B-1037(1)

小型ティーチングペンダント

HG1H形
開発環境構築マニュアル

IDEC株式会社

ご注意

- 本マニュアルおよび開発支援ツールのプログラムに関するすべての権利は、IDEC 株式会社に帰属しています。弊社に無断で複製することはできません。
- 本マニュアルおよび開発支援ツールのプログラムの内容を、予告なく変更することがあります。
- 本マニュアルおよび開発支援ツールを運用した結果の影響につきましては、弊社は一切責任を負いませんのでご了承ください。
- 製品の内容につきましては万全を期しておりますが、ご不審の点や誤りなど、お気付きの点がございましたら、お買い求めの販売店または弊社営業所・出張所までご連絡ください。

商標について

Microsoft、Windows、Windows NTは、米国マイクロソフト社の商標です。
AdobeはAdobe System Incorporatedの商標です。
記載されているその他の会社名、製品名は、各社の商標または登録商標です。

出版履歴

2006年10月　初版発行
2007年11月　第二版発行

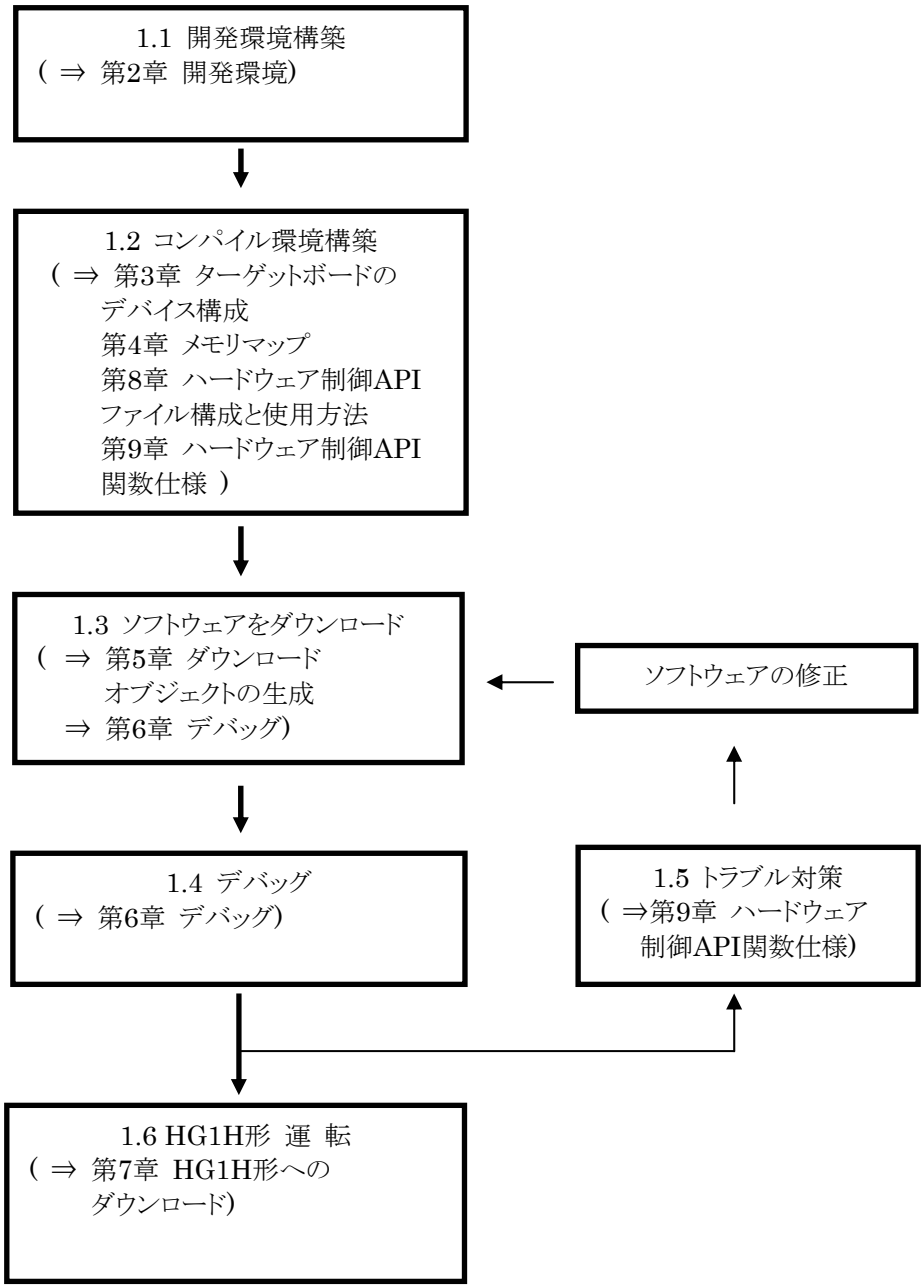
改訂履歴

出版履歴	改訂項目	改訂内容
初版	－	初版発行
第二版	定数定義	定数定義からHG_CLRを削除 定数定義にHG_ClrHomeを追加
	API関数一覧	概要項目の誤記を修正
	LCD制御関数	引数HG_CLRをHG_ClrHomeに変更、使用例を変更
	カーソル位置指定関数、 コントラスト調整関数	使用例の誤記を修正
	メンブレンスイッチ 全押下情報取得関数	使用例を追加

目　次

1. 開発手順

基本的な開発方式であるインサーキット・エミュレータを用いたときの開発環境構築から実機の運転動作までの一連フローを以下に示します。



1.1. 開発環境構築

開発環境を選定し、開発機材を購入します。インサーキット・エミュレータもしくはオンチップ・デバッグ・エミュレータを使用する場合は、弊社開発キット(HG9Z-HV1)も購入する必要があります。

1.2. コンパイル環境構築

コンパイラの設定をターゲットのメモリマップに合わせて設定し、ハードウェア制御APIのファイルをリンクします。続いてお客様ソフトウェアのコーディングを行います。

1.3. ソフトウェアのダウンロード

開発キットに同梱されているハードウェア制御APIのHEXファイルとコンパイラで出力されたお客様作成ソフトウェアをターゲットボードのエミュレーションメモリへダウンロードします。

1.4. デバッグ

インサーキット・エミュレータもしくはオンチップ・デバッグ・エミュレータ付属のデバッガでお客様作成ソフトウェアを動作させ、デバッグします。

1.5. トラブル対策

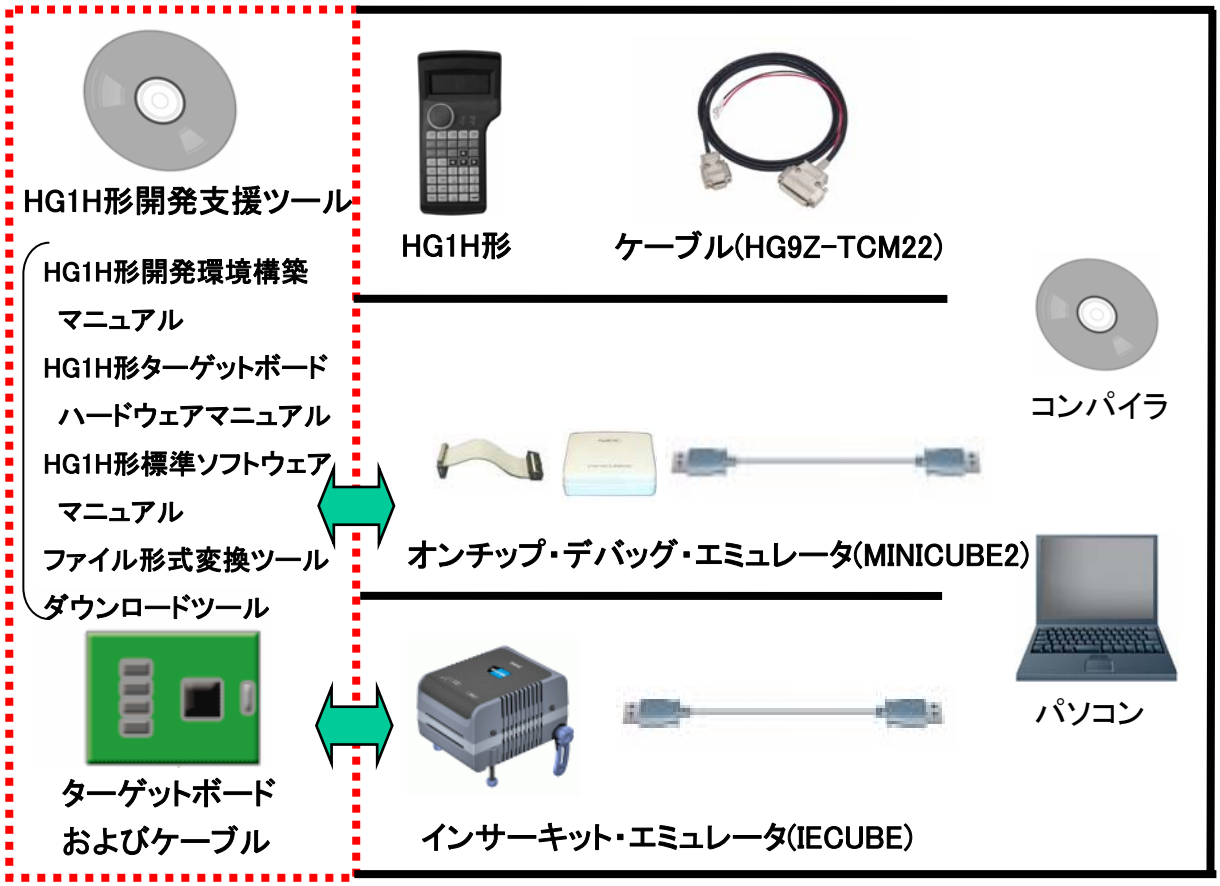
CPU、コンパイラ、デバッガ、ハードウェア制御APIのマニュアルに記述されている注意事項等を参照の上、不具合を調査し対策を検討します。

1.6. 運転

お客様作成ソフトウェアのHEXファイルを弊社ファイル形式変換ツールでバイナリ形式に変換し、ダウンロードツールでHG1H形にダウンロードを行い、運転動作をさせます。最終動作をHG1H形で再確認すれば、ソフトウェアの開発は終了です。

2. 開発環境

開発環境としては、以下のように弊社開発キットとインサーキット・エミュレータを使用してお客様作成ソフトウェアを開発する方法とHG1H形に直接ダウンロードして開発する方法がありますが、本書ではインサーキット・エミュレータを使用して開発する方法を中心に説明していきます。



HG1H形開発キットHG9Z-HV1

※ 開発キットを購入せずHG1H形でデバッグされる場合は、弊社Websiteより開発支援ツールおよびマニュアルを別途ダウンロードする必要があります。また、HG1H形がRS-422/RS-485タイプをご使用の場合は、別途お客様でパソコンと通信するためのケーブルおよび変換器をご用意頂けますようお願いいたします。

2.1. 開発キット

品名：開発キット（HG1H形搭載ソフトウェア開発用） [IDEC株式会社]

形番：HG9Z-HV1

開発キットにはHG1H形開発支援ツールとターゲットボードおよびケーブルが含まれますが、ハードウェア仕様については「HG1H形ターゲットボードハードウェアマニュアル」を参照してください。

---- 開発支援ツールCDROMの内容物 ----

フォルダ	ファイル	内容
ドキュメント	HG1H形ターゲットボード ハードウェアマニュアル.pdf	HG1H形ターゲットボード ハードウェアマニュアル
	HG1H形開発環境構築マニュアル.pdf	HG1H形開発環境構築マニュアル
	HG1H形標準システムソフト マニュアル.pdf	HG1H形標準システムソフトマニュアル
ソースコード	api.h	ハードウェア制御用APIヘッダファイル
	int_tbl.s	割り込み処理関数定義ファイル
	refresh_LCD.c	LCD表示リフレッシュ処理関数
オブジェクト	api.hex	インサーキット・エミュレータおよび オンチップ・デバッグ・エミュレーション 用APIオブジェクト
開発支援ソフトウェア	ConverttoHG1Hformat.exe	ファイル形式変換ツール
	DownloaderForHG1H.exe	ダウンロードツール
	clear.bin	内蔵ROM消去用バイナリファイル
ルート	Readme.txt	CDROM内のフォルダ構成および 取り扱い注意事項
	RevisionHistory.txt	開発支援ツールのバージョン履歴

2.2. インサーキットエミュレータ

品名：IECUBE [NECエレクトロニクス株式会社]

形番：QB-V850ESKX1H-S100GC

オプション機能などについてはお客様の開発環境に基づいて選定してください。

メイン・クロックについては4.9152Mzの発振子を使用してください。

2.3. オンチップ・デバッグ・エミュレータ

品名：MINICUBE2 [NECエレクトロニクス株式会社]

形番：QB-MINI2

2.4. コンパイラ

品名：CA850 [NECエレクトロニクス株式会社]

形番：μSAB17CA703000

※)ソフトウェア・パッケージSP850(μSAB17SP850)に同梱

2.5. デバッガ

品名：ID850 [NECエレクトロニクス株式会社]

形番：μSAB17ID703000

※)ソフトウェア・パッケージSP850(μSAB17SP850)に同梱

2.6. ハードウェア制御API

HG1H形には、ハードウェア制御APIおよび自己診断ソフトウェアを搭載しています。そのため、お客様でソフトウェアを開発する場合も、簡単にハードウェアデバイスの制御およびソフトウェアのダウンロード等を行うことができます。ソフトウェアを開発される前に「ハードウェア制御API」の章をお読みください。

2.7. 開発支援ソフトウェア

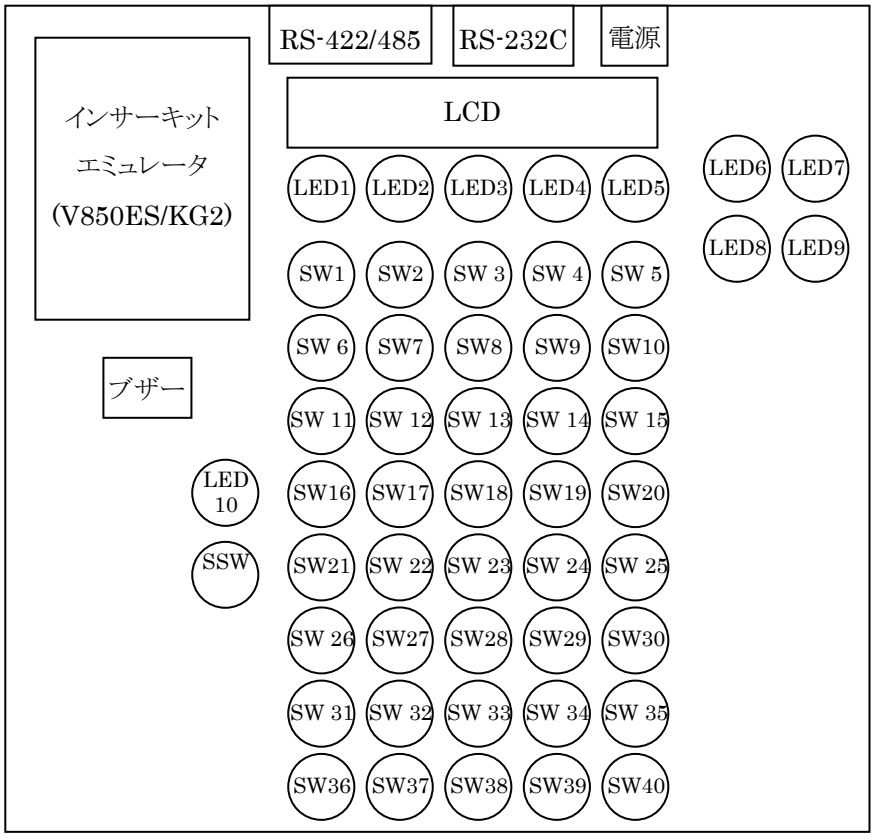
HG1H形のソフトウェア開発を支援するために、ファイル形式変換ツールおよびダウンロードツールが開発支援ツールCDROM内に用意されています。これらのソフトウエアのパソコン動作推奨環境は以下のとおりです。

項目	仕様
OS（基本ソフトウェア）	Window2000/XP
コンピュータ本体	PC-ATおよび互換機
CPU	Windowsが正常に動作するCPU(Pentium200MHz以上)
メモリ	64MB以上
ハードディスク	空きエリア10MB以上

3. ターゲットボードのデバイス構成

HG1H形ターゲットボードのデバイス構成は以下の通りです。詳細は「HG1H形ターゲットボードハードウェアマニュアル」をご参照ください。

- ***外部メモリ：フラッシュメモリ（512K）、SRAM（128K）***
  （本体では搭載あり/なしのタイプが存在）
- ***シリアル通信I/F(1ch.)：***
  ・RS-422/485：
  （RS-422/RS485の通信ポートをジャンパする：RS-485、ジャンパしない：RS-422）
  ・RS-232C
  （本体ではRS-422、RS-485もしくはRS-232Cのいずれかのタイプとなる）
- ***LCD***
- ***LED(9個+1個)***
- ***メンブレンスイッチ(40キー)：本体では標準35キー、最大40キー***
- ***ブザー***
- ***セレクタスイッチ(SSW)***



ターゲットボード

4. メモリマップ

4.1. メモリマップ

メモリアドレス	メモリ空間および用途	配置メモリ
3FFFFFFh 3FFF000h	CPU内蔵周辺I/O領域	
3FFEFFFh 3FFC800h	未使用	
3FFC7FFh 3FFC780h	予約領域 (128byte)	内蔵RAM (6kbyte)
3FFC77Fh 3FFB000h	ユーザ使用可能RAM領域 (約6kbyte)	
3FFAFFFh 0400000h	未使用	
03FFFFFh 0220000h	未使用	
021FFFFh 0200000h	ユーザ使用可能SRAM領域 (128kbyte)	外部SRAM (128kbyte) (メモリ搭載機種のみ)
01FFFFFh 0180000h	未使用	
017FFFFh 0100000h	ユーザ使用可能ROM領域 (512kbyte)	外部フラッシュメモリ (512kbyte) (メモリ搭載機種のみ)
00FFFFFh 0020000h	未使用	
001FFFFh 0008000h	ユーザ使用可能ROM領域 (96kbyte)	内蔵ROM (128kbyte)
0007FFFh 0000000h	予約領域 (32kbyte)	

予約領域はAPIおよび自己診断プログラムの領域として使用していますので、お客様のソフトウェアでは絶対に使用しないでください。使用された場合、正常な動作は保証致しかねます。
お客様作成ソフトウェアは内蔵ROMまたは外部フラッシュメモリ内のどちらかのユーザ使用可能ROM領域の先頭アドレスに配置してください。
パラメータ格納用API(ハードウェア制御APIの章を参照)を使用する場合、内蔵RAMの予約領域は3FFB000h-3FFB007h, 3FFC000h -3FFC7FFhになります。

4.2. お客様作成ソフトウェアのリンケージマップについて

お客様作成ソフトウェアの先頭から64バイトはヘッダー領域として最終的に「ファイル形式変換ツール」で作成されます。また、0x040以降には割り込み処理関数のアドレステーブルがハードウェア制御APIの割り込み処理関数定義ファイル(int_tbl.s)により自動的に登録されますので、お客様作成ソフトウェアは、0x110以降に配置するようにリンクディレクティブの設定を行ってください。

xxxx000h	ヘッダー情報領域	お客様作成ソフトウェアを弊社開発キットに付属されているファイル形式変換ツールでバイナリ形式に変換したときに、付加されます。
xxxx040h	お客様作成ソフトウェア （割り込み処理関数アドレステーブル）	割り込み処理関数定義ファイル(int_tbl.s)に使用する関数を定義し、リンクディレクティブファイルにてこのメモリ位置に配置されるように設定してください。
xxxx110h	お客様作成ソフトウェア （本体）	

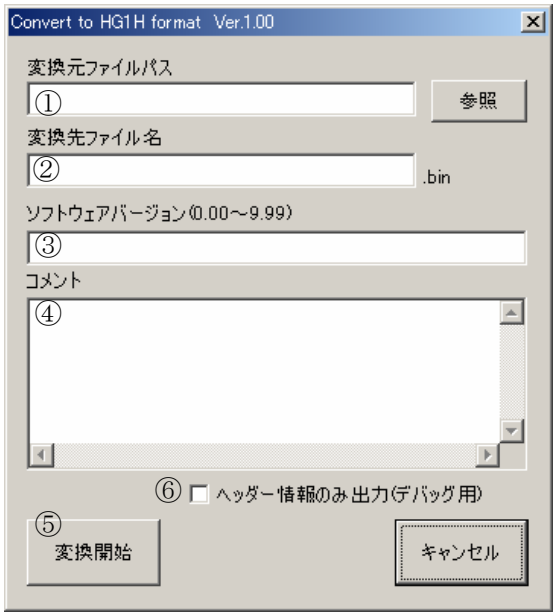
xxxx：　0008（内蔵ROM）
　　0100（外部ROM）

5. ダウンロードオブジェクトの生成

インサーキットエミュレータを使用してお客様ソフトウェアをターゲットのエミュレーションメモリへダウンロードする場合は、実行可能なオブジェクト形式であれば問題ありません。
ただし、ダウンロードツールでHG1H形にお客様作成ソフトウェアをダウンロードする場合は、ヘッダ情報付きのバイナリ形式ファイルが必要です。
コンパイラによりモトローラ形式(モトローラタイプS32ビット)のファイルを作成後、弊社が提供する「ファイル形式変換ツール」を使用してすることにより、ヘッダー情報をつけてバイナリ形式に変換します。

5.1. ファイル形式変換ツール

“ConverttoHG1Hformat.exe”を実行すると下記のダイアログボックスが起動し、各項目を設定し、変換開始を押下するとバイナリ形式のファイルが生成されます。また、統合開発環境PM+において[ビルド]-[ビルド設定]-[ビルド後の処理]にコマンドとして追加することにより、ビルド時にバイナリ形式に変換することもできます。このとき、引数として各項目を設定しておくとダイアログボックスは立ち上がらず、自動的にバイナリ形式へ変換することができます。



① 変換元ファイルパスに変換元となるモトローラ形式のファイルを選択してください。
② 変換先のファイル名を指定してください。
変換元ファイルがあるフォルダにデータを生成します。
③ 0.00～9.99のバージョンを設定してください。
④ ソフトウェア名、製品型番等のコメントを入力してください。(半角30文字以内)
⑤ 変換開始してください。
⑥ チェックボックスにチェックをするとデバッグ時にソフトウェア情報読み出しのハードウェア制御用APIで参照するヘッダー部分を作成することができます。

ソフトウェアバージョンとコメントはハードウェア制御用API関数で参照することが可能です。

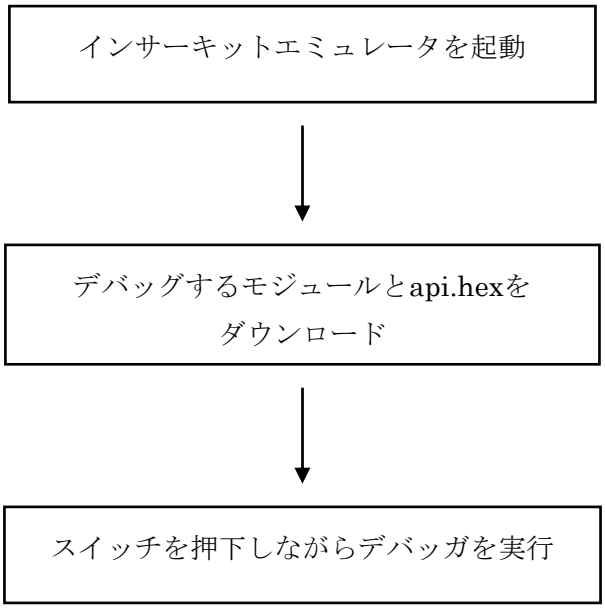
コマンドラインで実行する場合の引数の設定方法は次の通りです。
-S変換元ファイル名 -D変換先ファイル名 -Vバージョン -Cコメント (スペース区切り)
引数の数が不足している場合や不正な場合は、ダイアログボックスやエラーメッセージが表示されますので設定値を修正して変換してください。

変換先ファイル名がすでに存在している場合は上書き確認のメッセージが表示されます。ただし、コマンドラインで引数を設定してツールを実行した場合は、上書き確認のメッセージは表示されず、常に上書きで作成されます。

6. デバッグ

インサーキットエミュレータを使用してデバッグする場合、デバッグするモジュール以外にハードウェア制御APIの実行ファイルをダウンロードする必要があります。デバッグ時には"api.hex"のファイルをお客様作成ソフトウェアと同時にダウンロードしてください。
また、HG1H形では、起動時にお客様作成ソフトウェアのヘッダー情報を照合して動作しますので、デバッグ時にヘッダー情報がない場合お客様作成ソフトウェアが動作しません。よって、下記のようにキースイッチを2点同時に押下しながらデバッガを動作させることにより、ヘッダー情報がなくてもお客様作成ソフトウェアを実行させることができます。キースイッチの配置については「3. ターゲットボードのデバイス構成」を参照してください。

SW1+SW37：内蔵ROMのお客様作成ソフトウェア本体を起動
SW1+SW38：外部ROMのお客様作成ソフトウェア本体を起動



外部ROMにアクセスするためには以下の操作を行ってください。
①デバッガ起動後、api.hexをダウンロード
②FFFFF044hにHalf WordでWrite時にBreak Pointを設定
③ブレイク後、外部ROMにアクセス可能
また、外部ROMのデバッグを行うためにはインサーキットエミュレータにオプション機能をつける必要があります。

7. HG1H形へのダウンロード

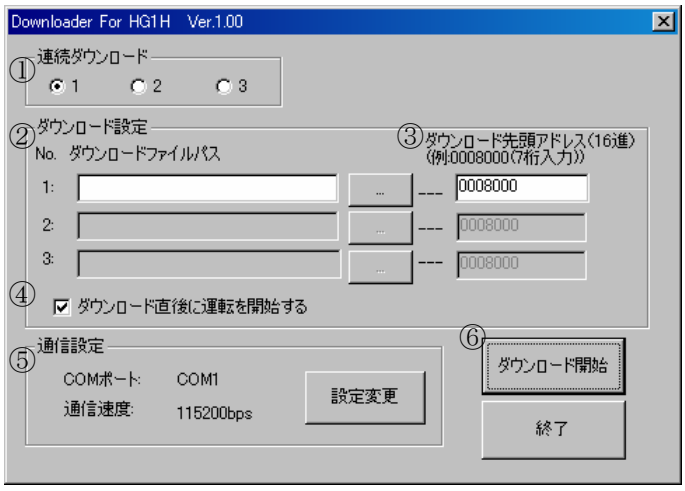
HG1H形へのお客様作成ソフトウェアの搭載はパソコンからダウンロードすることにより行います。お客様作成ソフトウェアをダウンロードする場合は、必ずホスト機器の運転を止め、弊社自己診断ソフトを起動して初期画面を表示した状態で行ってください。弊社自己診断ソフトはメンブレンスイッチ最上段の両端を同時に押下しながら電源を投入することにより起動します。

パソコンからは、ダウンロードツールを用いてHG1H形へ簡単にダウンロードすることができますが、ホスト機器からダウンロードする場合は、通信プロトコルの開示が必要となりますので、弊社営業までお問い合わせください。また、RS-422およびRS-485の機種にソフトウェアのダウンロードを行う場合、パソコンとHG1H形の間に通信方式を変換する変換器とケーブルを別途ご用意ください。

7.1. ダウンロードツール

“DownloaderForHG1H.exe”を実行すると下記のダイアログボックスが起動し、各項目を設定し、ダウンロード開始を押下するとファイルのダウンロードを開始します。連続ダウンロードを指定することにより、お客様作成ソフトウェア(プログラム)に加えて、内蔵ROMや外部ROMに複数の固定データを続けてダウンロードすることができます。

内蔵ROMまたは外部ROMにお客様作成ソフトウェアをダウンロードした場合、以前にダウンロードされていたソフトウェア(プログラム)は自動的に消去されます。



① 連続してダウンロードするファイル数を選択してください。

② ダウンロードするデータファイルを設定してください。

③ ダウンロードする領域の先頭アドレスを設定してください。

アドレス範囲:内蔵ROM 0x0008000～0x001F800 (0x800単位)

外部ROM 0x0100000～0x01F0000 (0x10000単位)

④ ダウンロード直後に運転を開始するかどうかを設定してください。

⑤ 通信ポートと通信速度を設定してください。

「設定変更」を押下すると通信設定のダイアログボックスが開き、設定できます。

⑥ ①～⑤を設定したあと、ダウンロード開始してください。

ダウンロードするファイルは必ず2kbyte単位になるように調整してください。弊社の変換ツールを使用してファイルを作成された場合は、自動的に2kbyte単位に調整されます。
ファイルをダウンロードする先頭アドレスやファイルサイズが不正な場合、通信設定が不正な場合は、エラーメッセージが表示されますので設定値を修正してダウンロードしてください。

注意）ダウンロード中はHG1H形本体やパソコンの電源を切らないでください。ダウンロード中に電源を切った場合、正常に終了しません。

8. ハードウェア制御APIファイル構成と使用方法

ここでは、HG1H形のハードウェアを制御するAPIについて、そのファイル構成と使用方法を記述しています。HG1H形が搭載しているCPUは[NECエレクトロニクス株式会社]製のV850ES/KG2(μPD70F3731)ですので、そちらのユーザーズマニュアルと合わせてご使用ください。

フォルダ	ファイル	内容
ソースコード	api.h	ハードウェア制御用APIヘッダファイル
	int_tbl.s	割り込み処理関数定義ファイル
	refresh_LCD.c	LCD表示リフレッシュ処理関数

- API関数を使用する場合、必ず"api.h"をインクルードしてください。また、割り込みを使用する場合、必ず"int_tbl.s"に使用する関数を定義し、リンクディレクティブファイルにて関数のテーブルをセクションに割り付けてください。
- お客様作成ソフトウェアのファイルと共に"refresh_LCD.c"をコンパイルしていただき、hg_resfesh_init関数を必ず1度実行してください。hg_refresh_LCD関数にて画面の全領域が周期的(400msec)にリフレッシュされます。もしくは、お客様作成ソフトウェアにてhg_refresh_LCD関数を、400msec毎に起動するようにしてください。これは、LCDコントローラ内の表示データが強いノイズで変化した場合の対策となります。
- API関数が用意されているハードウェアの制御は必ずAPI関数を通して行ってください。CPUのI/Oポートをお客様作成ソフトウェアで直接制御すると、ハードウェアのロットによって正しく動作しなくなるおそれがあります。
- ポート機能、バス制御機能、クロック発生機能、A/D・D/Aコンバータ、CSI0、CSIA、I2Cバスに関するレジスタはAPIで使用しますので操作しないでください。
- シリアル通信に関しては、アシンクロナス・シリアル・インタフェース(UART2)の機能を使用してください。
- コンパイラのリンクディレクティブでは、tp(テキストポインタ)を必ず0番地に設定してお客様作成ソフトウェアでは動的に値を変化させないようにしてください。割り込み処理が正常に動作しないことがあります。

(記述例)

(リンクディレクティブファイル内)

```
_tp_TEXT@%TP_SYMBOL
```

(スタート・アップ・ルーチン内)

```
.extern _tp_TEXT, 4
mov   #_tp_TEXT, tp
```

9. ハードウェア制御API関数仕様

9.1. 型定義

HG1H形ハードウェア制御APIで使用する型は以下の通りです。

定義	説明
char	符号付8ビット整数
unsigned char	符号無8ビット整数
int	符号付32ビット整数
unsigned int	符号無32ビット整数
HG_KEY_INFO	メンブレンスイッチ押下情報
HG_TARGET_INFO	ハードウェア情報
HG_SYSTEM_INFO	ソフトウェア情報

9.2. 構造体定義

HG1H形ハードウェア制御ドライバで使用する構造体は以下の通りです。

定義	説明
HG_KEY_INFO	typedef struct{ 　char　　key_table[5][8] }KEY_INFO

HG_TARGET_INFO	typedef union{
	　unsigned int　　WORD[2]:
	　struct{
	　　unsigned int　　COM　　:2;通信方式種別
	　　unsigned int　　BACKLIGHT :2;バックライトの有無
	　　unsigned int　　MEMORY　:2;メモリの有無
	　　unsigned int　　POWER　　:2;電源種別
	　　unsigned int　　REVISION　:4;ターゲットバージョン
	　　unsigned int　　reserve　　:4;予約
	　　unsigned int　　API_VER　:16;APIバージョン
	　　unsigned int　　reserve　　:8;予約
	　　unsigned int　　reserve　　:24;予約
	　}BIT;
	}TARGET_INFO

HG_SYSTEM_INFO	typedef struct{ 　int reserve[6]　　:予約　　: 　char version[2]　　:システムソフトバージョン 　　(version[0]－整数部:00-, 　　version[1]－小数部:00-　共にBCD2桁) 　char reserve[2]　　:予約 　int reserve　　:予約 　char name[]　　:コメント（ASCIIコード） 　char dummy　　:サイズ調整領域（領域を64byteに調整） }SYSTEM_INFO

9.3. 定数定義

HG1H形ハードウェア制御ドライバで使用する定数定義は以下の通りです。

定義	値	説明
HG_OFF	0x00	LEDやLCD、ブザー等をOFF
HG_ON	0x01	LEDやLCD、ブザー等をON
HG_ClrHome	0x81	全表示をクリア後、カーソル位置を(0,0)にセット
HG_DspON	0x0c	カーソル表示なしで表示をON
HG_CurON	0x0e	カーソル表示ありで表示をON
HG_Blink	0x0d	カーソル位置の文字を点滅
HG_DspOFF	0x08	表示をOFF
HG_CurSiftR	0x94	カーソルを右にシフト
HG_CurSiftL	0x90	カーソルを左にシフト

9.4. エラーステータス一覧

HG1H形ハードウェア制御ドライバで使用するエラーステータスは以下の通りです。

定義	値	説明
HG_NO_ER	0	正常終了
HG_ER	-1	異常終了
HG_BUSY	1	処理中

9.5. 割り込み処理

割り込み処理は割り込み処理関数分岐用のテーブルを介して行います。お客様作成ソフトウェアでは、割り込み処理の定義をせず、通常の関数として作成し、弊社開発キットに付属しているint_tbl.sに使用する関数の設定を行います。分岐用テーブルのメモリマップは環境開発資料を参照してください。

使用可能な割り込みは以下の通りです。

割り込み名称	使用	割り込み名称	使用	割り込み名称	使用
RESET	×	INTTM011	○	INTBRG	○
NMI	×	INTTM50	△	INTTM020	○
INTWDT1	○	INTTM51	△	INTTM021	○
INTWDT2	○	INTCSI00	×	INTTM030	△
TRAP0n	○	INTCSI01	×	INTTM031	△
TRAP1n	×	INTSRE0	×	INTCSIA1	×
ILGOP	×	INTSR0	×	INTSRE2	○
DBG0	×	INTST0	×	INTSR2	○
INTWDTM	○	INTSRE1	×	INTST2	○
INTP0	×	INTSR1	×	INTP7	×
INTP1	×	INTST1	×	INTTP0OV	○
INTP2	△	INTTMH0	○	INTTP0CC	○
INTP3	○	INTTMH1	○	INTTP0CC	○
INTP4	×	INTCSIA0	×	INTDMA0	○
INTP5	×	INTIIC0	×	INTDMA1	○
INTP6	×	INTAD	○	INTDMA2	○
INTTM000	○	INTKR	○	INTDMA3	○
INTTM001	○	INTWTI	○		
INTTM010	○	INTWT	○		

○:割り込み可、△:制限付で割り込み可、×:割り込み不可

内蔵ROM関連のAPIを使用する場合、TRAP10、INTTM50、51は使用しないでください。

LCDリフレッシュ関数を使用する場合、INTTM030、INTTM031は使用しないでください。

タイマに関して入力・出力端子は他の機能として使用していますので、端子への出力機能は使用しないでください。

割り込み処理は多重割り込みを行う場合もそうでない場合も、通常の関数として定義してください。

割り込み関数内で割り込み許可(__EI();)・禁止(__DI();)の制御を行うだけで、多重割り込みを実現することができます。

9.6. API関数一覧

関数名	概要	リエントラント
hg_ctrl_LED	LEDの点灯/消灯を制御	×
hg_ctrl_BZ	ブザーのON/OFFを制御	×
hg_get_SSW	セレクタスイッチの状態を取得	○
hg_ctrl_LCD	LCDの表示/カーソルを制御	×※1
hg_dsp_adr	LCDのカーソル位置を指定	×※1
hg_dsp_strings	LCDに文字を表示	×※1
hg_refresh_LCD	LCDの表示をリフレッシュ	×※1
hg_ctrl_Contrast	LCDのコントラストを制御	×
hg_ctrl_Backlight	LCDのバックライトを制御	×
hg_write_externalROM	外部フラッシュメモリへのデータの書き込み	×※2
hg_erase_externalROM	外部フラッシュメモリの消去	×※2
hg_get_keyinfo_all	メンブレンスイッチ全押下情報の取得	×※3
hg_get_keyinfo_indiv	メンブレンスイッチ個別押下情報の取得	×※3
hg_comopen	通信インタフェースの初期化	×
hg_ctrl_422	RS-422/485送受信イネーブル信号の出力	×
hg_ctrl_232CS	RS-232C通信送信許可信号の入力	×
hg_ctrl_232RS	RS-232C通信受信許可信号の出力	×
hg_get_targetinfo	ハードウェア情報の取得	○
hg_get_systeminfo	ソフトウェア情報の取得	○
hg_init_internalROM	内蔵ROM書き込み初期化	×※4
hg_write_internalROM	内蔵ROM書き込み	×※4
hg_read_internalROM	内蔵ROM読み出し	×※4

○:リエントラント可、×:リエントラント不可

「リエントラント」とは「再入可能」という意味であり、リエントラント可能な関数は、実行中に他のプロセスでその関数実行しようとした場合でも正しく実行できます。

リエントラント不可の関数で同じデバイスを制御する関数(※1～※4)について、排他制御処理はお客様作成ソフトウェアで行ってください。

内蔵ROM関連の関数ではマスカブル割り込みの禁止/許可を行っています。

9.7. API関数詳細

9.7.1. LED

9.7.1.1. LED制御関数

➢ *機能*

点灯/消灯の制御

➢ *関数名*

char hg_ctrl_LED(char led_no, char on_off)

➢ *引数*

型	引数	説明
char	led_no	LED番号 1-10: 制御するLEDの番号 0xff: 全LEDを制御
char	on_off	LEDの点灯/消灯 HG_ON: 点灯 HG_OFF: 消灯

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

制御するLED番号とその状態を指定することにより、LEDを点灯/消灯します。

第一引数で点灯/消灯するLEDの番号を指定します。0xffを指定した場合はすべてのLEDを点灯/消灯します。

第二引数でHG_ON(1)を指定した場合はLEDを点灯し、HG_OFF(0)を指定した場合はLEDを消灯します。範囲外の値が指定された場合、異常終了します。

範囲外のLED番号が指定された場合、異常終了します。

LEDのレイアウトは「3.ターゲットボードのデバイス構成」および「HG1H形ターゲットボードハードウェアマニュアル」を参照してください。LED10は照光セレクタスイッチのLEDです。

➢ *使用例*

LED1を点灯させる場合

```
char ret ;
/* LED1を点灯  */
ret = hg_ctrl_LED ( 1, HG_ON ) ;
```

9.7.2. ブザー

9.7.2.1. ブザー制御関数

➢ *機能*

ブザーON/OFFの制御

➢ *関数名*

char hg_ctrl_BZ(char on_off)

➢ *引数*

型	引数	説明
char	on_off	ブザーのON/OFF HG_ON: ブザー音を鳴らす HG_OFF: ブザー音を止める

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

引数でHG_ON(1)を指定した場合はブザーを鳴らし、HG_OFF(0)を指定した場合はブザーを止めます。範囲外の値が指定された場合、異常終了します。

➢ *使用例*

ブザーを鳴らす場合

```
char ret ;
/* ブザーを鳴らす */
ret = hg_ctrl_BZ ( HG_ON ) ;
```

9.7.3. セレクタスイッチ

9.7.3.1. セレクタスイッチ制御関数

➢ *機能*

セレクタスイッチの状態を取得

➢ *関数名*

char hg_get_SSW(void)

➢ *引数*

なし

➢ *戻り値*

戻り値	説明
HG_ON	セレクタスイッチ接点情報
HG_OFF	セレクタスイッチ接点情報

➢ *仕様*

セレクタスイッチの状態を取り込み、戻り値として返します。

✎ 照光式セレクタスイッチのLEDの点灯/消灯を連動させる場合は、LED制御関数の第一引数でセレクタスイッチのLEDを指定し、第二引数としてこの戻り値を設定してください。

➢ *使用例*

照光式セレクタスイッチの状態を取り込み、そのLEDを制御する場合

```
char status , ret ;
/*  セレクタスイッチの状態を取得  */
status = hg_get_SSW ( ) ;
/*  セレクタスイッチのLEDを制御  */
ret = hg_ctrl_LED ( 10, status ) ;
```

9.7.4. LCD

9.7.4.1. LCD制御関数

➢ *機能*

表示/カーソルの制御

➢ *関数名*

char hg_ctrl_LCD(char command)

➢ *引数*

型	引数	説明
char	command	LCD制御コマンド HG_ClrHome: 全表示をクリア後、カーソル位置を(0,0)にセットします。 HG_DspON: カーソル表示なしで表示をONします。 HG_CurON: カーソル表示ありで表示をONします。 HG_Blink: カーソル位置の文字を点滅します。 HG_DspOFF: 表示をオフします。 HG_CurSiftR: カーソルを右にシフトします。 HG_CurSiftL: カーソルを左にシフトします。

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了
HG_BUSY	処理中

➢ *仕様*

指定した引数に従って、LCDの表示を制御します。

引数に制御コマンド以外が指定された場合、異常終了となります。

カーソルが右端にある場合に、右シフトを行うとカーソルは次行の左端に移動します。また、最終行の右端にある場合には、先頭行の左端に移動します。

カーソルが左端にある場合に、左シフトを行うとカーソルは前行の右端に移動します。また、先頭行の左端にある場合には、最終行の右端に移動します。

カーソルとは文字下に表示される下線のことを示します。

戻り値が処理中の場合、コマンドは受け付けられませんので、その場合は、再度制御コマンドを送るようにしてください。

➢ *使用例*

LCDの表示をクリアしてカーソル位置を(0,0)にセットする場合

```
char ret ;
/*  表示クリア  */
ret = hg_ctrl_LCD ( HG_ClrHome ) ;
```

9.7.4.2. カーソル位置指定関数

➢ *機能*

カーソル位置を指定

➢ *関数名*

char hg_dsp_adr(char x, char y)

➢ *引数*

型	引数	説明
char	x	カーソル位置のX座標 (0-19)
char	y	カーソル位置のY座標 (0-3)

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了
HG_BUSY	処理中

➢ *仕様*

カーソル位置を指定します。

LCDの座標は以下の通りです。

(0,0)	(0,1)	…	(0,18)	(0,19)
(1,0)	(1,1)	…	(1,18)	(1,19)
(2,0)	(2,1)	…	(2,18)	(2,19)
(3,0)	(3,1)	…	(3,18)	(3,19)

範囲外の座標が指定された場合は、異常終了となります。

戻り値が処理中の場合、コマンドは受け付けられませんので、その場合は、再度カーソル位置を指定してください。

➢ *使用例*

カーソル位置を(5,1)に移動する場合

```
char ret ;
/* カーソル位置を指定 */
ret = hg_dsp_adr ( 5, 1 ) ;
```

9.7.4.3. 文字表示関数

➢ *機能*

文字を表示

➢ *関数名*

char hg_dsp_strings(char *data)

➢ *引数*

型	引数	説明
char *	data	表示する文字列の先頭アドレス

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

カーソル位置より文字列を表示します。指定位置より文字列を表示する場合は、カーソル位置指定関数により表示開始のカーソル位置を指定してください。

文字列の最後には必ず、NULLを設定してください。

文字列が行の右端まで表示された場合、次行の左端より引き続いて文字列が表示されます。最終段の右端まで表示した場合、文字列描画は終了し、戻り値が異常終了となります。

文字列は常に現在の表示状態に上書きで表示されます。新たに文字列を表示する場合は、現在表示されている文字列を一度クリアしてから表示してください。

表示可能な文字コードは以下の通りです。

➢ *使用例*

表示をクリアし、(2, 3)の位置から文字列を表示する場合

```
char ret ;
/*  表示クリア  */
ret = hg_ctrl_LCD ( HG_CLR ) ;
/*  文字列表示位置を指定  */
ret = hg_dsp_adr ( 2, 3 ) ;
/*「HG1H」の文字列を表示  */
ret = hg_dsp_strings ( “HG1H” ) ;
```

9.7.4.4. コントラスト調整関数

➢ *機能*

コントラスト調整値出力

➢ *関数名*

void hg_ctrl_Contrast(unsigned char data)

➢ *引数*

型	引数	説明
unsigned char	data	コントラスト調整値（0:濃 ～ 60:淡）初期値 20

➢ *戻り値*

なし

➢ *仕様*

コントラストの調整値を出力します。

本関数では、コントラストの温度特性を補正する処理を行っていますので、一分程度毎に本関数を実行させることを推奨します。定期的に実行するときの引数は同じ値を使用してください。

➢ *使用例*

コントラスト調整値50を出力する場合

```
char ret ;
/*  コントラスト調整値出力  */
hg_ctrl_Contrast ( 50 ) ;
```

9.7.4.5. バックライト制御関数

➢ *機能*

バックライト点灯/消灯の制御

➢ *関数名*

void hg_ctrl_Backlight(unsigned char data)

➢ *引数*

型	引数	説明
unsigned char	data	バックライトの点灯/消灯（0-255）初期値 0

➢ *戻り値*

なし

➢ *仕様*

0を指定した場合はバックライトを消灯します。大きな値を指定するほどバックライトは明るくなり、255を指定した時にバックライトの明るさは最大となります。

バックライト搭載機種でのみ有効です。

➢ *使用例*

バックライトを最大の明るさで点灯する場合

```
char ret ;
/*  バックライト点灯  */
hg_ctrl_Backlight ( 255 ) ;
```

9.7.4.6. LCDリフレッシュ関数

➢ *機能*

LCD全領域の表示を更新します。

➢ *関数名*

char hg_refresh_LCD(void)

➢ *引数*

なし

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_BUSY	処理中

➢ *仕様*

LCD全領域の表示を更新します。

hg_resfesh_initを実行することにより、400msごとにLCDリフレッシュ関数が自動的に実行されます。

hg_resfesh_initで設定される周期タイマの優先順位はレベル7(最低位)となっています。LCDリフレッシュ用のタイマ内では多重割り込みが許可されていますので、より優先順位が高い割り込みについてはLCDリフレッシュ関数処理中でも受け付けられます。

LCD関連のAPI実行中にLCDリフレッシュ関数が実行された場合、戻り値が処理中となります。この場合、画面のリフレッシュは実行されていません。

hg_resfesh_initを実行し、タイマ内で周期的にリフレッシュ関数処理を行わない場合は、お客様作成ソフトウェアにて定期的にリフレッシュ関数を実行してください。尚、リフレッシュ関数の実行処理時間は約5msです。

9.7.5. 外部フラッシュメモリ消去・書き込み

外部フラッシュメモリの消去および書き込みを行う関数です。お客様作成ソフトウェアが外部フラッシュメモリで動作している場合は、関数実行中は割り込み禁止状態になるようにしておく必要があります。また、書き込み・消去する領域は、お客様作成ソフトウェアと重複しないようご注意願います。

メモリアドレス	メモリ空間および用途	配置メモリ
017FFFFh 0100000h	ユーザ使用可能ROM領域 (512kbyte)	外部フラッシュメモリ (512kbyte) (メモリ搭載機種のみ)

9.7.5.1. フラッシュメモリ消去関数

➢ *機能*

外部フラッシュメモリの消去

➢ *関数名*

char hg_erase_externalROM(int erase_adr, int erase_byte)

➢ *引数*

型	引数	説明
int	erase_adr	消去するフラッシュメモリの先頭アドレス（0x100000-0x170000）
int	erase_byte	消去するバイト数（64- 512 Kbyte）

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

指定したアドレスから指定したバイト数分だけフラッシュメモリを消去します。

バイト数は必ず64の倍数で指定してください。

消去する先頭アドレスは必ず0x10000の倍数で指定してください。

外部フラッシュメモリ搭載機種のみ有効です。

➢ *使用例*

アドレス0x110000の位置から128Kバイト消去する場合

```
char ret ;
/*  外部フラッシュメモリを消去  */
ret = hg_erase_externalROM(0x110000, 128 ) ;
```

9.7.5.2. フラッシュメモリ書き込み関数

➢ *機能*

外部フラッシュメモリへのデータの書込み

➢ *関数名*

char hg_write_externalROM(char *dst, char *src , int data_byte)

➢ *引数*

型	引数	説明
char *	dst	書き込み先データの先頭アドレス
char *	src	書き込み元データの先頭アドレス
int	data_byte	書き込みバイト数

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

書き込み元の先頭アドレスから書き込みバイト数分のデータを指定した書き込み先の先頭アドレスから書き込みます。

➢ *使用例*

4バイトのデータをアドレス0x100000の位置から書き込む場合

```
char ret ;
int data ;
/*  書き込むデータ  */
data = 0x12345678 ;
/*  データを外部フラッシュメモリへ書き込む  */
ret = hg_write_externalROM( (char *)0x100000, (char *)&data, 4 ) ;
```

9.7.6. メンブレンスイッチ制御

メンブレンスイッチの押下状態の取り込み関数は、すべて、ノイズやチャタリング防止用のソフトウェアフィルタ処理が組み込まれていますので、安心してお使い頂けます。

9.7.6.1. メンブレンスイッチ全押下情報取得関数

➢ *機能*

メンブレンスイッチ全押下情報の取得

➢ *関数名*

void hg_get_keyinfo_all(HG_KEY_INFO *key_info)

➢ *引数*

型	引数	説明
HG_KEY_INFO *	key_info	メンブレンスイッチ押下情報の先頭アドレス

➢ *戻り値*

なし

➢ *仕様*

メンブレンスイッチの押下情報を取得します。

メンブレンスイッチのレイアウトは以下の通りです。

SW1 (0,0)	SW2 (1,0)	SW3 (2,0)	SW4 (3,0)	SW5 (4,0)
SW6 (0,1)	SW7 (1,1)	SW8 (2,1)	SW9 (3,1)	SW10 (4,1)
…	…	…	…	…
SW36 (0,7)	SW37 (1,7)	SW38 (2,7)	SW39 (3,7)	SW40 (4,7)

3箇所以上同時押しされている場合、メンブレンスイッチの押下情報は動作保証外です。詳細についてはHG1H形の製品仕様書を参照してください。

➢ *使用例*

座標(4, 7)のスイッチが押下されていたらブザーを鳴らす場合

```
char ret ;
HG_KEY_INFO key_info ;

/*  スイッチ全押下情報を取得  */
hg_get_keyinfo_all( &key_info ) ;

/*  座標( 4, 7 )のスイッチが押下されていたら  */
if ( HG_ON == key_info.key_table[4][7] ){
 /*  ブザーを鳴らす  */
 ret = hg_ctrl_BZ ( HG_ON ) ;
}
```

9.7.6.2. メンブレンスイッチ個別押下情報取得関数

➢ *機能*

指定座標のスイッチ押下情報を取得

➢ *関数名*

char hg_get_keyinfo_indiv(char x, char y)

➢ *引数*

型	引数	説明
char	x	押下情報を取得するX座標(0-4)
char	y	押下情報を取得するY座標(0-7)

➢ *戻り値*

戻り値	説明
HG_ON	スイッチが押されている
HG_OFF	スイッチが押されていない
HG_ERR	異常終了

➢ *仕様*

指定した座標位置のメンブレンスイッチの押下状態を取得します。範囲外の座標位置が指定された場合、戻り値は異常終了となります。

3箇所以上同時押しされている場合、メンブレンスイッチの押下情報は動作保証外です。詳細についてはHG1H形の製品仕様書を参照してください。

➢ *使用例*

座標(4, 7)のスイッチが押下されていたらブザーを鳴らす場合

```
char sratus, ret ;
/*  スイッチ押下情報を取得  */
status = hg_get_keyinfo_indiv( 4, 7 ) ;
if ( HG_ON == status ){
   /*  ブザーを鳴らす  */
   ret = hg_ctrl_BZ ( HG_ON ) ;
}
```

9.7.7. シリアル通信

シリアル通信はV850のアシンクロナス・シリアル・インタフェース(UART2)を用いて行います。シリアル通信I/Fを用いて送受信を行う場合は、UART2の設定を行ってください。

9.7.7.1. 通信インタフェース初期化関数

➢ *機能*

通信インタフェースの通信速度および通信条件の設定

➢ *関数名*

char hg_comopen(char baud, char mode)

➢ *引数*

型	引数	説明
char	boud	通信速度 *)
char	mode	ストップビット、データ長、パリティ **)

*)

ビット	内容	設定値
0-4	通信速度	1: 300bps 2: 600bps 3: 1200bps 4: 2400bps 5: 4800bps 6: 9600bps 7: 19200bps 8: 38400bps 9: 57600bps 10: 115200bps
5-7	予約	0

**)

ビット	内容	設定値
0	予約	0
1	ストップビット	0: 1ビット 1: 2ビット
2	データ長	0: 7ビット 1: 8ビット
3-4	パリティ	00: なし 10: 奇数 11: 偶数
5-7	予約	0

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

指定した通信速度と通信条件で通信インタフェースを初期化します。

関数内でUART2を送信禁止および受信禁止にしています。送信許可、受信許可、送信割り込みマスク解除、受信割り込み解除は、お客様作成ソフトウェアにて行ってください。

引数に範囲外の値を指定すると異常終了します。

➢ *使用例*

通信速度:9600bps、ストップビット:1ビット、データ長:8ビット、パリティ:偶数 で通信条件を設定する場合

```
char ret ;
/*  通信ポート初期化  */
/* 9600bps, 8, 1, even */
ret = hg_comopen ( 6, 0x1c ) ;
```

9.7.7.2. RS-422/485送受信イネーブル信号制御関数

➢ *機能*

RS-422/485通信のドライバに信号を出力

➢ *関数名*

char hg_ctrl_422(char enable)

➢ *引数*

型	引数	説明
char	enable	送受信のイネーブル信号を出力 *)

*)

ビット	内容	設定値
0	送信データ出力イネーブル信号	HG_OFF(0):禁止 HG_ON(1):許可
1	受信データ入力イネーブル信号	HG_OFF(0):禁止 HG_ON(1):許可
2-7	予約	

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

送受信データイネーブル信号をドライバに出力します。

RS-422/485の機種でのみ有効です。

予約領域が0以外の場合、戻り値が異常終了となります。

➢ *使用例*

```
char ret ;
char tx_enable, rx_enable, enable ;
/*  受信データ入力イネーブル信号許可  */
rx_enable = HG_ON ;
/*  送信データ出力イネーブル信号禁止  */
tx_enable = HG_OFF ;
enable = (rx_enable << 1) | tx_enable ;
/* RS-422/485通信のドライバに信号を出力  */
ret = hg_ctrl_422( enable ) ;
```

9.7.7.3. RS-232C送信許可信号制御関数

➢ *機能*

RS-232C通信送信許可信号の制御

➢ *関数名*

char hg_ctrl_232CS (void)

➢ *引数*

なし

➢ *戻り値*

戻り値	説明
HG_ON	送信許可
HG_OFF	送信禁止

➢ *仕様*

RS-232C送信許可信号を読み込みます。

RS-232Cの機種でのみ有効です。

➢ *使用例*

送信許可状態であれば、データを送信する場合

```
char status ;
/* RS-232C送信許可信号を取得 */
status = hg_ctrl_232CS ( ) ;
/* 送信許可ならデータを送信 */
if( HG_ON == status ){

  /* データを送信する */

}
```

9.7.7.4. RS-232C受信許可信号制御関数

➢ *機能*

RS-232C通信受信許可信号の制御

➢ *関数名*

char hg_ctrl_232RS (char on_off)

➢ *引数*

型	引数	説明
char	on_off	RS-232C通信受信許可信号の出力 HG_ON: 許可 HG_OFF: 禁止

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

引数でHG_ON(1)を指定した場合は受信許可を出力し、HG_OFF(0)を指定した場合は受信禁止を出力します。範囲外の値が指定された場合、異常終了します。

RS-232Cの機種でのみ有効です。

➢ *使用例*

受信許可を出力する場合

```
char ret, status ;
/* RS-232C受信許可  */
status = HG_ON ;
/* RS-232C受信許可信号の状態を出力  */
ret = hg_ctrl_232RS ( status ) ;
```

9.7.8. ハードウェア/ソフトウェア情報読み出し

9.7.8.1. ハードウェア情報読み出し関数

➢ *機能*

ハードウェア情報の読み出し

➢ *関数名*

void hg_get_targetinfo(HG_TARGET_INFO *target_info)

➢ *引数*

型	引数	説明
HG_TARGET_INFO *	target_info	ハードウェア情報の先頭アドレス

*)

ビット	内容	設定値
0-1	通信方式	00:RS232C 01:RS-485 10:RS-422
2	バックライト	0:なし 1:あり
4	メモリ	0:なし 1:あり
6	電源	0:DC24V 1:DC5V
8-11	ハードウェアバージョン	0-
12-15	予約	0
16-31	APIバージョン	0-

➢ *戻り値*

なし

➢ *仕様*

ハードウェアのバージョンや通信方式などの情報を読み出します。

バージョン以外の情報はターゲットボードでは無効です。

➢ *使用例*

通信方式を判別して処理を行う場合

```
HG_TARGET_INFO target_info ;
/*  ハードウェア情報を取得  */
hg_get_targetinfo( &target_info ) ;
/* RS232Cなら  */
if( 0 == target_info.BIT.COM ){

   /* RS232C通信処理  */

}
/* RS485なら  */
else if( 1 == target_info.BIT.COM ){

   /* RS485通信処理  */

}
/* RS422なら  */
else{

   /* RS422通信処理  */

}
```

9.7.8.2. ソフトウェア情報読み出し関数

➢ *機能*

ソフトウェア情報の読み出し

➢ *関数名*

void hg_get_systeminfo(HG_SYSTEM_INFO *sys_info)

➢ *引数*

型	引数	説明
HG_SYSTEM_INFO *	sys_info	ソフトウェア情報の先頭アドレス

➢ *戻り値*

なし

➢ *仕様*

ソフトウェアの情報を読み出します。ソフトウェア情報の詳細は「2.2 型定義」を参照してください。
ソフトウェアの情報を読み出すためには、ソフトウェア開発時にオブジェクト作成の手順に従ってオブジェクトを作成する必要があります。詳細は開発環境資料を参照してください。
デバッグ時には無効となります。ただし、弊社付属ツールを使用してデバッグ用のヘッダー情報を作成し、ダウンロードした場合、バージョン情報とコメントを参照することができます。

➢ *使用例*

システムソフトのバージョンにより処理を変更する場合

```
HG_SYSTEM_INFO system_info ;
/* ソフトウェア情報を取得 */
hg_get_systeminfo( &sys_info ) ;
/* バージョン1.00なら */
if((0x01 == sys_info.version[0] ) && (0x00 == sys_info.version[1] )){

   /* 処理 */

}
/* バージョン1.00以外なら */
else{

   /* 処理 */

}
```

9.7.9. パラメータ格納

V850ES/KG2のEEPROMエミュレーション機能を用いて、内蔵ROM領域の固定エリアにパラメータを格納することができます。

格納領域としては内蔵ROM領域の最終ブロック(ブロック63)から指定したブロック数分の領域(1ブロック=2Kバイト)が使用され、内蔵RAMの予約領域は3FFB000h-3FFB007h, 3FFC000h-3FFC7FFhとなりますので、お客様作成ソフトウェアと重複しないようご注意願います。

電源投入後に一度だけ内蔵ROM書き込み初期化関数を実行すれば、内蔵ROM書き込み関数や内蔵ROM読み出し関数は任意のタイミングで実行することができます。ただし、内蔵ROM書き込み初期化関数で異なるパラメータをセットして実行すると、以後これらの関数がすべてエラーになり動作しなくなる可能性がありますので、ご注意ください。万が一動作不能になった場合、ダウンロードツールにてユーザ使用可能内蔵ROM領域をすべて消去してください。消去はダウンロードツールにてclear.binを0008000hにダウンロードすることにより簡単に行えます。

内蔵ROMアドレス	ブロック番号
001FFFFh 001F800h	ブロック63(2Kバイト)
001F7FFh 001F000h	ブロック62(2Kバイト)
001EFFFh 001E800h	ブロック61(2Kバイト)
001E7FFh 001E000h	ブロック60(2Kバイト)
001DFFFh 001D800h	ブロック59(2Kバイト)
001D7FFh 001D000h	ブロック58(2Kバイト)
001CFFFh 001C800h	ブロック57(2Kバイト)
・ ・ ・	・ ・ ・
0010FFFh 0010800h	ブロック33(2Kバイト)
00107FFh 0010000h	ブロック32(2Kバイト)

9.7.9.1. 内蔵ROM書き込み初期化関数

➢ *機能*

EEPROMエミュレーション機能の初期化

➢ *関数名*

char hg_init_internalROM(char block, char num)

➢ *引数*

型	引数	説明
char	block	パラメータ格納領域として使用するブロック数(1-32)
char	num	書き込み数(ワード単位) (1-64)

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

パラメータ格納領域として使用するブロック数と内蔵ROMに書き込むデータの大きさを指定します。書き込みはワード単位で行います。また、1ブロックは2Kバイトです。

引数に範囲外の数値が指定された場合、異常終了となります。

初期化中はフラッシュ・プログラミング・モードとなります。

パラメータ格納領域のブロック数が同じであれば、ワード数が大きいと書き込み可能回数は少なくなります。

【例】2Kバイトの格納領域に、1ワード(4バイト)のデータを書き込むとした場合、見かけ上の書き込み可能回数は以下のようになります。

2Kバイト÷(データ(1ワード)＋デミリタ(1 ワード)×100回) = 25600回

EEPROMエミュレーションを使用しているため、インサーキット・エミュレータおよびオンチップ・デバッグ・エミュレータを使用したデバッグはできません。

➢ *使用例*

2ブロック格納領域を確保し、1ワード単位でデータを格納する場合

```
char ret ;
/*  パラメータ格納領域の初期化  */
ret = hg_init_internalROM( 2, 1 ) ;
```

9.7.9.2. 内蔵ROM書き込み関数

➢ *機能*

内蔵ROMにパラメータ値を書き込む

➢ *関数名*

char hg_write_internalROM(char *data)

➢ *引数*

型	引数	説明
char *	data	書き込みパラメータ値の格納先頭アドレス

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

設定値などのパラメータ値を内蔵ROMに書き込みます。

初期化関数で指定したワード数分だけ書き込みます。

書き込み中はフラッシュ・プログラミング・モードとなります。

書き込みデータは内蔵ROM以外の領域においてください。

関数内で割り込み禁止(__DI();)・許可(__EI();)の制御を行っていますので、関数実行中はすべての割り込みが禁止されます。処理時間は数100msec以上かかることがありますので、ウォッチドッグ等設定されている場合は、禁止にするか十分長い時間を設定してください。

EEPROMエミュレーションを使用しているため、インサーキット・エミュレータおよびオンチップ・デバッグ・エミュレータを使用したデバッグはできません。

EEPROMエミュレーション動作のため、スタック領域を736バイト消費しますのでご注意ください。

➢ *使用例*

1ワード単位でデータを格納する場合

```
char ret ;
int src ;
src = 0x1234 ;
/*  パラメータを格納  */
ret = hg_write_internalROM( (char *)&src ) ;
```

9.7.9.3. 内蔵ROM読み出し関数

➢ *機能*

内蔵ROMのパラメータ値を読み出し

➢ *関数名*

char hg_read_internalROM(char *data)

➢ *引数*

型	引数	説明
char *	data	読み出しパラメータ値の格納の先頭アドレス

➢ *戻り値*

戻り値	説明
HG_NO_ER	正常終了
HG_ER	異常終了

➢ *仕様*

設定値などのパラメータ値を内蔵ROMから読み出します。
初期化関数で指定したワード数分だけ読み出します。
読み出し中はフラッシュ・プログラミング・モードとなります。
データ格納の領域は内蔵ROM以外の領域においてください。
関数内で割り込み禁止(__DI();)・許可(__EI();)の制御を行っていますので、関数実行中はすべての割り込みが禁止されます。処理時間は数100msec以上かかることがありますので、ウォッチドッグ等設定されている場合は、禁止にするか十分長い時間を設定してください。
EEPROMエミュレーションを使用しているため、インサーキット・エミュレータおよびオンチップ・デバッグ・エミュレータを使用したデバッグはできません。
EEPROMエミュレーション動作のため、スタック領域を736バイト消費しますのでご注意ください。

➢ *使用例*

1ワード単位でデータを読み出す場合

```
char ret ;
int dst ;
/*  パラメータを読み出し  */
ret = hg_read_internalROM( (char *)&dst ) ;
```