Desk Top Lab

ICS　32bit 系列ライブラリ
マニュアル

はじめに

＊本説明書に記載された製品・無償提供品の仕様は、予告なく変更される可能性があります。
＊本説明書に記載された情報および本製品の仕様に起因する損害または特許権そのほか権利の侵害については、当社は一切の責任を負いません。
＊本説明書によって第三者または弊社の特許権そのほか権利の実施権を許諾するものではありまねん。
＊当社の書面許諾なく、本製品・本資料の一部または全部を無断で複製することを固くお断りします。
＊本製品を改造したものに関してましては、一切の責任を負いません。
＊本製品をお客様の装置に組み込む際には、バックアップやフェイルセイフ機能をシステムとして別途組み込んでください。
＊当社は、人命にかかわる装置として、特別に開発したものは用意しておりません。
＊記載されている会社名、商品名、各社の商標または登録商標です。

Copyright 2012, 2013　Desk Top Laboratories Inc.
All rights reserved. No part of this manual may be photocopied or reproduced in any form or by any means without the written permission of　Desk Top Laboratories Inc.

ICS は、株式会社ルネサスエレクトロニクス様の製品です。デスクトップラボは、ICS の使用方法、ライブラリなどの ICS 関連サポート業務を行っております。

Desk Top Lab

目次

ライブラリ使用上の注意事項

本ライブラリの通信仕様の詳細については、非公開となっています。ここでは、使用するにあたって重要な項目について説明します。

データ転送間隔の制限について

本 ICS ツールには、データ転送間隔の制限があります。本制約は、図 1 の通信路 B の通信レート上限により発生します。

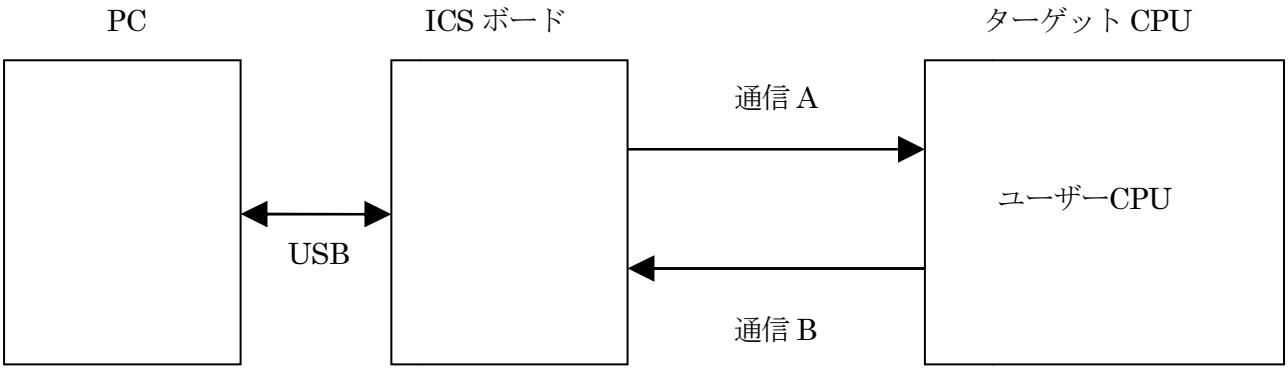

図 1　ICS システムの通信路

後述のデータ転送関数　ics_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存在しています。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。

また、ICS には、いくつかのバージョンがあり、転送間隔の最短時間制限は、バージョンに依存することに注意してください。転送速度が 1Mbps の場合、250us となっています。そのほかのバージョンにつきましては、各ライブラリ部分の記載を参照してください。

１６／３２ｂｉｔライブラリの違い　（RX, RH, V850, RZ, SH シリーズは 32bit ライブラリ）

本ツールでは、１６ビットライブラリと３２ビットライブラリとが存在しています。通常、SH , RX, V850 などの 32bit CPU では３２ビットライブラリを、 78K0R , RL78 などの 16bit CPU では、１６ビットライブラリを提供しています。これらの差について説明します。

【データ転送間隔の制限】の部分で述べた時間間隔以上で転送されるデータは、16byte 分です。従って、16bit 変数ならば 8ch 分、32bit 変数ならば 4ch 分しか送ることが出来ません。32bit を 8ch 分送信するためには、2 回に分けて送ります。

１） １６ビットライブラリの動作
　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。

２）３２ビットライブラリの動作
　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。ics_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 ics_watchpoint()関数が呼ばれたときには、データを取り込まず、前回に取り込み、未転送の残り 4ch 分のデータを転送します。
　つまり、３２ビットライブラリの場合には、２回の ics_watchpoint()関数により、１回の８ｃｈ分の転送が行われます。

　この１６ビットライブラリ、３２ビットライブラリは、CPU のビットで区別をされているわけではなく、波形表示で、32bit をサポートしていかどうかによって、区別されています。16bitCPU でも３２ビットライブラリを、32bitCPU でも１６ビットライブラリを使用することも可能です。非常準ライブラリを必要とする場合には、別途ご相談ください。

	メリット	デメリット
１６ビットライブラリ	波形情報更新間隔が短い	32bit の波形表示ができない
３２ビットライブラリ	32bit の波形表示が可能	波形情報更新間隔が 16bit の２倍

数値表示ウイ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。

　波形表示を行っているとき、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。

ファイルの構成・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
ics_<CPUNAME>.h
ics_<CPUNAME>.obj

通常関数として、
void ics_init(void* addr, char unitpin, char level);
void ics_watchpoint(void);
が提供されています。
ただし、他のライブラリではＣＰＵによって、一部設定が異なる場合があります。

※注意１
また、CPU によって、使用する割り込みが異なります:
使用する UART ポートが異なっても、割り込み処理関数名は同じ名称です。ご注意ください。

※注意２
DTC は、ＤＴＣは標準アドレスモードを使用します。
DTC のベクターテーブルは、ＲAM上に配置する必要があります。

※注意３
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際の状態でライブラリーを生成しています。メモリーモデル、エンディアン、レジスターモード、また、その他にもＣＰＵに特有のスイッチが存在する場合があります。これら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。

使用資源、ライブラリの説明

ＣＰＵ、クロック、使用リソース、サポート変数タイプ、コンパイラなどの組み合わせにより、必要となるライブラリ・バージョンやICSのタイプが異なります。

RX62T　R5F562TAADFP　(CubeSuite+)

CPU 名	RX62T 100pin　　R5F562TAADFP
開発環境	CubeSuite+ Ver.2.00.00a
CPU クロック	標準クロック　ICLK= 96MHz,　PCLKB = 48MHz　(1Mbps)
ステイタス	SCI0, SCI1, SCI2 サポート中
ライブラリ種別	32bit ライブラリ
ライブラリ	ics_RX62T.obj
ヘッダファイル	ics_RX62T.h

CPU 使用リソース	対応 ICS	サポート変数タイプ
・内部使用リソース SCI0 　INT　SCI0 RXI 　DTC　INT216　(TXI0) 　ICU.DTCER[216].BIT.DTCE 　SCI0　全て 　DTC　全て 　ICU.IPR[0x80].BYTE 　ICU.IER[0x1A].BIT.IEN6 　ICU.IER[0x1A].BIT.IEN7 　ICU.IER[0x1B].BIT.IEN0 　ICU.IER[0x1B].BIT.IEN1 　SYSTEM.MSTPCRA.BIT.B28 　SYSTEM.MSTPCRA.BIT.B31 　SYSTEM.MSTPCRB.BIT.B31 　PORTB.ICR.BIT.B1 = 1 　外部ピン 　PB2: TXD0 　PB1: RXD0 SCI1 　INT　SCI1 RXI 　DTC　INT220　(TXI1) 　ICU.DTCER[220].BIT.DTCE	○対応 ICS ハード 　２種類の対応が可能です。 標準版 H/W model　1 H/W Ver.　1 S/W Ver.　100.20　以降 または、 H/W model　4 H/W Ver.　1 S/W Ver.　1.22　以降 以上は、ICS のハードを接続した状態で、ICS アプリの 　Help -> About で表示可能 標準時 通信レート 1.00Mbps ICS PC ソフト Ver. 2.5.0.0　以降	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 　8bit　BOOL 型 　8bit　LOGIC 型

SCI1　全て DTC　全て ICU.IPR[0x81].BYTE ICU.IER[0x1B].BIT.IEN2 ICU.IER[0x1B].BIT.IEN3 ICU.IER[0x1B].BIT.IEN4 ICU.IER[0x1B].BIT.IEN5 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B30 PORTDICR.BIT.B5= 1 外部ピン PD3 TXD1 PD5: RXD1 SCI2 (PB5, PB6) INT　SCI2RXI DTC　INT224　(TXI2) ICU.DTCER[224].BIT.DTCE SCI2　全て DTC　全て ICU.IPR[0x82].BYTE ICU.IER[0x1B].BIT.IEN6 ICU.IER[0x1B].BIT.IEN7 ICU.IER[0x1C].BIT.IEN0 ICU.IER[0x1C].BIT.IEN1 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B29 PORTB.ICR.BIT.B6 = 1 IOPORT.PFFSCI.BIT.SCI2S 外部ピン PB5: TXD2 PB6: RXD2		

SCI2 (P81, P80) INT SCI2RXI DTC INT224 (TXI2) ICU.DTCER[224].BIT.DTCE SCI2 全て DTC 全て ICU.IPR[0x82].BYTE ICU.IER[0x1B].BIT.IEN6 ICU.IER[0x1B].BIT.IEN7 ICU.IER[0x1C].BIT.IEN0 ICU.IER[0x1C].BIT.IEN1 SYSTEM.MSTPCRA.BIT.B28 SYSTEM.MSTPCRA.BIT.B31 SYSTEM.MSTPCRB.BIT.B29 PORT8.ICR.BIT.B0 = 1 IOPORT.PFFSCI.BIT.SCI2S 外部ピン P81: TXD2 P80: RXD2		

Lib Ver.2.0　on　CubeSuite+ Ver.2.00.00a

初期化関数の呼び出し　void ics_init(void* addr, char　port,　char level);

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
　DTC で使用する DTC のベクトルテーブルの先頭アドレスを指定してください。ics_init()関数を呼び出す前にユーザーが DTC ベクトルテーブルを確保、および、ショートアドレス形式の転送情報を保存する変数のアドレスをサンプルの様に、DTC テーブルに設定する必要があります。このアドレスの下位 12bit は 0 である必要があります。

第２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、ICS_<CPUNAME>.h 内で定義されている文字列を使用してください。

第３パラメータ
　ICS で使用する SCI の割り込みレベルを指定します。下記の条件を満たすように設定してください。
　最小間隔で 2ms の割り込みが発生する可能性があるため、システムとして、この割り込み間隔を許容できる割り込みレベルを設定してください。
　SCI の受信割り込みが一番処理時間の長い割り込みです。10us 程度ですが、割り込み禁止時間を許容できない割り込みソースがある場合には、この割り込み設定レベルよりも高い割り込みレベルを設定してください。

転送関数の呼び出し　　void　　ics_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、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 1Mbps の際最小 250us 以上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。

$$最小通信間隔 = 1/(通信速度[bps]) \times 180 + 70[us]$$

通信速度が 1Mbps の際、上の式に数値を代入すると。

$$最小通信間隔 = 1/(1[us]) \times 180 + 70[us] = 250[us]$$

※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。

使用割り込み関数

下記の割り込みベクトルを使用しているため、ユーザーソフトの割り込みベクトルに下記の関数を登録してください。
　ルネサス標準のコンパイラで自動的に生成されるプロジェクトを使用する場合、intprg.c という割り込み処理を記載したファイルがあります。

　たとえば、SCI0 を使用する場合、下記のように記述してください。

SCI0 の場合
```
// SCI0 ERI0
void Excep_SCI0_ERI0(void){ ics_int_sci_eri(); }
// SCI0 ERI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

SCI1 の場合
```
// SCI1 ERI1
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

SCI2 の場合
```
// SCI2 ERI2
void Excep_SCI2_ERI2(void){ ics_int_sci_eri(); }
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

使用方法（RX62T）

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。

１）開発環境に BDTCTBL セクションを確保する。

開発環境を設定して BDTCTBL のセクションを RAM 上に下位 12bit が 0 になるようなアドレスに設定します。RX62T の RAM が最小のモデルはサイズが 8kByte なので、設定できるアドレスは、0x0000 もしくは、0x1000 となります。RAM が 16kByte のモデルの場合、設定できるアドレスは、0x0000, 0x1000, 0x2000, 0x3000 の３点となります。ここでは、0x0000 からに設定することにします。
E1 などのエミュレータを使用する場合、ユーザーRAM 領域もしくは、DTC テーブルの領域と、E1 エミュレータの使用領域とが重ならないようにしてください。正常に動作しないことがあります。

２）ユーザープログラムに、DTC テーブルを定義する。

ICS_sample.c の先頭に DTC テーブルの変数である dtc_table[256]を定義する。

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section
```

３）ics_init()を下記のように呼び出す。

初期化関数 ics_init((void*)dtc_table, ICS_SCI2_PB5_PB6, 6) を初期化部分に入れてください。
第１パラメータは、１）で確保した BDTCTBL の先頭アドレスを入力してください。
第２パラメータは、使用するポートを ICS_<CPUNAME>.h から選択してください。
第３パラメータは、ICS で使用する割り込みレベルを設定してください。通常は、キャリア割り込みより低い優先順位に設定します。

---------- List 1　main.c ----------

```
#pragma section DTCTBL
unsigned long dtc_table[256];        // caution alignment 0x000
#pragma section

void main(void)
{
      ics_init((void*)dtc_table, ICS_SCI2_PB5_PB6, 6);      /* Interrupt level 6            */
      while(1)
      {    nop();    }
}
```

４）ics_watchpoint()関数の組込み

このソフトでは、メインルーチンで ics_watchpoint()関数を呼び出していますが、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、250us 以上の間隔で呼び出されるようにしてください。割り込み処理関数が 250us 以下の間隔で呼び出される場合、List2 のように ics_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　ics_watchpoint()の間引き ----------

```
int     deci = 0;


void    int_TM0(void)     /* 100us間隔 */
{
        deci = deci + 1;
        if (deci >=3)
        {
            deci = 0;
            ics_watchpoint();
        }
}
```

６）intprg.c の修正

intprg.c の中の　RXI, ERI の割り込み関数から　ics_int_sci_eri(),　ics_int_sci_rxi()関数を呼び出すようにしてください。

SCI0 の場合

```
// SCI0 ERI0
void Excep_SCI0_ERI0(void){ ics_int_sci_eri(); }
// SCI0 RXI0
void Excep_SCI0_RXI0(void){ ics_int_sci_rxi(); }
```

SCI1 の場合

```
// SCI1 ERI1
void Excep_SCI1_ERI1(void){ ics_int_sci_eri(); }
// SCI1 RXI1
void Excep_SCI1_RXI1(void){ ics_int_sci_rxi(); }
```

SCI2 の場合

```
// SCI2 ERI2
void Excep_SCI2_ERI2(void){ ics_int_sci_eri(); }
// SCI2 RXI2
void Excep_SCI2_RXI2(void){ ics_int_sci_rxi(); }
```

ＩＣＳ用クロック（RX62T）

本ライブラリを使用する場合、ＣＰＵ側のクロックの設定と、ＩＣＳボード上のクロックの設定の関係を下記のようにしてください。
クロックを可変できないモデルの場合には、本ライブラリをクロックが 96MHz で使用する必要があります。

ICS ボード上のクロック周波数 ＝ (PCLKB / 6) MHz

例： PCLKB = 50MHz の場合、 ICS CLOCK = 50/6 = 8.333MHz
PCLKB = 48MHz の場合、 ICS CLOCK = 48/6 = 8.000MHz

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。

注意事項：

8MHz 以外の ICS 用クロックを使用する場合、ICS の型式で水晶発振器がソケット上に搭載されている製品のみで使用可能となります。
デスクトップラボ社の製品型番で W1003 というタイプとなります。

改訂履歴

バージョン	変更日	変更内容
Ver.1.00	2013-08-12	・RX62T Ver2.0 ライブラリ　初版作成