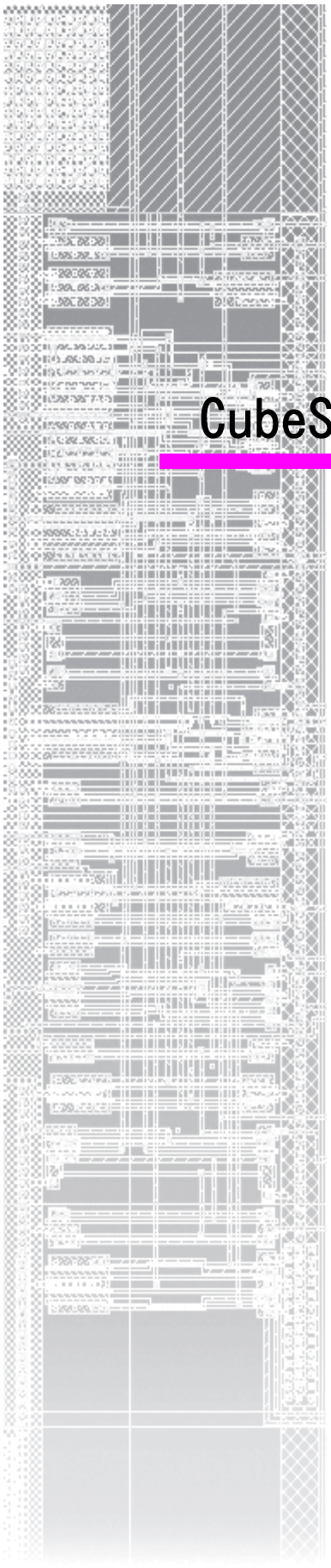


CubeSuite+ 版 RX シリアルデバッガ

取扱説明書

ルネサス エレクトロニクス
www.renesas.com

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して、お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
2. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
3. 本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害に関し、当社は、何らの責任を負うものではありません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を改造、改変、複製等しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、
   各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通用信号機器、
   防災・防犯装置、各種安全装置等

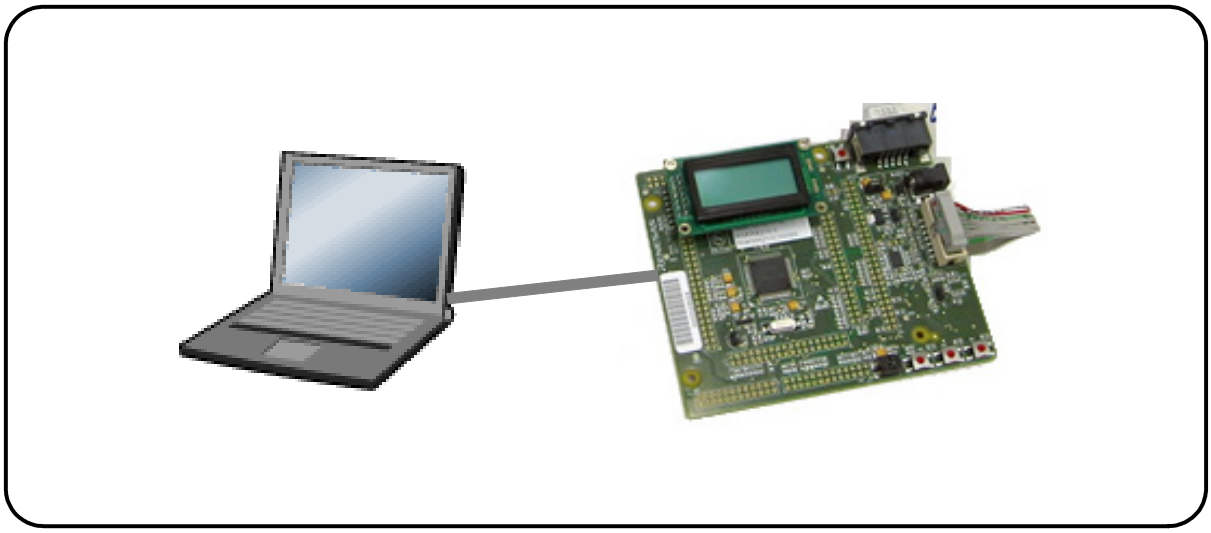
   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（原子力制御システム、軍事機器等）に使用されることを意図しておらず、使用することはできません。 たとえ、意図しない用途に当社製品を使用したことによりお客様または第三者に損害が生じても、当社は一切その責任を負いません。 なお、ご不明点がある場合は、当社営業にお問い合わせください。
6. 当社製品をご使用の際は、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他の保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 本資料に記載されている当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途に使用しないでください。当社製品または技術を輸出する場合は、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。
10. お客様の転売等により、本ご注意書き記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は何らの責任も負わず、お客様にてご負担して頂きますのでご了承ください。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注１において定義された当社の開発、製造製品をいいます。

はじめに

この取扱説明書は、“CubeSuite+版RXシリアルデバッガ”用に作成されたものです。RXシリアルデバッガはターゲット・CPUの内蔵フラッシュメモリに書き込み、シリアルポート（COMポート）経由でCubeSuite+からターゲット・CPUのデバッグを行うデバッグ・ツールです。E1/E20エミュレータのような機器を必要としませんが､E1/E20エミュレータの持つデバッグ機能の約60%を提供することが可能です。



この取扱説明書は、上段がスライド部、下段が解説部の構成となっています。解説部のマークは、スライドの説明には直接関係がありませんが、コラム的な内容としてまとめてあります。

■ RXシリアルデバッガ使用に際しての注意事項

RXシリアルデバッガは無償・評価版のソフトウェアツールです。RXファミリをご購入前に製品の機能や性能を評価するために無償でご利用いただけます。従って、RX シリアルデバッガは以下サービス提供の対象外です。評価版ソフトウェアツールをご使用の場合はコンタクトセンタへお問合せいただいてもお答えできない場合がございます

・評価版ソフトウェアツールに対する不具合改修
・技術的なお問い合わせに対するサポート
・リビジョンアップ情報などの案内メール送信

目次

第 1 章

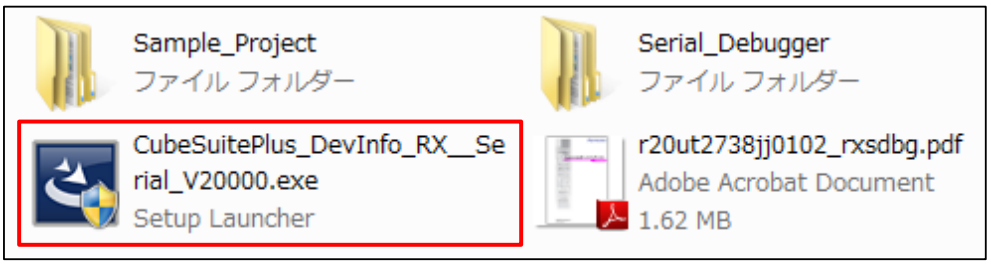
RX シリアルデバッガ用アドインツール

1.1　アドインツールのインストール

アドインツールのインストール

CubeSuite+ V2.02.00以降をインストール後(下記説明文を参照)

RXシリアルデバッガの展開フォルダ



ダブルクリックしてインストールを実施

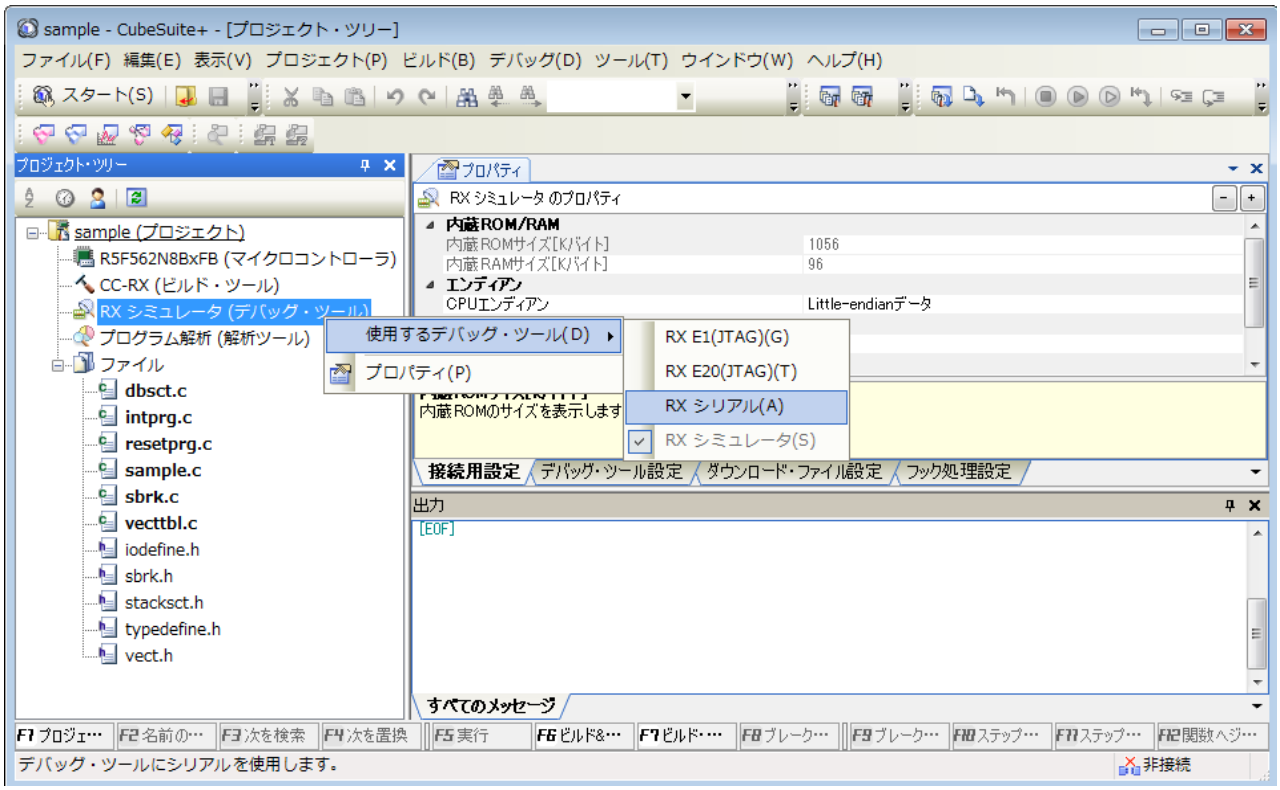
ガイダンスに従って
RXシリアルデバッガ用
アドインツールのインストールを
実施してください

アドインツールのインストール

CubeSuite+用のRXシリアルデバッガを使うに当たっては、インストール済みのCubeSuite+ V2.02.00以降に対して、RXシリアルデバッガ用のアドインツールをインストールする必要があります。Webよりダウンロードし、本説明資料が存在するフォルダ内にある「CubeSuitePlus_DevInfo_RX__Serial_V20000.exe」をダブルクリックし、RXシリアルデバッガ用のアドインツールをインストールしてください。

インストールの完了確認作業（必要であれば）

正しくインストールされれば、デバッグ・ツールでRXシリアルが選択可能となる



インストールの完了確認作業（必要であれば）

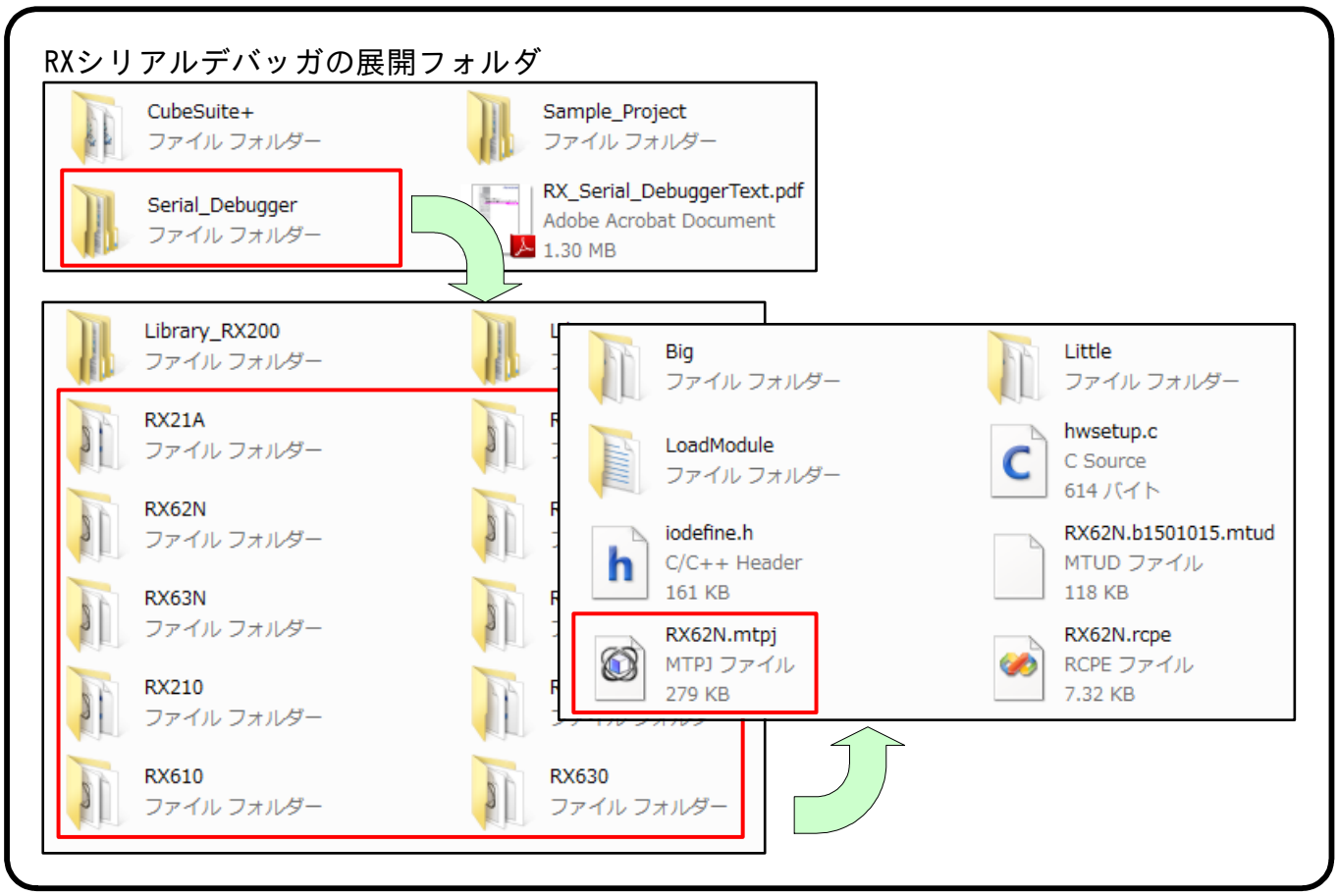
インストールの完了確認作業は特に必要ではありませんが、正しくインストールされたかどうかはRX600シリーズ、またはRX200シリーズのCubeSuite+用プロジェクトを開くことで分かります。特定のプロジェクトが開かれた状態でプロジェクト・ツリーのデバッグ・ルーツを右クリックします。表示されたポップアップメニューにおける「使用するデバッグ・ツール」に「RX シリアル」が表示されれば、正しくインストールが行われています。

第 2 章

RX シリアルデバッガの作成

2.1　プロジェクトの立ち上げ

プロジェクトの立ち上げ



プロジェクトの立ち上げ

ターゲットCPU内蔵のフラッシュメモリに書き込むRXシリアルデバッガは、使用用途に合わせた設定を行った後にビルドしてロードモジュールを作成する必要があります。RXシリアルデバッガは各グループ毎に準備されていますから、お使いのグループに対応したビルド用のプロジェクトを立ち上げてください。

Webよりダウンロードし、本説明資料が存在するフォルダ内にある「Serial_Debugger」フォルダ内に各グループ毎のフォルダがありますから、その中にあるCubeSuite+用のプロジェクト・アイコンをダブルクリックしてください。これでRXシリアルデバッガ用のプロジェクトが立ち上がります。

2.2　HardwareSetup関数の修正

2.2.1　RX610とRX62*系

HardwareSetup関数の修正（ソースファイル名：hwsetup.c）

```
 1: #include ”iodefine.h”
 2:
 3: #define  BRR    12     // Bitrate Register Value
 4:
 5: const unsigned char Brr = BRR;
 6:
 7: void HardwareSetup(void)
 8: {
 9:     SYSTEM.SCKCR.LONG = 0x00020100;      // If XTAL=12MHz Then
10:                                         // ICLK=96MHz,BCLK=24MHz,PCLK=48MHz
11: // If Used SCI1 and Used TxD1-B,RxD1-B
12: //  IOPORT.PFFSCI.BIT.SCI1S = 1;         // Use TxD1-B,RxD1-B
13:
14: // If Used SCI2 and Used TxD2-B,RxD2-B
15: //  IOPORT.PFFSCI.BIT.SCI2S = 1;         // Use TxD2-B,RxD2-B
16:
17: // If Used SCI3 and Used TxD3-B,RxD3-B
18: //  IOPORT.PFFSCI.BIT.SCI3S = 1;         // Use TxD3-B,RxD3-B
19:
20: // If Used SCI6 and Used TxD6-B,RxD6-B
21: //  IOPORT.PFFSCI.BIT.SCI6S = 1;         // Use TxD6-B,RxD6-B
22: }
```

HardwareSetup関数の修正

1.　システムクロックコントロールレジスタ（SCKCR）の設定

RXシリアルデバッガは内蔵シリアルコミュニケーションインタフェース（SCI）を利用してCubeSuite+とインタフェースを行います。このため、内蔵周辺機能の動作クロックを決定しているクロック発生回路のシステムクロックコントロールレジスタ（SCKCR）はRXシリアルデバッガ起動時に適切な値に設定する必要があります。

SCKCRの設定に関しては2-4頁を参照ください。

2.　TxD端子、 RxD端子の選択

RXシリアルデバッガでは内蔵されている全てのSCIチャネルがCubeSuite+とのインタフェースに利用可能です。ただし、特定のSCIチャネルはTxD端子とRxD端子が複数端子用意されているものがあります。 デフォルトの端子選択では利用できない端子を使われる場合、 TxD端子とRxD端子の選択を行う必要があります。

TxD端子、 RxD端子の選択に関しては2-5頁を参照ください。

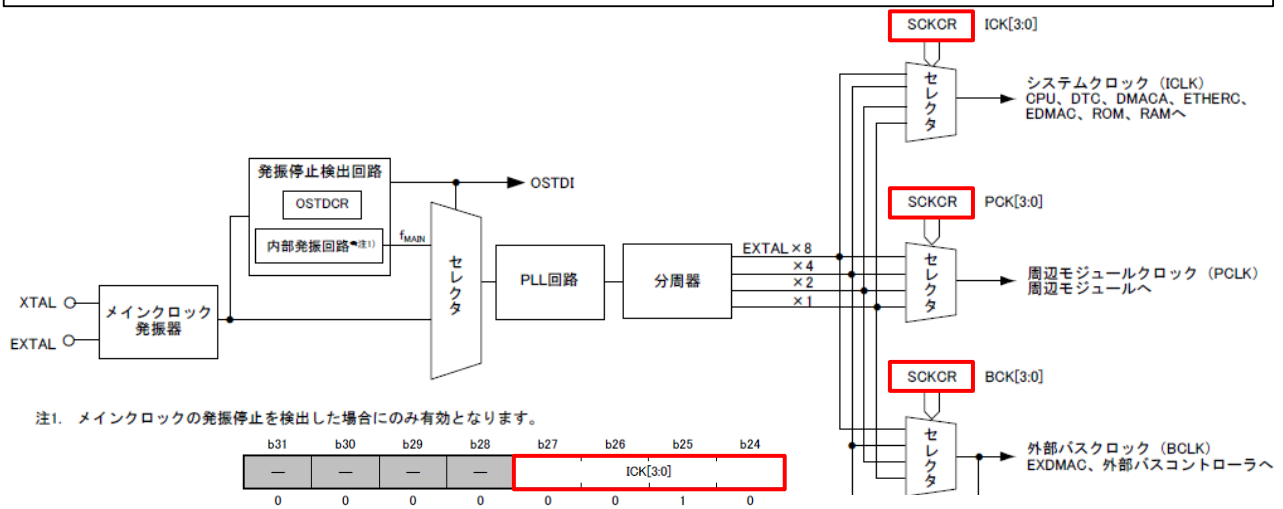
3.　ビットレートの設定

RX シリアルデバッガは CubeSuite+ とのインタフェースに利用する SCI のビットレート（回線速度）として、115200bps/57600bps/38400bps/19200bps が利用可能です。このため、選択したビットレートが得られるよう SCI のビットレートレジスタ（BRR）に適切な値に設定する必要があります。

ビットレートの設定に関しては2-6頁を参照ください。

システムクロックコントロールレジスタ（SCKCR）の設定

```
 9:      SYSTEM.SCKCR.LONG = 0x00020100;      // If XTAL=12MHz Then
10:                                          // ICLK=96MHz,BCLK=24MHz,PCLK=48MHz
```



注1.　メインクロックの発振停止を検出した場合にのみ有効となります。

b31	b30	b29	b28	b27	b26	b25	b24
—	—	—	—	ICK[3:0]			
0	0	0	0	0	0	1	0

b23	b22	b21	b20	b19	b18	b17	b16
PSTOP1	PSTOP0	—	—	BCK[3:0]			
0	0	0	0	0	0	1	0

b15	b14	b13	b12	b11	b10	b9	b8
—	—	—	—	PCK[3:0]			
0	0	0	0	0	0	1	0

b7	b6	b5	b4	b3	b2	b1	b0
—	—	—	—	—	—	—	—
0	0	0	0	0	0	0	0

システムクロックコントロールレジスタ（SCKCR）の設定

RX610とRX62*系はXTAL/EXTAL端子に接続したメインクロックをSCKCRで逓倍し、CPU（ICLK）、周辺（PCLK）、バス（BCLK）に与えます。RXシリアルデバッガでは内蔵SCIを使ってCubeSuite+とインタフェースを取るため、起動時にSCKCRの初期化が必要となります。

そこで2-3頁のリストにおける9行目の設定値を使用されるCPUボードに合わせて適切な設定値に変更してください。なお、SCKCRの設定値には各グループ毎に設定の注意事項がありますから、それらは対応したRXグループのハードウェア・マニュアルを参照ください。

【RXシリアルデバッガ特有の注意事項】

RX シリアルデバッガでは内蔵フラッシュメモリに対してデバッグ対象のプログラムのダウンロードを行います。ダウンロード時のフラッシュメモリに対する書き込み遅延を防ぐためにも周辺モジュールに対するPCLKは出来るだけ最速の設定を行ってください。

TxD端子、 RxD端子の選択

ポートファンクションレジスタ F（PFFSCI）

	b7	b6	b5	b4	b3	b2	b1	b0
	–	SCI6S	–	–	SCI3S	SCI2S	SCI1S	–
リセット後の値	0	0	0	0	0	0	0	0

ビット	シンボル	ビット名	機能	R/W
b0	–	予約ビット	読むと"0"が読めます。書く場合、"0"としてください	R/W
b1	SCI1S	SCI1端子選択ビット	0：P30をRxD1-A端子として設定 P27をSCK1-A端子として設定 P26をTxD1-A端子として設定 1：PF2をRxD1-B端子として設定 PF1をSCK1-B端子として設定 PF0をTxD1-B端子として設定	R/W
b2	SCI2S	SCI2端子選択ビット	0：P12をRxD2-A端 P11をSCK2-A端 P13をTxD2-A端 1：P52をRxD2-B端 P51をSCK2-B端 P50をTxD2-B端	R/W
b3	SCI3S	SCI3端子選択ビット	0：P16をRxD3-A端 P15をSCK3-A端 P17をTxD3-A端 1：P25をRxD3-B端 P24をSCK3-B端 P23をTxD3-B端	
b5-b4	–	予約ビット	読むと"0"が読めます	
b6	SCI6S	SCI6端子選択ビット	0：P01をRxD6-A端 P02をSCK6-A端 P00をTxD6-A端 1：P33をRxD6-B端 P34をSCK6-B端 P32をTxD6-B端	
b7	–	予約ビット	読むと"0"が読めます。書く場合、"0"としてください	R/W

```
11: // If Used SCI1 and Used TxD1-B,RxD1-B
12: //  IOPORT.PFFSCI.BIT.SCI1S = 1;
13:
14: // If Used SCI2 and Used TxD2-B,RxD2-B
15: //  IOPORT.PFFSCI.BIT.SCI2S = 1;
16:
17: // If Used SCI3 and Used TxD3-B,RxD3-B
18: //  IOPORT.PFFSCI.BIT.SCI3S = 1;
19:
20: // If Used SCI6 and Used TxD6-B,RxD6-B
21: //  IOPORT.PFFSCI.BIT.SCI6S = 1;
```

TxD端子、RxD端子の選択

RXシリアルデバッガでは内蔵されている全てのSCIチャネルがCubeSuite+とのインタフェースに利用可能です。ただし、特定のSCIチャネルはTxD端子とRxD端子が複数端子用意されているものがあります。 もし、 利用されるSCIチャネルのTxD端子とRxD端子がデフォルトの端子機能では利用できない場合、上記のポートファンクションルレジスタを設定し、TxD端子とRxD端子を利用可能とする必要があります。

そこで2-3頁のリストにおけるHardwareSetup関数では、ポートファンクションルレジスタで指定可能な設定を11行目から21行目のコメントで記載しています。必要であればコメントを解除して、TxD端子とRxD端子を利用可能としてください。なお、ポートファンクションルレジスタは各グループ毎に異なりますから、それらは対応したRXグループのハードウェア・マニュアルを参照ください。

ビットレートの設定

ビットレートレジスタ（BRR）

	b7	b6	b5	b4	b3	b2	b1	b0
リセット後の値	1	1	1	1	1	1	1	1

BRRレジスタの設定値

$$N = \frac{PCLK \times 10^6}{64 \times 2^{2n-1} \times B} - 1$$

```
3: #define BRR 12 // Bitrate Register Value
```

B　：ビットレート（bps）
N　：ボーレートジェネレータのBRRの設定値（0≦N≦255）
PCLK：動作周波数（MHz）
n　：0

ビットレート（bps）	動作周波数PCLK（MHz）											
	25			30			33			50		
	n	N	誤差（%）	n	N	誤差（%）	n	N	誤差（%）	n	N	誤差（%）
110	3	110	-0.02	3	132	0.13	3	145	0.33	3	221	-0.02
150	3	80	0.47	3	97	-0.35	3	106	0.39	3	162	-0.15
300	2	162	-0.15	2	194	0.16	2	214	-0.07	3	80	0.47
600	2	80	0.47	2	97	-0.35	2	106	0.39	2	162	-0.15
1200	1	162	-0.15	1	194	0.16	1	214	-0.07	2	80	0.47
2400	1	80	0.47	1	97	-0.35	1	106	0.39	1	162	-0.15
4800	0	162	-0.15	0	194	0.16	0	214	-0.07	1	80	0.47
9600	0	80	0.47	0	97	-0.35	0	106	0.39	1	40	-0.77
19200	0	40	-0.76	0	48	-0.35	0	53	-0.54	0	80	0.47
31250	0	24	0.00	0	29	0	0	32	0	0	49	0.00
38400	0	19	1.73	0	23	1.73	0	26	-0.54	0	40	-0.77

ビットレートの設定

RX シリアルデバッガは CubeSuite+ とのインタフェースに利用する SCI のビットレート（回線速度）として、115200bps/57600bps/38400bps/19200bps が利用可能です。このため、選択したビットレートが得られるよう SCI のビットレートレジスタ（BRR）に適切な値に設定する必要があります。BRRへの設定値は上記の計算式（得たいビットレートとPCLKの動作周波数）で求めることができます。また、代表的な動作周波数であれば、ハードウェアマニュアルに上記のような設定表が記載されています。ただし、115200bps や 57600bps 等の設定値は記載されていない場合があります。そのような場合は計算式を使って適切な設定値を算出してください。

算出後は2-3頁のリストの3行目にあるBRRマクロの値を算出した値に変更してください。なお、2-3頁のリストの3行目はPCLK=48MHzから115200bpsを得るときの値です。

2.2.2 RX63*とRX2**系
HardwareSetup関数の修正（ソースファイル名：hwsetup.c）

```
 1: #include ”iodefine.h”
 2:
 3: #define  BRR    10           // Bitrate Register Value
 4: #define  CLOCK  160000000    // Input Clock value at Frequency divider
 5:
 6: const unsigned char Brr = BRR;
 7: const unsigned long Clock = CLOCK;
 8:
 9: void HardwareSetup(void)
10: {
11: int i;
12: // If Used 10MHz Main Clock
13:     SYSTEM.PRCR.WORD = 0xA503;        // Protect Disable
 :
23:     SYSTEM.SCKCR.LONG = 0x21021222;  // FCLK=40MHz,ICLK=80MHz,BCLK=40MHz
24:                                      // PCLKA=80MHz,PCLKB=40MHz,PCLKD=40MHz
 :
28:     SYSTEM.PRCR.WORD = 0xA500;        // Protect Enable
 :
36: // If Used SCI1
37:     PORTD.PMR.BIT.B5 = 1;             // Use PD5 for RXD1
38: //  PORT9.PMR.BIT.B3 = 1;             // Use P93 for RXD1
39:     PORTD.PMR.BIT.B3 = 1;             // Use PD3 for TXD1
40: //  PORT9.PMR.BIT.B4 = 1;             // Use P94 for TXD1
 :
45: }
```

HardwareSetup関数の修正

1. システムクロックの設定

RXシリアルデバッガは内蔵シリアルコミュニケーションインタフェース（SCI）を利用してCubeSuite+とインタフェースを行います。このため、内蔵周辺機能の動作クロックを決定しているクロック発生回路やシステムクロック関係のレジスタはRXシリアルデバッガ起動時に適切な値に設定する必要があります。

クロック発生回路やシステムクロックの設定に関しては2-8頁を参照ください。

2. TxD端子、 RxD端子の設定

RXシリアルデバッガでは内蔵されている全てのSCIチャネルがCubeSuite+とのインタフェースに利用可能です。ただし、TxD端子とRxD端子は初期状態、 I/Oポートとして動作しています。 このため、 使用するTxD端子とRxD端子に対応したポートモードレジスタ （PMR） の設定を行う必要があります。

TxD端子、 RxD端子の設定に関しては2-9頁を参照ください。

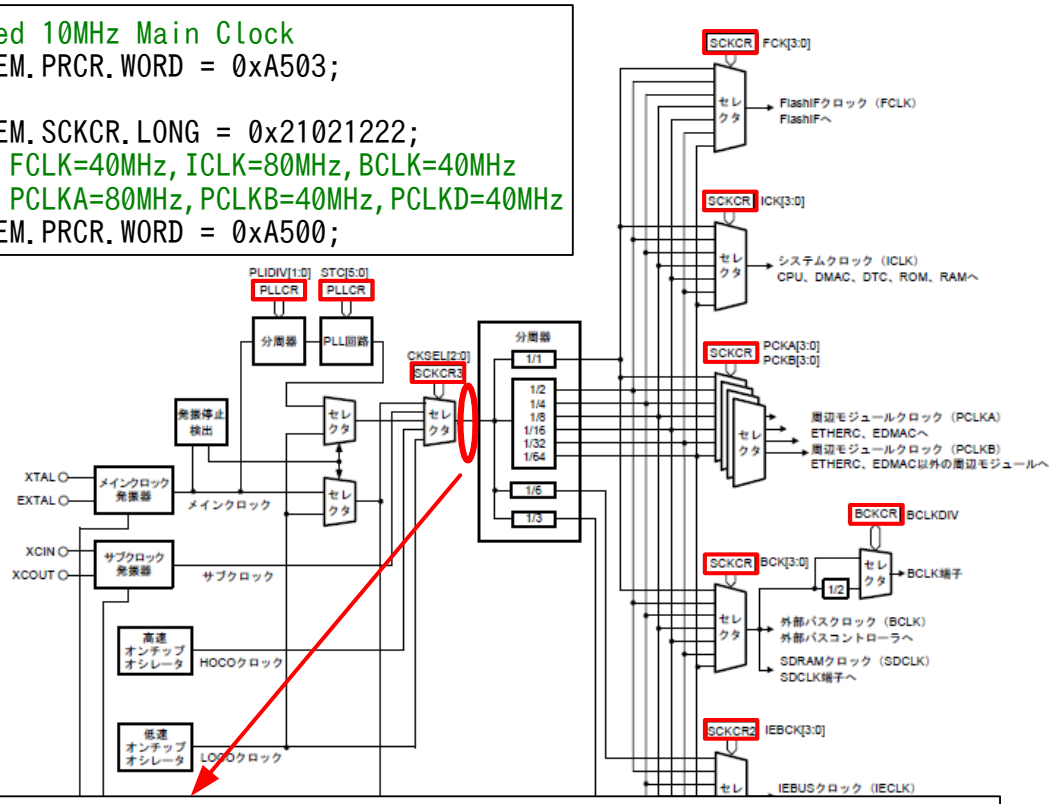
3. ビットレートの設定

RX シリアルデバッガは CubeSuite+ とのインタフェースに利用する SCI のビットレート（回線速度）として、115200bps/57600bps/38400bps/19200bps が利用可能です。このため、選択したビットレートが得られるよう SCI のビットレートレジスタ（BRR）に適切な値に設定する必要があります。

ビットレートの設定に関しては2-10頁を参照ください。

システムクロックの設定

```
12: // If Used 10MHz Main Clock
13:     SYSTEM.PRCR.WORD = 0xA503;
 :
23:     SYSTEM.SCKCR.LONG = 0x21021222;
24:       // FCLK=40MHz,ICLK=80MHz,BCLK=40MHz
 :        // PCLKA=80MHz,PCLKB=40MHz,PCLKD=40MHz
28:     SYSTEM.PRCR.WORD = 0xA500;
```




```
4: #define  CLOCK  160000000    // Input Clock value at Frequency divider
```

システムクロックの設定

RX63*系とRX2**系はシステムクロックとしてXTAL/EXTAL端子、高速オンチップオシレータ（HOCO）等が利用可能です。また、SCKCR3で選択したシステムクロックをSCKCRで分周し、フラッシュメモリ（FCLK）、CPU（ICLK）、周辺（PCLK*）、バス（BCLK）等に与えます。RXシリアルデバッガでは内蔵SCIを使ってCubeSuite+とインタフェースを取るため、起動時にクロック発生回路関係のレジスタとSCKCRの初期化が必要となります。

そこで2-7頁のリストにおける13行目から28行目の設定を使用されるCPUボードに合わせて適切なシステムクロックの設定に変更してください。なお、クロック発生回路関係のレジスタとSCKCRの設定値には各グループ毎に設定の注意事項がありますから、それらは対応したRXグループのハードウェア・マニュアルを参照ください。

また、RX63*系とRX2**系の場合、2-7頁のリストにおける4行目のCLOCKマクロの値を分周器に入力したクロックの値に変更する必要があります。上記の例は分周器に160MHzを入力した場合の例となっています。

【RXシリアルデバッガ特有の注意事項】

RX シリアルデバッガでは内蔵フラッシュメモリに対してデバッグ対象のプログラムのダウンロードを行います。ダウンロード時のフラッシュメモリに対する書き込み遅延を防ぐためにもフラッシュメモリに対するFCLKは出来るだけ最速の設定を行ってください。

TxD端子、 RxD端子の設定

```
36: // If Used SCI1
37:      PORTD.PMR.BIT.B5 = 1;          // Use PD5 for RXD1
38: //   PORT9.PMR.BIT.B3 = 1;          // Use P93 for RXD1
39:      PORTD.PMR.BIT.B3 = 1;          // Use PD3 for TXD1
40: //   PORT9.PMR.BIT.B4 = 1;          // Use P94 for TXD1
```

ポートモードレジスタ（PMR）

	b7	b6	b5	b4	b3	b2	b1	b0
	B7	B6	B5	B4	B3	B2	B1	B0
リセット後の値	0	0	0	0	0	0	0	0

ビット	シンボル	ビット名	機能	R/W
b0	B0	Pm0端子モード制御ビット	0：汎用入出力ポートとして使用 1：周辺機能として使用	R/W
b1	B1	Pm1端子モード制御ビット		R/W
b2	B2	Pm2端子モード制御ビット		R/W
b3	B3	Pm3端子モード制御ビット		R/W
b4	B4	Pm4端子モード制御ビット		R/W
b5	B5	Pm5端子モード制御ビット		R/W
b6	B6	Pm6端子モード制御ビット		R/W
b7	B7	Pm7端子モード制御ビット		R/W

TxD端子、RxD端子の設定

RXシリアルデバッガでは内蔵されている全てのSCIチャネルがCubeSuite+とのインタフェースに利用可能です。ただし、TxD端子とRxD端子は初期状態、 I/Oポートとして動作しています。 このため、 使用するTxD端子とRxD端子に対応したポートモードレジスタ （PMR） の設定を行う必要があります。

そこで2-7頁のリストにおけるHardwareSetup関数ではポートモードレジスタで指定可能な設定を36行目から40行目のコメントで記載しています。使用するTxD端子とRxD端子に対応したコメントを解除して、使用しないTxD端子とRxD端子にはコメントを設定してください。なお、ポートモードレジスタは各グループ毎に異なりますから、それらは対応したRXグループのハードウェア・マニュアルを参照ください。

ビットレートの設定

ビットレートレジスタ（BRR）

	b7	b6	b5	b4	b3	b2	b1	b0
リセット後の値	1	1	1	1	1	1	1	1

BRRレジスタの設定値

$$N = \frac{PCLK \times 10^6}{64 \times 2^{2n-1} \times B} - 1$$

```
3: #define BRR 12 // Bitrate Register Value
```

B　：ビットレート（bps）
N　：ボーレートジェネレータのBRRの設定値（0≦N≦255）
PCLK　：動作周波数（MHz）
n　：0

ビットレート（bps）	動作周波数PCLK（MHz）											
	25			30			33			50		
	n	N	誤差（%）	n	N	誤差（%）	n	N	誤差（%）	n	N	誤差（%）
110	3	110	-0.02	3	132	0.13	3	145	0.33	3	221	-0.02
150	3	80	0.47	3	97	-0.35	3	106	0.39	3	162	-0.15
300	2	162	-0.15	2	194	0.16	2	214	-0.07	3	80	0.47
600	2	80	0.47	2	97	-0.35	2	106	0.39	2	162	-0.15
1200	1	162	-0.15	1	194	0.16	1	214	-0.07	2	80	0.47
2400	1	80	0.47	1	97	-0.35	1	106	0.39	1	162	-0.15
4800	0	162	-0.15	0	194	0.16	0	214	-0.07	1	80	0.47
9600	0	80	0.47	0	97	-0.35	0	106	0.39	1	40	-0.77
19200	0	40	-0.76	0	48	-0.35	0	53	-0.54	0	80	0.47
31250	0	24	0.00	0	29	0	0	32	0	0	49	0.00
38400	0	19	1.73	0	23	1.73	0	26	-0.54	0	40	-0.77

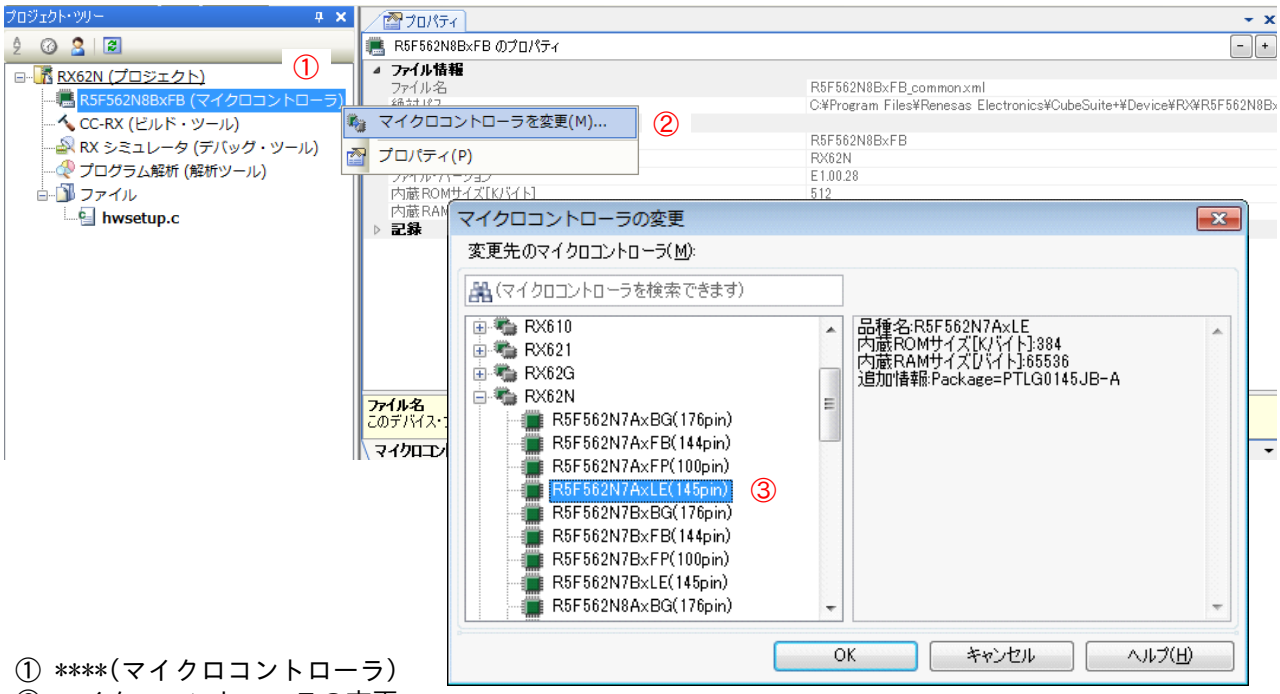
ビットレートの設定

RX シリアルデバッガは CubeSuite+ とのインタフェースに利用する SCI のビットレート（回線速度）として、115200bps/57600bps/38400bps/19200bps が利用可能です。このため、選択したビットレートが得られるよう SCI のビットレートレジスタ（BRR）に適切な値に設定する必要があります。BRR への設定値は上記の計算式（得たいビットレートと PCLK(B) の動作周波数）で求めることができます。また、代表的な動作周波数であれば、ハードウェアマニュアルに上記のような設定表が記載されています。ただし、115200bps や 57600bps 等の設定値は記載されていない場合があります。そのような場合は計算式を使って適切な設定値を算出してください。

算出後は 2-7 頁のリストの 3 行目にある BRR マクロの値を算出した値に変更してください。なお、2-7 頁のリストの 3 行目は PCLK(B)=40MHz から 115200bps を得るときの値です。

2.3　ビルド・ツールのオプション設定

マイクロコントローラの変更

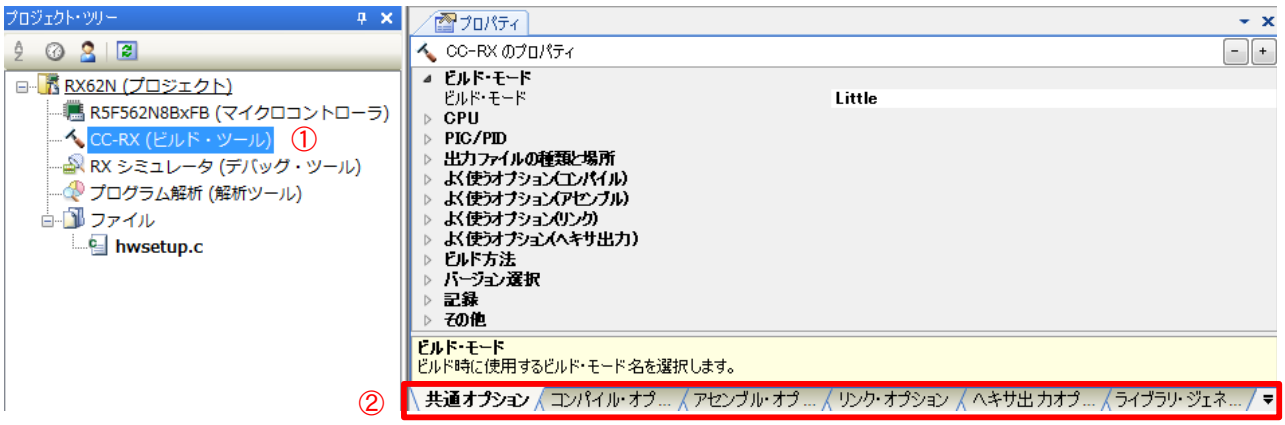


① ****(マイクロコントローラ)
② マイクロコントローラの変更
③ ご使用になられるマイクロコントローラを選択

マイクロコントローラの変更

RX シリアルデバッガのプロジェクトは、各グループ毎に適当なマイクロコントローラが選択されています。ご使用になられるマイクロコントローラとは異なるものが選択されていますから、必ず対応したマイクロコントローラへの変更をお願いします。

ビルド・ツールのオプション設定



① CC-RX(ビルド・ツール)
② タブ：共通オプション
　　コンパイル・オプション
　　アセンブル・オプション
　　リンク・オプション
　　ヘキサ出力オプション
　　ライブラリ・ジェネレート・オプション

・デフォルト設定は通常表示
・デフォルトではない場合はボールド表示
・ダブルクリックで設定を変更可能

ビルド・ツールのオプション設定

ビルド・ツールのオプション設定は、プロジェクト・ツリーの CC-RX（ビルド・ツール）から行います。目的のツールをクリック後、共通オプションを含めてツール毎にタブが分かれていますから、必要なツールを選択し、カテゴリ毎にオプションを設定します。

なお、デフォルトのオプション設定は通常表示です。もし、ボールド表示となっていれば、それはデフォルトのオプション設定ではないことを意味します。また、各設定はドロップダウンメニューまたはダブルクリックで設定の変更が可能です。

エンディアンの設定

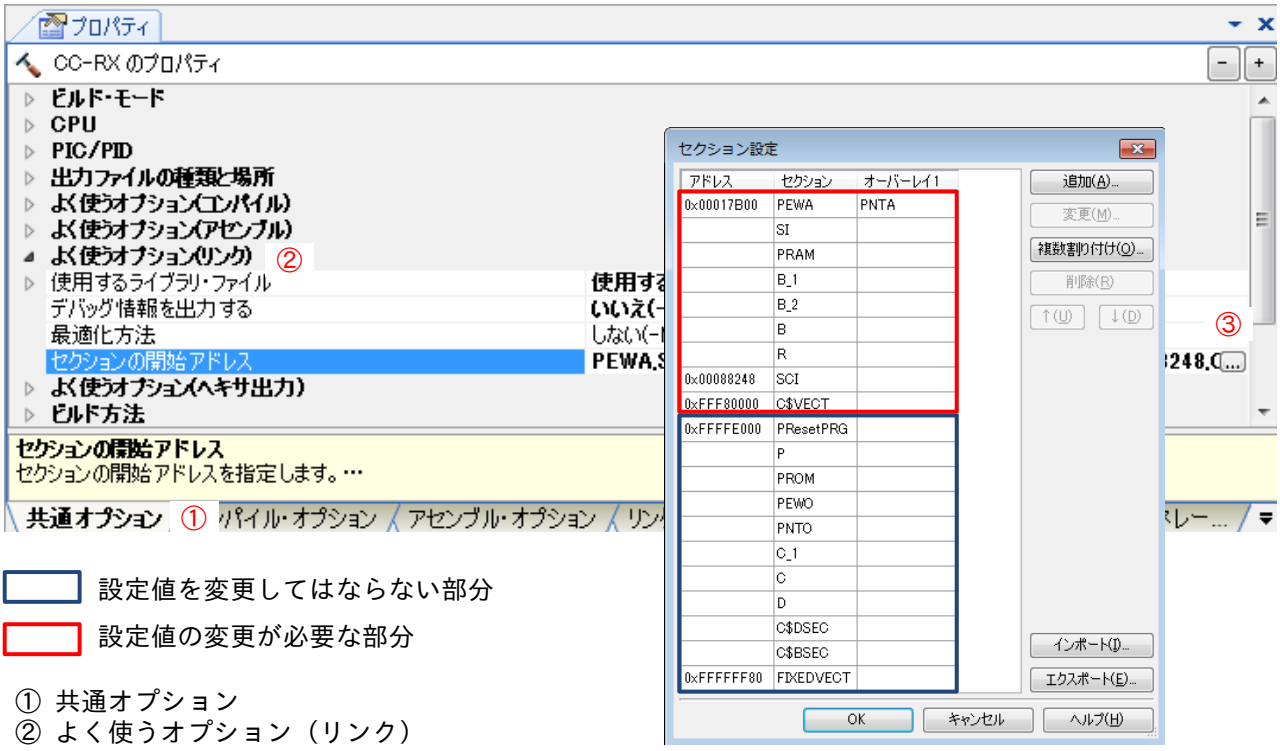
① 共通オプション
② ビルド・モード
③ ビルド・モード：defaultBuild（使用できません）
　　：Little（Littleエンディアン）
　　：Big（Bigエンディアン）

エンディアンの設定

RXシリアルデバッガはLittleエンディアンでもBigエンディアンでも動作可能です。デフォルトはLittleエンディアンに設定されていますが、Big エンディアンで動作させるのであれば、上記のオプションを変更してください。

エンディアンは共通オプションのカテゴリ「ビルド・モード」で選択します。設定値「Little」がLittleエンディアン、「Big」がBigエンディアンです。なお、設定値「DefaultBuild」は使用できません。「Little」または「Big」を選択ください。

セクションの設定



設定値を変更してはならない部分

設定値の変更が必要な部分

① 共通オプション
② よく使うオプション（リンク）
③ セクションの開始アドレス：[...]の編集ボタンをクリック

セクションの設定

RX シリアルデバッガのオプション設定（コンフィグレーション）、その殆どは上記のセクション・オプションとなります。セクション・オプションは共通オプションのカテゴリ「よく使うオプション（リンク）」の「セクションの開始アドレス」で設定します。

目的のオプションの右側にある [...] の編集ボタンをクリックすると上記の編集ダイアログが表示されますから、設定の変更は編集ダイアログで行ってください。変更が必要なものは上記の図では赤色の部分、具体的には以下の３つです。

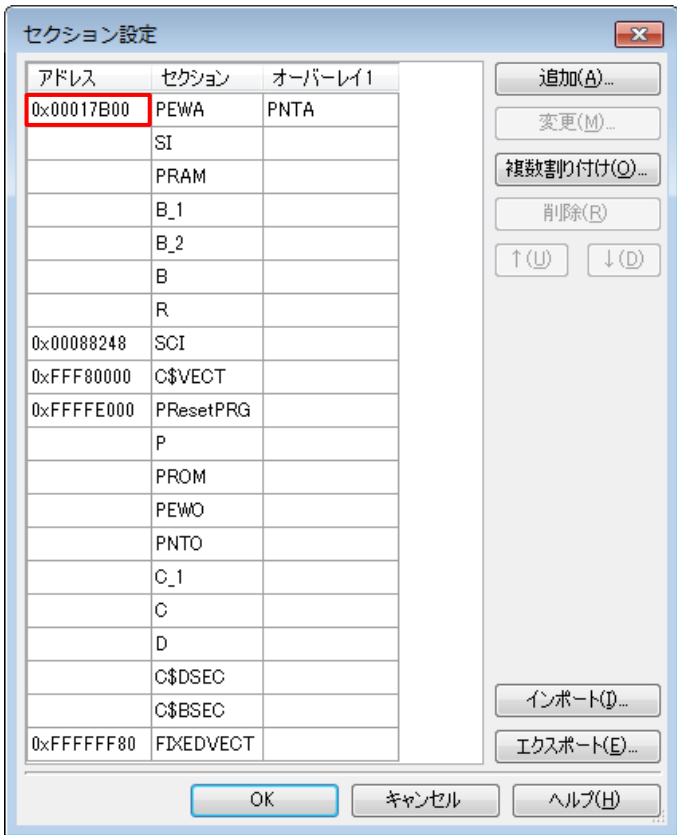
1. RXシリアルデバッガ使用するRAM領域の番地指定
   設定値の詳細は2-15頁を参照ください。

2. CubeSuite+とのインタフェースに利用するSCIチャネルの番地指定
   設定値の詳細は2-16頁を参照ください。

3. 可変ベクタテーブルの番地指定
   設定値の詳細は2-17頁を参照ください。

また、上記の図で青色の部分は設定値の変更が許されない部分です。必ずデフォルトの設定値をご使用ください。設定値を変更してしまった場合の動作は保証されません。

シリアルデバッガ用 RAM 領域の番地指定



RXシリアルデバッガのRAM領域

以下のセクションはRAM領域に配置
PEWA, PNTA, SI, PRAM, B_1, B_2, B, R

【推奨設定値】
内蔵RAMの最終番地+1 － 0x500

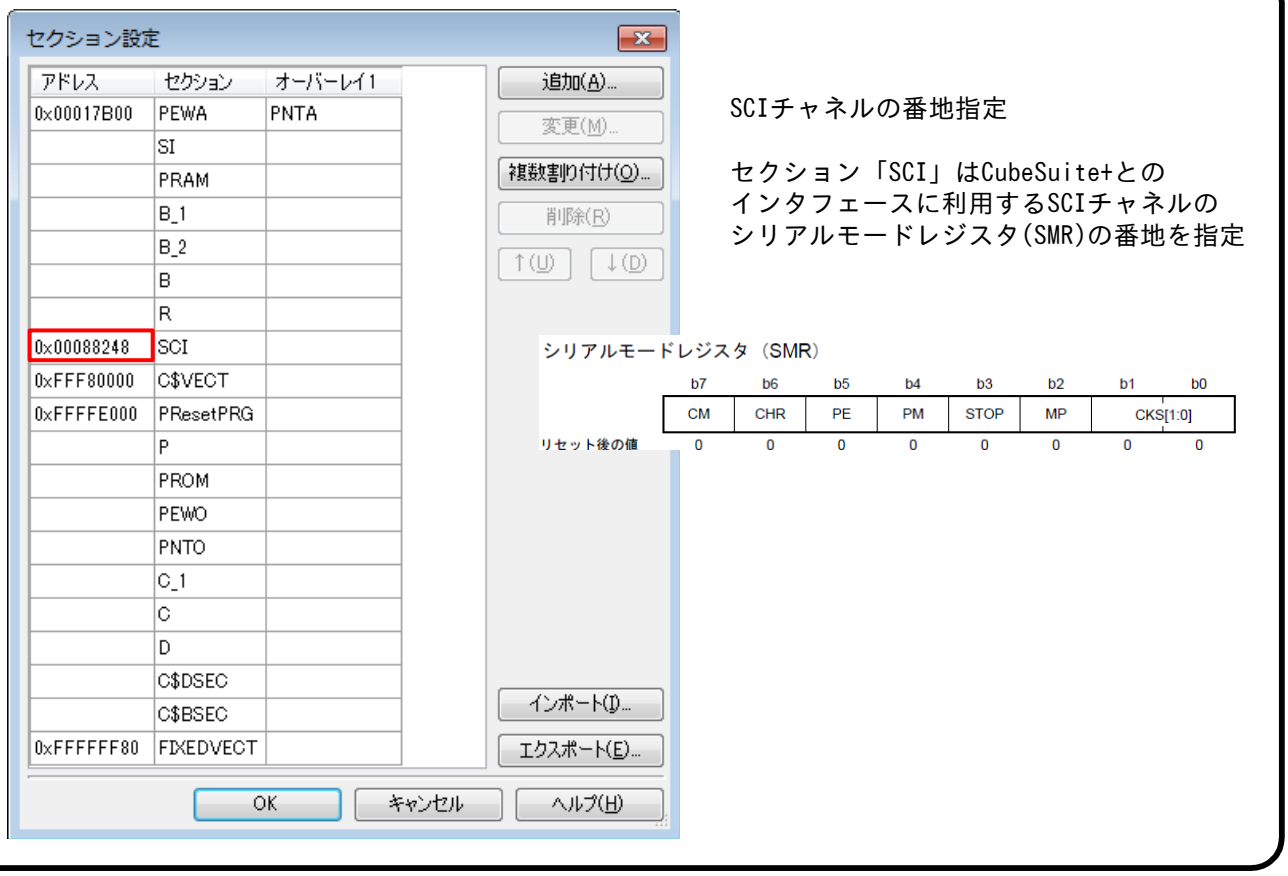
シリアルデバッガ用RAM領域の番地指定

RXシリアルデバッガは動作するに当たり、約1KByte（正確には1,280Byte[0x500Byte]）のRAM領域を使用します。セクションは「PEWA」から「R」までがRAM領域に配置するセクションです。目的のセクションは利用可能なRAM領域であれば番地は問いません。４バイトのアライメント境界であれば何番地でも動作します。

ただし、ユーザプログラムのデバッグを考えた場合、RXシリアルデバッガのRAM領域は使用可能なRAM領域の最後に配置した方が良いでしょう。例えば、RX62N、内蔵RAMが96KByte版であれば、0x00000000～0x00017FFFまで内蔵RAMがありますから、最終番地+1の0x00018000 － 0x500の0x00017B00番地に配置するのが最適となります。

デフォルトの設定値は各グループ毎に最大の内蔵 RAM を持つものに合わせて番地指定が行われています。内蔵RAM容量の少ないシリーズをお使いの場合は番地の変更を行う必要があります。

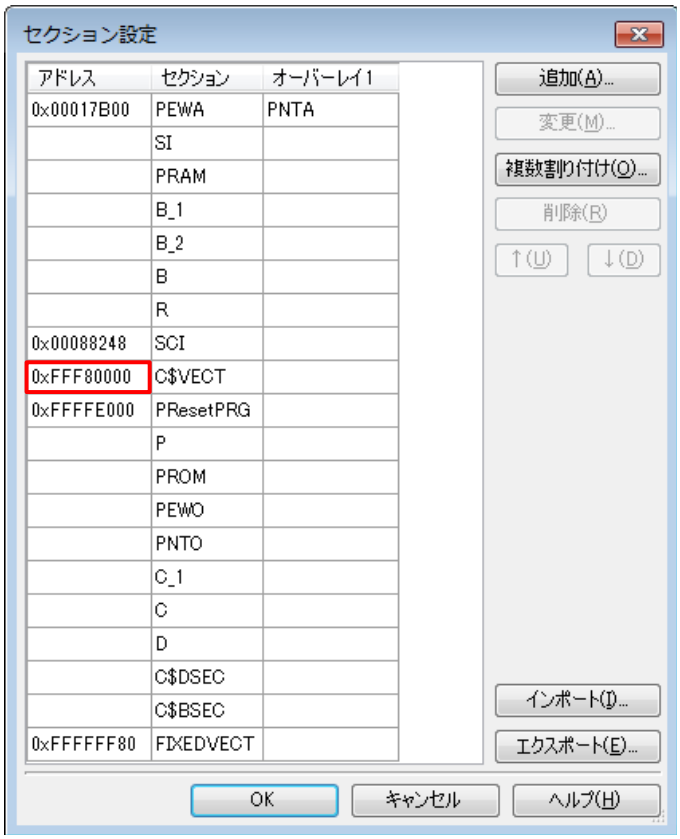
CubeSuite+とのインタフェースに利用するSCIチャネルの指定



CubeSuite+とのインタフェースに利用するSCIチャネルの指定

CubeSuite+とのインタフェースに利用するSCIチャネルは、セクション「SCI」の番地指定で決定します。目的のセクションを対応したSCIのシリアルモードレジスタ（SMR）の番地に合わせてください。

可変ベクタテーブルの番地指定



可変ベクタテーブルの番地指定

セクション「C$VECT」は内蔵フラッシュメモリの先頭番地を指定

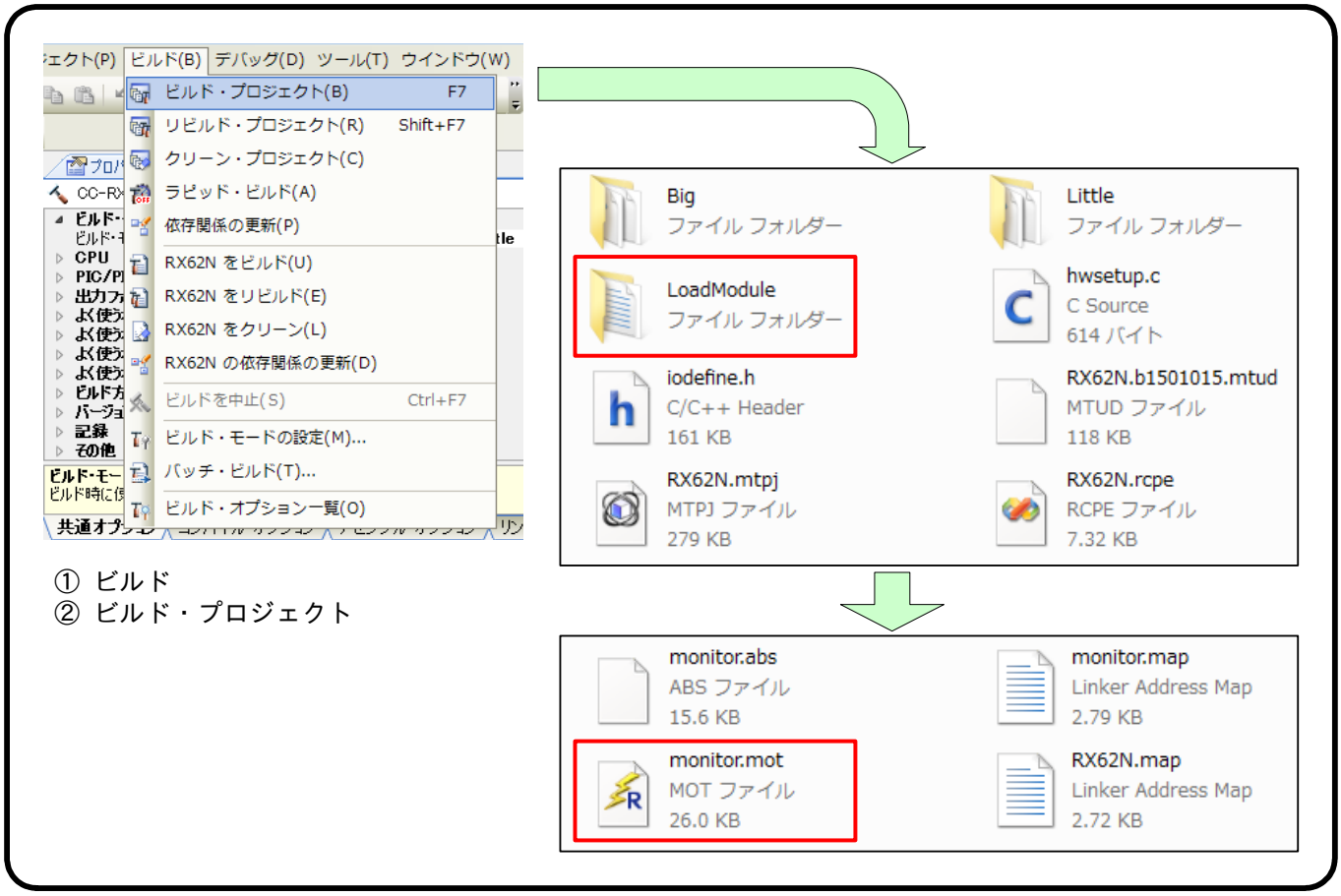
可変ベクタテーブルの番地指定

セクション「C$VECT」はデバッグ対象のユーザプログラムの可変ベクタテーブルの配置場所を意味します。RX シリアルデバッガでは内蔵フラッシュメモリの容量をセクション「C$VECT」の配置場所で識別します。このため、セクション「C$VECT」は必ず内蔵フラッシュメモリの先頭番地に配置しなければなりません。

これは 4-5 頁で紹介するユーザプログラムの制約事項と同じ意味を持ちますから、必ず内蔵フラッシュメモリの先頭番地に変更してください。

2.4　シリアルデバッガのビルドと内蔵フラッシュメモリへの書き込み

RXシリアルデバッガのビルド



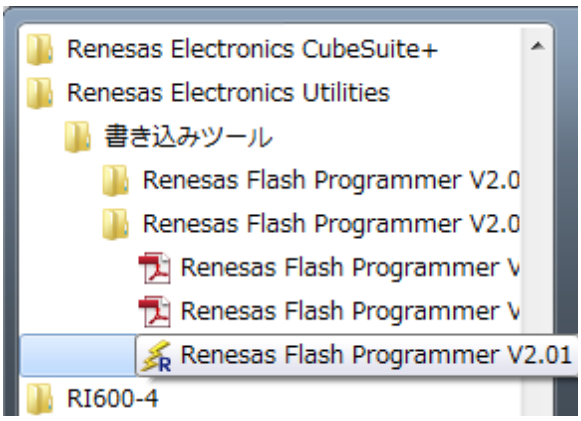
RXシリアルデバッガのビルド

2.3節の設定が完了したら、RXシリアルデバッガのビルドを行い、ロードモジュールを作成します。ビルドは、ビルド・メニューにあるビルド・プロジェクトで実行できます。エラーがなければ、ロードモジュールは各グループ毎のフォルダにある「LoadModule」フォルダに生成されます。そのフォルダ内にある「monitor.mot」がRXシリアルデバッガのロードモジュールです。

内蔵フラッシュメモリへの書き込み

Renesas Flash Programmer

フラッシュ書き込みソフトウエア



内蔵フラッシュメモリへの書き込み

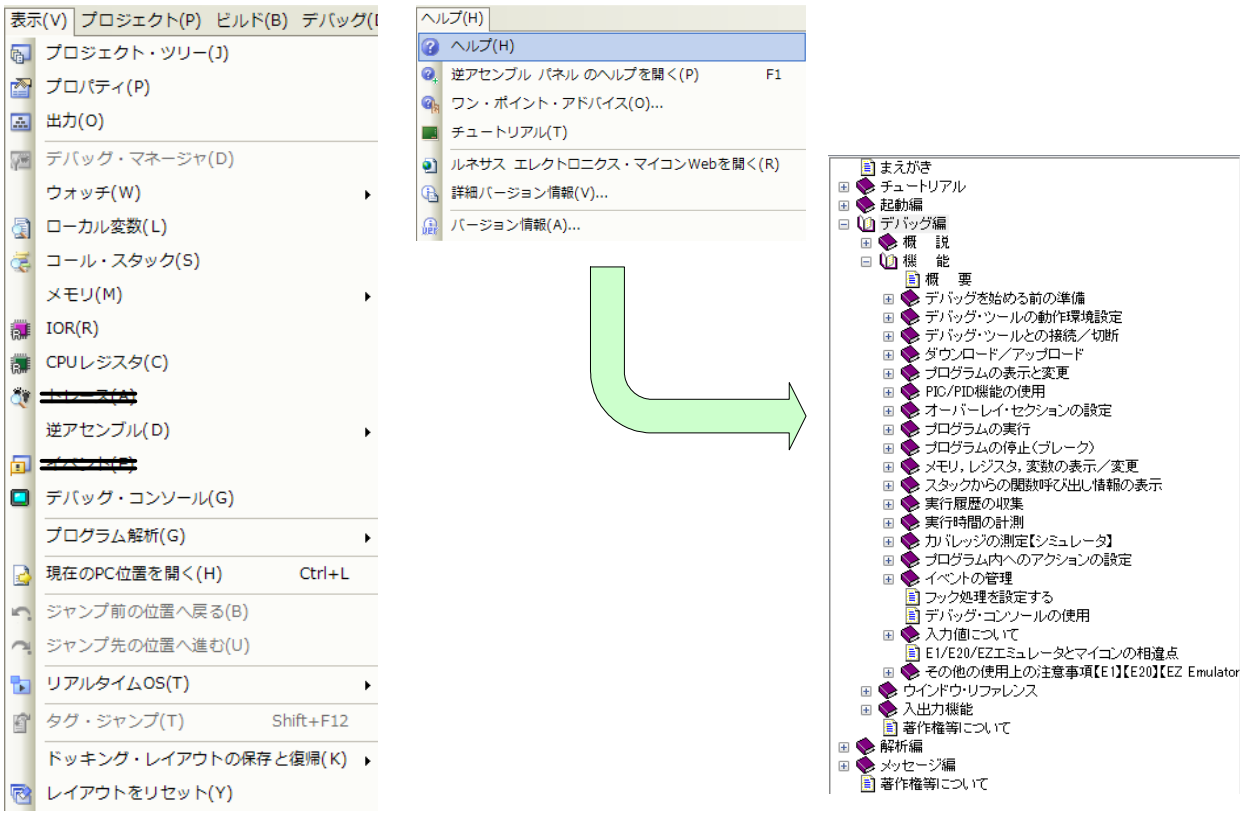
作成されたRXシリアルデバッガはRFP（Renesas Flash Programmer）等の書き込みツールを使って、使用するRXファミリの内蔵フラッシュメモリへ書き込みを行ってください。なお、書き込みツールの使用方法は当該ツールのユーザーズマニュアルを参照ください。

第 3 章

RX シリアルデバッガの機能と接続用設定

3.1　RXシリアルデバッガの機能

RXシリアルデバッガの機能



RXシリアルデバッガの機能

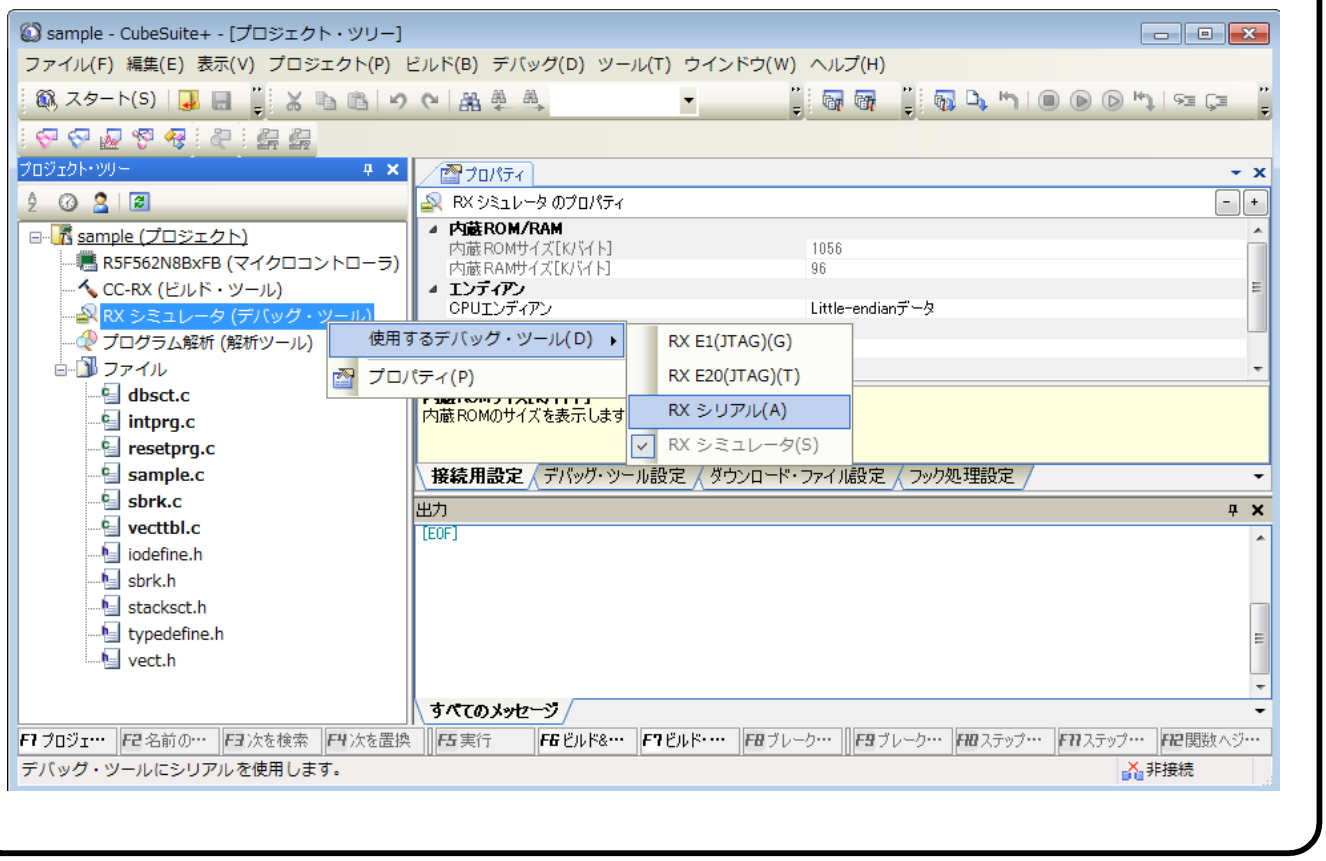
RXシリアルデバッガはE1/E20エミュレータの約60%程度のデバッグ機能を有しています。ただし、トレース、イベントの一部、外部フラッシュメモリやデータフラッシュへの書き込み機能はサポートされていません。以下がRXシリアルデバッガでサポートされている機能です。

- プログラムのダウンロード/アップロード
- プログラムの表示と変更
- プログラムの停止（ブレーク）
- メモリ、レジスタ、変数の表示/変更

なお、各機能の使用方法はCubeSuite+のRX Helpにおけるデバッグ編を参照ください。

3.2　接続用設定

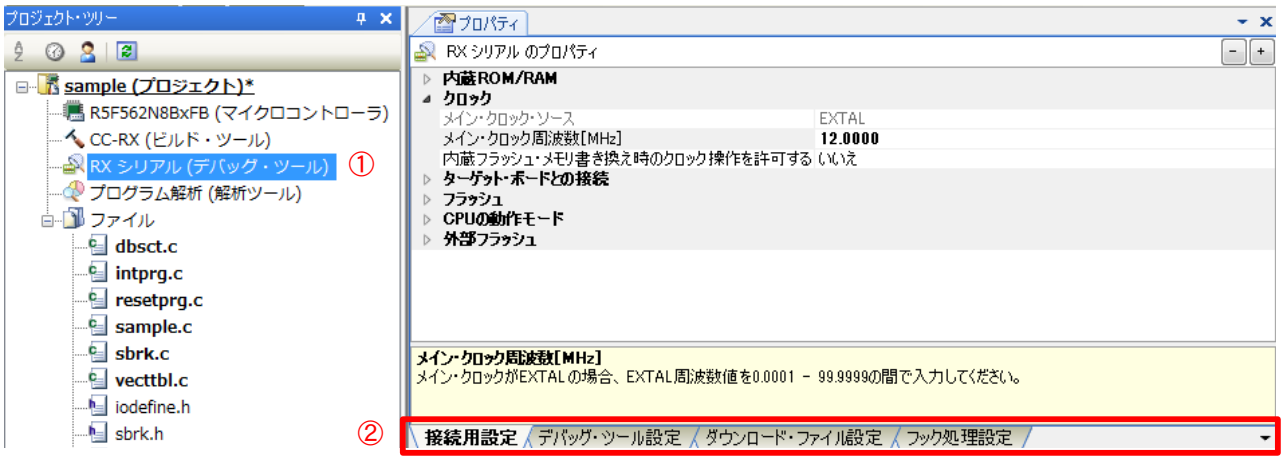
デバッグ・ツールの選択



デバッグ・ツールの選択

RXシリアルデバッガは、プロジェクト・ツリーのデバッグ・ツールで右クリックし、表示されたポップアップメニューの「使用するデバッグ・ツール」から「RXシリアル」を選択することで利用可能となります。

デバッグ・ツールのオプション設定



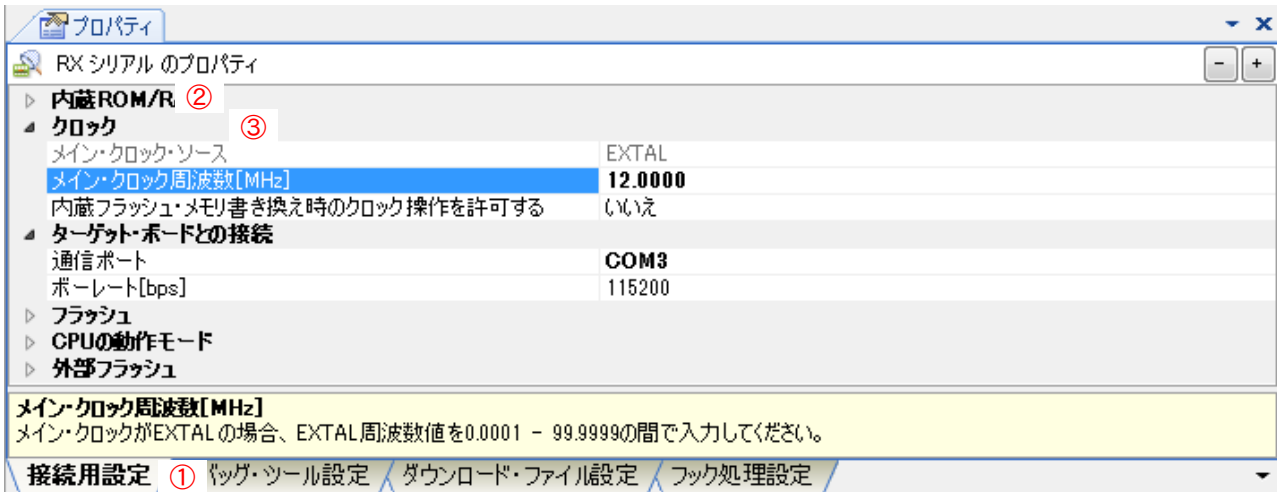
① RXシリアル(デバッグ・ツール)
② タブ：接続用設定
　　デバッグ・ツール設定
　　ダウンロード・ファイル設定
　　フック処理設定

・デフォルト設定は通常表示
・デフォルトではない場合はボールド表示
・ダブルクリックで設定を変更可能

デバッグ・ツールのオプション設定

　デバッグ・ツールのオプション設定は、プロジェクト・ツリーのRXシリアル（デバッグ・ツール）から行います。目的のツールをクリック後、接続用設定を含めて機能毎にタブが分かれていますから、必要なタブを選択し、カテゴリ毎にオプションを設定します。

メイン・クロック周波数[MHz]の指定



① 接続用設定
② クロック
③ メイン・クロック周波数[MHz]

本設定値はRX610、RX62*系のみ有効な値であり、
RX63*系、RX2**系では無効な値です。
指定可能な範囲で適当な値を設定してください。

メイン・クロック周波数[MHz]の指定

メイン・クロックの動作周波数をMHz単位で指定します。なお、本設定値はRX610、RX62*系のみ有効な値となります。その理由はRX610、RX62*系の場合、メイン・クロックはEXTALのみですが、RX63*系やRX2**系では内部の高速オンチップオシレータ（HOCO）等をメイン・クロックとして利用可能だからです。

このため、RXシリアルデバッガでは、RX610とRX62*系は本設定値と2-4頁で紹介したSCKCRの設定値で各種の動作周波数を判断します。一方、RX63*系とRX2**系では本設定値を無視し、2-8頁で紹介したCLOCKマクロの値とSCKCRの設定値で各種の動作周波数を判断します。

従って、RX610とRX62*系において本設定値に誤りがある場合、内蔵フラッシュメモリに対するユーザプログラムのダウンロードでエラーが発生することがあります。必ず、EXTAL に接続されている振動子と同一の周波数を設定してください。

通信ポートの選択

① 接続用設定
② ターゲット・ボードとの接続
③ 通信ポート：COM*（お使いのPC環境によって変化します）

通信ポートの選択

RXシリアルデバッガとのインタフェースに利用するPC側の通信ポートを選択します。この設定値はお使いのPC環境によって変化します。

ボーレート[bps]の選択

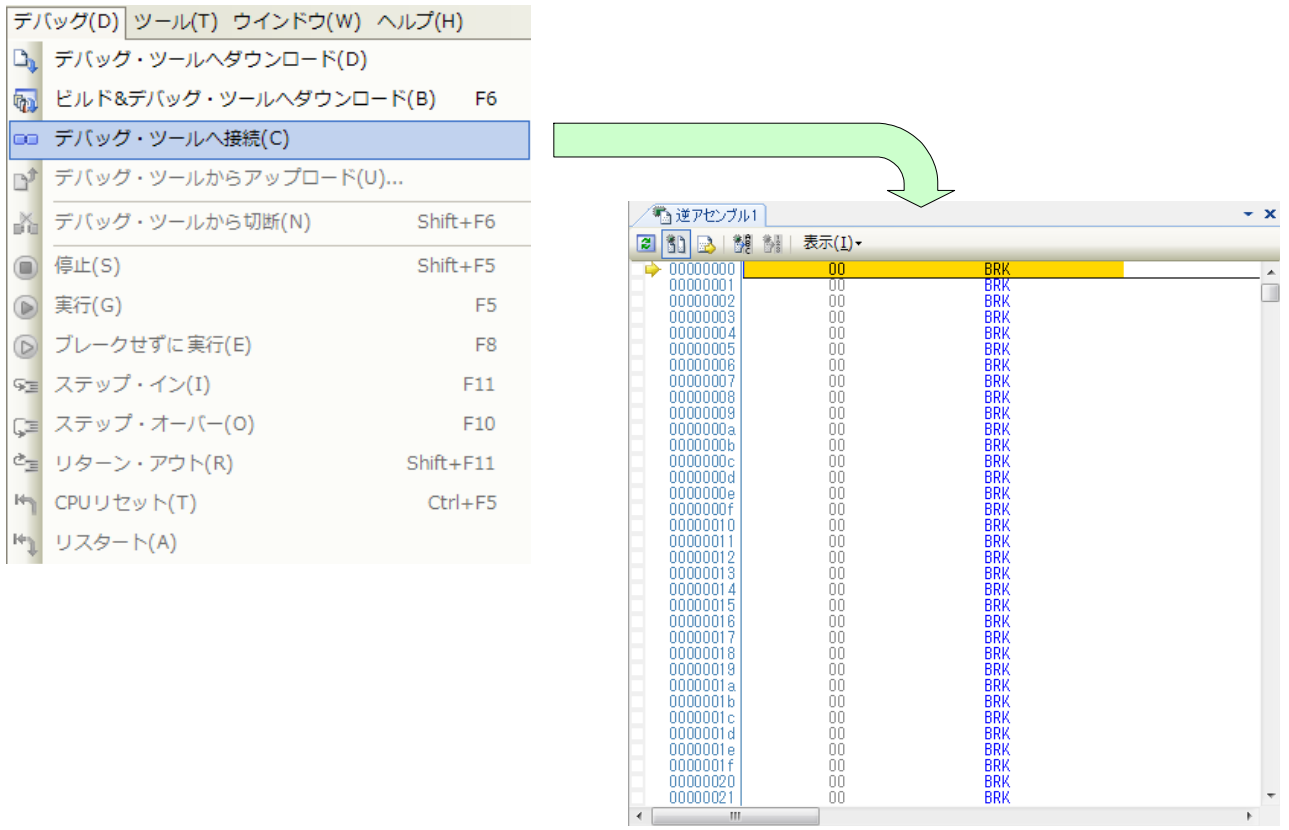
① 接続用設定
② ターゲット・ボードとの接続
③ ボーレート：115200
：57600
：38400
：19200

ボーレート[bps]の選択

RXシリアルデバッガとのインタフェースに使用するシリアルのボーレートを選択します。本設定値は2-6頁や2-10頁で紹介したSCIのビットレート（回線速度）と同じ値を選択しなければなりません。

なお、設定値は出来るだけ速い回線速度を選択した方が、応答性良くRXシリアルデバッガが利用できます。ただし、ユーザプログラムのダウンロード時に書き込みエラーが頻繁に発生するようであれば、遅い回線速度に変更してみてください。それによって書き込みエラーを抑えられる可能性があります。

ターゲットとの接続



ターゲットとの接続

接続用設定のオプション設定が完了したら、ターゲット・ボードの電源を投入し、デバッグ・メニューの「デバッグ・ツールへ接続」コマンドでRXシリアルデバッガへ接続を行います。正しく接続されれば、上記のような逆アセンブル・ウィンドウが表示されます。もし、何らかのエラーで接続が行われない場合は第２章と第３章の内容を再度、ご確認ください。

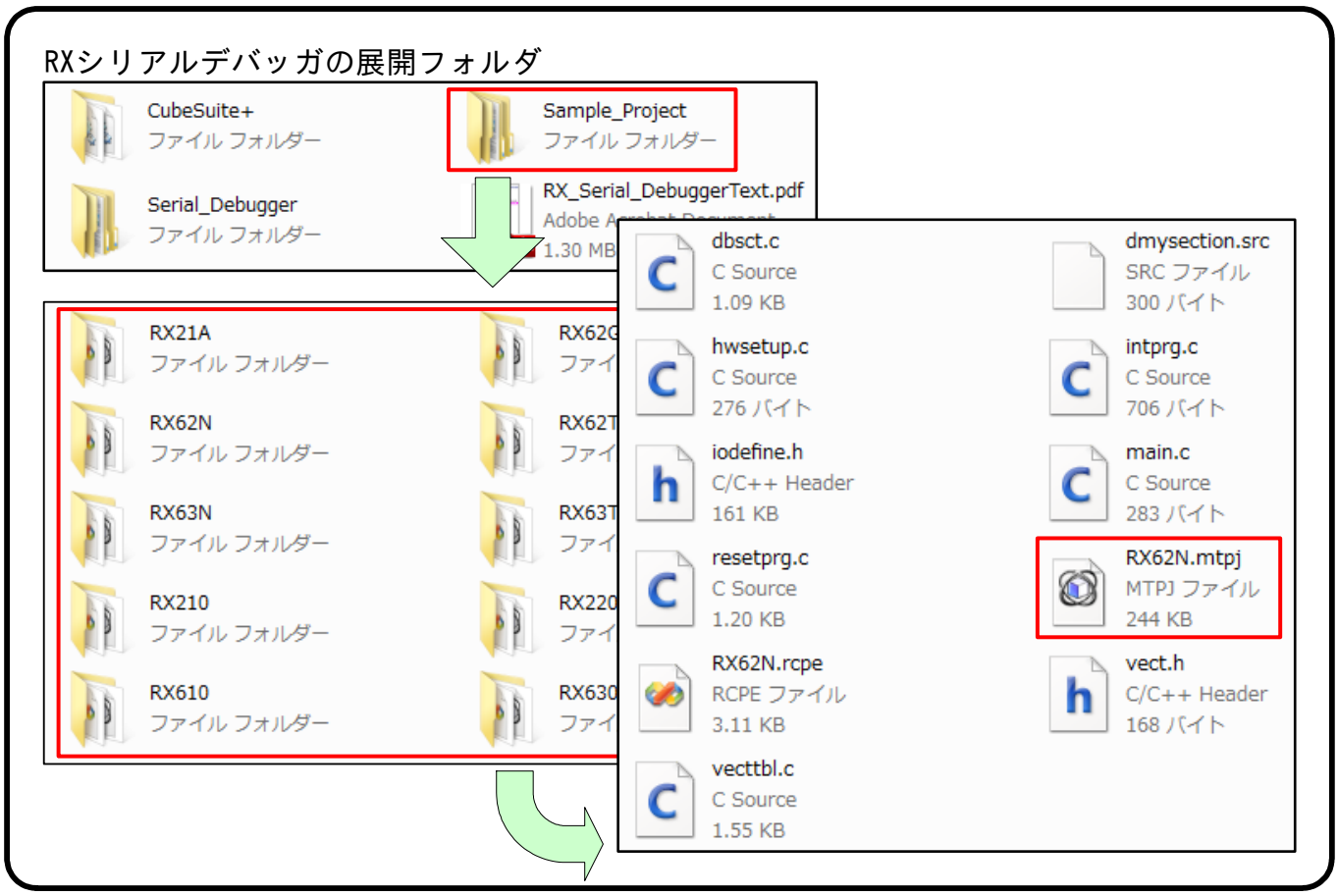
なお、接続後は3.1節で紹介したE1/E20の約60%程度のデバッグ機能が利用可能となります。

第 4 章

RX シリアルデバッガ用のサンプルプロジェクト

4.1　サンプルプロジェクトの立ち上げ

サンプルプロジェクトの立ち上げ



サンプルプロジェクトの立ち上げ

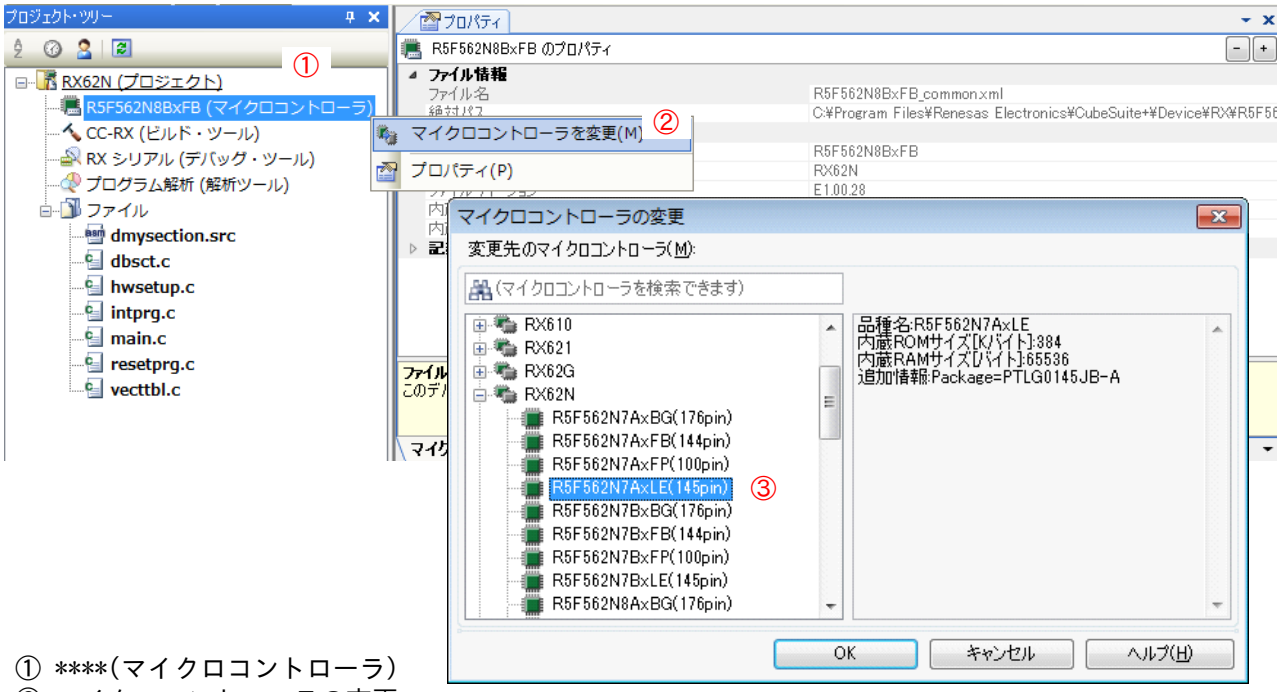
RXシリアルデバッガでは、RXシリアルデバッガを利用するために必要な設定を行ったデバッグ対象ユーザプログラムのサンプルプロジェクトを提供しています。必要に応じてご利用ください。なお、サンプルプロジェクトは各グループ毎に準備されていますから、お使いのグループに対応したサンプルプロジェクトを立ち上げてください。

Web よりダウンロードし、本説明資料が存在するフォルダ内にある「Sample_Project」フォルダ内に各グループ毎のフォルダがありますから、その中にあるCubeSuite+用のプロジェクト・アイコンをダブルクリックしてください。これでRXシリアルデバッガに対応したサンプルプロジェクトが立ち上がります。

立ち上げ後は第３章で紹介した 3-5 頁のメイン・クロックの設定から 3-7 頁のビットレートの選択までを実施し、RXシリアルデバッガへの接続準備を行ってください。

4.2　ビルド・ツールのオプション設定

マイクロコントローラの変更

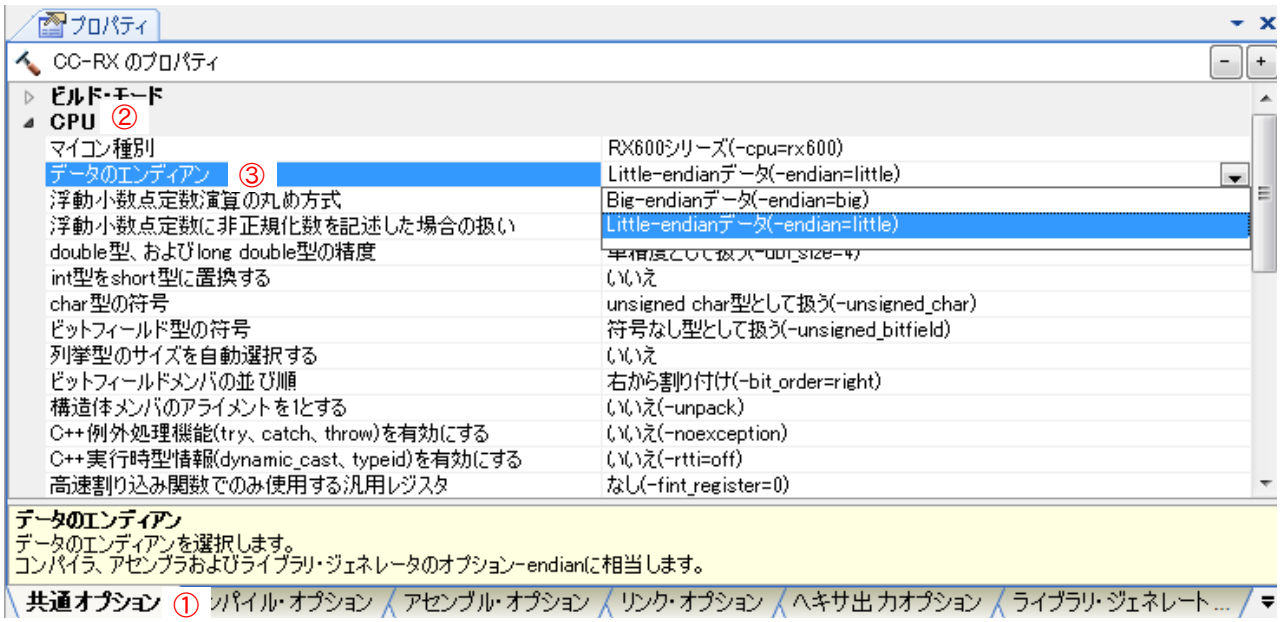


① ****(マイクロコントローラ)
② マイクロコントローラの変更
③ ご使用になられるマイクロコントローラを選択

マイクロコントローラの変更

サンプルプロジェクトは、各グループ毎に適当なマイクロコントローラが選択されています。ご使用になられるマイクロコントローラとは異なるものが選択されていますから、必ず対応したマイクロコントローラへの変更をお願いします。

エンディアンの設定



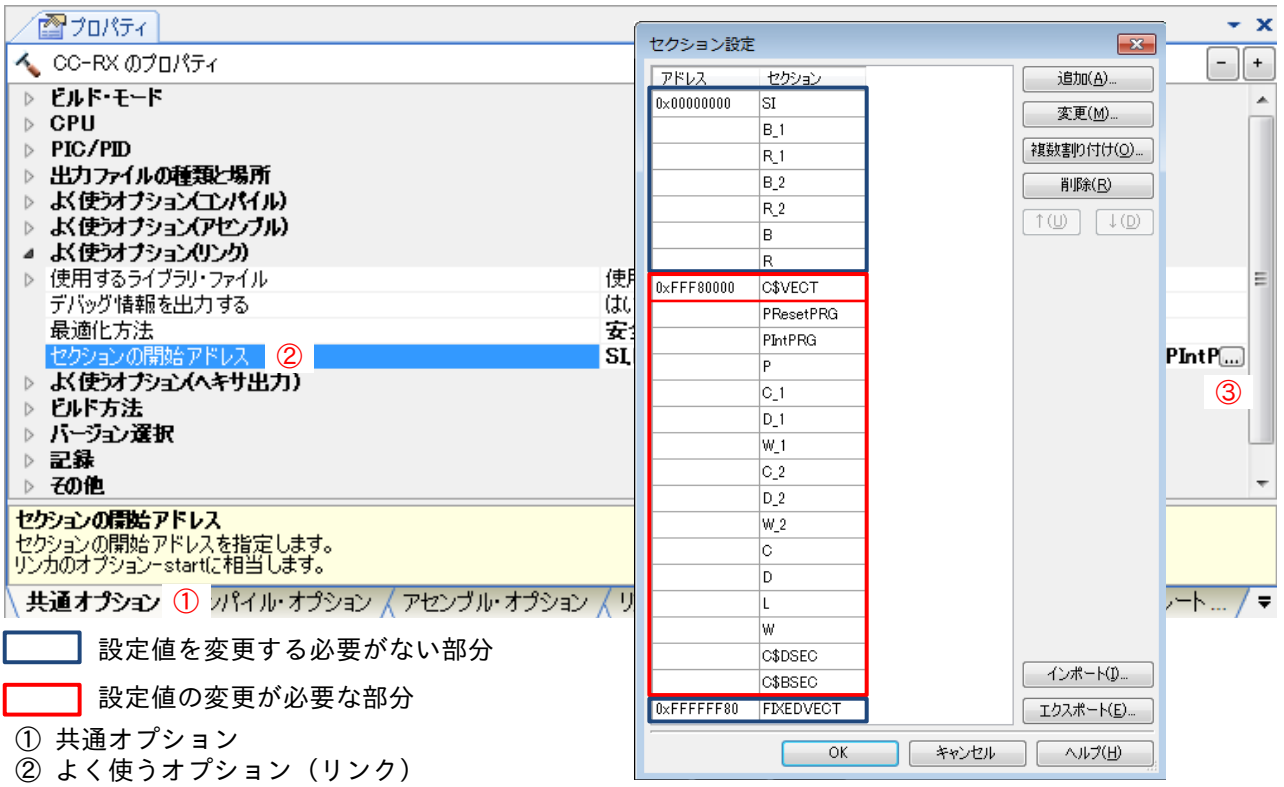
① 共通オプション
② CPU
③ データのエンディアン：Big-endianデータ（-endian=big）
　　：Little-endianデータ（-endian=little）

エンディアンの設定

RXシリアルデバッガはLittleエンディアンでもBigエンディアンでも動作可能です。デフォルトはLittleエンディアンに設定されていますが、Big エンディアンで動作させるのであれば、上記のオプションを変更してください。

エンディアンは共通オプションのカテゴリ「CPU」における「データのエンディアン」で選択します。設定値「Little-endian データ」がLittle エンディアン、「Big-endian データ」がBig エンディアンです。なお、本設定値は2-13頁で紹介したRXシリアルデバッガをビルドした際のエンディアンに合わせる必要があります。異なるエンディアンが設定されていた場合は正しく動作しません。

可変ベクタテーブルの番地指定



設定値を変更する必要がない部分

設定値の変更が必要な部分

① 共通オプション
② よく使うオプション（リンク）
③ セクションの開始アドレス：[...]の編集ボタンをクリック

可変ベクタテーブルの番地指定

セクション「C$VECT」は可変ベクタテーブルの配置場所を意味します。RX シリアルデバッガではデバッグ対象のユーザプログラムは内蔵フラッシュメモリの先頭番地に配置することが制約事項となっています。もし、配置場所が内蔵フラッシュメモリの先頭番地でない場合、設定値を変更し、必ずセクション「C$VECT」を内蔵フラッシュメモリの先頭番地に配置してください。この設定値は2-17頁で紹介したRX シリアルデバッガをビルドする際の設定値と同じでなければなりません。

なお、可変ベクタテーブルのセクション「C$VECT」以外は5-2頁で紹介するRX シリアルデバッガが使用するROM/RAM 領域以外であれば何番地に配置しても構いません。デバッグ対象のユーザプログラムの中で配置場所が固定なのは可変ベクタテーブルのセクション「C$VECT」のみです。

4.3　ビルドとダウンロード

ビルドとダウンロード

ビルドとダウンロード

　オプション設定が完了したら、必要なプログラムのコーディングを追加し、ビルド・メニューから「ビルド」を行ってください。その後はデバッグ・メニューの「デバッグ・ツールへ接続」を行い、RXシリアルデバッガと接続を行い、デバッグ・メニューの「デバッグ・ツールへダウンロード」を行い、デバッグを実施してください。なお、ビルド、接続、ダウンロードを一緒に行うデバッグ・メニューの「ビルド＆デバッグ・ツールへダウンロード」のコマンドもありますから、適宜コマンドを使い分けてください。

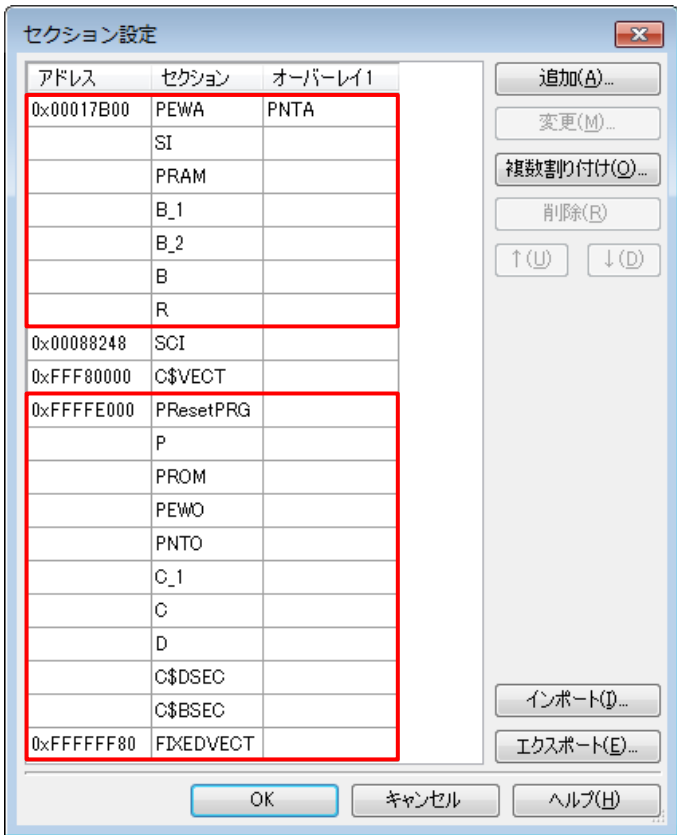
　また、デバッグ終了時はデバッグ・メニューの「デバッグ・ツールから切断」を行い、RXシリアルデバッガとの接続を解除してください。

第 5 章

デバッグ対象プログラムの制約事項

5.1　使用上の制約事項

使用可能なROM/RAM領域



【RXシリアルデバッガのRAM領域】
PEWA, PNTA, SI, PRAM, B_1, B_2, B, R

【RXシリアルデバッガのROM領域】
PResetPRG, P, PROM, PEWO, PNTO, C_1, C, D, C$DSEC, C$BSEC, FIXEDVECT

これらの領域にデバッグ対象のプログラムを配置することはできません。
ただし、ROM領域に関しては、無視してダウンロードを実施します。

使用可能なROM/RAM領域

デバッグ対象のプログラムが使用可能なROM/RAM領域は、RXシリアルデバッガが使用するROM/RAM領域を避けた領域となります。デフォルトの設定ならば、RXシリアルデバッガは

- ROM領域は内蔵フラッシュメモリの最後の8KByteまたは12KByte（0x2000Byteまたは0x3000Byte）
- RAM領域は内蔵RAMの最後の1,280Byte（0x500Byte）

が使用するROM/RAM領域です。デバッグ対象のプログラムは、上記以外のメモリ領域を使用しなければなりません。

ただし、デバッグ対象のプログラムにおいても固定ベクタ領域は内蔵フラッシュメモリの最後に配置しなければなりません。これはCPUの機能上、変更できない部分です。このため、ROM領域に関してはRXシリアルデバッガと同一のROM領域を使用しても構いません。もし、ダウンロード時にRXシリアルデバッガと同一のROM領域があった場合は内蔵フラッシュメモリへの書き込みを行わずにダウンロードを行います。つまり、RXシリアルデバッガと同一のROM領域を使用した場合、ダウンロードは正常に行われますが、その部分は内蔵フラッシュメモリには書き込まれていないことになります。結果、固定ベクタ領域以外は必ずRXシリアルデバッガ使用するROM領域とは異なる領域を使用しなければなりません。

SCIチャネル

2-16頁で紹介したCubeSuite+とのインタフェースに利用するSCIチャネル、及び対応したSCIのTxD端子とRxD端子は使用できません。

可変ベクタテーブルの配置

2-17頁で紹介した可変ベクタテーブルの配置場所であるセクション（通常は「C$VECT」セクション）は必ず内蔵フラッシュメモリの先頭番地に配置する必要があります。

割り込みの許可

エントリ関数の例

```
#include <machine.h>
#pragma entry PowerON_Reset
void PowerON_Reset(void)
{
        set_intb(__sectop("C$VECT"));
        setpsw_i( );
        set_fpsw(FPSW_init);
        HardwareSetup( );
        _INITSCT( );
        main( );
}
```

リセット後は可能限り、早い段階で
割り込みを許可する

可能な限り enable を指定する

CMI0割り込み関数の例

```
#include "iodefine.h"

#pragma interrupt CMI0(vect=VECT(CMT0,CMI0),enable)
void CMI0(void)
{
    PORT1.DR.BIT.B5 ^= 1;
}
```

割り込みの許可

RXシリアルデバッガでは、デバッグ・ツールに接続後は約300msの間隔で接続状況の確認、具体的には40バイト程度のデータ送受信を行います。この動作状況の確認はデバッグ対象のプログラムを実行中も行われます。この CubeSuite+ との接続状況の問い合わせに対応するため、RX シリアルデバッガではインタフェースに利用するSCIは送受信共に最上位のレベル15の割り込みを利用しています。

以上のことから、デバッグ対象のユーザプログラムではレベル15の割り込みが常に受け付け可能な状態で動作しなければなりません。もし、割り込みの禁止時間が長い場合、CubeSuite+からの接続状況の問い合わせに対応できず、接続が強制的に解除されます。デバッグ対象のユーザプログラムにおける割り込み禁止状態は可能な限り短くしなければなりません。具体的には上記のリストに示す通り、

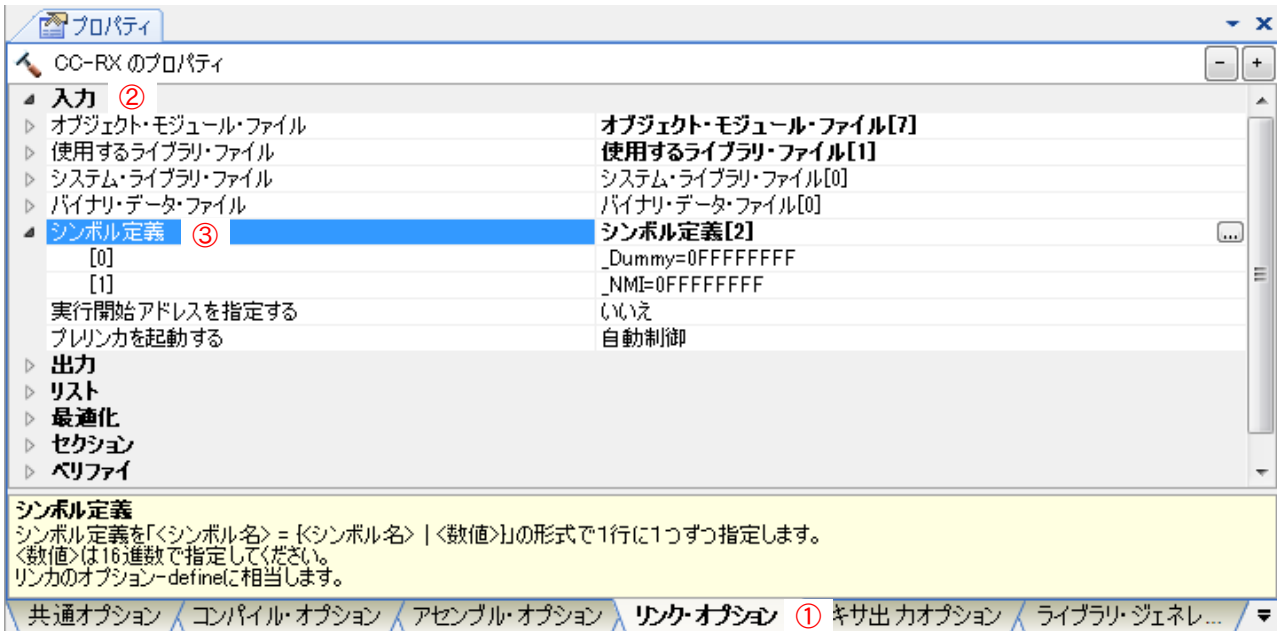
- リセット後は可能な限り速く割り込みを許可状態とする（setpsw_i組み込み関数の呼び出し）。
- 割り込み関数では常に多重割り込みの許可を行う（enable指定を行う）。
- 高速割り込み及びレベル15の割り込みは基本的に利用不可とする。

としてください。特に高速割り込み及びレベル15の割り込みは利用不可ではありませんが、RXシリアルデバッガの動作に影響を与えるため、利用する際は割り込み関数が短くなるように記述してください。

割り込みスタック領域

RXシリアルデバッガでは、デバッグ対象のプログラム実行中もCubeSuite+との接続状況の問い合わせに対応するため、SCI の送受信割り込みに応答します。また、その処理には割り込みスタック領域を 40 バイト消費します。このため、デバッグ対象のプログラムでは、自身が使用する割り込みスタック領域のサイズより、40バイト多くスタック領域を確保する必要があります。

NMI と例外処理



① リンク・オプション
② 入力
③ シンボル定義：_Dummy（例外処理関数のアドレス）
　　：_NMI（NMI関数のアドレス）

NMI と例外処理

NMI と未定義命令等の例外処理は固定ベクタテーブルにアドレスを登録しなければなりません。しかしながら、RX シリアルデバッガでは固定ベクタテーブルの領域は RX シリアルデバッガが使用する領域となっています。このため、基本的に NMI と例外に対する処理は、デバッグ対象のプログラムでは使用することができません。

ただし、上記の示すように RX シリアルデバッガをビルドする際のオプション設定を変更することによって、NMI と例外に対応した処理をデバッグ対象のプログラムで利用することが可能となります。予め、デバッグ対象のプログラムにおいて、NMI や例外処理に対応した関数を作成し、その関数が特定の番地の配置となるように #pragma やビルドのオプション設定を行ってください。

RX シリアルデバッガでは、以下の２つのシンボルが

- _Dummy　⇒　例外処理に対応した関数のアドレス
- _NMI　　⇒　NMI に対応した関数のアドレス

に対応しています。これらのシンボルの値を変更することによって、デバッグ対象のプログラムで NMI と例外処理をサポートすることが可能となります。

5.2　デバッグ完了後の操作

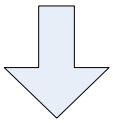
HardwareSetup関数の修正（ソースファイル名：hwsetup.c）

RXシリアルデバッガのHardwareSetup関数（RX610、RX62*系）

```
void HardwareSetup(void)
{
    SYSTEM.SCKCR.LONG = 0x00020100;      // If XTAL=12MHz Then
}                                        // ICLK=96MHz,BCLK=24MHz,PCLK=48MHz
```

RXシリアルデバッガのHardwareSetup関数（RX63*、RX2**系）

```
void HardwareSetup(void)
{
// If Used 10MHz Main Clock
    SYSTEM.PRCR.WORD = 0xA503;         // Protect Disable
 :
    SYSTEM.SCKCR.LONG = 0x21021222;   // FCLK=40MHz,ICLK=80MHz,BCLK=40MHz
                                      // PCLKA=80MHz,PCLKB=40MHz,PCLKD=40MHz
 :
    SYSTEM.PRCR.WORD = 0xA500;         // Protect Enable
}
```



HardwareSetup関数内に記述されているクロックの設定処理を
デバッグ対象のユーザプログラムにコピー

ユーザプログラムのHardwareSetup関数

```
void HardwareSetup(void)
{

}
```

HardwareSetup関数の修正（ソースファイル名：hwsetup.c）

システムクロックの設定はRXシリアルデバッガのHardwareSetup関数内に記述されているため、デバッグ中はデバッグ対象のユーザプログラムでシステムクロックの設定を行う必要がありません（正確にはシステムクロックの設定を行ってはなりません）。

ただし、デバッグが完了し、RXシリアルデバッガなしでユーザプログラムを単独実行させる場合、ユーザプログラムにシステムクロックの設定処理が必要となります。そこでRXシリアルデバッガののHardwareSetup関数内に記述されているシステムクロックの設定処理をユーザプログラムの HardwareSetup 関数内にコピーしてください。なお、その他の BRR の設定値や TxD 端子や RxD 端子の設定処理はコピーする必要がありません。システムクロックの設定のみコピーしてください。

その後、ビルドを行い、ユーザプログラムのロードモジュールを RFP（Renesas Flash Programmer）等で内蔵フラッシュメモリに書き込めば、ユーザプログラムを単独実行させることが可能となります。

付録 1

1

1.1 サポートMCU一覧

サポートMCU一覧

RX610グループ

RX62N、RX621グループ
RX62Tグループ
RX62Gグループ

RX630グループ
RX63N、RX631グループ
RX63Tグループ

RX210グループ
RX21Aグループ
RX220グループ

1.2　RXシリアルデバッガとユーザプログラムの共存方法

RXシリアルデバッガとユーザプログラムの共存方法

```
 1: #include "iodefine.h"
 2:
 3: #define  BRR    12     // Bitrate Register Value
 4:
 5: const unsigned char Brr = BRR;
 6:
 7: void HardwareSetup(void)
 8: {
 9:     SYSTEM.SCKCR.LONG = 0x00020100;          // If XTAL=12MHz Then
10:                                             // ICLK=96MHz,BCLK=24MHz,PCLK=48MHz
11:     if( !PORT0.PORT.BIT.B7 )                // Check Running Mode
12:         ((void(*)(void))0x********)( ); // Goto User Program
13:
14: // If Used SCI1 and Used TxD1-B,RxD1-B
15: //  IOPORT.PFFSCI.BIT.SCI1S = 1;           // Use TxD1-B,RxD1-B
16:
17: // If Used SCI2 and Used TxD2-B,RxD2-B
18: //  IOPORT.PFFSCI.BIT.SCI2S = 1;
19:
20: // If Used SCI3 and Used TxD3-B,RxD3-B
21: //  IOPORT.PFFSCI.BIT.SCI3S = 1;
22:
23: // If Used SCI6 and Used TxD6-B,RxD6-B
24: //  IOPORT.PFFSCI.BIT.SCI6S = 1;           // Use TxD6-B,RxD6-B
25: }
```

ユーザプログラム実行の判断
（判断の要因はユーザ固有）

ユーザプログラムの実行
********はユーザプログラムの先頭番地
通常はユーザプログラムをビルド時の
PResetPRGセクションの先頭番地

RXシリアルデバッガとユーザプログラムの共存方法

RXシリアルデバッガとユーザプログラムを共存させることは可能です。例えば、通常はRXシリアルデバッガを使用してデバッグを行い、動作確認後、そのままユーザプログラムを単独実行させたいのであれば、予めRXシリアルデバッガをビルドする際、HardwareSetup関数内に上記のような記述を追加してください。

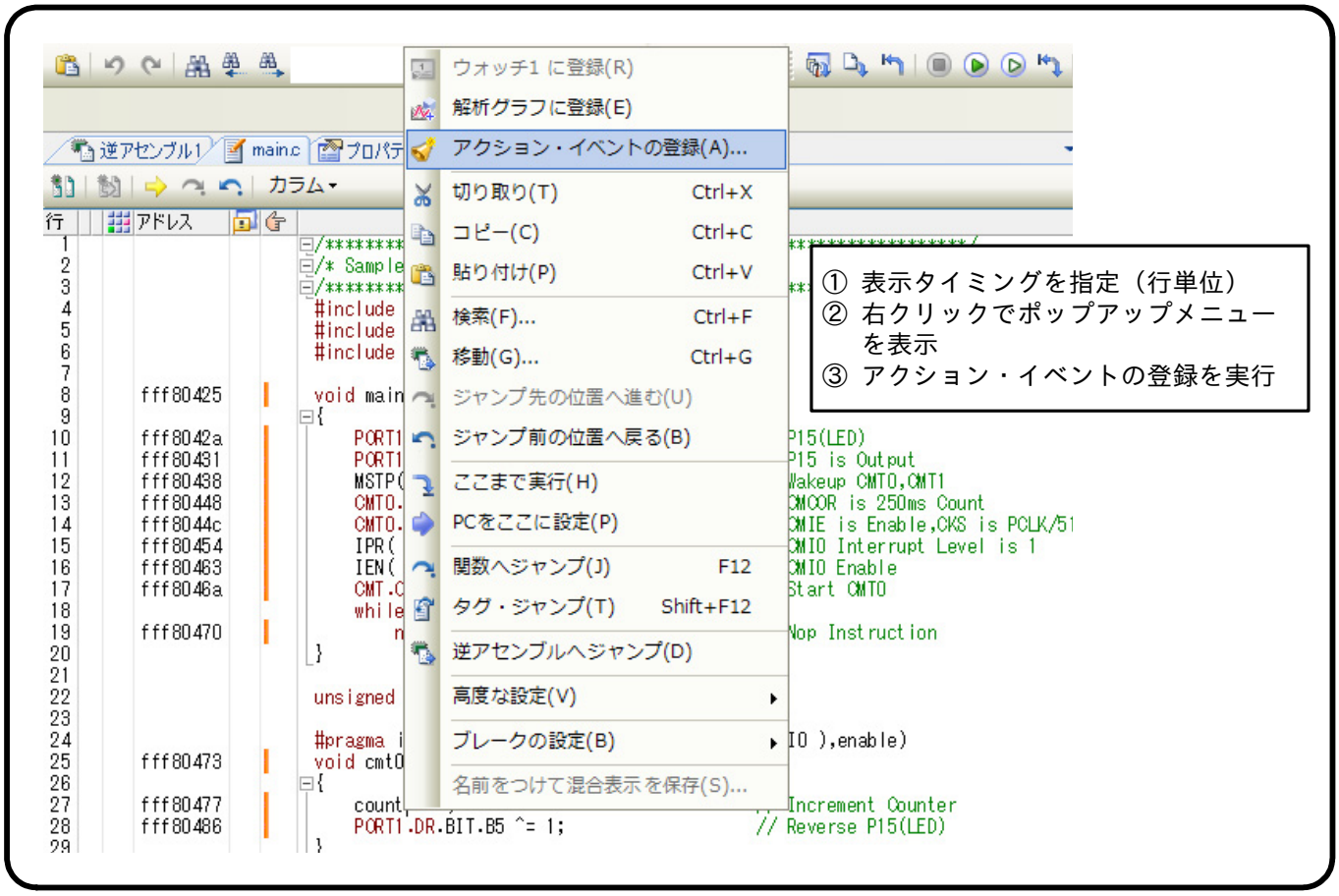
まず、11 行目は RX シリアルデバッガを起動するか、ないしはユーザプログラムを単独実行させるかの判断を行ってください。判断要因はユーザの自由となりますが、上記の例はP07に接続のSWで判断を行っています。もし、判断の結果、ユーザプログラムを単独実行させるのであれば、12行目のようにユーザプログラムの先頭に関数コールを行ってください。ユーザプログラムはRXシリアルデバッガとは別のプロジェクトでビルドを行っていますから、予めユーザプログラムの先頭番地、正確にはリセットベクタに登録したアドレス、通常の場合はPResetPRGセクションの先頭番地を調べておき、その番地に関数コールを行ってください。

【補足説明】

上記とは逆にユーザプログラムを単独実行中、RXシリアルデバッガを強制的に起動することも可能です。上記の12行目と同じようにRXシリアルデバッガをビルド時のPResetPRGセクションの先頭番地に関数コールを行えば、RXシリアルデバッガを起動できます。ただし、この方法の場合、強制停止したユーザプログラムに制御を戻すことはできません。ユーザプログラムの再起動はリセット状態からの実行となります。

1.3　ユーザプログラム実行中の変数表示（アクション・イベント）

ユーザプログラム実行中の変数表示（アクション・イベント）

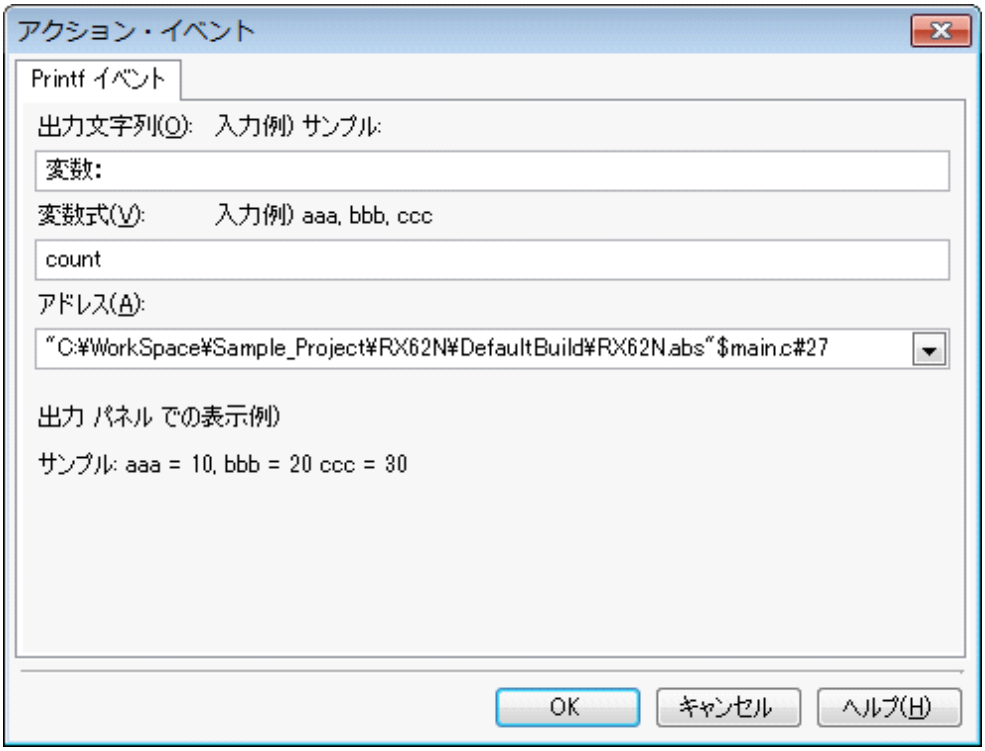


ユーザプログラム実行中の変数表示（アクション・イベント）

ユーザプログラム実行中、ユーザプログラム内の変数をprintf関数と同じような扱いでCubeSuite+の出力ウィンドウに表示することができます。なお、ユーザプログラム実行中の変数表示を行うための設定はRXシリアルデバッガに行う必要がありません。具体的にはCubeSuite+が持っているアクション・イベントの機能を使って実現します。

まず、上記のように表示タイミングを指定します。表示タイミングは行単位に指定可能です。次に右クリックでポップアップメニューを表示し、アクション・イベントの登録を実行します。

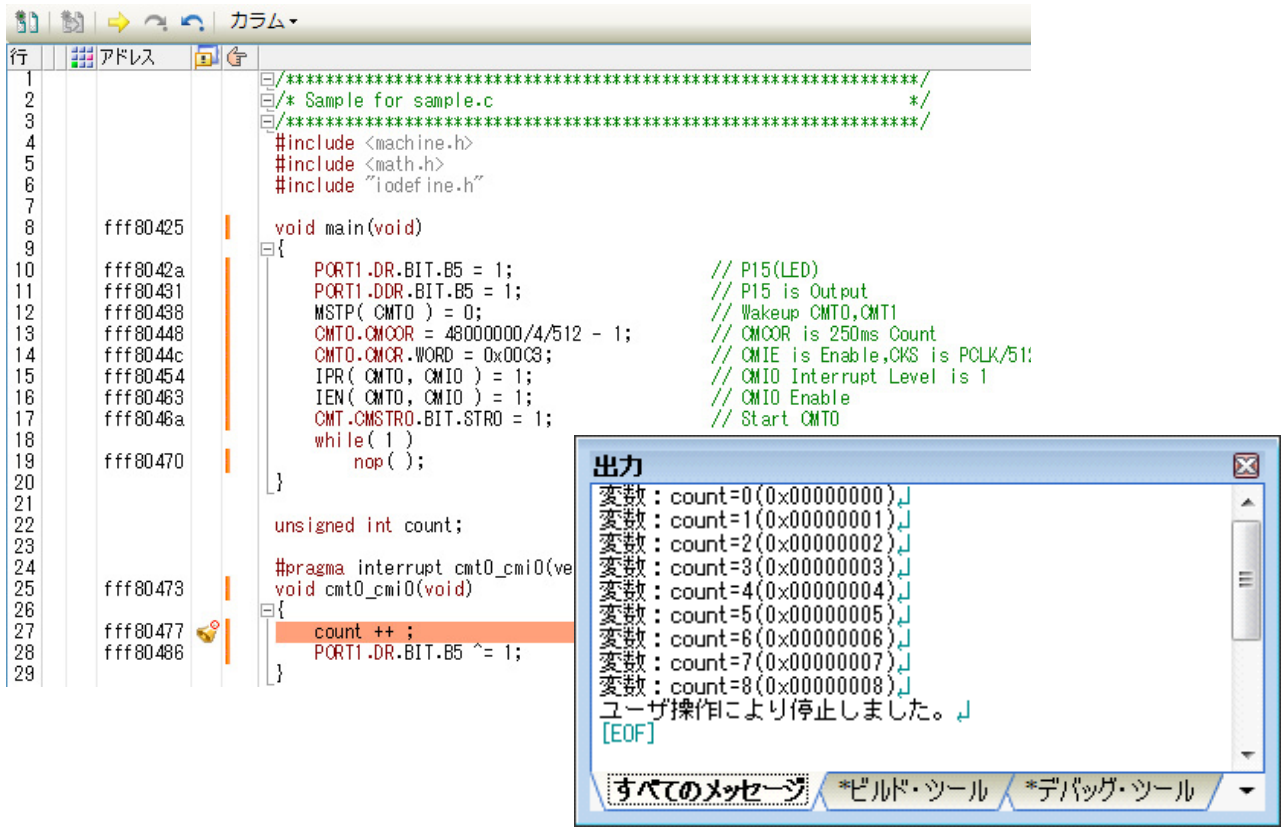
アクション・イベントの登録



アクション・イベントの登録

　アクション・イベントの登録を実行すると上記のようなダイアログが表示されます。ダイアログの指示に従い、出力文字列と変数式を入力します。ここでは外部変数の count の値を「変数：」の後に表示することとしましたが、一度に複数個の変数表示も可能となっています。

アクション・イベントの実行



アクション・イベントの実行

　アクション・イベントの登録するとソースプログラム上にはアクション・イベントが設定されていることを示すベルのマークが表示されます。また、その状態でプログラム実行すると出力ウィンドウには、プログラムが目的の行を走行時、指定した文字列形式で変数の値が表示されます。

【注意事項】

　アクション・イベントを登録、及びプログラムが目的の行を走行すると文字列表示のためにRXシリアルデバッガとCubeSuite+との間でデータの送受信が行われ、著しくユーザプログラムの実行に影響を与えます。このため、RXシリアルデバッガでのアクション・イベントの登録は、あまりお勧めできません。

CubeSuite+ 版 RX シリアルデバッガ　取扱説明書

発行年月日　2013 年 7 月 7 日 Rev.1.00
2014 年 4 月 7 日 Rev.2.00

発行　ルネサス エレクトロニクス株式会社
〒 100-0004 東京都千代田区大手町 2-6-2

CubeSuite+ 版 RX シリアルデバッガ

取扱説明書

ルネサス エレクトロニクス株式会社
東京都千代田区大手町2-6-2 日本ビル 〒100-0004