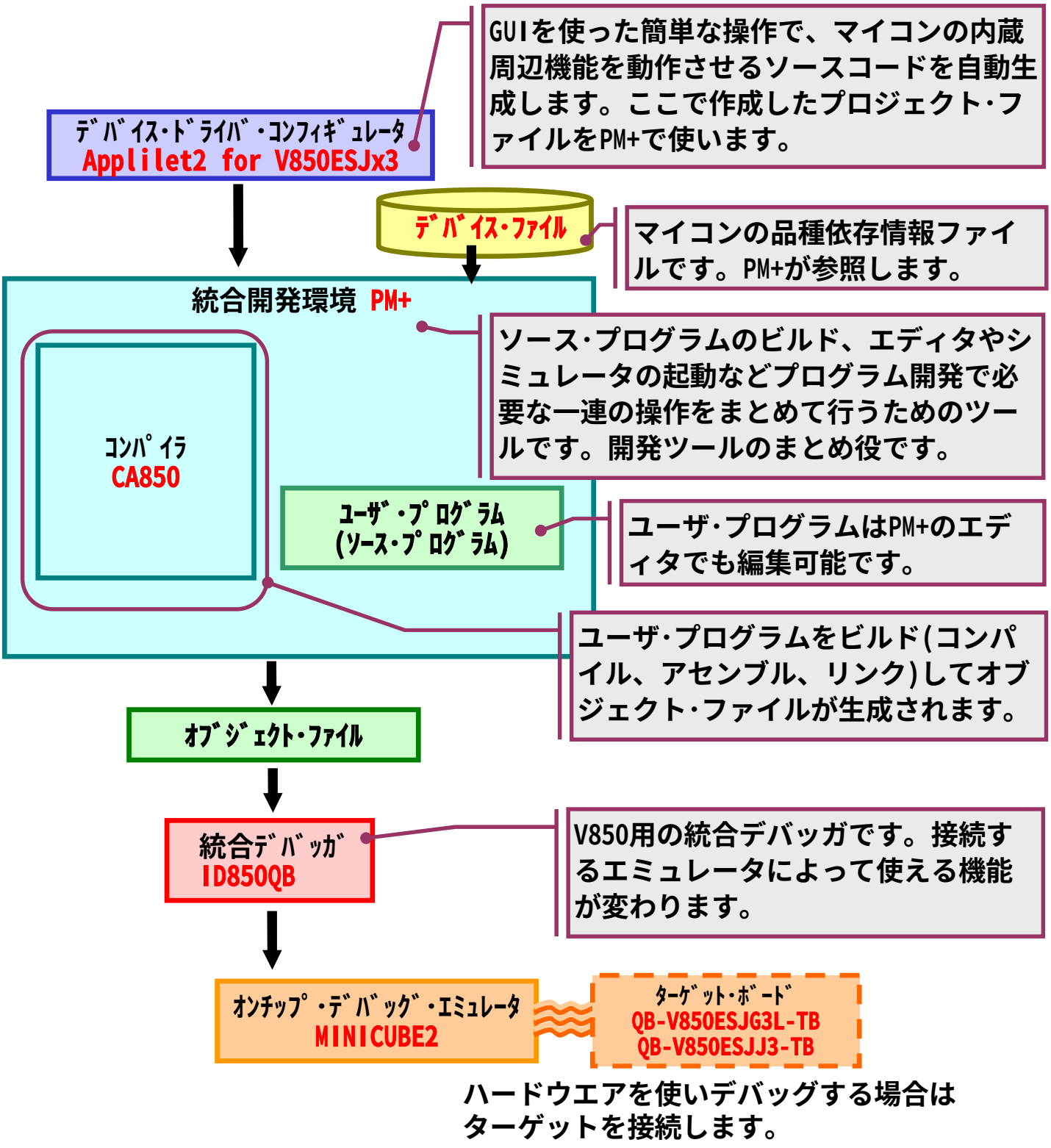
Applilet2 for V850ESJX3インストール

Applilet2 for V850ESJX3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

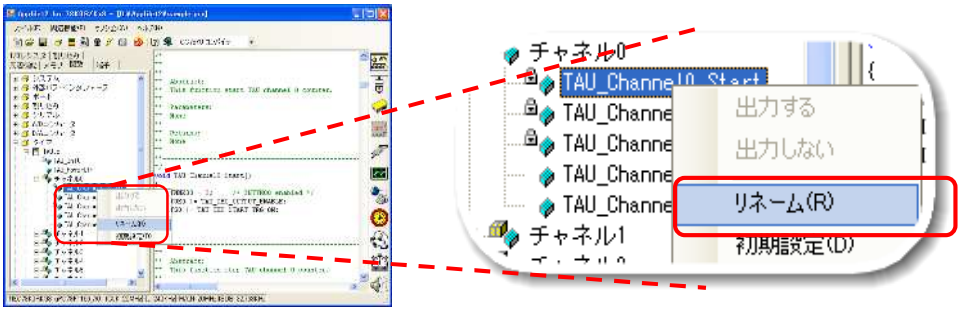
ソースコードのガード機能

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]のファイルを上書きするか、そのまま残すし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 */
}
```

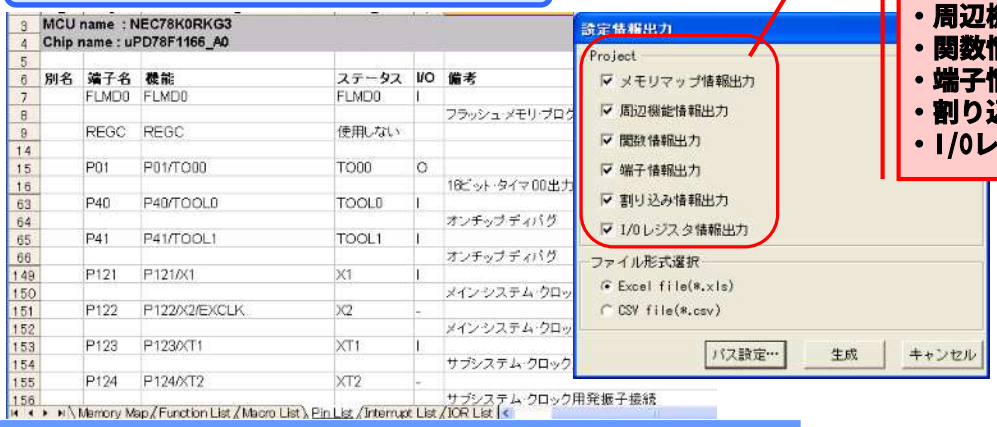
このコメントに挟まれた場所にプログラムを書けば、Applilet2のコード生成で上書きされることはありません。以前に書いたコードは、そのまま残ります。

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

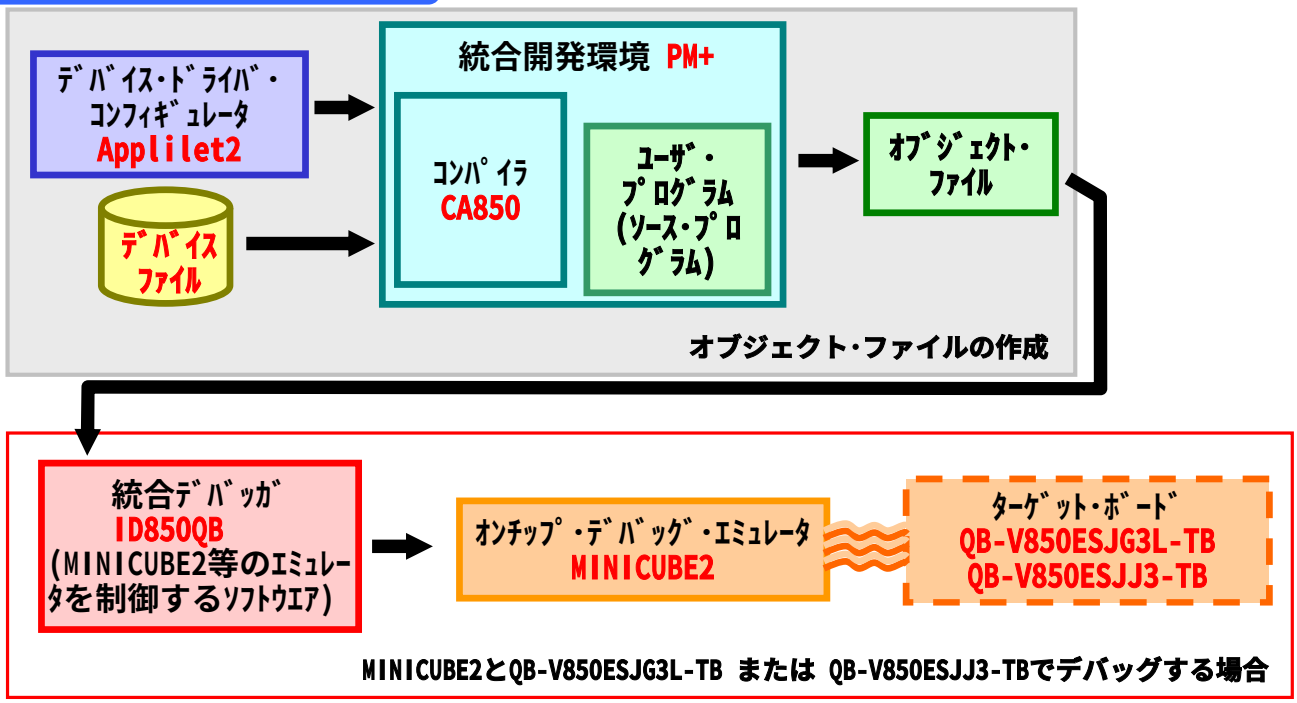


下記の情報をEXCELファイルへ出力します
- メモリマップ
- 周辺機能情報
- 関数情報
- 端子情報
- 割り込み情報
- I/Oレジスタ情報

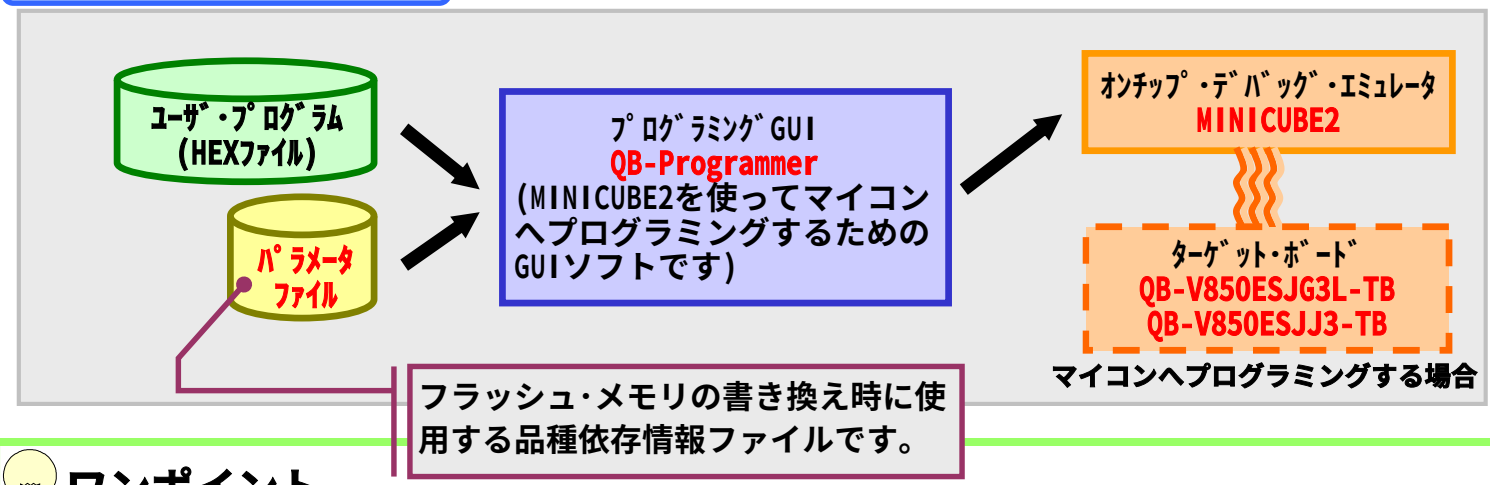
システム構成図

プログラム開発の流れとプログラミング(マイコンへHEXデータの書き込み)の流れを説明します。
[プログラム開発時]
・MINICUBE2とターゲット・システムを使う(デバッグ時にはマイコンを使います)
[プログラミング時]
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)

プログラム開発の流れ



プログラミングの流れ



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。

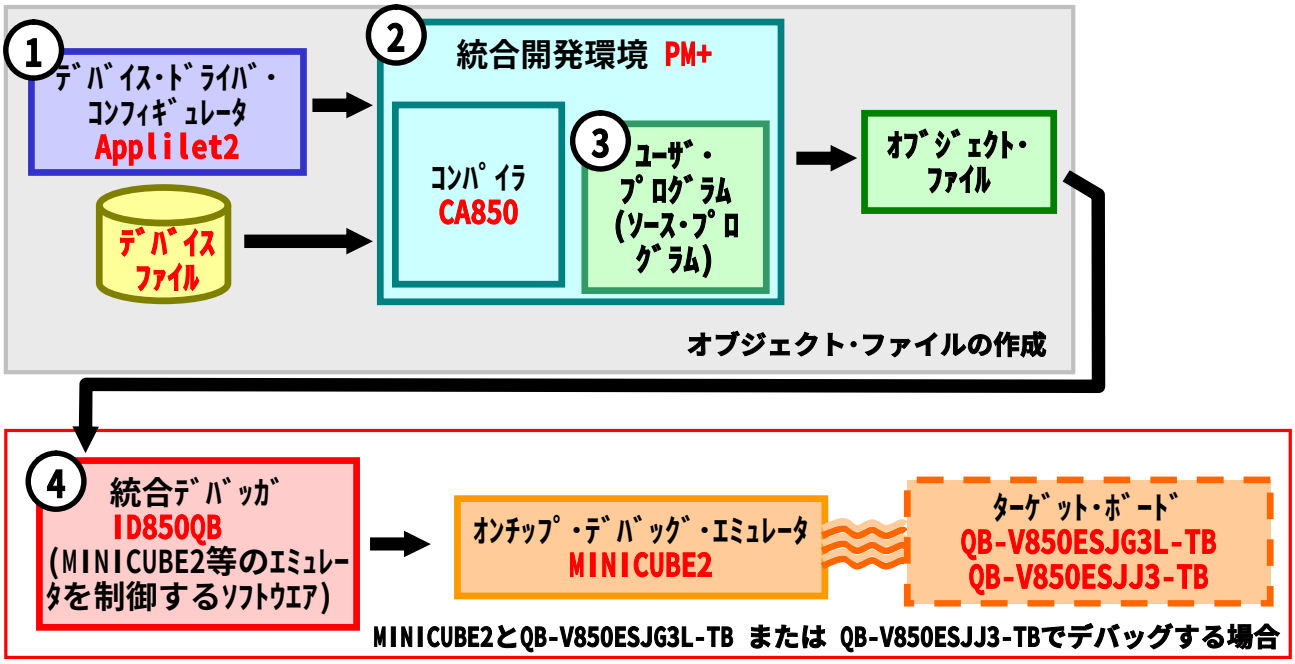
作成手順

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。

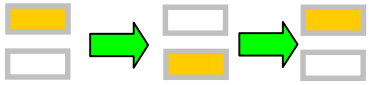
② ①で生成したプロジェクトを読み込みます。PM+はソースコードのビルド、エディタや統合デバッガの起動などプログラム開発で必要な一連の操作をまとめて行うためのツールです。

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。

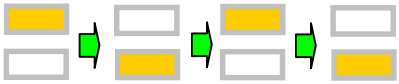
④ ③で作成したオブジェクト･ファイルを元にデバッグします。MINICUBE2とQB-V850ESJG3L-TBまたはQB-V850ESJJ3-TBを使います。実際にマイコンを動作させます。

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

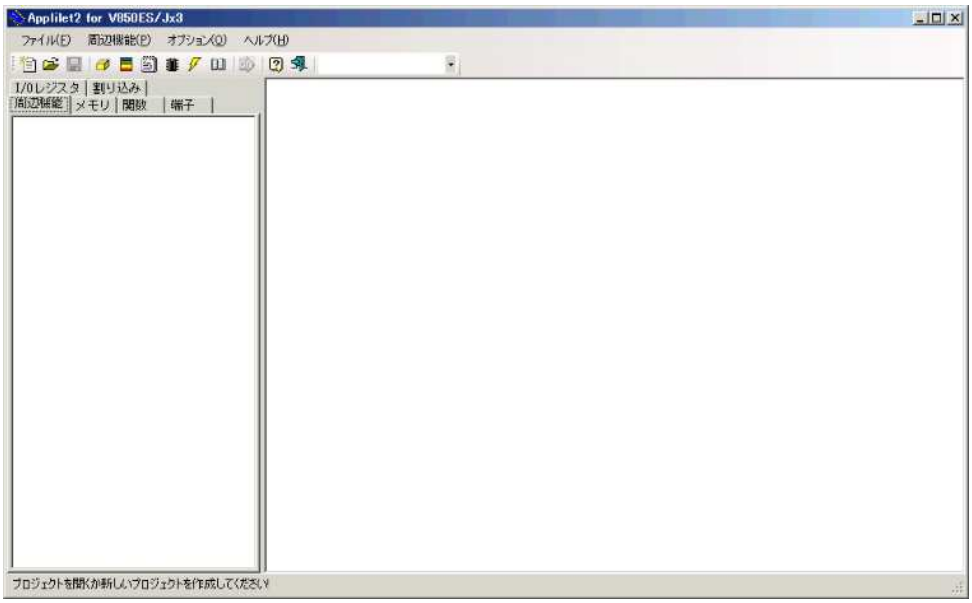

LEDの点滅速度が変化します

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。

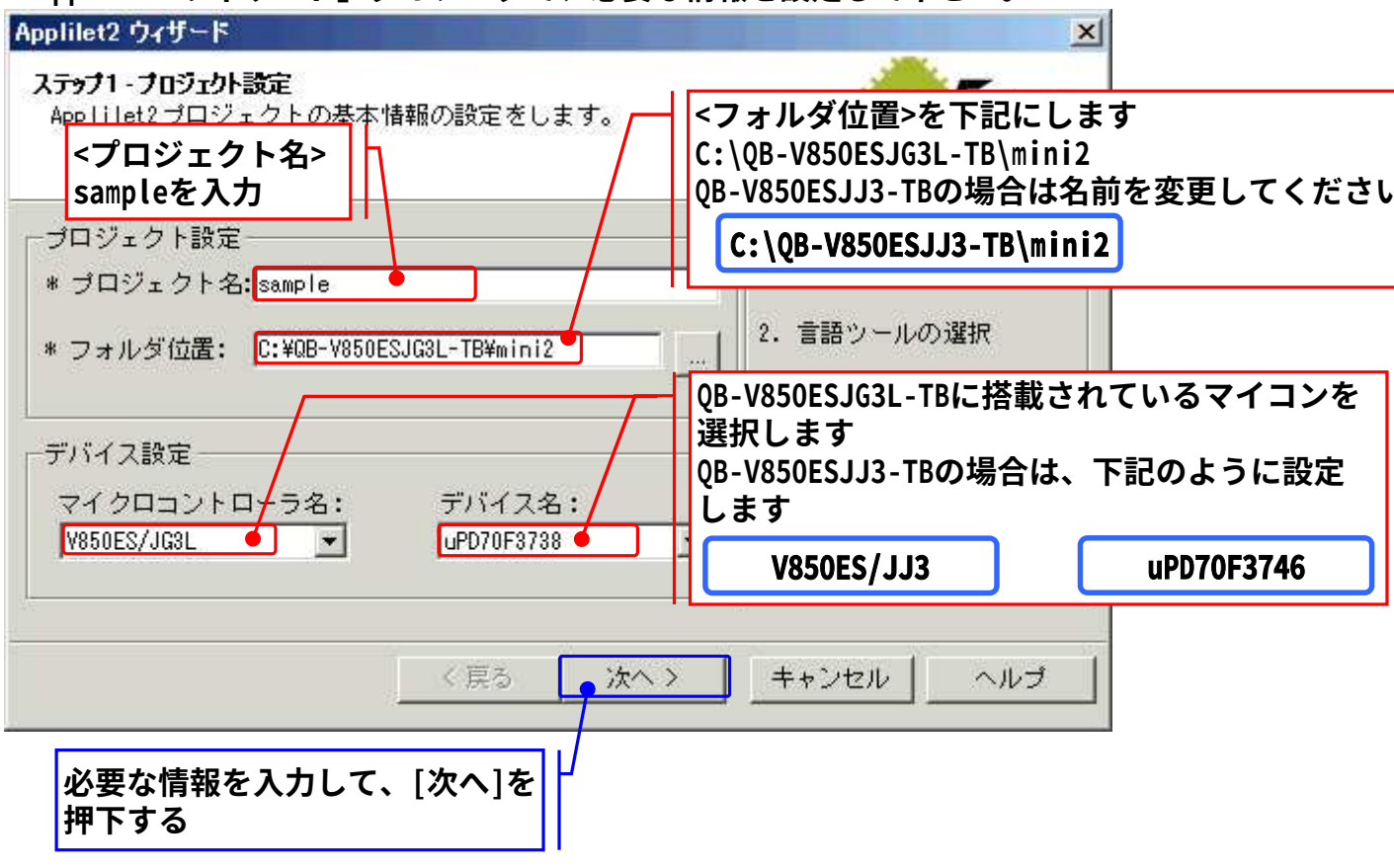
a. Applilet2を起動します。

[スタート] → [プログラム(P)] → [NEC Electronics Tools] →
[Applilet2 for V850ESJX33] → [Vx.xx] → [Applilet2 for V850ESJX3 Vx.xx]



b. Applilet2のプロジェクトファイルを新規に作成します。

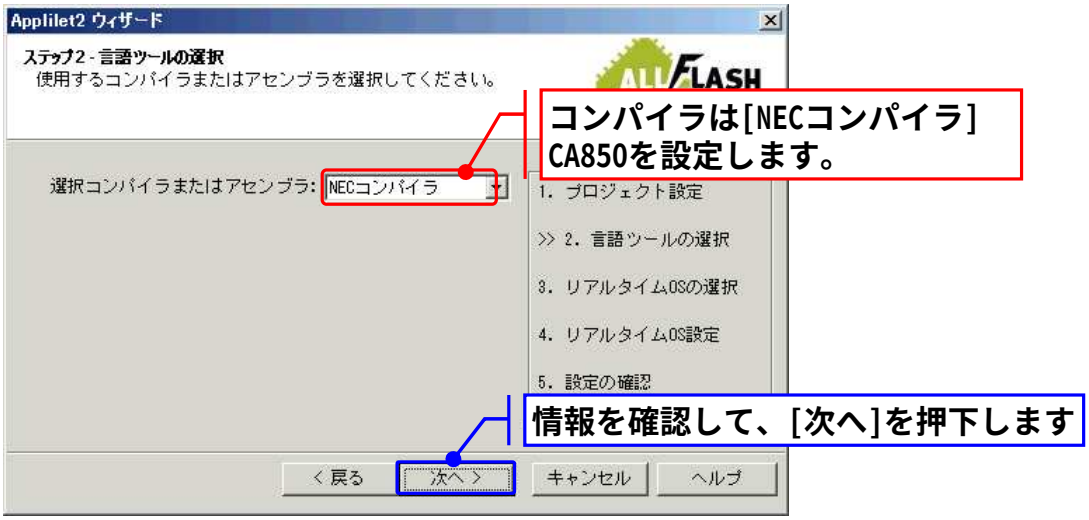
メニュー・バーの[ファイル(F)]→[新規作成(N)...]を選択します。
「Applilet2ウィザード」ダイアログで、必要な情報を設定して下さい。



Applilet2でプロジェクト作成

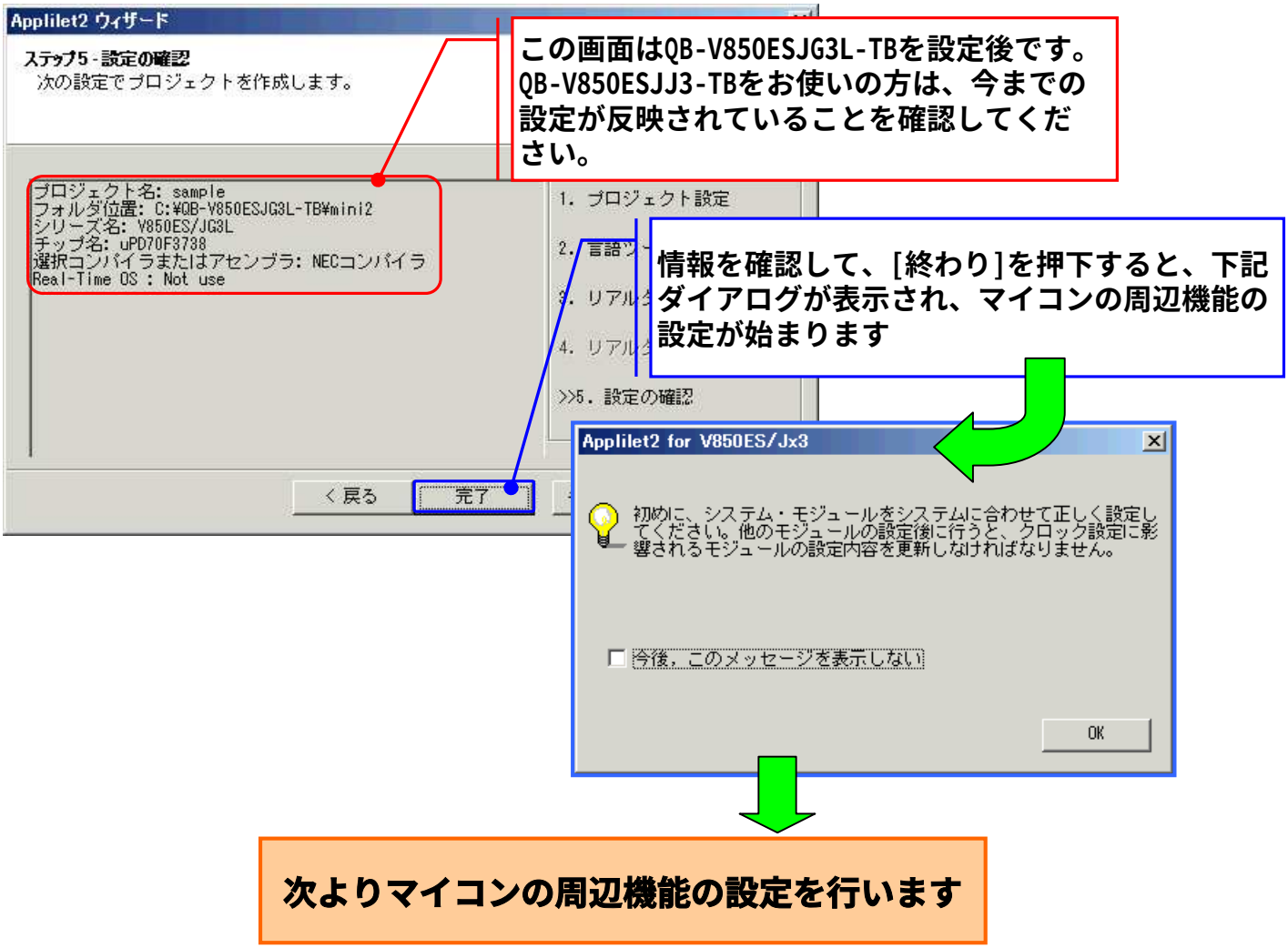
c. 言語ツールを選択します。

[NECコンパイラ]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。

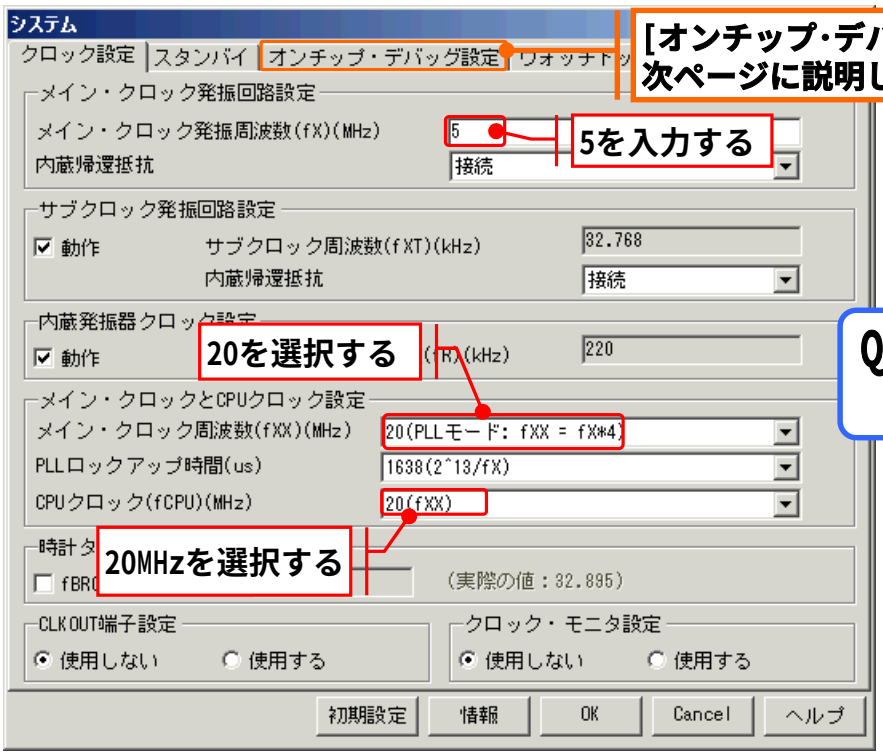


次よりマイコンの周辺機能の設定を行います

周辺機能設定[システム]

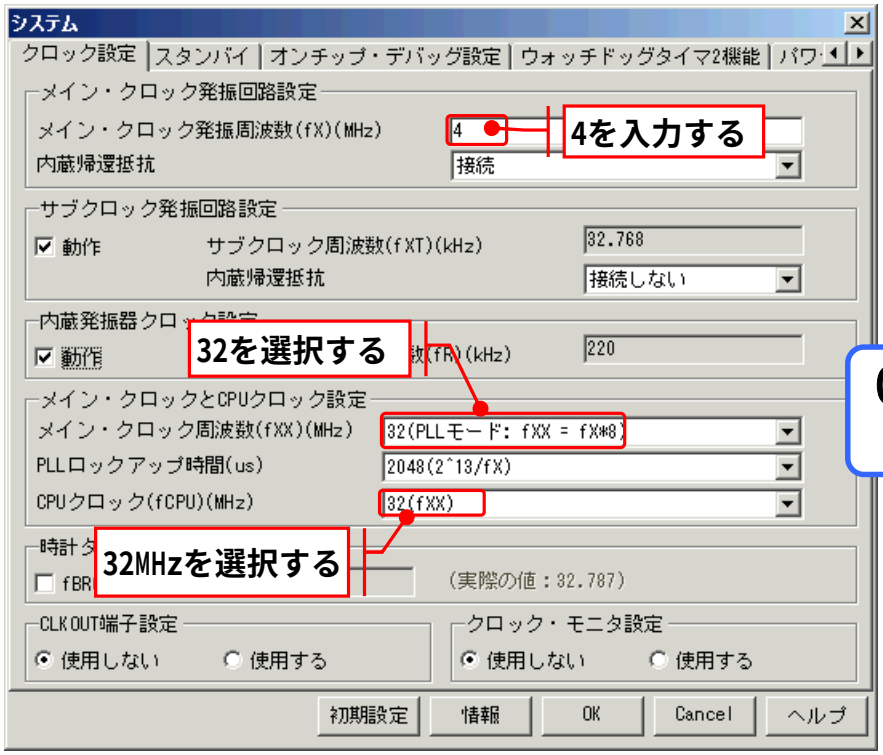
e. [システム]を設定します。

[システム]ダイアログは、クロック、デバッグ方法などマイコンの基本を設定します。



QB-V850ESJG3L-TB の場合

設定の違いはTB搭載クロックの周波数が異なるためです



QB-V850ESJJ3-TB の場合

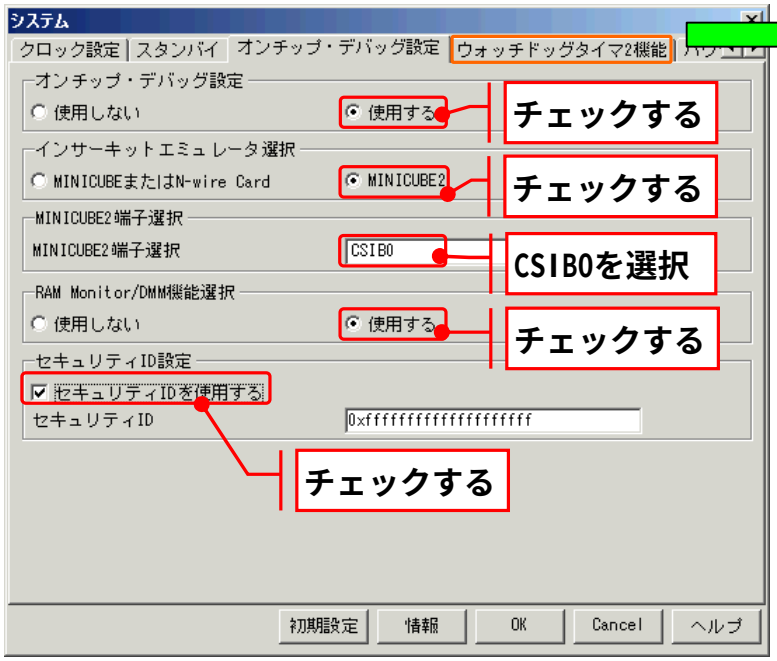
ワンポイント

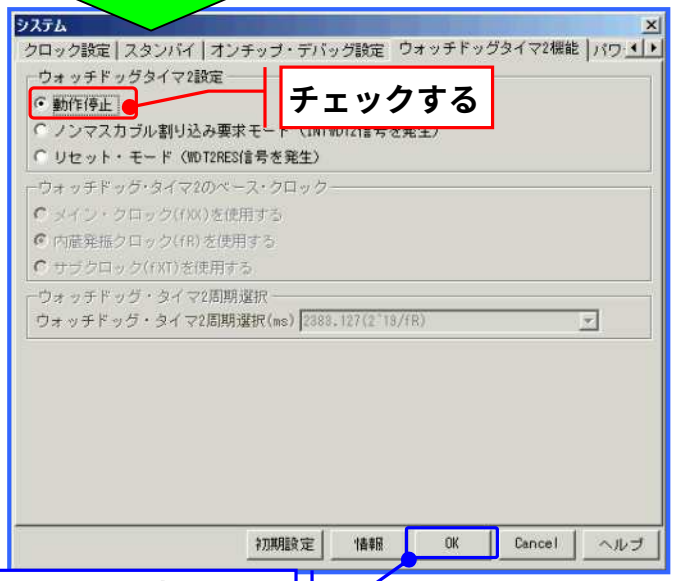
システムについて

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。

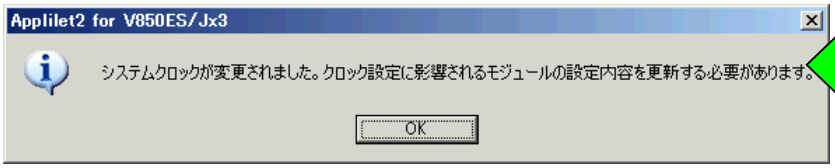
周辺機能設定[システム]

f. [システム]のオンチップ・デバッグ、ウォッチドッグタイマ2機能を設定します。





情報を確認して、[OK]を押下すると確認ダイアログが表示されます



QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
ともに同じ設定です

ワンポイント

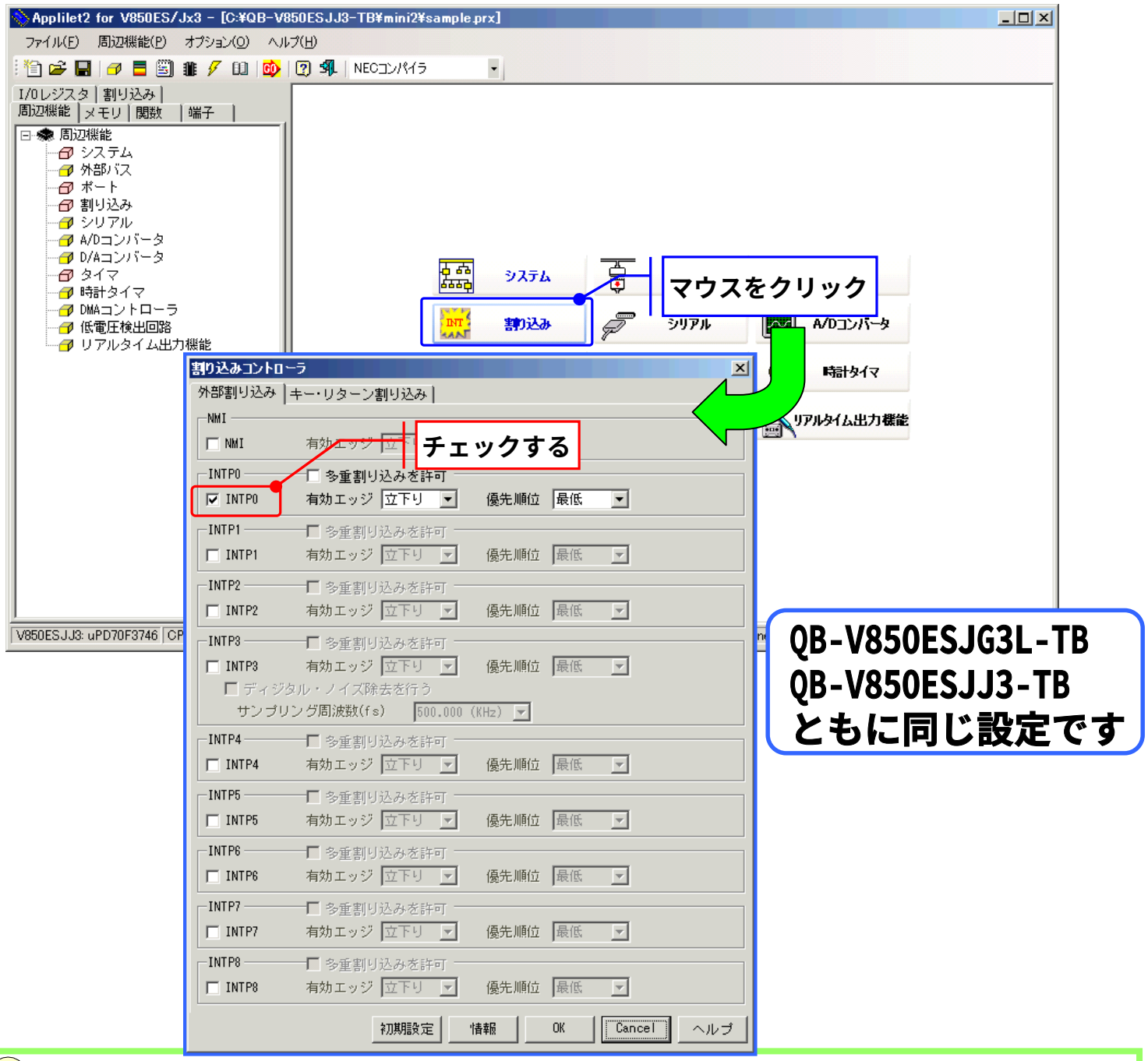
ウォッチドッグ･タイマについて

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。

周辺機能設定[割り込み]

g. [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。

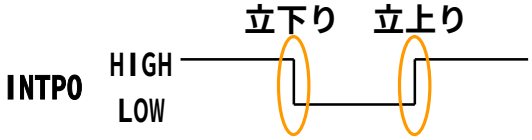


QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
ともに同じ設定です

ワンポイント

外部割り込みについて

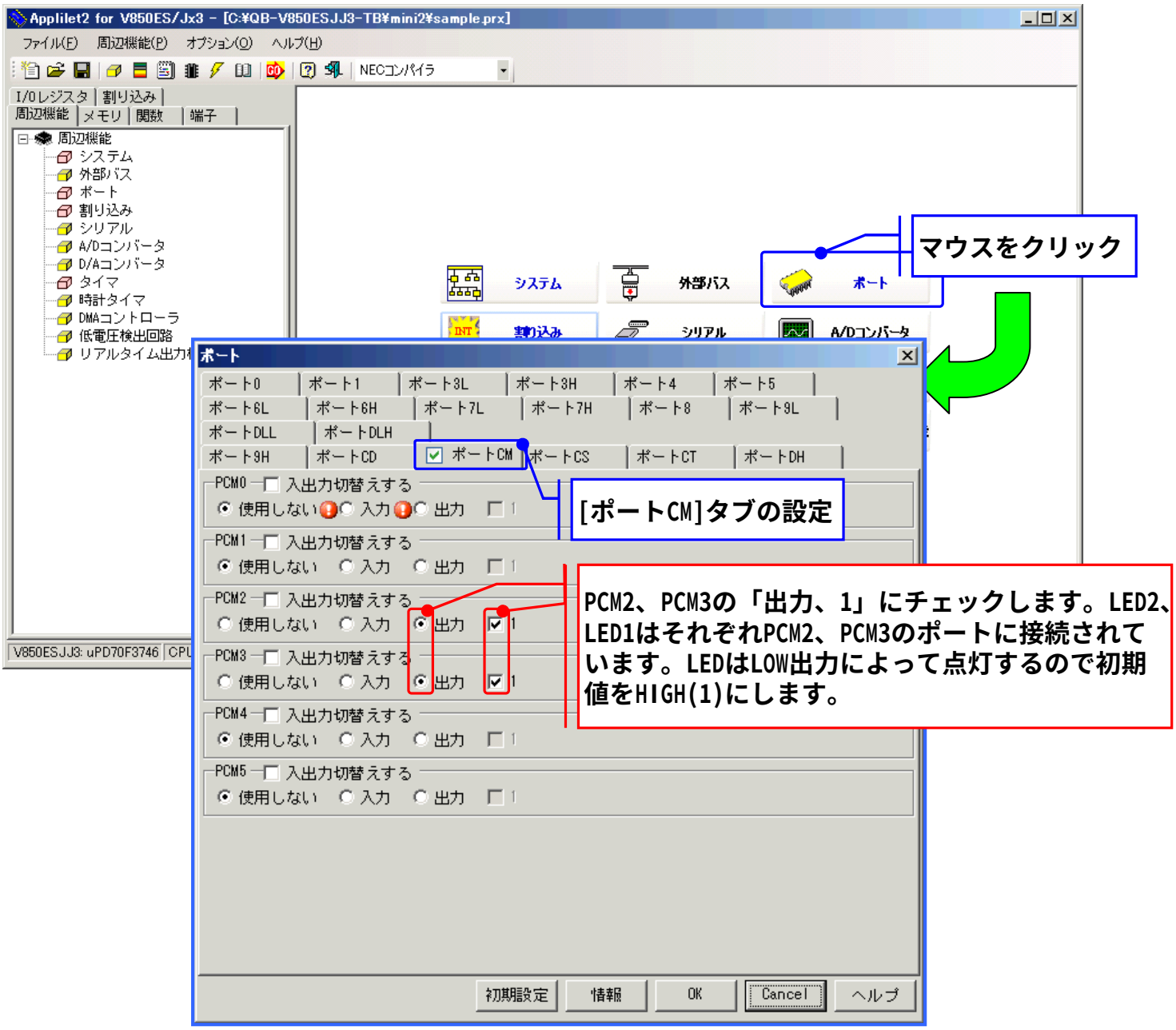
外部端子割り込み、キー割り込みの設定を行います。サンプルプログラムではINTP0の外部端子にSWが接続されています。有効エッジの**立下がりエッジ**とは、信号が1から0へ変化するときに有効とする設定です。ターゲット･ボードのSWを押した時が立ち下がり、押してからSWを離した時が立ち上がりになります。「立下がり」の逆が「立上がり」です。



周辺機能設定[ポート]

h. [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。



QB-V850ESJJ3-TBは上記の画面、QB-V850ESJG3L-TBについては画面は異なりますがPCM2、PCM3ともに同じ設定にします

ワンポイント

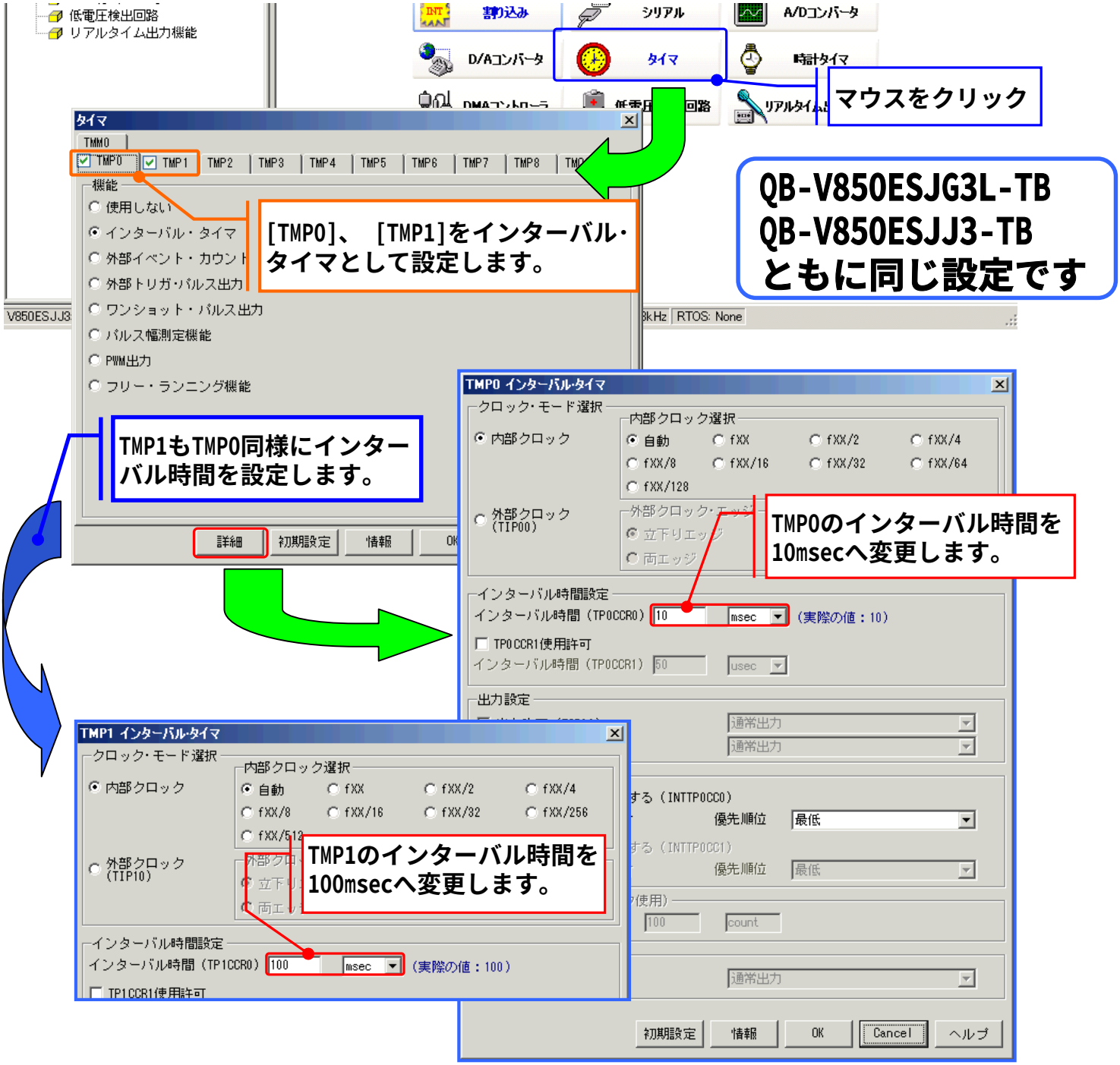
ポートについて

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]

i. [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

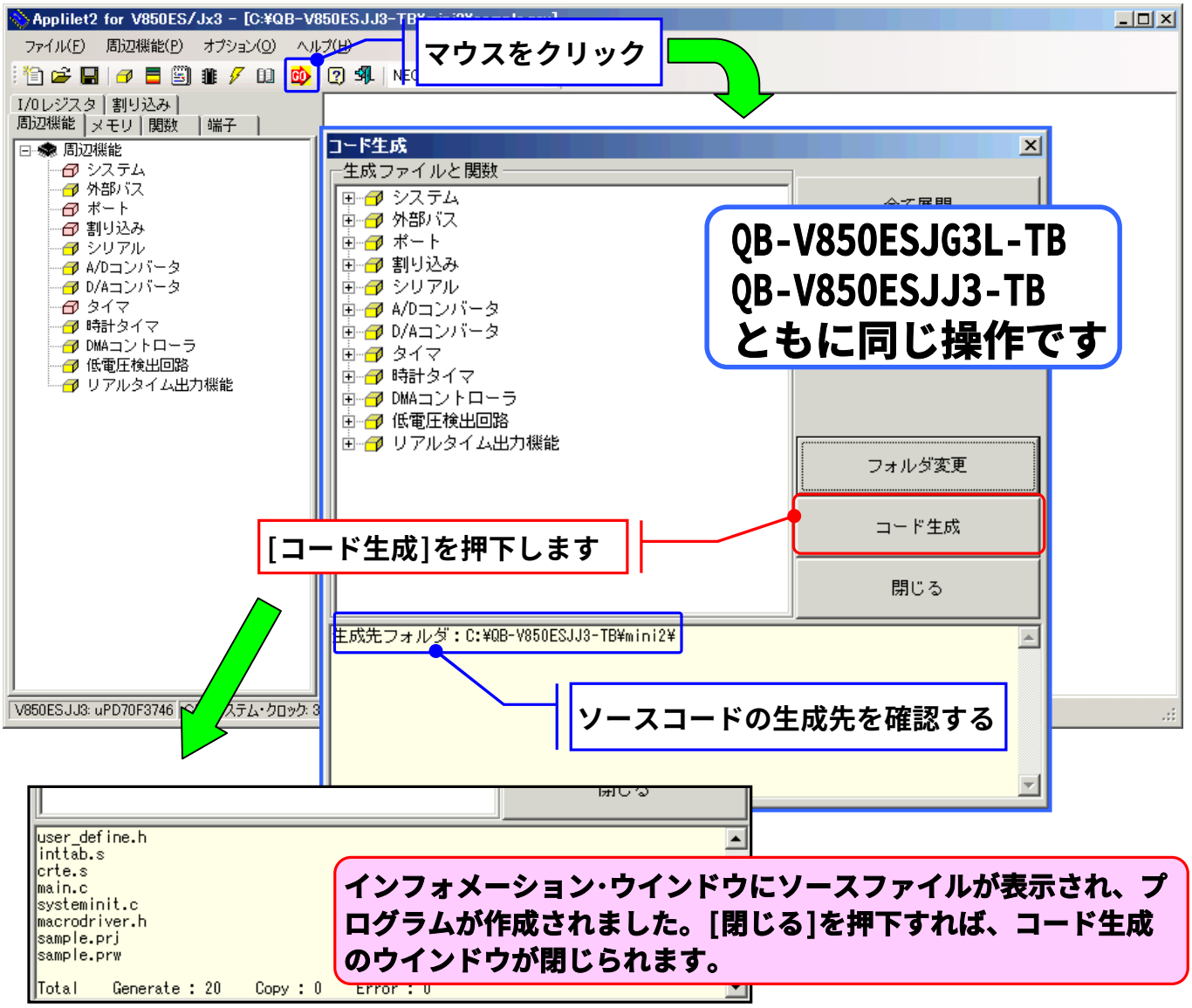
タイマにはインターバル・タイマの他にも様々な機能があります。

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

j. 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。



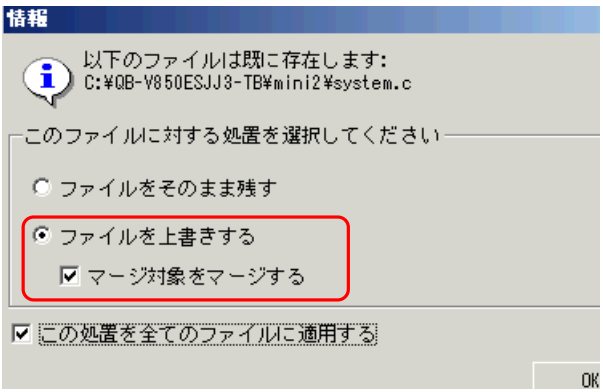
ワンポイント

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 */
```

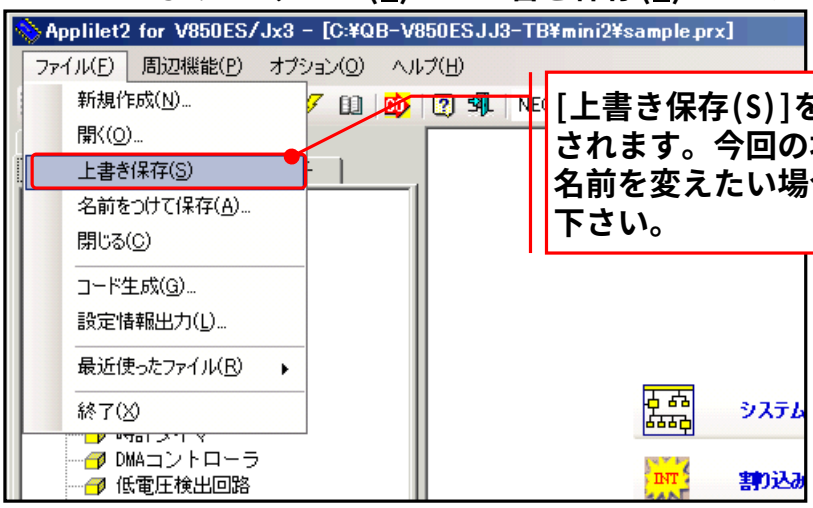
これはApplilet2の新機能「ソースコードのガード機能」です。



プロジェクトの保存

k. プロジェクトの保存

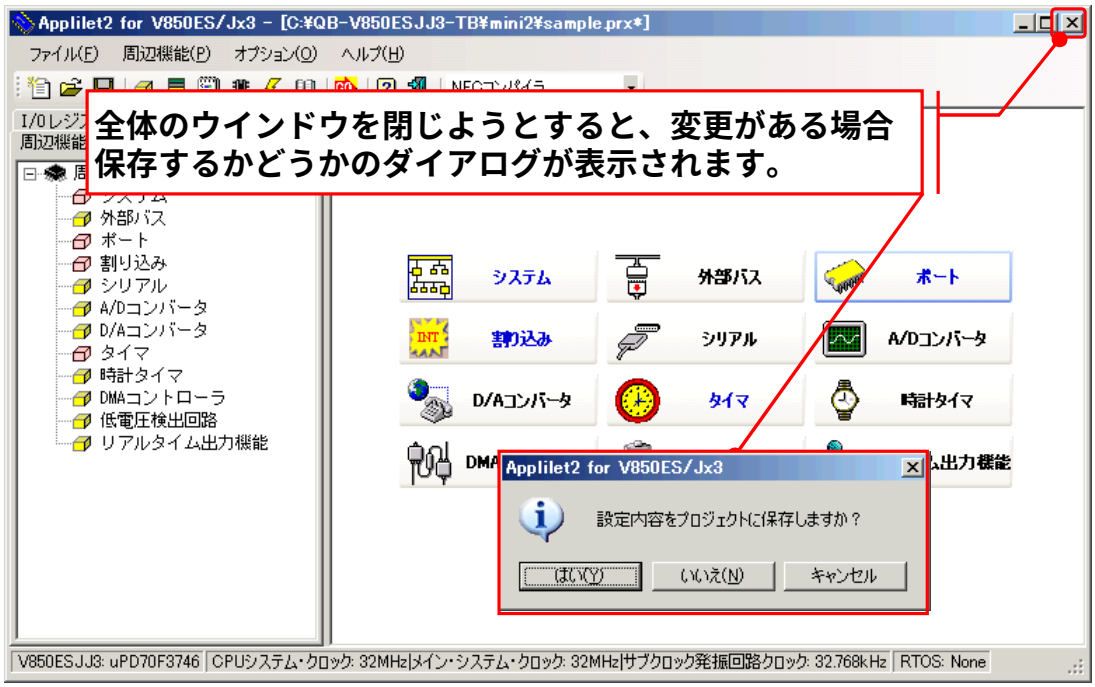
メニューより ファイル(F) -> 上書き保存(S)...でプロジェクトを保存します。



[上書き保存(S)]を行うと「プロジェクト名.prx」へ保存されます。今回の場合、「sample.prx」へ保存されます。名前を変えたい場合は[名前をつけて保存(A)]を選択して下さい。

QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
ともに同じ操作です

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

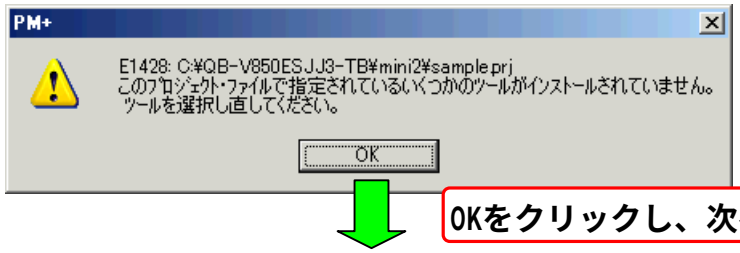
② 統合開発環境 PM+でプロジェクトファイルを読み込み環境の設定を行います。

k. PM+を起動します。

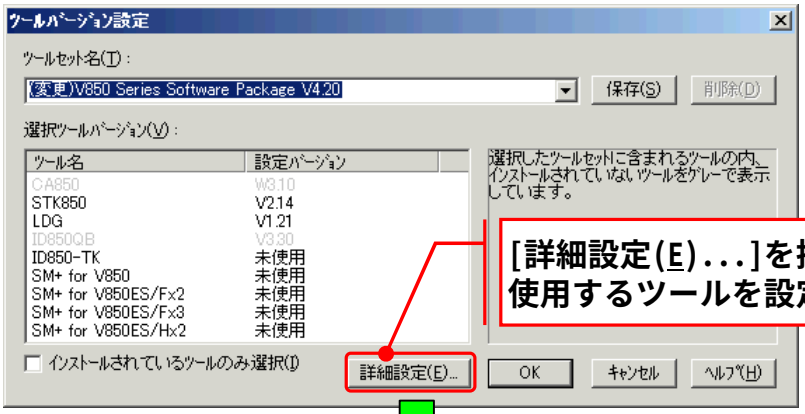
[スタート] → [プログラム(P)] → [NEC Electronics Tools] → [PM+ Vx.xx]

l.ワークスペース(sample.prw)を開きます。

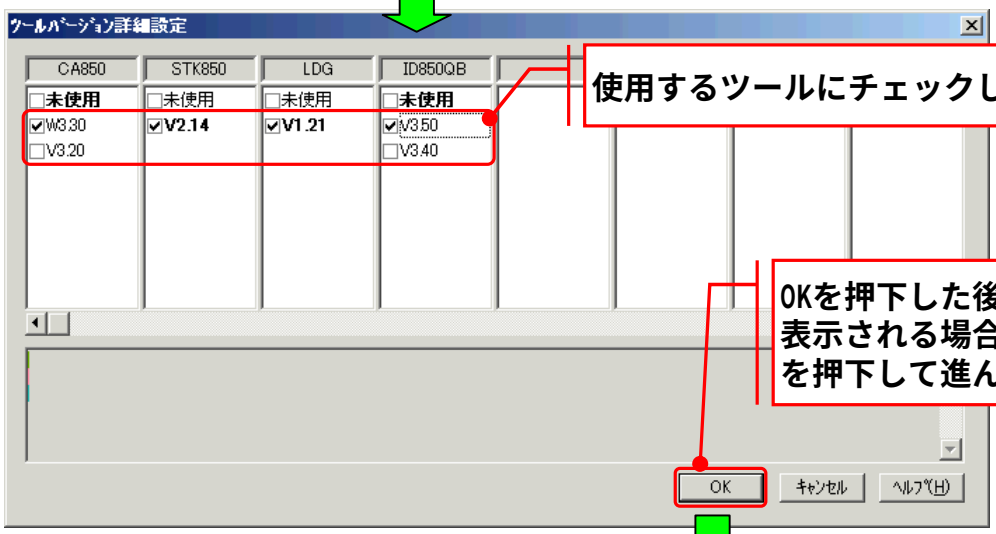
メニューの[ファイル(F)] → [ワークスペースを開く(W)...] を選択し、
”C:\QB-V850ESJJ3-TB\mini2\sample.prw ”を指定して、[開く(O)] ボタンを押して下さい。
インストールされているツールのバージョンが異なる場合、下記のダイアログが表示されます。



OKをクリックし、次へ進みます

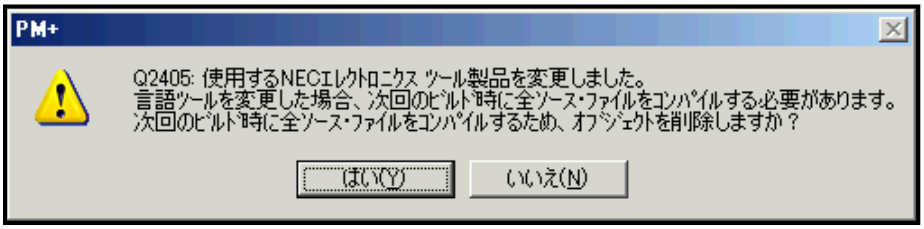


[詳細設定(E)...]を押下し、
使用するツールを設定します。



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。

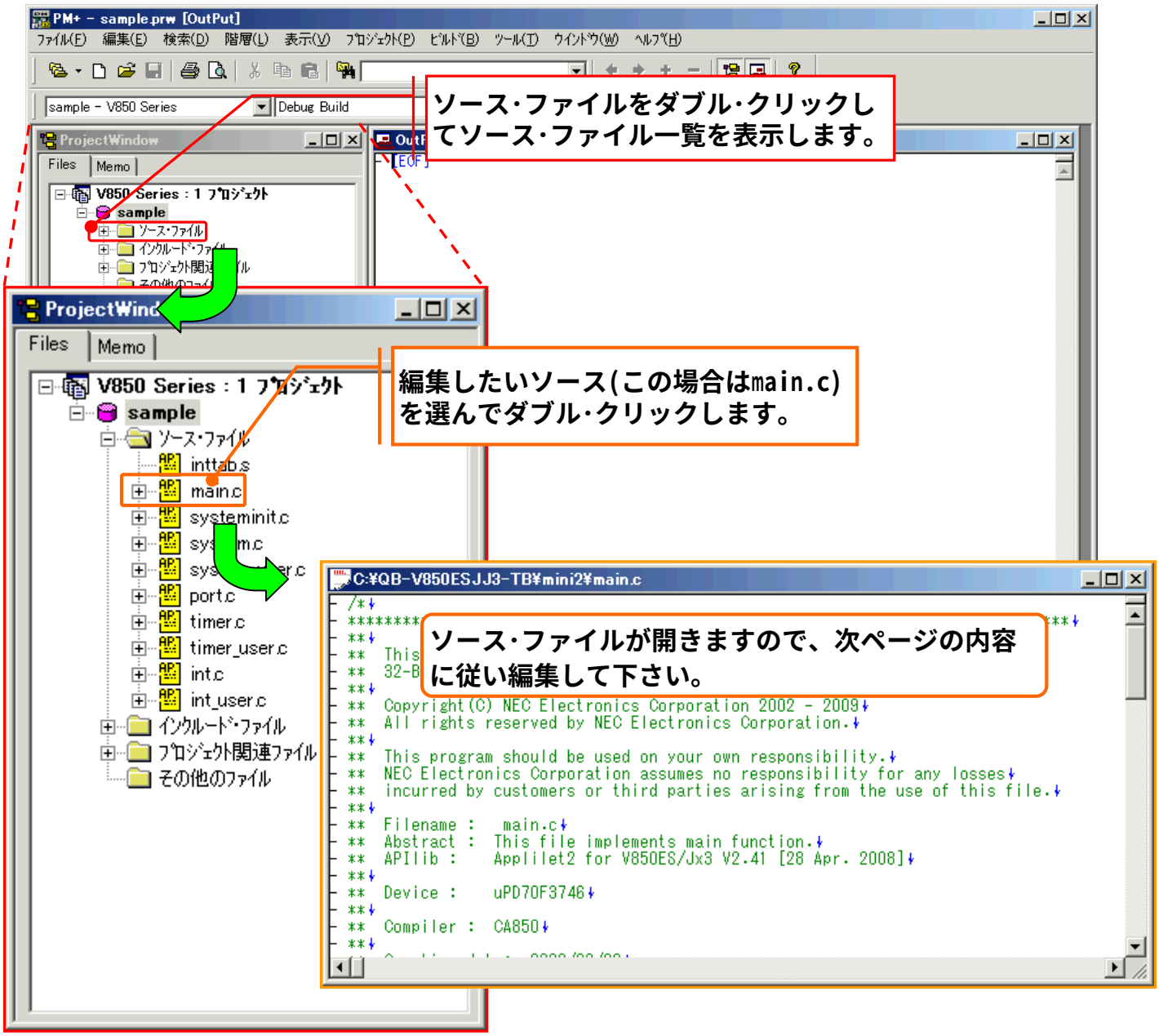


プログラムの編集

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。

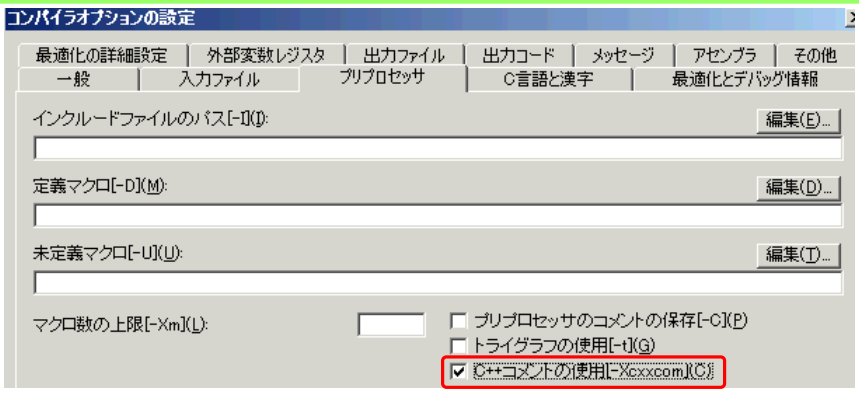
[ProjectWindow] の ソース･ファイルをダブル･クリック → 編集したいソースをダブル･クリックします。次ページの内容に従って、編集して下さい。



ワンポイント

ソースの記述について

ソースファイル中のコメント//で記述することができます。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。
プリプロセッサタブを選択し、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。

QB-V850ESJG3L-TB、QB-V850ESJJ3-TB どちらでも同じプログラムです

main.c

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

       TMP0_Start();
       TMP1_Start();
       INTP0_Enable();

    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;
UCHAR g_onetime_sw;

/* 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

timer_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 ] = { 1, 3, 7, 15, 31, 47, 63, 127 };

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

/*-----------------------------------------------------------------------
**  Abstract:   This function is INTTPOCC0 interrupt service routine.
**-----------------------------------------------------------------------
*/
__interrupt void MD_INTTPOCC0(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 = PCM.2;
        outreg = inreg ^ 1;
        PCM.2 = outreg;
        PCM.3 = inreg;
    }
    g_interval_count--;
    /* End user code. Do not edit comment generated here */
}

/*-----------------------------------------------------------------------
**  Abstract:    This function is INTTP1CC0 interrupt service routine.
**-----------------------------------------------------------------------
*/
__interrupt void MD_INTTP1CC0(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

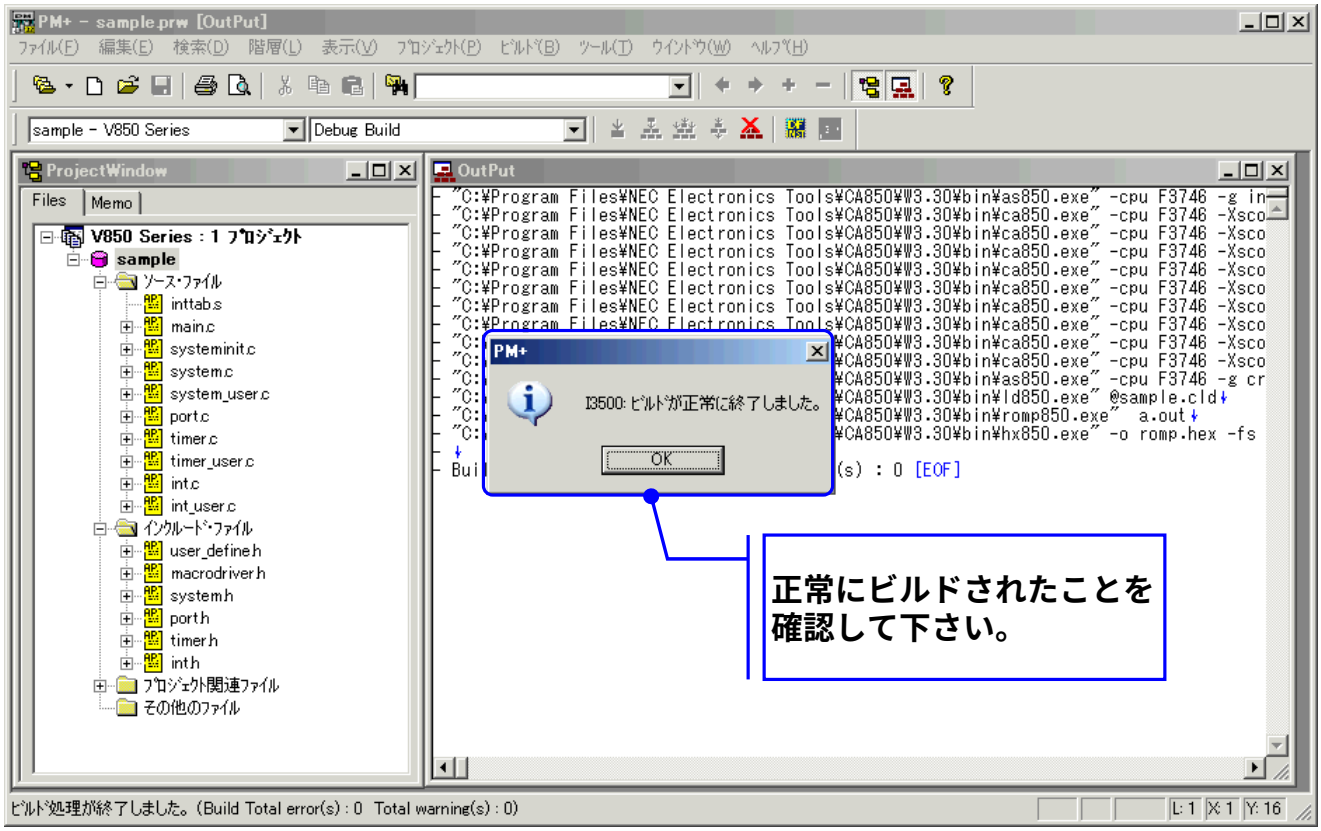
```
/***********************************************************************
**  Macro define
***********************************************************************
*/
   /* Start user code for definition. Do not edit comment generated here */
   extern UCHAR g_onetime_sw;
   extern UCHAR g_interval_sw;
   extern UINT  g_interval_count;
   /* End user code for definition. Do not edit comment generated here */
```

コードを追加して下さい

プログラムのビルド

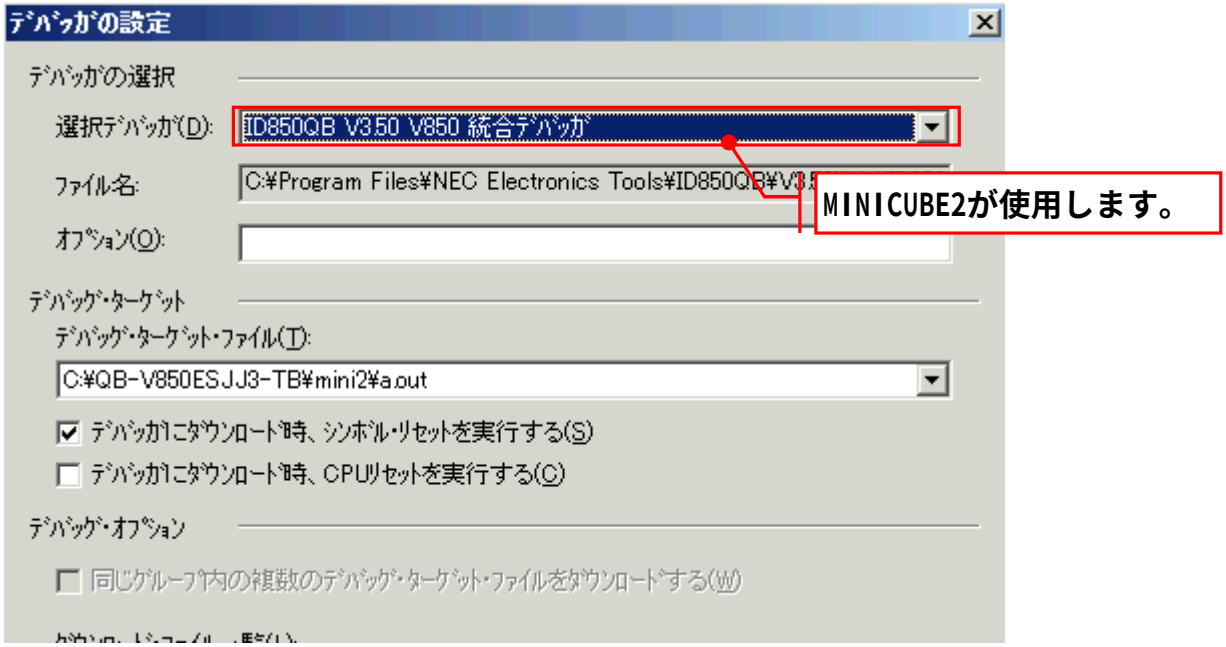
o. プログラムをビルドします。

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
「ID850QB Vx.xx V850 統合デバッガ」を選択して下さい。



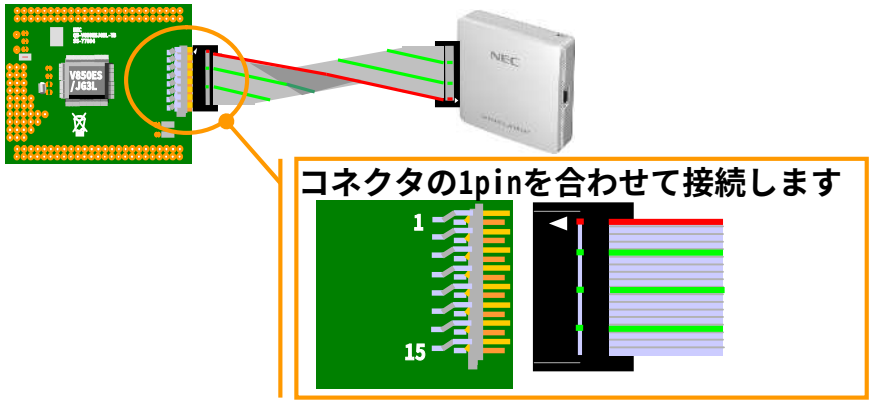
MINICUBE2の接続

④ 実際にQB-V850ESJG3L-TB、またはQB-V850ESJJ3-TB上でプログラムを実行します。

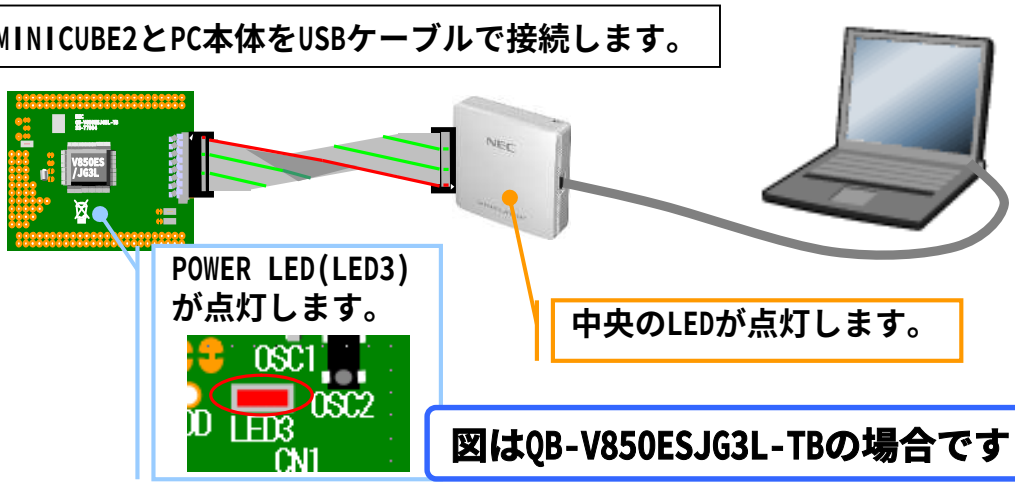
q. PC本体、MINICUBE2、ターゲットボードを接続します。
接続する順番に注意して接続します。

1.MINICUBE2のスイッチを設定します。

2.QB-V850ESJJ3-TBまたはQB-V850ESJG3L-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。



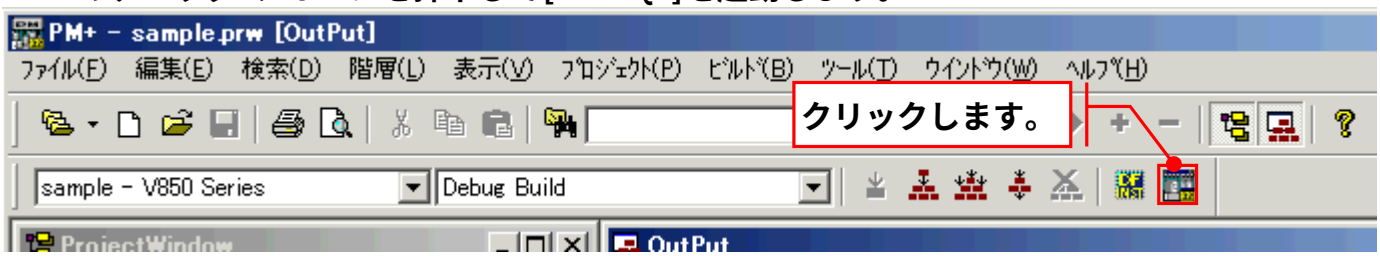
ワンポイント

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。V850ES/Jx3の動作電圧が2.7～3.6Vなので3V供給で動作します。ターゲットに周辺機能など回路を追加する場合、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

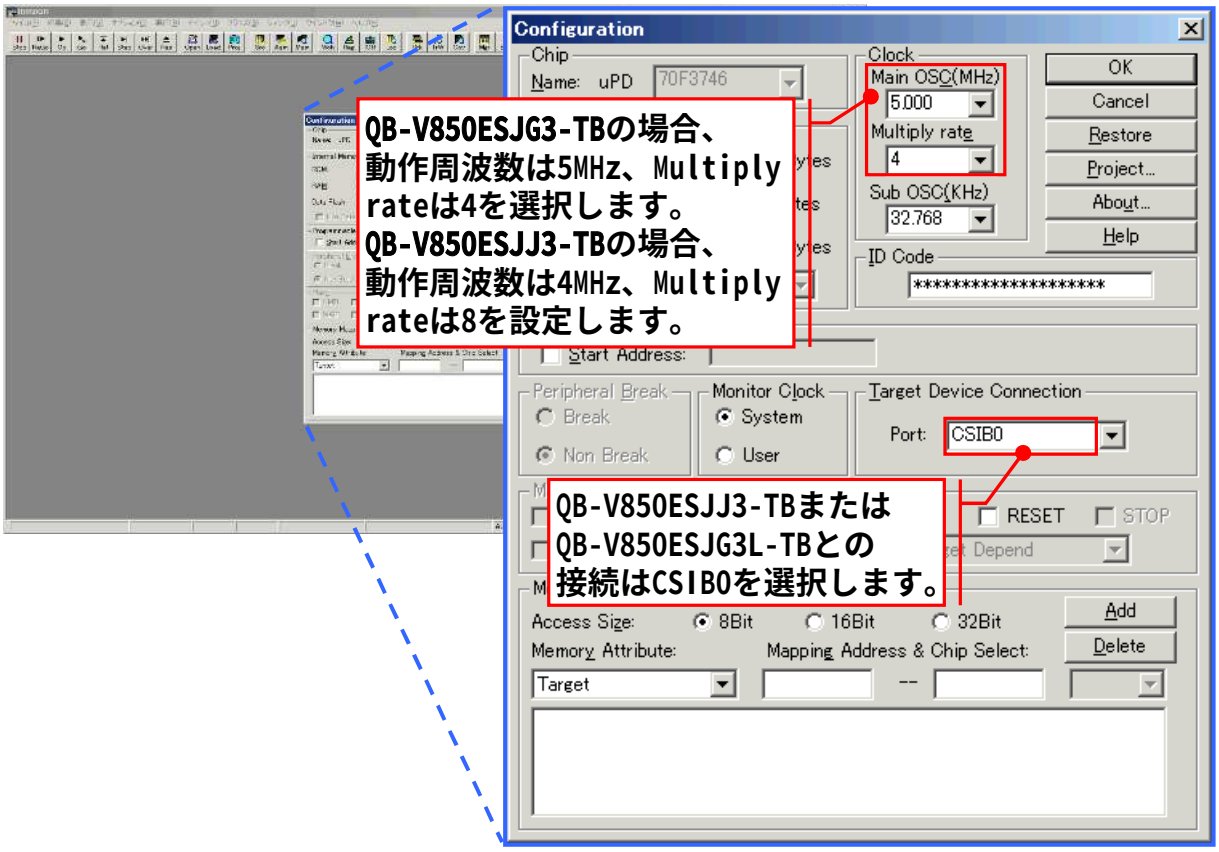
r. デバッガを起動します。

PM+のデバッグ・アイコンを押下して[ID850QB]を起動します。



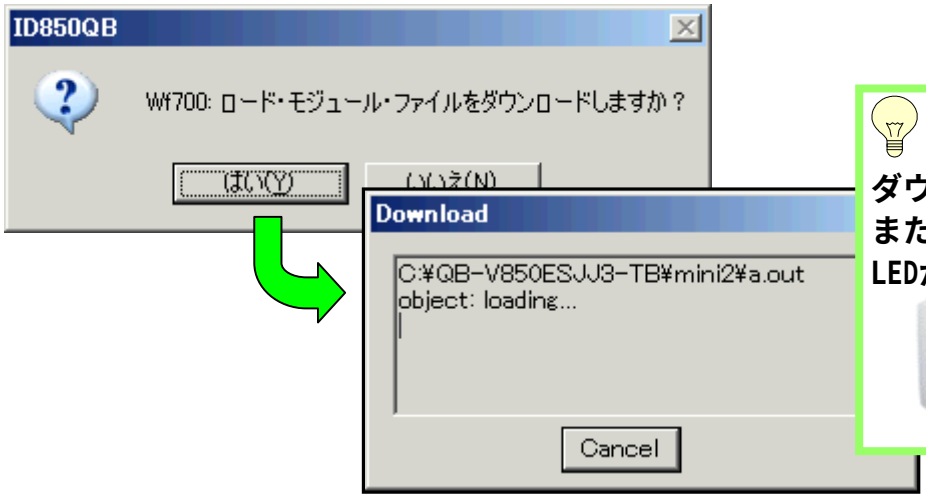
s. デバッガの設定を行います。

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。



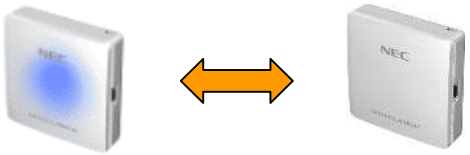
t. プログラムをダウンロードします。

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

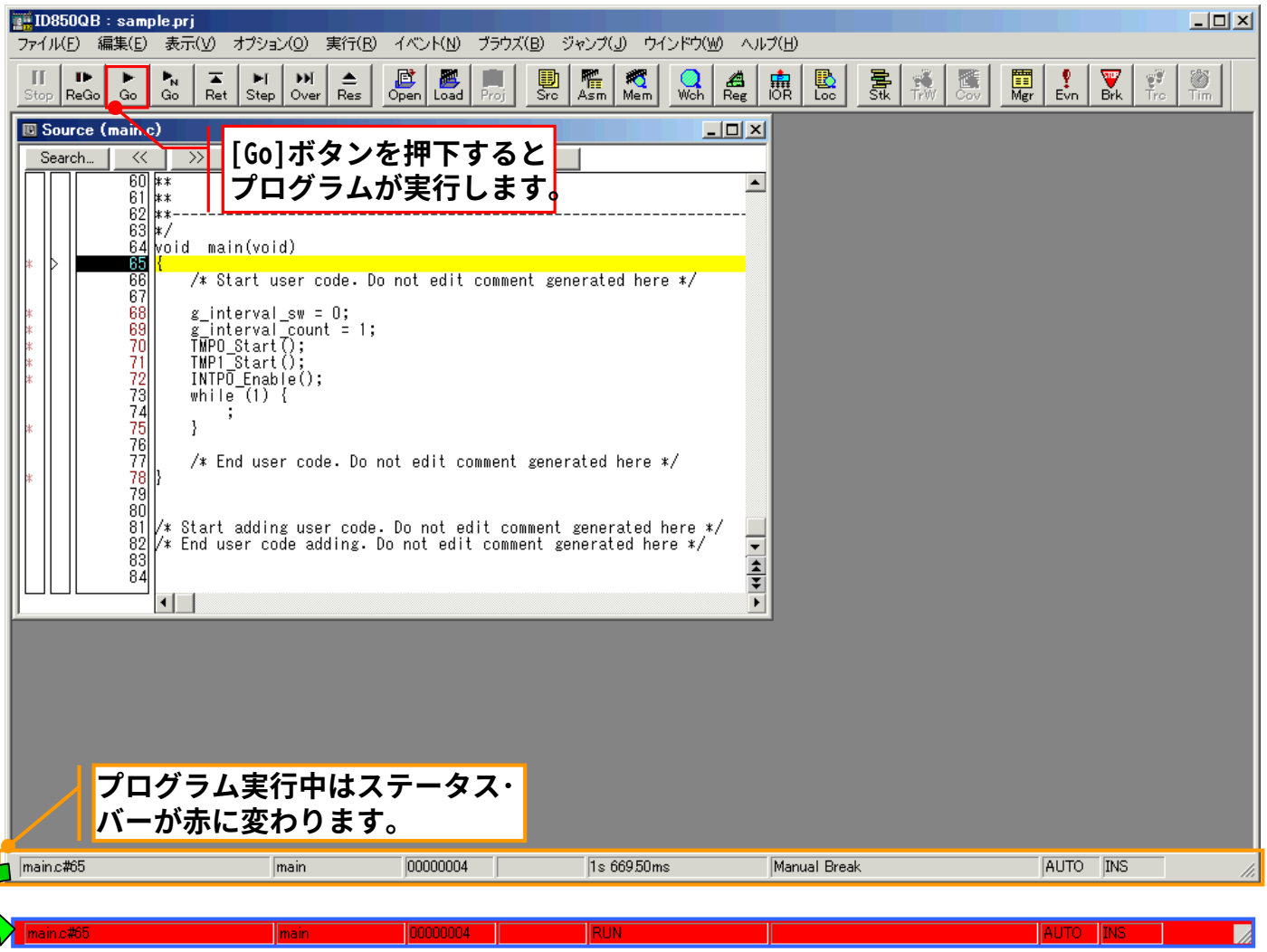
ダウンロードの時間は4～5秒ほどです。
また、ダウンロード中はMINICUBE2のLEDが1秒間隔で点滅します。

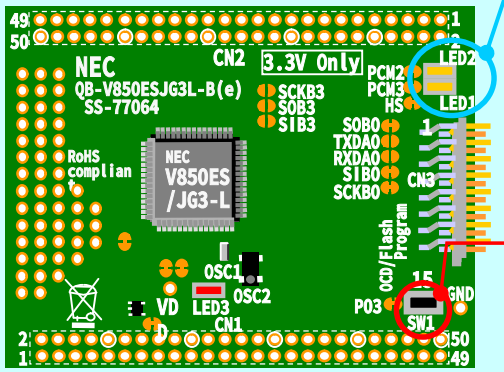


プログラムの実行

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。





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。

SWを押下するとLEDの点滅速度が変化します。

QB-V850ESJG3L-TB ターゲット・ボード動作の場合

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。

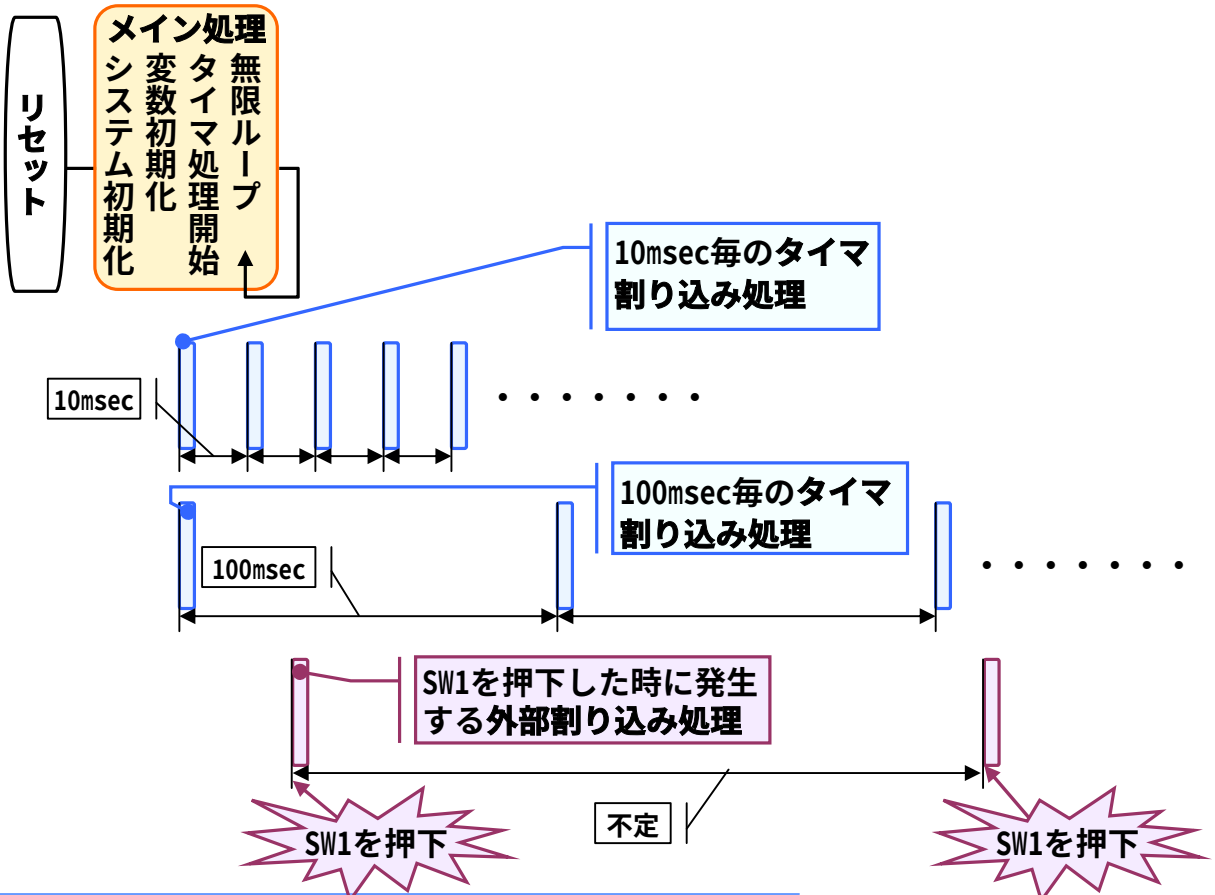
メイン処理:
マイコンがリセットされてから動作する処理です。変数初期化、タイマの開始を行った後は無限ループになります。

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:
ターゲットボード上の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

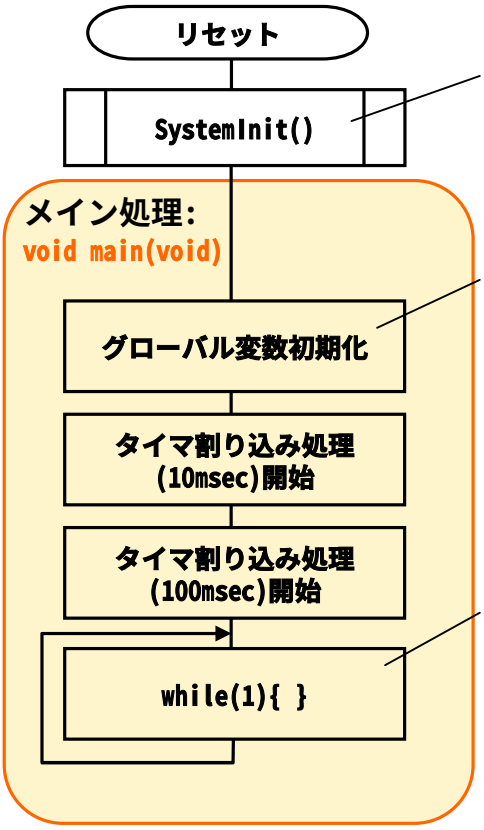
[メイン]は初期化とタイマ処理開始だけの処理です。プログラムは[タイマ割り込み]、[外部割り込み]で動作します。



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PCM2,PCM3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。

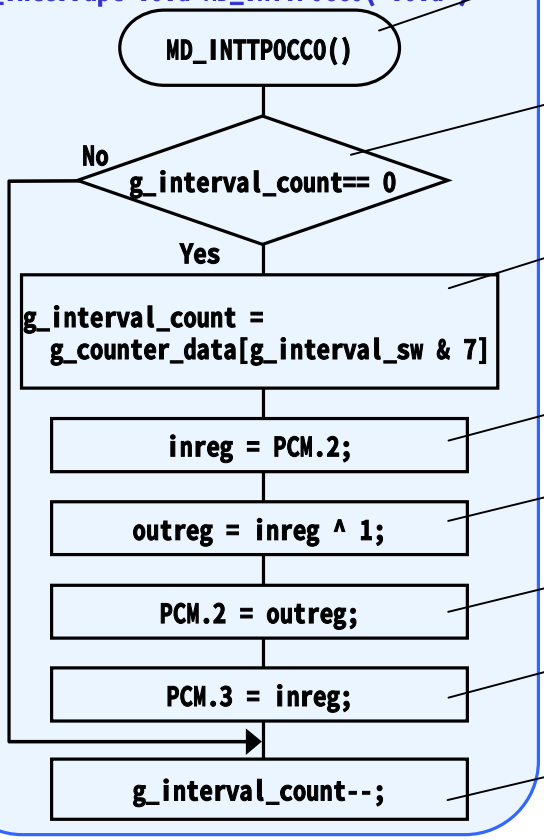
グローバル変数の説明

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_INTTPOCC0( 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LED2(ポートCMのbit2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。

LED2(ポートCMのbit2接続)に outregの値を入れます。

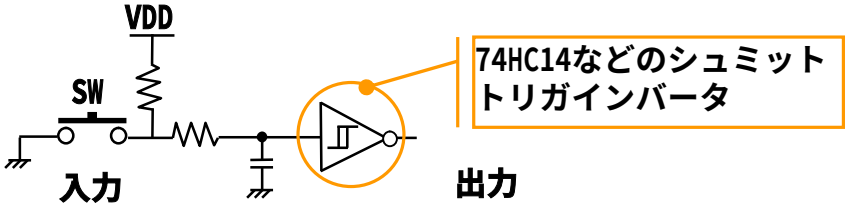
LED1(ポートCMのbit3接続)に inregの値を入れます。

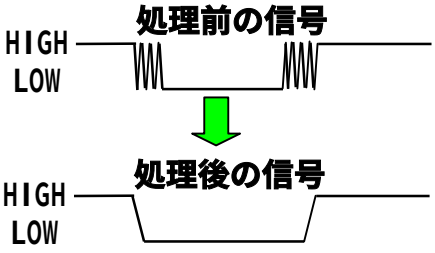
g_interval_countの値を減らします。

ワンポイント

チャタリングについて

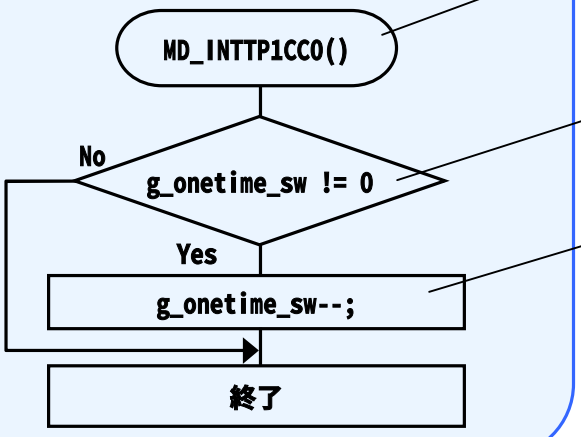
SWを押した時、実際には機械的な接点では瞬間的にON<->OFFを繰り返します。これを「チャタリング」と言います。QB-V850ESJJ3-TB及びQB-V850ESJG3L-TBは、ハードウエア回路でチャタリング防止を行っていません。そのためチャタリング防止をソフトウエアで行う必要があります。ハードウエアで、チャタリング防止する方法としてはシュミットトリガインバータ(74HC14など)を使うのが一般的です。この方法はノイズ除去にも使われます。





プログラムの説明

タイマ割り込み処理(100msec毎):
`__interrupt void MD_INTTP1CC0( void )`

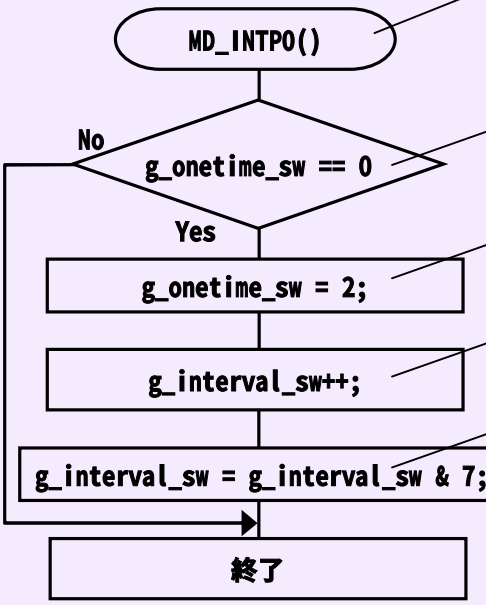


100msec毎に発生するインターバル･タイマです。一度押したSWを200msec以内は無視するための処理です。

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。

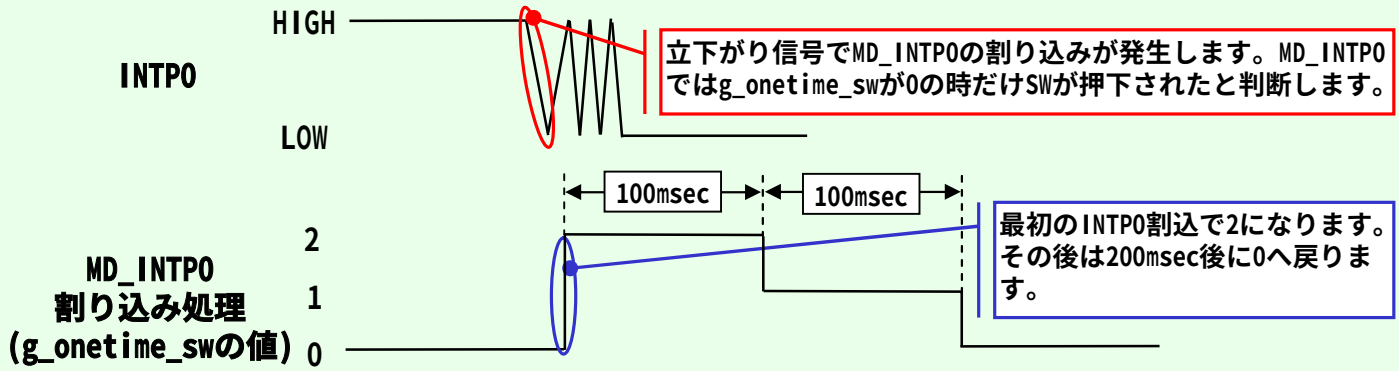
g_onetime_swをチェックします。

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。

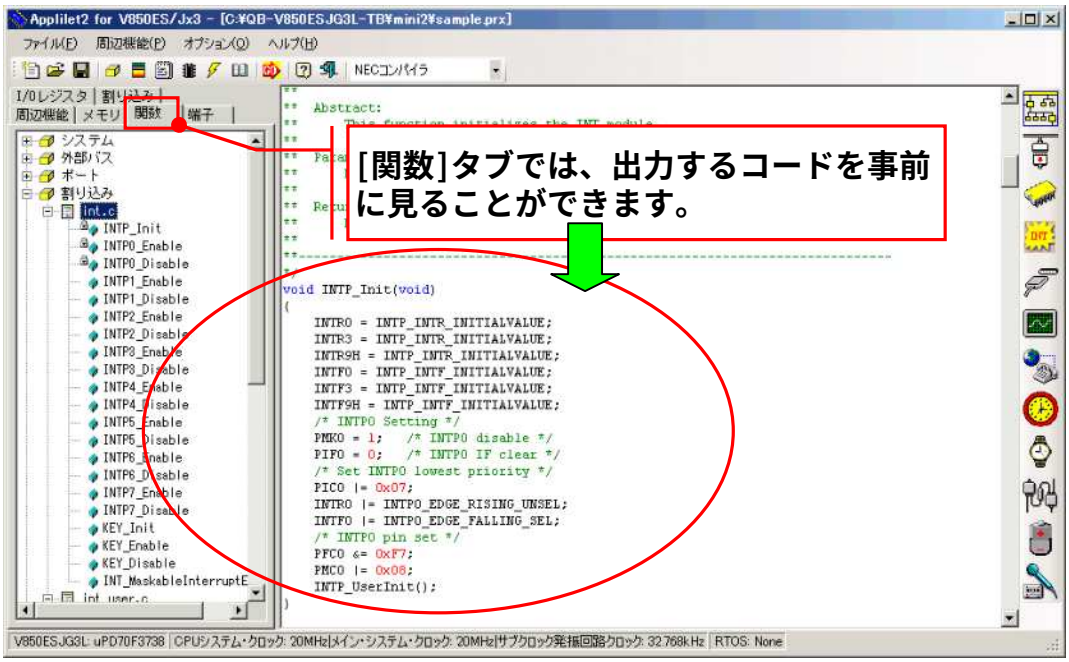
プログラムの説明

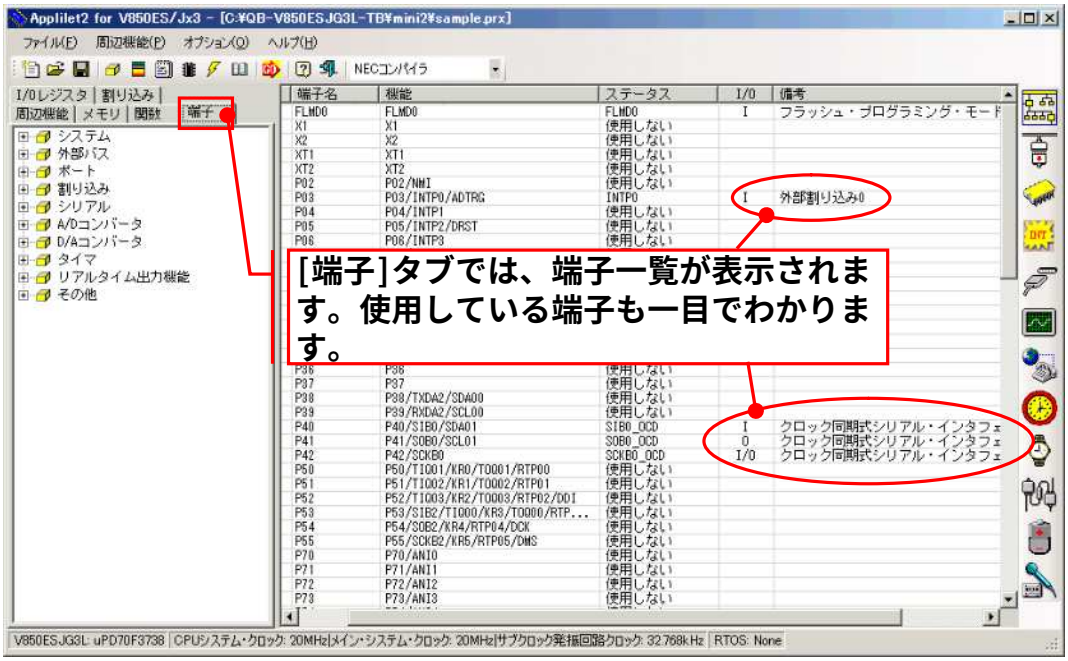
プログラムの説明は以上です。追加するコードの小さい事に驚いたのではないでしょうか？デバイスドライバ･コンフィギュレータ「Applilet2」を使えば、誰でも手軽にマイコンの開発ができる事でしょう。Applilet2の情報については、下記URLよりアクセスして下さい。
http://www.necel.com/micro/ja/development/asia/applilet2

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。

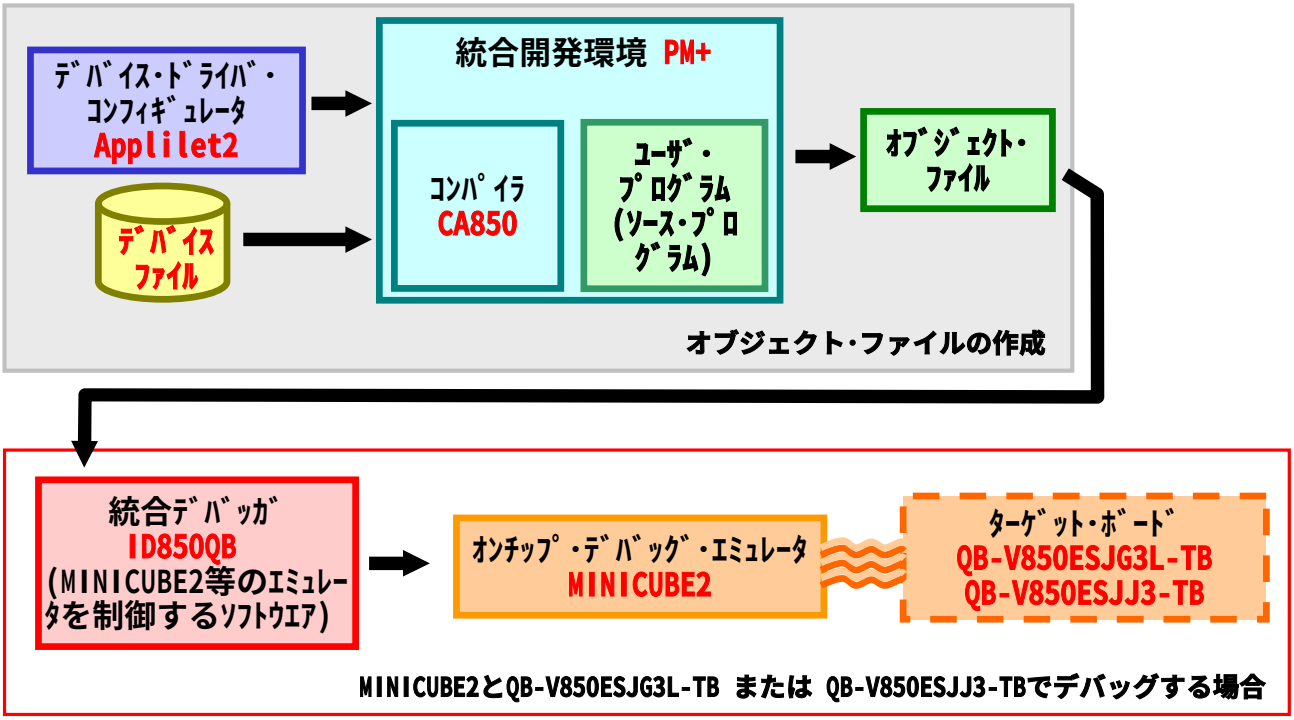




デバッグの基本

[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 + ターゲットボード)で統合デバッガ(ID850QB)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(ID850QB)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/v850/icemulator.html

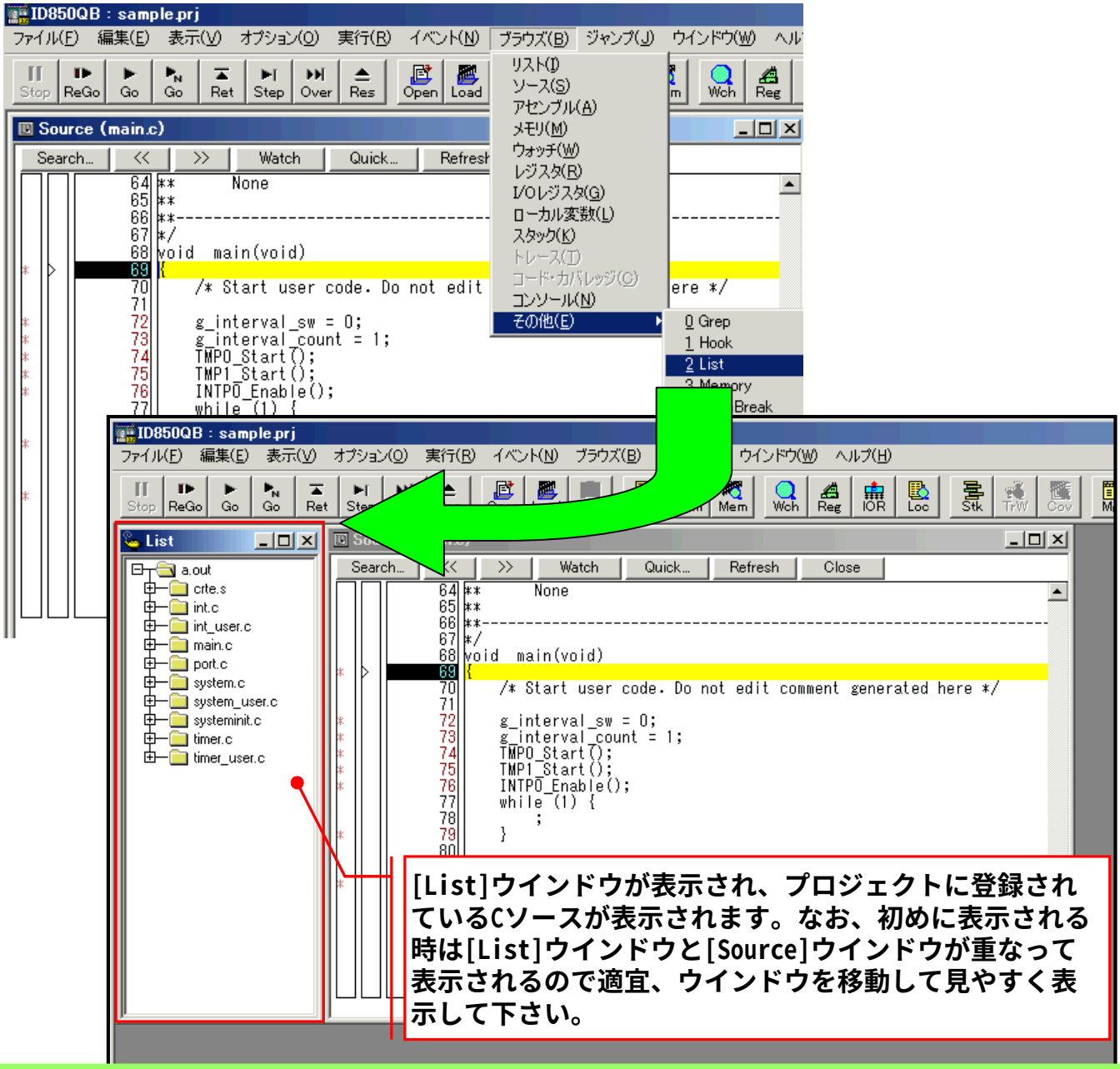
デバッガの基本画面

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。



[List]ウインドウが表示され、プロジェクトに登録されているCソースが表示されます。なお、初めに表示される時は[List]ウインドウと[Source]ウインドウが重なって表示されるので適宜、ウインドウを移動して見やすく表示して下さい。

ワンポイント

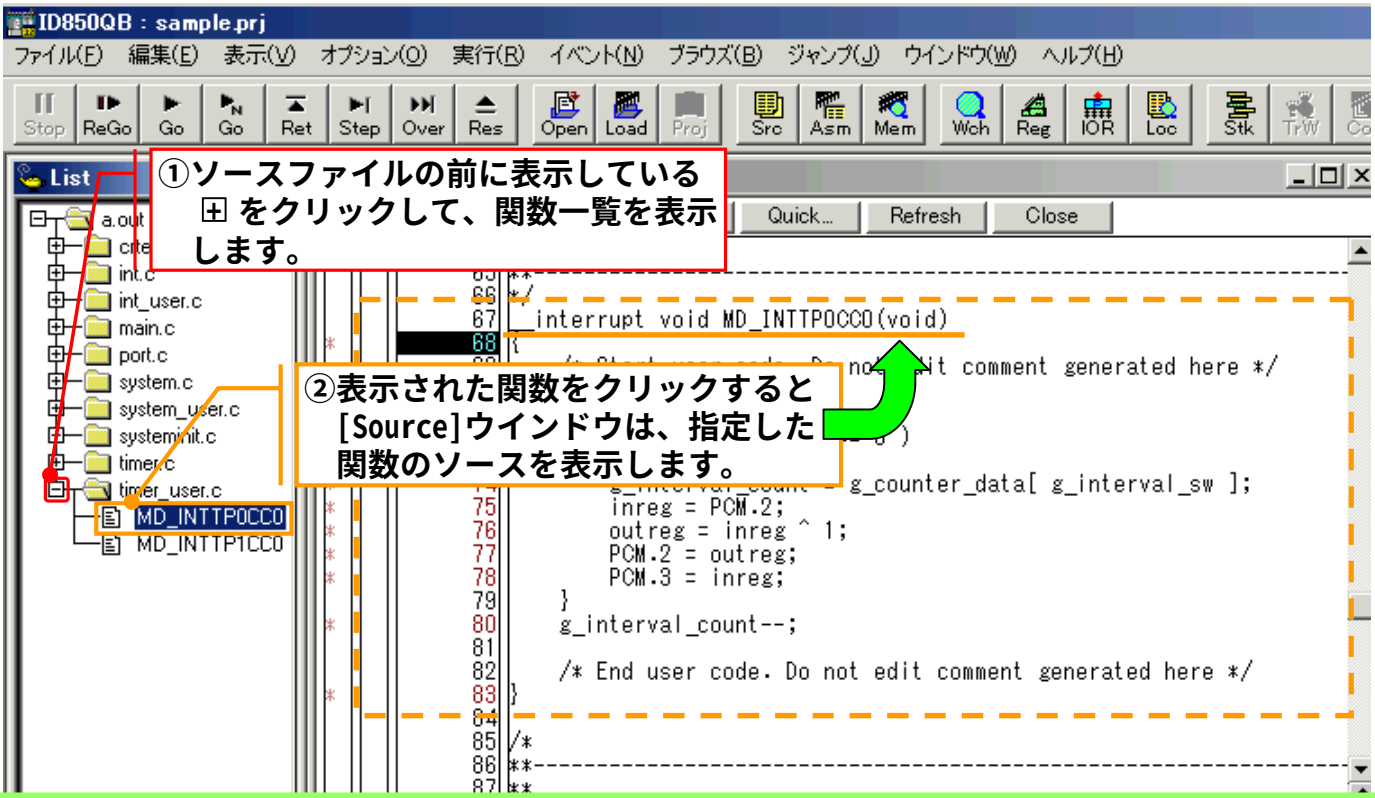
デバッガ(ID850QB)の環境ファイルについて

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。

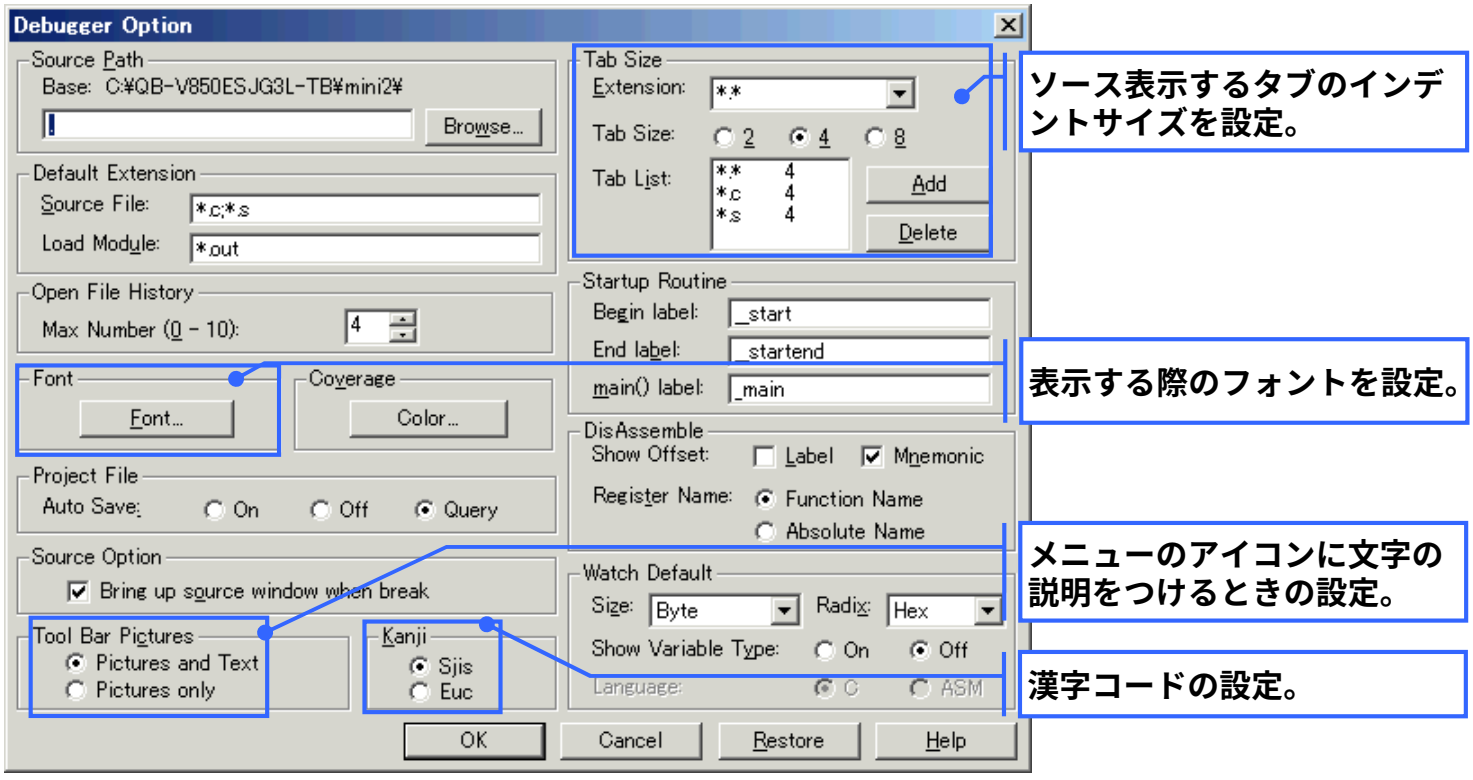
[List]ウインドウのソースファイルをクリックして関数一覧を表示します。関数が表示されたらその関数をクリックすると、[Source]ウインドウの表示が変わります。



ワンポイント

デバッガ(ID850QB)のオプションについて

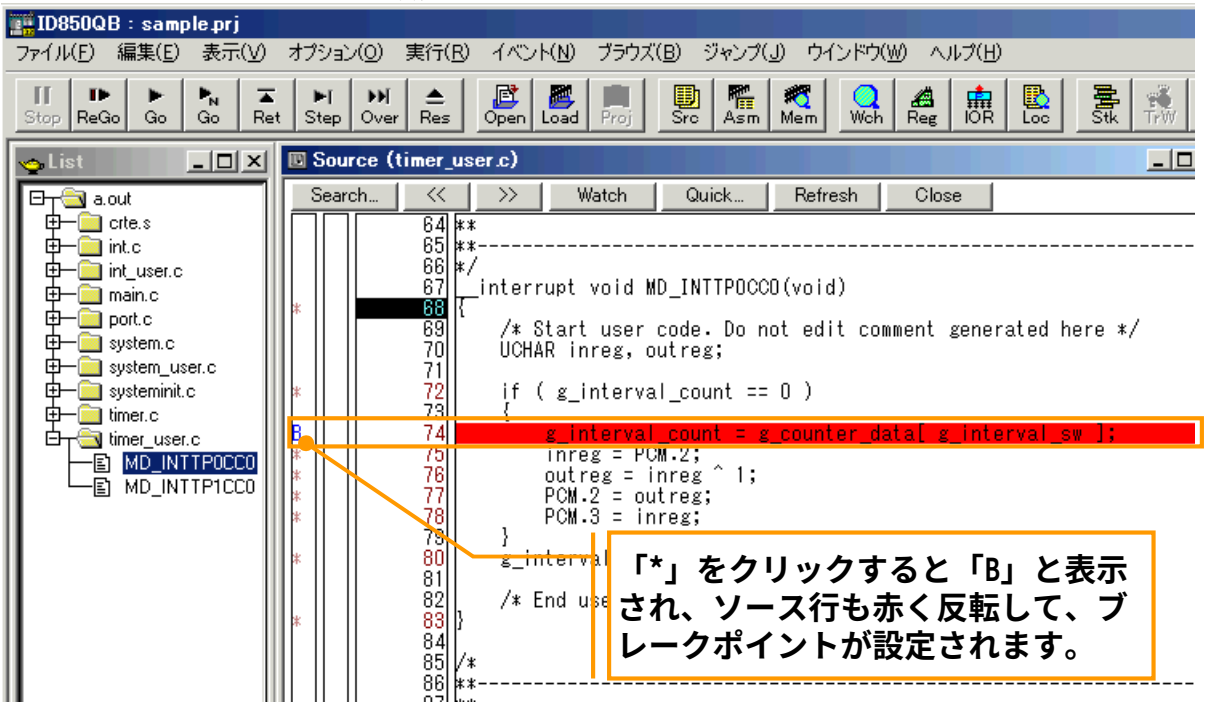
メニューの[オプション(O)] → [デバッガ･オプション(G)...] で表示の変更ができます。[Debugger Option]ダイアログが表示されますので、各種設定して下さい。



ブレークポイントの設定

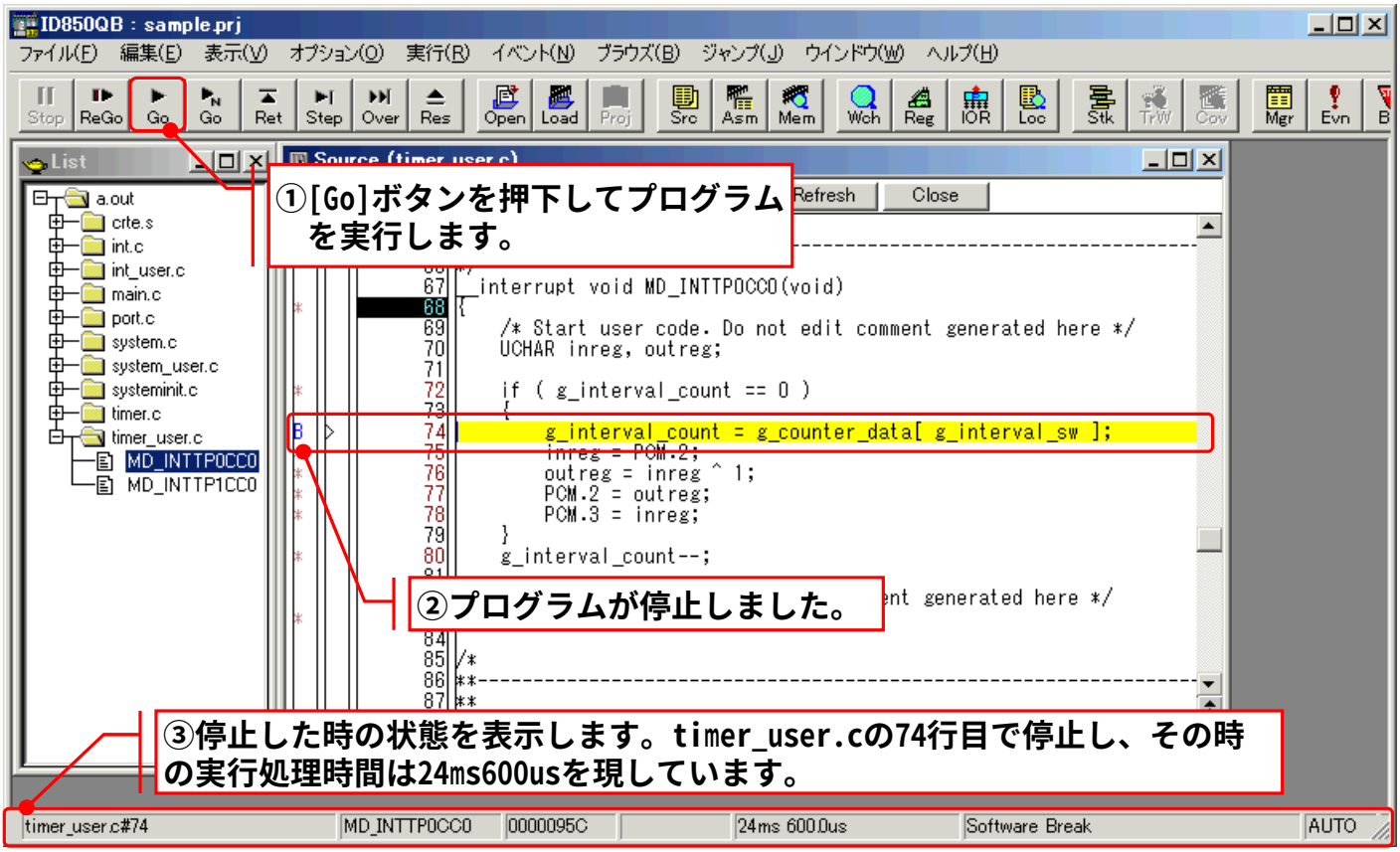
d. ブレークポイントを設定したい画面を表示します。

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。

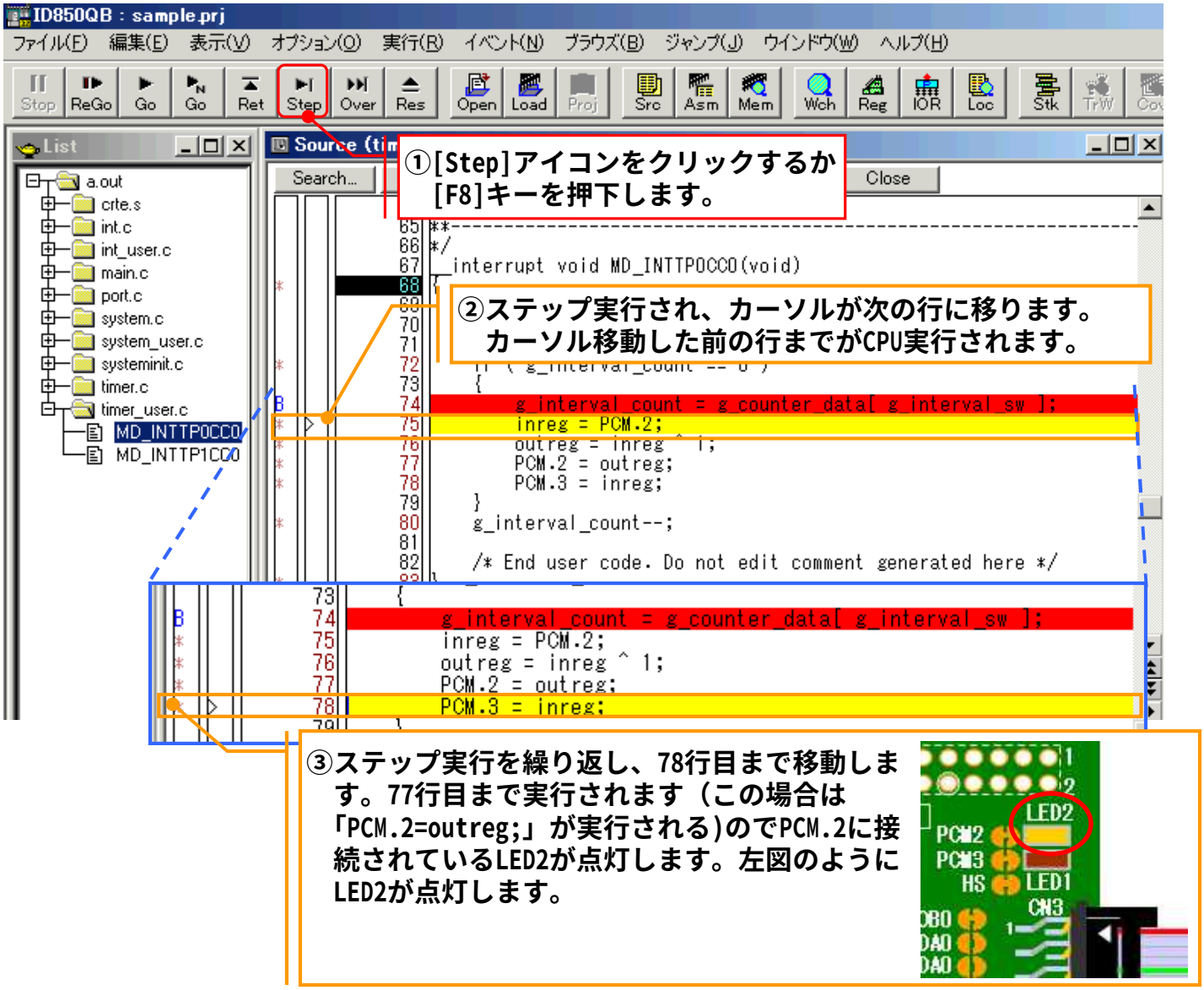
[Go]ボタンを押下して実行し、プログラムが停止することを確認します。



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

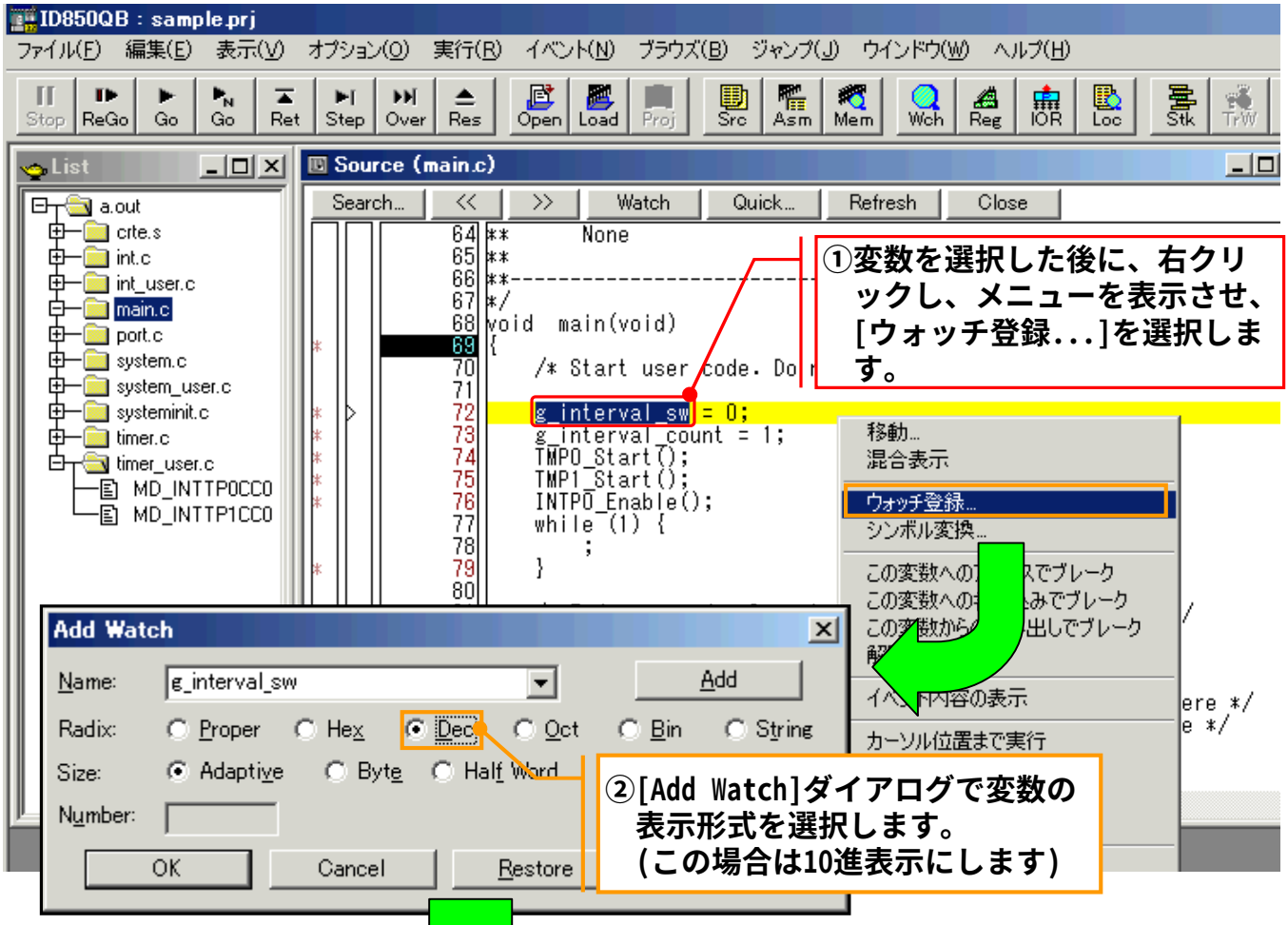
ステップ実行には色々な種類があります。[ステップ･イン(F8)]，[ネクスト･オーバー(F10)]，[リターン･アウト(F7)]など他にも[カーソル位置まで実行]，[カーソル位置から実行]もあります。

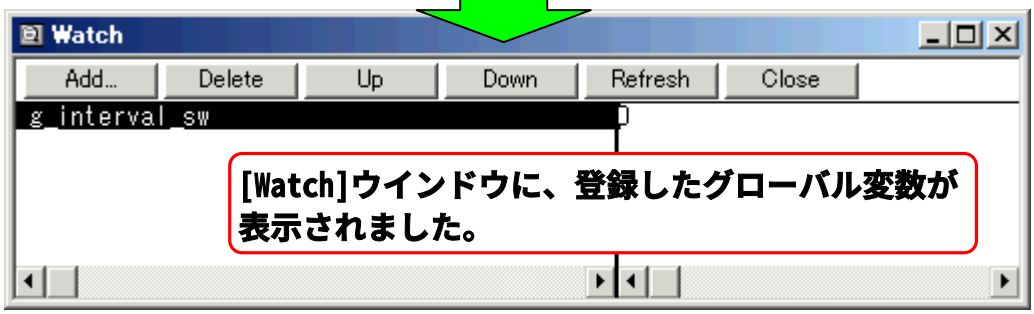
- [ステップ･イン(F8)]
  関数呼び出しがあった場合、その呼び出し関数内もステップ実行します。全てのシーケンスを確認したい時に使用します。
- [ネクスト･オーバー(F10)]
  関数呼び出しがあった場合、その呼び出し関数は一度に処理されます。ブレークしたソース内のみをデバッグしたい場合に便利です。
- [リターン･アウト(F7)]
  ブレークした関数内をリターンするまで一度に処理します。

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。

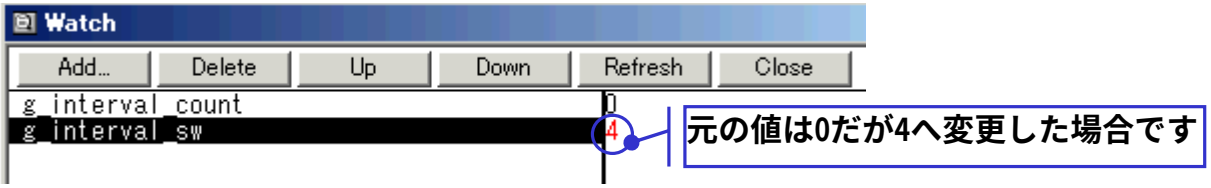




ワンポイント

Watchウインドウに表示している変数の書き換えについて

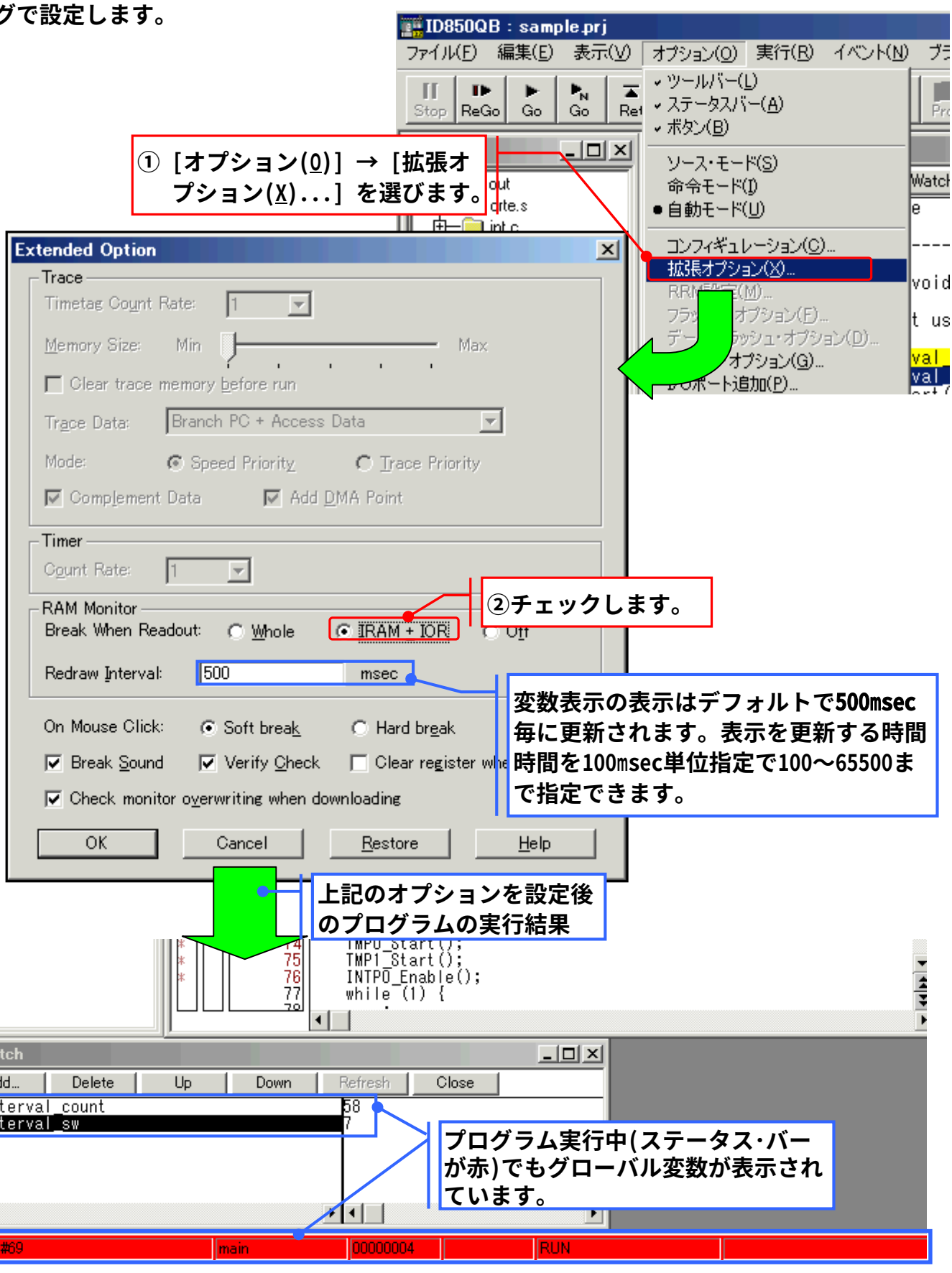
プログラム実行中にメニューの[実行(R)] → [ストップ(S)]またはStopボタン でプログラムを停止させた時Watchウインドウに表示されている変数の値を変更することができます。



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。

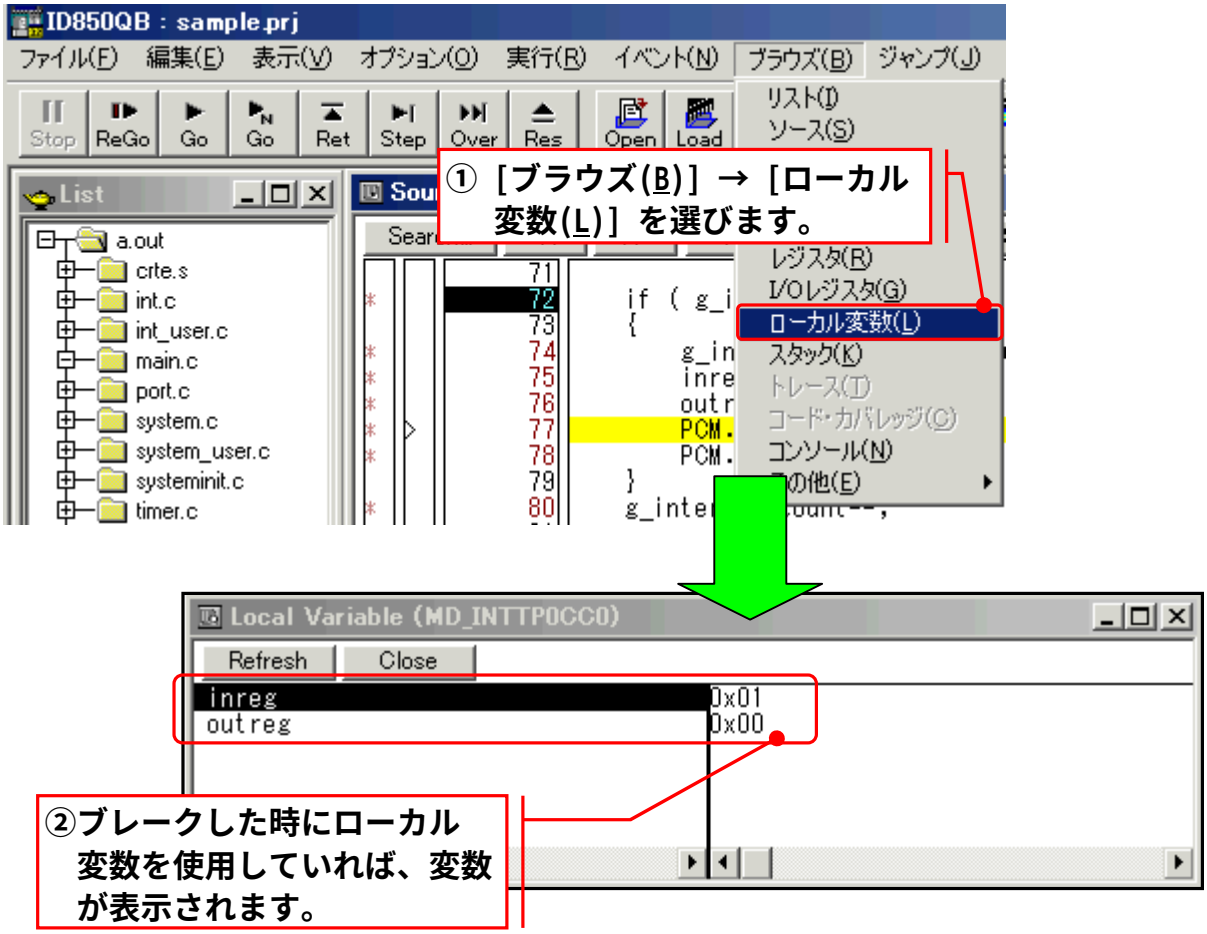
プログラム実行中にグローバル変数を表示したい場合、メニューの[オプション(O)] → [拡張オプション(X)...] を選び、[Extended Option]ダイアログを表示します。 [Extended Option]ダイアログで設定します。



変数表示

i．ローカル変数の表示方法。

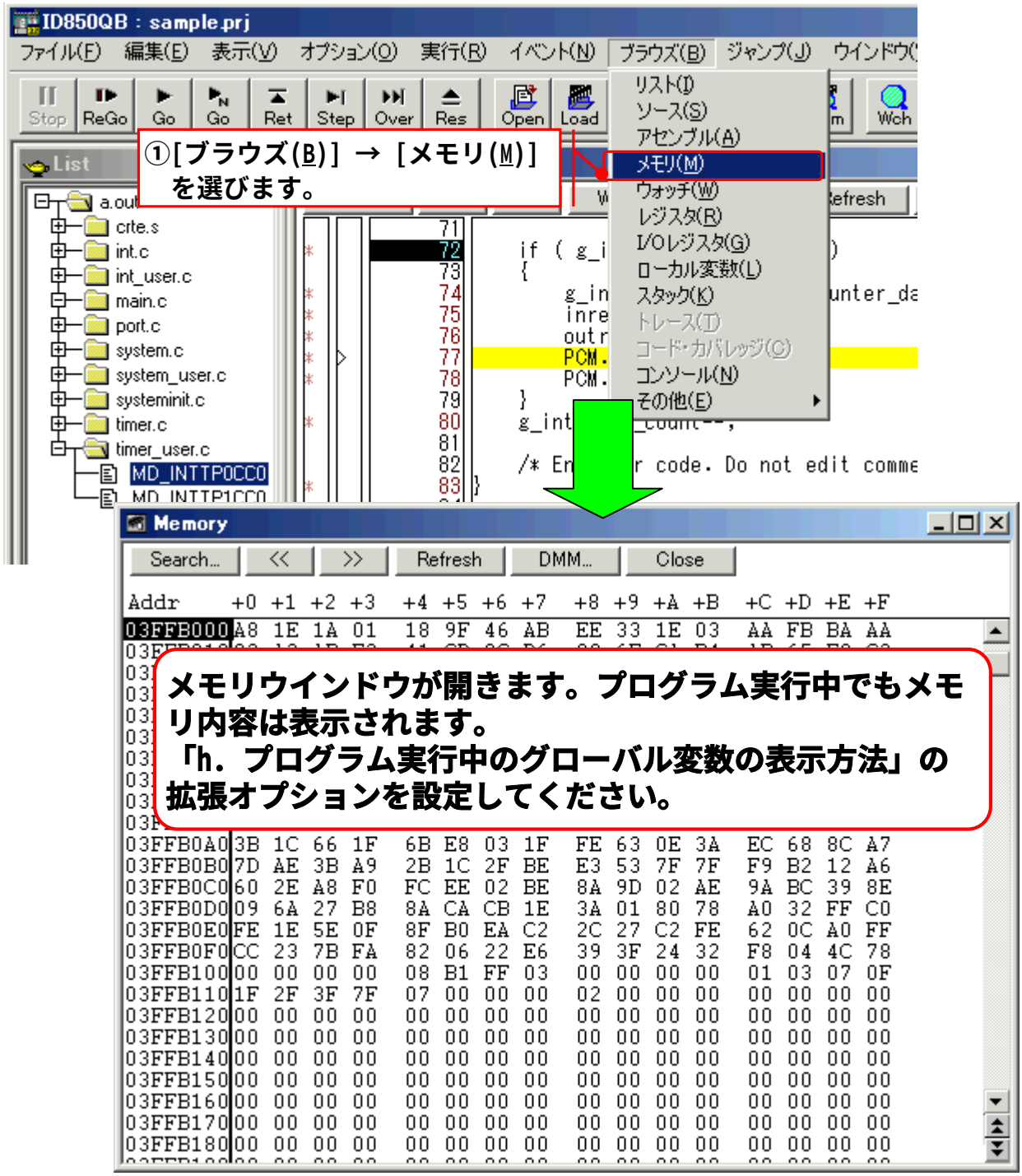
メニューの[ブラウズ(B)] → [ローカル変数(L)] を選ぶと、[Local Variable]ウインドウが表示されます。ローカル変数は実行中には表示されません。ブレークした時のみ表示します。



メモリ表示

j．メモリ表示方法。

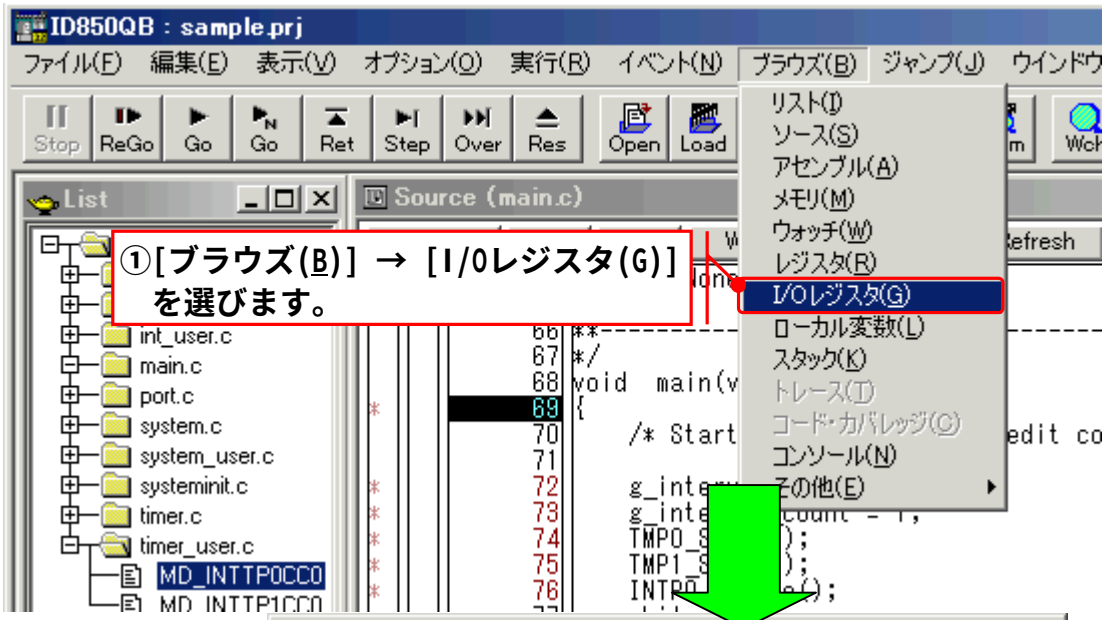
メニューの[ブラウズ(B)] → [メモリ(M)] を選ぶと、[Memory]ウインドウが表示されます。プログラム実行中は通常、メモリ内容を表示できません。しかし、一部であれば[RRM設定...]を行うことにより、プログラム実行中でもメモリ内容の一部を表示できます。

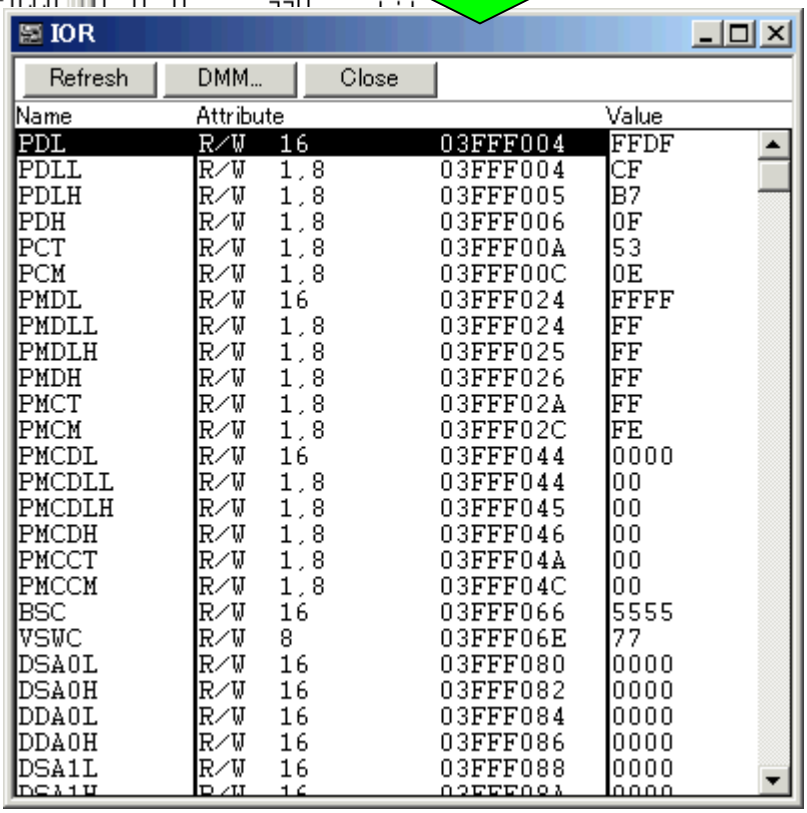


I/Oレジスタ表示

k. I/Oレジスタ表示方法。

メニューの[ブラウズ(B)] → [I/Oレジスタ(G)] を選ぶと、[IOR]ウインドウが表示されます。





Name	Attribute			Value
PDL	R/W	16	03FFF004	FFDF
PDLL	R/W	1,8	03FFF004	CF
PDLH	R/W	1,8	03FFF005	B7
PDH	R/W	1,8	03FFF006	0F
PCT	R/W	1,8	03FFF00A	53
PCM	R/W	1,8	03FFF00C	0E
PMDL	R/W	16	03FFF024	FFFF
PMDLL	R/W	1,8	03FFF024	FF
PMDLH	R/W	1,8	03FFF025	FF
PMDH	R/W	1,8	03FFF026	FF
PMCT	R/W	1,8	03FFF02A	FF
PMCM	R/W	1,8	03FFF02C	FE
PMCDL	R/W	16	03FFF044	0000
PMCDLL	R/W	1,8	03FFF044	00
PMCDLH	R/W	1,8	03FFF045	00
PMCDH	R/W	1,8	03FFF046	00
PMCCT	R/W	1,8	03FFF04A	00
PMCCM	R/W	1,8	03FFF04C	00
BSC	R/W	16	03FFF066	5555
VSWC	R/W	8	03FFF06E	77
DSA0L	R/W	16	03FFF080	0000
DSA0H	R/W	16	03FFF082	0000
DDA0L	R/W	16	03FFF084	0000
DDA0H	R/W	16	03FFF086	0000
DSA1L	R/W	16	03FFF088	0000

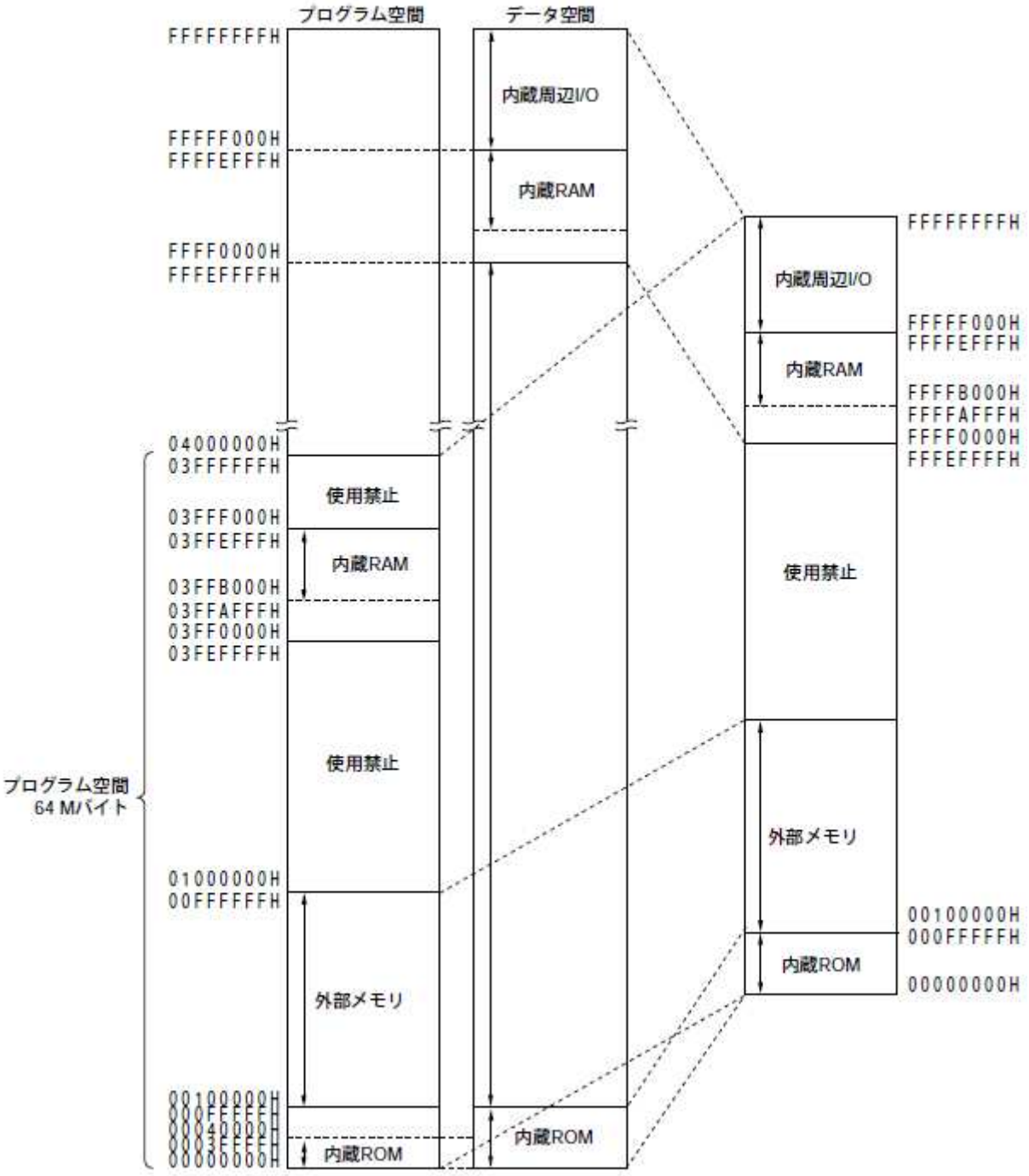
ワンポイント

IORについて

周辺I/O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。
03FFF000h～03FFFFFFhの4096バイトが周辺I/Oレジスタ(IOR)と呼ばれます。このメモリ空間は実際にはFFFFF000h～FFFFFFFFhに配置されていますが、ミラーとして03FFF000h～03FFFFFFhでもアクセス可能です。なお、メモリウインドウではIORの内容は表示されません。

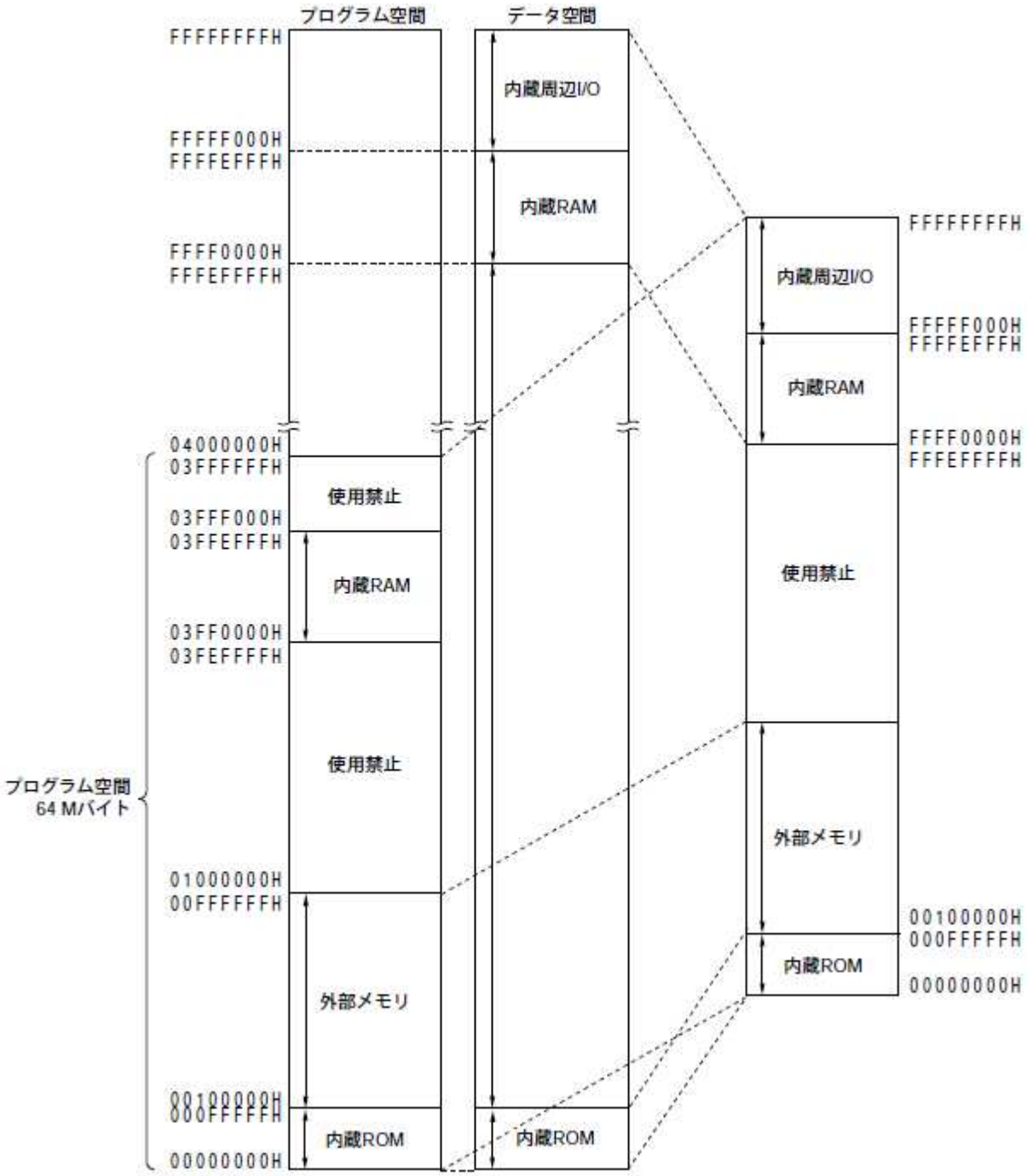
メモリマップ

l. QB-V850ESJG3L-TBに搭載されているマイコン(μPD70F3738)のメモリマップです。



メモリマップ

m. QB-V850ESJJ3-TBに搭載されているマイコン(μPD70F3746)のメモリマップです。



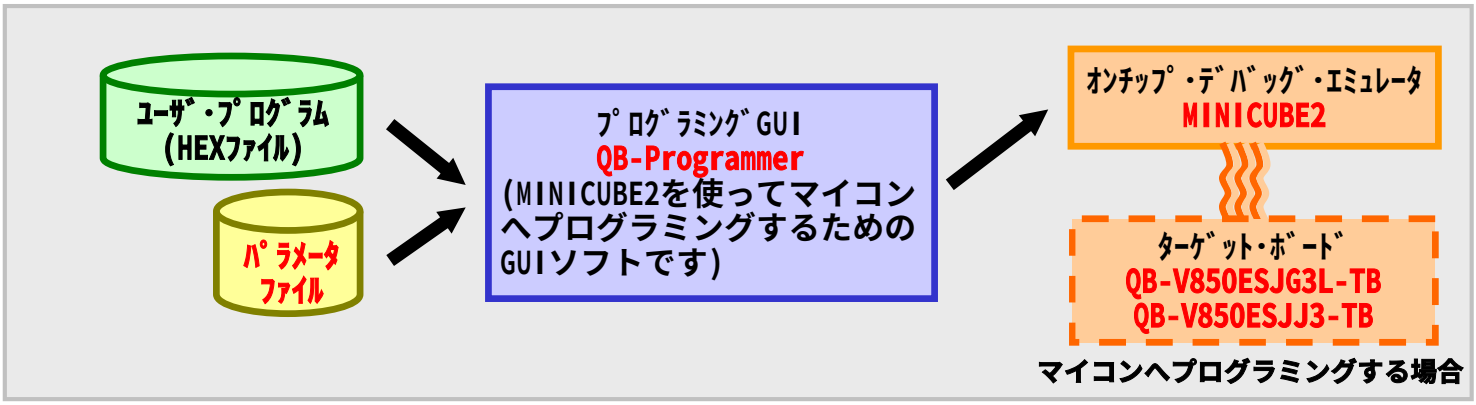
「デバッグしたい」の章は以上です。ここに記したデバッグ手法は基本に過ぎません。プログラムが複雑になるほど多くの問題が発生します。多くの例では、スタックの問題が挙げられます。マイコンはRAMの容量が限られています。配列などを多く使うと知らない内にプログラムが使うスタック領域が足りなくなる問題が発生します。プログラムを作る際はマイコンのメモリマップも意識する必要があります。
次章より「マイコンへ書き込み」を説明します。マイコンへプログラミング(書き込み)する場合は、次章を読んで下さい。

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。

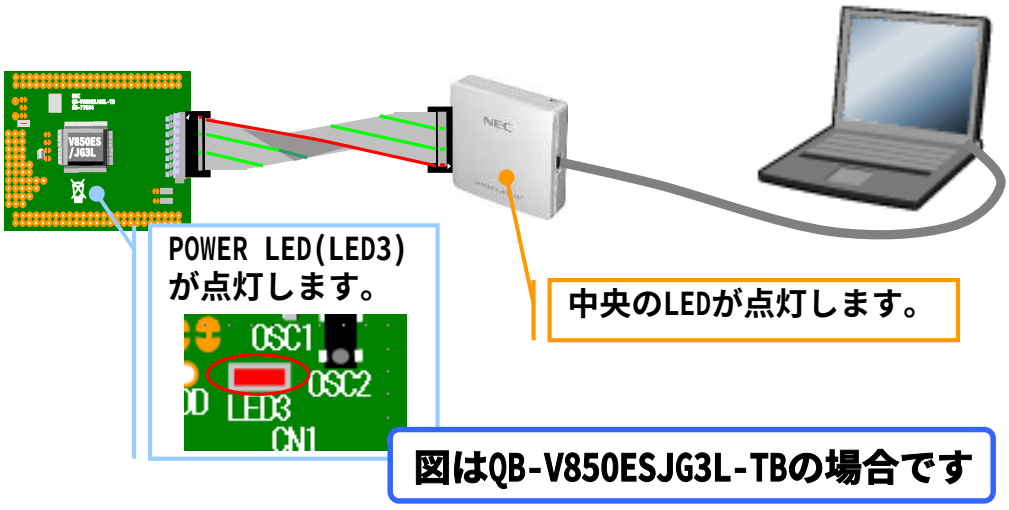
プログラミング方法



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。

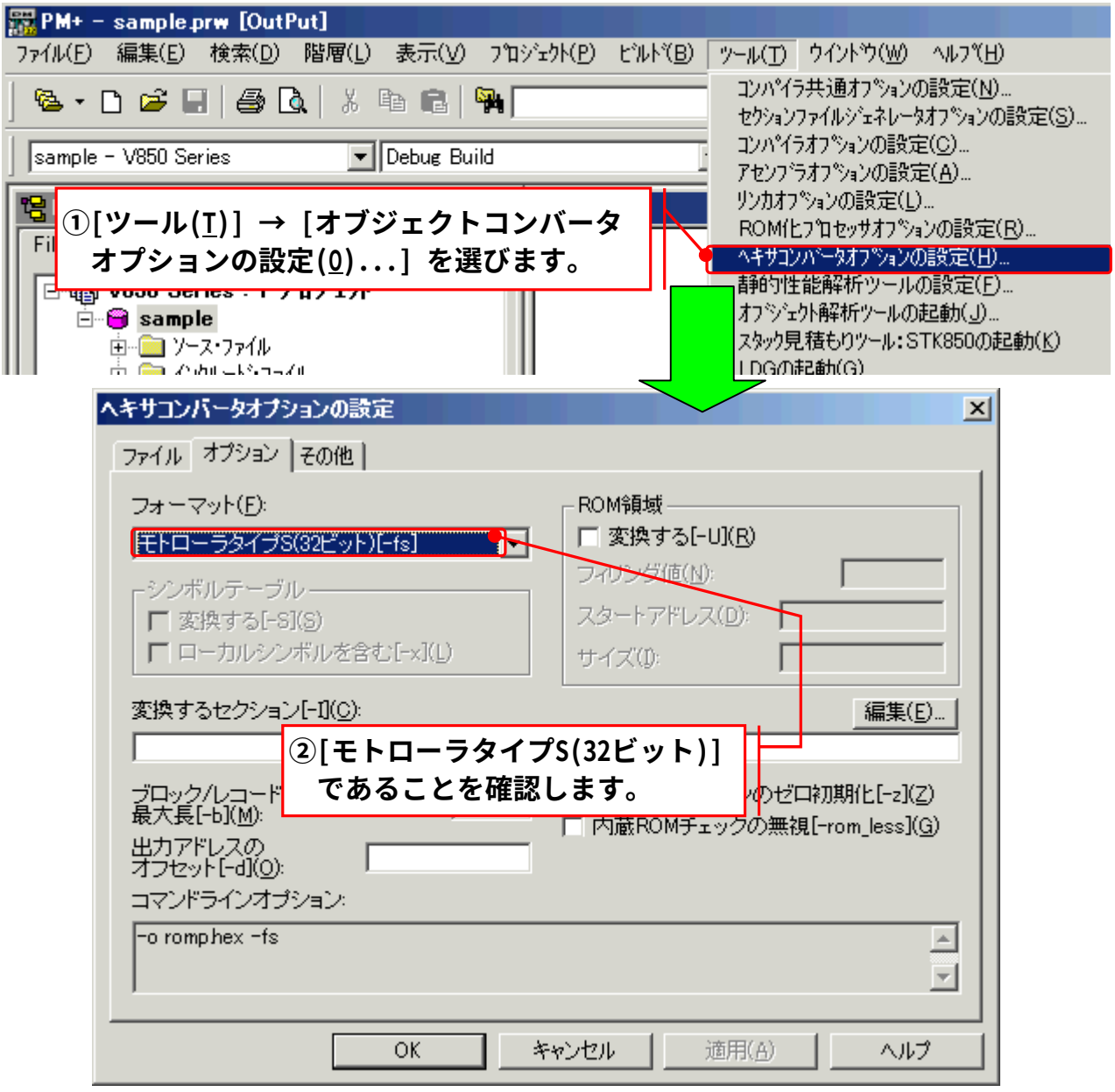


図はQB-V850ESJG3L-TBの場合です

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。[ヘキサコンバータオプションの設定]ウインドウが開きます。

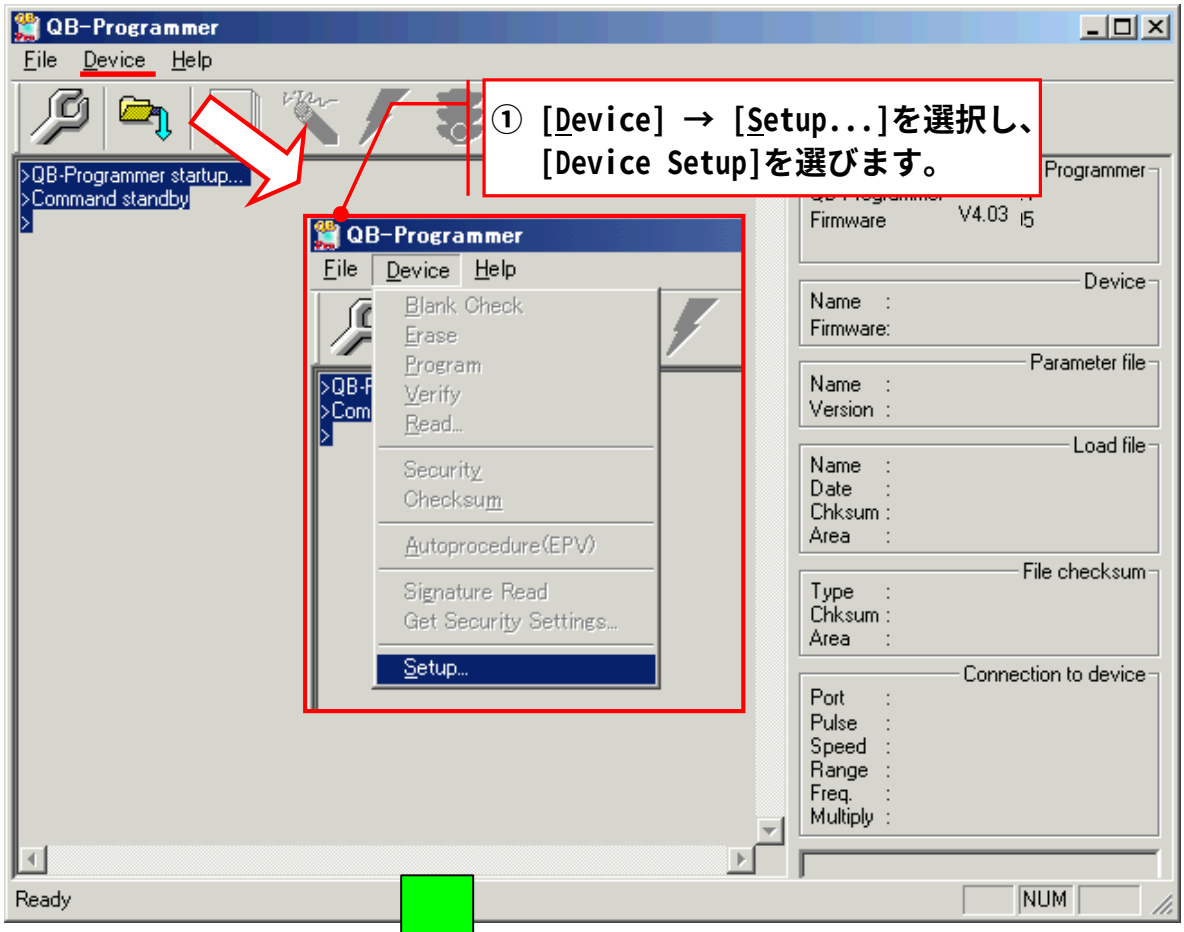


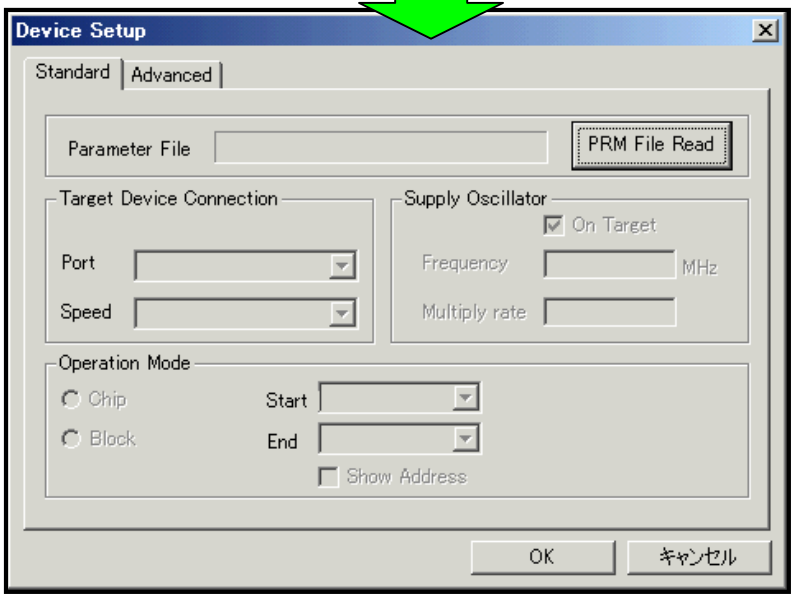
b. ビルドを行い、HEXファイルを作成します。

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。

QBP(QB-Programmer)の起動

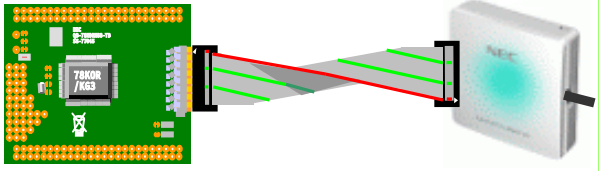
c. マイコンへプログラミングするソフトウエア(QBP)を起動します。
[スタート] → [プログラム(P)] → [NEC Electronics Tools] → [QBP] → [Vx.xx] → [QBP Vx.xx QB-Programmer]でQBPを起動します。起動後に[Device] → [Setup...]を選択し、[Device Setup]ウインドウを開きます。





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。

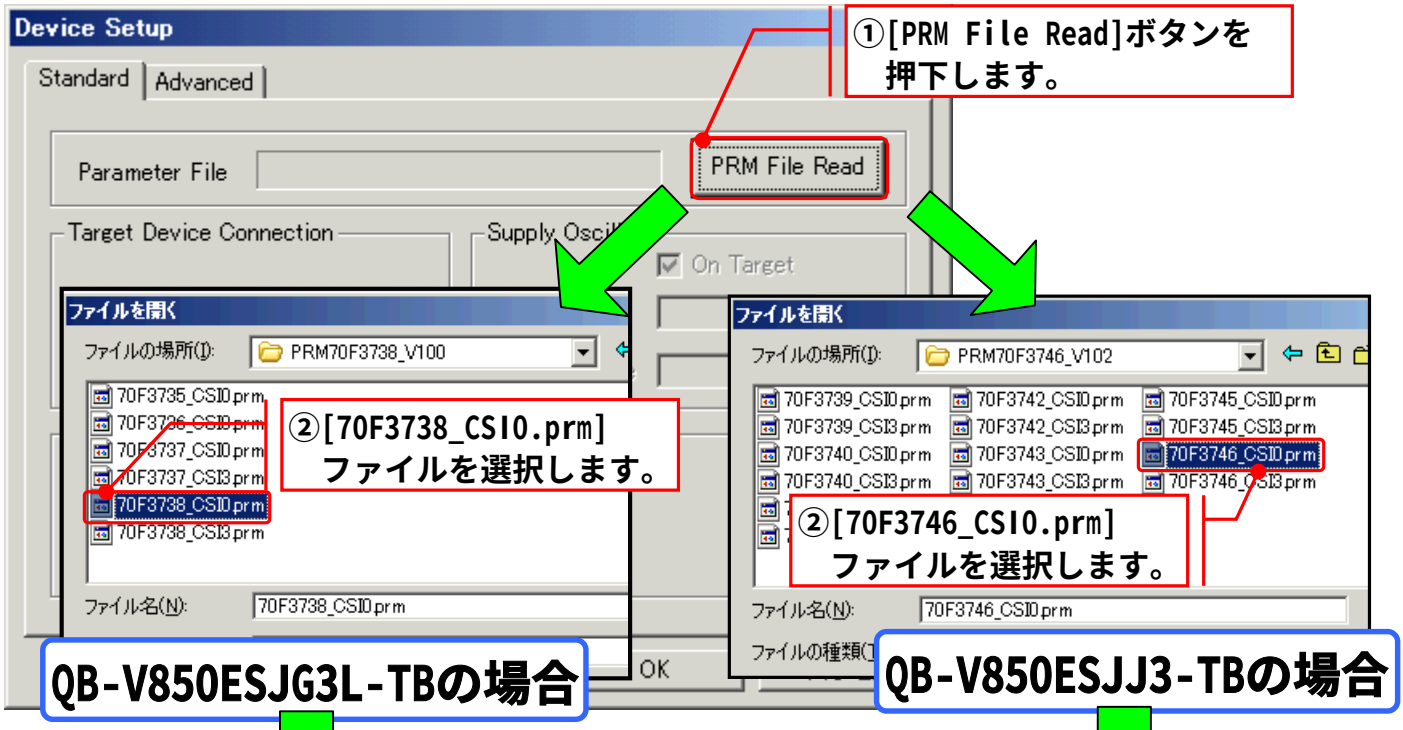


図はQB-V850ESJG3L-TBの場合です

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。

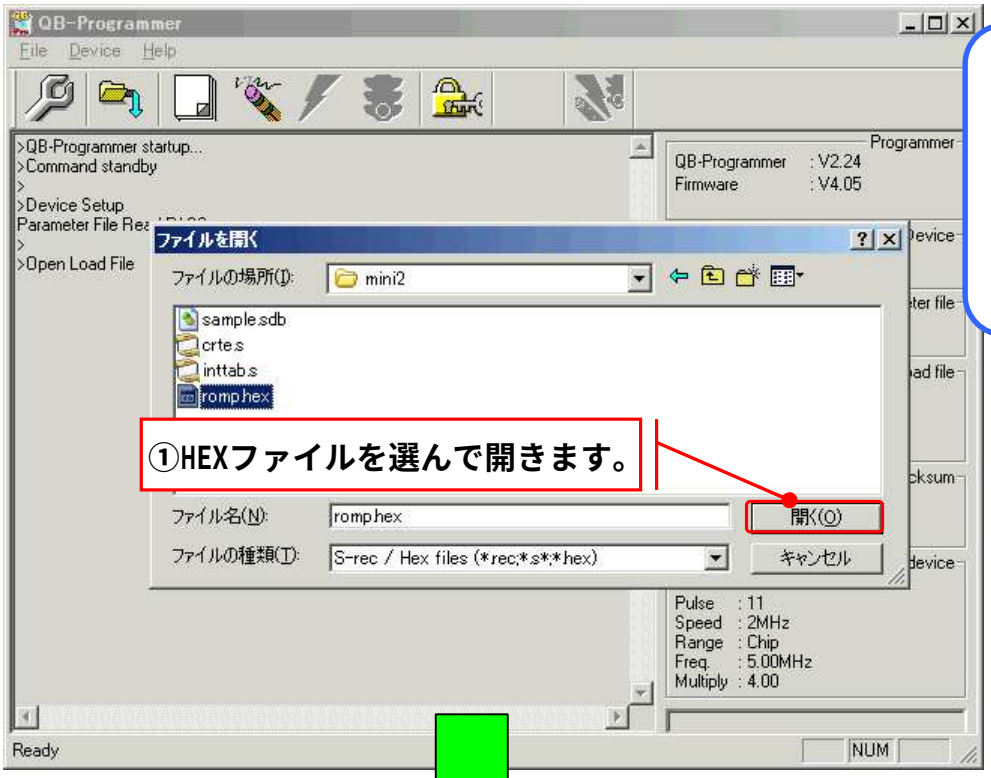
[Device Setup]ウインドウより[PRM File Read]ボタンを押下してパラメータファイルを読んで下さい。ターゲットボードに応じてパラメータファイルが異なりますので注意してください。

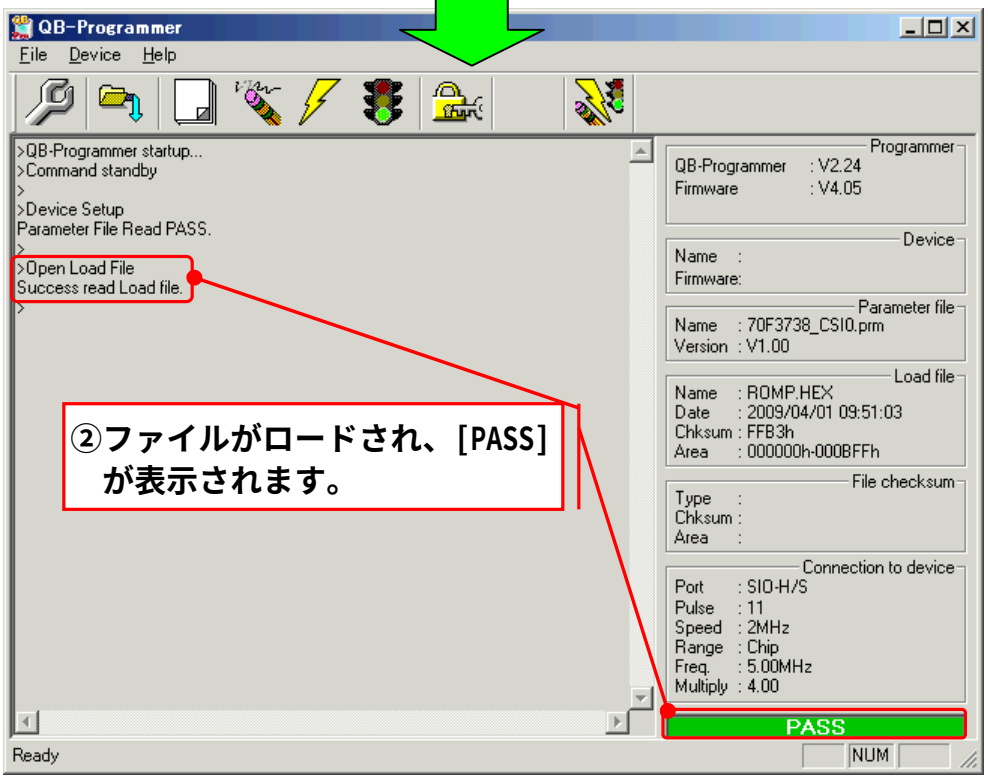


[Device Setup]ウインドウの内容が更新されます。

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。





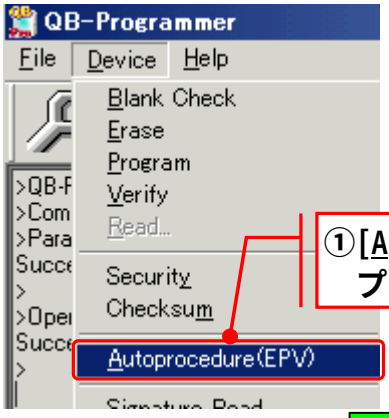
QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
それぞれ、生成したHEXファイルを選択してください。

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。

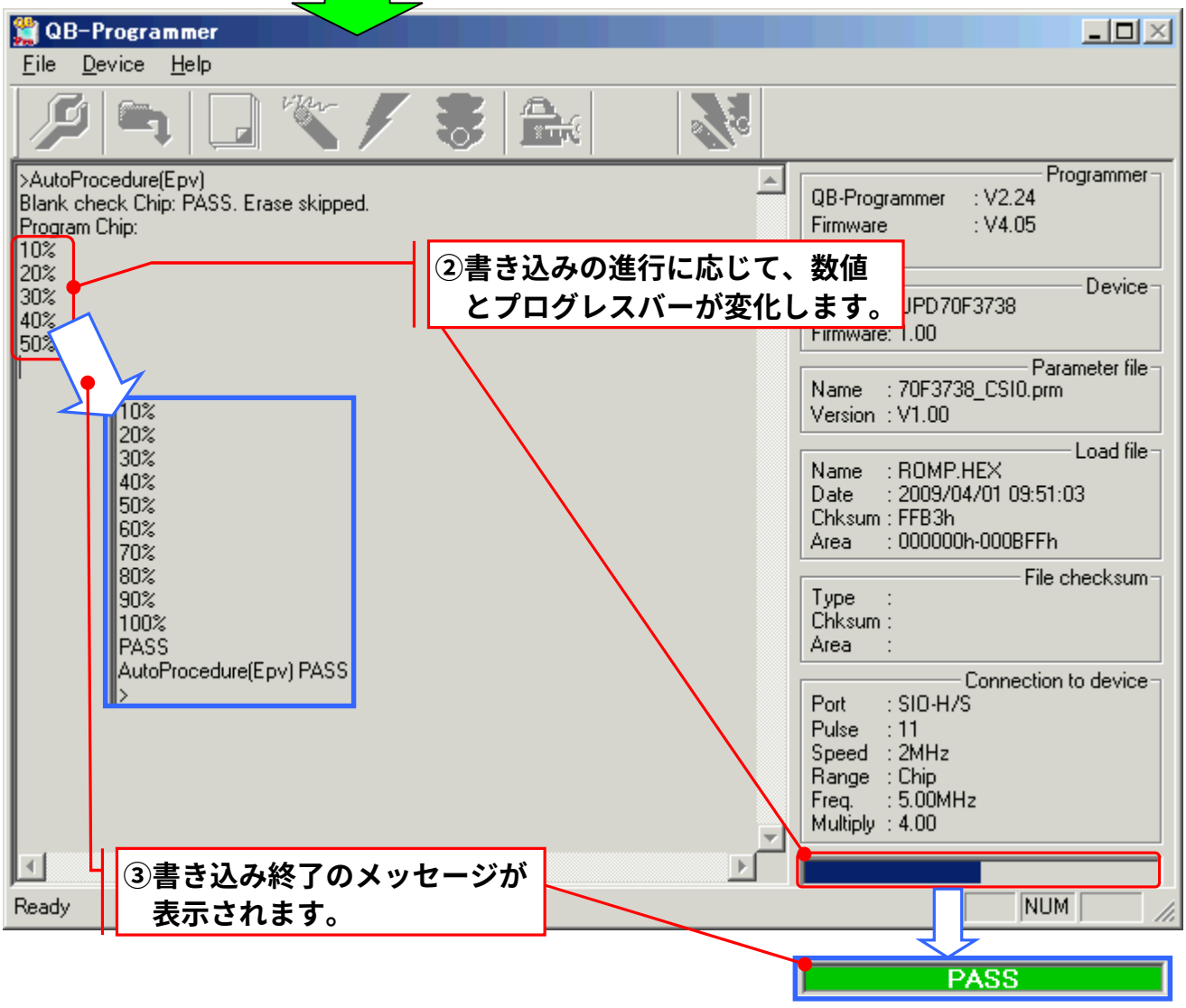
マイコンへプログラミング

f. 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。



①[Autoprocedure(EPV)] を選ぶとプログラミングが開始されます。

QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
操作は共通です。

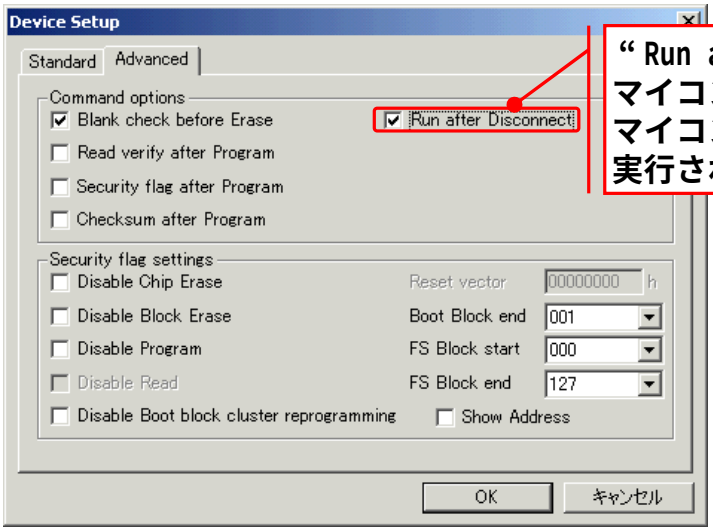


②書き込みの進行に応じて、数値とプログレスバーが変化します。

③書き込み終了のメッセージが表示されます。

動作確認

g. 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。

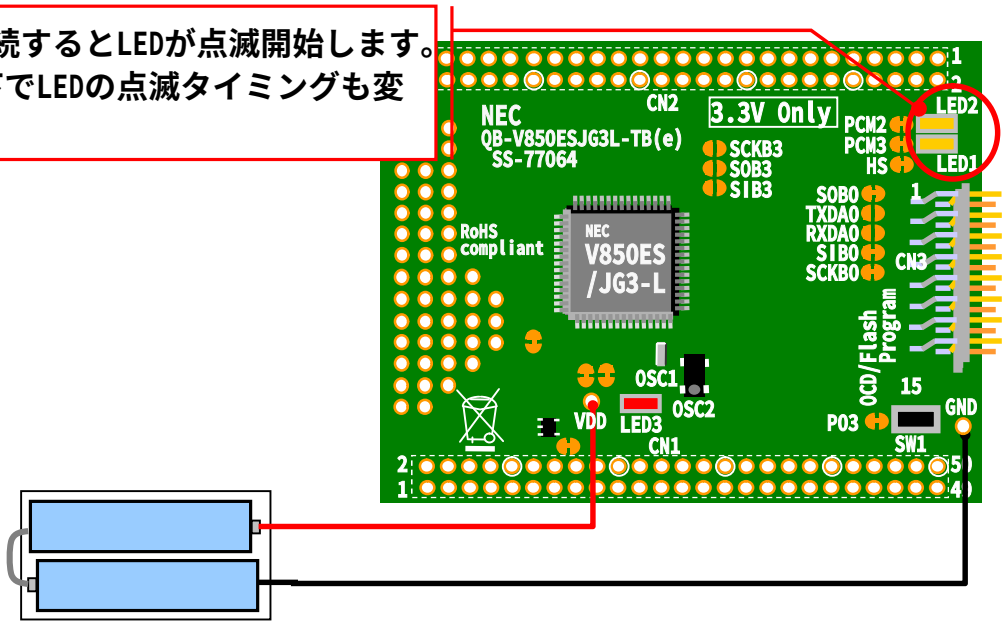


“Run after Disconnect” へチェックすると、「e. マイコンへプログラミングします。」の処理を行い、マイコンの書き込みが終了した後に、プログラムが実行されます。

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ケーブルも外してターゲットだけにします。
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。



上記はQB-V850ESJG3L-TBの場合ですがQB-V850ESJJ3-TBでもVDD,GNDに電源を供給すれば動作します。

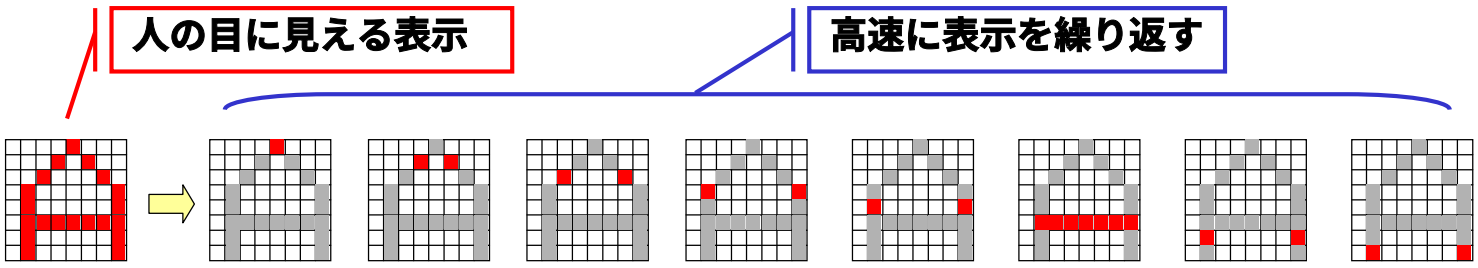
「マイコンへ書き込み」の章は以上です。現在までの章「動かしてみよう」、「デバッグしたい」とあわせて、マイコンの開発一通りを学びました。次章ではターゲットボードを応用した電光掲示板を作成します。

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。

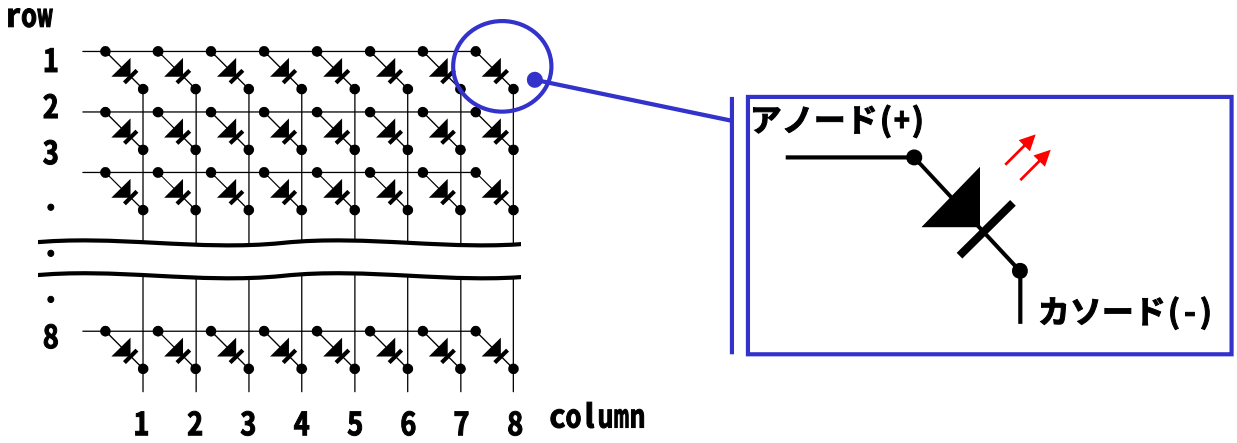
ターゲット･ボードを使用した回路を作成します。ここでは8x8のドットマトリクスLEDをダイナミック点灯させます。ダイナミック点灯とは表示を分割して(今回は8個のLED)行う方法です。表示を高速に順次繰り返す事によって人の目には全てが表示されているように見えます。
ダイナミック点灯には下記の特徴があります。
利点:常時点灯していないので消費電力が少ない。ポート制御が少なく済む。
欠点:スタティック(常時)点灯に比べて表示が暗い。

ダイナミック点灯の原理



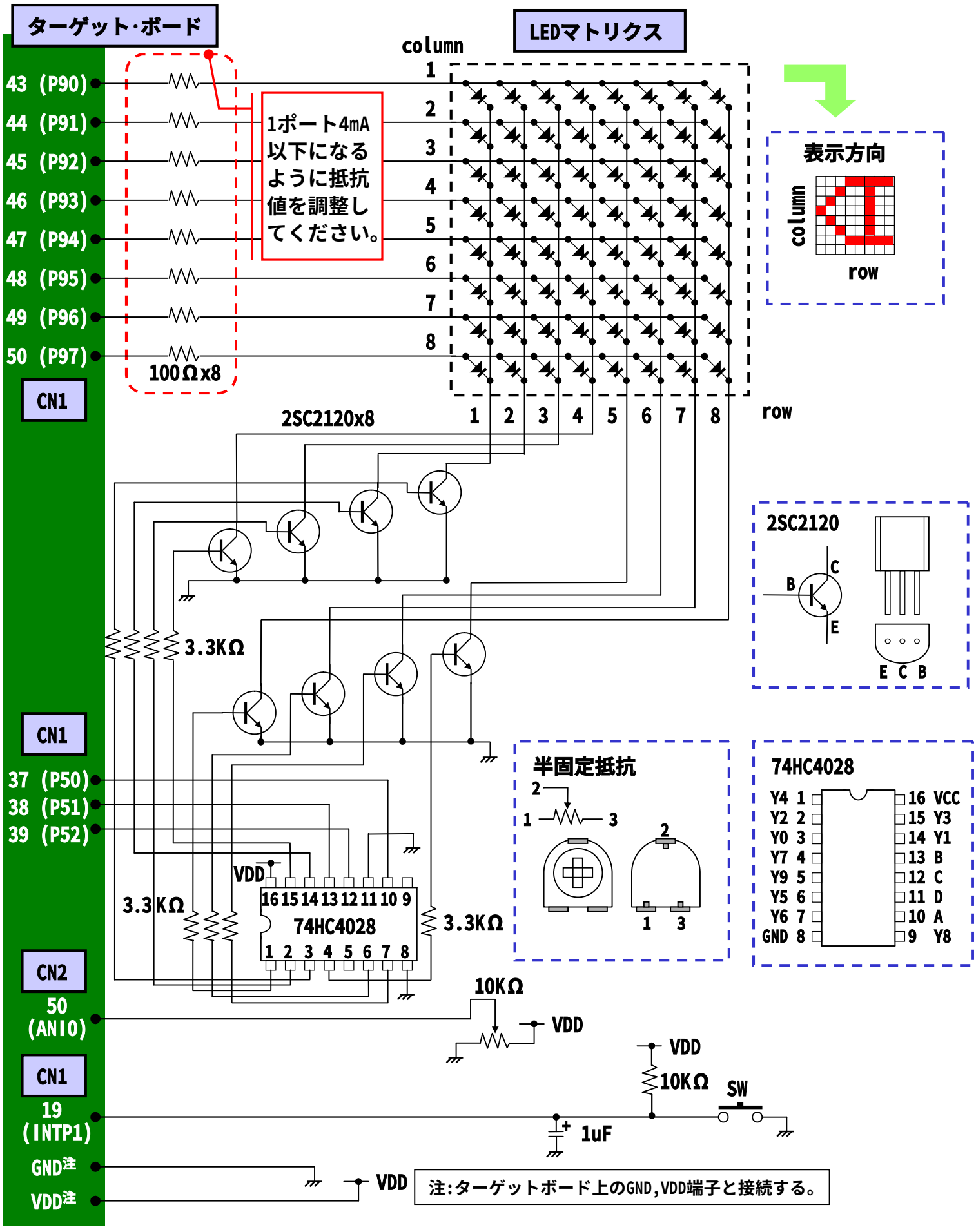
ワンポイント

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。

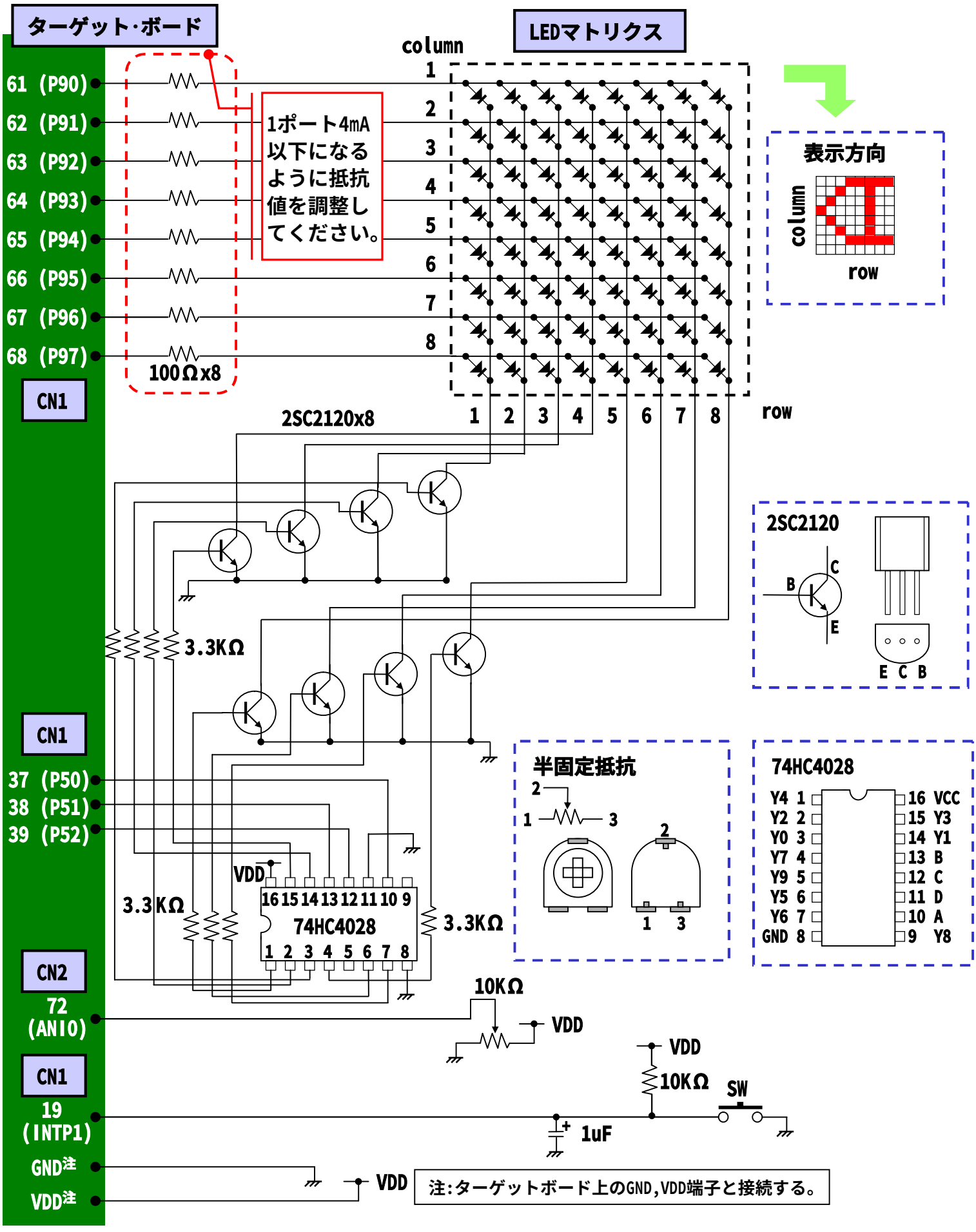


例えば、上記の青丸内のLEDを点灯させるにはrow1に+(プラス)、column8に-(GND)を接続すれば点灯します。LEDを1つ点灯させるには3mA～20mA必要です(LEDの色、LEDの特性によって異なります)。マトリクスLEDは1列に8個接続されていますので24mA～160mAが必要です。マイコンポートの最大電流は1ポート4mA、いくつかの端子合計で50mAですので、LEDを接続する際は注意が必要です。1ポート4mAで光るように次ページのcolumnに接続する抵抗を調整してください。

回路図(QB-V850ESJG3L-TBとの接続)



回路図(QB-V850ESJJ3-TBとの接続)



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。
電光掲示板の流れる速度を調整できるようにA/Dコンバータ(ANI0)を使用します。半固定抵抗の抵抗値によって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。(プルアップ抵抗と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

V850ES/JG3LとV850ES/JJ3のポートはソース電流で4mA(端子合計50mA)、シンク電流で4mA(端子合計50mA)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})

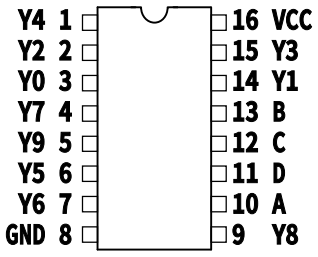
ソース電流(I_{OH})

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



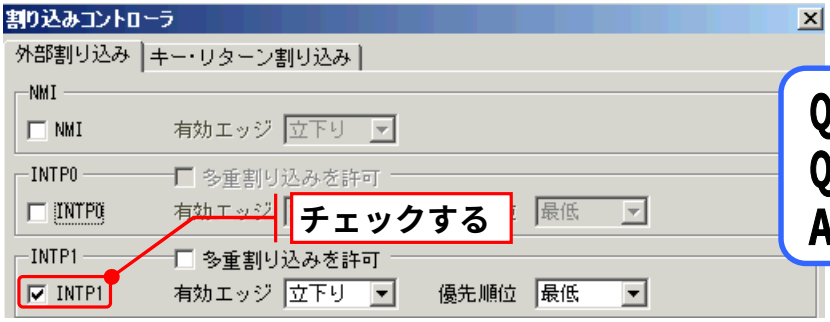
プログラム設計

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。以降の設定はQB-V850ESJG3L-TB、QB-V850ESJJ3-TB共に同じです。

b. [割り込み]を設定します。

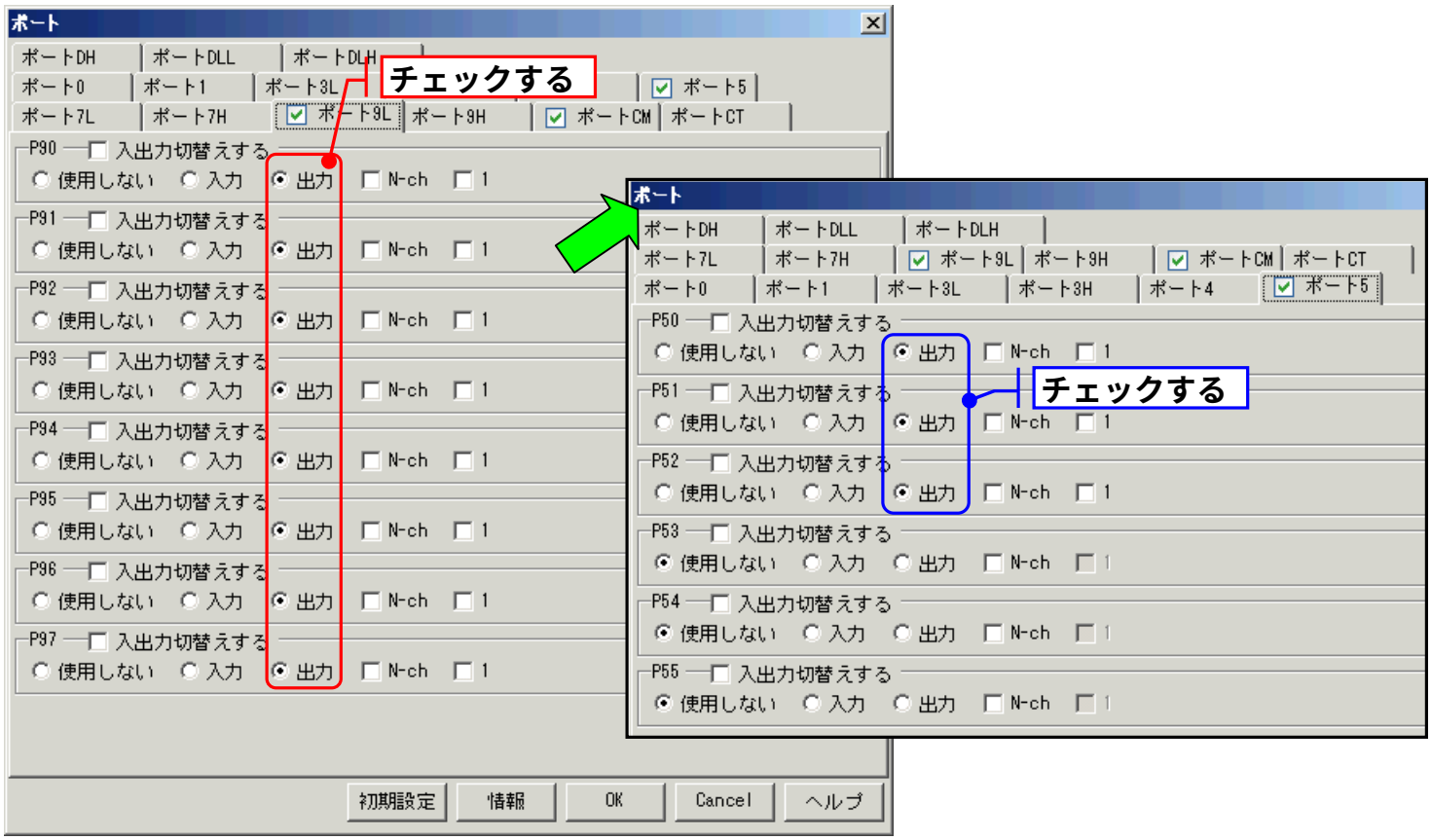
[割り込み]ダイアログは、外部割り込みを設定します。



QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
Appliletの設定は共通です。

c. [ポート]を設定します。

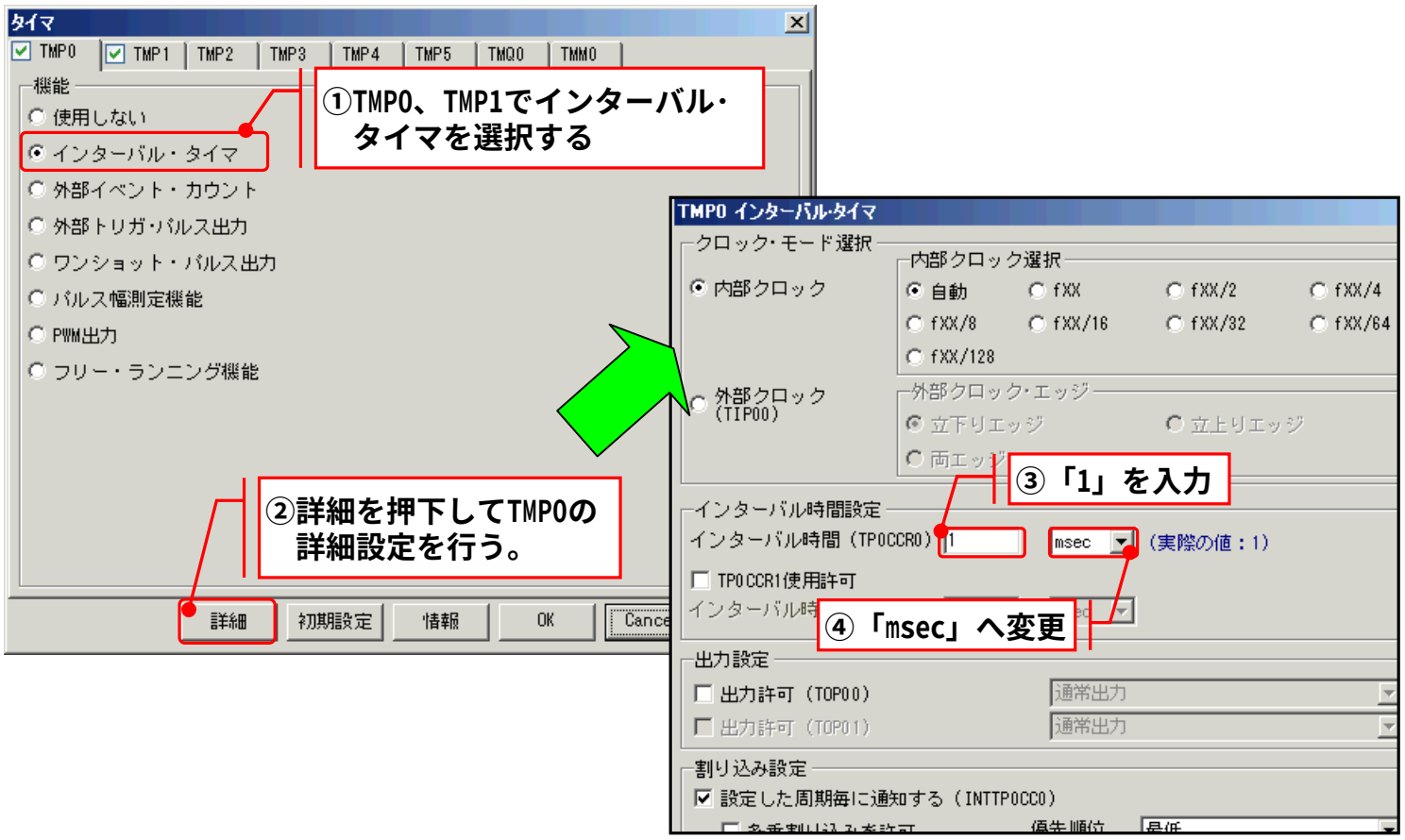
ポート9L[P90～P97]の出力にチェック、ポート5[P50～P52]の出力にチェックにチェックします。



プログラム設計

d. [タイマ]を設定します。

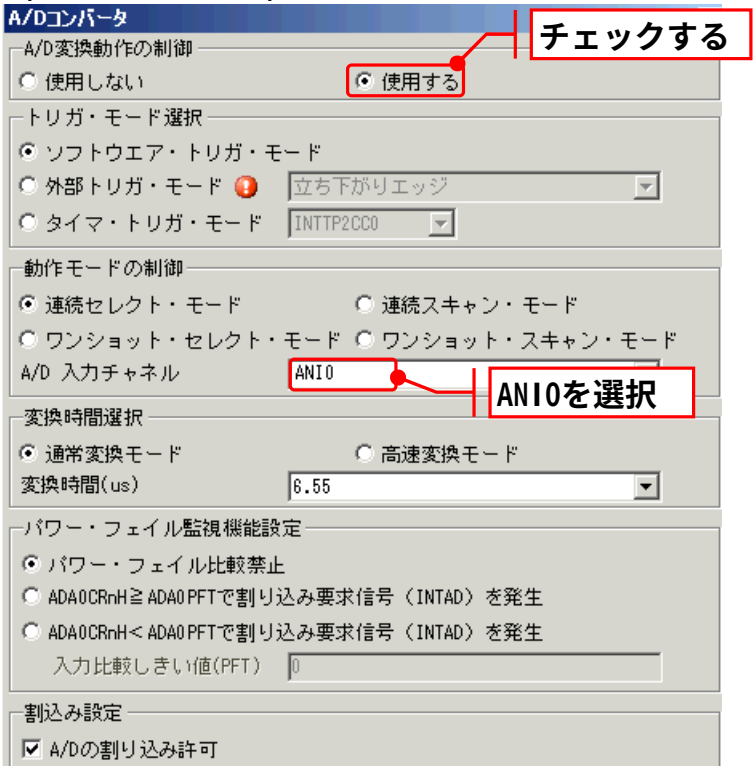
TMP0、TMP1にインターバル･タイマ1msを設定します。



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、A/D入力チャネルANI0を選択します。その他はデフォルト設定のままでOKです。



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。

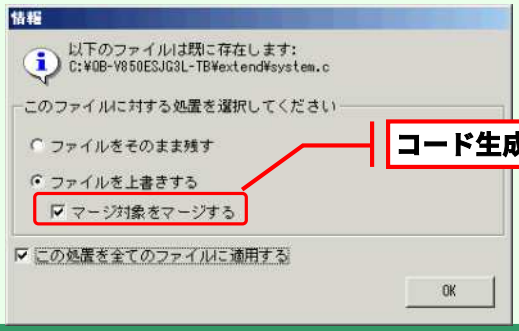
- user_define.h　ユーザが定義する定数などを書く
- main.c　ユーザのメイン関数を含む
- int_user.c　ユーザが書く割り込み処理
- timer_user.c　ユーザが書くタイマ割り込み処理
- ad_user.c　ユーザが書くA/D変換終了の割り込み処理

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。



コード生成時に、チェックして下さい。

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

QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
プログラムは共通です。

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 */

/* メインで処理を行うイベントコード */
#define D_EV_INTP1          0x0000002   /* INTP1押下                          */
#define D_EV_DYNAMIC        0x0000008   /* マトリックスLEDをダイナミック点灯させる */
#define D_EV_SCROLL         0x0000010   /* マトリックスLEDを1ドットスクロールする  */

/* マトリックスLED(表示関係)の定義 */
#define D_MLED_CNT          2           /* マトリックスLEDの数(仮想数)          */
#define D_MLED_ROW          8           /* マトリックスLEDの row (縦列数)       */
#define D_FONT_WIDTH        6           /* 表示するフォント幅                   */
#define D_MAXTEXT          34           /* スクロールして表示させる最大文字数    */

/* 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 */
#include "string.h"

/* in This file. */
void initialize_hard( void );
void initialize_value( void );
void initialize_text( void );
void disp_putfont( UCHAR font_ );
void disp_move1dot( void );
void disp_scroll( void );
void disp_dynamic( void );
void switch_textstring( void );
void eventcheck( 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 gtFontNumeric[] =
{ /* 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

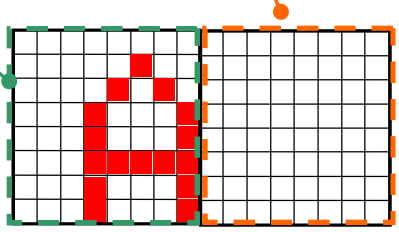
main.c(リスト2)

```
const UCHAR gtFontAlphabet[] =
{ /* 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 gtTextData[][ D_MAXTEXT ] =
{   /* text data */
  "THIS IS V850ESJx3 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
};

/* - - - - - - - - このファイルでの定義 - - - - - - - - - - */
ULONG   gEventflag;                     /* メインループでチェックするイベント。D_EV_xxxxで定義  */
                                        /* bit1 : INTP1押下                                    */
                                        /* bit3 : マトリックスLEDをダイナミック点灯させる      */
                                        /* bit4 : マトリックスLEDを1ドットスクロールする       */
                                        /* bit5 - bit31: reserve(予約)                         */

UINT    g1msecCounterA;                 /* Interval timer 1msec(tmp0使用)                      */
UINT    g1msecCounterB;                 /* Interval timer 1msec(tmp1使用)                      */
UCHAR   gTextSwitch;                    /* 表示する文字列を選択する変数                        */
UCHAR   gTextStringCount;               /* 表示する文字列の文字単位のカウンタ                  */
UCHAR   gTextScrollCount;               /* 表示する文字列のドット単位のカウンタ                */
USHORT  gTextScrollSpeed;               /* 表示する文字列のスピード                            */

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

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

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

    /* 変数初期化 */
    initialize_value();
```

g_matrix_ram[1][0～7]（マトリックスLEDを2個目とする仮想エリア）
スクロールは仮想エリアも含めて行う。1文字のスクロールが終了したら、仮想エリアに1文字表示を行う。

gtMatrixVram[1][0～7]（マトリックスLEDを2個目とする仮想エリア）
スクロールは仮想エリアも含めて行う。

gtMatrixVram[0][0～7]（実際に表示されるエリア）



プログラムリスト

main.c(リスト3)

```
    /* ハードウェア初期化 */
    initialize_hard();

    /* メインループ */
    while ( 1 )
    {
        eventcheck();
    }
    /* End user code. Do not edit comment generated here */
}

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

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* ハードウェア初期化                                                      */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void initialize_hard( void )
{
    /* Target-Board LED off */
    PCM.2 = 1;
    PCM.3 = 1;

    INTP1_Enable();

    /* start timer */
    TMP0_Start();
    TMP1_Start();
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 変数初期化                                                              */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void initialize_value( void )
{

    g1msecCounterA = 0;
    g1msecCounterB = 0;
    gTextSwitch = 0;
    gTextStringCount = 0;
    gTextScrollCount = 0;
    gTextScrollSpeed = 0;

    memset( gtMatrixVram, 0x00, sizeof( gtMatrixVram ));
    gTextSwitch = 0;

    initialize_text();
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* テキストを初期化し、最初の文字をLEDバッファへ転送する                  */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void initialize_text( void )
{
    UCHAR font;

    memset( gtMatrixVram, 0x00, sizeof( gtMatrixVram ));
    gTextScrollCount = 0;
    gTextStringCount = 0;

    font = gtTextData[ gTextSwitch ][ 0 ];
    disp_putfont( font );
}
```

プログラムリスト

main.c(リスト4)

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* テキストを初期化し、最初の文字をLEDバッファへ転送する                        */
/*                                                                             */
/* input:  font_            表示するキャラクタ文字                             */
/* output: gtMatrixVram     仮想VRAMへキャラクタデータを転送                   */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
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 = gtFontAlphabet[ cnvf*8 + i ];
            gtMatrixVram[ 1 ][ i ] = (fnt << 2);
        }
    }
    else if ( cnvf>='0' && cnvf<='9' )
    {   /* Numeric font */
        cnvf -='0';
        for ( i=0; i<D_MLED_ROW; i++ )
        {
            fnt = gtFontNumeric[ cnvf*8 + i ];
            gtMatrixVram[ 1 ][ i ] = (fnt << 2);
        }
    }
    else
    {   /* Null font */
        for ( i=0; i<D_MLED_ROW; i++ )
        {
            gtMatrixVram[ 1 ][ i ] = 0;
        }
    }
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 仮想LEDバッファを1ドットスクロールする                                       */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void disp_move1dot( void )
{
    int i;
    UCHAR vtmp, vtmp2;

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

プログラムリスト

main.c(リスト5)

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 必要であれば仮想LEDバッファへ文字を表示し、                                   */
/* 仮想LEDバッファを1ドットスクロールする                                       */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void disp_scroll( void )
{
    UCHAR font;

    if ( ( gTextScrollCount % D_FONT_WIDTH ) == 0 )
    {   /* next font */
        font = gtTextData[ gTextSwitch ][ gTextStringCount ];
        disp_putfont( font );

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

    /* scroll */
    disp_move1dot();

    gTextScrollCount++;
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* LEDマトリックスをダイナミック点灯する                                        */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void disp_dynamic( void )
{
    UCHAR row, line;

    /* Matrix LED line disp */
    row = (UCHAR)(g1msecCounterA) & 7;

    /* display column */
    P9L = 0;
    line = gtMatrixVram[ 0 ][ row ];
    P9L = line;

    /* display row */
    P5 = row;
}


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 表示する文字列を切り替える                                                   */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void switch_textstring( void )
{
    gTextSwitch++;
    if (  gtTextData[ gTextSwitch ][ 0 ] == 0 )
    {
        gTextSwitch = 0;
    }
    /* テキストを初期化し、最初の文字をLEDバッファへ転送する */
    initialize_text();
}
```

プログラムリスト

main.c(リスト6)

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* D_EV_xxxxで定義したイベント処理を行う                                      */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void eventcheck()
{

    /* INTP1チェック */
    if ( gEventflag & D_EV_INTP1 )
    {
        /* 表示する文字列を変更する */
        switch_textstring();

        /* イベントクリア */
        gEventflag &= ~D_EV_INTP1;
    }

    /* ダイナミック点灯処理チェック */
    if ( gEventflag & D_EV_DYNAMIC )
    {
        /* ダイナミック点灯処理 */
        disp_dynamic();

        /* イベントクリア */
        gEventflag &= ~D_EV_DYNAMIC;
    }

    /* マトリックスLEDスクロール処理チェック */
    if ( gEventflag & D_EV_SCROLL )
    {
        /* マトリックスLEDスクロール処理 */
        disp_scroll();

        /* イベントクリア */
        gEventflag &= ~D_EV_SCROLL;
    }

}

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

int_user.c

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

extern ULONG gEventflag;

/* End user code for global definition. Do not edit comment generated here */
            リスト省略
/*
**-----------------------------------------------------------------------
** 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 */

    gEventflag |= D_EV_INTP1;

    /* 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

timer_user.c

リスト省略

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

#include "ad.h"

/* End user code for include definition. Do not edit comment generated here */
#include "user_define.h"

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

extern ULONG  gEventflag;
extern UINT   g1msecCounterA;
extern UINT   g1msecCounterB;
extern USHORT gTextScrollSpeed;

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

/*
**-----------------------------------------------------------------------------
** Abstract:            This function is INTTP0CC0 interrupt service routine.
** Parameters:          None
** Returns:             None
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTTP0CC0(void)
{
    /* Start user code. Do not edit comment generated here */

    gEventflag |= D_EV_DYNAMIC;
    g1msecCounterA++;

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

/*
**-----------------------------------------------------------------------------
** Abstract:            This function is INTTP1CC0 interrupt service routine.
** Parameters:          None
** Returns:             None
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTTP1CC0(void)
{
    /* Start user code. Do not edit comment generated here */

    AD_Start();

    if ( g1msecCounterB > gTextScrollSpeed )
    {
        gEventflag |= D_EV_SCROLL;
        g1msecCounterB = 0;
    }
    g1msecCounterB++;

    /* 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 "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 */
#include "user_define.h"

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

extern USHORT gTextScrollSpeed;

/* 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 );
    gTextScrollSpeed = adval/2 + 1;
    if ( gTextScrollSpeed > 500 )
    {
        gTextScrollSpeed = 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変数と関数について説明します。

main.c

○変数について

```
const UCHAR gtFontNumeric[]; /* 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 gtFontAlphabet[]; /* 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 gtTextData[][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

```
ULONG gEventflag;
```
メインループでチェックするイベント変数です

```
UINT g1msecCounterA;
UINT g1msecCounterB;
```
1msecのタイマ割り込み時に+1されるカウンタです

```
USHORT gTextSwitch;
```
表示する文字列(テキスト)を選択します

```
UCHAR gTextStringCount;
```
表示されている文字列が何文字目かを示します

```
UINT  gTextScrollCount;
```
表示されている文字列が何ドット移動したかを示します

```
USHORT gTextScrollSpeed;
```
表示する文字列のスクロールするスピードを示します

```
UCHAR gtMatrixVram[ 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

○関数について

```
void initialize_hard( void );
void initialize_value( void );
void initialize_text( void );
```
初期化用の関数です

```
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 );
void disp_scroll( 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

```
void disp_dynamic( void );
```
マトリックスLEDのダイナミック点灯表示します

```
void switch_textstring( 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

```
void eventcheck( void );
```
イベント変数をチェックし、発生したイベントを処理します

timer_user.c

```
__interrupt void MD_INTTP0CC0()
```
1msec毎に呼ばれる割込ハンドラ関数です。
- ダイナミック点灯用のイベントを発生します
- g1msecCounterA変数を+1します

```
__interrupt void MD_INTTP1CC0()
```
1msec毎に呼ばれる割込ハンドラ関数です。
- A/D変換を開始します
- 移動のチェックを行い、スクロールするならスクロールイベントを発生します
- g1msecCounterB変数を+1します

ad_user.c

```
__interrupt void MD_INTAD( void )
```
A/D変換終了時に呼ばれる割込ハンドラ関数です。
- A/D値を読み込み gTextScrollSpeed へ反映させます

int_user.c

```
__interrupt void MD_INTP1( void )
```
外部割り込みINTP1に呼ばれる割込ハンドラ関数です。
- SW1が押下されたらINTP1押下イベントを発生します。

user_define.h

下記のイベントを定義します

```
#define D_EV_INTP0   0x0000001    /* INTP0押下    */
#define D_EV_INTP1   0x0000002    /* INTP1押下    */
#define D_EV_INTP3   0x0000004    /* INTP3押下    */
#define D_EV_DYNAMIC 0x0000008
 /* マトリックスLEDをダイナミック点灯させる */
#define D_EV_SCROLL  0x0000010
 /* マトリックスLEDを1ドットスクロールする  */
```

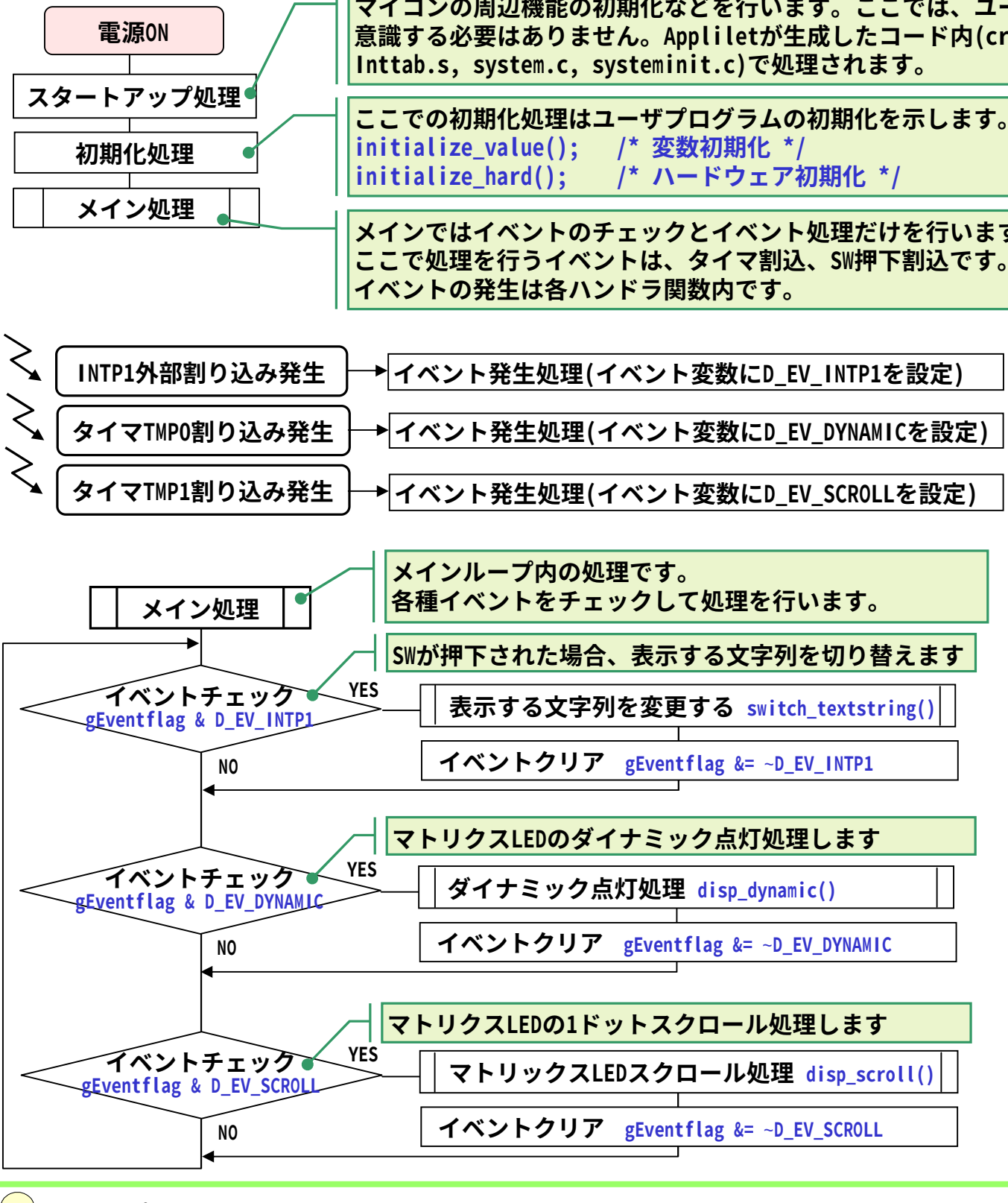
マトリックスLED(表示関係)の定義

```
#define D_MLED_CNT          2
 /* マトリックスLEDの数(仮想数)              */
```
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
 /* マトリックスLEDの row (縦列数)           */

#define D_FONT_WIDTH        6
 /* 表示する1文字のフォント幅                */

#define D_MAXTEXT           34
 /* スクロールして表示させる最大文字数       */
```

動作概要

動作の概要をフローチャートを使って説明します。このプログラムの作りは一般的な作りになっています。詳細は次ページからの動作詳細を参考にしてください。

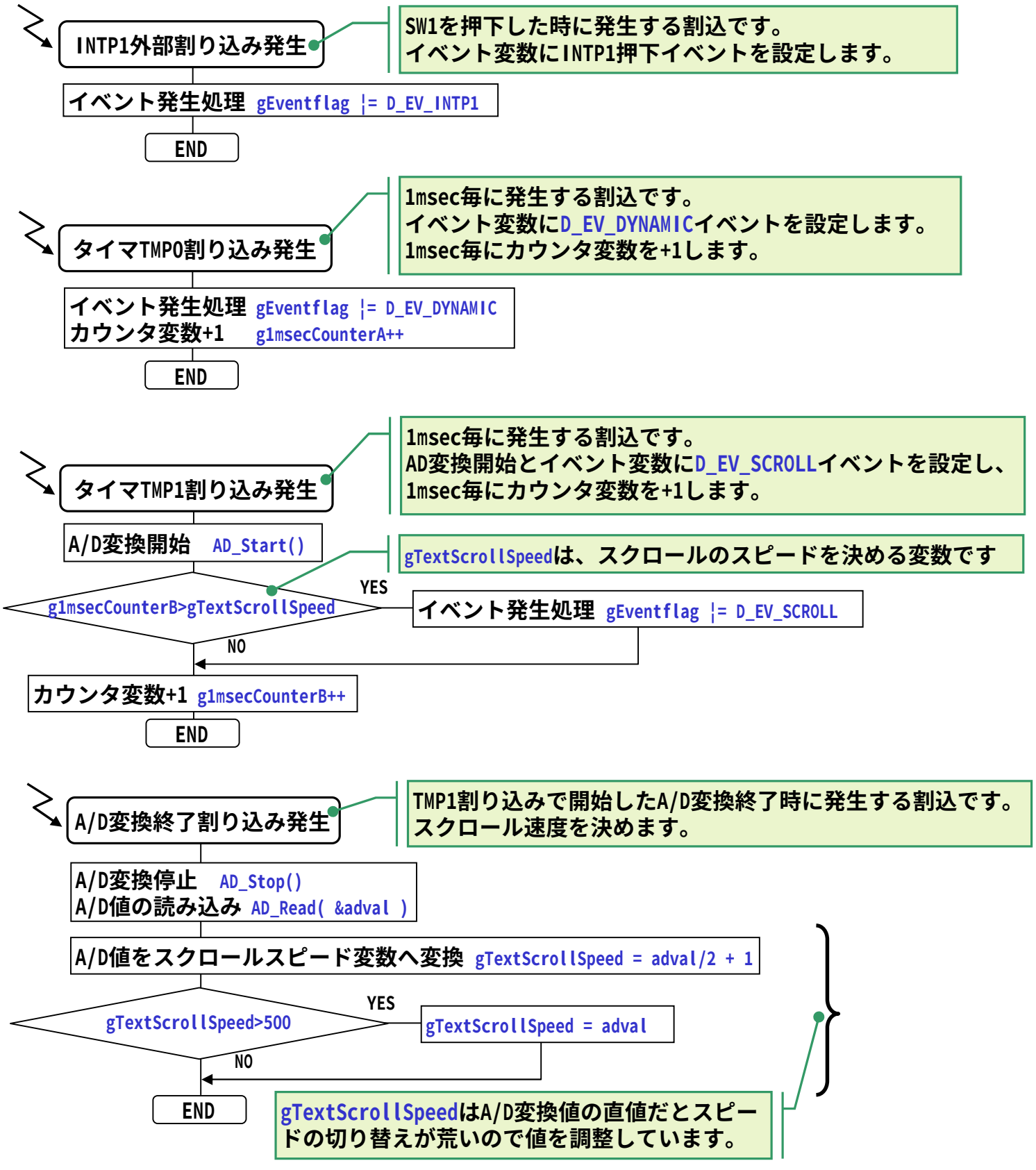


ワンポイント

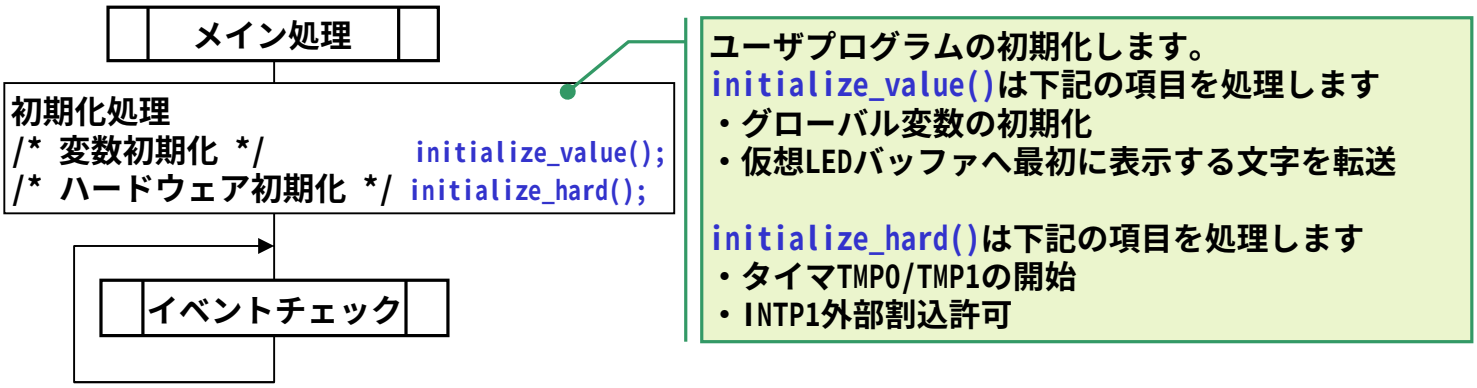
ここでは基本的な流れのみを説明しています。重要なことは**「割り込み内では処理を最少にする」**、そして割り込みで設定したイベントを**「メインループで処理する」**の2点です。

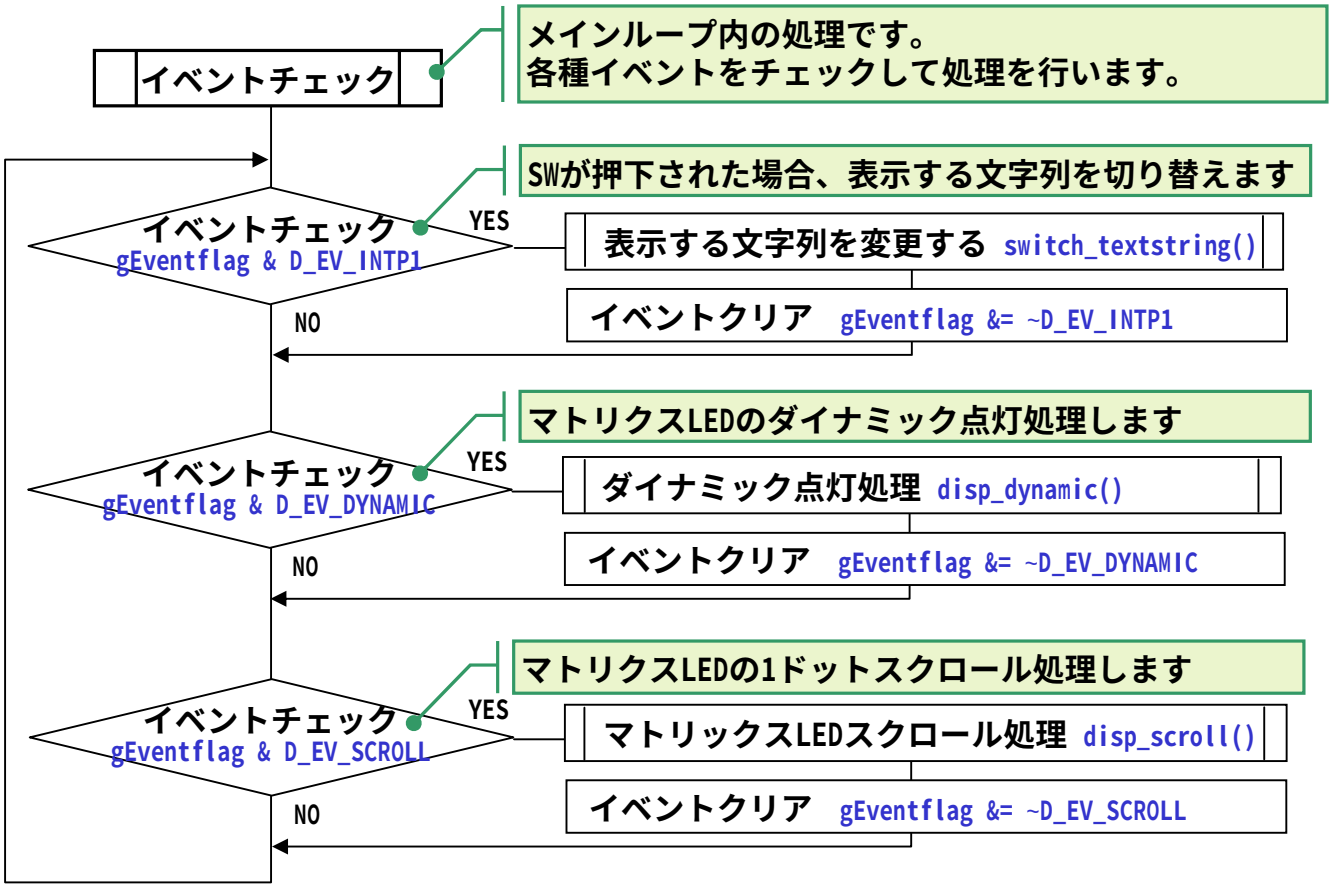
動作詳細(その1)

動作の詳細をフローチャートを使って説明します。プログラムリストと合わせてフローチャートを参照するとプログラムへの理解が深まります。

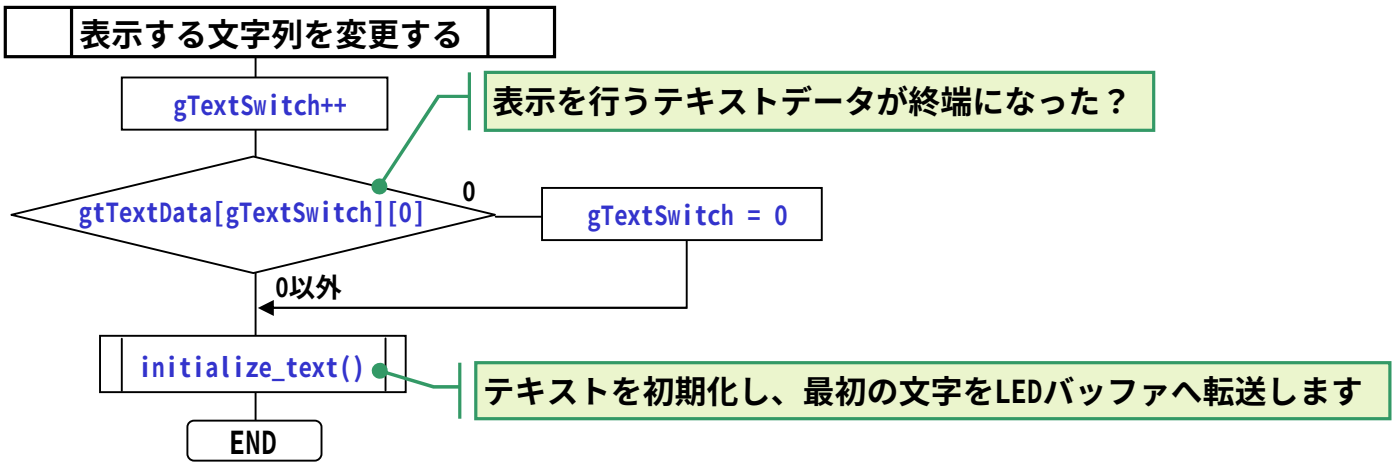


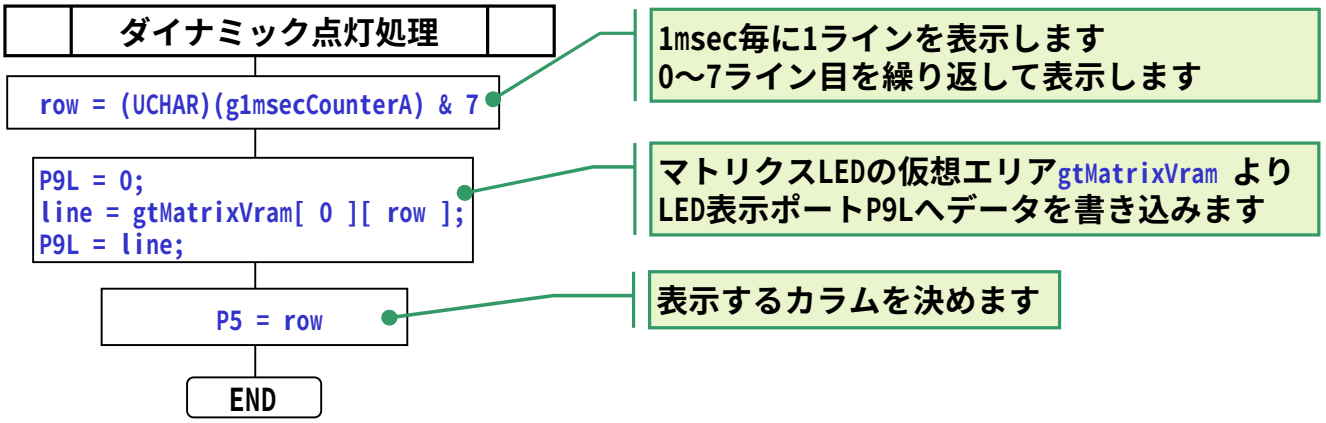
動作詳細(その2)

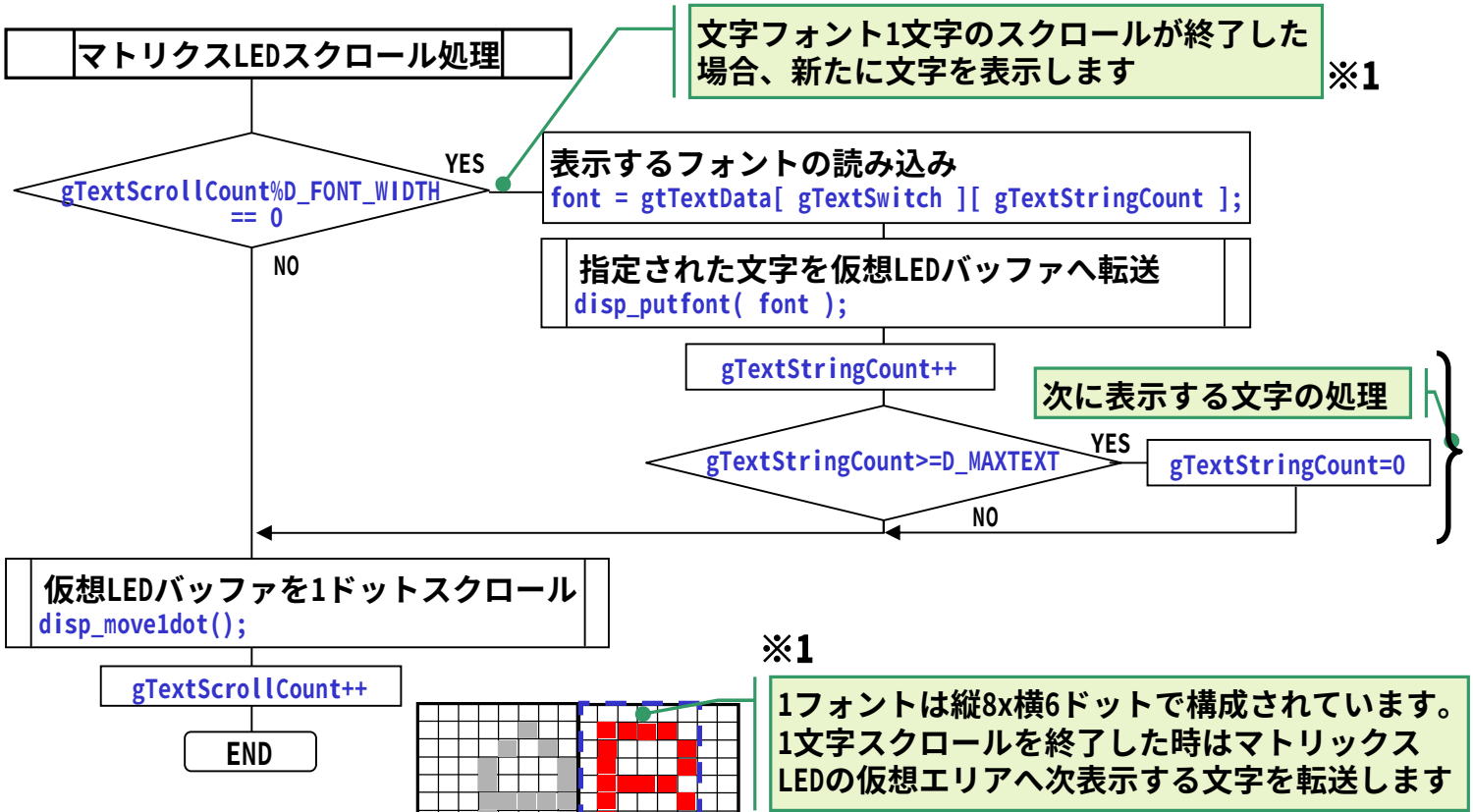

メイン処理
初期化処理
/* 変数初期化 */ initialize_value();
/* ハードウェア初期化 */ initialize_hard();
イベントチェック
ユーザプログラムの初期化します。
initialize_value()は下記の項目を処理します
・グローバル変数の初期化
・仮想LEDバッファへ最初に表示する文字を転送
initialize_hard()は下記の項目を処理します
・タイマTMP0/TMP1の開始
・INTP1外部割込許可


イベントチェック
メインループ内の処理です。
各種イベントをチェックして処理を行います。
SWが押下された場合、表示する文字列を切り替えます
イベントチェック
gEventflag & D_EV_INTP1
YES
表示する文字列を変更する switch_textstring()
イベントクリア gEventflag &= ~D_EV_INTP1
NO
マトリクスLEDのダイナミック点灯処理します
イベントチェック
gEventflag & D_EV_DYNAMIC
YES
ダイナミック点灯処理 disp_dynamic()
イベントクリア gEventflag &= ~D_EV_DYNAMIC
NO
マトリクスLEDの1ドットスクロール処理します
イベントチェック
gEventflag & D_EV_SCROLL
YES
マトリックスLEDスクロール処理 disp_scroll()
イベントクリア gEventflag &= ~D_EV_SCROLL
NO

動作詳細(その3)



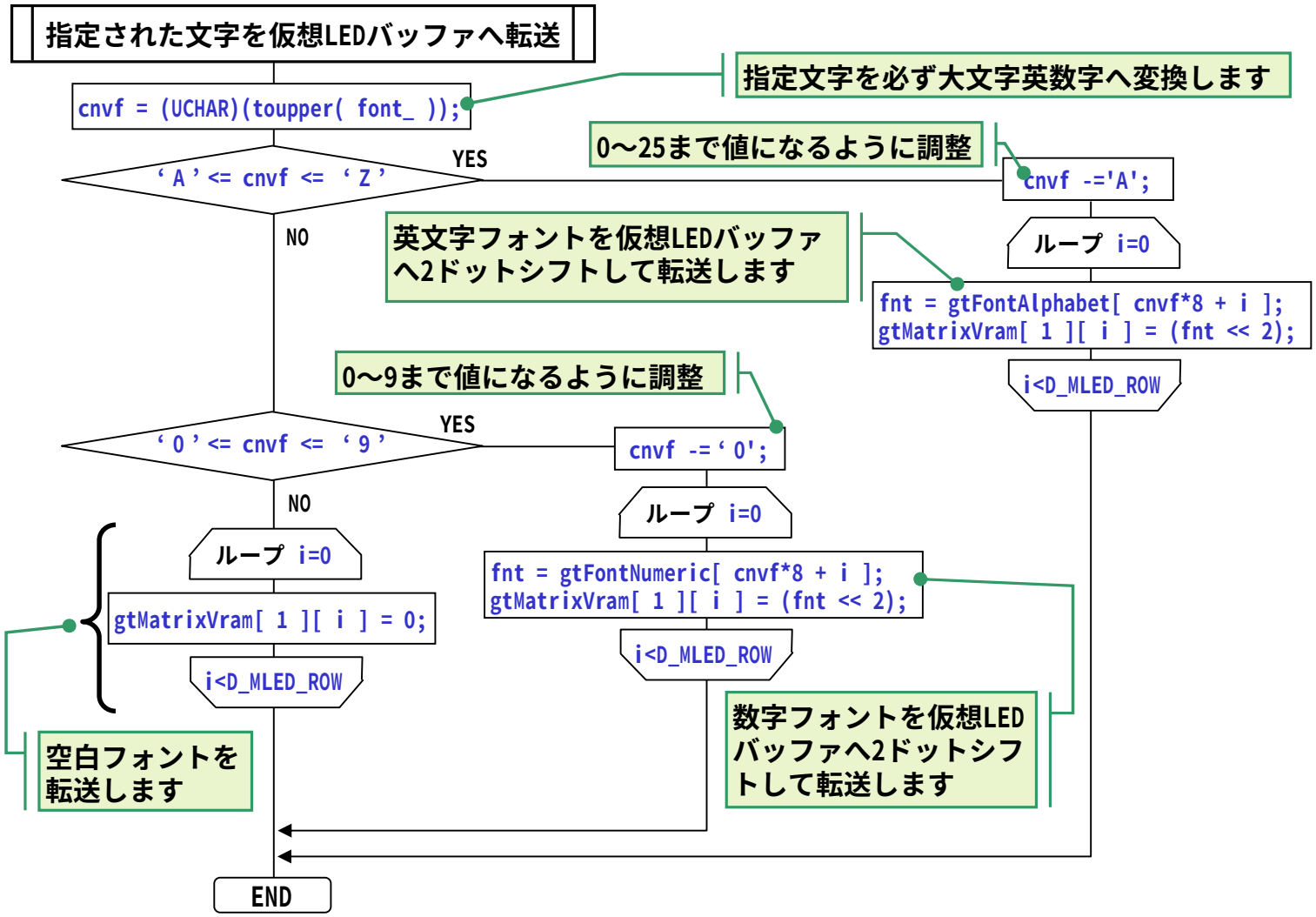


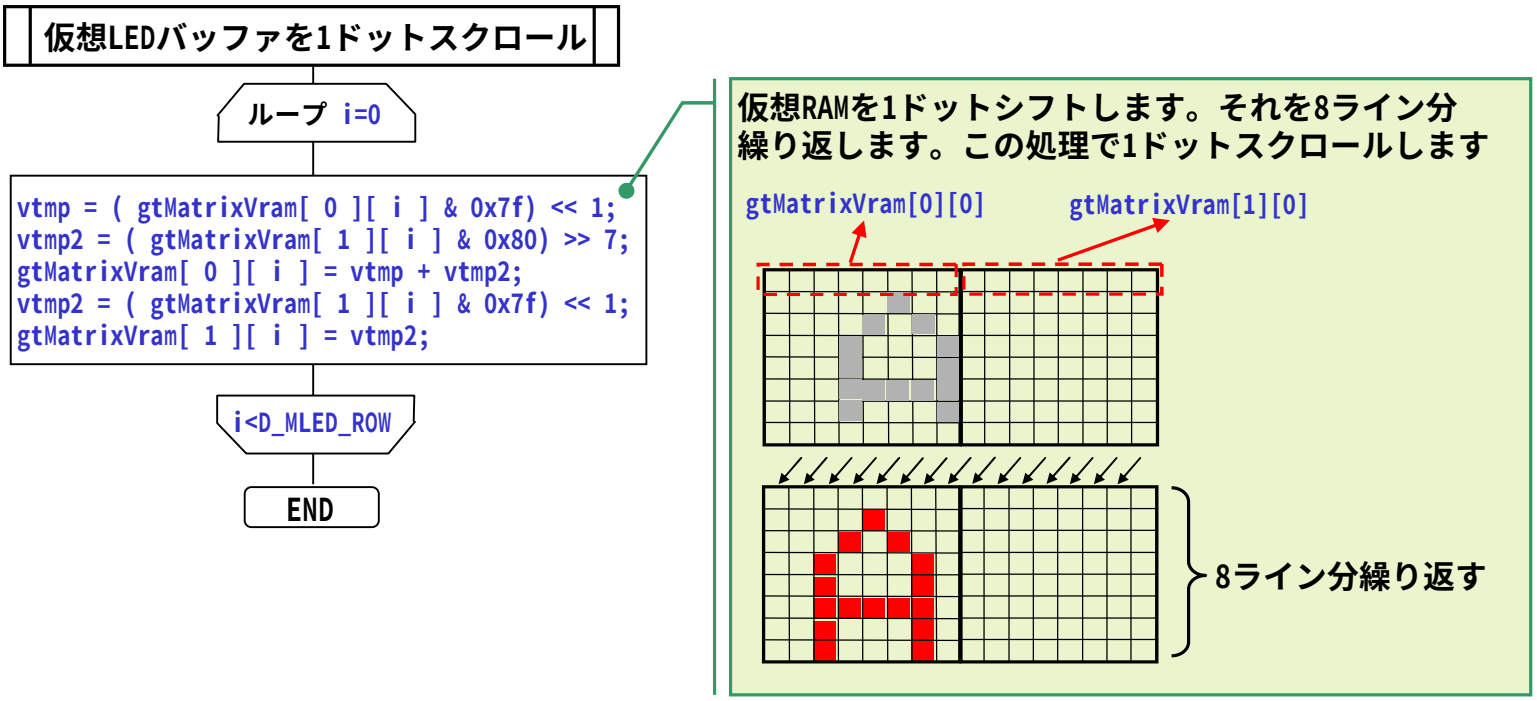


※1

1フォントは縦8x横6ドットで構成されています。
1文字スクロールを終了した時はマトリックス
LEDの仮想エリアへ次表示する文字を転送します

動作詳細(その4)


指定された文字を仮想LEDバッファへ転送
cnvf = (UCHAR)(toupper(font_));
指定文字を必ず大文字英数字へ変換します
'A' <= cnvf <= 'Z'
YES
0～25まで値になるように調整
cnvf -='A';
NO
英文字フォントを仮想LEDバッファへ2ドットシフトして転送します
ループ i=0
fnt = gtFontAlphabet[cnvf*8 + i];
gtMatrixVram[1][i] = (fnt << 2);
i<D_MLED_ROW
0～9まで値になるように調整
'0' <= cnvf <= '9'
YES
cnvf -= '0';
NO
ループ i=0
fnt = gtFontNumeric[cnvf*8 + i];
gtMatrixVram[1][i] = (fnt << 2);
i<D_MLED_ROW
ループ i=0
gtMatrixVram[1][i] = 0;
i<D_MLED_ROW
空白フォントを転送します
数字フォントを仮想LEDバッファへ2ドットシフトして転送します
END


仮想LEDバッファを1ドットスクロール
ループ i=0
vtmp = (gtMatrixVram[0][i] & 0x7f) << 1;
vtmp2 = (gtMatrixVram[1][i] & 0x80) >> 7;
gtMatrixVram[0][i] = vtmp + vtmp2;
vtmp2 = (gtMatrixVram[1][i] & 0x7f) << 1;
gtMatrixVram[1][i] = vtmp2;
i<D_MLED_ROW
END
仮想RAMを1ドットシフトします。それを8ライン分繰り返します。この処理で1ドットスクロールします
gtMatrixVram[0][0]
gtMatrixVram[1][0]
8ライン分繰り返す

プログラム動作説明

g.全体の処理

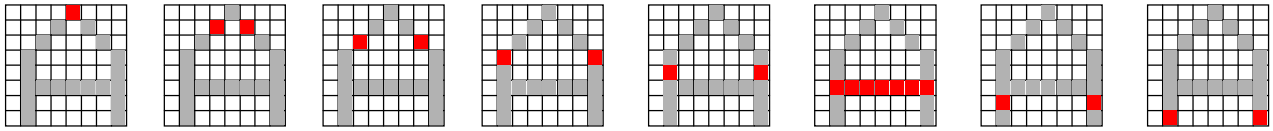
プログラムの動作として、下記の5つに分けられます
1.メイン処理
2.タイマTMP0(インターバル･タイマとして使用)
3.タイマTMP1(インターバル･タイマとして使用)
4.A/Dコンバータ完了割り込み
5.INTP1外部割り込み

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TMP0/TMP1を開始します。
処理としては無限ループとなります。ループ内ではイベントチェックを行います。

2.TMP0タイマ0 __interrupt void MD_INTTP0CC0( )

1msec毎に起動されます。D_EV_DYNAMICイベント設定します。メインで行うイベント処理では
gtMatrixVram[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.TMP1タイマ1 __interrupt void MD_INTTP1CC0( )

1msec毎に起動されます。
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 gTextScrollSpeed 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1は呼ばれる毎に g1msecCounterBを+1していますので、gTextScrollSpeedとg1msecCounterB
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D_EV_SCROLLイベントを設定します。

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

タイマ1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 gTextScrollSpeed へ代入されます。gTextScrollSpeed の値は 0 以外を想定していますので、
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調整しています。

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

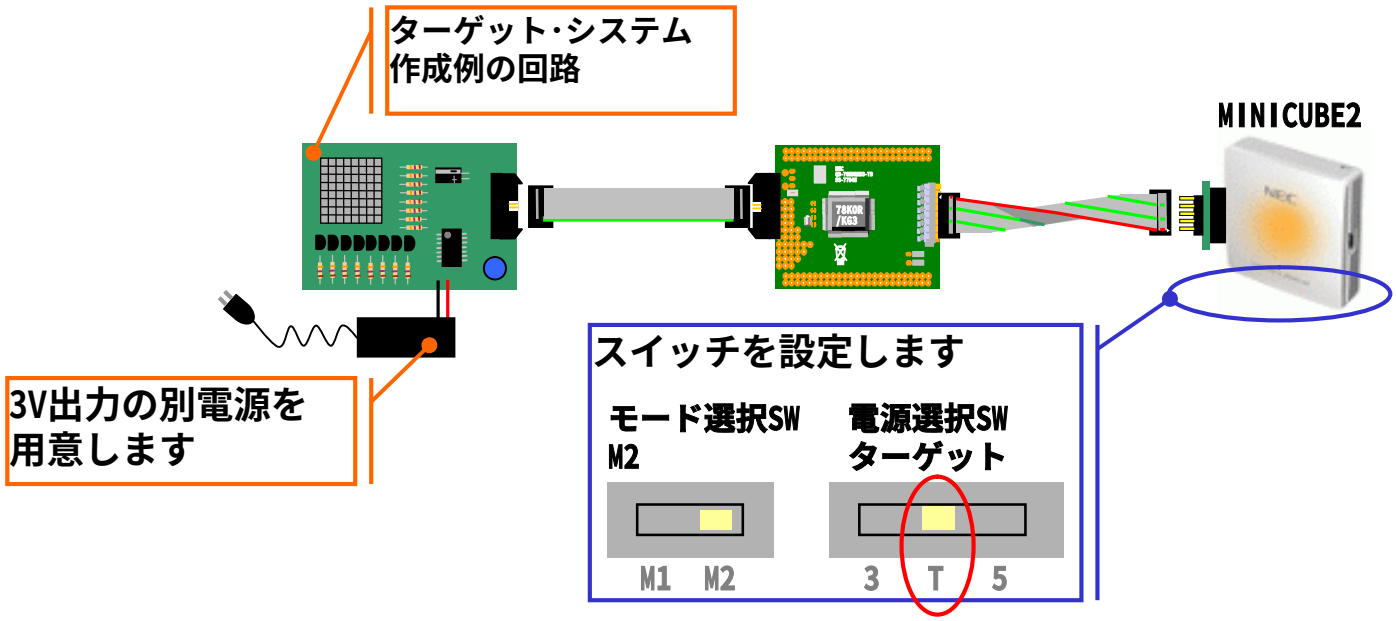
INTP1端子がLOWになった場合(SWが押下された場合)に呼ばれます。
D_EV_INTP1イベント設定します。メインで行うイベント処理ではスクロールするテキストを変えます。
表示させるテキストは1文が32文字で構成されています。
gtTextData[][D_MAXTEXT]の内容を書き換えればマトリックスLEDへ表示させる事ができますが、
表示するデータは必ず32byteで構成して下さい。

プログラム動作説明

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3V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3V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。

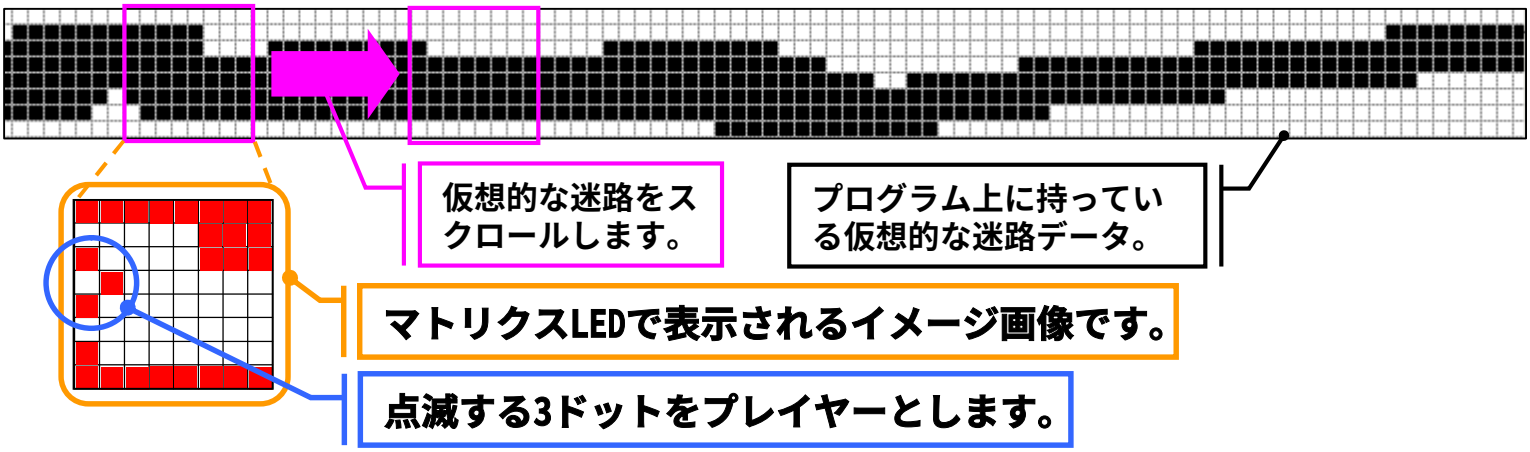
1. MINICUBE2のSWを設定する
2. 16pinターゲット･ケーブルをターゲット･ボードへ接続する
3. ターゲット･システム回路とターゲット･ボードを接続する
4. MINICUBE2とPC本体を接続する(MINICUBE2のLEDが白点滅します)
5. ターゲット･システム回路へ電源を供給する(MINICUBE2のLEDが白点灯します)

ゲームを作る

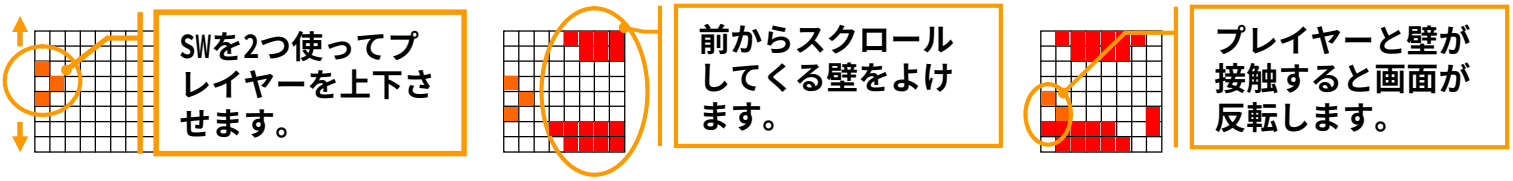
この章ではターゲット･システム作成例2を紹介します。電光掲示板では文字をスクロールさせましたが、今度は文字ではなく迷路をスクロールさせます。ただそれだけではゲーム性に乏しいのでプレイヤーを操作して迷路(障害物)をよけるゲームにします。ハードウエアが同じでもソフトウエアによって、いろいろ応用が広がることを体験して下さい。

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。



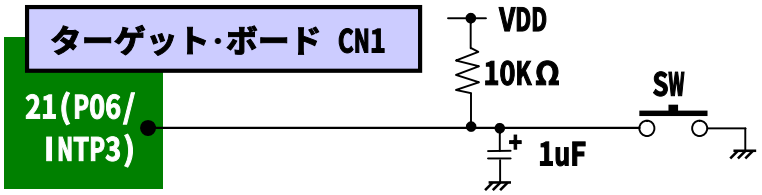
基本操作



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(INTP3外部入力)を一つ追加します。



QB-V850ESJG3L-TB、QB-V850ESJJ3-TBともにCN1のNo.21に回路を追加してください

プログラム設計

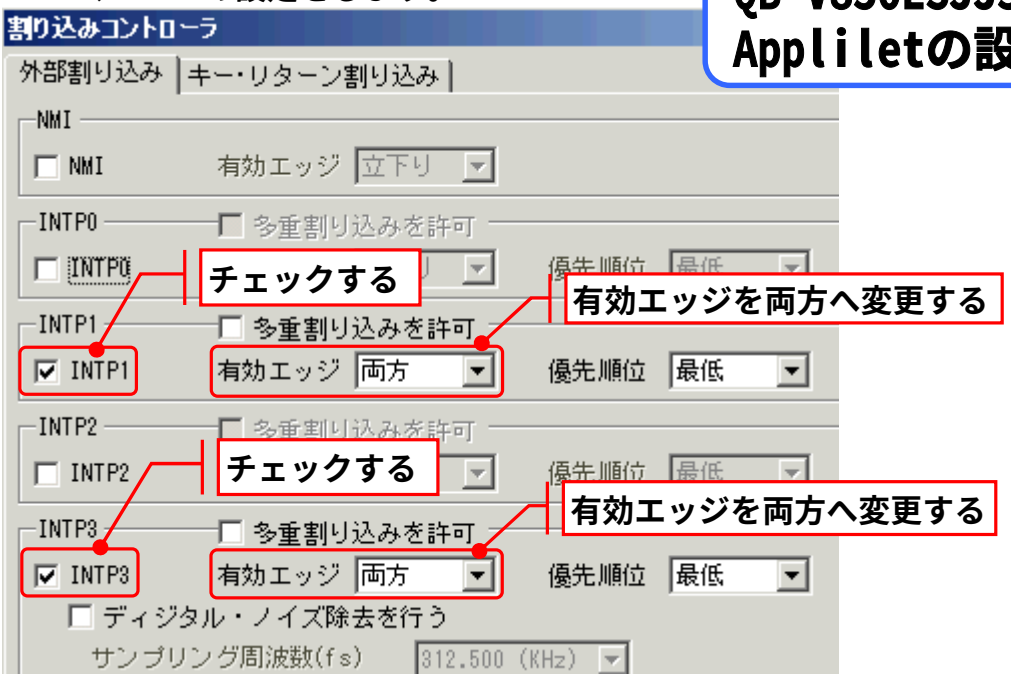
a. Applilet2を使いプログラムを設計します。

[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、INTP3の設定をします。

QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
Appliletの設定は共通です。



c. 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　ユーザが書く外部割り込み処理
- timer_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

QB-V850ESJG3L-TB
QB-V850ESJJ3-TB
プログラムは共通です。

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 */

/* メインで処理を行うイベントコード */
#define D_EV_INTP0          0x0000001   /* INTP0押下                          */
#define D_EV_INTP1          0x0000002   /* INTP1押下                          */
#define D_EV_INTP3          0x0000004   /* INTP3押下                          */
#define D_EV_DYNAMIC        0x0000008   /* マトリックスLEDをダイナミック点灯させる */
#define D_EV_SCROLL         0x0000010   /* マトリックスLEDを1ドットスクロールする  */
#define D_EV_BOMB           0x0000020   /* プレイヤー衝突中                   */

/* マトリックスLED(表示関係)の定義 */
#define D_MLED_CNT           2          /* マトリックスLEDの数(仮想数)        */
#define D_MLED_ROW           8          /* マトリックスLEDの row (縦列数)     */
#define D_FONT_WIDTH         8          /* 表示するフォント(迷路)幅           */
#define D_MAZE_WIDTH       125          /* 表示する迷路の幅(125x8)ドット      */
#define D_BOMB_MAX        3000          /* プレイヤーと迷路が衝突した際の点滅回数 */

/* 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 */
#include "string.h"

/* in This file. */
void initialize_hard( void );
void initialize_value( void );
void trans_virtual2real( void );
void eventcheck( 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 gtMazeData[ D_MAZE_WIDTH * D_MLED_ROW ] =
{
                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,0xff,0xe0,0x00,0x00,
  0x00,0xff,0xff,0xff,0xff,0xff,0xff,0xf0,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0xc0,0x00,0x00,0x00,0x07,0xf8,0x00,0x00,
  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,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
```

プログラムリスト

main.c(リスト2)

```
                0x00,0x00,0x08,0x0e,0x06,0x3e,0x30,0x83,0xe0,0x1e,0x1f,0x80,0x70,
  0x0e,0x00,0x1f,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,

                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,
  0x00,0x00,0x00,0x00,0x00,0x00,0x06,0x00,0x00,0x00,0x01,0xff,0xff,0xff,0xff,
  0xf8,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x80,0x00,0x00,0x00,0x00,0x00,0x00,
  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,
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x02,0x00,0x00,
  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,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,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,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x04,0x02

};

ULONG   gEventflag;                     /* メインループでチェックするイベント。D_EV_xxxxで定義  */
                                        /* bit0 : INTP0押下                                    */
                                        /* bit1 : INTP1押下                                    */
                                        /* bit2 : INTP3押下                                    */
                                        /* bit3 : マトリックスLEDをダイナミック点灯させる      */
                                        /* bit4 : マトリックスLEDを1ドットスクロールする       */
                                        /* bit5 - bit31: reserve(予約)                         */
```

プログラムリスト

main.c(リスト3)

```
UINT    g1msecCounterA;                 /* Interval timer 1msec(tmp0使用)                  */
UINT    g1msecCounterB;                 /* Interval timer 1msec(tmp1使用)                  */
USHORT  gMazeScrollCount;               /* 表示する迷路のドット単位のカウンタ              */
USHORT  gMazeSpeed;                     /* 表示する迷路のスピード                          */
BOOL    gPlaneBomb;                     /* プレイヤーと迷路が衝突したかのフラグ            */
USHORT  gPlaneBombCounter;              /* プレイヤーと迷路が衝突した時の表示点滅カウンタ  */
UCHAR   gPlaneLocation;                 /* プレイヤーの表示位置                            */
UCHAR   gIntp1Sw;
UCHAR   gIntp3Sw;

/* gtMatrixVram is virtual vram. gtMatrixRealVram is real vram. */
UCHAR   gtMatrixVram[ D_MLED_CNT ][ D_MLED_ROW ];
UCHAR   gtMatrixRealVram[ D_MLED_ROW ];

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

    /* 変数初期化 */
    gMazeScrollCount = 0;
    initialize_value();

    /* ハードウェア初期化 */
    initialize_hard();

    /* メインループ */
    while ( 1 )
    {
        eventcheck();
    }
    /* End user code. Do not edit comment generated here */
}
/* Start adding user code. Do not edit comment generated here */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* ハードウェア初期化                                                          */
void initialize_hard( void )
{
    /* Target-Board LED off */
    PCM.2 = 1;
    PCM.3 = 1;

    INTP1_Enable();
    INTP3_Enable();

    /* start timer */
    TMP0_Start();
    TMP1_Start();
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 変数初期化                                                                  */
void initialize_value( void )
{
    g1msecCounterA = 0;
    g1msecCounterB = 0;
    gMazeScrollCount = 0;
    gMazeSpeed = 0;
    gPlaneLocation = 0;
    gPlaneBomb = MD_FALSE;
    gPlaneBombCounter = D_BOMB_MAX;
    gIntp1Sw = 0;
    gIntp3Sw = 0;

    memset( gtMatrixVram, 0x00, sizeof( gtMatrixVram ));
    memset( gtMatrixRealVram, 0x00, sizeof( gtMatrixRealVram ));
}
```

ここで初期化しているのはプレイヤーが迷路に衝突した場合、途中から再スタートさせるため、一度のみの初期化とする。

プログラムリスト

main.c(リスト4)

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

    for ( i = 0; i < D_MLED_ROW; i++ )
    {
        fnt = gtMazeData[ (i * D_MAZE_WIDTH) + (cnt_ / D_MLED_ROW) ];
        gtMatrixVram[ 1 ][ i ] = fnt;
    }
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 仮想LEDバッファを1ドットスクロールする                                 */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void disp_move1dot( void )
{
    int i;
    UCHAR vtmp, vtmp2;

    if ( ( gMazeScrollCount % D_FONT_WIDTH ) == 0 )
    {   /* next maze */
        disp_putmaze( gMazeScrollCount );
    }

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

    gMazeScrollCount++;
    if ( gMazeScrollCount >= (D_MAZE_WIDTH * D_MLED_ROW) )
    {
        gMazeScrollCount = 0;
    }

    /* player postion control */
    if ( gIntp3Sw )
    {
        if ( gPlaneLocation > 0 )
        {
            gPlaneLocation--;
        }
    }
    if ( gIntp1Sw )
    {
        if ( gPlaneLocation < 5 )
        {
            gPlaneLocation++;
        }
    }
}
```

プログラムリスト

main.c(リスト5)

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* 仮想LED表示バッファから実際にLED表示バッファへ転送する                      */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void trans_virtual2real( void )
{
    int i;

    for ( i = 0; i < D_MLED_ROW; i++ )
    {
        gtMatrixRealVram[ i ] = gtMatrixVram[ 0 ][ i ];
    }
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* プレイヤー衝突処理                                                          */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void control_bomb( void )
{
    int i;
    UCHAR dat;

    for ( i = 0; i < D_MLED_ROW; i++ )
    {
        dat = gtMatrixVram[ 0 ][ i ];
        if ( g1msecCounterB & 0x40 )
        {
            dat = dat ^ 0xff;
            gPlaneBombCounter--;
        }
        gtMatrixRealVram[ i ] = dat;
    }

    if ( gPlaneBombCounter <= 0 )
    {   /* re-start game */
        initialize_value();
    }
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* LEDマトリックスをダイナミック点灯する                                       */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void disp_dynamic( void )
{
    UCHAR row, line, pdat;

    /* Matrix LED line disp */
    row = (UCHAR)(g1msecCounterA) & 7;

    /* display column */
    P9L = 0;

    /* disp player */
    if ( g1msecCounterA & 0x40 )
    {
        if ( row == gPlaneLocation )
        {
            pdat = 0x80;
        }
        else if ( row == ( gPlaneLocation + 1 ) )
        {
            pdat = 0x40;
        }
        else if ( row == ( gPlaneLocation + 2 ) )
        {
            pdat = 0x80;
        }
        else
        {
            pdat = 0;
        }
```

プログラムリスト

main.c(リスト6)

```
        if ( gtMatrixRealVram[ row ] & pdat )
        {   /* Bomb judgment */
            gPlaneBomb = MD_TRUE;
        }
        else
        {
            gtMatrixRealVram[ row ] |= pdat;
        }
    }
    line = gtMatrixRealVram[ row ];
    P9L = line;

    /* display row */
    P5 = row;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* D_EV_xxxxで定義したイベント処理を行う                                        */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
void eventcheck()
{
    /* INTP1チェック */
    if ( gEventflag & D_EV_INTP1 )
    {
        gIntp1Sw = P0.4 ^ 1;

        /* イベントクリア */
        gEventflag &= ~D_EV_INTP1;
    }

    /* INTP3チェック */
    if ( gEventflag & D_EV_INTP3 )
    {
        gIntp3Sw = P0.6 ^ 1;

        /* イベントクリア */
        gEventflag &= ~D_EV_INTP3;
    }

    /* プレイヤー衝突処理チェック */
    if ( gEventflag & D_EV_BOMB )
    {
        /* プレイヤー衝突処理 */
        control_bomb();

        /* イベントクリア */
        gEventflag &= ~D_EV_BOMB;
    }

    /* マトリックスLEDスクロール処理チェック */
    if ( gEventflag & D_EV_SCROLL )
    {
        /* マトリックスLEDスクロール処理 */
        disp_move1dot();

        /* イベントクリア */
        gEventflag &= ~D_EV_SCROLL;
    }

    /* ダイナミック点灯処理チェック */
    if ( gEventflag & D_EV_DYNAMIC )
    {
        /* ダイナミック点灯処理 */
        disp_dynamic();

        /* イベントクリア */
        gEventflag &= ~D_EV_DYNAMIC;
    }
}

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

extern ULONG gEventflag;

/* End user code for global definition. Do not edit comment generated here */
                リスト省略
/*
**-----------------------------------------------------------------------------
** 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 */

    gEventflag |= D_EV_INTP1;

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

/*
**-----------------------------------------------------------------------------
** Abstract:            This function is INTP3 interrupt service routine.
** Parameters:          None
** Returns:             None
**-----------------------------------------------------------------------------
*/
__interrupt void MD_INTP3(void)
    /* Start user code. Do not edit comment generated here */

    gEventflag |= D_EV_INTP3;

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

ad_user.c

```
                リスト省略
/***********************************************************************************
**  Global define
***********************************************************************************/
/* Start user code for global definition. Do not edit comment generated here */
extern USHORT gMazeSpeed;
/* End user code for global definition. Do not edit comment generated here */
                リスト省略
/*-----------------------------------------------------------------------------
**  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 );
    gMazeSpeed = adval/2 + 1;
    if ( gMazeSpeed > 500 )
    {
        gMazeSpeed = adval;
    }
    /* End user code. Do not edit comment generated here */
}
```

プログラムリスト

timer_user.c

リスト省略

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

extern void trans_virtual2real( void );

/* End user code for include definition. Do not edit comment generated here */
#include "user_define.h"

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

extern ULONG  gEventflag;
extern UINT   g1msecCounterA;
extern UINT   g1msecCounterB;
extern USHORT gMazeSpeed;
extern BOOL   gPlaneBomb;

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

/*-----------------------------------------------------------------------------
**  Abstract:           This function is INTTP0CC0 interrupt service routine.
**  Parameters:         None
**  Returns:            None
**-----------------------------------------------------------------------------*/
__interrupt void MD_INTTP0CC0(void)
{
    /* Start user code. Do not edit comment generated here */

    gEventflag |= D_EV_DYNAMIC;
    g1msecCounterA++;

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

/*-----------------------------------------------------------------------------
**  Abstract:           This function is INTTP1CC0 interrupt service routine.
**  Parameters:         None
**  Returns:            None
**-----------------------------------------------------------------------------*/
__interrupt void MD_INTTP1CC0(void)
{
    /* Start user code. Do not edit comment generated here */

    AD_Start();

    if ( gPlaneBomb )
    {
        gEventflag |= D_EV_BOMB;
    }
    else
    {
        if ( g1msecCounterB > gMazeSpeed )
        {
            gEventflag |= D_EV_SCROLL;
            g1msecCounterB = 0;
        }

        trans_virtual2real();
    }

    g1msecCounterB++;

    /* End user code. 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変数と関数について説明します。

main.c

○変数について

`const UCHAR gtMazeData[];`
マトリックスLEDに表示する迷路データです

`ULONG gEventflag;`
メインループでチェックするイベント変数です

`UINT g1msecCounterA;`
`UINT g1msecCounterB;`
1msecのタイマ割り込み時に+1されるカウンタです

`UINT gMazeScrollCount;`
表示されている迷路が何ドット移動したかを示します

`USHORT gMazeSpeed;`
表示する迷路のスクロールするスピードを示します

`BOOL gPlaneBomb;`
プレイヤーと迷路が衝突したかのフラグ。衝突時は1です

`USHORT gPlaneBombCounter;`
プレイヤーと迷路が衝突した時の表示点滅カウンタ

`UCHAR gPlaneLocation;`
プレイヤーの表示位置(X軸は固定なのでY軸のみ)

`UCHAR gtMatrixVram[ D_MLED_CNT ][ D_MLED_ROW ];`
マトリックスLEDに表示する仮想の迷路データです。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 gtMatrixRealVram[ D_MLED_ROW ];`
マトリックスLEDに実際に表示するバッファです。迷路データとプレイヤーデータの重ね合わせして表示します。

○関数について

`void initialize_hard( void );`
`void initialize_value( void );`
初期化用の関数です

`void disp_putmaze( UCHAR cnt_ );`
パラメータ cnt_(asciiコード)で指定された迷路データをマトリックス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

`void control_bomb( void );`
プレイヤーと迷路の衝突処理します

`void disp_dynamic( void );`
マトリックスLEDのダイナミック点灯表示します

`void eventcheck( void );`
イベント変数をチェックし、発生したイベントを処理します

timer_user.c

__interrupt void MD_INTTP0CC0()
1msec毎に呼ばれる割込ハンドラ関数です。
・ダイナミック点灯用のイベントを発生します
・g1msecCounterA変数を+1します

__interrupt void MD_INTTP1CC0()
1msec毎に呼ばれる割込ハンドラ関数です。
・A/D変換を開始します
・衝突フラグをチェックして衝突イベントを発生します
・移動のチェックを行い、スクロールするならスクロールイベントを発生します
・g1msecCounterB変数を+1します

int_user.c

__interrupt void MD_INTP1(void)
外部割り込みINTP1に呼ばれる割込ハンドラ関数です。
・SW1が押下されたらINTP1押下イベントを発生します。

__interrupt void MD_INTP3(void)
外部割り込みINTP3に呼ばれる割込ハンドラ関数です。
・SW1が押下されたらINTP3押下イベントを発生します。

ad_user.c

__interrupt void MD_INTAD(void)
A/D変換終了時に呼ばれる割込ハンドラ関数です。
・A/D値を読み込み gMazeSpeed へ反映させます

user_define.h

下記のイベントを定義します

```
#define D_EV_INTP1    0x0000002 /* INTP1押下 */
#define D_EV_INTP3    0x0000004 /* INTP3押下 */
#define D_EV_DYNAMIC 0x0000008 /* マトリックスLED
をダイナミック点灯させる */
#define D_EV_SCROLL   0x0000010 /* マトリックスLED
を1ドットスクロールする  */
#define D_EV_BOMB     0x0000020 /* プレイヤー衝突中*/
```

マトリックスLED(表示関係)の定義

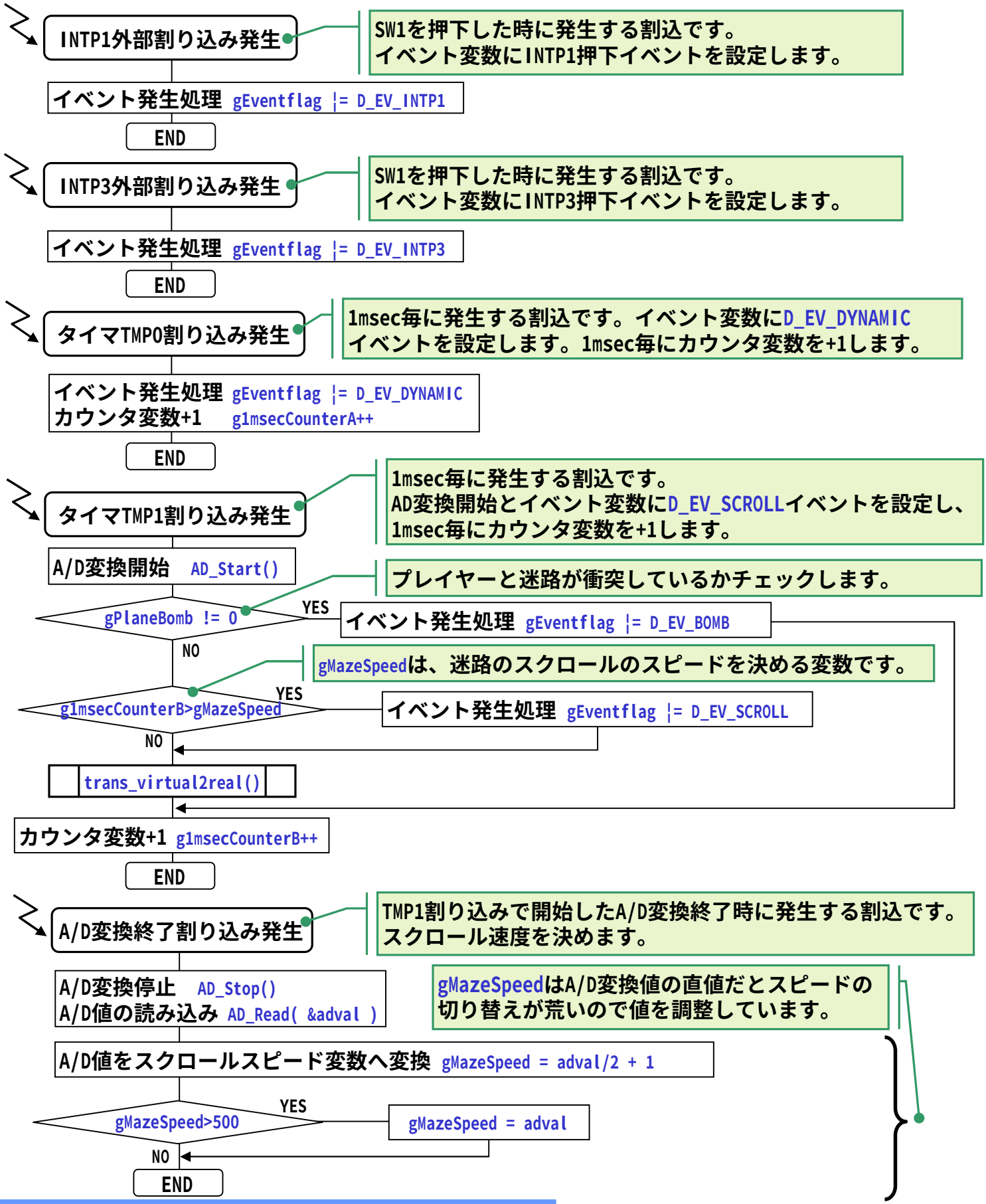
```
#define D_MLED_CNT          2
 /* マトリックスLEDの数(仮想数)              */
```

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。

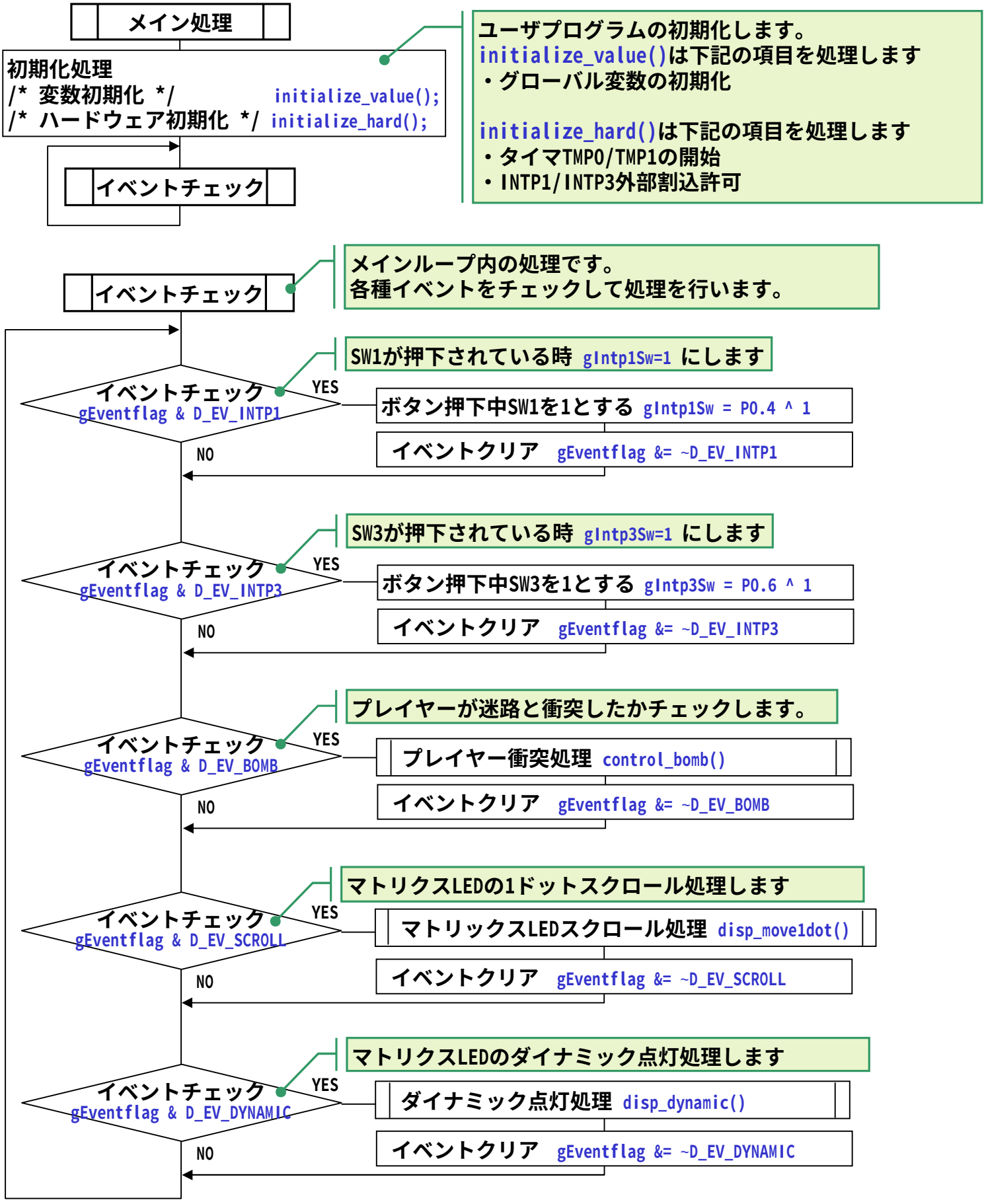
```
#define D_MLED_ROW          8
 /* マトリックスLEDの row (縦列数)           */

#define D_MAZE_WIDTH       125
 /* 表示する迷路の幅(125x8)ドット            */

#define D_BOMB_MAX        3000
 /* プレイヤーと迷路が衝突した際の点滅回数   */
```

動作詳細(その1)

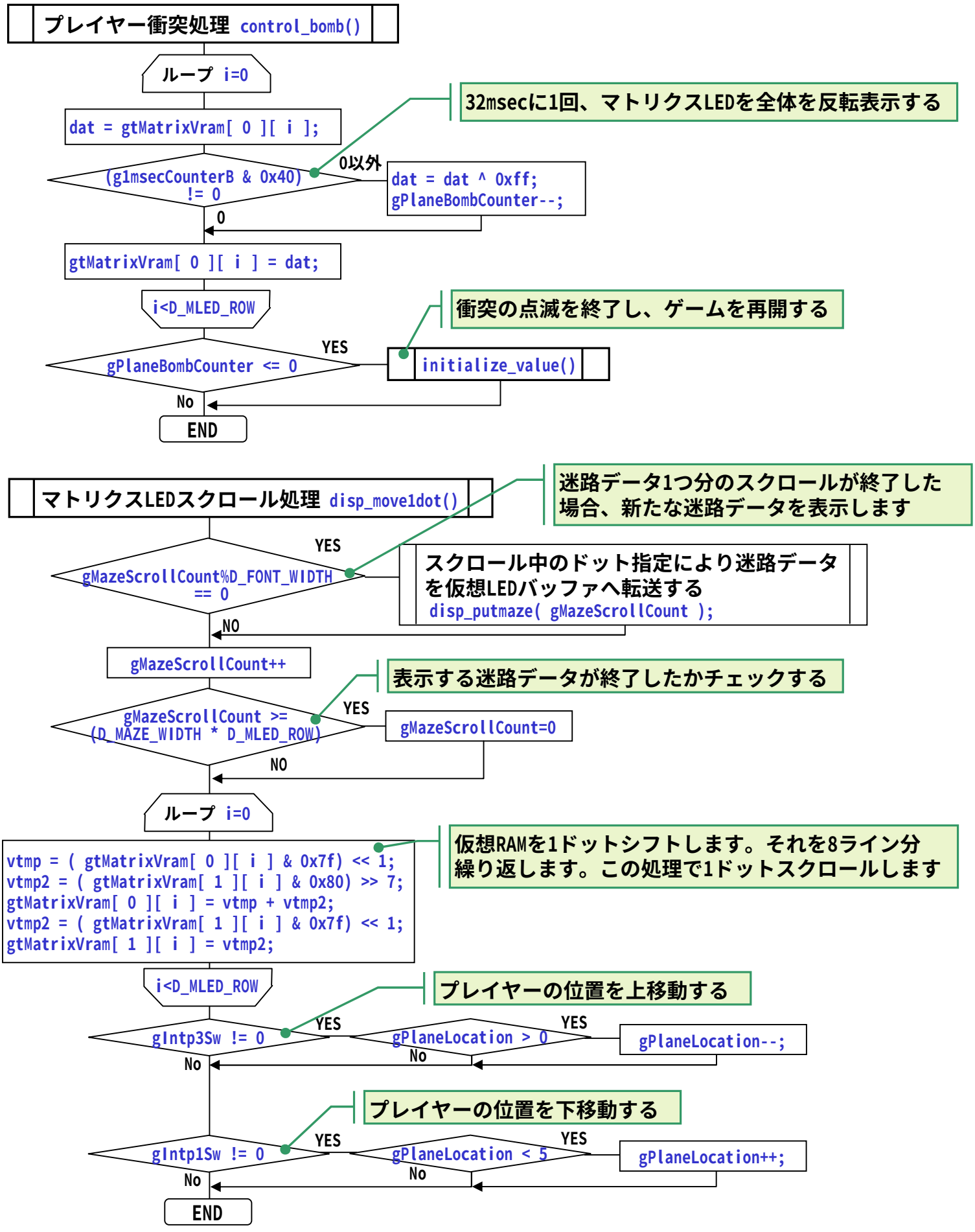
動作の詳細をフローチャートを使って説明します。プログラムリストと合わせてフローチャートを参照するとプログラムへの理解が深まります。



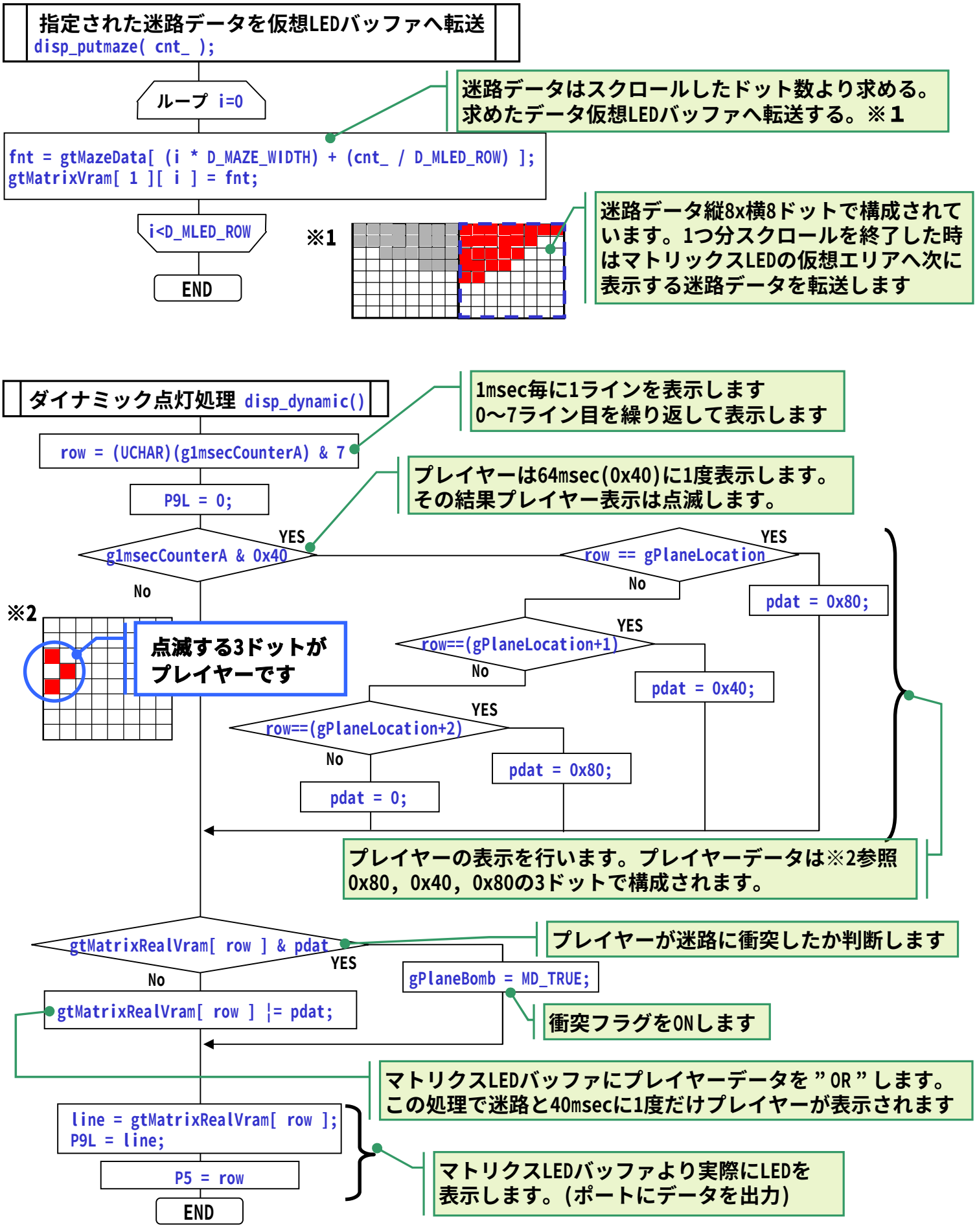
動作詳細(その2)


メイン処理
初期化処理
/* 変数初期化 */ initialize_value();
/* ハードウェア初期化 */ initialize_hard();
イベントチェック
ユーザプログラムの初期化します。
initialize_value()は下記の項目を処理します
・グローバル変数の初期化
initialize_hard()は下記の項目を処理します
・タイマTMP0/TMP1の開始
・INTP1/INTP3外部割込許可
イベントチェック
メインループ内の処理です。
各種イベントをチェックして処理を行います。
イベントチェック gEventflag & D_EV_INTP1
YES
SW1が押下されている時 gIntp1Sw=1 にします
ボタン押下中SW1を1とする gIntp1Sw = P0.4 ^ 1
イベントクリア gEventflag &= ~D_EV_INTP1
NO
イベントチェック gEventflag & D_EV_INTP3
YES
SW3が押下されている時 gIntp3Sw=1 にします
ボタン押下中SW3を1とする gIntp3Sw = P0.6 ^ 1
イベントクリア gEventflag &= ~D_EV_INTP3
NO
イベントチェック gEventflag & D_EV_BOMB
YES
プレイヤーが迷路と衝突したかチェックします。
プレイヤー衝突処理 control_bomb()
イベントクリア gEventflag &= ~D_EV_BOMB
NO
イベントチェック gEventflag & D_EV_SCROLL
YES
マトリクスLEDの1ドットスクロール処理します
マトリックスLEDスクロール処理 disp_move1dot()
イベントクリア gEventflag &= ~D_EV_SCROLL
NO
イベントチェック gEventflag & D_EV_DYNAMIC
YES
マトリクスLEDのダイナミック点灯処理します
ダイナミック点灯処理 disp_dynamic()
イベントクリア gEventflag &= ~D_EV_DYNAMIC
NO

動作詳細(その3)


プレイヤー衝突処理 control_bomb()
ループ i=0
dat = gtMatrixVram[0][i];
(g1msecCounterB & 0x40) != 0
0以外
dat = dat ^ 0xff;
gPlaneBombCounter--;
0
gtMatrixVram[0][i] = dat;
i<D_MLED_ROW
gPlaneBombCounter <= 0
YES
initialize_value()
No
END
32msecに1回、マトリクスLEDを全体を反転表示する
衝突の点滅を終了し、ゲームを再開する
マトリクスLEDスクロール処理 disp_move1dot()
gMazeScrollCount%D_FONT_WIDTH == 0
YES
スクロール中のドット指定により迷路データを仮想LEDバッファへ転送する
disp_putmaze(gMazeScrollCount);
NO
gMazeScrollCount++
gMazeScrollCount >= (D_MAZE_WIDTH * D_MLED_ROW)
YES
gMazeScrollCount=0
NO
ループ i=0
vtmp = (gtMatrixVram[0][i] & 0x7f) << 1;
vtmp2 = (gtMatrixVram[1][i] & 0x80) >> 7;
gtMatrixVram[0][i] = vtmp + vtmp2;
vtmp2 = (gtMatrixVram[1][i] & 0x7f) << 1;
gtMatrixVram[1][i] = vtmp2;
i<D_MLED_ROW
gIntp3Sw != 0
YES
gPlaneLocation > 0
YES
gPlaneLocation--;
No
No
gIntp1Sw != 0
YES
gPlaneLocation < 5
YES
gPlaneLocation++;
No
No
END
迷路データ1つ分のスクロールが終了した場合、新たな迷路データを表示します
表示する迷路データが終了したかチェックする
仮想RAMを1ドットシフトします。それを8ライン分繰り返します。この処理で1ドットスクロールします
プレイヤーの位置を上移動する
プレイヤーの位置を下移動する

動作詳細(その4)


指定された迷路データを仮想LEDバッファへ転送
disp_putmaze(cnt_);
ループ i=0
fnt = gtMazeData[(i * D_MAZE_WIDTH) + (cnt_ / D_MLED_ROW)];
gtMatrixVram[1][i] = fnt;
i<D_MLED_ROW
END
迷路データはスクロールしたドット数より求める。
求めたデータ仮想LEDバッファへ転送する。※1
※1
迷路データ縦8x横8ドットで構成されています。1つ分スクロールを終了した時はマトリックスLEDの仮想エリアへ次に表示する迷路データを転送します
ダイナミック点灯処理 disp_dynamic()
row = (UCHAR)(g1msecCounterA) & 7
1msec毎に1ラインを表示します
0～7ライン目を繰り返して表示します
P9L = 0;
g1msecCounterA & 0x40
YES
No
プレイヤーは64msec(0x40)に1度表示します。
その結果プレイヤー表示は点滅します。
row == gPlaneLocation
YES
No
pdat = 0x80;
row==(gPlaneLocation+1)
YES
No
pdat = 0x40;
row==(gPlaneLocation+2)
YES
No
pdat = 0x80;
pdat = 0;
※2
点滅する3ドットがプレイヤーです
プレイヤーの表示を行います。プレイヤーデータは※2参照
0x80，0x40，0x80の3ドットで構成されます。
gtMatrixRealVram[row] & pdat
YES
No
プレイヤーが迷路に衝突したか判断します
gPlaneBomb = MD_TRUE;
衝突フラグをONします
gtMatrixRealVram[row] |= pdat;
マトリクスLEDバッファにプレイヤーデータを”OR”します。
この処理で迷路と40msecに1度だけプレイヤーが表示されます
line = gtMatrixRealVram[row];
P9L = line;
P5 = row
END
マトリクスLEDバッファより実際にLEDを
表示します。(ポートにデータを出力)

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 gtMazeData[] データ作成方法

main.cのconst UCHAR gtMazeData[] 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。

最後の応用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。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。

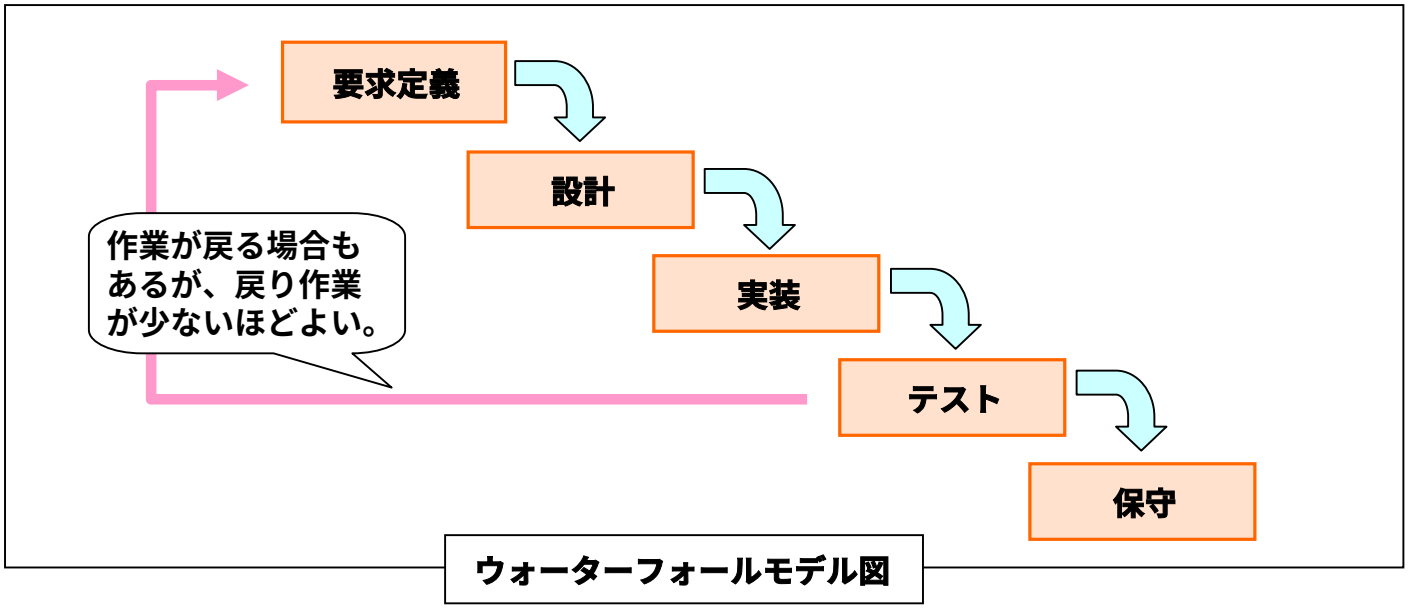
開発前に考えること(1)

この章ではプログラムを組めるようになった先の「次へのステップ」として、プログラムの設計～開発におけるポイントをQ&Aの形式で説明します。

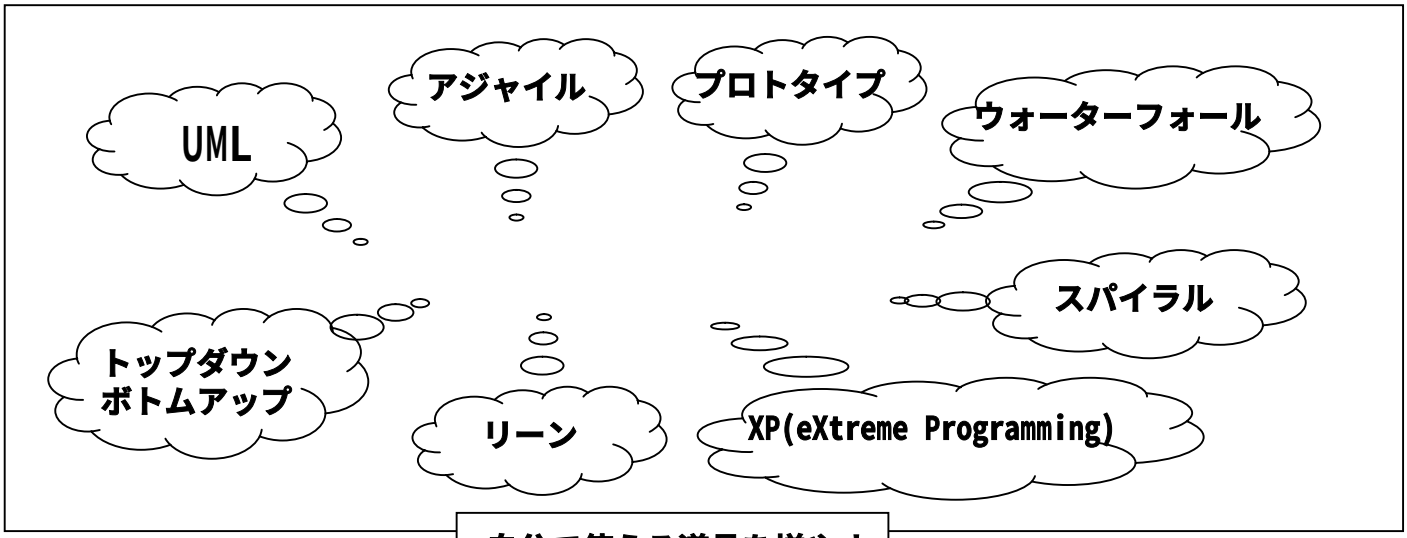
Q.開発手法は何を使えばいいですか？

昔からの開発手法にウォーターフォールモデルがあります。簡単に説明すると次のような順で開発を行うことを示します。「要求定義、設計、実装、テスト、保守」ただ、問題も指摘されており要求定義が曖昧だと戻りの作業が多く発生します。その他の開発手法としてスパイラルモデル、XP、アジャイルソフトウエア開発など多種多様あります。

では何がいいのか？答えは目的によって異なります。**目的、仕様が明確ならウォーターフォールモデルでも充分機能しますし、1人で最初から最後までの工程を行うなら、それが最適と言えます。重要なのは自分で使える道具を増やすことと、自らの成功パターンがあるならそれに則って開発することです。**



ウォーターフォールモデル図

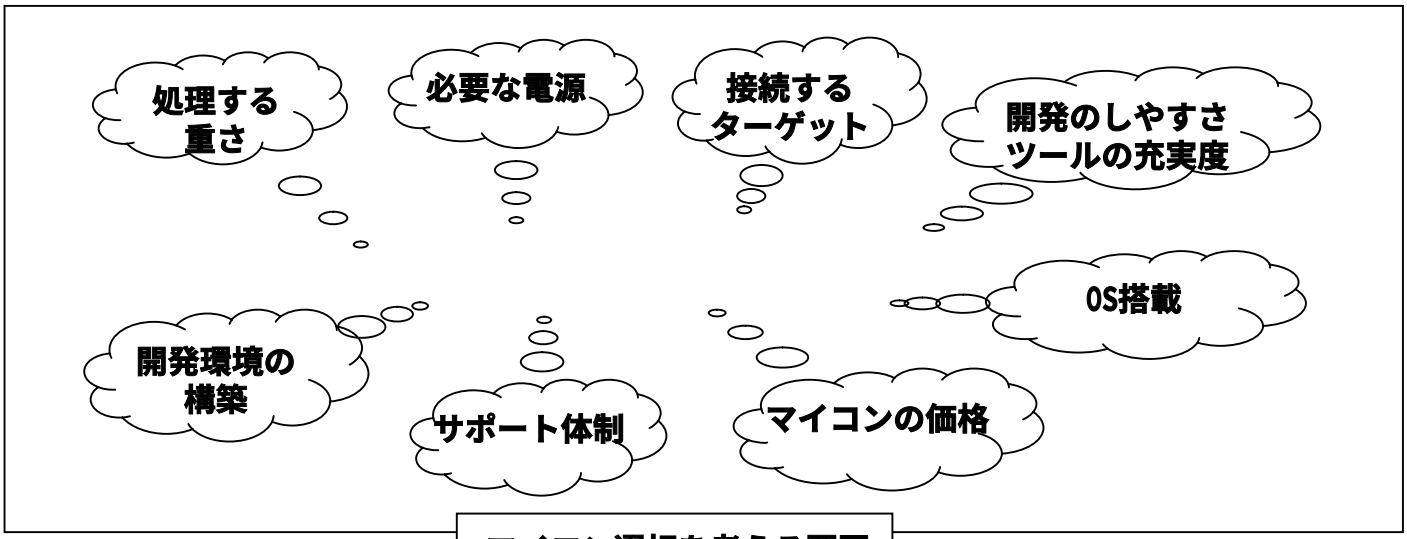


自分で使える道具を増やす

開発前に考えること(2)

Q.資源の見積もりが難しいです

マイコンには種類が豊富で選択に迷うこともしばしばあると思います。ポイントとしては
・必要とする電源(1.8V～5.5V)
・処理の重さ(8/16/32ビット)
・扱う周辺機能
・用意されているツール(本書で扱っているApplilet2等の便利なツールがあるか?)
おおまかには上記を考えればOKです。
まず、処理の重さを考え8/16/32ビットを選択します。開発時には周辺機能の豊富な、ROM容量など大きめのマイコンを使います。開発が進んだ段階で必要ない機能を見極めてROM容量/周辺機能を絞ったマイコンに置き換えればよいのです。



マイコン選択を考える要因

ワンポイント

開発環境の構築について
「MINICUBE2」http://www.necel.com/micro/ja/development/asia/minicube2/minicube2.html
最も簡単に弊社のマイクロコントローラ開発が始められます。また、このMINICUBE2に接続可能なオプションのCPUボードは、8～32ビットマイクロコントローラの代表的なラインアップから1つ選択されています。CPUボードは入手性も優れていますのでマイコン選択に迷ったら、これらのボードを入手して考えるとよいでしょう。

設計時に考えること

設計時に考えること

[設計は単純にする]

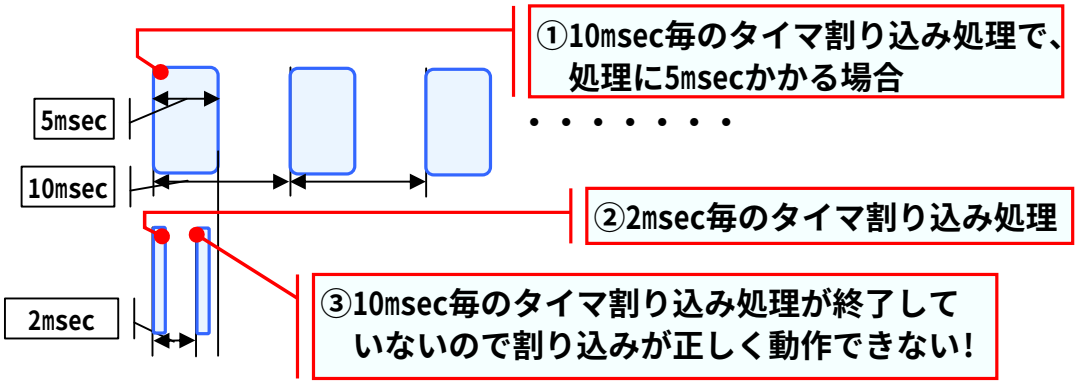
例えばアルゴリズムで回避できるなら、多重割り込みは使わない方がよいです。また、例外処理など後から考慮できるように、なるべく単純な設計にすべきです。仕様の変化に耐えられるように解りやすい設計を心がけてください。

[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。

[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。



ワンポイント

プログラムの階層構造について

プログラムを作る際は階層構造にするべきです。OSI参照モデルまで細かくする必要はありませんが、少なくとも物理層(ハードウエアにアクセスする層)とアプリケーション層は分けるべきです。そうすれば、作成したプログラムを他のマイコンに移植する際は物理層だけを入れ替えれば大丈夫になります。本書ではApplilet2を使用しているので、基本的な階層構造はできています。

コーディング時に考えること

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]

- 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;
    }
}
```

①このif命令を判断している時点ではP4.0=1の値だとここを通らない。

②上記でelseとなり、このif命令を判断する時にP4.0=0だとここを通らない。

③その結果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。

コーディング時に考えること

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。

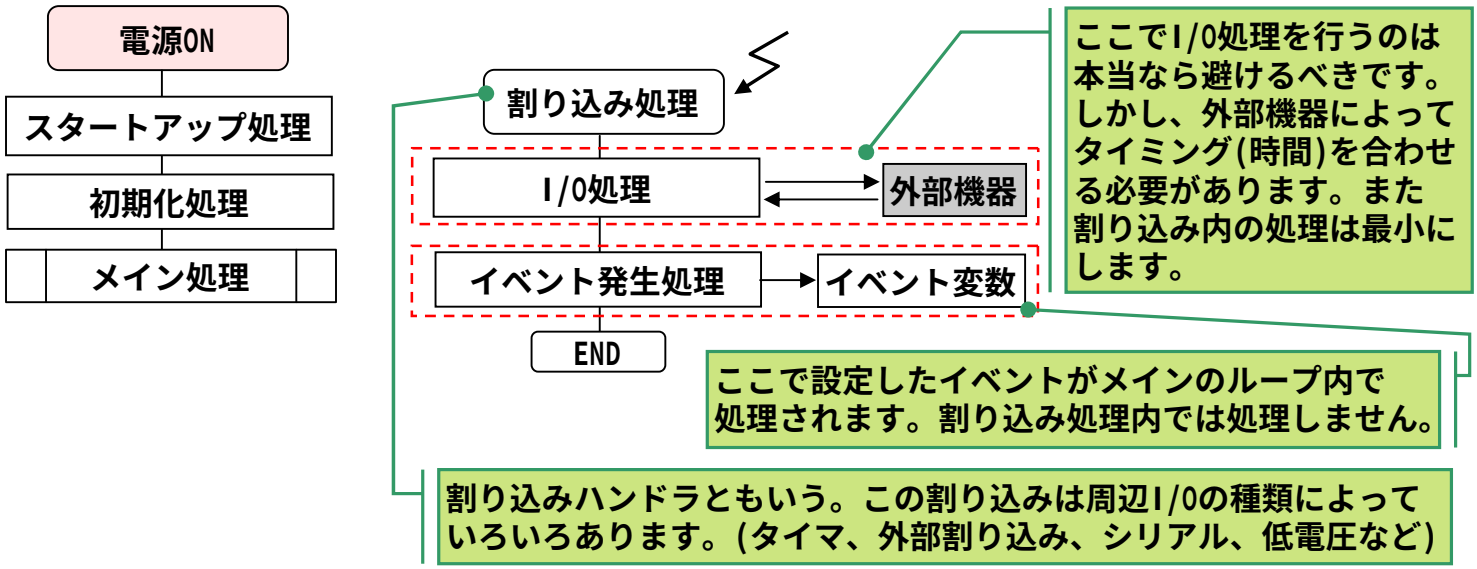
```
void set_function( UCHAR *para )
{
    UCHAR data = *para;
              :
              :
}

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

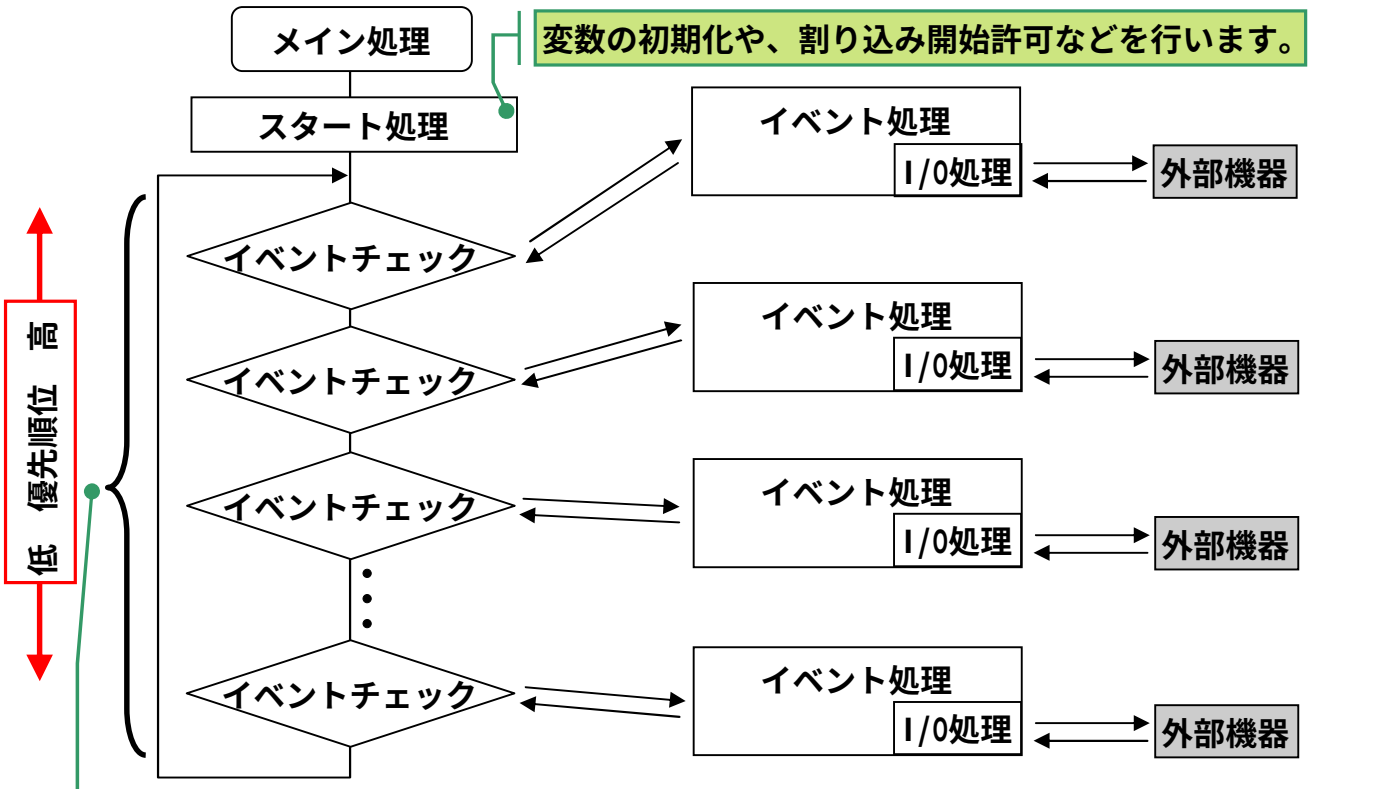
```
void set_function( UCHAR para )
{
    UCHAR data = para;
              :
              :
}

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。

基本的なプログラムの構成

ここでは組み込みプログラムの基本構成を説明します。もちろん、このガイドで製作するシステムも基本構成に沿ってプログラム作成しています。以下、フローチャートで全体構成を説明します。





メイン処理内でイベントのチェックを行います。また、イベントチェックの順番を入れ替えることで処理するイベントに優先順位をつけられます。最初にチェックするイベントを記述する程、優先順位が高く処理されます。

本書でのコーディング記述(1)

コーディング記述は設計と同じように重要です。世の中には設計書のないソフトウエアは数多く存在します。その時に頼りになるのはソースのみです。勿論、設計をせずにソフトウエアを書くことはお勧めしませんが、少なくとも1つのプロジェクトでコーディング記述は統一されなければなりません。ただ、統一されるのではなく、意志を持って統一することが重要です。本書では、どのような考えでコーディング記述を考えているか説明します。

1.マクロ名は大文字にします。
すぐに見分けがつくようにします。また#defineで定義したものは接頭語として”D_”を付加しています。ユーザー定義は[user_define.h]にまとめて記述しています。

```
例･･･ #define D_EV_INTP0          0x0000001    /* INTP0押下 */
      #define D_EV_SWCHECK        0x0000080    /* SW値のA/Dを読み込む */
      #define D_EV_UARTSEND       0x0008000    /* UART送出処理中 */
```

上記の定義はdefineを意味し、EVはイベント系の定義、最後に内容を示しています。

2.変数名の定義。
ローカル変数は全て小文字で構成します。また、グローバル変数には接頭語として”g”を付加します。またポインタ変数には”p”を付加します。構造体/配列変数には”t”をつけます。
これはApplilet2が生成するコードにも共通します。

```
例･･･ g1msecCounter    → グローバル変数
      gpUserBuffer     → グローバルポインタ変数
      gtLcdBuufer[32]  → グローバル配列変数
```

3.改行の定義。
中カッコ{}は独立して改行します。インデントと中カッコの位置で処理の単位を見易くするためです。現在の開発環境において、なるべく改行をつめる意味はなくなっています。

```
例･･･ if ( xx ) {                          if ( xx )
      }                  ──────→           {          → 改行します
                                           }

      for ( xx ) {                         for ( xx )
      }                  ──────→           {          → 改行します
                                           }
```

中カッコのインデントを揃えて、処理の単位を見易くしています。

4.関数の引数についての定義。
関数の引数には最後に”_”(アンダーバー)をつけます。これは、ローカル変数、グローバル変数引数を明確に区別するためです。また、引数に戻り値がある場合は必ずポインタ変数にします。
下記のように関数の戻り値の有無が一目で理解できると思います。

```
例･･･MD_STATUS i2c_write( USHORT dataadrs_, UCHAR *p_sadrs_, UINT length_ )
     dataadrs_ → 戻り値なし
     *p_sadrs_ → 戻り値あり
     length_   → 戻り値なし
```

本書でのコーディング記述(2)

5.関数の作り方について。
関数の入口、出口は1箇所にします。なぜでしょうか？
例･･･メモリリークをしにくくする

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

関数の引数によって、メモリリーク(メモリ解放のし忘れ)が発生している。

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

関数途中で終了することがないので、領域確保した場合は、必ず領域解放を行う処理になっている。

例･･･デバッグしやすくする。下記のように値を返す関数をデバッグすることを考えます。

```
int function( void )
{
    switch ( xx )
    {
        case xx :
            return 1;
            break;
        case xxx :
            return 2;
            break;
        default :
            return 3;
            break;
    }
    retrun 0;
```

関数の戻りを確認するには、それぞれの箇所にブレークを張る必要がある。

```
int function( void )
{
    int ret = 0;
    switch ( xx )
    {
        case xx :
            ret = 1;
            break;
        case xxx :
            ret =2;
            break;
        default :
            ret =3;
            break;
    }
    retrun ret;
```

関数の戻りを確認するには、1箇所のみにブレークを張ればよくなった。

上記の例は少し極端かもしれませんが、とても有効な方法です。また、どうしても関数途中で抜け出す場合は、関数分割や、アルゴリズム変更などで、入口/出口を1つにすることを勧めます。

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, 0x11, 0x11, 0x11, 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 */
```

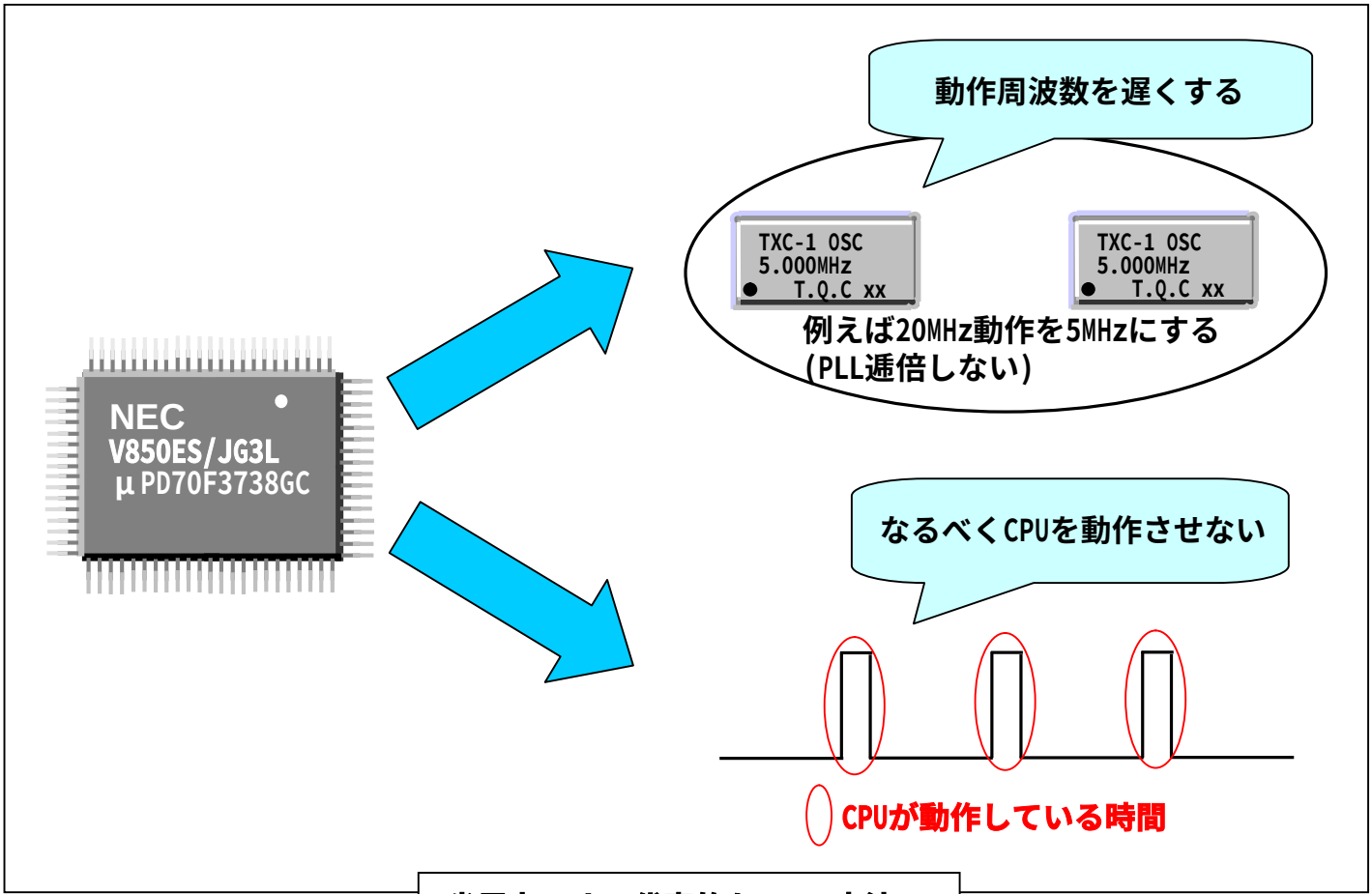
省電力処理

省電力処理とは

マイコンはいつも100%で処理をしていません。ほとんどが割り込み待ち、イベント待ちの処理です。100%連続で使う代表的な処理は、画像の圧縮/展開などです。本書のプログラムでは、そのような処理はありませんので、マイコンの実働時間は非常に短いです。そこで空いてる時間(割り込み待ち、イベント待ち)はスタンバイ機能でHALTモードにすると消費電力を大幅に減らすことができます。このようなスタンバイ処理は、製品化する上では一般的と思ってください。

省電力処理で注意しなければならない点があります。
・HALTモードから復帰するには数マイクロ秒～数ミリ秒かかる場合がある。
・HALTモードに入るタイミングを重要な処理の前にならないように考慮する。

省電力処理は電池を使ったターゲットシステムでは避けて通れません。このプログラムを作成するには構築するシステムの理解を深めることが大切です。マイコンをHALTモードにすれば消費電力を下げられますが、マイコンだけでなく動作させる周辺機能にも注意してください。消費される電力を見極めることが重要です。



省電力にする代表的な2つの方法

最後に

マイコンは組み込みシステムで使われます。この組み込みシステムを作るのにはハードウエア、ソフトウエア両方の知識が不可欠です。本書に掲載しているシステムでさえ、複雑なものになっていると思われた方も多いのではないでしょうか？

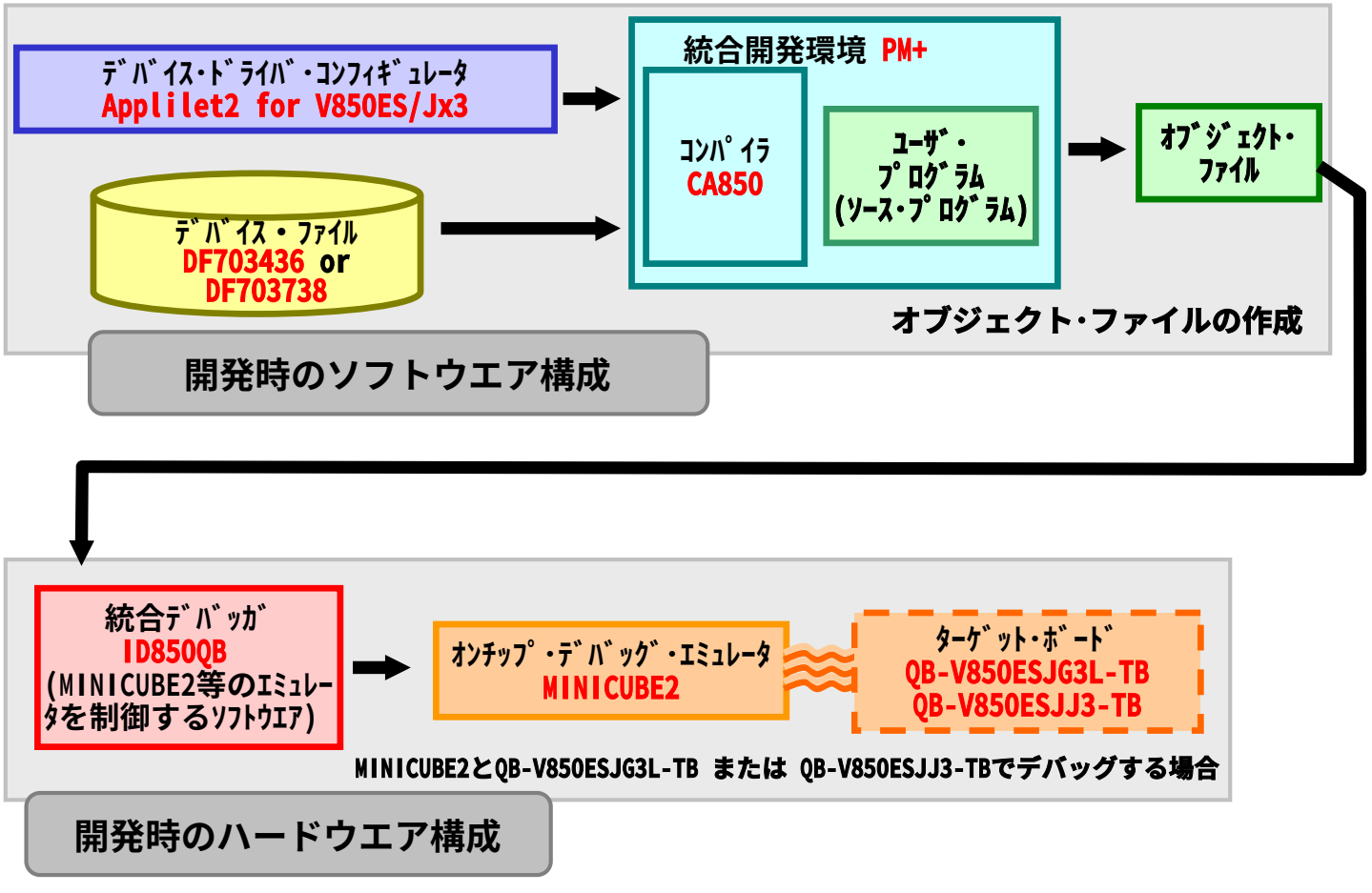
これらの知識を深めるのにはどうすればいいのか？近道はありません。ただ、実践する(実際に体験する)とより理解が早まります。是非、実際に試して知識を深めて頂ければ幸いに思います。

参考資料

V850開発環境のダウンロード

http://www.necel.com/micro/ja/freesoft/v850/jx3/
下記ファイルがダウンロードできます。
・CA850(統合開発環境PM+を含みます)
・V850ES/Jx3用 Applilet2
・V850ES/Jx3用 デバイス・ファイル

https://www5.necel.com/micro/tool_reg/0dsListTool.do?code=460&lang=ja
下記ファイルがダウンロードできます。
・ID850QB
・QB-Programmer



MINICUBE2の情報について

http://www.necel.com/micro/ja/development/asia/minicube2/minicube2.html
MINICUBE2の情報については、このURLにまとめられています。最新ユーザーズマニュアル、ファームウエア、また各種CPUボード(ターゲット･ボード)のプログラムも掲載しています。
MINICUBE2はWEBを通して情報提供しますのでチェックすることをお勧めします。

参考資料

マイコンについての基礎が理解できます。

http://www.necel.com/micro/ja/campaign/allflash-zemi/

マイコンの基礎知識を解りやすく説明します。講座形式で1回ごとにテーマを決めて説明します。

- 第1回　マイコン
- 第2回　CPU
- 第3回　メモリ
- 第4回　ポート
- 第5回　タイマ
- 第6回　A/Dコンバータ
- 第7回　シリアル・インタフェース
- 第8回　割り込み

マイコンについて基礎～中級の知識が学習できます。

http://www.necel.com/micro/ja/eLearning/

学習形式でマイコンの基本が理解できます。「基礎編」「初級編」「中級編」で構成されております。

「基礎編」

- 基礎編はマイコンを学習するうえで必要となる２進数や論理回路等の基礎知識を習得する事ができるコースです。学習効果を高めるために各章ごとに問題を用意しています。

「初期編」

- 初級編はマイコンをよく知らないかた向けの入門編です。小ピンマイコン（78K0S/Kx1+）を題材に簡単ツールを使ってプログラム作成ができます。

「中級編」

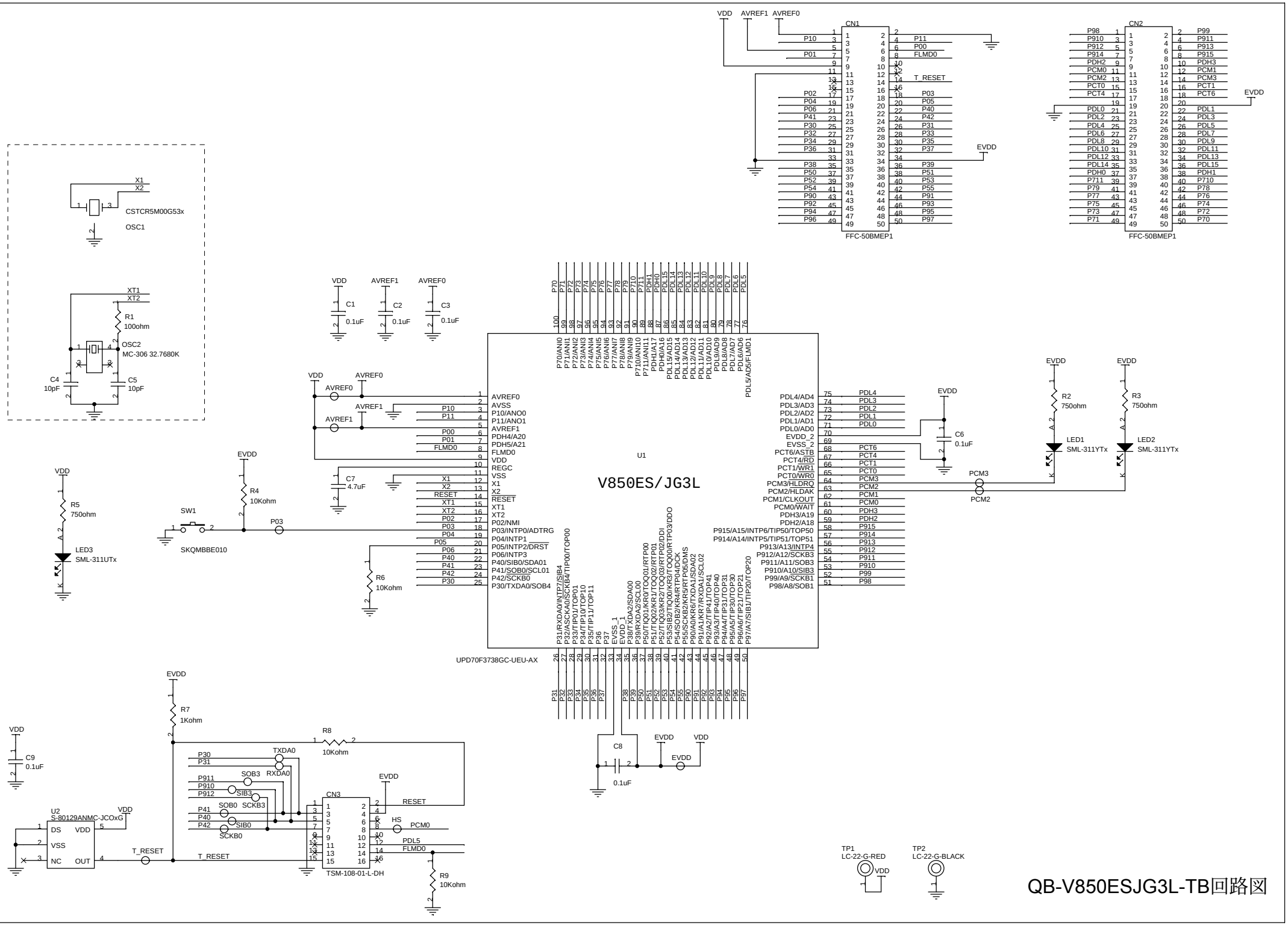
- 中級編はマイコンをより詳しく知りたいかた向けのコースです。78K0/KE2マイコンを題材に、搭載されている周辺機能をどのように使うか学習することができます。章ごとに簡単なプログラム実習がついています。

川崎、名古屋、大阪では「マイコンセミナ」も開催されております。無料セミナもありますので、興味のある方は下記URLをご参照ください。

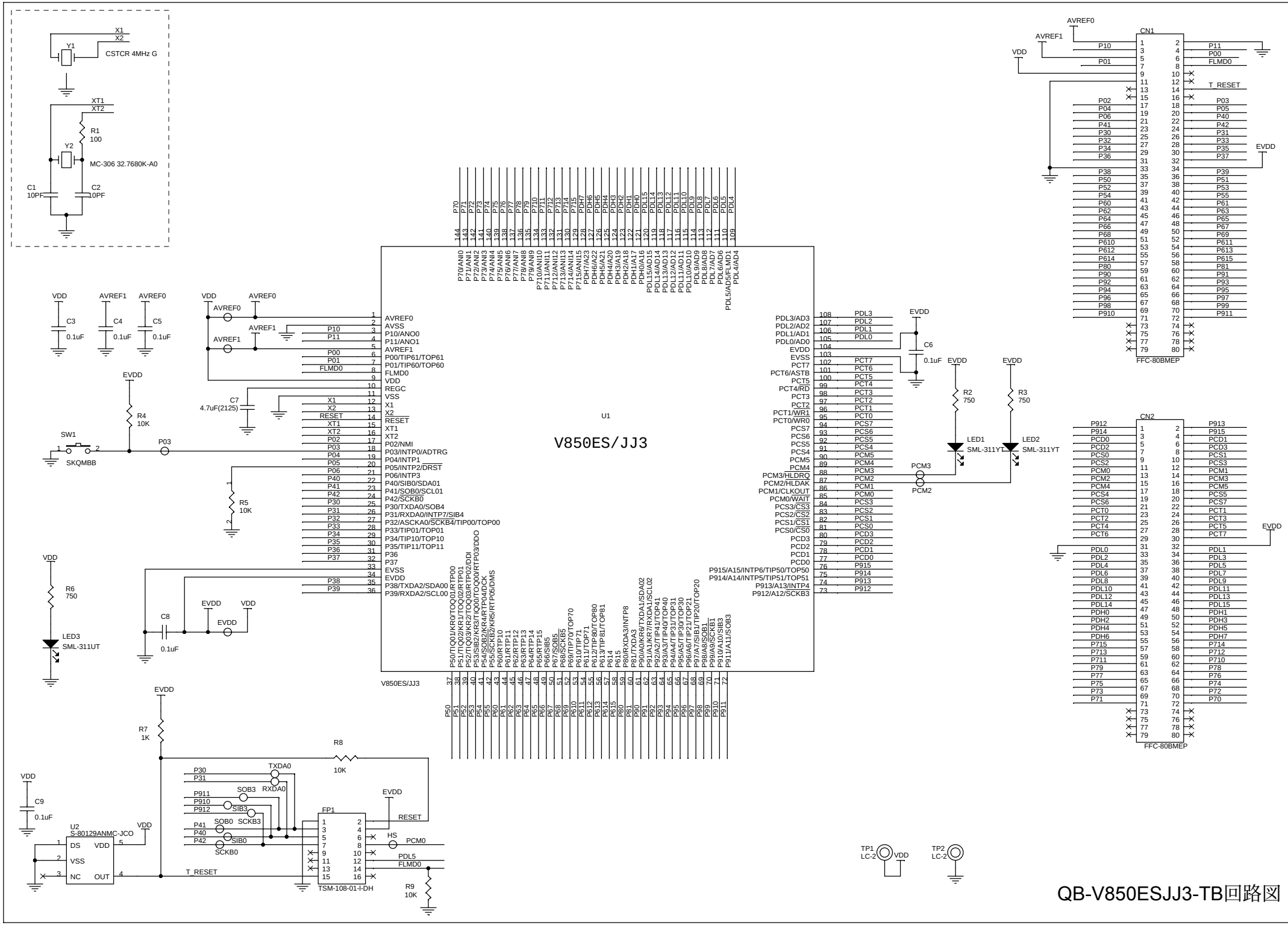
http://www.necel.com/seminar/ja/

またFAQでは、よくある質問などマイコン、開発ツール、またマイコン以外のNECエレクトロニクスの製品の事なら何でも掲載されています。困った時は、FAQ。

http://www.necel.com/ja/faq/



QB-V850ESJG3L-TB回路図


X1
X2
Y1
CSTCR 4MHz G
XT1
XT2
R1
100
Y2
MC-306 32.7680K-A0
C1
10PF
C2
10PF
U1
V850ES/JJ3
V850ES/JJ3
C3
0.1uF
C4
0.1uF
C5
0.1uF
C6
0.1uF
C7
4.7uF(2125)
R2
750
R3
750
R4
10K
R5
10K
R6
750
R7
1K
R8
10K
R9
10K
SW1
SKQMBB
LED1
SML-311YT
LED2
SML-311YT
LED3
SML-311UT
C8
0.1uF
C9
0.1uF
U2
S-80129ANMC-JCO
FP1
TSM-108-01-I-DH
CN1
FFC-80BMEP
CN2
FFC-80BMEP
TP1
LC-2
TP2
LC-2
QB-V850ESJJ3-TB回路図