B C S H 7 2 X X 用 フォース[®]デパッカ S H の使い方

第1版2008.12.05

描 Debug	
- コンパイル C sample1 C sample5 C sample9 C sample2 C sample7 C sample4 C sample7 C sample4 C sample8 C sample4 C sample8 C sample4 C sample8 C sample5 C samp	ブレークポイント設定、動作 アドレス
C.¥GNUSH7243¥sample_7243¥sample2>sh-elf-objdump -h sample2elf 1> sample2.map	(1) (1) (1) (1) (1) (1) (1) (1) (1)
ソースファイル + アドレス	アドレス(最大B桁 fffffff)
800: 00000800 <_start>:	
810: 00000810 <main_k>:</main_k>	データ数 リード
00000820 <_init SIO>: unsigned short hexascii(unsigned char dat);	最大2桁(ff) 1/0、メモリ ライト
void init_SIO(void) //38400bps	
unsigned short sbuff;	バイト C ワード C ロングワード
・ 必ず動作周波教設定後(周辺クロック設定後)にコールし 例: FRQCR = 0×0101; //B φ =50MHz 1 φ =101 🛩	Debug Monitor
▼ auto_load ロード 画面クリア main_ 検索 次を検索 ▲ ▼ FOUスト	ADI=962F AD2=062F TEST No? 1:A/D 2:I/O 3:SIO? 134241 134642 134662 TEST No? 1:A/D 2:I/O 3:SIO?
sh7243.h 2008.5.9 AA	
_FRQCR 検索 max5桁比較 ▲ ↓ ▼	画面クリア 「タイムスタンプ ファイル出力」

1.デバッカ機能、使い方

- 1-1 概要
- 1-2 コンパイル
- 1-3 ソースファイル+アドレス
- 1-4 IOリスト
- 1-5 ブレークポイント
- 1-6 変数アドレス
- 1-7 コマンド
 - a)リードライト
 - b)ウオッチ
 - c)ファンクション
 - d) キーボード
- 1-8 タイムスタンプ、デバック状況ファイル出力

1-1 概要

本デバッカはフラッュ ROM 書き込みソフトウエア フォース®ライタの拡張機能として開発されました。 また、従来、単独プログラムとして提供されていた「コンパイラー」がデバック画面に組み込まれていま す。これにより、プログラムの開発に必要な、コンパイル、フラッシュROM書き込み、デバックという 一連の作業が有利な連結を伴ってシームレスに行えるようになっています。

デバッカは CPU 内蔵の UBC (ユーザーブレークコントローラ)を使用したリモートデバッカです。イン サーキットエミュレータを使用しなくとも、単体でプログラムをデバックできます。

内部フラッシュ ROM に書かれたプログラムが実行している最中に UBC のハードウエアでブレークを掛けるため、デバックと、実動作時で速度が変わらない理想のリアルタイムデバックです。 1

プログラムの変更なしに好きなアドレスにブレークポイントを設定してメモリや I/O の値をリード、ラ イトすることができます。 2

また、この他に特定のメモリ、1/0 の値と設定値を比較してアラームを出力するウオッチ機能が使用で きます。デバックログのファイル化や、タイムスタンプ機能と相まって頻度が少ないエラー発見、長時間 の正常動作確認、ログファイルによるエラー解析等、今までのデバッカを越える、強力な機能も使用でき ます。

1 デバック時にデバック機構の制限により CPU の最高性能が低下することはありません。

2 ブレークポイントの変更の度にフラッシュ ROM を書き換えることがありません。

フォース®および FORCE®は弊社の登録商標です。

勝手ながら以下の説明は「SH72XX 開発セット」全般、特に「sample6」の取扱説明書を読了されていると いう前提となっています。以下の説明は SH7243 用フォースライタで行いますが、SH7211 用も使い方は同 じです。

1-2 コンパイル

BRE SH7243 FLASH	I ROM W	RITER	
FWRITE 100MH	z専用	SOUN	D
0.8	008		
SIO SEL		×	
💿 🖸 COM1 🔿 CO	M4	Debug	
C COM2 C CO	M5		
C COM3 C CO	M6		

例として開発セットのsample2をコンパイルしてみます。sample2を選択し、コンパイルを クリックします。

🙀 Debug	
コンパイル結果 Sample1 Sample5 Sample9 Sample2 Sample5 Sample9 Sample3 Sample7(user) Sample4 Sample8 TEST コンパイル正常終了 C+ END C+ END C+ END C+ END -ソースファイル+アドレス	ブレークポイノト設定、動作 アドレス セット 解除 クリア 使火ト 解除 クリア 「#10000 ロセット & スタート 継続スタート コマンド 変数アドレス 「#180008 000000002 data1 リードライト ウオッチ ファンクション) ▲ ▼ リードライト ウオッチ ファンクション) キー アドレス(最大8桁i fffffff) リード データ数 リード レの、メモリ リード データ数 リード 最大2桁(ff) レク、メモリ ライト データ数 ライト 最大8桁i(fffffff) ライト データ数 ライト
	画面クリア 「タイムスタンプ ファイル出力」

問題なければ「コンパイル結果」欄に「コンパイル正常終了」と表示されます。問題がある場合「コンパ イルエラー」「リンクエラー」と赤字で表示されますので、ソースファイルを修正する必要があります。

下記例ではsample2.cにわざと「pppp」と書き込み、エラーを出しました。

エラー行が矢印 で表示されますので、エディタでその行か、1行上の行を調べてエラーを見つけてください。

例では167行目にエラーがあると表示されています。

🙀 Debug	
ユンパイル C sample1 C sample5 C sample9 C sample2 C sample6 C sample3 C sample7(user) C sample4 C sample7 TEST コンパイルエラー C ¥GNUSH7243¥sample_7243¥sample2>sh-elf-gcc -m2 -g -c -Wall sample2.c ▲ -o sample2.c → sample2.c 167: error: syntax error before "int" -ソースファイル+アドレス	ブレークボイント設定、動作 アドレス ・ セット 解除 クリア ・ セット 解除 クリア ・ ・ セット 解除 クリア ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
▲ ■ auto_load ロード 画面クリア - ^{main_} 検索 次を検索 ▲ ▼ 10リスト ● ^{/*} sh7243h 20085.9 AA	
, 	画面クリア 「タイムスタンプ ファイル出力」

なお、コンパイラーは warning (警告)は通常、問題なしとしますが、「コンパイル正常終了」と表示されても、motファイルが生成されない場合が稀にあります。No such file or der ectoryと表示されますので、ソースファイルの修正、再コンパイルが必要です。

C sample1 C sample5 C sample9 コンパイル正常終 C sample2 C sample6 C sample3 C sample7 user			コンバイル結果		バイル
C sample3 C sample7 user	{7	E常終	コンパイル正	C sample9	sample1 C sample5 sample2 C sample6
C sample4 C sample8 コンパイル C+ EN	タ連動 ・		コンパイル	user	sample3 C sample7 sample4 C sample8

プログラムを修正した場合、必ずコンパイル、書き込みが必要です。「ライタ連動」をチェックしておくと、 コンパイルが正常終了したときのみ、ライタが起動して書き込み準備に入りますから便利です。 「コンパイル正常終了」 「プログラム書き込み」を行います。

例:sample2をコンパイルし、「コンパイル正常終了」だったので、自動的に書き込むファイルが表示されているところ。ここでは sample2.mot を選択し、書き込みます。

B.R.E SH7243 FLASH	ー (ファイルを開く		? [>
書き込みファイル選択 FWRITE 100MHz 0X SIO SEL ○ COM1 ○ CON ○ COM2 ○ CON ○ COM3 ○ CON	ファイルの場所 Φ: sample2.mot sh_start0.s	Sample2	▼ ← È r ∰•
Copyright Beyond th	ファイル名(<u>N</u>): ファイルの種類(<u>T</u>):	MOTファイル(*.MOT,*.S) ▼ 読み取り裏用ファイルとして聞く(B)	■ 「開(Q) ★ャンセル
コンパイル C sample1 C samp Sample2 C samp C sample3 C samp C sample4 C samp	ole5 O sample9 le6 le7 user le8	コンパイル結果 コンパイル正常終了 コンパイル ロコンパイル コンパイル コンパイル	

1-3 ソースファイル+アドレス

コンパイルをすると、ユーザーが製作したソースファイルに絶対番地が割り振られます。ソースファイ ル+アドレス(窓)はそれを表示し、検索等行い、ブレークポイント設定時に参照するための窓です。

画面はエディタではありません。コピーはできますが、書き込みはできないのでご注意ください。本環 境のエディタはユーザーが使い慣れた個々のエディタをご使用いただく前提です。

ソースファイル + アドレスファイルは c:¥GNUSH72XX¥bre-sample¥common の中にある cplus.lst ファイル です。コンパイルのたびに更新されます。プリンターで印刷できます。

C:¥GNUSH7243¥sample_7243¥common

	名前 🔺	サイズ	種類
レとフォルダのタスク 📀	CmdSH7243.cmd	1 KB	Windows NT コマンド
しいフォルガを作成する	CmdSH7243_user.cmd	1 KB	Windows NT コマンド
ションリガナ いた (二八日子ろ	🗩 cplus.lst	16 KB	LST ファイル
カフォルジを Web にム(第)9つ	📄 symbol.tbl	1 KB	TBL ファイル

例としてsample6のソース+アドレスファイルが読み込まれた状態を示します。右がユーザーが 作成したプログラム、左がコンパイラが割り振ったアドレスです。

「autoload」をチェックしておくと、コンパイル正常終了時に自動的に読み込まれ便利です。

-ソースファイル+アドレス
800: 00000800 <_start>:
810: 00000810 <main_k>:</main_k>
00000820 <_init_SIO>: unsigned short hexascii(unsigned char dat);
void init_SIO(void) //38400bps 820: { unsigned short sbuff; /*
 必ず動作周波数設定後(周辺クロック設定後)にコールし 例:FRQCR = 0x0101; //B φ = 50MHz I φ = 10
✓ auto_load (二二下) 画面クリア (検索) 次を検索 ▲ ▼

例えばアドレス 820 番地

void init_SIO(void) //38400bps

820: {

は関数 init_SIO の先頭アドレスを示しています。自分で書いた記憶の無いソースファイルですが、

#include "SI0_SH72XX.h"

と書いて include した SIO_SH72XX.h がここに展開されています。ユーザーが書いたmain関数は普通、 一番後ろにあります。

下のキーは

- 「 a u t o _ l o a d 」: コンパイルが正常終了すると自動的にソース + アドレスファイルを読み込み 、表示します。
- 「ロード」 : ソース + アドレスファイルの読込

「画面クリア」 :画面クリアです。

- 「検索」 : ユーザーがブレークポイントを掛けたい部分のアドレスを知るために文字検索が 行えます。0番地から検索します。最大6文字入力です。デホルトは_mainです。
 「次を検索」 : 0番地からでなく次を検索します。
- 「 」 : 1 行下を表示します。

例としてデホルトの_mainを検索してみます。

検索して、ヒットすると左のがになり、右にヒット文字が表示されます。

レン-	-スファイ)	レ+アドレス			
	1bca: 1bd2: 1bda: 1be2:	char_out0('A'); char_out0('K'); char_out0('>'); char_out0(0x0d);		^	
	1bea:	comm_mov0;	//コマンド解析、実行		
	16f0: ;}	EINT;	//割り込み許可(レベル7設定)		
	00001c10 <_main>:				
	unsi∉	med short loopm,sdata1,sdata2;			
•	int m	nain (void)		*	
	<	10	>		
	auto_load	ロード 画面クリア -main-		<u> </u>	

[「]次を検索」をクリックしましたが、__mainは無いようです。 になります。

-ソ-	-スファイ)	レキアドレス				
	1bca: 1bd2: 1bda: 1be2:	char_out0('A'); char_out0('K'); char_out0('>'); char_out0('0x0d);				
	1bea:	comm_mov0;	//コマンド解析、実行			
	1Ьf0: ;}	EINT;	//割り込み許可(レベル7設定)			
	00001c10 <_main>:					
	unsigned short loopm,sdata1,sdata2;					
•	int m	nain(void)		~		
	<	10	>			
•	auto_load	ロード 画面クリア -main_	検索 [沈宏横索] _▲ _▼			

「 」を何回かクリックして__m a i n の中を表示させます。

[ノースファ	イル+アドレス	
	ur	nsigned short loopm,sdata1,sdata2;	
	in 1-10-1	t main(void)	
	1c10: 1 1c16:	FRQCR = 0x0101;	//Вф=50MHz、Iф=100MHz、 Р
		//SIO1 initial	
	1c1c:	init_SIO0;	//SIO0を使用する
		//Debug 設定	
	• 1c22:	init_debug0;	//デバック初期設定 🛛 🗸 🥃
	<	III	
ſ	✓ auto_lo	ad ロード 画面クリア -main_	検索 次を検索 ▲

「 」「 」は1クリック1行変わります。スクロールバーはマウスで連続して変わります。

1-4 IOUスト

10 リストは CPU の 1/0 の番地、形を知るための窓です。SH72XX.hファイルを読み込んでいます。

-10リスト			
• /* si	n7243.h 2008.5.	9 AA	▲ ▼
FRQCR	検索	max5桁比較	▲ _ ▼

他の窓と同様に検索ができます。ヒットすると左のがになります。最大 5 桁文字や数値が入力できます。

#0	define FR	QCR	(*(volati	le unsigned :	short *)0;	(fffe0010)	
[F	RQCR	ł	*	max5桁fb	上車交	A	
L	:1行_	上を表示	します。				
L	:1行	下を表示	します。				

例として FRQCR が表示されています。アドレスが fffe0010H、short なので 2 バイトデータであることがわ かります。後述する I/O リード、ライトを行うときに、アドレスをこの表示を参考に書くか、コピー&ペ ーストすることができます。

1-5 ブレークポイント

ブレークポイントはプログラムを一時停止させたい部分のアドレスを設定します。プログラムがそのアドレスになると一時停止して、メモリや I/O の読み書き等が出来ます。

ーブレークポイント記 アドレス	設定、動作		
•	セット	解除	クリア
停止中	リセット&スタート		維続スタート

「アドレス」 :アドレスを設定 0~fffffff 8桁 左詰めで設定します。

「セット」 : ブレークポイントをアドレスで示したアドレスに設定します。

「解除」 :ブレークポイントを解除します。

「クリア」 :アドレスをクリアします。

「リセット&スタート」 : CPU ボードにリセットを掛けてプログラムを0番地から動作させます。 過去に設定したプレークポイントも解除されますので、必要に応じて 下記条件時に再度「セット」してください。 DebugMonitorにFAST BREAK> と表示されます。

「継続スタート」 :プログラムが一時停止した番地の次から動作させます。設定等は継続 されます。消去されません。

「セット」「解除」「クリア」はいずれも DebugMonitor に FAST BREAK > または UBC BREAK > と表示されていて、CPU が「停止中」である時、操作可能です。

FAST BREAK> UBC BREAK>	
	<u></u>
画面クリア 「タイムスタンプ	ファイル出力

 FAST BREAK >
 : init_debug() 関数が実行されたときに表示されます。CPU はパソコンからの

 コマンド待ち状態です。「リセット&スタート」クリックで表示されます。

UBC BREAK > : ブレークポイントでブレークした時に表示されます。CPU はパソコンからのコマンド待ち状態です。「継続スタート」でブレークしたアドレスの次から動作を再開します。

1-6 変数アドレス

コンパイルすると自動的にシンボルテーブルアドレスが製作され、読み込まれまれ、表示されます。 シンボルテーブルファイルは c:¥GNUSH72XX¥bre-sample¥common の中にある symbol.tbl ファイルです。コ ンパイルのたびに更新されます。プリンターで印刷できます。

	名前 🔺	サイズ	種類
レとフォルダのタスク 🔹	CmdSH7243.cmd	1 KB	Windows NT コマンド
しいフォルガを作成する	CmdSH7243_user.cmd	1 KB	Windows NT コマンド
	🔎 cplus.lst	16 KB	LST ファイル
ルノオルダを Web にム南9つ	📄 symbol.tbl	1 KB	TBL ファイル

C:¥GNUSH7243¥sample_7243¥common

sample6 を例に挙げるとグローバル変数 unsigned short loopm,sdata1,sdata2; がその振られた絶対番地とデータサイズを確認することができます。

「 」「 」で上下できます。検索機能はありません。

sdata2=fff80036H 2バイトデータだと分かります。

דידר	変数アドレス
fff80036 00000002 _sdata2	
fff80000 0000000 _bss	

loopm=fff8003cH 2バイトデータだと分かります。

-コマンド	変数アドレス	
fff8003c 00000002 _loopm		
11180000 0000000 _data		

sdata1=fff8003eH 2バイトデータだと分かります。

変数アドレス	

この窓のアドレスを次項のリード、ライトアドレス等にコピー&ペーストして変数の内容を読み書きす ることができます。本機能はローカル変数は対応していません。調べたい変数がある場合、グローバル変 数として設定願います。

1-7 コマンド

a)リードライト

1/0、メモリのリードライトを行います。数値設定は全て左詰めです。

読み込みたい 1/0、メモリの先頭アドレスを設定



書き込むデータを設定 最大 8 桁 0~FFFFFFFF

SH や H8 はメモリマップド I/0 なので、基本的にアクセスは I/0 もメモリも同じです。 例として、sdata1 のデータを読んでみます。

変数アドレスで sdata1 のアドレスをコピーします。

変数アドレス	
SCSATT DA	

I/0、メモリリードライトのアドレスにペーストします。

リードライト	ウオッチ	ファン
アドレス(量	良大8桁 f	(tittitt)
fff8003e		

データサイズは2バイトなので、/データ数に2を設定し、「リード」すると

リードライト ウオッチ ファンクション キーボード アドレス(最大8桁 fmfffff)	
fff8003e	
I/O、メモリリード 2 データ数 最大2桁(ff)	
VO、メモリ ライト データ 最大8桁(fffffff)	
Debug Monitor	
FAST BREAK> R adrs=FFF8003E Data=000A	

DebugMonitor にアドレス FFF8003E=sdata1 のデータ AH = 10D が表示されました。(16 進数表示では A、10 進数表示では 10 の意味)



今度は sdata1 に 1234H を書いてみます。読んでみると確かに 1234 が書き込まれています。

b)ウオッチ

ウオッチコマンドは登録したアドレスをブレークイベントが発生するたびに自動的に読み込み、比較す る機能です。

1)自動読込機能

読み込むデータのアドレス	データサイズ	読込(これとは別にブレークが発生するたびに自動的
		に読み込みます)
コマンド (fff80036 00000002_sdata2 (fff80000 00000000_bss	変数アドレ	
リードライト ウオッチ ファンク アドレス(8桁i) C バイト 「fff80036 C ロングワート	ション キーボー リード ▶ ▽ アクティブ	
自動読込を有効にします		

例として sample6 で、ブレークポイントを 1c54 にセットして「継続スタート」をクリックする度に、

-ブレークボイント設定、動 アドレス ● 1c54 停止中 リセッ	作 解除 クリア ト&スタート 継続スタート
-ソースファイル+アドレス	
1c22: init_debug/	0; //デバック初期設定 ^
: while()	
1c28:	sdata1 = 0:
1c2e:	sdata2 = 10;
1c34:	for(loopm = 0;loopm < 10;loopm++)
1c4a:	1 sdata1++;
1054:	sdata2;
+ C+Address File END	Copyright BRE/2008
<	

読み込むデータのアドレス sdata2 = fff80036 が A、9、8 と減少していくことが確認できます。

1	Debug Monitor
	FAST BREAK> UBC BREAK>
	R adrs=FFF80036 Data=000A UBC BREAK>
	R adrs=FFF80036 Data=0009 UBC BREAK>
	R adrs=FFF80036 Data=0008

1まで減少すると、再び10D(AH)を読み込んで繰り返すのが目視できます。

~

先のリードライトコマンドと比べるとウオッチ機能は 1 回設定すると毎回、ブレーク発生のたびに自動 的に読み込むところが異なります。

2)比較機能



比較条件に合致するとアラームを鳴らします。

比較が終了すると自動的に動作を再開します。連続動作、ブレークイベント監視ができます。

例として1)の条件(sdata2 1~Aまで変化する)で、数値0002でイコールという設定にします。 「継続スタート」の度に Data は変化していきますが、DebugMonitor の中のように Data=0002 のときに WATCH:16bit=HIT!と表示され、比較データと sdata2 が0002 で一致した記録が残ります。また、アラーム が鳴ります。 \



ウオッチ アンド ラン機能:通常、ブレークイベントで UBC BREAK>が表示されると、動作の再開は「継 続スタート」の人手によるクリックが必要です。「ウオッチ アンド ラン」をアクティブにしておくと「継 続スタート」なしで、比較終了後、動作が再開します。無人で連続して比較する場合などに使用します。 変数、ポートのデータ、A / D等アナログ量の比較等、あらゆる事象の解析に使用できます。また、頻度 の少ない現象も確実に捉えることができます。

コンピュータを連続で動かすときは、常にフォース®デバッカの画面が見えている状態にしてください。 コントロールパネル 画面 スクリーンセイバー なし にします。 コントロールパネル 電源オプションのプロパティ 常にオン にします。 WindowsXP の場合

画面のプロパティ ? 🛛	電源オブションのブロバティ		? 🗵
テーマ デスクトップ スクリーン セーバー デザイン 設定	電源設定 詳細設定 休止状態 U	IPS	
	ユノビュータの使い方に最も 変更すると、選択された電 電源設定 (2) (約にオン)	○通した電源設定を選択して、 源設定も変更されます。 名前を付けて(保存(S)	(ださい。下の設定を す) 前)除(<u>D</u>)
	- [常にオン] の電源設定		
	モニタの電源を切る(M):	なし	
スのレーシャーでの (ない) マレビュー(の)	ハード ディスクの電源を切る ①	なし	
(待ち時間())): 10 念 分 □パスワードによる(保護(P)	システム スタンバイ(①)	なし	
モニタ電源	システム休止状態(出):	なし	×.
したい 100月1 をクリックしていたさい。 電源①…			an a
OK キャンセル 適用(A)		OK キャンセノ) 適用(<u>A</u>)

c)ファンクション

現在ファンクションはなにもサポートされていません。



d)キーボード

キーボードはアプリケーションによって、パソコンから CPU ボードへ出力して動作を見たい場合などに 使用します。



例えば sample2 では動作選択に 1,2,3 のいずれかの入力をまっています。例えば「1」をマウスでクリック するとパソコンから CPU ボードに「1」が出力されます。それを受信した CPU ボードは A/D コンバータデ ータを連続して出力するプログラムが動作します。



Debug Monitor	
AD1=06C1 AD2=06C1	^
AD1=06C1 AD2=06C1 AD1=06C1 AD2=06C1	
AD1=06C1 AD2=06C1	
AD1=06C1 AD2=06C1 AD1=06C1 AD2=06C1	
AD1=06C1 AD2=06C1	
	~

このような無限ループプログラムの初期化は「リセット&スタート」で行います。

1-8 タイムスタンプ、デバック状況ファイル出力

D	Debug Monitor	
4 4 1	AD1=06C1 AD2=06C1 AD1=06C1 AD2=06C1 13:13:25 ⊾	
A A A	AD1=06C1\AD2=06C1 AD1=06C1 AD2=06C1 AD1=06C1 AD2=06C1	
f	AD1=06C1 AD2=06C1	•
	画面クリア マタイムスタンラ ファー	
画面	面クリア タイムスタンプ デバッ	グモニタ画面ファイル出力

画面クリア	: DebugMonitor 画面をクリアします。
タイムスタンプ	:約1分ごとに DebugMonitor 画面に現在の時刻を入れます。
ファイル出力	:DebugMonitor 画面の内容をファイルにセープします。

例えば、log.txt という名前でセーブできます。(お好きな名前、拡張子が付けられます)

名前を付けて保存				? 🔀
保存する場所①:	🚞 sample2		+ 🗈 🖻	* 📰 -
 compile.txt log.txt sample2.BAK sample2.c sample2.elf sample2.lst 	 sample2.map sample2.mot sample2.o sh7243.h SH7243_rom.BAK SH7243_rom.x 	▶ sh_start0.o ▶ sh_start0.s ₿ SIO_SH7243.h		
ファイル名(11):	log.txt			保存(S)
ファイルの種類(工):			•	キャンセル

また、この窓は書き込みも可能で、デバック中のコメント(下例「これはデータの異常である。 2008.10.17」)をユーザーが書き加えることが可能です。



このログファイルにはデバックで行ったこと、時刻(タイムスタンプオン時)、ウオッチ比較のヒットあ り、なし、コメント等すべてが記録されています。通常のアスキーファイルですので、プリンターで印刷 したり、エディタで検索等掛けられます。

プログラム作成で最後まで残りやすい、頻度の少ない、稀な現象の強力な原因究明ツールになるよう考 案されています。

WindowsXP、VISTA はマイクロソフト社の登録商標です。

フォース®および FORCE®は弊社の登録商標です。

1.本文章に記載された内容は弊社有限会社ビーリバーエレクトロニクスの調査結果です。

2.本文章に記載された情報の内容、使用結果に対して弊社はいかなる責任も負いません。

3.本文章に記載された情報に誤記等問題がありましたらご一報いただけますと幸いです。

4.本文章は許可なく転載、複製することを堅くお断りいたします。

〒350-1213 埼玉県日高市高萩1141-1

TEL 042 (985) 6982

FAX 042 (985) 6720

Homepage:http://beriver.co.jp e-mail:support@beriver.co.jp

有限会社ビーリバーエレクトロニクス ©Beyond the river Inc 20081017