インターフェース増刊

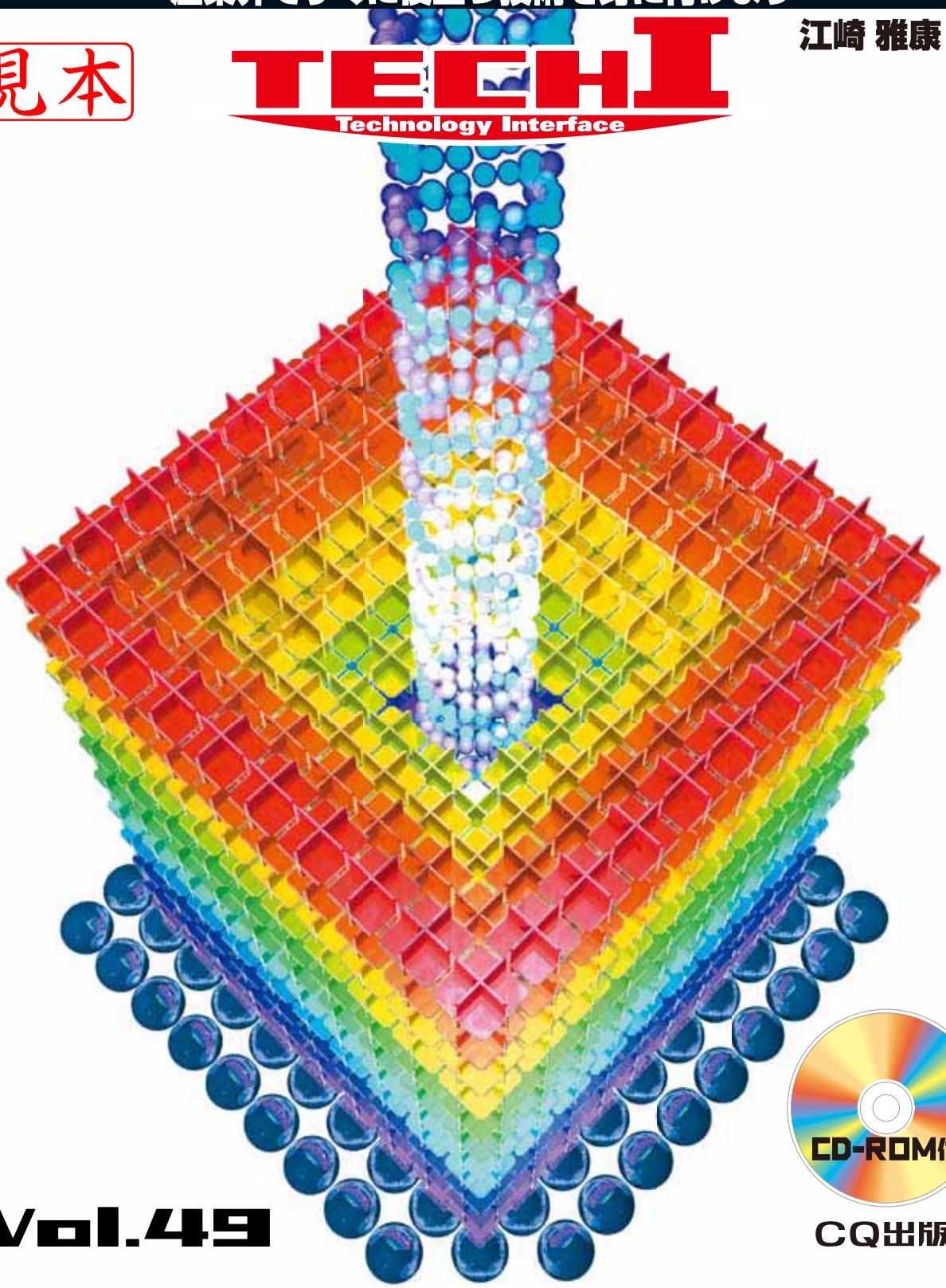
SH-2Aマイコン組み込み技術講座

産業界ですぐに役立つ技術を身に付けよう

江崎 雅康 著

見本

TECHI
Technology Interface



CD-ROM付

Vol.49

CQ出版社

第1章 開発ツールのインストールとその使い方

本書で紹介するソフトウェア開発の実例を実践するために，その前準備としてパソコンにSH-2Aの組み込み開発環境を構築します．ソフトウェア開発を理解するためにはかならず必要になるので，この作業は実行しておいてください．

1.1 SH-2Aの教育・学習用ツールの選択

● 組み込みソフトウェア開発に必要なツールの選択…開発現場のツールを使えば，即戦力のスキルが身に付く

プログラムを開発するためには開発ツールが必要です．『開発ツール』とは，アセンブラ[注1]，Cコンパイラ[注2]などのプログラミング言語処理ツールやデバッガ，エミュレータ[注3]およびフラッシュ・メモリ[注4]書き込みツールなどのシステム開発に必要な『道具』の総称です．SH-2Aの開発ツールには，メーカ純正ツール(マイクロプロセッサ・メーカがユーザの開発用に用意したツール)やサードパーティ(マイクロプロセッサ・メーカ以外のメーカ)製ツール，フリーのツールといろいろあります．

筆者は教育や学習においても産業界の開発現場で使われているツールを使うのが望ましいと考えています．学術研究用にしか使われていない無償ツールでの学習は回り道になるからです．本書ではSH-2Aの開発現場で最も多く使われているルネサス エレクトロニクス社のツールを使うことにしました．

ルネサス エレクトロニクス社の純正ツールを教育や学習に使えば，即戦力のスキルを最速で身につけることができるからです．製品版は10万円以上するので，メーカの協力により本書付属CD-ROMに収録した評価版を使います．

● 組み込みソフトウェア開発の手順と必要な開発ツール

図1.1は組み込みソフトウェアの開発手順です．プログラム作成はソース・コード[注5]を書く作業です．ここでは，システム設計に基づいて作成された要求仕様を，実際にソース・コードに書き落とします．

ソース・コードはC言語(**リスト1.1**)もしくは一部アセンブリ言語(**リスト1.2**)で記述します．このソース・コードをコンピュータのマシン語[注6](**リスト1.3**)に変換するのがコンパイラおよびリンカです．

コンパイラはC言語で記述されたソース・コードをリロケータブル(再配置可能)な中間言語に変換します．アセンブリ言語で記述されたソース・コードもアセンブラと呼ばれる処理系によって中間言語に変換されます．

リンカは生成された中間言語モジュールを結合して，オブジェクトと呼ばれる実行可能なマシン語コードに変換します．

こうしてできたオブジェクト・コードは，フラッ

注1：マシン語命令と1対1に対応したプログラム開発言語をアセンブリ言語という．マシン語は異なるコンピュータ間の互換性がほとんどない．20～40年前のプログラム開発はほとんどこの言語で行われた．アセンブリ言語で記述されたプログラムをマシン語に変換する処理系(プログラム)をアセンブラという．

注2：C言語は人間の言葉に近い記述ができるプログラム開発言語である．マシン語が異なるコンピュータ間での移植性に優れているため，20年ほど前から多く使われるようになった．C言語で記述されたプログラムをマシン語に翻訳する処理系(プログラム)をCコンパイラという．

注3：マイコン・システムのハードウェアおよびソフトウェアの不具合を見つけ出して，不具合を取り除くためのツールをデバッガという．マイコン自体の動作を模擬する装置をエミュレータと呼ぶ．

注4：一括消去型の不揮発性メモリ．単体チップ部品やマイコンに内蔵されたプログラム用などがある．

注5：人間(プログラマ)が記述したプログラムをソース・コードという．

注6：コンピュータ内部で使われる機械語．

シュ・メモリ書き込みツールによってマイコン内蔵フラッシュ・メモリに書き込みます．SH7285 にはフラッシュ・メモリ書き込みのために二つの方法が用意されています．

一つは，FDT(Flash Development Toolkit)と呼ばれるフラッシュ・メモリ書き込みツールです．SCI(シリアル・コントローラ)または USB ポートから読み込まれたメモリのデータは SH7285 に組み込まれたブー

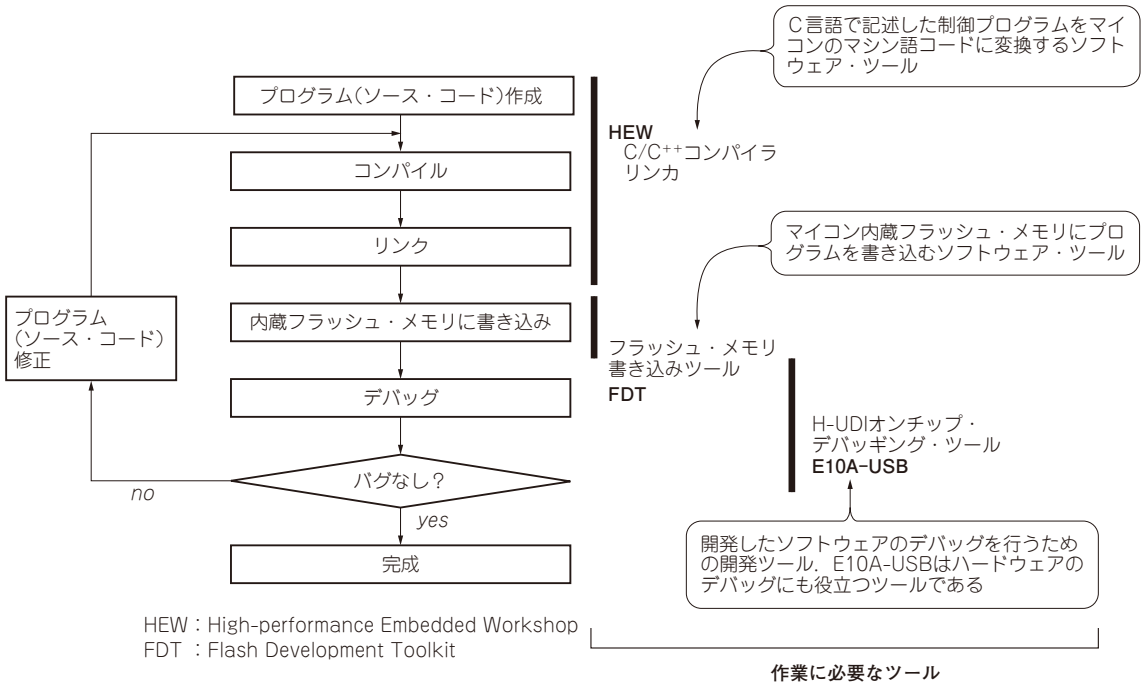


図 1.1　組み込みソフトウェア開発の手順
組み込みソフトウェアの開発は，対象機器，使うマイコンが異なっても，おおむねこの手順で進められる．この中で「デバッグ」は開発ソフトウェアの品質を左右する重要な工程である．

リスト 1.1　C言語で記述されたプログラム例

```
/***********************************************/
#include"iodefine.h"
/***********************************************/
void main(void)
{
     PFC.PECRL3.BIT.PE9MD = 0;
          /* PE9 ポート機能選択 */
     PFC.PEIORL.BIT.B9 = 1;
          /* PE9 ポート方向出力選択 */
     PE.DR.BIT.B9 = 1;
          /* n7SEGON0 を H レベルに設定 */
     PFC.PDCRH4.WORD = 0x0000;
          /* ① PD28~31 ポート機能選択 */
     PFC.PDCRH3.WORD = 0x0000;
          /* ② PD24~27 ポート機能選択 */
     PFC.PDIORH.BYTE.H = 0xFF;
          /* ③ PD24~31 ポート方向出力選択 */
     PD.DR.BYTE.HH = 0xFF;
          /* ④ 7 セグメント LED 全て消灯 */
     PD.DR.BYTE.HH = 0xA4;
          /*「2」を表示するようなパターン設定 */
     PE.DR.BIT.B9 = 0;
     /* n7SEGON0 を L レベルにして 7 セグメント LED0 表示 */
     /* 無限ループ */
     while(1){
     }
}
```

リスト 1.2　アセンブリ言語記述プログラム例

```
??00022:  dc.l      $FFFF842A
??00024:  dc.l      $FFFF842C
??00026:  dc.l      $FFFF842E

          segment   TEXT ATR_CODE
public    _wait
;         w         r3
_wait:
          mov.l     r11,@-sp
          mov.l     r12,@-sp
          mov.l     r13,@-sp
;
          mov.l     ??00031,r4
          mov.l     ??00032,r5
          mov.w     r5,@r4
;
          mov.l     ??00033,r6
          mov       #0,r7
          mov.w     r7,@r6
;
          align     4
??00036:  dc.l      _i
??00037:  dc.l      $00000080
??00032:  dc.l      $000002ED
??00035:  dc.l      $FFFF83D0
??00034:  dc.l      $FFFF83D2
??00033:  dc.l      $FFFF83D4
??00031:  dc.l      $FFFF83D6
```

第2章 SH7285の入出力インターフェースとGPIOの使い方

2.1 組み込み機器の入出力インターフェース回路

ここではGPIO(汎用入出力ポート)，シリアル，カウンタ/タイマなどの入出力インターフェース回路の基本を解説します．

● 組み込みマイコンと入出力装置の役割

図2.1は組み込みシステムの制御対象(機器)や入出力装置，組み込みマイコンの関係を図示したものです．

組み込み技術は航空機から家電機器まで多岐に使われています．組み込みマイコンが取り込むセンサ情報や制御出力は多種多様です．

たとえば自動車はエンジンの回転数，燃料ガスの濃度，排気ガスの圧力などのセンサからの情報を基に組み込みマイコンが適切な判断を行い，点火プラグや燃料バルブの開閉，パワー・ステアリングやパワー・ウィンドウ用駆動モータなどのアクチュエータを操作

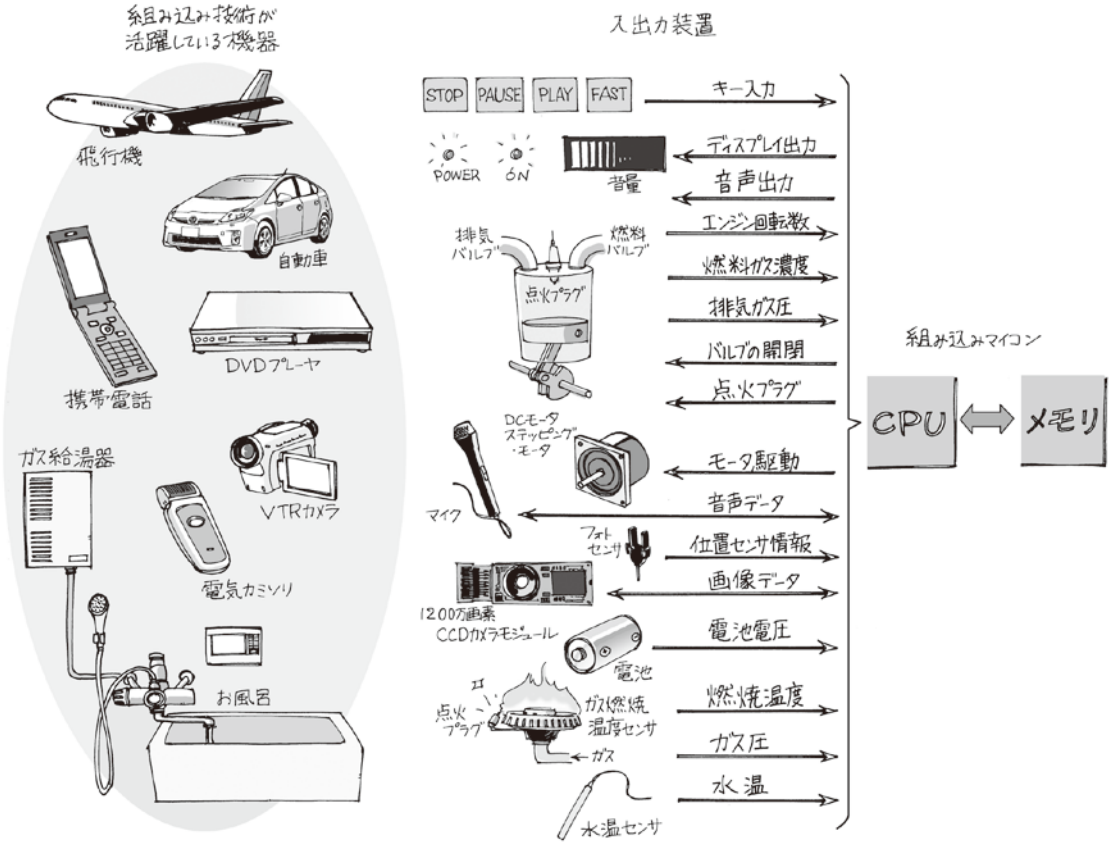


図2.1　組み込みシステムの入出力装置
組み込み機器はCPU，メモリ，入出力装置で構成される．多様な入出力装置が機器の機能を支える．

します．

制御対象の入出力装置は機器によって異なります．飛行機から携帯電話，DVD プレーヤから家電製品まで，組み込みシステムに使われている入出力装置をすべて列挙することはおそらく不可能でしょう．

しかし制御対象機器は異なっても，組み込みシステムの仕組みは共通です．組み込みマイコンは，次の三つの機能を備えています．

① 制御対象の情報を取り込む

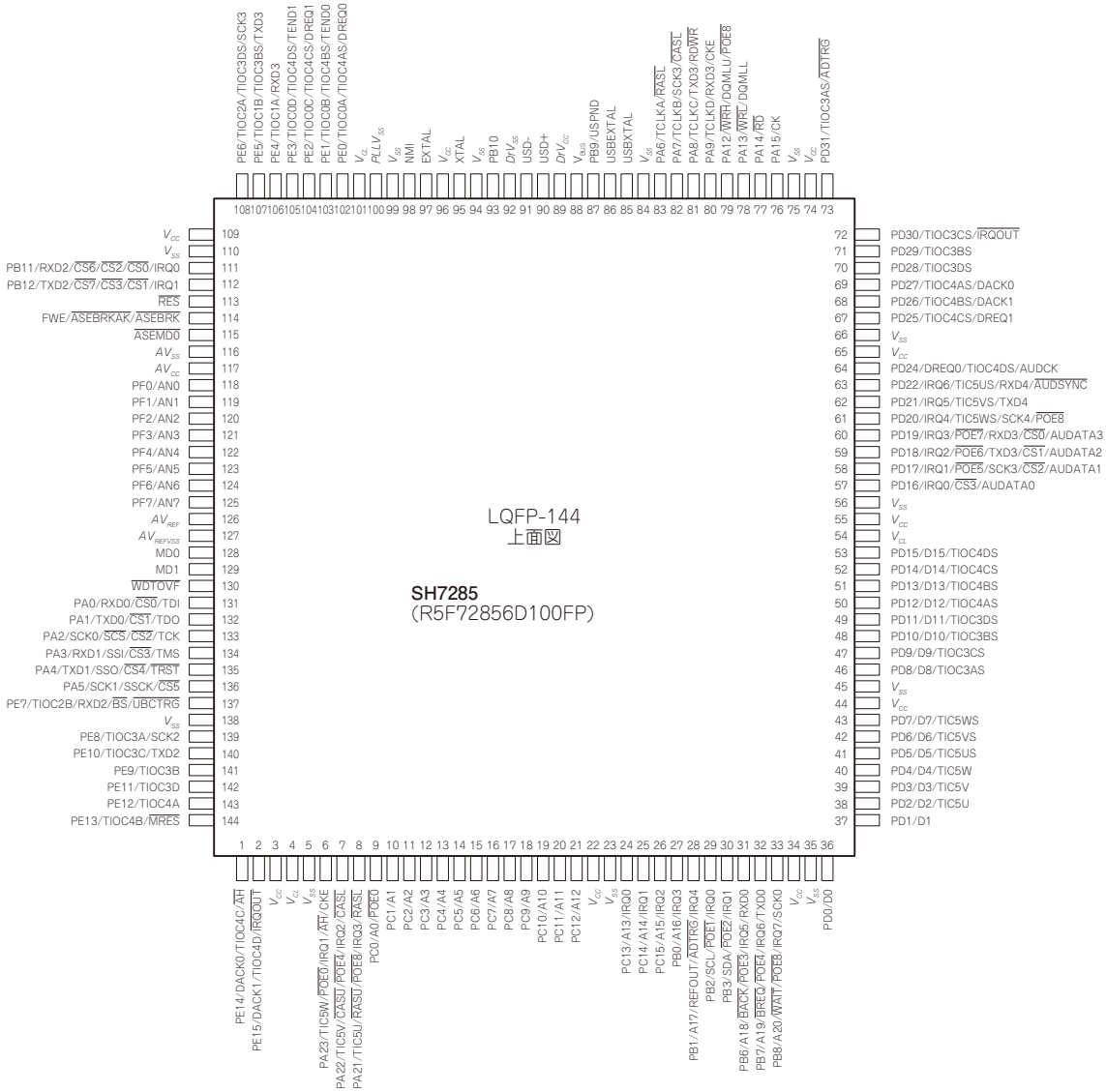
② 取り込んだ情報に基づいて必要な演算を行う

③ 演算結果に基づいて制御対象に操作を加える

この『情報を取り込む』，『操作を加える』役割を果たすのが入出力インターフェース回路です．組み込みシステムの入力回路はコンピュータの目や耳，アクチュエータは手足の役割を果たしています．

● 組み込みマイコン SH7285 は CPU，メモリ，入出力装置を内蔵

図 2.2(a) は SH7285（発注型名は R5F72856D100FP）のパッケージ・ピン配列です．組み込みマイコンと外



(a) SH7285 のパッケージ・ピン配列

SH7285 の 144 ピン・パッケージに多くの機能を搭載するため，信号ピンは複数の機能を兼用するものが多い．

図 2.2　SH-2A コア内蔵 32 ビット・マイコン SH7285 のピン配列とメモリ・マップ

第3章 H-UDIによるデバッグの方法

3.1 組み込みシステムの品質を決めるデバッグ

組み込みシステムのハードウェアおよびソフトウェアの不具合を見つけて修正する作業を「デバッグ」といいます.

組み込みシステムの開発ではデバッグが製品の品質を左右します.

システムによってはソース・プログラムのコーディングより,プログラムの不具合を洗い出して修正するデバッグ作業の方に時間がかかる場合もあるほどです.「デバッグ・ツールを使いこなしてこそ,プロの技術者」といわれる理由もここにあります.

3.2 H-UDIの概要と設定,接続方法

SH7285はルネサス エレクトロニクス社独自のH-UDI(User Debag Interface)と呼ばれるデバッグ用の専用ハードウェアを備えています.

機能的には「JTAGデバッガ」ですが,ルネサス エレクトロニクス社のマニュアルには一度も「JTAG」という言葉は出てきません.これは,「JTAG仕様の詳細な規定を厳格には満たしていないから」という理由のようです.

H-UDIはSH7285に内蔵されたデバッグのためのハードウェアですが,CPUと独立に存在しています.この点がソフトウェア方式のデバッガと違う点です.

本章ではH-UDIを使ったデバッグ方法について解説します.

● H-UDIを使うために必要なハードウェア・ツール …E10A-USB

SH7285に内蔵されているH-UDIを使ってデバッグ作業を行うためには,専用のエミュレータが必要です.**写真3.1**はルネサスエレクトロニクス社のエミュレータE10A-USBです.E10A-USB[注1]は基本機能を備えた標準型が15万円前後します.

ルネサス エレクトロニクス社のH-UDIインターフェースのテクニカル情報を取り入れて独自に開発されたサードパーティの製品もありますが,操作方法やツールの画面は異なります.

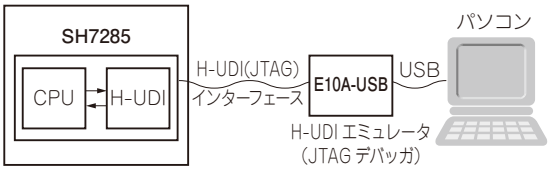
そこで本書執筆のために開発した教育&開発プラットホーム基板CQK-SH2A[注2]にルネサス エレクトロニクス社の協力を得てE10A-USBと同等の環境を使えるようにしました.

図3.1(a)はH-UDIのしくみです.SH7285に内蔵されたH-UDIはCPUとは独立しているパソコンと接続してデバッグを行います.

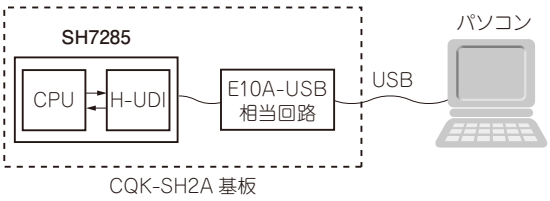
写真3.1　USBエミュレータE10A-USB(ルネサス エレクトロニクス社)

注1:ルネサス エレクトロニクス社独自のユーザ・デバッグ・インターフェース規格H-UDIに対応したハードウェア・エミュレータ.
注2:ルネサス エレクトロニクス社のマイクロプロセッサSH-2Aを中心に組み込みシステムの学習に必要な周辺回路を備えた組み込み教育と試作開発向けプラットホーム基板.

H-UDI エミュレータ E10A-USB は H-UDI とパソコンを接続するためのツールです．CQK-SH2A は**図 3.1**（**b**）に示すように，E10A-USB と同等の回路を備えています．



（**a**）H-UDI（ユーザ・デバッグ・インターフェース）のしくみ



（**b**）組み込み教育＆開発プラットホーム基板
CQK-SH2A には E10A-USB 相当の機能が搭載されている

図 3.1　SH-2A マイコンに搭載されている H-UDI の機能

このため図に示すように，USB ケーブルでパソコンと接続して E10A-USB と同じデバッグの環境を作ることができます．実際のデバッグでは，基板上の USB コネクタ J8，CN2 を 2 本の USB ケーブルでパソコンと接続します．

CQK-SH2A 基板上という制約はありますが，純正の E10A-USB と同じ使い勝手でソフトウェアのデバッグが行えます．

E10A-USB のコントロール・インターフェースは HEW[注3] に組み込まれているため，デバッグ操作はすべて HEW 上で行えます．

● デバッグ用の最適化設定

それでは実際にデバッグを行ってみましょう．まず H-UDI によるデバッグの実習を行うためプログラムを用意します．新規プロジェクトの作成方法に従って，プロジェクトを用意し，**リスト 3.1** に示すプログラムを入力します．

リスト 3.1　JTAG デバッグ サンプル・プログラム

```
/*********************************************/
/* 3.JTAG デバッグ　サンプル・プログラム 1          */
/*                                           */
/*  デバッグ演習用プログラム．最後に押したプッシュ・  */
/*  スイッチの番号を                           */
/*  7 セグメント LED に表示する                */
/*********************************************/

#include "iodefine.h"

#define ON          1
#define OFF         0

void seven_segment_LED_init(void);
void seven_segment_LED_set(int pattern_code);
void seven_segment_LED_power(int segment_No,
                                    int on_off);
void push_switch_init(void);
int push_switch_get(int switch_No);

void main(void){
     int n;
     int switch_status;
     int last_pushed_button;
     int i;

     /* ハードウェアの初期化 */
     seven_segment_LED_init();
                    /* 7 セグメント LED の初期化 */
     seven_segment_LED_power(0,ON);
               /* 7 セグメント LED0 を常に表示可能状態 */
     push_switch_init(); /* プッシュスイッチの初期化 */

     /* ソフトウェアの初期化 */
     last_pushed_button = 0;

     /* メインルーチン */
     while(1){
          /* SW3～6 の状態を調べ押下状態なら
                                スイッチ番号を記録 */
          for(n=3; n<=6; n++){
            switch_status = push_switch_get(n);
            if(switch_status == ON){
                    last_pushed_button = n;
              }
          }

          seven_segment_LED_set
                    (last_pushed_button);

          for(i=0; i<250000; i++){ /* 時間稼ぎ */
            }
        }
}

void seven_segment_LED_init(void){
     /* ピン機能設定 */
     PFC.PDCRH3.WORD = 0;
     PFC.PDCRH4.WORD = 0;
     PFC.PECRL3.BIT.PE9MD = 0;

     /* 入出力方向設定 */
     PFC.PDIORH.BYTE.H = 0xFF;
     PFC.PEIORL.BIT.B9 = 1;

     /* 全て消灯 */
     PD.DR.BYTE.HH = 0xFF;
     PE.DR.BIT.B9 = 1;
}

const unsigned char seven_segment_LED_number_
                    pattern[] = {0xC0, 0xF9, 0xA4,
     0xB0, 0x99, 0x92, 0x82, 0xD8, 0x80, 0x90};

void seven_segment_LED_set(int pattern_code){
     /* 0～9 ならその数値の表示パターンを設定する */
     if( (0 <= pattern_code) && (pattern_code <= 9) ){
     PD.DR.BYTE.HH = seven_segment_LED_
                    number_pattern[pattern_code];
     }
}
```

第4章 シリアル通信制御プログラムの作成

4.1 調歩同期式シリアル通信 RS232C のしくみ

シリアル通信とパラレル通信，各種シリアル通信，RS232C の呼称で広く使われてきた調歩同期式[注1]シリアル通信について解説します．

● 通信機能はどのようなところに使われているか

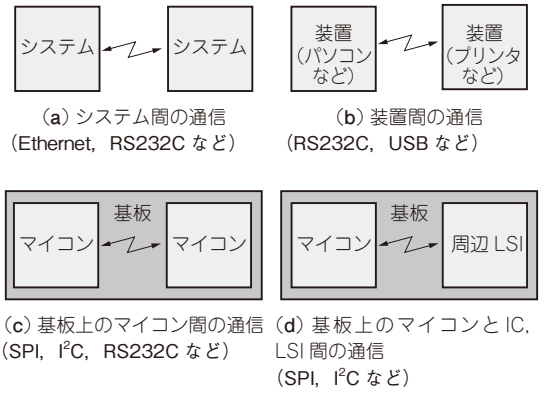
インターネットが生活の隅々まで普及している現在，「通信の意義と必要性」を改めて説く必要はないでしょう．

図 4.1 に示すように，組み込みの世界では，

▶システム間のデータ交換
▶装置間のデータ交換
▶基板上のマイコン間のデータ交換
▶マイコンと周辺 LSI の間のデータ交換

などに通信技術が使われています．

組み込み教育＆開発プラットホーム基板 CQK-SH2A は，システム間通信と装置間通信に，



(a) システム間の通信
(Ethernet，RS232C など)

(b) 装置間の通信
(RS232C，USB など)

(c) 基板上のマイコン間の通信
(SPI，I²C，RS232C など)

(d) 基板上のマイコンと IC，LSI 間の通信
(SPI，I²C など)

図 4.1　組み込みシステムで使われる通信

▶ USB 通信インターフェース
▶ Ethernet 通信インターフェース
▶ RS232C 通信インターフェース

を備えています．パソコンなどの外部装置あるいはほかのシステムとの通信プログラムの演習を行うことができます．

また，基板上では，

▶ SPI 通信(SH7285 ⟷ SD カード)
▶ I²C 通信(SH7285 ⟷ カレンダ IC)

を使っています．

本章では RS232C の呼称で広く使われてきた調歩同期式通信のプログラム演習を通して，シリアル通信の基本をマスタします．

● シリアル通信とパラレル通信

通信方式には，シリアル通信とパラレル通信があります．

シリアル通信は**図 4.2**(**a**)に示すように，1 本の伝送路を使ってデータを 1 ビットずつ「シリアル(直列的，連続的)」に送信する方式です．通信線は送信と受信に各 1 本だと合計 2 本になりますが，同期クロックや制御線が追加されて数本になることもあります．

一般に「1 本の伝送路にシリアルにデータを並べて順送りに送信する方式」をすべてシリアル通信と総称しています．

これに対して**図 4.2**(**b**)に示すように複数の通信線にデータを載せて送信する方式を「パラレル通信」と呼びます．10 年ほど前までパソコンとプリンタ間のデータ通信に使われていたセントロニクス仕様[注2]のインターフェースは典型的なパラレル通信です．

1 本の通信線でデータを送るより，複数の通信線を

注 1：RS232C シリアル通信でスタート・ビットを起点に，あらかじめ約束したビット・レートによりデータ・ビットを読み出す方式．
注 2：セントロニクス社(Centronics Data Computer 社)が開発したパラレル・ポート仕様のインターフェース規格で，コンピュータとプリンタを接続するのに広く使われてきた．

同時に使って送信するほうが大量のデータを高速に送れると考えられてきました．

図 4.3 に示すデータの 0 / 1 の変化の密度をビット・レート（通信速度）といいます．このビット・レートを一定とすれば，1 本の通信線より 8 本の通信線を使ったほうが 8 倍高速に通信ができます．

しかし 10 年ほど前から事情が変わりました．半導体技術および回路技術がどんどん進化してビット・レートが上がるにつれて，パラレル通信で使う複数の通信線間のスキュー（到達時間差）などが問題になってきました．

その結果，現在はシリアル通信全盛時代です．インターネットの世界で使われる Ethernet 通信，パソコン機器間の通信に使われる USB，制御基板上のデバイス（IC，LSI）間の通信に使われる SPI，I^2C，パソコン内の PCI Express バスなど，すべてシリアル通信です．

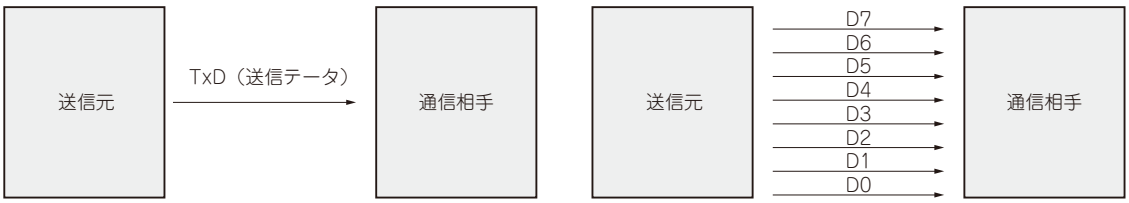
● シリアル通信方式…全二重通信と半二重通信，クロック同期式と調歩同期式

シリアル通信にはいくつかの方式があります．**図 4.4** に示す半二重通信は送信と受信を同時に行うことができません．送信と受信でデータ通信路を共有したり，送受信ハードウェアを共有したりしている場合は，この方式になります．

図 4.5 の全二重通信方式は送信と受信を同時に並行して行えます．

図 4.6 はシリアル通信のデータ同期のしくみを説明するものです．シリアル通信では 1 本のデータ線に 0 か 1 のビット・データが順次送られてきます．どのタイミングで各ビット・データを取り込むかを決めるため，なんらかの同期方式を備える必要があります．

図 4.7 はクロック同期方式です．データ線のほかに同期用のクロック信号線を備えていて，クロックの立ち上がり（もしくは立ち下がり）エッジでデータを取り込みます．



（a）シリアル通信　　（b）パラレル通信（8 ビット・パラレルの場合）

TxD: Transmit exchange Data

図 4.2　シリアル通信とパラレル通信

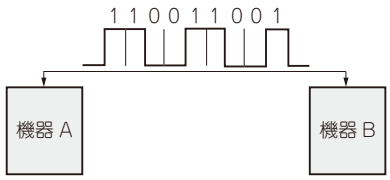


図 4.3
ビット・レート
データの 0 / 1 の変化の密度をビット・レートという．

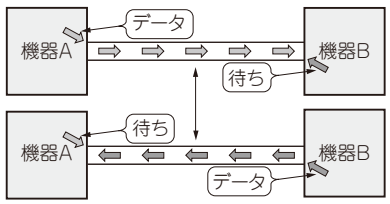


図 4.4
半二重通信
送信と受信は同時にできない．

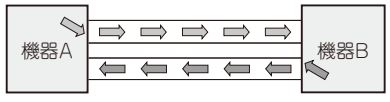


図 4.5
全二重通信
送信と受信を同時に並行して行うことができる．

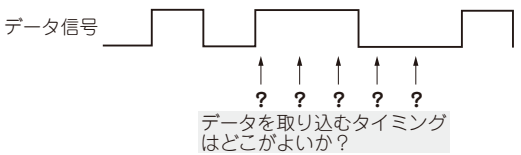


図 4.6　シリアル通信はデータの同期が重要
どのタイミングでビット・データを取り込むか…シリアル通信は同期方式を考慮する必要がある．

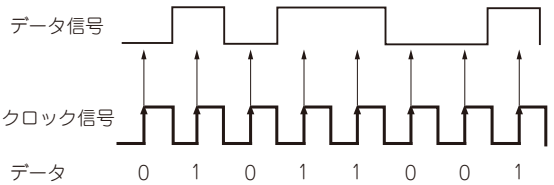


図 4.7　クロック同期方式
クロック同期方式のシリアル通信．データ信号線のほかに，同期用クロック信号線が必要．

第5章 数や時間を計測するカウンタ/タイマの使い方

5.1 カウンタ/タイマのしくみと機能

● 数を数えるカウンタ，時間の経過をはかるタイマ

カウンタは数を数える機能です．具体的には，モータの回転数やゲートを通過する人数を数えることができます．

モータが回転すると，エンコーダからパルスが発生します．そのパルスを数えることにより，モータの回転角や回転数を正確に知ることができます．この機能は，組み込み機器の速度制御や位置制御に欠かせません．

タイマは時間の経過をはかる機能です．一定の時間間隔で発生する基準パルスをカウンタで数えることにより時間を計測します．

例えば100m走のランナーがスタートすると同時に1 ms間隔のパルスのカウントを開始したとします．ゴールのテープを切った瞬間のカウント数が，

10250

であれば，このランナーのタイムは，

10.250秒

という具合です．

カウンタ/タイマはいずれも組み込みシステムの定番機能で，制御用のマイコンにはかならず内蔵されています．そして不思議なことに「カウンタ/タイマ」とセットにして扱われています．その理由は「一定の時間間隔で発生する基準パルスを数える」というタイマのしくみにあります．

● カウンタ/タイマでできること

タイマを使うと第2章の演習(**リスト2.4**)で，

```
for(i=0; i<25000; i++){  /* ③待ち時間 */
}
```

とプログラムの空ループで行った「およその時間待ち」処理を正確な時間処理に置き換えることができます．

また第6章で学ぶ割り込みと組み合わせると，「10msごとにスイッチを確認する」という正確な周期処理も可能です．

カウンタ/タイマを使って電源のON/OFF時間を制御することにより，モータの回転速度制御，ヒータのパワー制御などを行えます．この制御方式はPWM(Pulse Width Modulation)制御[注1]と呼ばれ，組み込みシステムの重要な制御手法になっています．

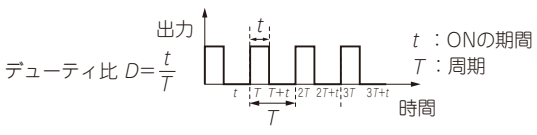
● SH-2Aマイコンに備わっている豊富なカウンタ/タイマ機能

C言語による開発が一般的になり，組み込みマイコンは，CPUコアの比較ではその開発スタイルに大きな違いが見えなくなりました．

しかし内蔵周辺機能はチップごとに大きな違いがあります．組み込み制御の分野で長い歴史をもつSuperHシリーズのカウンタ/タイマは，その機能の豊富さに驚かされます．

ひょっとすると機能の豊富さに戸惑う人がいるかも知れません．なにしろ今回使用するSH-2Aシリーズのマイコン SH7285のハードウェア・マニュアルはカウンタ/タイマの解説だけで246頁程度もあります．本章の解説は20頁程度ですから，すべてを解説することはできません．しかし，組み込みマイコンのカウ

注1：パルス幅変調のこと．図に示すようにパルス波のONの時間とOFFの時間の比率を変えて電流や電圧，電力を制御する方法．PWM波形は図のように周期やデューディ比によって決まる．



ンタ/タイマの使い方の基本をマスタすることはできます.

あとは実際の開発業務の中でいろいろな使い方を経験すればスキルアップを図ることができます.

● カウンタ/タイマのしくみ

図 5.1 はカウンタ/タイマの一般的な構成です.

まず，①でタイマ動作の基本となる源クロックを選択します．CPU のシステム・クロックもしくはその分周クロック，外部端子からのクロックなどから選びます.

②のプリスケーラは基本クロックを分周します．長時間タイマが必要な場合は，基本クロックそのままでは高速すぎることがあります．そのようなときはプリスケーラを 1/16，1/256 などに設定します.

③は実際のカウンタです．プリスケーラの分周クロックをカウントします．図のようにカウント・アップ動作もしくはカウント・ダウン動作が選択できるものもあります.

カウンタは基本的にカウント動作(アップもしくはダウン)を続けるだけです．次のようにカウント・アップしていきます.

0→1→2→3→…→65534→65535→0…

フル・カウント(16 ビット・カウンタの場合 65535)になると，0 に戻り，再びカウント動作を続けます.

このように単純にカウント・アップ動作を続けるカウンタを「フリーラン・カウンタ」と呼びます.

④の比較レジスタは，③のフリーラン・カウンタと比較して，設定値が一致した瞬間にフラグをつけたり，割り込みを発生させてプログラムに知らせたりするためのレジスタ[注2]です.

⑤のステータス・レジスタはカウンタの状態を保持するレジスタです．カウンタが比較値を超えた場合は「オーバフロー」，比較値を下回った場合は「アンダフロー」，比較値と一致した場合は「マッチ」のフラグを付けてプログラムに知らせます.

プログラムは，必要に応じてレジスタ値を読むことにより，イベントの発生を知ります．また状態が変化したときに割り込みを発生させることもできます.

5.2 SH7285の CMT のしくみとプログラミング

SH7285 には CMT(Compare Match Timer；コンペア・マッチ・タイマ)が内蔵されています．**図 5.2** の CMT のブロック図に示すように，同じカウンタ・モジュールが 2 組内蔵されています.

クロック源は 4 種類から選択できます．ペリフェラル・クロック・プリスケーラは備えていません.

CMT はシンプルなタイマで，制御レジスタは次の四つです.

▶ CMSTR(コンペア・マッチ・タイマ・スタート・レジスタ)

CMSTR はモジュール共通のレジスタで，各モジュールのタイマ・カウント・スタート/ストップを制御します.

▶ CMCSR(コンペア・マッチ・タイマ・コントロール/ステータス・レジスタ)

CMCSR はカウンタ・クロックの選択，割り込みの制御とコンペア・マッチの有無を調べるレジスタです.

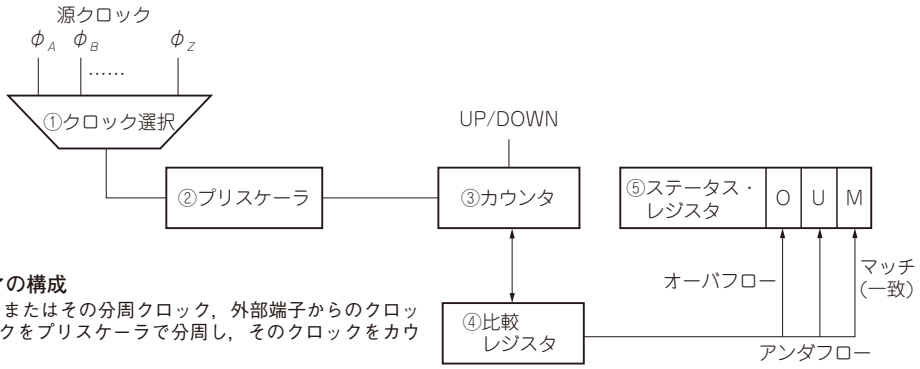


図 5.1
一般的なカウンタ/タイマの構成
CPU のシステム・クロックまたはその分周クロック，外部端子からのクロックなどから選んだ源クロックをプリスケーラで分周し，そのクロックをカウンタに入力する.

注 2：コンピュータ内部でデータを一時的に保持する回路．レジスタは計算結果を一時的に保持したり，メモリを読み書きする際のアドレスを保持したり，プロセッサや周辺機器の動作状態を保持または変更したりする.

見本

インターフェース増刊 デジタルアイ Vol.43
CQ出版社 〒170-8461 東京都豊島区巣鴨1-14-2 TEL.(03)5395-2141(販売部)
定価2,520円 本体2,400円

内容・購入方法などにつきましては以下のホームページをご覧下さい.
内容　http://shop.cqpub.co.jp/hanbai//books/MIF/MIFZ201108.htm
購入方法　http://www.cqpub.co.jp/hanbai/order/order.htm

雑誌　01620-8
Ⓛ-9/14
4910016200812
02400