ICS++ library Ver.3.61
Function reference manual

For RZ/T1 series
on IAR EWARM 8.32.1

Index

1. はじめに

1.1. はじめに

本ドキュメントは、RZ/T1 用の ICS++ライブラリについて記述します。本ライブラリは、2019/3/20 時点では、W2002 でのみ動作します。

ICS++ RZ/T1 library for EWARM は、IAR システムズ株式会社と株式会社デスクトップラボとの共同開発品です。

1.2. サポートポート

本ライブラリでは、RZ/T1 に存在する SCIFA0, SCIFA1, SCIFA2, SCIFA3, SCIFA4 をサポートします。ただし、RZ/T1 の SCIFAx で割り当てられた全てのピンはサポートされていませんのでご注意をお願いします。

1.3. 注意事項

7. 本製品は、実験・開発段階用として設計されています。特に、交通システム（自動車、電車、船舶）、交通用信号機器、防災・防犯装置、各種安全機器、医療機器、生命維持機器、航空機器、原子力制御機器などに使用なさらないようお願いいたします。
8. 本資料に記載された弊社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他、諸条件につきましては、弊社提案範囲内でご使用ください。
9. 弊社は、弊社製品の品質および信頼性の向上に努めておりますが、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。弊社製品は、耐放射線設計については、行っておりません。弊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、社会的損害などを生じさせない様、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対策およびエージング処理等、機器またはシステムとしての保証をお願いいたします。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、お客様が製造、実験なさる最終の機器・システムとしての安全検証をお願いいたします。
9. 本資料の全部または一部を弊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固くお断りいたします。

2. ICS ハードウェアによる機能の違い

2.1. ICS / ICS++ シリーズの種類

ICS / ICS++シリーズには、多くの種類が配布／販売されています。ただし、本ライブラリは、W2002 のみで動作いたします。

※注意　RZ/T1 シリーズの ICS++ライブラリは、2019/3/20 時点では、W2002 でのみ動作します。

2.1.1. ICS++ W2002 シリーズ （現行販売機種）

光ファイバーで接続するタイプの新しい ICS++シリーズです。
0.5Mbps～8Mbps の範囲をサポートします。加えて、12ch モードをサポートしています。

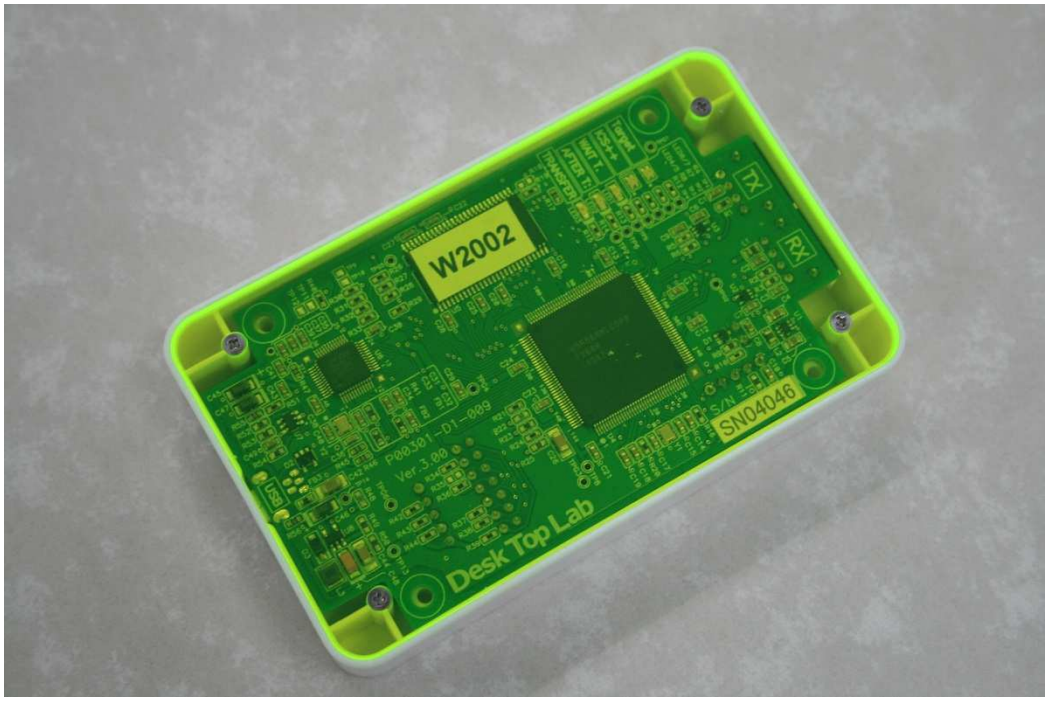


図 1　W2002　ICS++

2.2. 転送レートの設定方法（重要）

ライブラリを使用する際には、転送レートを決定する必要があります。通常は、可能な限り早い通信レートに設定する方が良いのですが、使用する ICS/ICS++のハードウェアや、使用する CPU の種類やクロック周波数により制約を受けます。通常は、以下の手順で最も高速な通信レートを設定してください。

2.2.1. ICS / ICS++ ハードウェアによる制約

W2002 では、0.5Mbps から 8Mbps までが通信可能な転送レートです。この制約の範囲内になるように、ICS++ライブラリの通信レートを設定してください。

2.2.2. ターゲット CPU／クロックによる制約

各 CPU や、実際に使用するクロック周波数、ライブラリのバージョンにより、設定可能な周波数が飛び飛びに存在しています。RZ/T1 の場合、以下のようになります。

$$\text{通信レート} = \frac{SERICLK}{8 \times (N+1)} \quad (0 < N < 255)$$

ここで、SERICLK は、実際に使用する CPU のクロック周波数。N は 0 以上の整数とします。

2.3. 実際のシステムにおける通信レート設定例

例 A） RZ/T1 SERICLK =150MHz

通信レートは、離散的な値を設定可能で以下の表のようになります。

N	通信レート	関数への設定値	DTLScope への設定 Communicationn clock 値
0	18.75Mbps	0	
1	9.375Mbps	1	
2	6.25Mbps	2	50000000
3	4.6875Mbps	3	37500000

0.5Mbps～8Mbps が選択できるので、通常は 6.25Mbps を選択します。

例 B） RZ/T1 SERICLK =120MHz

通信レートは、離散的な値を設定可能で以下の表のようになります。

N	通信レート	関数への設定値	DTLScope への設定 Communicationn clock 値
0	15Mbps	0	
1	7.5Mbps	1	60000000
2	5Mbps	2	40000000
3	3.75Mbps	3	37500000

0.5Mbps～8Mbps が選択できるので、通常は 7.5Mbps を選択します。

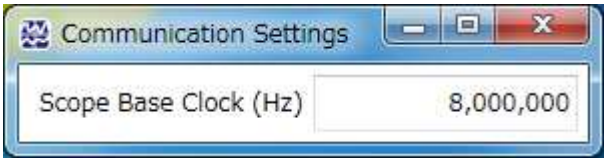
2.4. 通信レート決定用クロックの ICS++ハードウェアへの設定方法

本ライブラリを使用する場合、ＣＰＵ側のクロックの設定に従って、ICS++ボード上のクロックを下記のように選択してください。

2.4.1. DTLScope での周波数設定方法(W2002)

W2002 では、DTLScope.exe から通信レートの設定を行うことが可能です。

DTLScope.exe　を立ち上げ、
Settings -> Communication Settings
をクリックすると、下記のようなウィンドウが表示されます。
下記に、通信レートの８倍の数値を入力してください。



3. ICS++ライブラリの基本仕様・動作

3.1. 通信規約・ソースコード

ICS++のライブラリソースコードや詳細な通信プロトコルは、非公開となっております。ここでは、使用するに当たって重要な項目について説明します。

3.2. データ転送間隔の制限

ICS++では、ユーザー側の CPU からデータを転送するため、後述の ics2_watchpoint()関数を呼び出します。この関数を呼び出し方に、以下の制約があります。ICS++のモデルにより性能が異なるため、制約も異なります。モデルの確認方法は、DTLScope を立ち上げた時のステイタスバーに表示される名称です。

ICS++シリーズ　W2002 の場合
例：
　最小 54us　　(通信レート 7.5Mbps の場合)
　　最小時間 = 180/(通信レート[Mbps]) + 30 [us]　（ICS++ハードウェアの最大通信レートは 8Mbps）
　最大　5ms

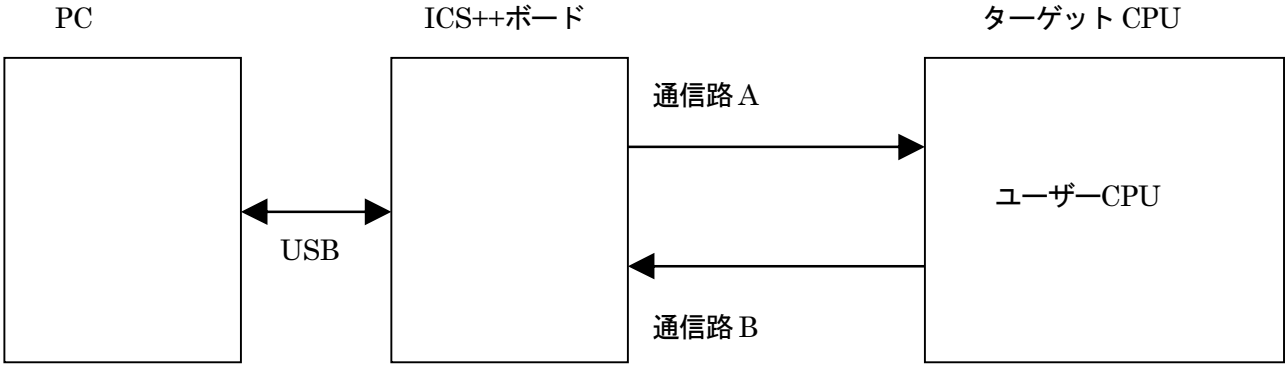


図 2　ICS++システムの通信路

本 ICS++には、データ転送間隔の制限があります。本制約は、図 2 の通信路 B の通信レート上限により発生します。後述のデータ転送関数　ics2_watchpoint()関数を呼ぶ度に、固定長のデータがターゲットから ICS++ボードに送られます。このデータ転送時間、ターゲット側の割り込みなどによる時間の遅れ、ICS++ボード側のオーバーヘッドなどから、転送間隔の最短時間制限が発生します。この時間以下になると、転送がうまく行われず、ICS++が正常な動作をしなくなることがあります。

ICS++の転送間隔の最短時間制限は、転送速度に大きく依存します。そのほかの通信速度については、各ライブラリ部分の記載を参照してください。また、ics2_watchpoint()関数の最大呼び出し時間間隔の制限もあり、ライブラリにかかわらず 5ms となっています。

3.3. 転送モード 0, 1, 2, 3, 4, 5, 6 の違い

ICS++には、2018 年 8 月現在、７種類の転送モードが存在しています。以下、(mode 0), mode1, mode2, mode3,(mode4), (mode5), (mode6)と呼びます。これらのモードの違いは、波形表示でサポートする最大ビット長と、１サンプリングのデータを何回の ics2_watchpoint()関数で転送するかの差です。(将来、この転送モードは拡張される予定があります)
RZ/T1 の標準ライブラリでは、mode 0, mode4, mode5, mode6 のモードはサポートされません。

１）mode 0　（１６ビット８チャンネル　１回転送モード動作）
（標準配布版 RZ/T1 ライブラリでは使えません。）
数値表示に関しては、８／１６／３２ビットのすべて型に対して動作します。しかしながら、波形表示に関しては型の制約があります。８ビットデータならば変数の型に応じて１６ビットに拡張し、１６ビットならばそのままで 8ch 分を１回で転送します。３２ビットデータは転送することはできません。通常、32bit　CPU ではサポートしません。**全 ICS モデルでサポートされます。**

２）mode 1　（３２ビット８チャンネル　2 回転送モード動作）
数値表示に関しては、８／１６／３２ビットのすべて型に対して動作します。ics2_watchpoint()関数が呼ばれると、波形表示に関しては、一度に指定された８ｃｈ分の８ビット、１６ビット、３２ビットデータを取り込みます。さらに、4ch 分のデータを転送します。次に ics2_watchpoint()関数が呼ばれた時、データを取り込まず、未転送の残り 4ch 分のデータを転送します。
つまり、３２ビット８チャンネルモードの場合には、２回の ics2_watchpoint()関数により、１回の８ｃｈ分の転送が行われます。**全 ICS モデルでサポートされます。**

３）mode 2　（３２ビット４チャンネル　１回転送モード動作）
数値表示に関しては、８／１６／３２ビットのすべて型に対して動作します。ics2_watchpoint()関数が呼ばれると、波形表示に関しては、毎回、指定された４ｃｈ分の８ビット、１６ビット、３２ビットデータを取り込みます。そして、その 4ch 分のデータを転送します。
つまり、３２ビット４チャンネルモードの場合には、１回の ics2_watchpoint()関数呼び出しにより、１回の４ｃｈ分の転送が行われます。５チャンネル以上の波形表示の機能はありません。

４）mode 3　（３２ビット１２チャンネル　３回転送モード動作）
数値表示に関しては、８／１６／３２ビットのすべて型に対して動作します。ics2_watchpoint()関数が呼ばれると、波形表示に関しては、一度に指定された１２ｃｈ分の８ビット、１６ビット、３２ビットデータを取り込みます。さらに、4ch 分のデータを転送します。次に ics2_watchpoint()関数が呼ばれた時、データを取り込まず、未転送のデータの内の残り 4ch 分のデータを転送します。さらに次に ics2_watchpoint()関数が呼ばれた時に、最後の 4ch 分のデータを転送します。
つまり、３２ビット１２チャンネルモードの場合には、３回の ics2_watchpoint()関数により、１回の８ｃｈ分の転送が行われます。**※注意　W2002 モデルでのみ使用可能です。**

５）mode 4　（８／１６ビット１５チャンネル　２回転送モード動作）
（標準配布版 RZ/T1 ライブラリでは使えません。）
数値表示に関しては、８／１６／３２ビットのすべて型に対して動作します。ics2_watchpoint()関数が呼ばれると、波形表示に関しては、一度に指定された１５ｃｈ分の８ビット、１６ビットデータを取り込みます。さらに、8ch 分のデータを転送します。次に ics2_watchpoint()関数が呼ばれた時、データを取り込まず、未転送のデータの内の残り 7ch 分のデータを転送します。
つまり、１６ビット１５チャンネルモードの場合には、２回の ics2_watchpoint()関数により、１回の１５ｃｈ分の転送が行われます。

※注意　このモードは、W2002 の Firmware Ver.1.2 以降のバージョンのみでサポートされます。

６）mode 5　（将来の予約）

７）mode 6　（１６ビットのみ８チャンネル　１回転送モード動作）

（標準配布版 RZ/T1 ライブラリでは使えません。）

このモードは mode 0 とほぼ同じですが、８ビットの変数に対する波形表示をサポートしません。そのかわりに、ics2_watchpoint()関数の実行時間が短くなっています。

数値表示に関しては、８／１６／３２ビットのすべて型に対して動作します。ics2_watchpoint()関数が呼ばれると、波形表示に関しては、一度に指定された１５ｃｈ分の１６ビットデータを取り込みます。さらに、8ch 分のデータを転送します。次に ics2_watchpoint()関数が呼ばれた時、データを取り込まず、未転送のデータの内の残り 7ch 分のデータを転送します。

※注意　全 ICS モデルでサポートされます。

表 1 転送モード

	メリット	デメリット
8/16bit 8ch 1 time mode （モード０）	波形情報更新間隔が短い 8 チャンネルの波形表示が可能	32bit の波形表示ができない
8/16/32bit 8ch 2 times mode （モード１）	32bit の波形表示が可能 8 チャンネルの波形表示が可能	波形情報更新間隔が 16bit の２倍
8/16/32bit 4ch 1 time mode （モード２）	32bit の波形表示が可能 波形更新間隔が短い	4 チャンネルしか波形を表示できない 16bit モードと同じ間隔
8/16/32bit 12ch 3 times mode （モード３）	32bit の波形表示が可能 12 チャンネルの波形表示が可能	波形情報更新間隔がモード１の 1.5 倍
8/16bit 15ch 2 times mode （モード４）	最大 15ch の波形表示が可能。	32bit の波形表示ができない
将来の予約 （モード５）		
16bit 8ch 1 times mode （モード６）	波形情報更新間隔が短い 8 チャンネルの波形表示が可能 ics2_watchpoint()関数の実行時間が短い	8bit, 32bit の波形表示ができない。

3.4.　数値表示ウィンドウ使用時の制約

ICS++では、数値表示と波形表示とを１本の通信路で共用しているため、数値表示と波形表示とを同時に行う場合、波形表示の制約が発生します。

波形表示を行っており数値表示が行われていない場合、波形データは毎回送信されるので、データはそのまま表示されます。しかしながら、数値表示と波形表示とが同時に行われている場合、数十 ms に１サンプリング分だけ波形が更新されず、表示される波形の一部が平らになる場合があります。データ測定をする場合など、このような状況が適当でない場合、一時的に、ICS++のオートリフレッシュ機能を停止してください。

3.5. ファイル構成・ライブラリ

RZ/T1　ICS++ライブラリは以下のような３つのファイルの構成になっています。

ヘッダファイル
ics2_RZT1.h
ics2_RZT1.o

通常関数として、
void　ics2_init(uint32_t port, uint8_t　int_level, uint8_t speed,　uint8_t mode);
void　ics2_watchpoint(void);
uint32_t　ics2_version(void);

※注意１
使用する UART ポートが異なっても、受信割り込み処理関数名は同じ名称です。ご注意ください。
(標準の RZ／T1　ICS++ライブラリでは、割り込みは使用しません)

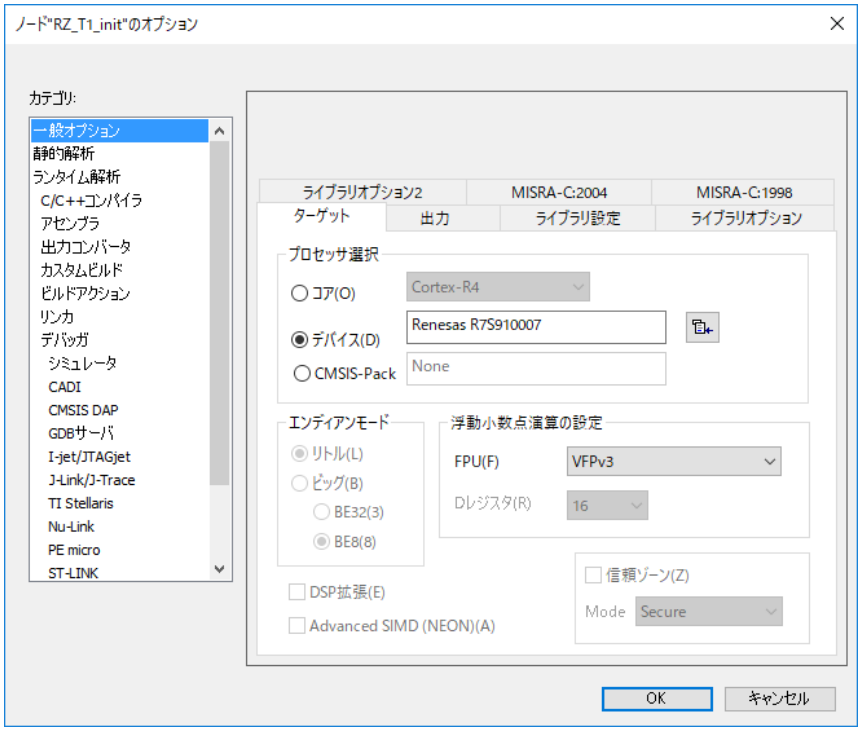
※注意２
標準ライブラリのコンパイラ・アセンブラ・リンカーのオプションスイッチは、プロジェクトをデフォルトで生成した状態を利用しています。お客様のプロジェクトにおいてご使用になるメモリーモデル、エンディアン、レジスターモード、などを変更していた場合、ICS++ライブラリの一部、もしくは、全部の機能が動作しない場合があります。お使いになる予定のコンパイラースイッチの状態をご確認の上、ご使用になってください。

3.6. 開発環境の設定

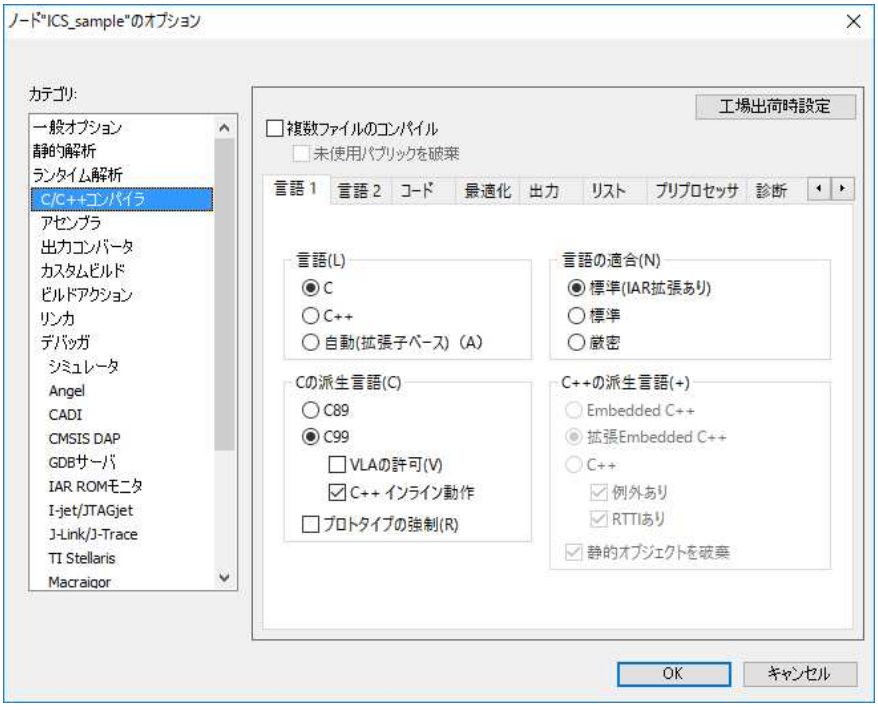
開発環境の設定オプションについて記載します。本ライブラリでは、下記の設定でコンパイル使用することを想定しています。

3.6.1. 一般オプション → ターゲット

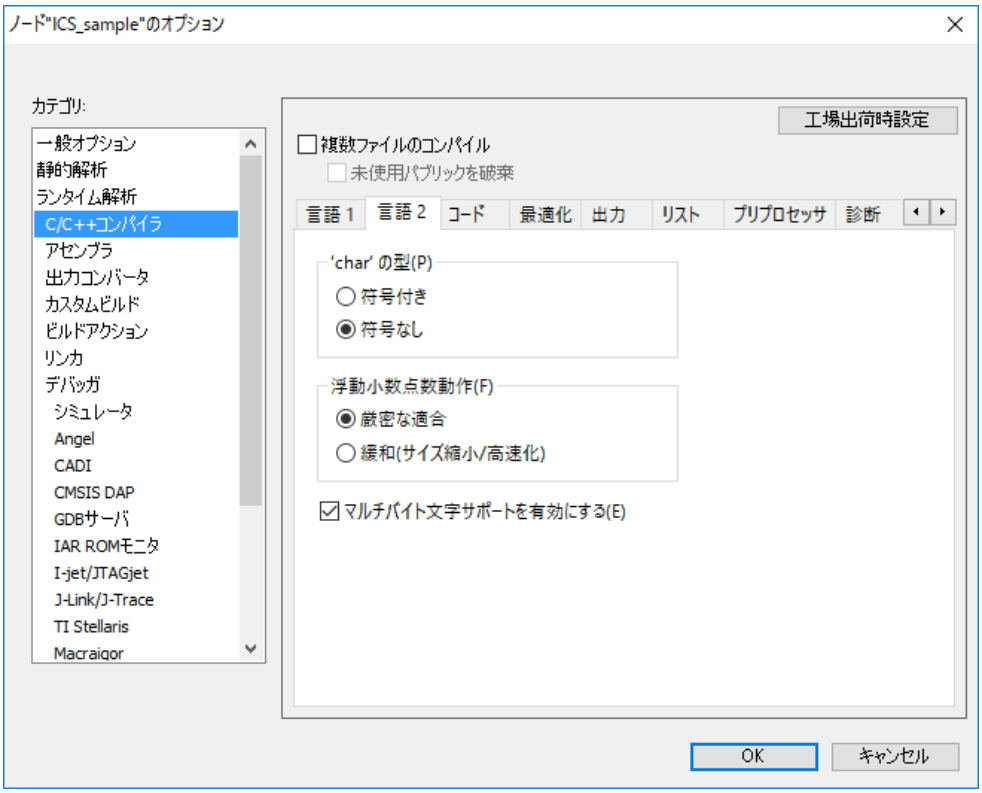
選択する CPU は、お客様のご使用になる CPU のモデルに合わせてください。



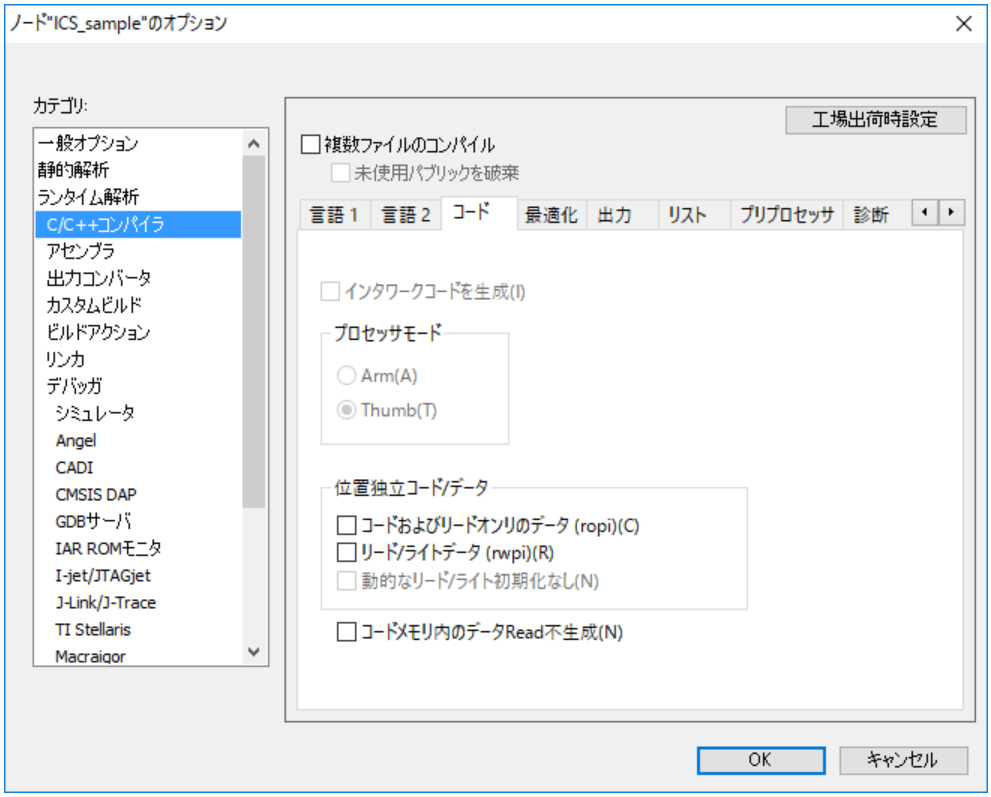
3.6.2. C/C++コンパイラ → 言語 1



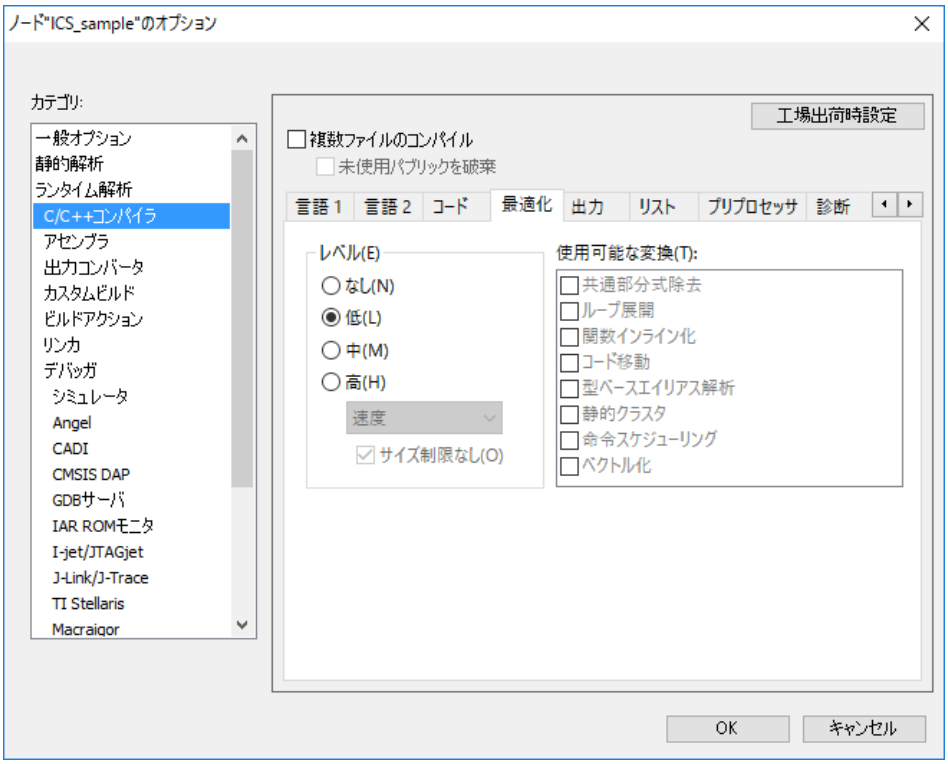
3.6.3. C/C++コンパイラ → 言語２



3.6.4. C/C++コンパイラ → コード



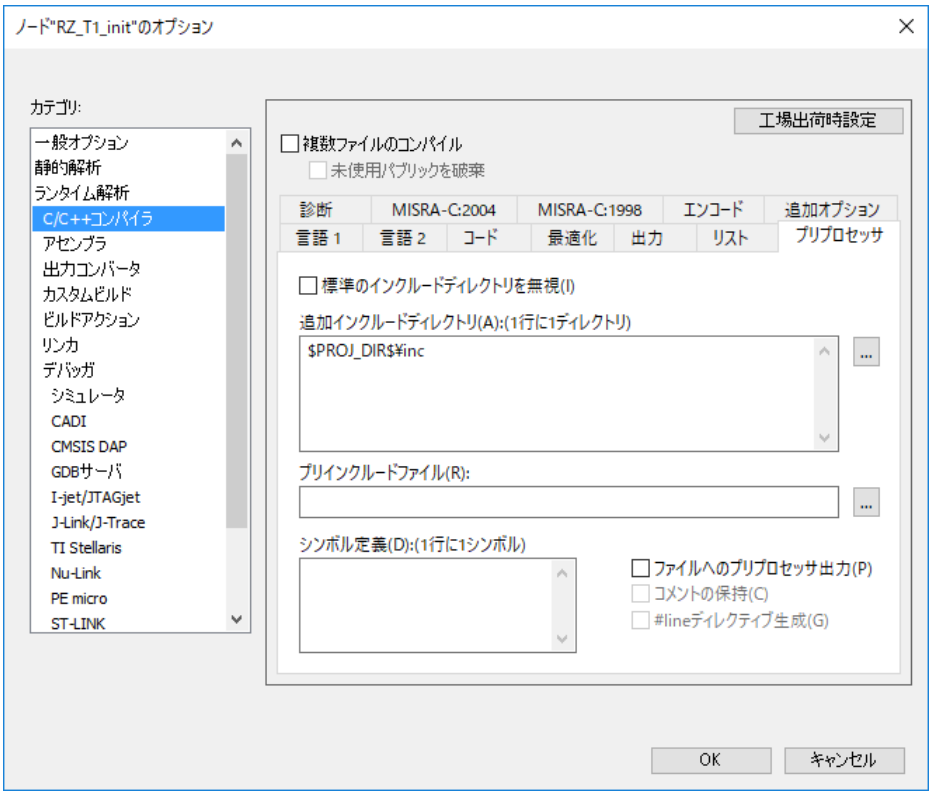
3.6.5. C/C++コンパイラ → 最適化



本ライブラリは、「最適化：低」で動作確認を行っております。最適化レベルを上げる場合、アプリケーションによっては、マニュアル通りの動作にならない場合がありますので、ご了承ください。

3.6.6. C/C++コンパイラ → プリプロセッサー

追加インクルードディレクトリは、お客様の環境に合わせてください。

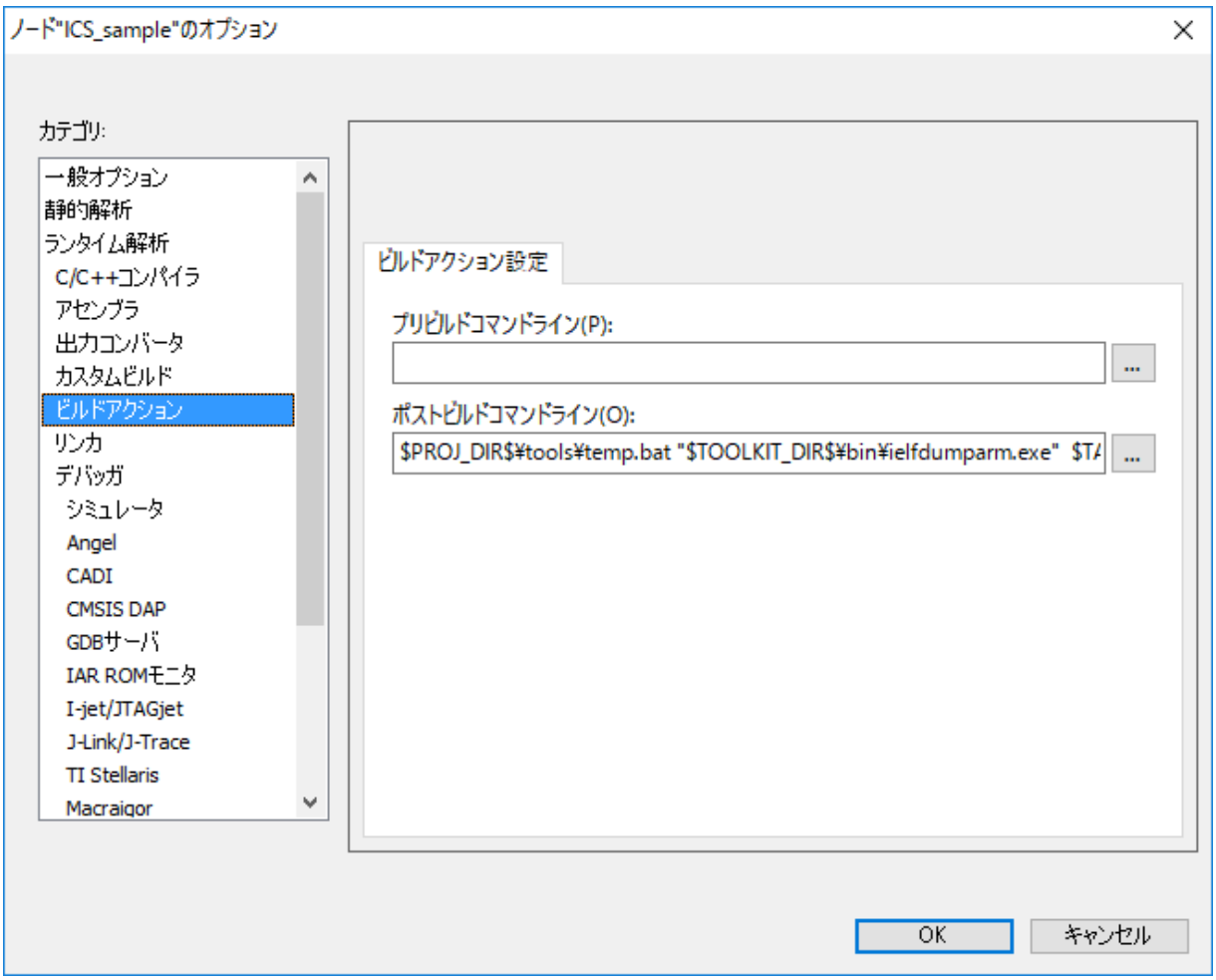


3.6.7.　ビルドアクション　→　（ポストビルドコマンドライン）

この行は、ICS を使用する際に変数情報を必要としますが、その変数情報を生成するためのツールを起動するための設定です。下記のの５行に書かれている文字列を　一行にしてポストビルドコマンドラインの行に入れてください。また、$PROJ_DIR$の下のフォルダーに、サンプルプログラムに入っている Tools のフォルダー内の実行ファイル３つと、bat ファイル１つをユーザープログラムに入れてください。

$PROJ_DIR$¥tools¥temp.bat
"$TOOLKIT_DIR$¥bin¥ielfdumparm.exe"
$TARGET_PATH$
$PROJ_DIR$
$TARGET_BPATH$

※注意　Tools フォルダー内の実行ファイル ewb1.exe は、Notron internet security がインストールされていると、ウイルスがあると誤検知し、削除される場合があります。その場合、Notron ユーティリティーで検索するフォルダーを除外するなどの対策を行うようにしてください。



4. 使用資源・ライブラリの説明

4.1. RZ/T1 シリーズ

4.1.1. RZ/T1 使用資源

CPU 名	RZ/T1 シリーズ
開発環境	IAR 社　EWARM　Ver.8.32.1
Library version	Ver.3.61
通信レート	ライブラリの設定可能通信レート 0.5Mbps～7.5Mbps 通信レート＝ $\frac{SERICLK}{8\times(N+1)}$　$(0 < N < 255)$ 標準通信レート　SERICLK = 150MHz,　speed=2　の場合 6.25Mbps ご使用になる ICS / ICS++の通信レート範囲に合致するように、設定してください。
サポートポート	SCIFA0: (TXD0: P40, RXD0: P24) SCIFA0: (TXD0: P40, RXD0: P42) SCIFA1: (TXD1:PE5, RXD1:PE6), (TXD1:P72, RXD1:P73) SCIFA2: (TXD2:PS7, RXD2:PS6), (TXD2:P91, RXD2:P92), (TXD2:PA5, RXD2:PA4) SCIFA3: (TXD3: PU3, RXD3: PU2), SCIFA4: (TXD4: PM0, RXD4: PU7)
ライブラリ	ics2_RZT1.o
ヘッダファイル	ics2_RZT1.h

CPU 使用リソース	サポート変数タイプ
・内部使用リソース SCIFA1 の場合 SCIFA1 内のレジスタすべて	数値表示・設定 8bit 符号なし　整数型 8bit 符号あり　整数型 16bit 符号なし　整数型 16bit 符号あり　整数型 32bit 符号なし　整数型 32bit 符号あり　整数型 32bit IEEE754 浮動小数点型 8bit　BOOL 型 8bit　LOGIC 型 波形表示 8bit 符号なし　整数型 8bit 符号あり　整数型 16bit 符号なし　整数型 16bit 符号あり　整数型 32bit 符号なし　整数型 32bit 符号あり　整数型 32bit IEEE754 浮動所数点型

4.1.2. RZ/T1 関数説明

初期化関数 void ics2_init(uint32_t port, uint8_t int_level, uint32_t speed, uint8_t　mode);
本関数内部で、ピン定義を含む ICS++関連の初期化を行います。本関数の初期化後に、前項で記載された ICS++で使用する資源ピンの定義や、スタンバイコントロールレジスタなどの設定を破壊しないように注意してください。 第１パラメータ 　SCI のポート番号や、SCI の使用するピンを設定します。このパラメータは、ICS2_RZT1.h 内で定義されている文字列を使用してください。 第２パラメータ 　現在のライブラリでは使用しません。 第３パラメータ 　ICS++システムで利用する通信レートを定義します。通信レートの計算方法は以下の通りです。 通信レート＝ $\frac{SERICLK}{8\times(N+1)}$　$(0 < N < 255)$ 通信レートは、0.5Mbps~8Mbps の間になるように設定してください。 各機種のサポート範囲に収まるように speed の値を設定します。 設定値は、通信レートを直接記述してください。 第４パラメータ 　通信モードの設定 　１：32bit８チャンネル同時サンプリング・２回転送モード 　　（２回の ics2_watchpoint()関数呼び出しで１回のサンプリング） 　２：32bit４チャンネル同時サンプリング・１回転送モード 　　（１回の ics2_watchpoint()関数呼び出しで１回のサンプリング） 　３：32bit１２チャンネル同時サンプリング・３回転送モード 　　（１回の ics2_watchpoint()関数呼び出しで１回のサンプリング） 通常は、１を設定します。波形更新レートを速くしたい場合には２を設定します。ただし、波形表示のチャンネル数は、４チャンネルになります。
転送関数の呼び出し　　void　　ics2_watchpoint(void);
本転送関数は、データの転送セットアップ用の関数です。通常は、キャリア割り込み内部に置きます。ただし、サンプルソフトでは、記述方法をわかりやすくするために、メインルーチン内に記述しています。 　本関数は、ＰＣで指定された変数のデータを読み出し、ＤＴＣ用の転送バッファーにコピーします。 　この関数は、通信速度が 6.25Mbps の際最小 58.8us 以上、最大 5ms の間隔を保って呼び出すようにしてください。通信速度が 1Mbps 以外の場合には、次の式で定義される時間を置いて呼び出してください。 最小通信間隔 $=1/(\text{通信速度}[bps])\times 180+30[us]$ ※注意 : ユーザソフトウェアでの割り込み間隔は、他の割り込みの関係で、割り込みの発生が遅延する場合があります。割り込みタイミングがずれることにより ics2_watchpoint()関数も呼び出しタイミングがずれます。これらを考慮して、呼び出すようにしてください。

バージョン読み出し関数　int　　ics2_version(void);
上位 16bit　　バージョン整数部 下位 16bit　　バージョン小数部
使用割り込み関数
本ライブラリでは、割り込みは使用しません。

4.1.3. RZ/T1　関数使用方法

ICS++を使用するためのユーザープログラムの設定方法を、付属のサンプルソフトを例に説明します。
ここでは、RZ/T1 の　SCIFA4 を利用した場合の例を示します。

１）ics2_init()を下記のように呼び出す。

初期化関数
ics2_init(ICS_SCIFA2_P91_P92, 0, 2, 1);
を初期化部分に入れてください。
第 1 パラメータは、使用するポートを ICS2_RZT1.h から選択してください。
第 2 パラメータは、0 を設定してください。
第 3 パラメータは、通信速度に応じて選択してください。SERICLK=150MHz の場合、通常は 2 にします。
第 4 パラメータは、転送モードに応じて選択してください。通常は、１にしてください。

---------- List 1　main.c ----------

```
void main(void)
{
    ics2_init(ICS_SCIFA2_P91_P92,  0,  2, 1);

    while(1)
    {
        /* no code */
    }
}
```

２）ics2_watchpoint()関数の組込み
ics2_wathpoint()は、この関数を呼び出した時点の変数の値を送信します。
使用法としては、通常は、キャリア割り込みの内部で呼び出します。 また、この関数は、各 ICS ユニットで可能な時間間隔で呼び出すようにしてください。割り込み処理関数が指定間隔で呼び出される場合、List2 のように ics2_watchpoint()の呼び出しを間引くようにソフトウェアを組んでください。

---------- List 2　ics2_watchpoint()の間引き ----------

```
int     deci = 0;

void    Timer_ISR(void)      /* 50us間隔 */

{
        deci = deci + 1;
        if (deci >=1)
        {
            deci = 0;
            ics2_watchpoint();
        }
}
```

5. サンプルプログラムの使用方法

5.1. サンプルプログラムの概要

本サンプルプログラムは、内部クロック 600MHz で動作するようになっています。従って、対応する CPU シリーズならば、比較的容易にどの CPU でも変更できるような設定がされています。

項目	仕様
クロック	600MHz
通信レート	SERICLK = 150MHz 6.25Mbps　　初期化関数によって、変更可能
CPU	
動作確認 ハードウェア	RZ/T1
ICS 設定ポート	初期設定　SCIFA2　P91, P92
使用リソース	初期設定　SCIFA2 のみ　　(DMA, 割り込みは不使用)
エミュレータ	i-jet

5.2. 開発環境の設定

使用する CPU に合わせて、開発環境を設定する必要があります。
『3.6.1　一般オプション　→　ターゲット』
『3.6.6　C/C++コンパイラ　→　プリプロセッサー』
『エラー! 参照元が見つかりません。　エラー! 参照元が見つかりません。』
に記載された内容に従って、修正してください

5.3. ソースコードの修正

ICS に接続するポートに従って、プロジェクト中の ics2_init()のパラメータを修正する必要があります。
デフォルトは、ics2_init(ICS_SCIFA2_P91_P92, 0, 2, 1);
になっていますが、使用するポートが異なる場合には、ics2_RZT1.h の定義ファイルの中から
適切なポートを含む文字列を選択してください。
文字列は、以下の構成の物から選択する必要があります。

ICS_AAAA_BBBB

AAAA：UART ポート名　SCIFA0, SCIFA1, SCIFA2, SCIFA3, SCIFA4

BBBB：ピン名：P91, P92 など

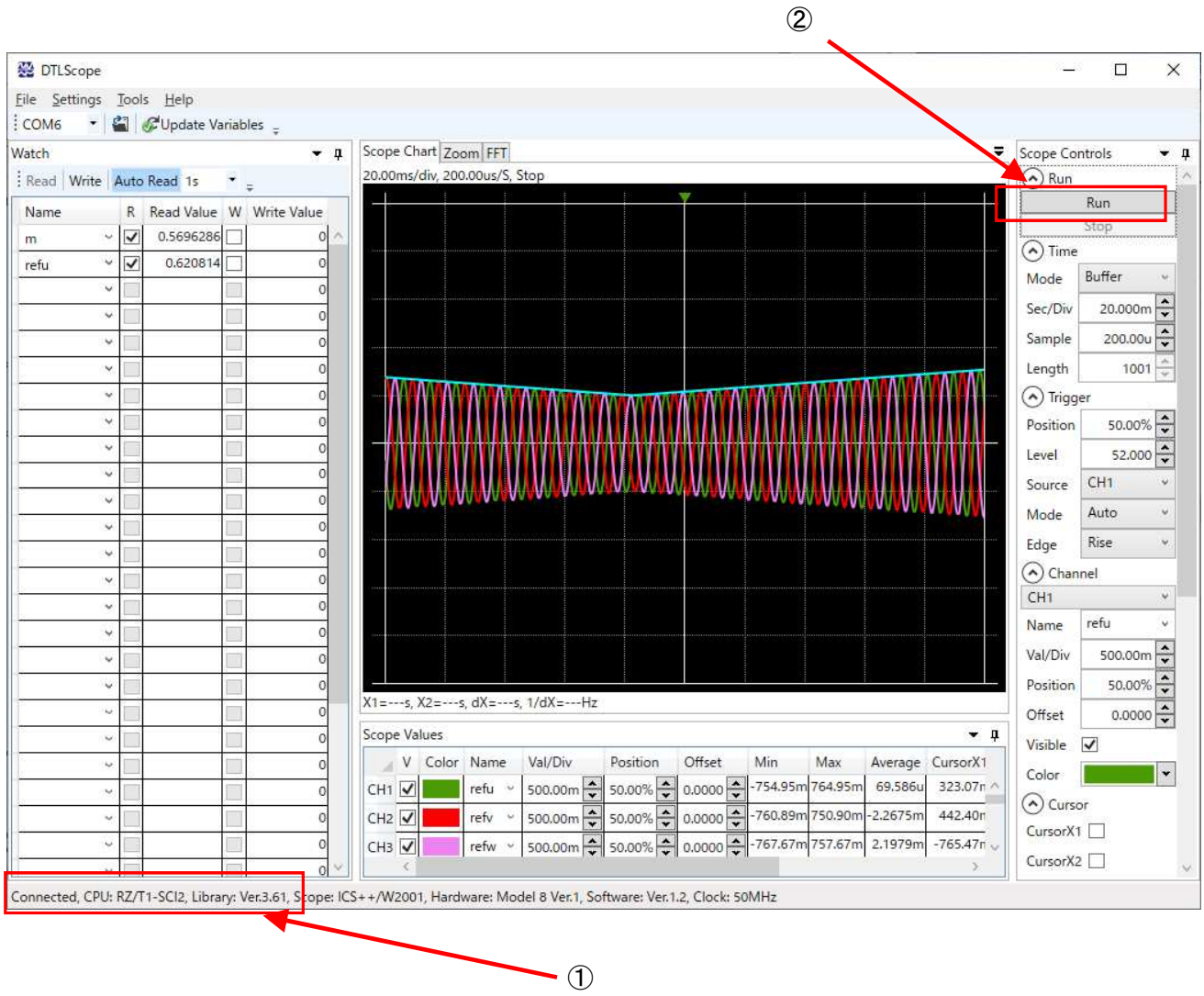
5.4. DTLScope の実行

ICS++ (W2002)のハードウェアを接続し、DTLScope を立ち上げてください。
この時点では、通常はターゲットが認識されません。
以下の操作をしてください。
File → Open → (サンプルプロジェクトの中の) ICS_sample¥sample.dtlsp
をロードしてください。
適切に設定されていれば、DTLScope の左下に
“Connected, CPU:RZ/T1-SCI2, Library: Ver.3.61”などと表示されます。————①

通信レートが変更されている場合、設定通信レートの８倍の数値を
DTLScope の
Setting -> Communication Setteings に設定してください

RZ/T1 SERICLK = 150MHz, 通信レート 6.25Mbps の時、6.25M x 8 = 50000000 を設定します。

この状態で、DTLScope の右側の[RUN] ボタンを押すと、下のような波形が表示されます。————②



6. 改訂履歴

バージョン	変更日	変更内容
Ver.1.00	2019/03/20	・初版作成、RZ/T1 用

ICS++ Library　Function manual

発行年月日　2019 年 3 月 20 日　　Ver.1.00JP

発行　　IAR システムズ株式会社
株式会社デスクトップラボ