

【H-debugger 対応】
TOPPERS/OSEK カーネル
アプリケーションノート

SH-2シリーズ (BOOT-UBC)
SH7051F KPIT-GNU [Hew] 版

2008/10/02

Rev1.50 (2008/10/02)

TOPPERS/OSEK Kernel

Toyohashi Open Platform for Embedded Real-Time Systems/
OSEK Kernel

Copyright (C) 2000-2003 by Embedded and Real-Time Systems Laboratory

Toyohashi Univ. of Technology, JAPAN

Copyright (C) 2004 by Embedded and Real-Time Systems Laboratory

Graduate School of Information Science, Nagoya Univ., JAPAN

Copyright (C) 2004-2006 by Witz Corporation, JAPAN

Copyright (C) 2008 by A-one Corporation, JAPAN

上記著作権者は、以下の (1)~(4) の条件か、**Free Software Foundation** によって公表されている **GNU General Public License** の **Version 2** に記述されている条件を満たす場合に限り、本ソフトウェア（本ソフトウェアを改変したものを含む。以下同じ）を使用・複製・改変・再配布（以下、利用と呼ぶ）することを無償で許諾する。

- (1) 本ソフトウェアをソースコードの形で利用する場合には、上記の著作権表示、この利用条件および下記の無保証規定が、そのままの形でソースコード中に含まれていること。
- (2) 本ソフトウェアを、ライブラリ形式など、他のソフトウェア開発に使用できる形で再配布する場合には、再配布に伴うドキュメント（利用者マニュアルなど）に、上記の著作権表示、この利用条件および下記の無保証規定を掲載すること。
- (3) 本ソフトウェアを、機器に組み込むなど、他のソフトウェア開発に使用できない形で再配布する場合には、次のいずれかの条件を満たすこと。
 - (a) 再配布に伴うドキュメント（利用者マニュアルなど）に、上記の著作権表示、この利用条件および下記の無保証規定を掲載すること。
 - (b) 再配布の形態を、別に定める方法によって、**TOPPERS** プロジェクトに報告すること。
- (4) 本ソフトウェアの利用により直接的または間接的に生じるいかなる損害からも、上記著作権者および **TOPPERS** プロジェクトを免責すること。

本ソフトウェアは、無保証で提供されているものである。上記著作権者および **TOPPERS** プロジェクトは、本ソフトウェアに関して、その適用可能性も含めて、いかなる保証も行わない。また、本ソフトウェアの利用により直接的または間接的に生じたいかなる損害に関しても、その責任を負わない。

目 次

第1章	概要.....	5
1-1.	はじめに.....	5
1-2.	関連文書.....	5
1-3.	開発環境.....	5
1-4.	製品梱包内容.....	5
1-5.	サポート.....	5
第2章	TOPPERS/OSEK カーネルの開発階層.....	6
第3章	サンプルアプリケーションの概要(SH7051).....	7
3-1.	サンプルアプリケーションの構成.....	7
3-2.	動作フロー (ゼネラル).....	8
3-3.	H-debugger 対応に必要なポーティング.....	13
3-4.	H-debugger でのプロファイル表示(DEF 7.00B 以上).....	15
第4章	ハード構成およびシステム構成.....	16
4-1.	ハード構成.....	16
4-2.	システム構成.....	18
4-2-1.	SH7051 のプログラムメモリ MAP.....	18
4-2-2.	プログラムサイズの詳細 MAP.....	18
4-2-3.	CPU 基板(CAT68201)の I/O マップ表.....	19
第5章	サンプルアプリケーションの準備と動作確認.....	21
5-1.	Hew での準備 (ルネサス製).....	21
5-2.	H-debugger(DEF.exe)での準備 (Aone 製).....	26
5-3.	サンプルアプリケーションを走らせる前の準備.....	30
5-4.	サンプルアプリケーションを走らせません。.....	31
第6章	新規プロジェクトを追加する場合の手順例.....	33

6-1. プロジェクトタイプの作成.....	33
6-2. 新規プロジェクトを登録します。	35
第7章 備考.....	43
7-1. おわりに.....	43

第1章 概要

1-1. はじめに

本アプリケーションノートは、TOPPERS/OSEK カーネルをもとに「H-debugger」と「SH7051」用にポータリングしたサンプルソフトです。

サンプルソフトの他アプリケーションへの利用/変更に関しての制限は一切ありませんので自由にお使い下さい。ただし、このサンプルソフトの不具合により発生した損害に対しての責任、及び、修正の義務は負いません。また、このサンプルソフトに関する質問の回答義務も負えませんが、メールでのお問い合わせに関しては、弊社責任の範囲内でしたら出来るだけ御答えるように努めます。Mail: cat-i@aone.co.jp

1-2. 関連文書

本アプリケーションは、下記ドキュメントを参考にして作成しました。

- 1) TOPPERS/OSEK カーネル外部仕様書 ----- 株式会社ヴィッツ製
- 2) TOPPERS/OSEK カーネル SG 取扱説明書 ----- 株式会社ヴィッツ製
- 3) TOPPERS/OSEK カーネルアプリケーションノート ----- 株式会社ヴィッツ製
- 4) OSEK/VDX Operating System Ver2.2.1 ----- OSEK/VDX 仕様
- 5) OSEK/VDX Binding Specification Ver1.4.2 ----- OSEK/VDX 仕様
- 6) OSEK/VDX OIL Specification Ver2.5 ----- OSEK/VDX 仕様

*OSEK/VDX が公開している仕様書は、<http://www.osek-vdx.org/> よりダウンロードにより入手して下さい。

1-3. 開発環境

- 1) Hew Version 4.04.01.001 ----- Renesas 製
 - 2) KPIT GNUSH[ELF] Toolchain v0801 ----- KPIT 製
- にて作成しましたので各自用意して下さい。

1-4. 製品梱包内容

- 1) サンプルソフト用 CD (本書 PDF ファイルも含む) ----- 1 枚
 ・本製品の価格体系は、CD 配布の実費のみになっております。

1-5. サポート

TOPPERS/OSEK に関するサポートが必要な場合は、TOPPERS ホームページの「関連製品」「サポート・サービス」の項をご覧になり、御利用下さい。

<http://www.toppers.jp/> <--- TOPPERS ホームページ

第 2 章 TOPPERS/OSEK カーネルの開発階層

¥toppers_osek	
¥config	// 機種依存階層
¥sh2-kpitgnu-7051	// 開発環境分類
+CPU 依存部	
¥SH7051	// システム依存部
¥include	// インクルードファイル階層
¥kernel	// カーネル共通部階層
¥sample	// サンプル階層(未使用)
¥sg	// システムジェネレータ階層
¥impl_oil	// OIL 記述の実装定義部階層
¥syslib	// システムライブラリ階層
¥sh2-kpitgnu-7051	// 開発環境分類
+CPU 依存部	
¥SH7051	// システム依存部
¥tools	// Hew 管理階層
¥sh2-kpitgnu-7051	// 開発環境分類
+ワークスペース	
¥SH7051	// プロジェクト 1
¥appsrc	// サンプルソース [* .c]
¥debug	// オブジェクト [* .x]
¥Project	// 新規プロジェクト作成用
	// テンプレート

本サンプルソフトは、1本のプロジェクトを用意しています。

1) プロジェクト 1 ¥SH7051 実行オブジェクト **【SH7051.x/SH7051.mot】**

 本アプリケーションノートのサンプルプロジェクト

青字部分は、TOPPERS/OSEK 正式リリース（オープンソース）になります。その他は弊社で改造および作成をしました。

第3章 サンプルアプリケーションの概要(SH7051)

3-1. サンプルアプリケーションの構成

サンプルアプリケーションは、下記の構成にて作成しました。

1) 5個のタスク

- ① **MainTask** ----- 各 **Task** の起動処理
- ② **Task1** ----- センサー入力(8点)の状態表示
- ③ **Task2** ----- PWM1/2/3 の3点パルス出力
- ④ **Task3** ----- RS232C による PWM デューティ比変更
- ⑤ **HighPriorityTask** ----- 各タスクのスタックオーバー監視

2) 4個の割込み処理

- ① システムタイマー **CMT0** 1ms 割込み **SysTimerInt()**
- ② **SCI0** 受信エラー **ERI0** **ErrHwSerialInt()**
- ③ **SCI0** 受信 **RXI0** **RxHwSerialInt()**
- ④ **SCI0** 送信終了 **TXI0** **TxHwSerialInt()**

3) 5個のイベント

MainEvt/T1Evt/T2Evt/T3Evt/TEvt

4) 2個のアラーム

MainCycArm/SetEvtHArm

5) 2個のコールバック処理

CallBackArm(TimerCallBack コールバック利用のソフトタイマ)

6) 1個のウォッチドッグタイマー

WatchDogClear

7) スタートアップフックルーチン

ErrorHook

8) シャットダウンフックルーチン

ShutdownHook

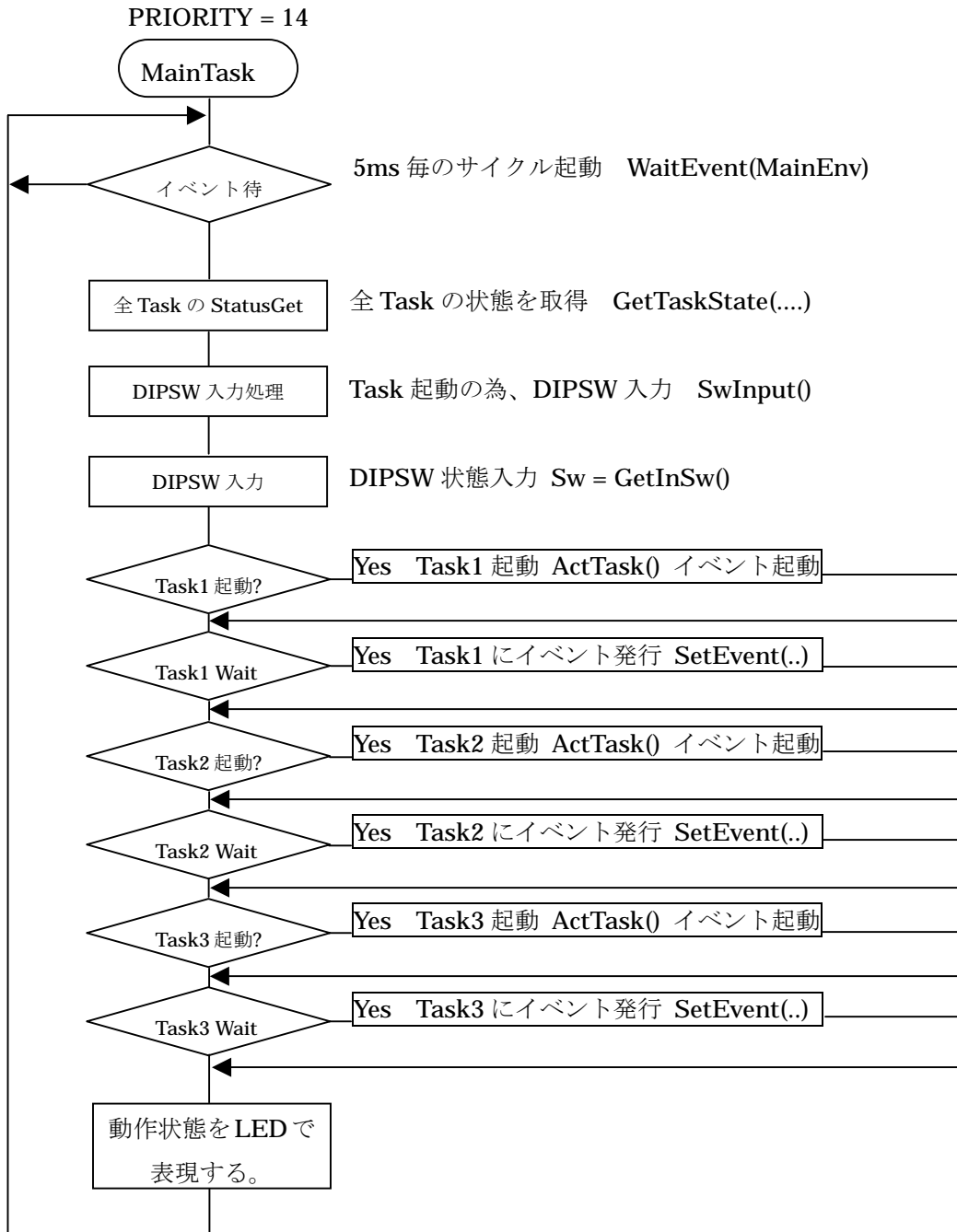
9) プレタスク/エラーフックルーチン

PreTaskHook/ErrorHook

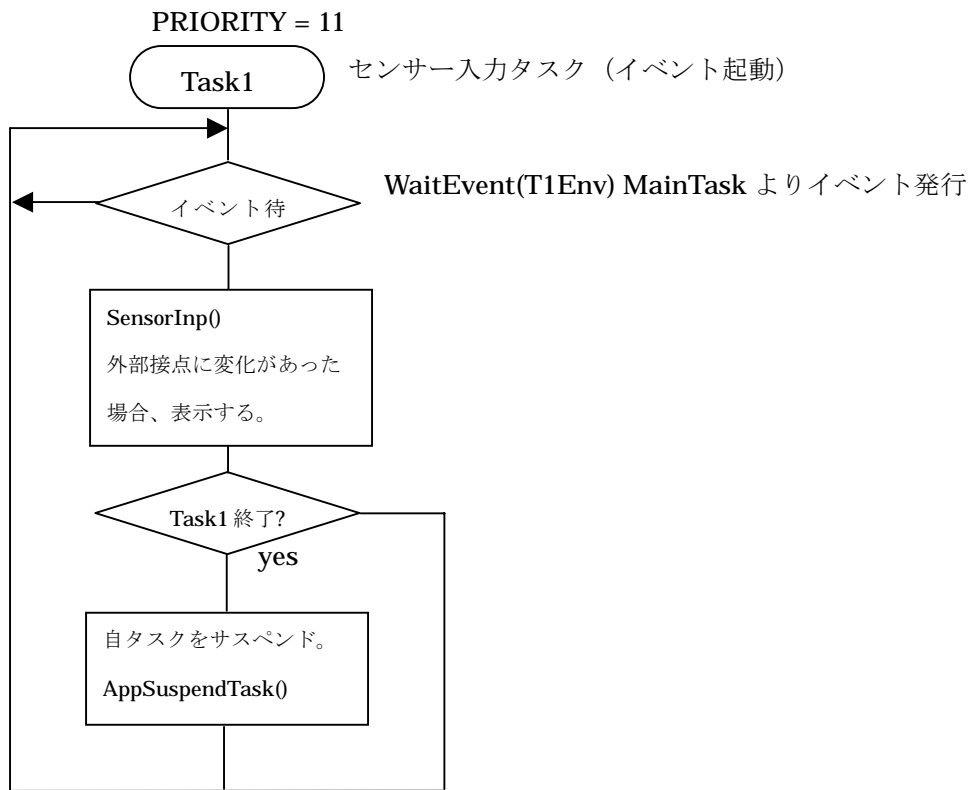
10) 1個のアプリケーションモード

AppMode1

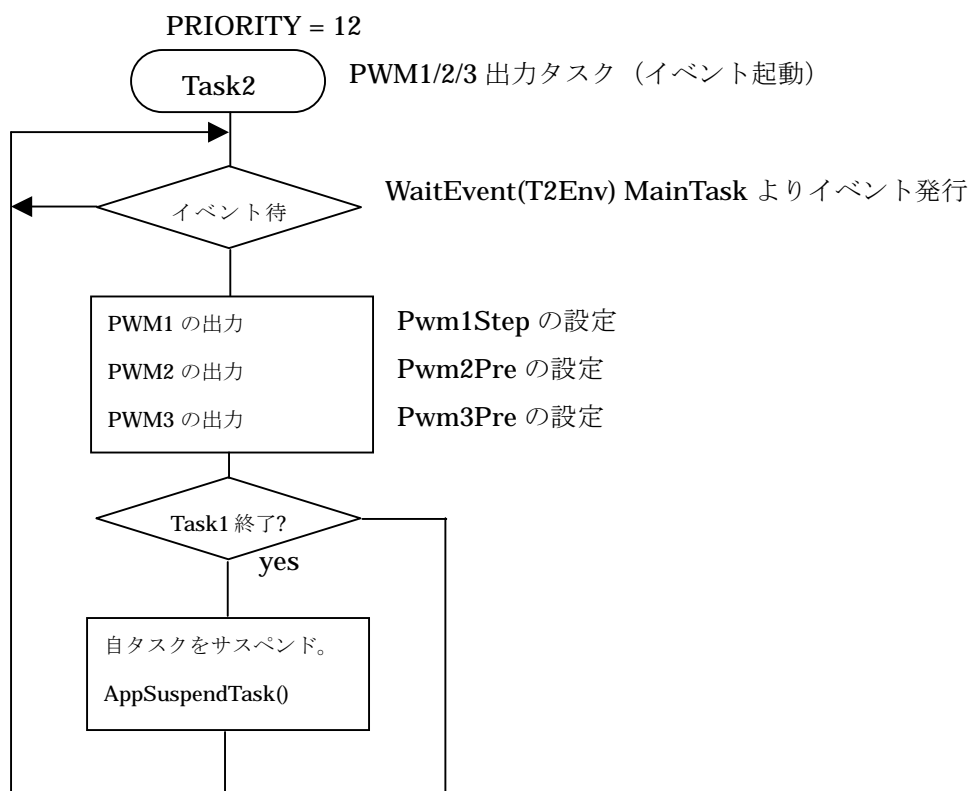
3-2. 動作フロー (ゼネラル)



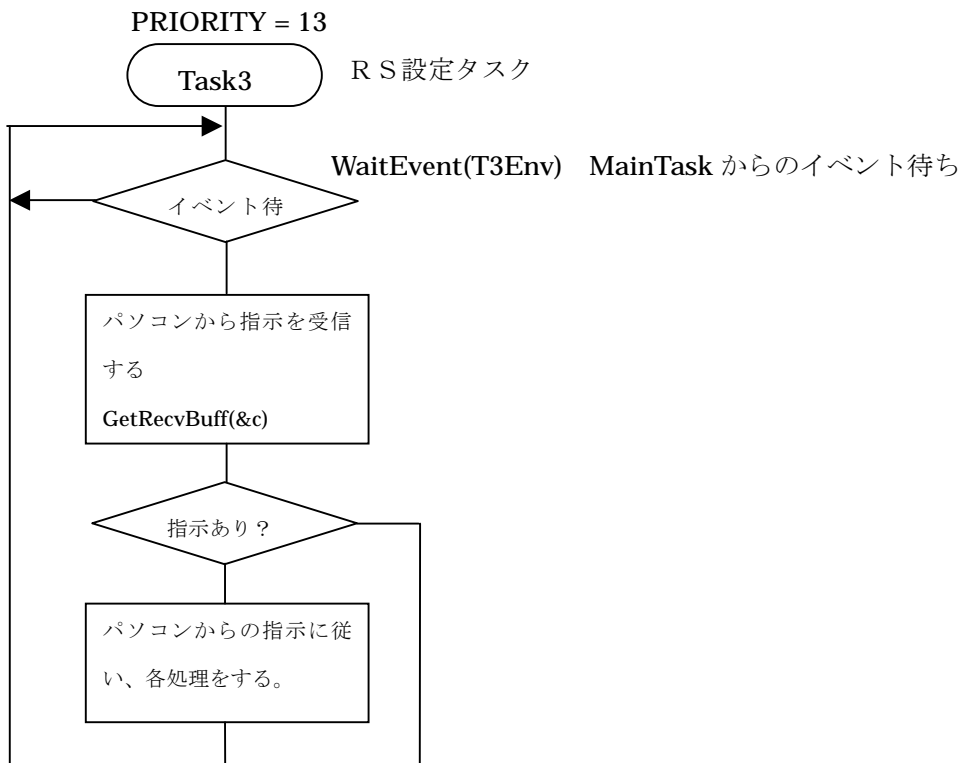
- ①Task1(センサー入力)の起動条件 --- DIPSW(1)ON で起動する。
- ②Task2(PWM出力)の起動条件 --- PushSW が ON
- ③Task3(R S 設定)の起動条件 ---- パソコンよりスペース入力



①Task1 の終了条件 ー ー DIPSW(1)でなくなった時、終了とする。

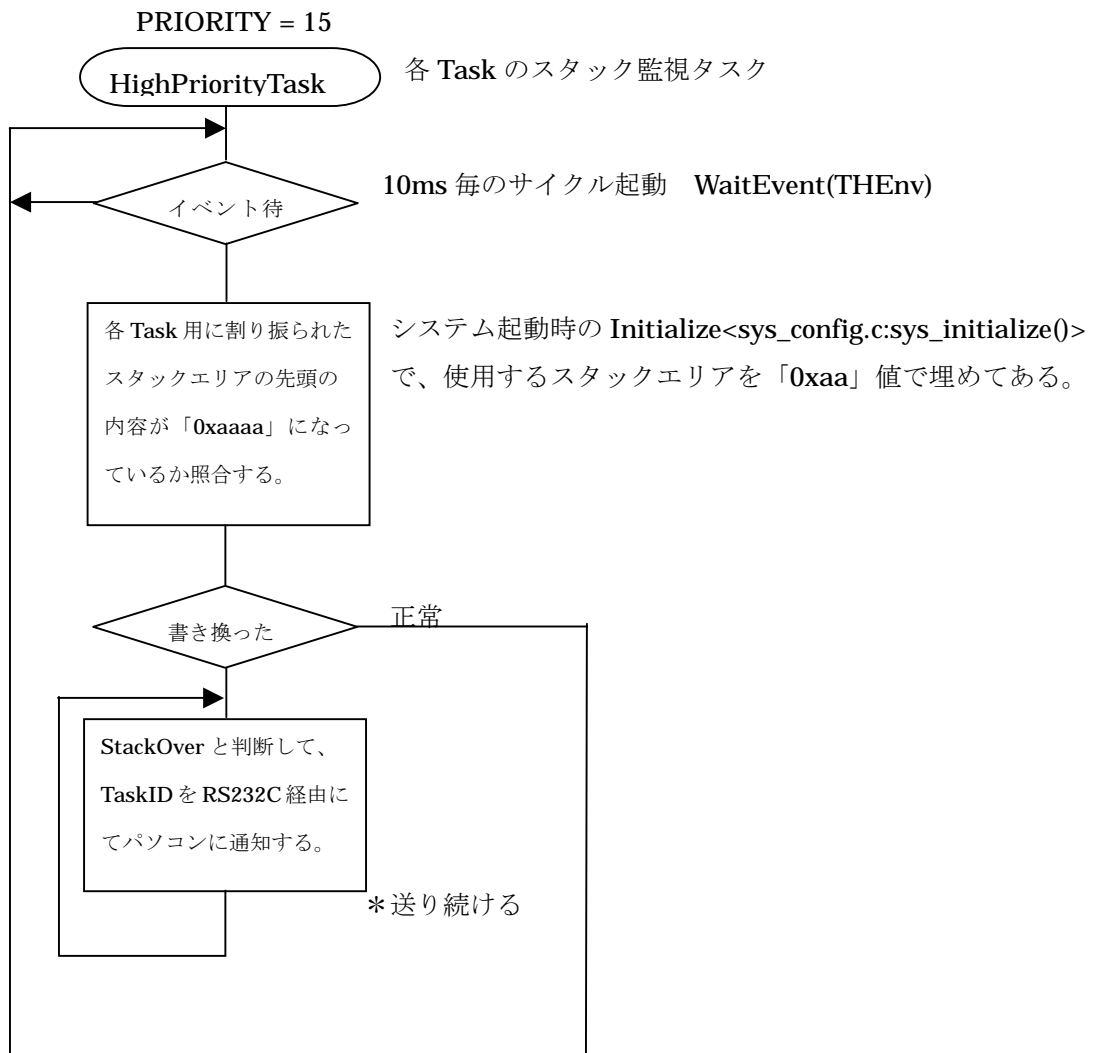


①Task2 の終了条件 ー PushSW の ON で終了とする。



<パソコンからの指示コード>

- | | | | | |
|---|-------|-------|--------------|------------|
| ① | 'Q'キー | ----- | PWM1 センタ+1 | (-50~ 50) |
| ② | 'W'キー | ----- | PWM1 センタ-1 | (-50~ 50) |
| ③ | 'A'キー | ----- | PWM2 パーセント+1 | (-100~100) |
| ④ | 'S'キー | ----- | PWM2 パーセント-1 | (-100~100) |
| ⑤ | 'Z'キー | ----- | PWM3 パーセント+1 | (-100~100) |
| ⑥ | 'X'キー | ----- | PWM3 パーセント-1 | (-100~100) |
| ⑦ | Space | ----- | メニュー再表示 | |
| ⑧ | 'E'キー | ----- | Task3 の終了指示 | |



3-3. H-debugger 対応に必要なポーティング

SH7051 は、BOOT ポート(SCI1)使用による UBC(ユーザブレイクコントローラ) を利用したデバッグモードになります。

UBC を利用するにおいて、UBC の割込みプライオリティを 15 にして、他の割込みプライオリティを 14 以下にする必要があります。

1) ポーティング手続き 1

```

.¥config¥sh2-kpitgnu-7051¥SH7051¥sys_support.S

.EQU          SR_Init,   0x000000E0

hardware_init_hook:
    sts.l     pr,@r15
    /* SR の初期化 */
    mov.l     _SR_Init,r0
    ldc      r0,sr
    /* H-debugger 用 リセット遅延の為、20msWait */

wait20ms:
    mov      #20,r0

waitrst:
    bsr      _SOFT1MS
    nop
    dt      r0
    bf      waitrst
    lds.l    @r15+,pr
    rts
    nop

/* SOFT1MS()      1ms ソフトタイマー Non Wait */
_SOFT1MS:
    mov.l    r0,@-sp
    mov.l    _T1ms,r0
wait1ms: dt    r0                ; 1 CLK
    bf      wait1ms            ; 3 CLK
    mov.l    @sp+,r0
    rts
    nop
    
```

<理由>

CAT68201(SH7051)の場合、**H-debugger** からのリセット信号が **RST-IC** 経由で CPU につながっている為、**RESET** 信号の立ち上がりに遅延が発生します。

BOOT-UBC 仕様でのデバッガ処理は、**RESET** 立ち上げ後、**20ms** 毎に最大 **20** 回まで **NMI** が認識するまでターゲットに対して割り込を要求します。

その **20ms** 分 CPU の走行を停止させる為、上記プログラムのようにソフトタイマーを入れます。

なお、「**DEF-CPU** 設定にて **200ms** 遅延回路を使用する」に設定した場合は、**RESET** 立ち上げしてから、**200ms** 経過後 **NMI** 要求を開始します。

3-4. H-debugger でのプロファイル表示(DEF 7.00B 以上)

ターゲット側の RAM を使用して、各 Task のプロファイル表示をします。

1) ポーティング手続き

```
.¥syslib¥sh2-kpitgnu-7051¥sys_timer.h

#define USE_PROFILE
#ifndef USE_PROFILE
    #define PROFILESIZE          512
    #define PROFILEPRETASK      1
#endif
```

① 「USE_PROFILE」を有効にすると、プロファイル表示が可能になります。不要になった場合は、コメントアウトして下さい。

② 「PROFILESIZE」は、ターゲット側で確保する RAM のバイト数になります。アプリケーションに応じて調整して下さい。

③ 「USE_PROFILE」を有効にする事により、下記変数が確保されます。

```
UINT8      _TaskProFile[PROFILESIZE];
UINT8      TaskProFlg;
UINT16     TaskProFileIdx;
```

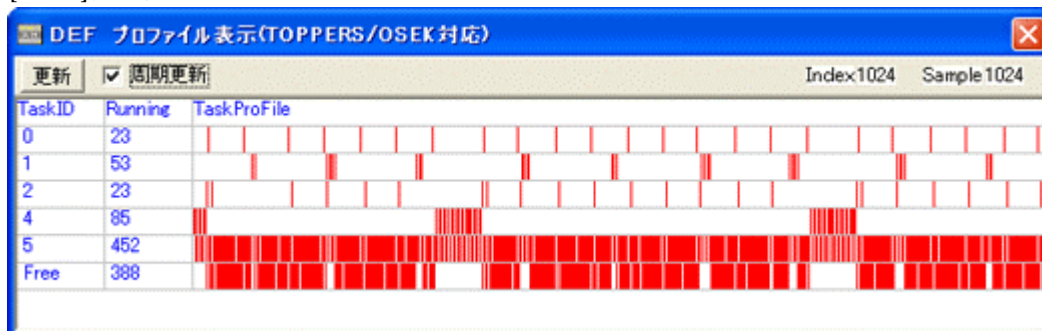
④ システムタイマー割り込みのタイミングで現 Running 中の TaskID を「_TaskProFile」に順次記憶していきます。(リングバッファ)

⑤ サンプルング場所は、「プレタスクフックルーチン」と「ISR(SysTimerInt)」の 2 箇所です。

⑥ プロファイル表示させたい場合は、

DEF メニュー<データ>-<プロファイル表示>をクリックして下さい。

[3-4-1]表示例



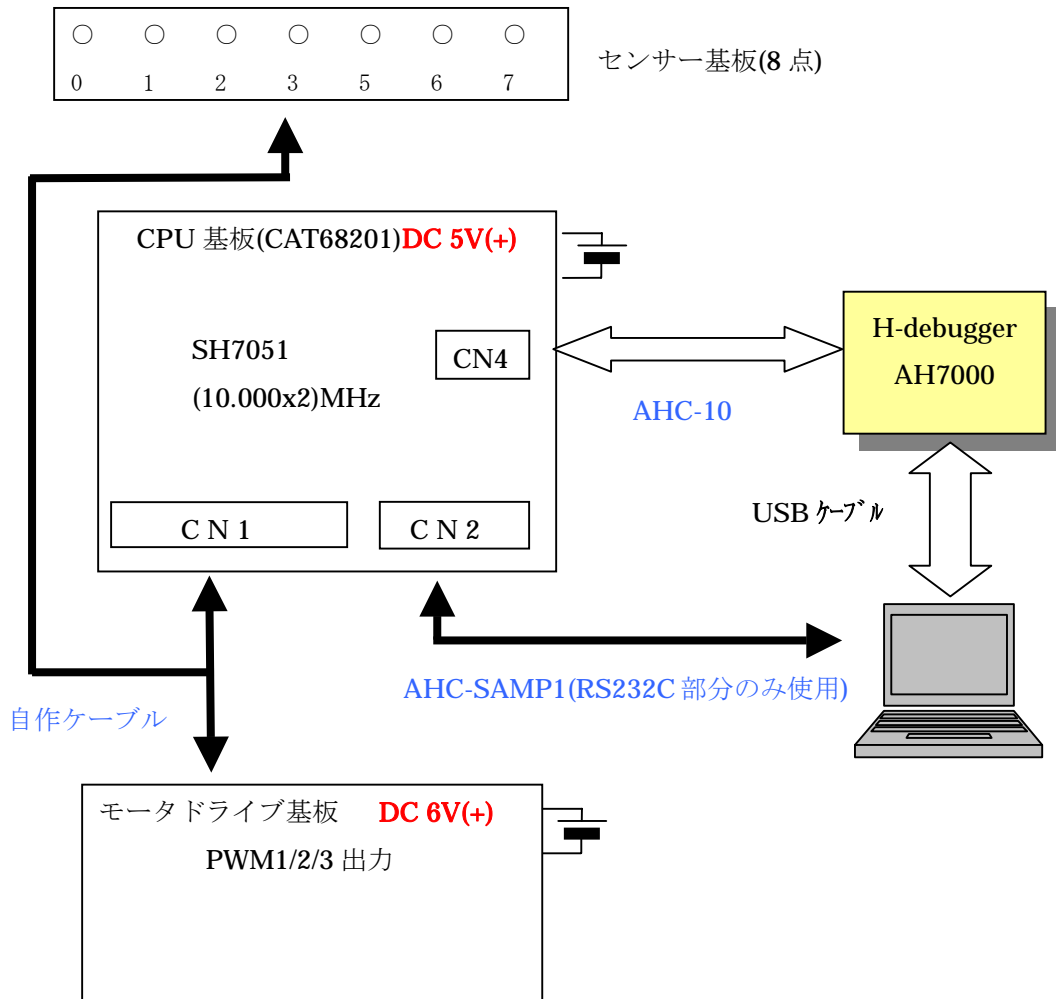
「更新」PB をクリックしますと最新データを表示します。

「周期更新」にチェックしますと、常時最新状態をターゲットからオンザフライ機能により、データを収集し、Task 状態を表示します。

第4章 ハード構成およびシステム構成

4-1. ハード構成

この解説書を進めるにあたり、下記ハード構成の準備をお願いします。

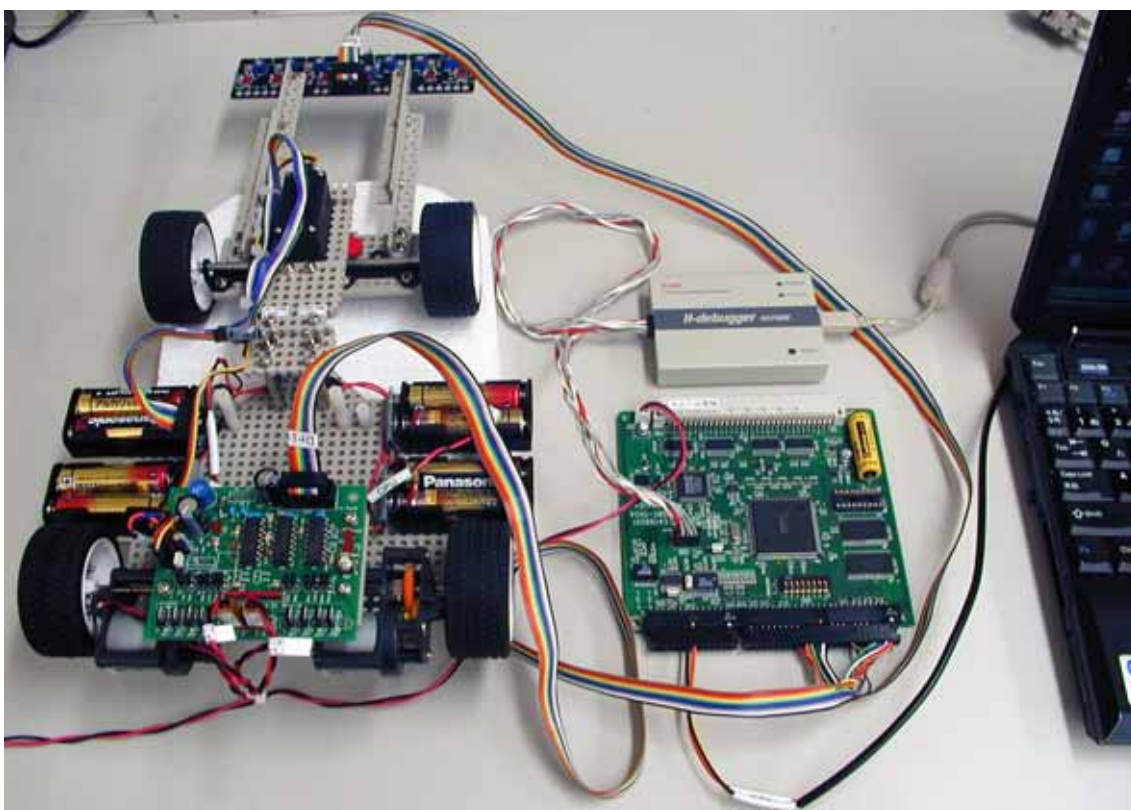


<デバッグに必要なオプションケーブル>

①AHC-10	ターゲット側標準ケーブル	1本
②AHC-SAMP1	RS232C 信号のみ使用	1本
③自作ケーブル	信号を振り分けるケーブル	1本

- 1) CPU 基板(CAT68201)の CN1 とセンサー基板をケーブル接続します。
- 2) CPU 基板(CAT68201)の CN1 とモータドライブ基板をケーブル接続します。
- 3) CPU 基板(CAT68201)の CN4 と AH7000 をケーブル(AHC-10)で接続します。
- 4) AH7000 とパソコンを USB ケーブルで接続します。
- 5) CPU 基板(CAT68201)の CN2 とパソコンをケーブル(AHC-SAMP1(一部))で接続します。
- 6) CPU 基板(CAT68201)に電源 (DC +5V)が供給できるように接続します。
- 7) モータドライブ基板に電源 (DC +6V)が供給できるように接続します。

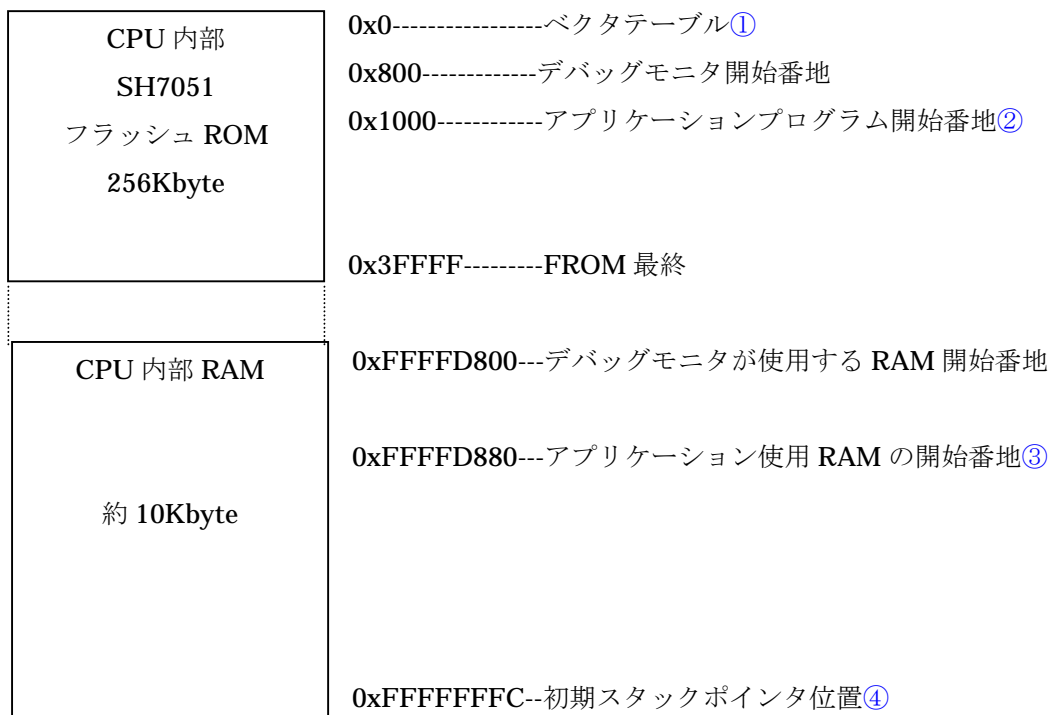
[接続例]



4-2. システム構成

サンプルのシステムブロック図およびメモリマップと I/O 表を記述します。

4-2-1. SH7051 のプログラムメモリ MAP



<セクション名>

- ① ベクタテーブル----- .vects
- ② アプリケーションプログラムの開始番地----- .text,.init,.rodata....etc
- ③ アプリケーション使用 RAM の開始番地----- .data,.bss....etc
- ④ 初期スタックポインタ位置----- .stack

4-2-2. プログラムサイズの詳細 MAP

開始番地	サイズ	分類
0x800	0x800	デバッガモニタ (ファーム)
0x1000	0x11F4	アプリケーション (main....etc)
0x21F4	0x179C	割込みハンドラ、内部 I/O 関係の関数
0x3990	0x3618	TOPPERS/OSEK カーネル
0x6FA8	0x4EC	C ライブラリ、ROM テーブル、etc
0x7494		最終アドレス

4-2-3. CPU基板(CAT68201)のI/Oマップ表

CPU基板のDIP-SW					
	ポートシンボル CPU基板			方向	信号名
PHDR	PH15			入力	DIP-SW8(0=ON)
	PH14			”	DIP-SW7(0=ON)
	PH13			”	DIP-SW6(0=ON)
	PH12			”	DIP-SW5(0=ON)
	PH11				DIP-SW4(0=ON)
	PH10				DIP-SW3(0=ON)
	PH09				DIP-SW2(0=ON)
	PH08				DIP-SW1(0=ON)

センサー関係 センサ基板 J1					
	センサ基板 ピン番号	CPU基板 ポートシンボル	CPU基板 CN1番号	方向	信号名
PEDR	1	VCC	13A	VCC	+5V
	2	PE7	17A	入力	センサ 7
	3	PE6	16B	”	センサ 6
	4	PE5	16A	”	センサ 5
	5	PE4	15B	”	センサ 4(スタートバ)
	6	PE3	15A	”	センサ 3
	7	PE2	14B	”	センサ 2
	8	PE1	14A	”	センサ 1
	9	PE0	13B	”	センサ 0
	10	GND	12B	GND	GND

モータドライブ関係 ドライブ (DRV)基板 J3					
	DRV基板 ピン番号	ポートシンボル CPU基板	CPU基板 CN1番号	方向	信号名
PBDR	1	VCC	8A	VCC	+5V
	2	PB11	1A	出力	LED1 0=点灯
	3	PB10	1B	”	LED0 0=点灯
	4	TO6	4B	”	PWM1 (サーボ)
	5	TO7	4A	”	PWM2 (右モータ)
	6	PB3	3A	”	右モータ方向 0=正
	7	PB4	2B	”	左モータ方向 0=正
	8	TO8	3B	”	PWM3 (左モータ)
	9	PB5	2A	入力	PushSW 0=ON
	10	GND	7B	GND	GND

RS232C 関係 CPU 基板 CN2 適用ケーブル AHC-SAMP1(一部使用)					
		CPU 基板 ポートシボ ^ル	CPU 基板 CN2 番号	方向	信号名
		PG3	7B	入力	SCI0-RXD
		PG2	7A	出力	SCI0-TXD
		GND	8A	GND	GND

CPU 基板(CAT68201)の JP2 1-2 ショート(RS232C レベルで使用)

第5章 サンプルアプリケーションの準備と動作確認

5-1. Hew での準備 (ルネサス製)

1) TOPPERS のホームページより、OSEK カーネル最新リリースをダウンロードして下さい。

URL:<http://www.toppers.jp/osek-os.html>

2) ダウンロードした「osek_os-x.x.lzh」を、適当なディレクトリに置き解凍して下さい。

3) サンプルアプリケーション用 CD の「.¥toppers_osek_Kpit_7051」の指定ファイルを、OSEK カーネルにコピーします。

<CD 側>

<DL した OSEK 側>

.¥toppers_osek_Kpit_7051¥config¥sh2-kpitgnu-7051 --> .¥toppers_osek¥config

.¥toppers_osek_Kpit_7051¥syslib¥sh2-kpitgnu-7051 --> .¥toppers_osek¥syslib

.¥toppers_osek_Kpit_7051¥tools¥sh2-kpitgnu-7051 --> .¥toppers_osek¥tools

上記ディレクトリ下の全ファイルを、OSEK カーネルにディレクトリごと全コピーして下さい。

4) Hew を起動します。

- Hew Version 4.04.01.001 -----Renesas 製
- KPIT GNUSH[ELF] Toolchain v0801 -----KPIT 製

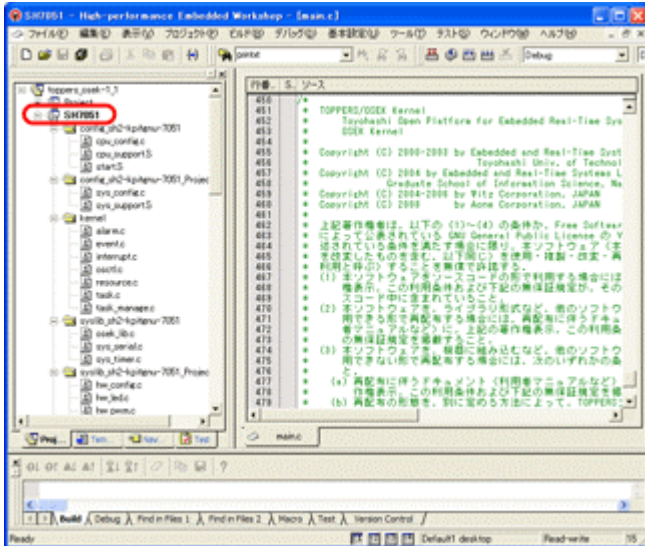
5) Hew メニューの<ファイル>-<ワークスペースを開く>でワークスペースを開きます。

- ".¥toppers_osek¥tools¥sh2-kpitgnu-7051¥sh2_kpitgnu_7051.hws"を指定します。
- ディレクトリ情報が変わりますので、下記ウォーニングが表示されますが、気にせず「はい」を指定して下さい。



[5-1-1]

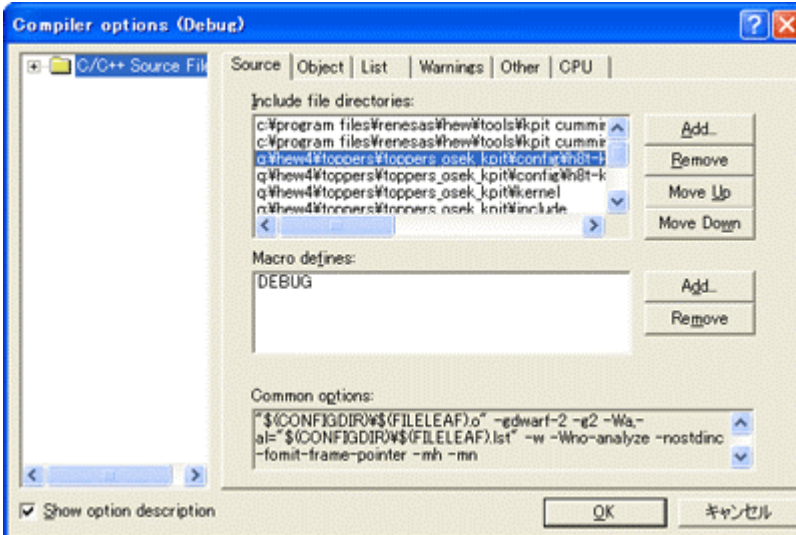
- 6) アクティブプロジェクトが「SH7051」になっていることを確認します。SH7051 になっていない場合は、Hew メニューの<プロジェクト>-<アクティブプロジェクトに設定>で「SH7051」を指定して下さい。



[5-1-2]

- 7) 現 KPIT 版では、ディレクトリの相対指定[.. ¥]が出来ない為、インクルードディレクトリの変更が必要です。

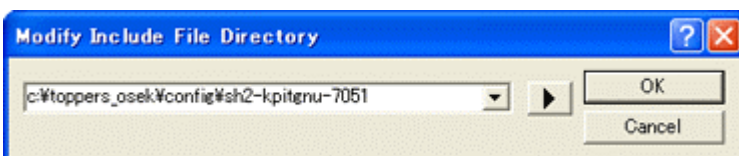
Hew メニューの<ビルド>-<Compiler>をクリックして下さい。



[5-1-2-1]

全てのディレクトリを現ワークスペースのディレクトリに変更して下さい。

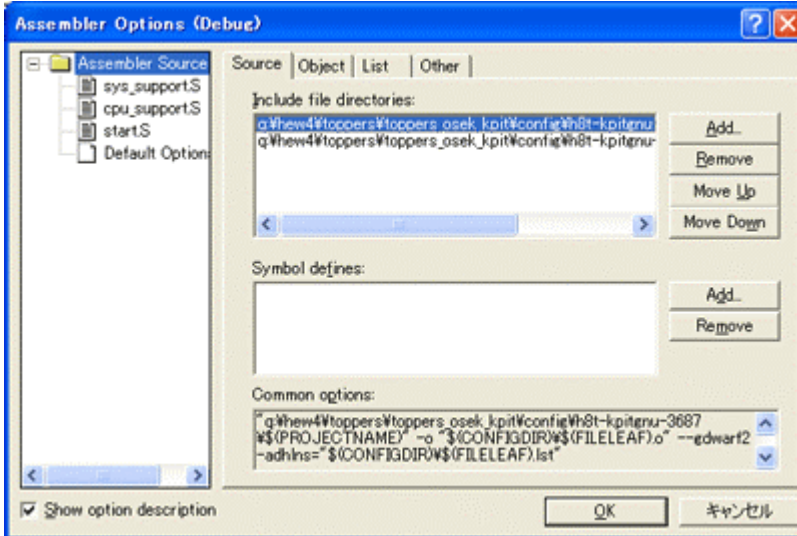
マウスでのダブルクリックで変更できます。



[5-1-2-2]

<Brows>指定が早いかと思います。

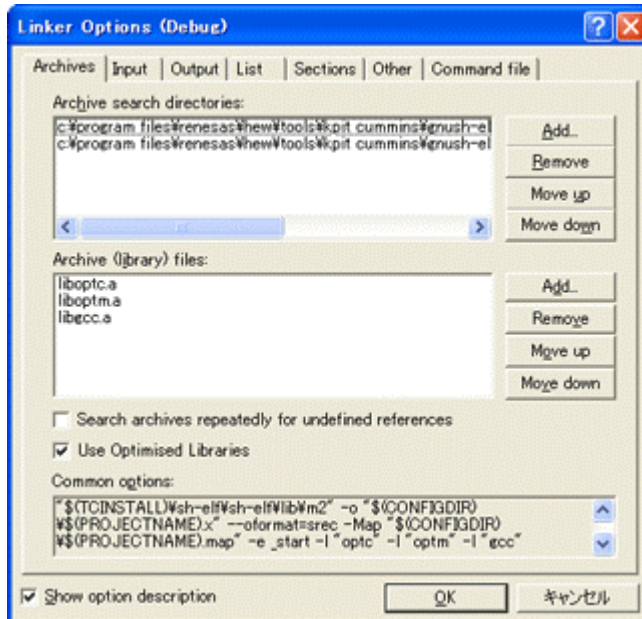
Hew メニューの<ビルド>-<Assembler>をクリックして下さい。



[5-1-2-3]

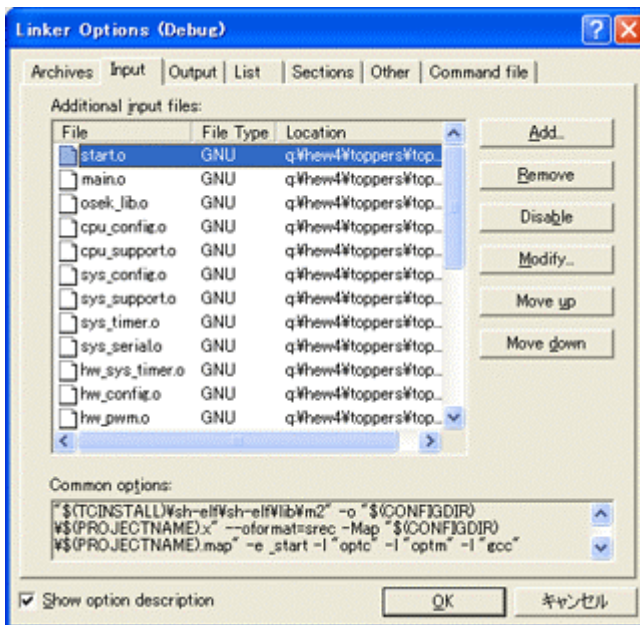
全てのディレクトリを現ワークスペースのディレクトリに変更して下さい。

Hew メニューの<ビルド>-<Linker>をクリックして下さい。



[5-1-2-4]

<Input>タグをクリックします。

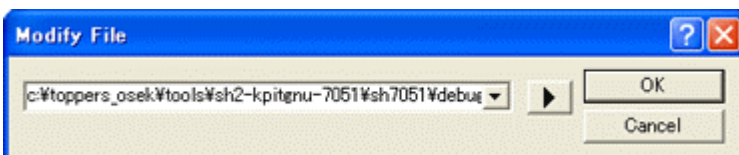


[5-1-2-5]

全てのディレクトリを現ワークスペースのディレクトリと相違があった場合は現ディレクトリに変更して下さい。

設定は、「\$(CONFIGDIR)*.o」にしましたので変更する必要は無いかと思えます。

マウスでのダブルクリックで変更できます。



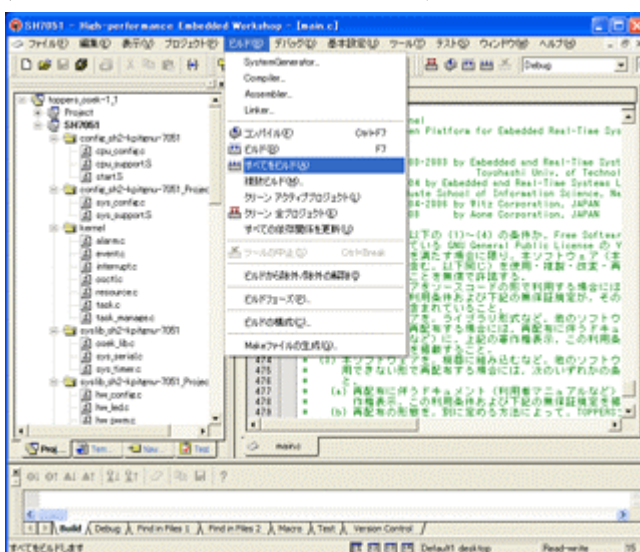
[5-1-2-6]

⚠️ 新規プロジェクトの作成時に必要です。

8) アクティブプロジェクトが「Project」に変更して、7) 項と同じく<Compiler>と<Assembler>と<Linker>のディレクトリを変更と確認をして下さい。

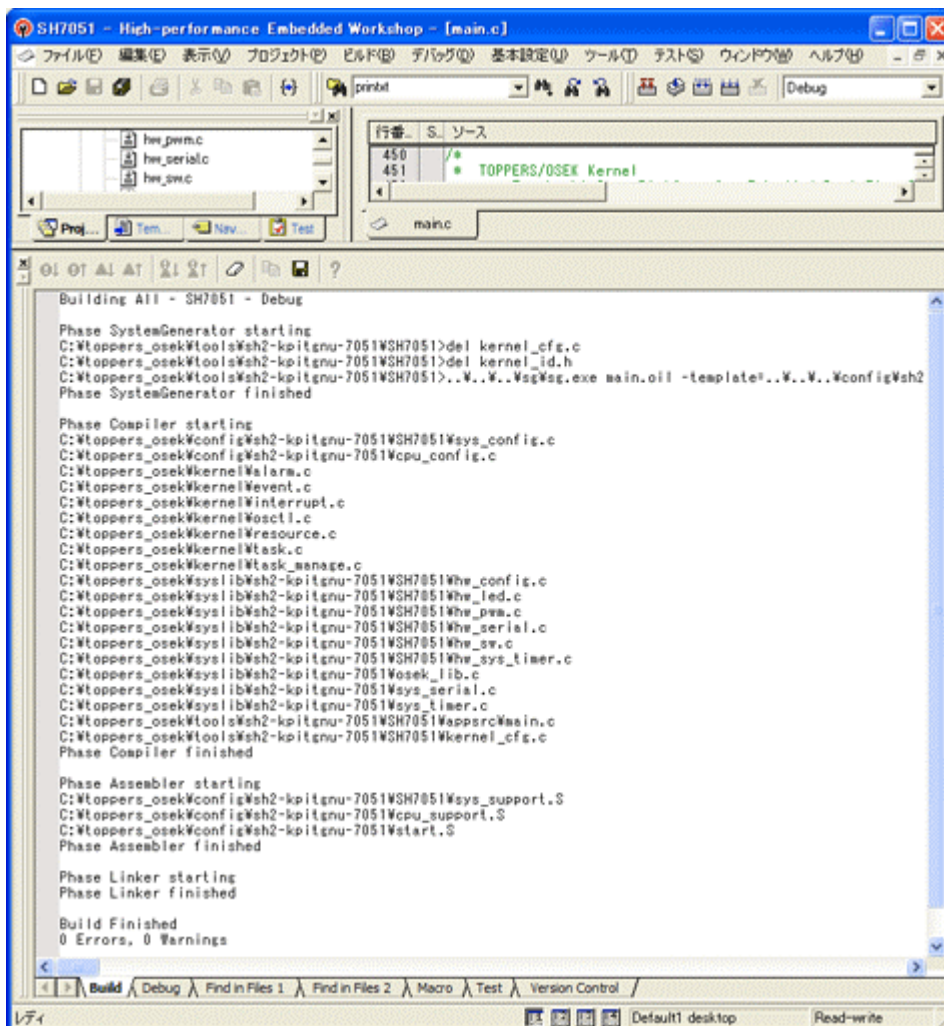
9) アクティブプロジェクトを「SH7051」に戻します

10) Hew メニューの<ビルド>-<すべてをビルド>をクリックして下さい。



[5-1-3]

1 2) ビルド結果

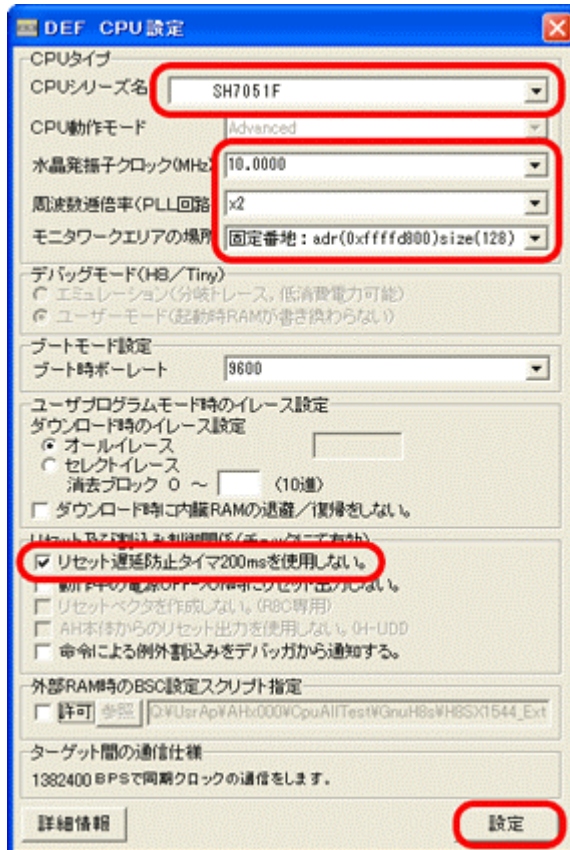


[5-1-4]

Build Finished 「0 Errors 0 Warnings」 になれば成功です。

5-2. H-debugger(DEF.exe)での準備 (Aone 製)

- 1) H-debugger コントロールソフト「DEF.exe」を起動します。
- 2) DEF メニューの<オプション>-<環境設定>の「本体機種設定」が正しい機種と COM ポートの選択をされているか確認して下さい。
- 3) DEF メニュー<オプション>-<CPU 設定>をクリックします。



[5-2-1]

CPU シリーズ名 : SH7051F

クロック (MHz) : 10.0000

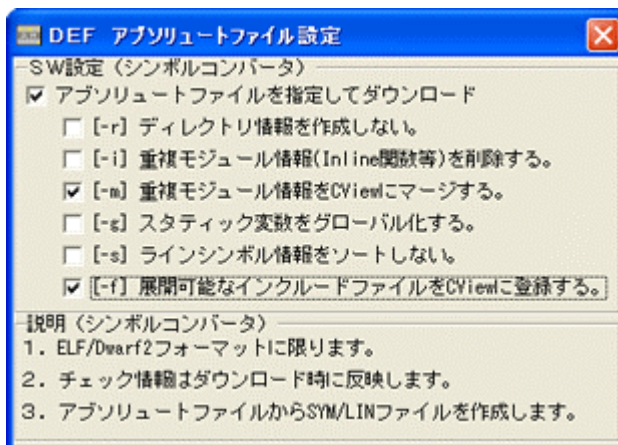
周波数通倍率 : x2

モニタワークエリアの場所 : 固定番地

リセット遅延防止タイマ使用しない :
チェック

最低上記 5 項目を設定後、「設定」をクリックします。

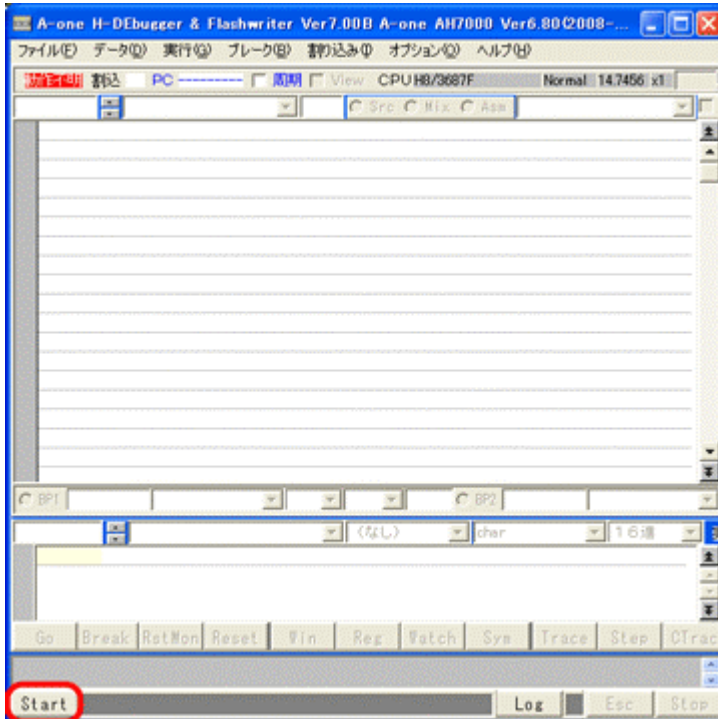
- 4) DEF メニュー<ファイル>-<アブソリュートファイル設定>をクリックします。



[5-2-2]

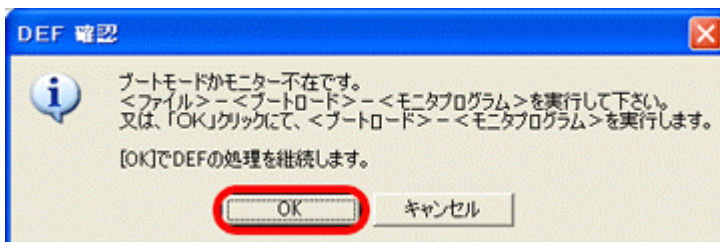
左図のように 3 箇所にチェックします。

- 5) DEF 画面、左下隅の「Start」をクリックします。(ターゲット側の電源は ON の事)



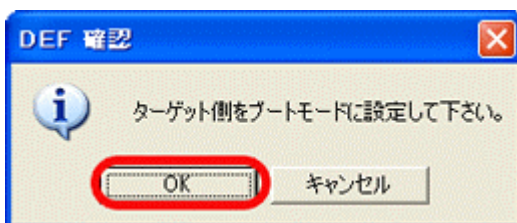
[5-2-3]

- 6) 最初だけモニタ不在の通知が表示されます。



[5-2-4]

「OK」をクリックします。



[5-2-4-1]

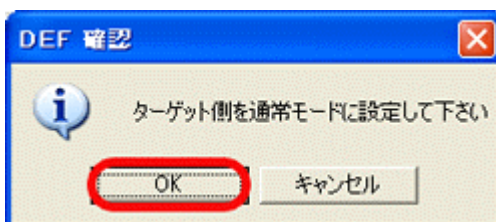
ブートモードに設定します。CPU 基板の JP1 2-3 をショートします。(MD1:LOW)

「OK」をクリックします。



[5-2-4-2]

モニタ書き込み中にインジケータが表示されます。

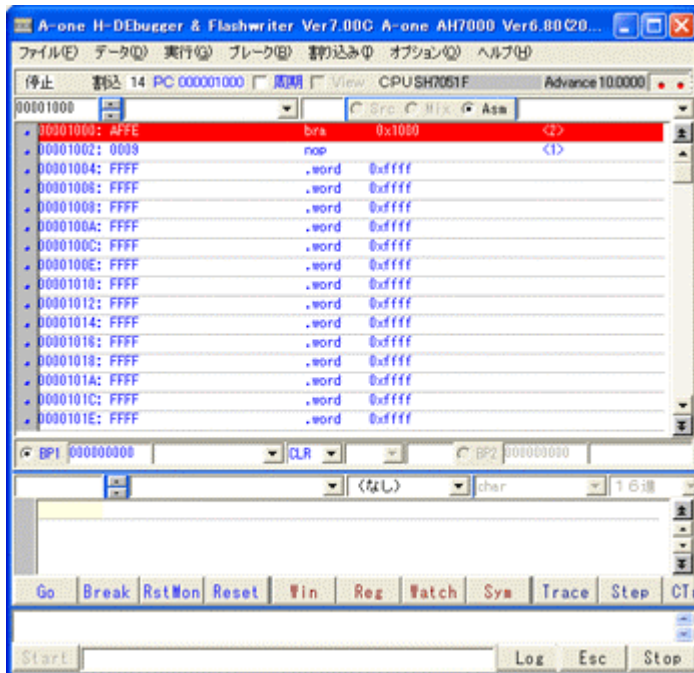


[5-2-4-3]

通常モードに設定します。CPU 基板の JP1 1-2 をショートします。(MD1-HIGH)

「OK」をクリックします。

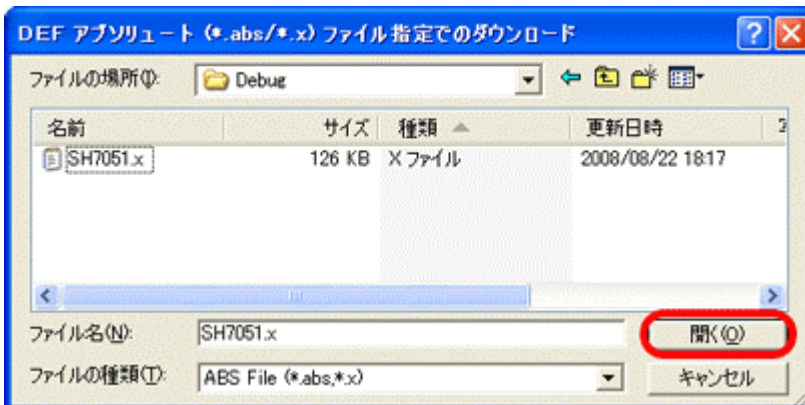
7) ターゲット側と正常な通信を確立しますと下図の様な画面になります。



[5-2-5]

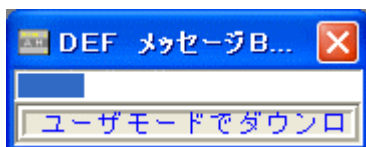
8) DEF メニュー<ファイル>-<ダウンロード>をクリックします。

.¥toopers_osek¥tools¥sh2-kpitgnu-7051¥tools¥SH7051¥Debug まで降ります。



[5-2-6]

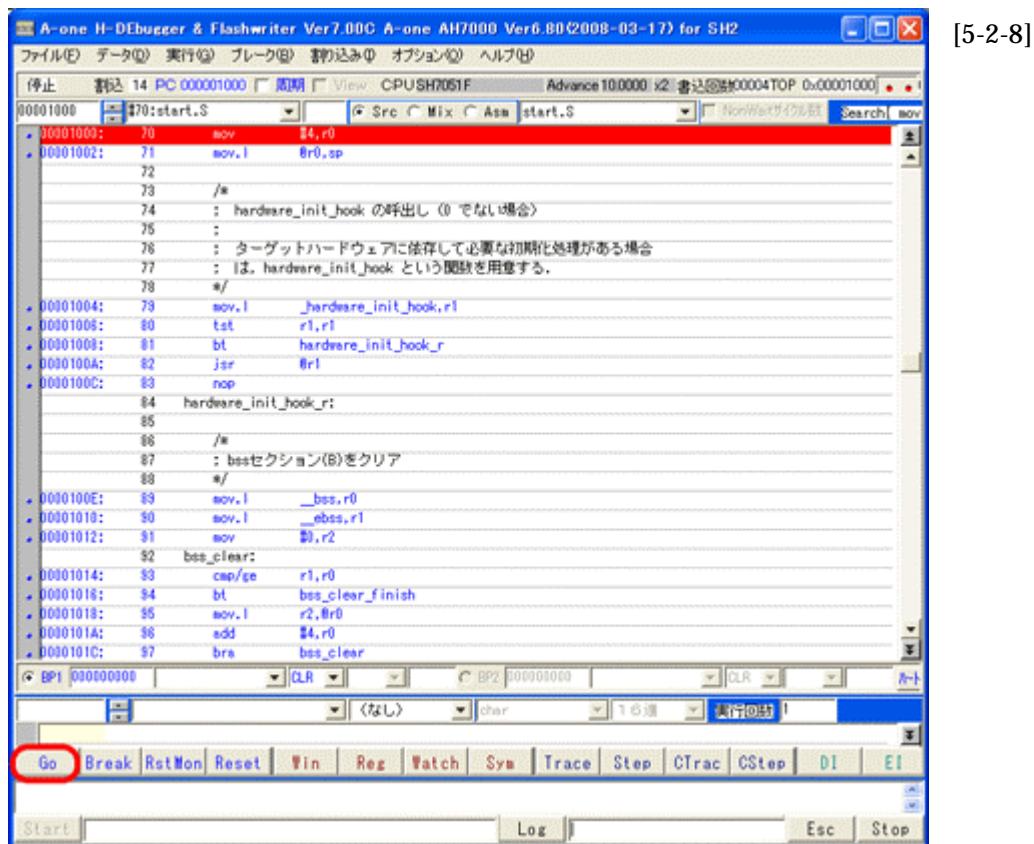
SH7051.x を選択後
「開く」をクリック
します。



[5-2-7]

ダウンロード中は、このようなインジケータ表示します。

9) ダウンロードが成功しますと、下図のような DEF 画面になります。

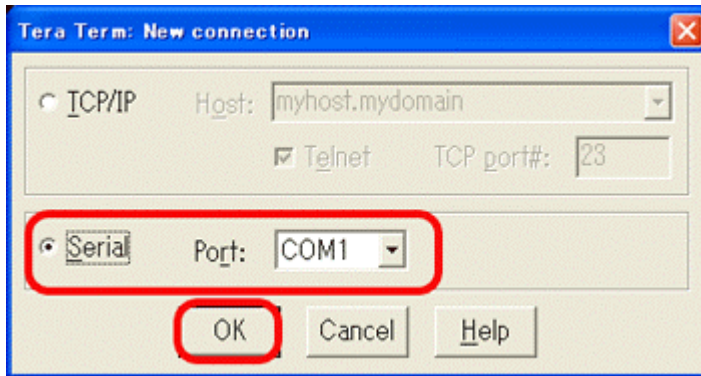


ShortPB[RstMon]On でこの View 画面になります

5-3. サンプルアプリケーションを走らせる前の準備

1) CPU 基板「CAT68201」上の DIP-SW(1)を OFF にします。

2) パソコン側のソフト「Tera Term」を起動します。

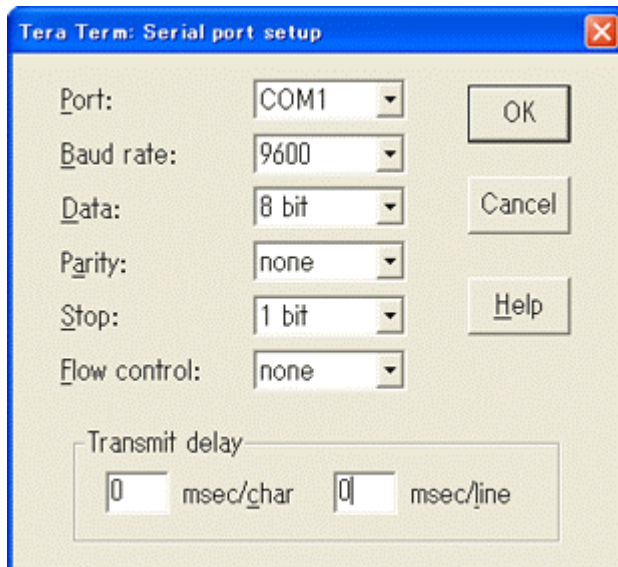


[5-3-1]

「Serial」にチェックを入れ、使用パソコンのシリアル COM 番号を指定します。

「OK」をクリックします。

3) 「Tera Term」メニューの<Setup>-<Serial port>をクリックします。



[5-3-2]

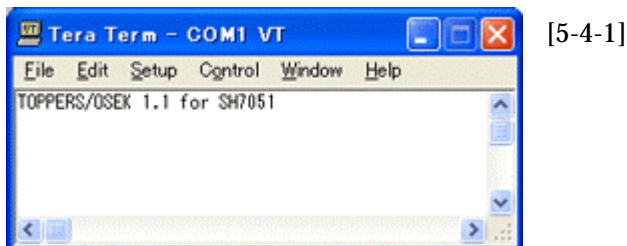
Port 以外の通信仕様を左図のように設定して下さい。

「OK」をクリックします。

5-4. サンプルアプリケーションを走らせます。

1) DEF のショート PB 「Go」 をクリックします。[5-2-8]図を参照

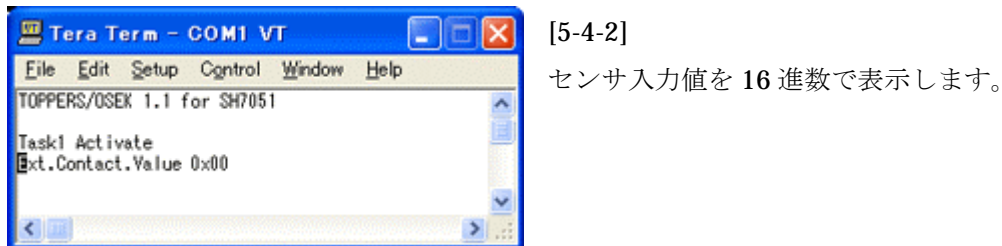
「Tera Term」画面は下画面のようになります。



2) センサ入力タスク(Task1)を起動します。

①CPU 基板「CAT68201」上の DIP-SW(1)を ON にします。

「Tera Term」画面は下画面のようになります。

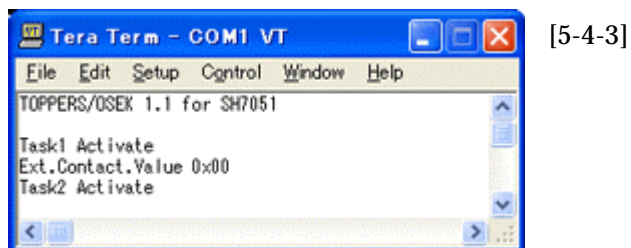


3) PWM 出力タスク(Task2)を起動します。

①モータドライブ基板側の電源「DC+6V」を供給します。

②モータドライブ基板の PushSW を ON します。

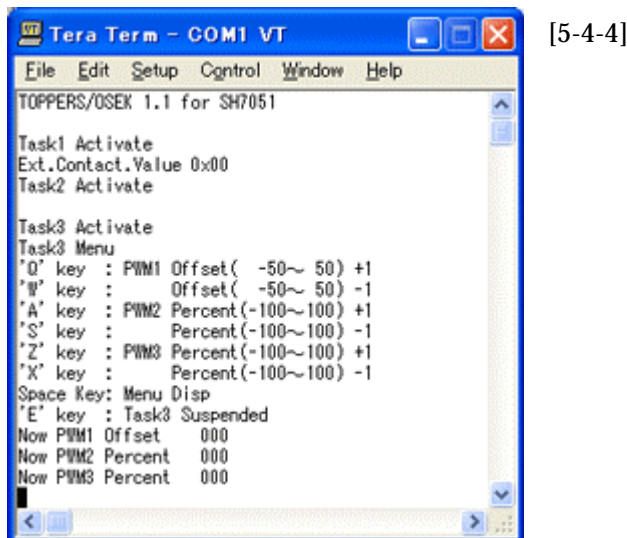
「Tera Term」画面は下画面のようになります。



4) RS 設定タスク(Task3)を起動します。

① 「Tera Term」 よりスペースキーを入力します。

「Tera Term」 画面は下画面のようになります。



②PWM1 (サーボ) を出力します。

[Q]と**[W]** キーを押して下さい。数値にもとずいてサーボが動作します。

③PWM2 (右モータ) を出力します。

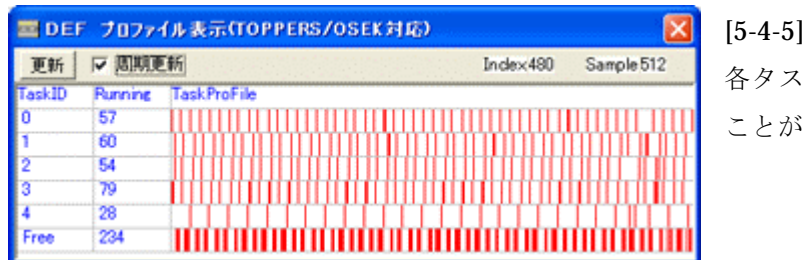
[A]と**[S]** キーを押して下さい。数値にもとずいて右モータが動作します。
+数値は正転、-数値は逆転します。

④PWM3 (左モータ) を出力します。

[Z]と**[X]** キーを押して下さい。数値にもとずいて左モータが動作します。
+数値は正転、-数値は逆転します。

5) 各タスクの **Running** の状態をしてみます。

①DEF メニュー<データ>-<プロファイル表示>をクリックします。



各タスクの **Run** 状態を見ることが出来ます。


6) 作者より

本サンプルの利用により、TOPPERS/OSEK カーネル下でのユーザーアプリケーションプログラムの作成に少しでもお役に立てれば幸いです。

第6章 新規プロジェクトを追加する場合の手順例

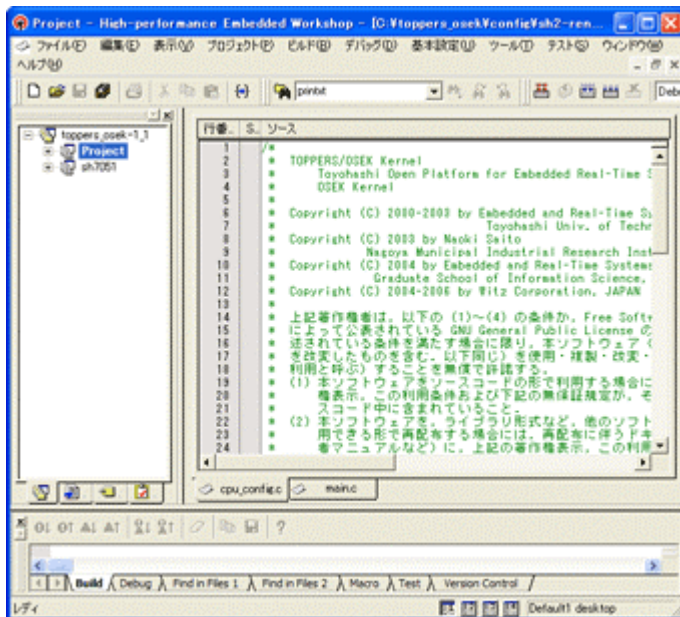
6-1. プロジェクトタイプの作成

TOPPERS/OSEK 下で新規プロジェクトを追加する場合、**Hew** 設定を簡略化するための手順記述します。

 **SH7051** 用のプロジェクトテンプレートは作成してありますので、下記手順でプロジェクトタイプ (カスタム) を作成して下さい。

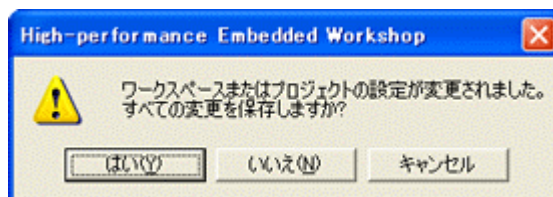
(必ず Compiler] と Assembler と Linker のディレクトリが変更済みである事!!)

- 1) 空プロジェクト「Project」をアクティブプロジェクトに指定します。



[6-1-1]

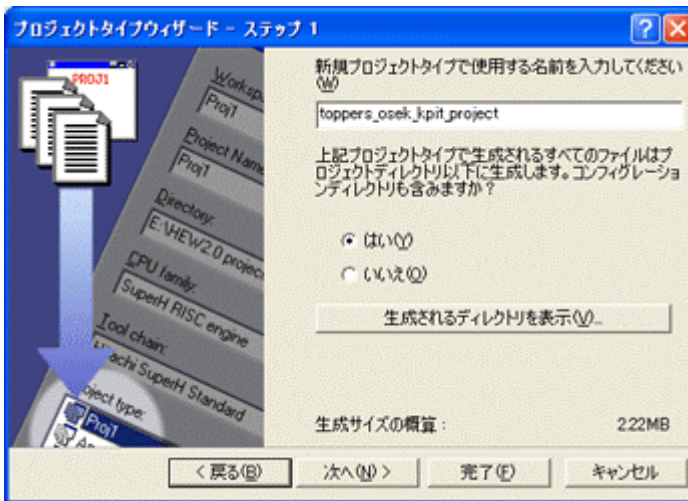
- 2) **Hew** メニュー<プロジェクト>-<プロジェクトタイプの作成>をクリックします。



[6-1-2]

保存を促すメッセージです。

「はい」をクリックします。

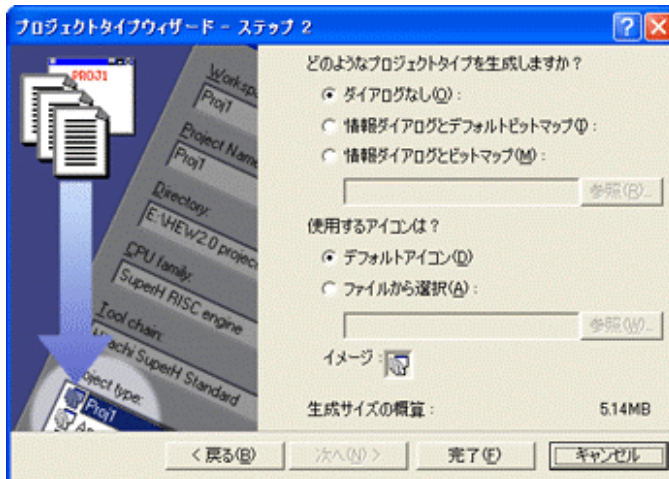


[6-1-3]

プロジェクト新規登録時に指定するタイプ名を入力する。
ex)toppers_osek_kpit_project

はい(Y)側をチェック

「次へ」をクリックします。



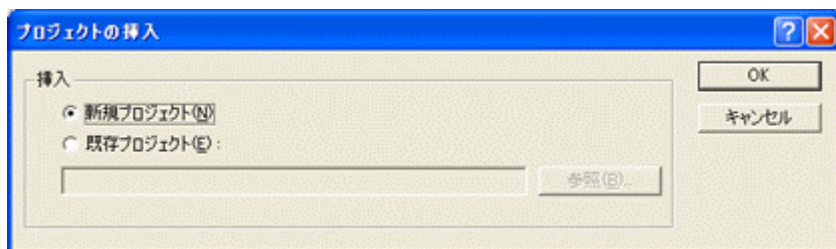
[6-1-4]

左図の様に「デフォルト」のままで、

「完了」をクリックします。

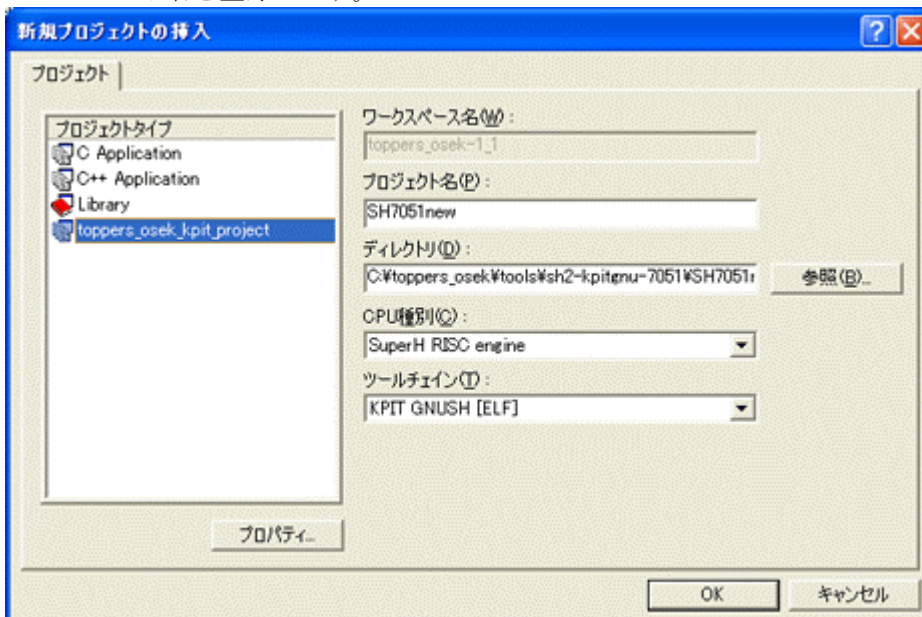
6-2. 新規プロジェクトを登録します。

1) Hew メニュー<プロジェクト>-<プロジェクトの挿入>をクリックします。



[6-2-1]
新規プロジェクト側チェックにて「OK」をクリックします。

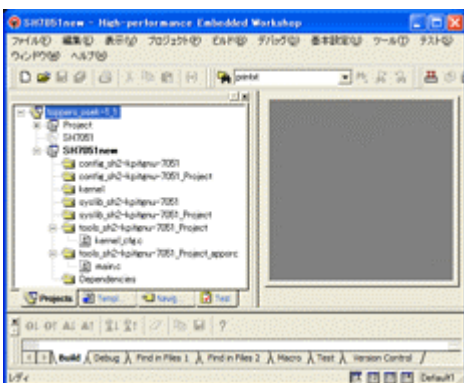
2) プロジェクト名を登録します。



[6-2-2]

- ①プロジェクトタイプを前項で登録した「toppers_osek_kpit_project」を指定します。
- ②プロジェクト名に任意な目的プロジェクト名を入力します。後の説明でプロジェクト名が必要になりますので、ここでは新規プロジェクト名を「SH7051new」とします。
- ③上記設定で「OK」をクリックします。

3) Hew に新規プロジェクトが作成されます。



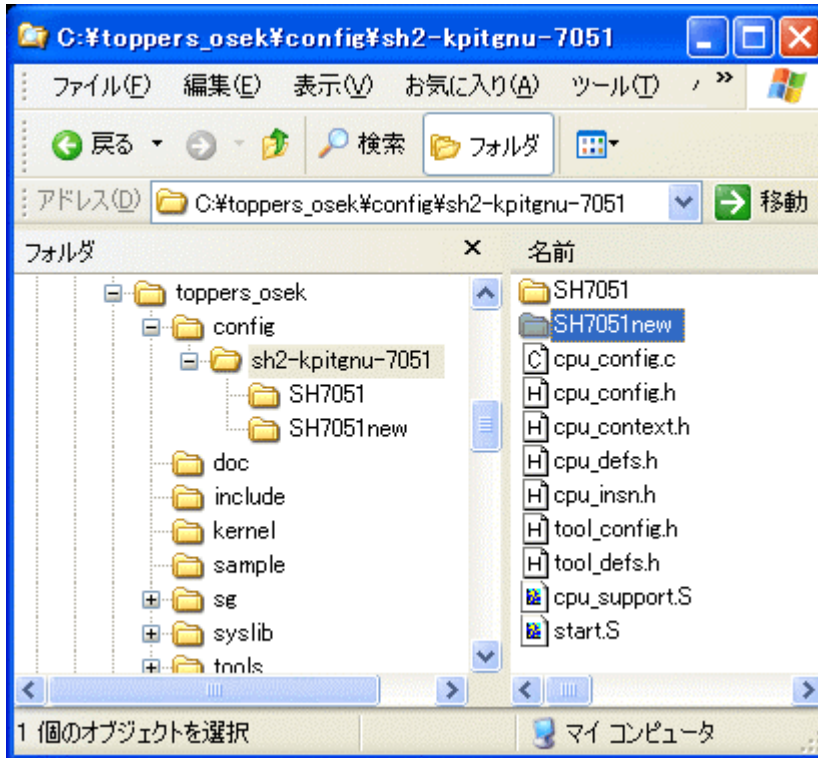
[6-2-3]

左図のように、新規プロジェクトが Hew に登録されます。

4) 新規プロジェクト用に新規ディレクトリ作成とファイルをコピーします。

① .¥toppers_osek¥config¥sh2-kpitgnu-7051 の下に、

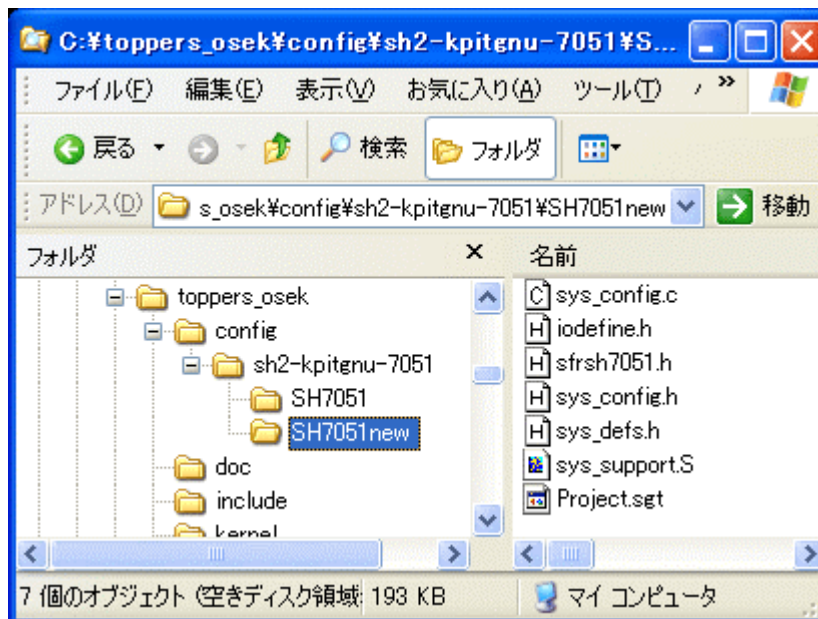
新規プロジェクト名「SH7051new」のディレクトリを作成します。



[6-2-4-1]

<追加 1 >

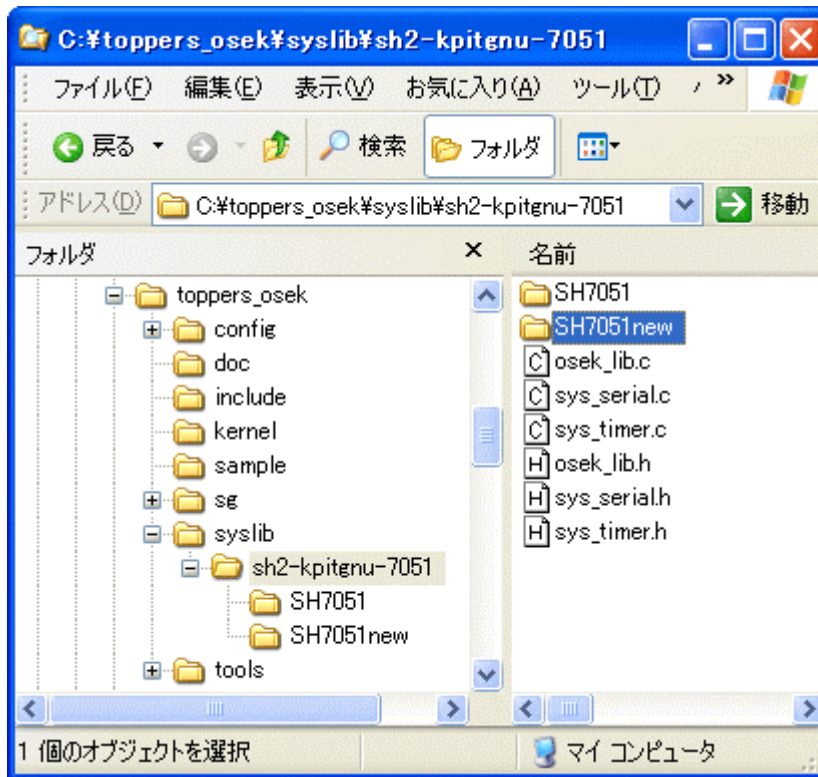
②作成した「SH7051new」に、「SH7051」下の全ファイルをコピーします。



[6-2-4-2]

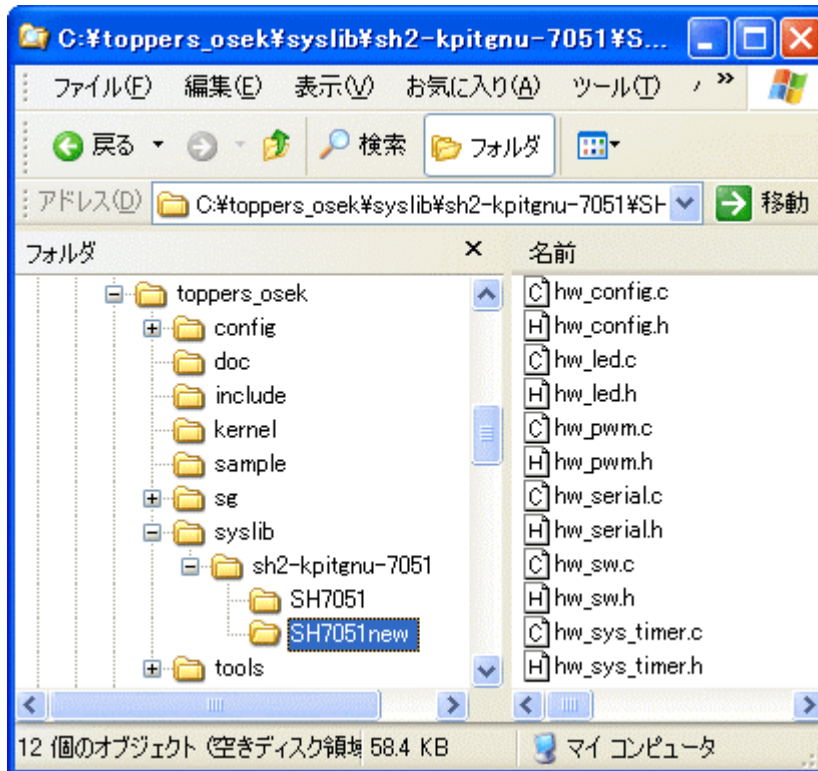
コピー確認

- ① .¥toppers_osek¥syslib¥sh2-kpitgnu-7051 の下に、
新規プロジェクト名「SH7051new」のディレクトリを作成します。



[6-2-4-3]
<追加 2 >

- ②作成した「SH7051new」に、「SH7051」下の全ファイルをコピーします。



[6-2-4-4]
コピー確認

5) システムジェネレータ用バッチファイルの一部を変更します。

`.\tools\sh2-kpitgnu-7051\SH7051new\call_sg.bat`

のファイルを何らかのエディタで開きます。

<元ファイル> [6-2-5-1]

```
@REM SG 実行バッチファイル

@REM カーネルコンフィグレーション
del kernel_cfg.c
del kernel_id.h
..\..\sg\sg.exe main.oil
-template=..\..\config\sh2-kpitgnu-7051\Project\Project.sgt
-I..\..\sg\impl_oil -os=ECC2
```

<変更ファイル> [6-2-5-2]

```
@REM SG 実行バッチファイル

@REM カーネルコンフィグレーション
del kernel_cfg.c
del kernel_id.h
..\..\sg\sg.exe main.oil
-template=..\..\config\sh2-kpitgnu-7051\SH7051new\Project.sgt
-I..\..\sg\impl_oil -os=ECC2
```

上記の様に、**Project** を新規プロジェクト名「**SH7051new**」に変更します。

6) 新規プロジェクト「SH7051new」にユニット「ソースファイル」を登録します。

①フォルダ名「config_sh2-kpitgnu-7051」に登録

Hew メニュー<プロジェクト>-<ファイルの追加>をクリックします。

.¥toppers_osek¥config¥sh2-kpitgnu-7051 に移動します。

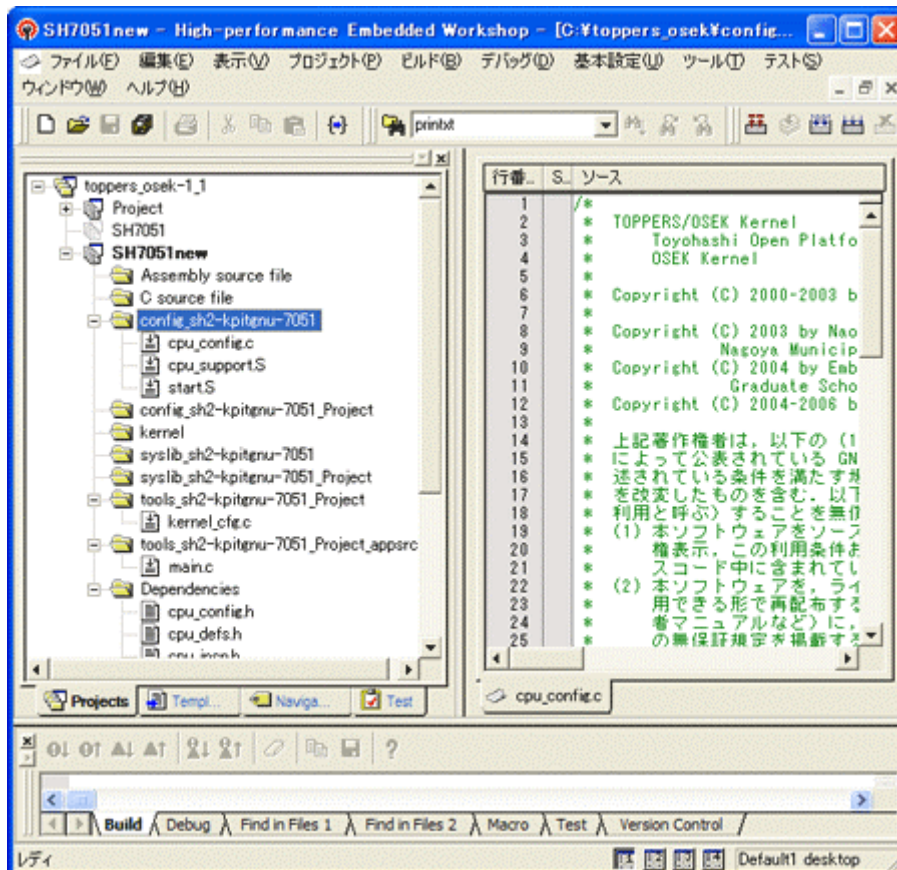


[6-2-6-1]

- ①cpu_config.c
- ②cpu_support.s
- ③start.s

の 3 ファイルを「追加」します。

ファイル選択後「追加」をクリックしますと、Hew ツリーでは一旦「Assembly souce file」と「C souce file」のフォルダに入りますので、マウドロップにて目的フォルダに移動して下さい。



[6-2-6-2]

移動後の状態

②フォルダ名「config_sh2-kpitgnu-7051_Project」に登録

Hew メニュー<プロジェクト>-<ファイルの追加>をクリックします。

.¥toppers_osek¥**config**¥sh2-kpitgnu-7051¥**SH7051new** に移動します。



[6-2-6-3]

- ①sys_config.c
- ②sys_suport.s

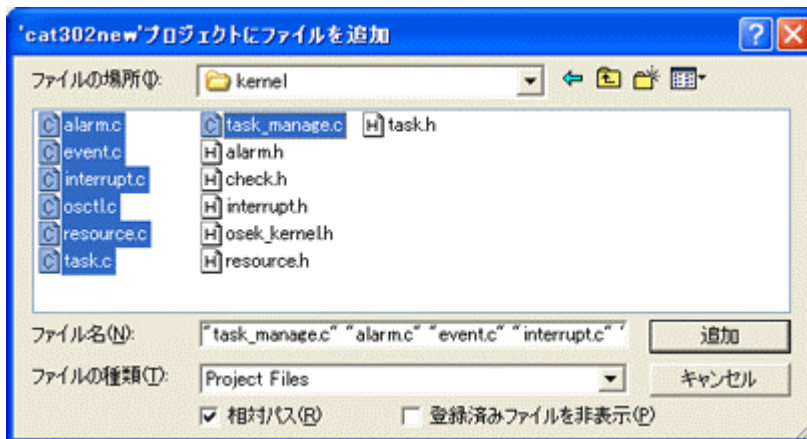
の 2 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウสดロップで目的フォルダに移動して下さい。

③フォルダ名「kernel」に登録

Hew メニュー<プロジェクト>-<ファイルの追加>をクリックします。

.¥toppers_osek¥**kernel** に移動します。



[6-2-6-4]

- ①alarm.c
- ②event.c
- ③interrupt.c
- ④osctl.c
- ⑤resource.c
- ⑥task.c
- ⑦task_manage.c

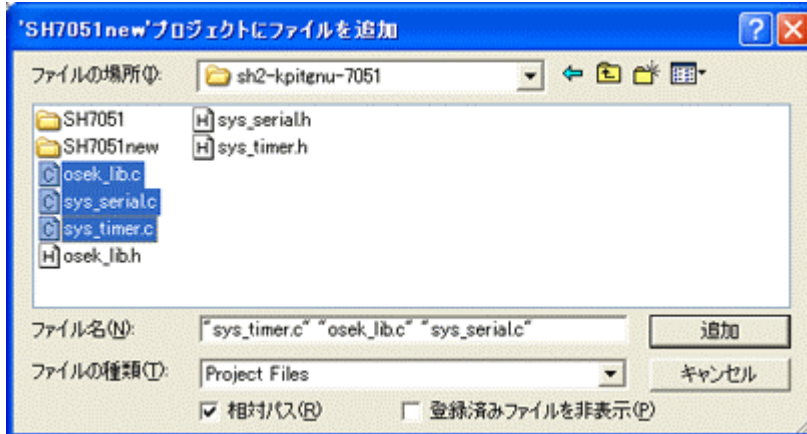
の 7 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウสดロップで目的フォルダに移動して下さい。

④フォルダ名「syslib_sh2-kpitgnu-7051」に登録

Hew メニュー<プロジェクト>-<ファイルの追加>をクリックします。

.¥toppers_osek¥**syslib**¥sh2-kpitgnu-7051 に移動します。



[6-2-6-5]

- ①sys_timer.c
- ②sys_serial.c
- ③osek_lib.c

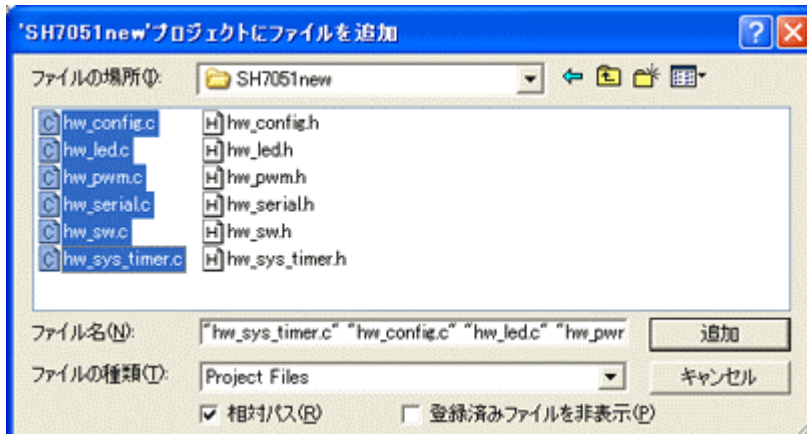
の 3 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウสดロップで目的フォルダに移動して下さい。

⑤フォルダ名「syslib_sh2-kpitgnu-7051_Project」に登録

Hew メニュー<プロジェクト>-<ファイルの追加>をクリックします。

.¥toppers_osek¥**syslib**¥sh2-kpitgnu-7051¥**SH7051new** に移動します。



[6-2-6-6]

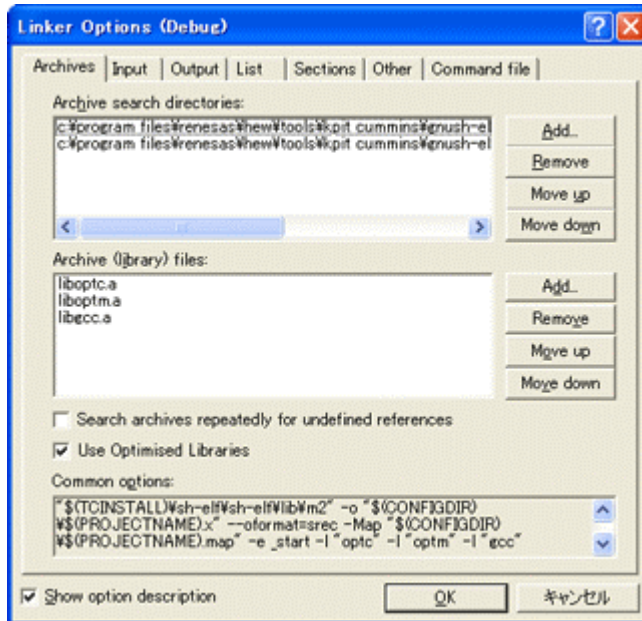
- ①hw_config.c
- ②hw_led.c
- ③hw_pwm.c
- ④hw_serial.c
- ⑤hw_sw.c
- ⑥hw_sys_timer.c

の 6 ファイルを「追加」します。

前項と同じ様に、別フォルダに入りますので、マウสดロップで目的フォルダに移動して下さい。

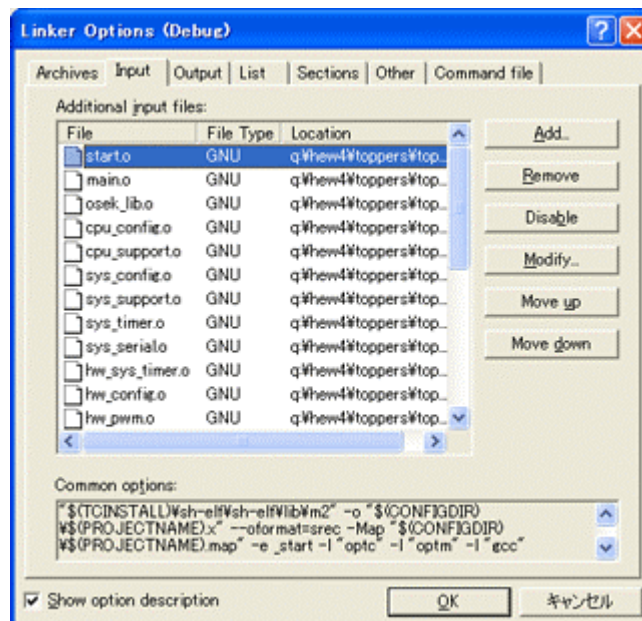
7) 新規プロジェクト「SH7051new」のリンク順番を指定します。

Hew メニューの<ビルド>-<Linker>をクリックして下さい。



[6-2-7-1]

<Input>タブをクリックします。

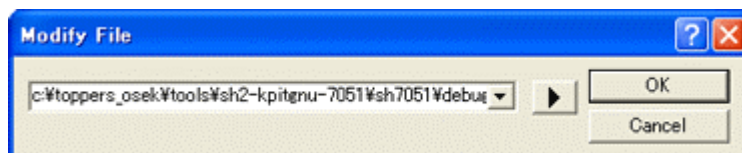


[6-2-7-2]

全てのディレクトリを現ワークスペースのディレクトリと相違があった場合は現ディレクトリに変更して下さい。


設定は、「\$(CONFIGDIR)*.o」にしましたので変更する必要は無いかと思えます。

マウスでのダブルクリックで変更できます。



[6-2-7-3]

変更が必要な場合は、「SH7051new」のディレクトリに変更する。

 以上の作業で、新規プロジェクトの追加作業は終了です。目的のプロジェクト仕様に合わせた「main.c」を作成して下さい。

フォルダ名「tools_sh2-kpitgnu-7051_Project_appsrc」

ディレクトリ「.¥toppers_osek¥tools¥sh2-kpitgnu-7051¥SH7051new¥appsrc」に空ファイルとして用意してあります。

- 8) OS 定義および Task の追加やプライオリティを変更したい場合は、ディレクトリ「.¥toppers_osek¥tools¥sh2-kpitgnu-7051¥SH7051new」にある、「main.oil」のテキストファイルを変更して下さい。

定義仕様に関しては、「OPPERS/OSEK カーネル SG 取扱説明書」株式会社ヴィッツ製をご覧ください。

第7章 備考

7-1. おわりに

本アプリケーションノートは、いたらない所が多々有ると思います。意味不明な箇所がありましたら、遠慮なくメールにて申し付け下さい。積極的に改訂し、より判り易いノートにしたいと思いますので、皆様の御協力を御願い申し上げます。

2008年8月 著者

〒486-0852
愛知県春日井市下市場町 6-9-20
エーワン株式会社
Tel 0568-85-8511
Fax 0568-85-8501
E-mail cat-i@aone.co.jp
URL <http://www.aone.co.jp>