Desk Top Lab

ICS++ library RH850 Ver.3.61
Function reference manual
For RENESAS CC-RH Compiler

Index

1. はじめに

1.1. はじめに

本ドキュメントは、ICS シリーズ W1001, W1002, W1003, T2001A/B、T2002A/B、および、ICS++ シリーズ W1004, W2001, W2002, W2203, W3002, W3203, T2001C, T2006A 用関数マニュアルです。

1.2. 注意事項

1. この資料に記載されたすべての情報は、本資料発行時点の物であり、予告なく変更することがあります。弊社製品のご購入およびご使用にあたりましては、必ず最新の資料を参照していただけるようお願いいたします。
2. 本資料に記載された弊社製品、技術情報の仕様に関連し発生した第三者の特許権、著作権、その他の知的財産権の侵害に関し、弊社は一切その責任を負いません。弊社は、本資料によって弊社または第三者の特許権、著作権、その他の知的財産権を許諾するものではありません。
3. 弊社製品の複製等を行わないでください。
4. 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、インバータ製品の動作例、応用例を説明するための物です。お客様の機器の設計、実験において、回路、ソフトウェアおよびこれ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。これらの仕様に起因して、お客様または、第三者に生じた損害に関し、弊社は一切その責任を負いません。
5. 輸出に際しては、「外国為替および外国貿易法」その他、輸出関連法令を順守し、かかる法令の定めるところにより必要な手続きを行ってください。本資料に記載されている弊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、その他軍事用途の目的で使用しないでください。また、弊社製品および技術を国内外の法令および規制により製造・使用・販売を禁止されている機器に使用することはできません。
6. 本資料に記載されている情報は、正確を期すために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よる損害がお客様に生じた場合においても、弊社は、一切その責任をおいません。
7. 本製品は、実験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。

ICS++は、株式会社デスクトップラボの製品です。

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、下記のように、多くの種類が配布／販売されています。
下記の説明に応じて、シリーズ名を把握して以下の関数の説明をお読みください。

2.1.1. ICS++ W2002 シリーズ

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、Max 15ch モードをサポートしています。

図 1 W2002 ICS++

基板上に W2002 とシールで記載されています。初期出荷分の一部のロットには、シールがない物もあります。これらの場合には、基板上のシルクで記載された番号、もしくは、PC 上のソフト DTLScope で表示される型番で判別が可能です。
シルクでの判別： P00301-D1-009 と記載がある場合には、W2002 となります。

ファームウェアのバージョンにより、波形表示チャンネル数が異なります。

Frimware version	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6
1.0	○	○	○	○	×	×	○
1.2	○	○	○	○	○	×	○

モードは、ics2_init()関数で設定されるモードです。

2.1.2. T2001C / T2006A 搭載 サブセット ICS

T2001C / T2006A に搭載された ICS は、W2002 シリーズに分類されます。
W2002 との主な違いはメモリー長で、T2001C / T2006A との違いは 2 点あります。

1) レコード長が 1024 点まで
2) 波形表示チャンネル数が 8ch まで

以上のように機能が制限されています。

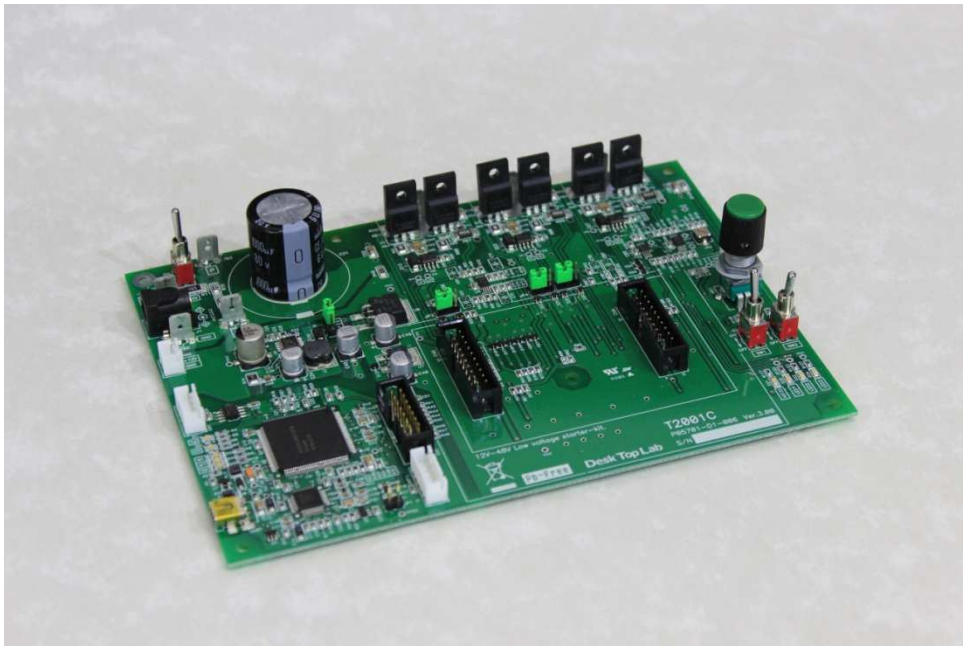


図 2 T2001C 低電圧インバータ（T2001B の後継機種）

図 3 T2006A 低電圧インバータ（三相インバータ 3ポート版）

2.1.3. ICS++ W1004 シリーズ（販売終了品）

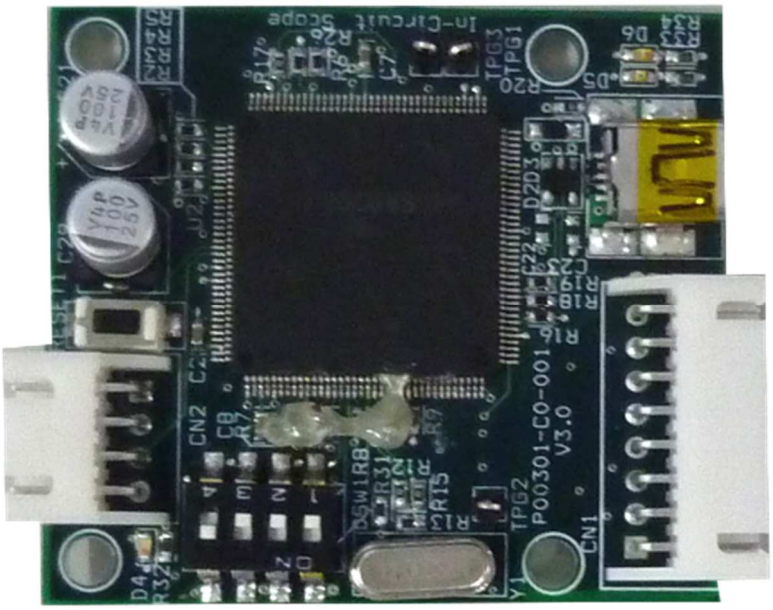
光ファイバーで接続するタイプの ICS++シリーズです。

このタイプの ICS++をお持ちの方で、Firmware が Ver.1.35 以前の方は、弊社にお送りしていただければ、無償で Ver.1.52 以降に Version UP させていただきます。Version UP することにより、ターゲット CPU の通信レートが 0.5Mbps～1.25Mbps までの範囲だったものが、0.5Mbps～1.25Mbps の範囲に加えて、1.5Mbps, 3Mbps をサポートできるようになります。

2.1.4. ICS W1001 シリーズ（販売終了品）

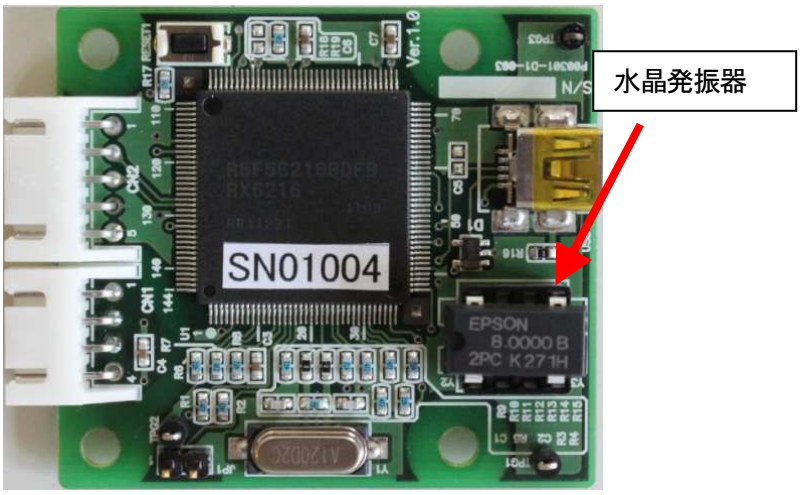
W1001

下記の写真のような、1Mbps 固定タイプの ICS です。



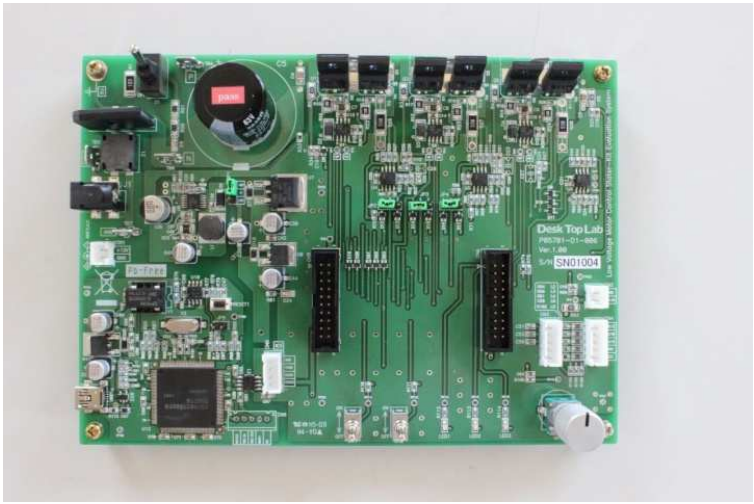
2.1.5. ICS W1003 シリーズ（販売終了品）

下記の写真のような、ボード上のソケットに実装された水晶発振器を交換して、通信レートを固定するタイプの ICS です。



2.1.6. T2001B / T2002B 搭載 サブセット ICS（販売終了品）

T2001B / T2002B に搭載された ICS は、W1003 シリーズに分類されます。
T2001B, T2002B に搭載された ICS は、W1003 のサブセットとなっています。
お試し版との位置づけのツールのため、レコード長が 1024 点までと非常に短くなっています。



2.1.7. ICS++ W2001 シリーズ （販売終了品）

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～3.2Mbps の範囲に加えて、3.75Mbps, 5Mbps をサポートします。
一般販売はしておりません。

2.2. 各シリーズの機能の差

表 1 各 ICS / ICS++仕様

	ICS series W1001 (販売終了) Renesas MRSSK	ICS series W1003 T2002B T2001B (販売終了)	ICS++ series W1004 (販売終了)	ICS++ series W2002 (現行品) W2203(現行品) T2001C (現行品) T2006A (現行品)	ICS++ series W3002 W3203
通信レート	1Mbps 固定	0.5Mbps ～ 1.25Mbps (固定)	○Ver.1.35 以前 0.5Mbps～1.25Mbps ○Ver.1.52 以降 0.5Mbps～1.25Mbps 1.5Mbps, 3.0Mbps	0.5Mbps～**8Mbps**	0.5Mbps～**25Mbps**
波形表示 チャンネル数	8ch	8ch	8ch	**W2002: Firm V1.0** **32bit 12ch** **16bit 8ch** **W2002: Firm V1.2** **32bit 12ch** **16bit 15ch** T2001C, T2006A は 8ch	**32bit 15ch** **16bit 15ch**
サポート転送モード	32bit 8ch 2 分割 16bit 8ch 1 回	32bit 8ch 2 分割 16bit 8ch 1 回	32bit 4ch 1 回 32bit 8ch 2 分割 16bit 8ch 1 回	32bit 4ch 1 回 32bit 8ch 2 分割 32bit 12ch 3 分割 16bit 8ch 1 回 16bit 15ch 2 分割	32bit 4ch 1 回 32bit 8ch 2 分割 32bit 12ch 3 分割 16bit 8ch 1 回 16bit 15ch 2 分割 **32bit 15ch 1 回** **32bit 15ch 1 回**
絶縁の方法	IC による絶縁	IC による絶縁	**光ファイバー**	**W2002: 光ファイバー** T2001C T2006, W2203 は IC 絶縁	**W3002 光ファイバー** **W3203 IC 絶縁**
USB 転送速度	11Mbps	11Mbps	11Mbps	**480Mbps**	**480Mbps**
ロールモード 波形をスクロールしながら表示するモード	なし	なし	**サポート**	**サポート**	**サポート**
信号発生機能 任意波形を CPU 上の変数に書き込んで、実機試験やデモ運転を行う機能	なし	なし	サポート DTLScope 1.5.使用時	**サポート** **DTLScope 1.5 使用時**	**サポート** **DTLScope 1.5 使用時**
対応 PC soft	InCircuitScope DTLScope	InCircuitScope DTLScope	DTLScope.exe	DTLScope.exe	DTLScope.exe

2.3. 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。

2.3.1. ICS / ICS++ ハードウェアによる制約

『表 1　各 ICS / ICS++仕様』に示されるように、各ハードウェアにより、通信可能な最高転送レートが異なります。この制約の範囲内になるように、通信レートを設定してください。

2.3.2. 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により、設定可能な周波数が飛び飛びに存在しています。RH850F1KM-S1 の場合、以下のようになります。RLIN クロック（C_ISO_LIN)をクロックに使用します。
クロック設定は、ICS++ライブラリの本体(ICS2_RH850F1KM-S1.o)では行いません、ユーザー側のクロックの設定、および、サンプルプログラムの初期化関数 ICS2_RH850F1KM-S1_init.c を参照してください。
通信レートは、以下の式で決定されます。

$$通信レート = \frac{C\_ISO\_LIN}{6 \times (speed + 1)} [Mbps] \quad speed \in [0, 1, 2, \ldots 39]$$

ここで、C_ISO_LIN は、実際に使用する RH850F1KM-S1 の SCI のクロック周波数。speed は、0 以上の整数値です。

2.4. 実際のシステムにおける通信レート設定例

例 A）　RH850F1KM-S1　S_ISO_LIN = 40MHz の場合、
通信レートは、以下表のようになります。

Speed	通信レート	DTLScope の設定クロック
0	40/6Mbps=6.667Mbps	53.333MHz
1	40/12Mbps=3.333Mbps	26.667MHz
2	40/18Mbps=2.222Mbps	17.778MHz
3	40/24Mbps=1.667Mbps	13.333MHz
4	40/30Mbps=1.333Mbps	10.667MHz
5	40/36Mbps=1.111Mbps	8.889MHz
6	40/42Mbps=0.952Mbps	7.619MHz

ICS モデル毎の推奨転送レート

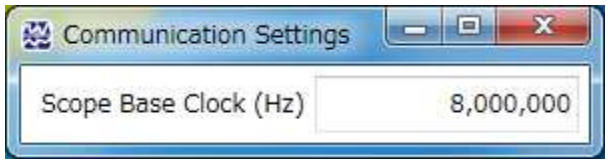
ICS / ICS++ model	対応転送レート	推奨転送レート
W3002 / W3203	0.5Mbps ～ 25Mbps	6.667Mbps
W2002 / W2203	0.5Mbps ～ 8Mbps	6.667Mbps
W1004　V1.52 以降	0.5Mbps ～ 1.25Mbps, 1.5Mbps, 3Mbps	1.111Mbps
W1004　V1.52 以前	0.5Mbps ～ 1.25Mbps	1.111Mbps
W1003	0.5Mbps ～ 1.25Mbps	使用不可
W1001	1Mbps fix	使用不可

2.5. 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。

2.5.1. W1004, W2001, W2002, W2203, W3002, W3203, T2001C, T2006A の場合

可変クロックを内蔵しているため、PC 側からの操作が可能となります。
PC ソフト(DTLScope.exe)で通信レートの８倍の周波数を設定してください。
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。



2.5.2. ICS　W1001 の場合（ケースがない ICS、水晶発振器のソケットがないタイプ）

クロックが固定されているため、通信クロック 8MHz 以外のクロックでの使用はできません。

2.5.3. ICS　W1003 の場合（ケースがない ICS、水晶発振器のソケットがあるタイプ）

ボード上のソケットに実装されている水晶発振器を交換することにより、クロックを変更することが可能です。通信レートの８倍の周波数の水晶発振器モジュールに交換してください。
設定するクロック周波数の計算方法は、下記の通りです
通信レートを 1.25Mbps 以下に設定する必要があります。
選択したクロックの８倍の水晶発振子をボード上の水晶発振器と交換してください。
デスクトップラボでは、 標準品として 8.000MHz, 8.333MHz, 10.000MHz の在庫を用意しております。

推奨品は、EPSON　SG-8002DC　3.3V タイプです。
この推奨品は、Digikey で購入可能です。周波数の指定が可能です。

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　W3002, W3203 の場合
　ics2_init()関数で初期化の場合
　転送間隔計算式 = 180/(通信レート[Mbps]) + 10[us]　　（最大通信レートは 25Mbps）
例：
　最小転送間隔　19us　（通信レート 20Mbps の場合）
　最大転送間隔　5ms

ICS++シリーズ　W2002, W2203, T2001C, T2006A の場合
　転送間隔計算式 = 180/(通信レート[Mbps]) + 30 [us]　　（最大通信レートは 8Mbps）
例：
　最小転送間隔 57us　(通信レート 6.667Mbps の場合)
　最大転送間隔　5ms

旧製品 ICS：W1003, T2001B の場合、ICS++シリーズ　W1004 の場合
　転送間隔計算式 = 180/(通信レート[Mbps]) + 70 [us]　　（最大通信レートは 1.25Mbps）
例：
　最小転送間隔　214us　(@W1003, W1004 通信レート 1.25Mbps の場合)　最大通信レートは、1.25Mbps
　最大転送間隔　5ms

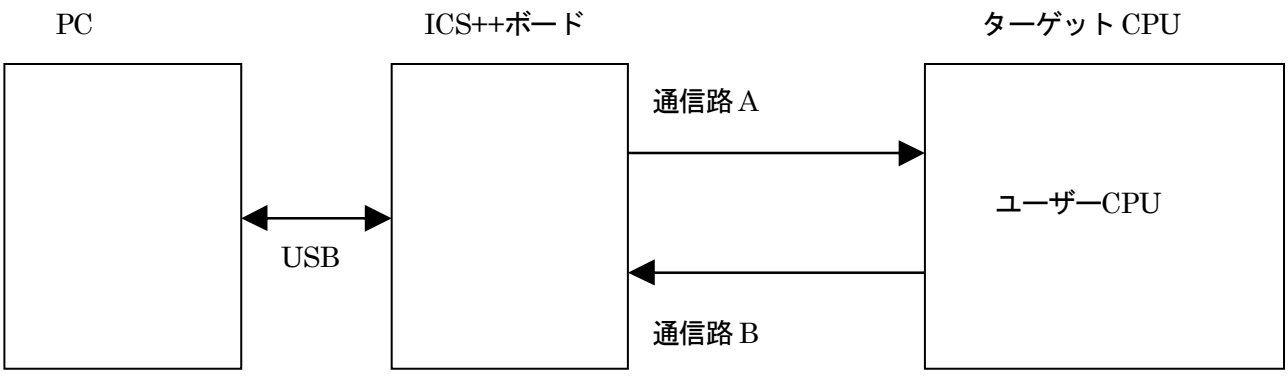


図 4　ICS++システムの通信路

本 ICS++には、データ転送間隔の制限があります。本制約は、図 4 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、2023 年 1 月現在、7 種類の転送モードが存在しています。以下、mode0, 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)

１） mode 0　（１６ビット８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。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 モデルで使用可能です。

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 モデルで使用可能です。

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。

※注意　このモードは、W1001, W1003、T2001A/B, T2002A/B ではサポートされていません。W1004, W2001, W2002, W2203, T2001C, T2006A, W3002, W3203 のモデルで使用可能です。

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　このモードは、W1001, W1003、W1004、W2001, T2001A/B, T2002A/B ではサポートされていません。W2002, W2203, W3002, W3203 モデルで使用可能です。（T2001C, T2006A では、8ch 分のみ使用可能です）

４）mode 4　（８／１６ビット１５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, W2203, W3002, W3203 のバージョンでサポートされます。

５）mode 5　（将来の予約）

４）mode 6　（１６ビットのみ８チャンネル　１回転送モード動作）

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　このモードは、全 ICS モデルでサポートされます。

表 2 転送モード

	メリット	デメリット
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
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
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
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	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

ICS++ライブラリは以下のような２つのファイルの構成になっています。

ヘッダファイル
ics2_<CPUNAME>.h
ics2_<CPUNAME>.lib

通常関数として、
void ics2_init(void* addr, uint8_t unitpin, uint8_t level, uint8_t speed, uint8_t mode);
void ics2_watchpoint(void);
が提供されています。
ただし、ＣＰＵによっては、一部名称が異なる場合があります。

※注意１
RH850/F1KM-S1 のライブラリは、割り込みを使用しません。

※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。

4. ICS++ ライブラリの説明

4.1. RH850/F1KM-S1 シリーズ (CS+　CC-RH コンパイラ)

4.1.1. RH850/F1KM-S1 使用資源

CPU 名	RH850/F1KM-S1 シリーズ
開発環境	CS+ V8.08　　CC-RH
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～6.667Mbps $通信レート = \frac{C\_ISO\_LIN}{6 \times (speed + 1)} [Mbps]$　$speed \in [0, 1, 2, \ldots 39]$ ただし、C_ISO_LIN=40MHz, speed=0 の時、6.667Mbps ※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	ICS_RLIN30_P002_P003 ICS_RLIN31_P005_P004 ICS_RLIN30_P1010_P1009 ICS_RLIN31_P1012_P1011
ライブラリ本体	ICS2_RH850F1KM-S1.o
I/O 初期化設定例	ICS2_RH850F1KM-S1_init.c
ヘッダファイル	ICS2_RH850F1KM-S1.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 RLIN30, RLIN31,　DMA ※注意：割り込みを使用します。	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 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 浮動小数点

4.1.2. RH850/F1KM-S1 関数説明

初期化関数の呼び出し　void ics2_init(uint8_t　port,　uint8_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

RLIN のポート番号や、RLIN の使用するピンを設定します。このパラメータは、ICS2_<CPUNAME>.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

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。

$$通信レート = \frac{C\_ISO\_LIN}{6 \times (speed + 1)} [Mbps] \quad speed \in [0, 1, 2, \ldots 39]$$

ICS ユニットの種類や Firmware のバージョンにより、使用可能な通信レートの範囲が異なります。
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。

第３パラメータ

通信モードの設定

０：設定禁止（将来の予約）

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）
※注意：W1001, W1003 では使用できません。

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）
※注意：W1001, W1003, W1004 では使用できません。

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で置きます。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、DMA で転送します。 下記に示すように、この関数は、ICS のモデルと通信速度とによって決まる最小間隔以上の間隔で呼び出すようにしてください。 ○ICS++　W3002, W3203 の場合 $最小通信間隔 = 1/(通信速度[Mbps]) \times 180 + 10[us]$ ○ICS++　W2001, W2002, W2203, T2001C, T2006A の場合 $最小通信間隔 = 1/(通信速度[Mbps]) \times 180 + 30[us]$ ○ICS　W1001,　ICS W1003, T2001B, T2002B,　ICS++ W1004 の場合 $最小通信間隔 = 1/(通信速度[Mbps]) \times 180 + 70[us]$ ※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も考慮して、呼び出すようにしてください。

4.2. ICS++ 組み込み方法

ICS++を使用するためのユーザープログラムの設定方法を説明します。
RH850/F1KM-S1 の場合、他の ICS ライブラリより、少し複雑になっていますが、自由度も高くなっており、かつ、I/O 初期化部分が公開されています。

4.2.1. ICS++ライブラリのコピー

ICS++ライブラリは、CPU メーカーが提供しているファイルではないため、弊社のサンプルプログラムからコピーする必要があります。 RH850/F1KM-S1 用のサンプルから以下の３つのファイルを組み込もうとするプロジェクトにコピーしてください。

1) ICS2_RH850F1KM-S1.h /* ヘッダーファイル */
2) ICS2_RH850F1KM-S1_init.c /* UART / dma など CPU I/O へのアクセス部分 */
3) ICS2_RH850F1KM-S1.o /* ICS++ライブラリ（I/O へのアクセスはありません) */

4.2.2. ics2_init()関数の組み込み

以下の ics2_init()の呼び出しのためのパラメータ設定方法を説明します。

1) 第 1 パラメータ: 使用するポートを ICS2_RH850F1KM-S1.h から選択してください。
   ヘッダファイルに定義されていないポートを使用する場合、ICS2_RH850F1KM-S1_init.c を変更することによりユーザーで設定をすることも可能です。
2) 第 2 パラメータ: UART 割込みレベル
   割込みレベルを設定してください。最小 2ms 間隔で割込みが発生します。通常は、比較的低い 13, 14 程度を設定してください。
3) 第 3 パラメータ: 通信速度を設定してください。
   CPU のクロック設定と ICS++のモデルによって最適な値を設定します。2023 年 1 月時点で最新の W2002 を使用する場合、最高速の 0 を設定します。 UART 用のクロック C_ISO_LIN=40MHz に設定し、本パラメータを最高速の 0 に設定する場合、6.666666Mbps になります。DTLScope の通信クロックは 8 倍の 53.333333MHz を設定します。
4) 第 4 パラメータ: 転送モード
   8ch モードを選択する場合、1 を設定します。必要に応じて以下のモードを選択します。
   2: 4ch mode (ics2_watchpoint()を呼ぶ度に 4ch サンプリング、かつ転送)
   1: 8ch mode (ics2_watchpoint()を呼ぶと 2 回に 1 回 8ch サンプリング。2 回分割転送)
   3: 12ch mode (ics2_watchpoint()を呼ぶと 3 回に 1 回 12ch サンプリング。3 回分割転送)

詳細は関数の説明部を確認してください。
以上のようにパラメータを決定し、たとえば、
ics2_init(ICS_RLIN31_P1012_P1011, 14, 0, 1); を初期化部分に入れてください。

---------- List 1 main.c ----------

```
void main(void)
{
    ics2_init(ICS_RLIN31_P1012_P1011, 14, 0, 1);
    while(1)
    {
        NOP();
    }
}
```

4.2.3. ics2_watchpoint()関数の組込み

変数をサンプル、および転送処理をする関数が ics2_watchpoint()関数です。通常、この関数はキャリア割り込みの内部から呼び出します。
この関数は、各 ICS ユニットで使用可能な時間間隔を確保して呼び出すようにしてください。計算方法は、ics2_watchpoint()割込み関数の説明部分に記載しています。
割込み周期が計算した時間間隔より短い場合、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    int_TM0(void)     /* 100us間隔  */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics2_watchpoint();
        }
```

4.2.4. 割込み関数の設定

割込み関数は、ics2_RH850F1KM-S1_init.c に記述されています。
これらの対応する RLIN3x の割込みを有効化するため、ICS2_RH850F1KM-S1.h の先頭に定義されている、

```
//#define     ICS_RLIN30
  #define     ICS_RLIN31
//#define     ICS_RLIN32
//#define     ICS_RLIN33
```

の内、実際に使用するものを有効化してください。
ただし、ICS_RLIN32, ICS_RLIN33 は現在提供しているライブラリでは定義していません。

4.2.5. 割込みベクトルの設定

割込みベクトルを修正してください。
RLIN30 ならば、
　Vector　35 に r_Config_UARTx_interrupt_receive
　Vector　36 に r_Config_UARTx_interrupt_error
RLIN31 ならば、
　Vector　122 に r_Config_UARTx_interrupt_receive
　Vector　123 に r_Config_UARTx_interrupt_error
を記述してください。
サンプルには、r_cg_intvector.c に記述例があります。
r_Config_UARTx_interrupt_receive, r_Config_UARTx_interrupt_error の実体は、ICS2_RH850F1KM-S1_init.c 内で定義されています。

4.3. 変数情報生成ツールの CS+への組み込み

PC 上のソフトウェア DTLScope を使用するためには、ターゲット CPU 上のソフトウェアの変数名と変数の型名のデータが必要です。これらを生成される map file から取り出すために、弊社ではツールを用意しています。以下の作業をして、ツールを CS+に組み込んでください。

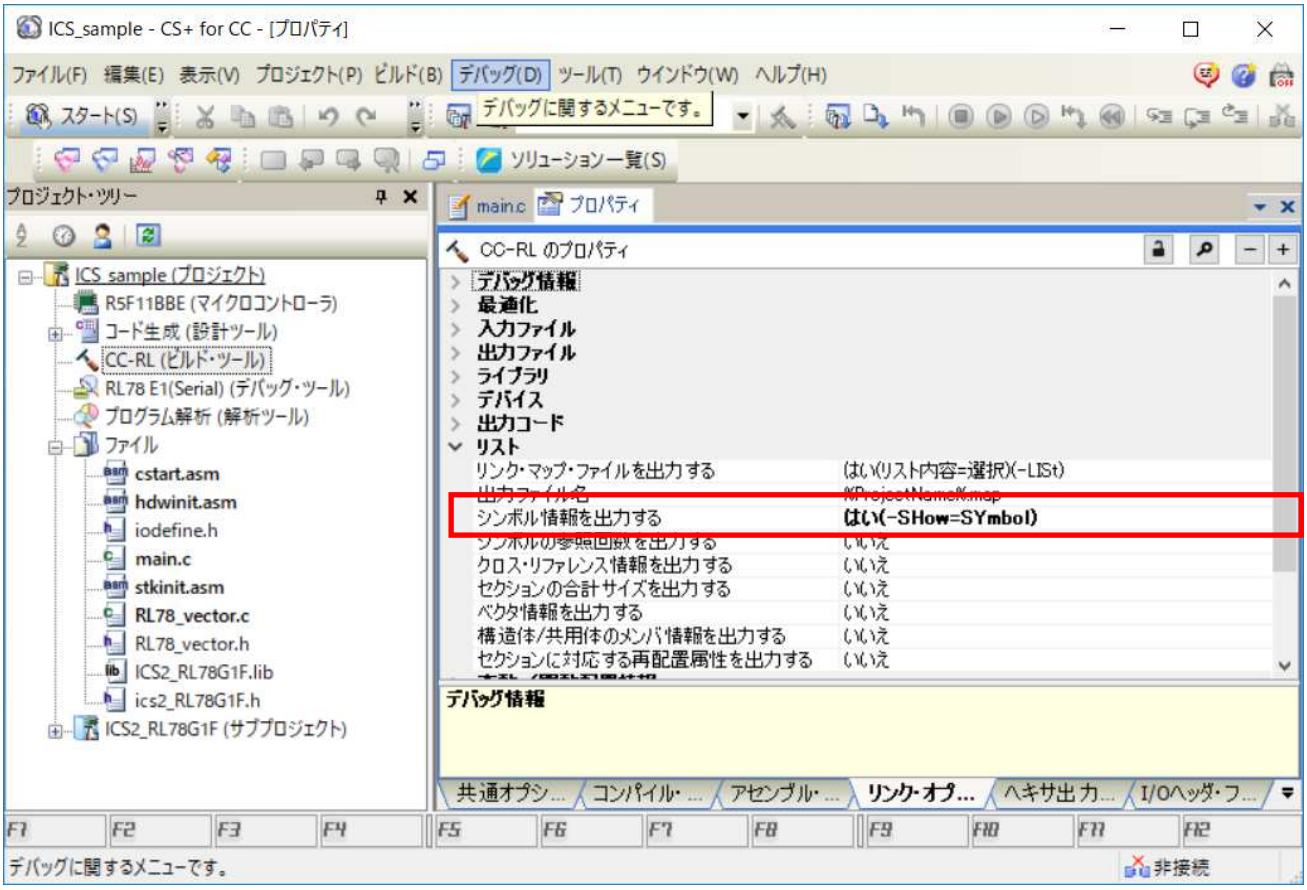
4.3.1. 変数情報生成ツール一式のコピー

サンプルプログラムから以下のファイル一式を、組み込むプロジェクトの *.mtpj というプロジェクト設定ファイルが存在しているフォルダーと同じフォルダーにコピーしてください。

1) map2csv_REL.exe
2) remove.def
3) map2csv.def

4.3.2. CS+への設定変更 1

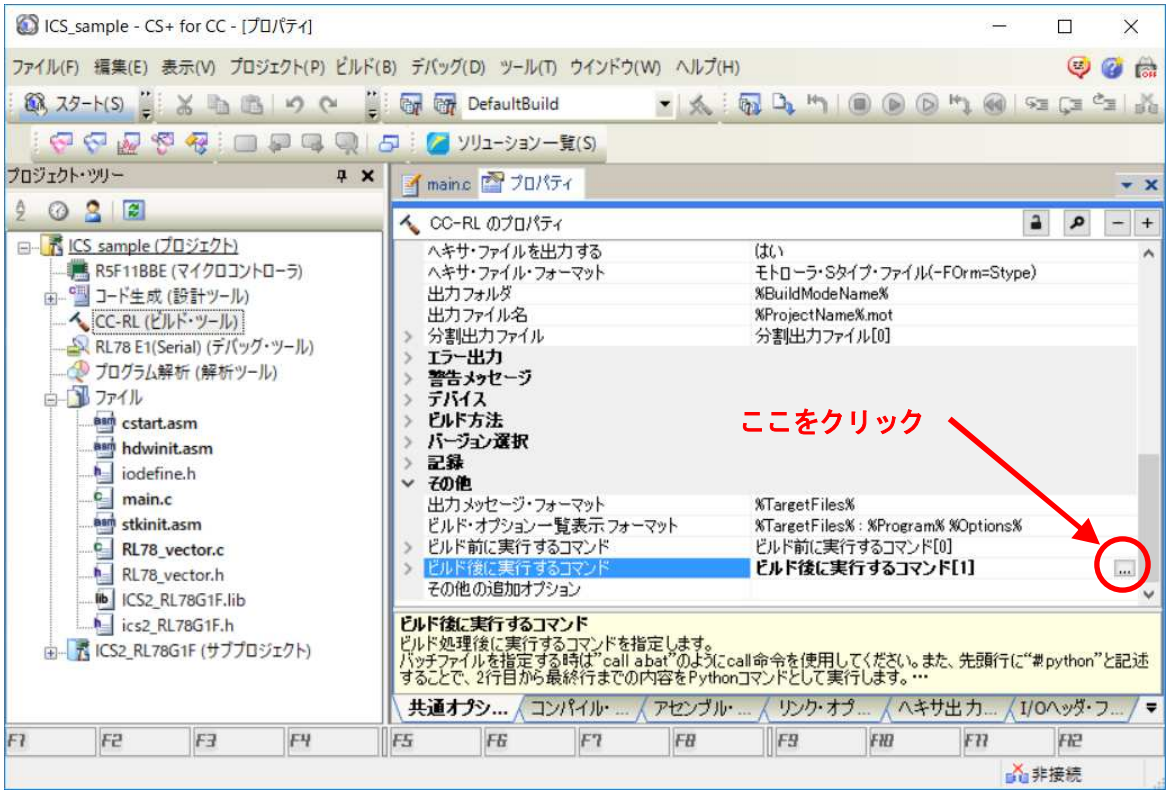
プロジェクトツリー
→リンク・オプション
→リスト
→シンボル情報を出力する 「はい(-SHow=SYmbol)」に変更する。

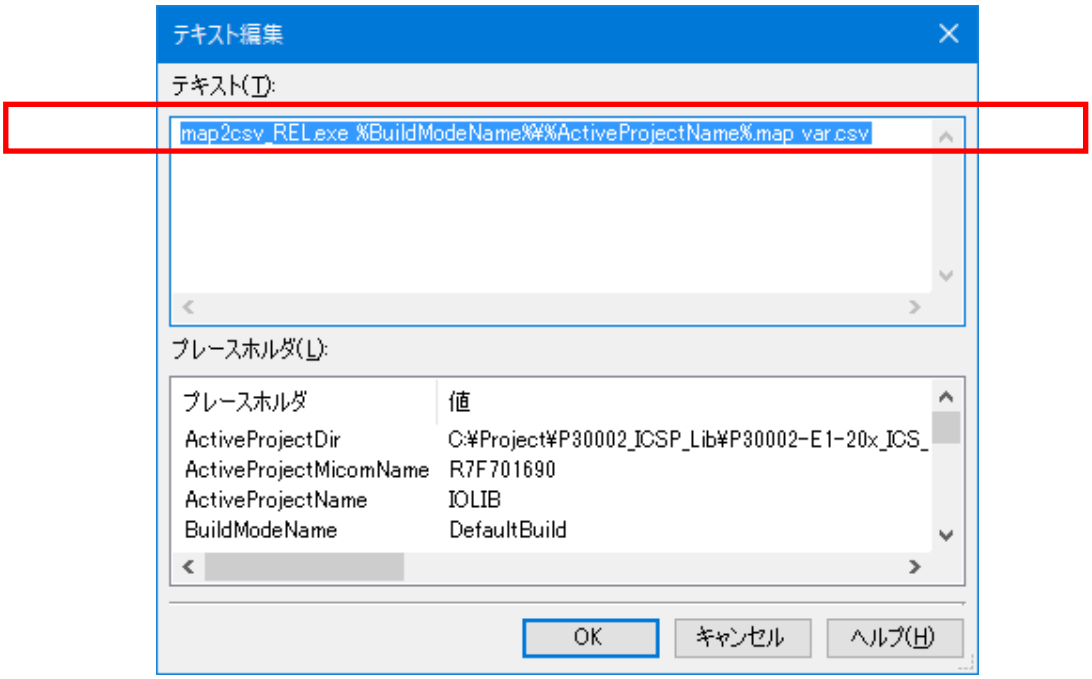


4.3.1. CS+への設定変更２

プロジェクトツリー
→共通オプション
→その他
→ビルド後に実行するコマンド　に下記のコマンドを指定する

map2csv_REL.exe　%BuildModeName%¥%ActiveProjectName%.map　　var.csv





4.3.2. 動作確認

上記を設定した上で、コンパイルをすると、正常にコンパイルが終了すれば、
var.csv というファイルが生成されます。

DTLScope でこの変数ファイルをよみこむと、変数波形を表示したり、変数に読み書きしたりすることができるようになります。

5. ICS++用ハード・ソフトのセットアップ

5.1. ICS++(DTLScope)使用前の準備

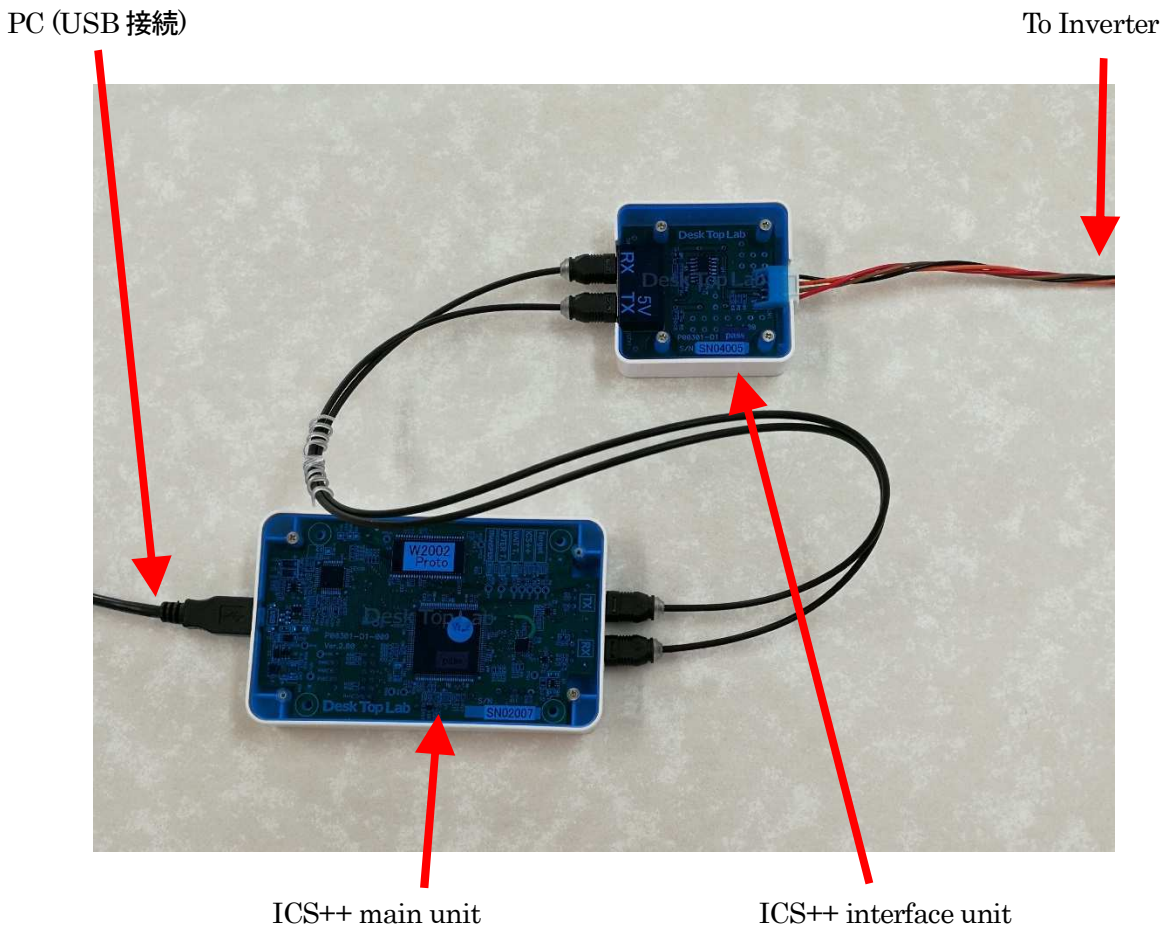
5.1.1. USB ドライバーのインストール

ICS++ W2002, W3002 のユニットには、FTDI の USB チップが入っています。PC から FTDI のチップを認識させるために、FTDI VCP driver の最新版ドライバーをインストールしてください。

弊社のツールセット ICS PC soft Download にも入っています。

5.1.2. 光ファイバーの接続

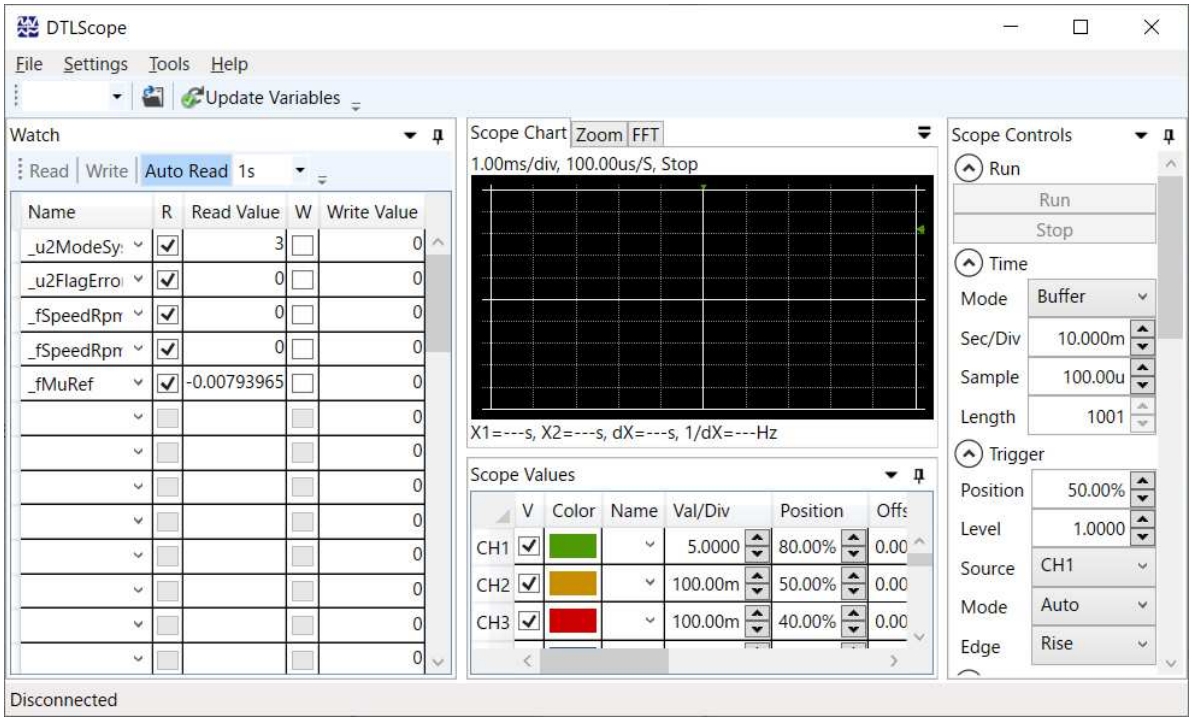
PC ＜－＞ ICS＋＋ ＜－＞ インバータ間を以下のように接続してください。
ICS++ main unit の RX と ICS++ interface unit の TX
ICS++ main unit の TX と ICS++ interface unit の RX



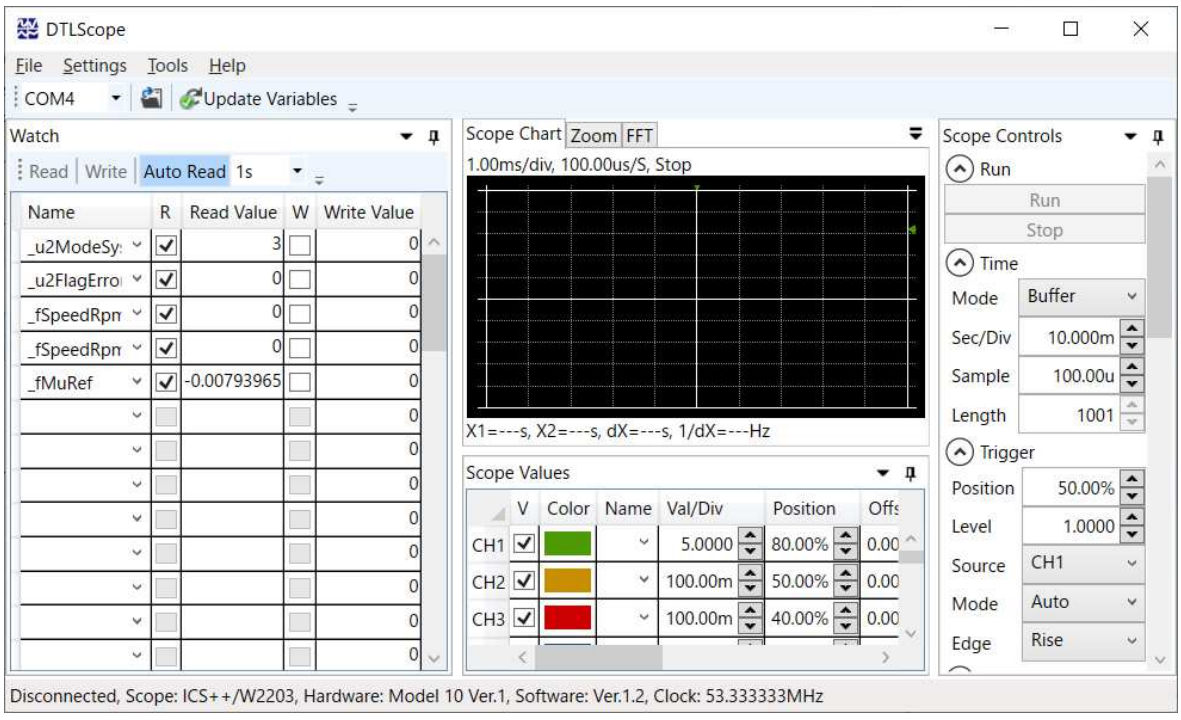
5.2. DTLScope の立ち上げ

5.2.1. COM ポートの設定

COM ポートの設定をしてください。
ICS++ユニットが認識されていない場合、下記のように Window の左下に Disconnected のみが表示されます。
この場合には、ICS++が PC に認識されていません。USB ドライバーを再確認してください。



ICS++ユニットが PC に正しく認識されている場合、下のように Disconnected の後に Scope: ICS++/W2002 などの ICS++のモデル名が表示されます。



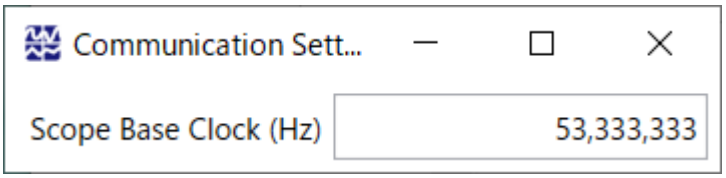
5.2.2. DTLScope の通信レートの設定

DTLScope を使うためには、Target CPU 上のソフトに設定した通信クロックと、DTLScope 上の通信クロックとを一致させる必要があります。

RH850/F1KM-S1　UART CLK の最高速の 40MHz を設定し、ics2_init()の速度パラメータに最高速の"0"を設定した場合には、通信レートは、6.666666Mbps になるため、通信クロックは 8 倍の 53.333333MHz を設定します。

DTLScope
⇒ Settings
⇒ Communicatio settings
をクリックすると、下記の Window が表示されます

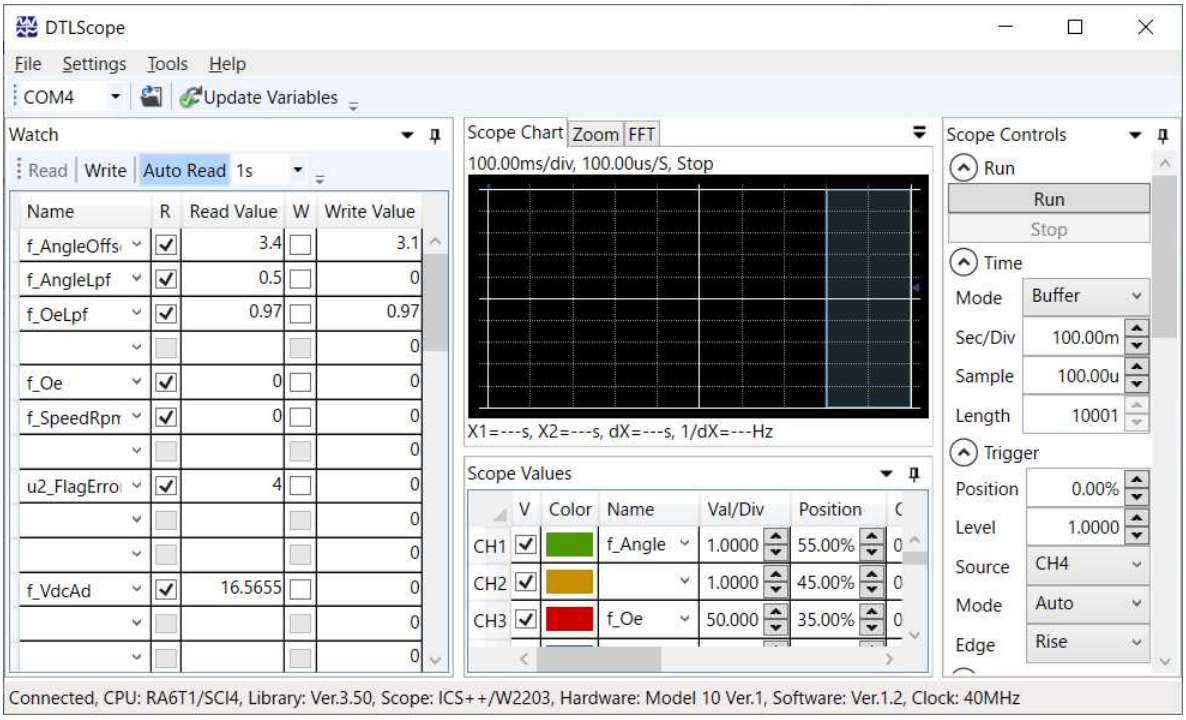
下記の Window に Hz 単位で設定してください。



Window を閉じると、通信クロックが設定され、Main window の status line に 53.333333MHz と表示されます。

5.2.3. Target CPU との接続の確認

COM port を正しく選択し、光ファイバーを接続し、通信クロック設定を行い、正常にターゲット CPU と接続できると、下記のように Connected: CPU: RA6T1 などと、Connected とともに接続できた CPU 名が表示されます。



5.2.4. DTLScope の設定の保存

DTLScope を Windows10 / Windows11 の Start メニューなどから立ち上げると、前回の設定情報をロードするような仕様になっています。また、DTLScope は、ソフトを落とすときに落とす前の状態を保存するため、複数の設定を使いたい場合には、毎回 Start メニューから立ち上げると毎回設定が必要になり不便です。

これをさけるため、プロジェクトや表示したい設定毎に環境を保存することをお勧めします。

File

⇒ Save as

で環境を設定してください。拡張子 dtlsp という拡張子のファイルが生成されます。

5.2.5. DTLScope の設定の再読み込み

同じ設定で DTLScope を使いたい場合には、Start メニューや DESKTOP のアイコンから立ち上げるのではなく、保存した拡張子 dtlsp の環境ファイルをダブルクリックすることにより、DTLScope を立ち上げてください。通信レートや変数など、保存した環境で立ち上げることができます。

6. 改訂履歴

バージョン	変更日	変更内容
Ver.1.00	2023-1-23	・初版作成、RH850/F1KM-S1 を掲載

ICS++ Library　Function manual

発行年月日　2023 年 1 月 23 日　　Ver.1.00JP

発行　　デスクトップラボ株式会社
〒192-0362 東京都八王子市松木３５－７事務所１０１