HEW Tcl/Tk 実用例

アプリケーションノート

R20AN0069JJ0100
Rev.1.00
2011.01.11

本ドキュメントでは、統合開発環境(High-performance Embedded Workshop)の Tcl/Tk 拡張機能の使用方法について、サンプルプログラムを用いて説明します。

目次

1. はじめに

1.1. HEW における Tcl/Tk の概要

統合開発環境 High-performance Embedded Workshop（以下 HEW という）では、拡張機能としてスクリプト言語 Tcl/Tk をサポートしています。

Tcl/Tk は、スクリプト言語であるクラスメンバ“Tcl(Tool Command Language の略称)”と、グラフィカルユーザインタフェース(GUI)をプログラミングするためのツールキット“Tk”で構成されているプログラミング言語です。Tcl/Tk を用いて作成したプログラムは、コンパイルせずに利用することができます。

HEW 上で動作する Tcl/Tk を使用してプログラミングすることで、使用方法に合わせて HEW をカスタマイズすることができます。HEW の使用例としては以下のようなものがあります。

①Tcl/Tk で作成したボタンなどの GUI に HEW のコマンドを割り当てる

②Tcl/Tk で作成した GUI から HEW に対してコマンドを発行する

③HEW 上のシミュレーションの制御

本アプリケーションノートでは、Tcl/Tk と HEW 間での情報のやり取りや、HEW の実行環境の制御などを行うサンプルプログラムについて説明します。

なお、HEW の Tcl/Tk のプログラミング方法については、「Tcl/Tk アプリケーションノート」および HEW マニュアルに含まれる「Tcl/Tk 添付資料」を参照してください。また、HEW のコマンドについては HEW メニューバー「ヘルプ->トピック」から、シミュレータのコマンドについては HEW メニューバー「ヘルプ->シミュレータヘルプ」から、それぞれ表示されるヘルプのコマンドラインの章を参照してください。

1.2. 動作確認環境

本アプリケーションノートのサンプルプログラムは、以下の環境で確認しています。

表 1　動作確認環境

プログラム名	バージョン
H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ	V.7.00 Release 00
High-performance Embedded Workshop	V.4.08.00
H8SX,H8S,H8 ファミリ シミュレータデバッガ	V.5.08.00
Tcl/Tk for Windows	V.8.4.4

なお、H8SX,H8S,H8 ファミリ用 C/C++コンパイラパッケージ V.7.00 Release 00 を使用する場合は、HEW のバージョンが V.4.06.00 なので、HEW を V.4.08.00 以上にアップデートしてご使用ください。

2. Tcl/Tk 実用例

Tcl/TkからHEW環境を制御する実用例として、表 2に示す３種類のサンプルプログラムを使用して説明します。なお、以下の説明において、各サンプルプログラムのZIPファイルを“C:¥WorkSpace”に展開していることを前提としています。また、以降の説明において、“ターゲット”とは実機（マイコン）またはシミュレータのことを示し、“ターゲットプログラム”とは実機またはシミュレータ上で動作するプログラムのことを示します。

表 2　サンプルプログラム一覧

サンプル名称	機能概要	ワークスペース名
LED サンプル	ターゲットの LED の点灯状態を視覚的に表示するサンプル	LED_sample
LCD サンプル	ターゲットの LCD 制御レジスタの状態や LCD の表示内容を視覚的に表示するサンプル	LCD_sample
メモリ検証サンプル	モトローラ S 形式ファイルのデータをターゲットにダウンロードし、さらに、正しくダウンロードできたか検証するサンプル	MemoryVerify_sample

2.1. LED サンプル

LED サンプルは、ターゲットの I/O ポートの値を Tcl/Tk で読み込み、I/O ポートに割り当てられた LED の点灯状態を視覚的に表示するサンプルです。LED の制御を含むターゲットプログラムをシミュレータで動作確認する場合に便利です。なお本サンプルで使用するワークスペースは、H8/38024F を対象に作成しています。

(1) 使用方法

LED サンプルの使用方法について説明します。

①HEW ワークスペースのオープン

“C¥WorkSpace¥LED_sample¥LED_sample.hws”を HEW で開きます。

②ターゲットプログラムのダウンロード

ターゲットプログラム（LED_sample.abs）をダウンロードします。

③LED スクリプトファイルの読み込み

HEWメニューバー[表示->TCLツールキット]よりTCLツールキットを立ち上げ、LED用のスクリプトファイル“C:¥WorkSpace¥LED_sample¥LED_sample.tcl”を開きます。図 1は、LEDサンプルの起動直後の画面です。

図 1 LED サンプルのトップレベルウィンドウ

④LED サンプルの設定

次に、LED-1 にLEDの設定をします。LED-1 の画像を左クリックすると、「LED-1 setting」の設定ダイアログが表示されます。ターゲットプログラムでは、0xffdc番地のI/Oポートにおいて、最下位ビットから 3 ビット目の値が 1 の状態（16 進数で 0x04 に相当）をLEDの点灯状態に割り当てています。そこで、「LED-1 setting」には以下の値を入力します。値を設定したあとのLED設定ダイアログを図 2 に示します。

- Address (in hex) … ffdc
- Value to turn on LED (in hex) … 04

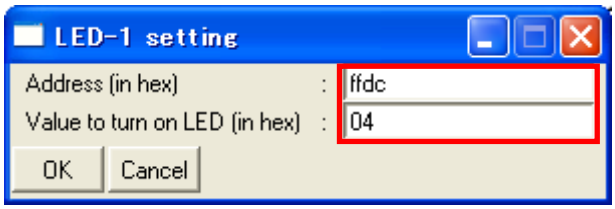


図 2　LED 設定ダイアログ（LED-1 setting）

⑤ターゲットのリセットと実行

HEW メニューバー[デバッグ->リセット後実行]を選択します。

⑥LED サンプルの実行と停止

トップレベルウィンドウで「Start」をクリックすると、LED-1 が点灯と消灯を繰り返します。また、「Stop」で実行を停止します。

図 3　LED-1 点灯時のトップレベルウィンドウ

(2) ターゲットプログラムの説明

ターゲットプログラムについて説明します。ターゲットプログラムでは、I/O ポートに割り当てられた LED の点灯状態を一定時間ごとに反転します。

①I/O ポートの定義

LED 用の I/O ポートを定義します。サンプルプログラムでは、ポート 9（0xffdc 番地）の最下位ビットから 3 ビット目を LED に割り当てており、P92 として定義しています。ここで、P92 の値が 0 の場合は LED の消灯状態を示し、値が 1 の場合は LED の点灯状態を示します。

```
13:#include "iodefine.h"
14:
15:#define P92 IO.PDR9.BIT.B2
```

②メイン処理

ターゲットプログラムのメイン処理について説明します。始めに LED を消灯したあと、一定時間ごとに P92 の値を反転することにより、LED の点灯と消灯を繰り返します。なお、本サンプルでは、I/O ポートの入出力設定などの初期化は行っていません。必要に応じて I/O ポートの初期化処理を追加してください。

```
27:void main(void)
28:{
29:    P92 = 0;
30:
```

```
31:    while(1) {
32:        wait();
33:        P92 = ~P92;
34:    }
35:}
```

(3) Tcl/Tk スクリプトの説明

Tcl/Tk スクリプトについて説明します。LED サンプルの Tcl/Tk スクリプトには、以下の処理が含まれています。メイン処理において、ターゲットの I/O ポートの状態を一定時間ごとに監視しながら、LED の点灯状況をトップレベルウィンドウに表示します。

- トップレベルウィンドウの作成
- LED 設定ダイアログの作成と設定内容の確認
- メイン処理

①トップレベルウィンドウの作成

下記の処理により、図 1に示したトップレベルウィンドウを作成します。

```
  1:####################################
  2:#LED Simulator
  3:wm protocol . WM_DELETE_WINDOW exit
  4:image create photo on -file redon.gif
  5:image create photo off -file redoff.gif
  6:set stop 0
  7:
  8:for {set i 1} {$i<=5} {incr i} {
  9:    frame .f$i -bd 2 -relief groove
 10:    pack .f$i -side left -padx 2 -pady 4 -fill both -expand 1
 11:    label .f$i.led -image off
 12:    pack .f$i.led -side top
 13:    label .f$i.label -text LED-$i
 14:    pack .f$i.label -side top -anchor center
 15:}
 16:
 17:frame .f6
 18:pack .f6 -pady 4 -padx 2
 19:button .f6.b1 -text Start -font *-times-normal-r-*-12-* -width 8 -command {testLED}
 20:button .f6.b2 -text Stop -font *-times-normal-r-*-12-* -width 8 -command {set stop 1}
 21:pack .f6.b1 .f6.b2 -side top -pady 2 -fill both -expand 1
 22:
 23:bind .f1.led <Button-1> {assign 1}
 24:bind .f2.led <Button-1> {assign 2}
 25:bind .f3.led <Button-1> {assign 3}
 26:bind .f4.led <Button-1> {assign 4}
 27:bind .f5.led <Button-1> {assign 5}
 28:
 29:foreach j {1 2 3 4 5} {set assigned($j) 0 ; set label($j) "LED-$j"}
```

②LED 設定ダイアログの作成と設定内容の確認

下記の処理により、図 2に示したLEDの設定ダイアログを作成します。トップレベルウィンドウでLEDの画像をクリックすると、対応するLEDのダイアログ作成処理が呼ばれます。

```
 31:####################################
 32:#display the "assign value" window
 33:proc assign {n} {
 34:
 35:    global assigned
 36:    global address
```

```
37:     global value
38:     global label
39:
40:     toplevel .t
41:     wm resizable .t 0 0
42:     wm title .t "LED-$n setting"
43:     frame .t.up
44:     pack .t.up -fill both -expand 1
45:     frame .t.low
46:     pack .t.low -fill both -expand 1
47:
48:     frame .t.up.f1
49:     pack .t.up.f1 -fill both -expand 1 -side left
50:     label .t.up.f1.l1 -text "Address (in hex)"
51:     label .t.up.f1.l2 -text "Value to turn on LED (in hex)"
52:     foreach b {1 2} {pack .t.up.f1.l$b -side top -anchor nw -padx 5}
53:     frame .t.up.f2
54:     pack .t.up.f2 -fill both -expand 1 -side left
55:     foreach b {1 2} {
56:         label .t.up.f2.l$b -text ":"
57:         pack .t.up.f2.l$b -side top -anchor nw
58:     }
59:
60:     frame .t.up.f3
61:     pack .t.up.f3 -fill both -expand 1 -side left
62:     foreach b {1 2} {
63:         entry .t.up.f3.e$b -textvariable e$b
64:         .t.up.f3.e$b delete 0 end
65:         pack .t.up.f3.e$b -side top -anchor nw -padx 5
66:     }
67:
68:     catch {
69:         .t.up.f3.e1 insert end $address($n)
70:         .t.up.f3.e2 insert end $value($n)
71:     }
72:
73:     global numb
74:
75:     set numb $n
76:
77:     button .t.low.b1 -text OK -width 6 -command {setValues $numb $e1 $e2}
78:     button .t.low.b2 -text Cancel -width 6 -command {destroy .t}
79:     bind .t.low.b1 <Key-Return> {setValues $numb $e1 $e2}
80:     foreach m {1 2} {bind .t.up.f3.e$m <Key-Return> {setValues $numb $e1 $e2}}
81:     bind .t.low.b2 <Key-Return> {destroy .t}
82:
83:     pack .t.low.b1 .t.low.b2 -side left -fill both -anchor center -padx 1 -pady 3
84:}
```

図 2の設定ダイアログで「OK」をクリックすると、下記の処理が呼ばれます。この処理により、設定ダイアログに入力された設定値を保存します。I/Oポートのアドレスや LED点灯時の値を入力していない場合、入力を促すエラーメッセージを出力します。

```
86:##########################################
87:#set the value for the LED variables
88:proc setValues {numb e1 e2} {
89:
90:     global address
91:     global value
92:     global assigned
93:
94:     if {$e1=="" || $e2==""} {
95:         tk_messageBox -message "Please fill the address and value fields!" -icon warning ; return
96:     }
97:
98:     set assigned($numb) 1
```

```
 99:    set address($numb) $e1
100:    set value($numb) [string toupper $e2]
101:
102:    destroy .t
103:}
```

③メイン処理

トップレベルウィンドウの「Start」をクリックすると、下記のメイン処理が呼ばれます。ターゲットから I/O ポートの値を読み込み、LED の点灯状態に応じて、トップレベルウィンドウに表示する LED の画像を切り替えます。このとき、HEW の“md（memory_display）”コマンドを使用してターゲットから I/O ポートの値を読み込みます。この処理を、トップレベルウィンドウの「Stop」がクリックされるまで繰り返します。

```
105:###############################################
106:# to test whether LED should on or off
107:proc testLED {} {
108:
109:    global stop
110:    global env
111:    global assigned
112:    global address
113:    global value
114:    global label
115:
116:    set stop 0
117:
118:    while {!$stop} {
119:        foreach led { 1 2 3 4 5 } {
120:            if {$assigned($led)} {
121:                .f$led.label configure -text $label($led)
122:                set fd [open $env(TMP)/log.txt {RDWR TRUNC CREAT}]
123:                close $fd
124:
125:                if [catch {puts [md h'$address($led) H'1]}] {
126:                    tk_messageBox -icon error -message "Error in getting memory content.
                        Please  make sure your entry is valid and device is set properly!" ; return
127:                }
128:
129:                set fd [open $env(TMP)/log.txt {RDWR}]
130:                set b ""
131:                set formatted [format "%08X" 0x$address($led)]
132:
133:                seek $fd 0 start
134:
135:                while {$b!=$formatted} {
136:                    set e [gets $fd]
137:                    set d [split $e ""]
138:                    set b ""
139:
140:                    for {set j 0} {$j<8} {incr j} {append b [lindex $d $j]}
141:                }
142:
143:                set e [split $e]
144:
145:                close $fd
146:
147:                if {[lindex $e 2]==$value($led)} {
148:                    .f$led.led configure -image on
149:                } else {
150:                    .f$led.led configure -image off
151:                }
152:            }
153:        }
```

I/O ポートの値読み込み

点灯状態に応じた LED 画像の切り替え

```
154:        update
155:
156:        sleep (500)
157:    }
158:}
```

2.2. LCD サンプル

LCD サンプルは、ターゲットの LCD 制御レジスタの値を Tcl/Tk で読み込み、LCD 制御レジスタの値と、それに対応した LCD の状態を視覚的に表示するサンプルです。LCD の制御を含むターゲットプログラムをシミュレータで動作確認する場合に便利です。なお本サンプルで使用するワークスペースは、H8/38024F を対象に作成しています。

(1) 使用方法

LCD サンプルの使用方法について説明します。

①HEW ワークスペースのオープン

“C¥WorkSpace¥LCD_sample¥LCD_sample.hws”を HEW で開きます。

②ターゲットプログラムのダウンロード

ターゲットプログラム（LCD_sample.abs）をダウンロードします。

③LCD スクリプトファイルの読み込み

HEWメニューバー[表示->TCLツールキット]よりTCLツールキットを立ち上げ、LCD用のスクリプトファイル“C:¥WorkSpace¥LCD_sample¥LCD_sample.tcl”を開きます。図 3は、LCDサンプルの起動直後の画面です。

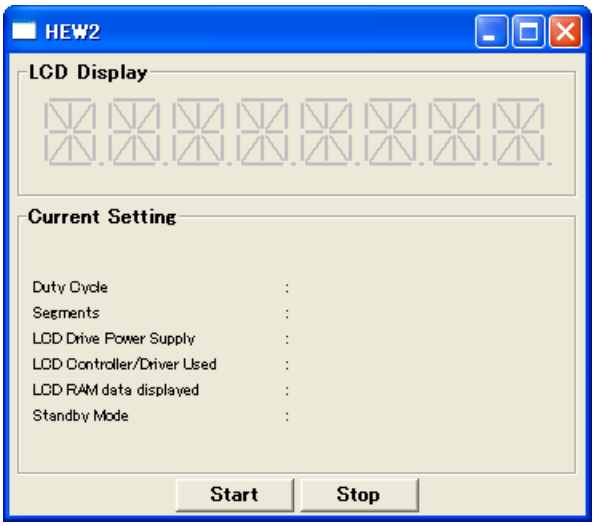


図 3 LCD サンプルのトップレベルウィンドウ

④ターゲットのリセットと実行

HEW メニューバー[デバッグ->リセット後実行]を選択します。

⑤LCD サンプル(Tcl)の実行と停止

トップレベルウィンドウで「Start」をクリックすると、図 4のようにLCDの状態表示を開始します。「LCD Display」にはLCDの表示内容を描画し、「Current Setting」にはLCD制御レジスタの状態を表示

します。また、「Stop」で処理を停止します。

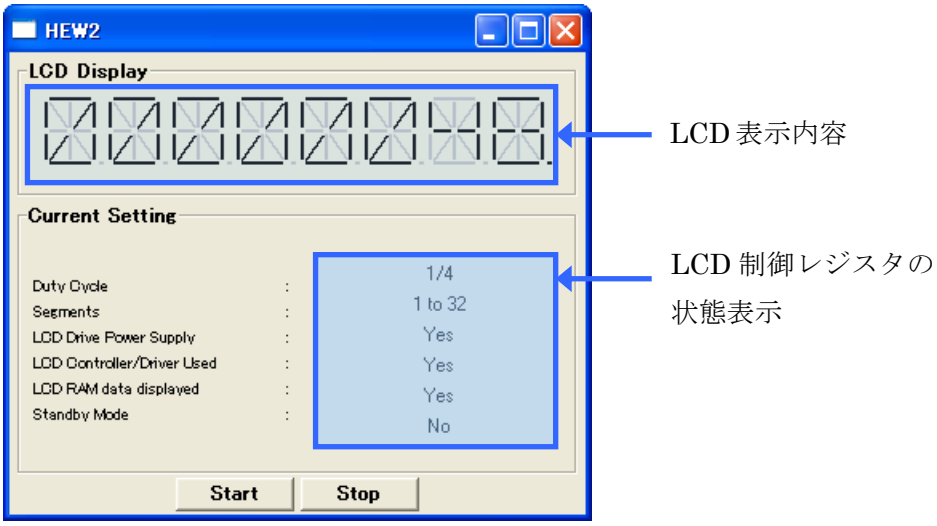


図 4　LCD サンプルの実行画面

(2) ターゲットプログラムの説明

ターゲットプログラムについて説明します。ターゲットプログラムでは、始めに LCD 制御レジスタを初期化したあと、整数型のカウンタをカウントアップしながら、そのカウンタの値を LCD に表示します。なお、本サンプルでは、LCD 以外のレジスタの初期化は行っていません。必要に応じてその他の初期化処理を追加してください。

①LCD データ領域の定義

LCD_RAM は、LCD のパターンデータを格納するためのデータ領域です。

```
12:#include "iodefine.h"
13:
14:#define LCD_RAM     ((unsigned short*)0xF740)
```

②LCD パターンデータの定義

下記は、LCD RAM に格納する、LCD のパターンデータを定義しています。英数字と記号、スペースを含む 1 文字ごとに 1 つのパターンが割り当てられます。本アプリケーションノートでは、LCD のパターンデータやその書式の説明は省略します。

```
16://for 1/4 duty cycle
17:const unsigned short    lcd_data[] ={
18:     0xE724, //0. '0'
19:     0x0600, //1. '1'
20:     0xC342, //2. '2'
21:     0x8742, //3. '3'
22:     0x2642, //4. '4'
23:     0xA542, //5. '5'
24:     0xE542, //6. '6'
25:     0x0700, //7. '7'
26:     0xE742, //8. '8'
27:     0x2742, //9. '9'
28:     0x00E7, //10. '*'
29:     0x0000, //11. Blank: All segments OFF
30:     0x6742, //12. 'A'
31:     0xE442, //13. 'B'
```

```
 32:     0xE100, //14. 'C'
 33:     0xC642, //15. 'D'
 34:     0xE142, //16. 'E'
 35:     0x6142, //17. 'F'
 36:     0xE540, //18. 'G'
 37:     0x6642, //19. 'H'
 38:     0x8118, //20. 'I'
 39:     0xC600, //21. 'J'
 40:     0x60A2, //22. 'K'
 41:     0xE000, //23. 'L'
 42:     0x6621, //24. 'M'
 43:     0x6681, //25. 'N'
 44:     0xE700, //26. 'O'
 45:     0x6342, //27. 'P'
 46:     0xE780, //28. 'Q'
 47:     0x63C2, //29. 'R'
 48:     0xA542, //30. 'S'
 49:     0x0118, //31. 'T'
 50:     0xE600, //32. 'U'
 51:     0x0681, //33. 'V'
 52:     0x6684, //34. 'W'
 53:     0x00A5, //35. 'X'
 54:     0x0029, //36. 'Y'
 55:     0x8124, //37. 'Z'
 56:};
```

③LCD の初期化

LCD RAM のクリアと、LCD 制御レジスタの初期化をします。

```
 58:/*
 59:**  init_lcd()
 60:**  a.   clear LCD RAM
 61:**  b.   set to 1/4 duty, SEG1 to SEG32, frame frequency = 30.5, waveform A
 62:*/
 63:void init_lcd(void)
 64:{
 65:     unsigned char   temp_a;
 66:     unsigned char   *dest;
 67:
 68:     //clear LCD RAM
 69:     dest = (unsigned char *)LCD_RAM;
 70:     for (temp_a = 0 ; temp_a < 16 ; temp_a++)
 71:     {
 72:         *dest++ = 0;
 73:     }
 74:
 75:     //CKSTPR2 : |---|---|---|PW2CKSTP|AECKSTP|---|PW1CKSTP|LDCKSTP|
 76:     CKSTPR2.BYTE = 0xFF;
 77:
 78:     //LPCR : |DTS1|DTS0|CMX|---|SGS3|SGS2|SGS1|SGS0| : |1|1|0|0|0|1|1|0|
 79:     //|DTS1|DTS0| = |1|1| : 1/4 duty
 80:     //|CMX | = 0
 81:     //Bit 4 is reserved; only 0 can be written to this bit
 82:     //|SGS3|SGS2|SGS1|SGS0| = |0|1|1|0| : Use SEG1 to SEG24
 83:     LCD.LPCR.BYTE = 0xC8;   //1/4 duty cycle
 84:
 85:     //LCR : |---|PSW|ACT|DISP|CKS3|CKS2|CKS1|CKS0| : |1|1|1|1|1|1|1|1|
 86:     //Bit 7 is reserved; always read as 1 and cannot be modified
 87:     //PSW = 1 : LCD drive power supply on
 88:     //ACT = 1 : LCD controller/driver operates
 89:     //DISP = 1 : LCD RAM data is displayed
 90:     LCD.LCR.BYTE = 0xFF;    //display is faint
 91:
 92:     //LCR2 : |LCDAB|---|---|---|---|---|---|---|
 93:     //LCDAB : 0 : drive using A waveform
 94:     //Bits 6 and 5 are reserved; always read as 1 and cannot be modified
```

LCD RAM のクリア

LCD 制御レジスタの初期化

```
 95:     //Bits 4 to 0 are reserved; only 0 can be written to these bits
 96:     LCD.LCR2.BYTE = 0x60;
 97:}
```

ここで、LCD 制御レジスタには下記の設定をします。

表 3 LCD 制御レジスタの設定

設定名	設定値
デューティ比	1/4
セグメント選択	SEG1 to SEG32
LCD 駆動電源制御	電源 ON
LCD コントローラ／ドライバの動作制御	動作 ON
LCD 表示データ制御	LCD RAM データ使用
LCD 波形選択	A 波形
スタンバイモード	解除（LCD 表示可能）

④LCD の表示

下記の処理により、1 桁分の数字のパターンデータを LCD RAM に書き込みます。decimal_point に 0 以外の値を渡すと、LCD の表示桁に小数点を表示します。

```
119:/*
120:**  display_number(position, number, decimal point)
121:**
122:**  position : 0 to 7
123:**  number : 0 to 9
124:**  decimal_point = 1 : display decimal point
125:*/
126:void display_number(unsigned char position, unsigned char number, unsigned char decimal_point)
127:{
128:    unsigned short  *dest;
129:
130:    if (position > 7) {
131:        return;
132:    }
133:    if (number > 9) {
134:        return;
135:    }
136:
137:    dest = &LCD_RAM[position];
138:
139:    if (decimal_point) {
140:        *dest = (unsigned short)(lcd_data[number] | 0x0800);
141:    } else {
142:        *dest = lcd_data[number];
143:    }
144:}
```

下記の処理により、8 桁分の数字のパターンデータを LCD RAM に書き込みます。右端の数字（1 桁目）には小数点を表示します。

```
146:/*
147:**  display_lcd()
148:*/
```

```
149:void display_lcd(unsigned long number)
150:{
151:    int i;
152:
153:    for (i=0; i<8; i++) {
154:        if (i==0) {
155:            display_number(i, number % 10, 1);
156:        } else {
157:            display_number(i, number % 10, 0);
158:        }
159:        number /= 10;
160:    }
161:}
```

⑤メイン処理

メイン処理について説明します。始めに LCD 制御レジスタを初期化します。その後、整数型のカウンタ（変数 number）をカウントアップしながら、カウンタの値を LCD に表示します。なお、LCD の表示桁が 8 桁であるため、カウンタの値が 8 桁を超えないようにオーバフローチェックしています。

```
163:void main(void)
164:{
165:    unsigned long number = 0;
166:
167:    init_lcd();
168:
169:    while (1) {
170:        display_lcd(number);
171:        delay(10);
172:
173:        number++;
174:        if (number == 100000000UL) {
175:            number = 0;
176:        }
177:    }
178:}
```

(3) Tcl/Tk スクリプトの説明

Tcl/Tk スクリプトについて説明します。LCD サンプルの Tcl/Tk スクリプトには、以下の処理が含まれます。メイン処理において、ターゲットの LCD 制御レジスタの値を一定時間ごとに監視しながら、LCD の表示状況をトップレベルウィンドウに描画します。

- LCD 描画パターンの定義
- トップレベルウィンドウの作成
- LCD 制御レジスタの読み込み
- LCD の描画処理
- メイン処理

①LCD 描画パターンの定義

LCD の描画パターンを定義します。トップレベルウィンドウに LCD を描画するときの LCD の各セグメントの描画座標や、LCD に英数字などを表示するときに点灯させるセグメントを定義しています。本アプリケーションノートでは、LCD のパターンデータやその書式の説明は省略します。

```
  6:##############################################################################
  7:#Start of LCD Simulator code
```

```
  8:
  9:set LPCR 0xFFC0
 10:set LCR 0xFFC1
 11:set LCR2 0xFFC2
 12:set LCD_RAM 0xF740
 13:set CKSTPR2 0xFFFB
 14:set Static [list a g h]
 15:
 16:set seg1to4 0
 17:set seg5to8 0
 18:set seg9to12 0
 19:set seg13to16 0
 20:set seg17to20 0
 21:set seg21to24 0
 22:set seg25to28 0
 23:set seg29to32 0
 24:set segText ""
 25:set PSW 0
 26:
 27:for {set i 0} {$i<8} {incr i} {
 28:    set ram_val($i) 0
 29:}
 30:
 31:
 32:# The shapes of individual elements of a digit
 33:array set lcdshape {
 34:    a {12 10 38 10 }
 35:    b {40 12 40 28}
 36:    c {40 32 40 48}
 37:    d {12 50 38 50}
 38:    e {10 32 10 48}
 39:    f {10 12 10 28}

～省略～

176:    W {a d g h i j k m o}
177:    x {a b c d e f h j k m o}
178:    X {a b c d e f h j k m o}
179:    y {a b c d e f h j k l n o}
180:    Y {a b c d e f h j k l n o}
181:    z {b c e f g h j k m n o}
182:    Z {b c e f g h j k m n o}
183:    * {a b c d e f h m o}
184:    { } {a b c d e f g h i j k l m n o}
185:}
```

②トップレベルウィンドウの作成

下記の処理により、図 3に示したトップレベルウィンドウを作成します。

```
188:#################################################################################
189:# Proc        : showLCD
190:# Arg         : width, default is 8 digit
191:# Return: -
192:# Function: Displays the unfilled LCD
193:
194:proc showLCD {{width 8}} {
195:    global llcd ulcd lcdshape
196:    set lcdoffset 0
197:    .l.c delete lcd
198:    for {set i 0} {$i<8} {incr i} {
199:        foreach symbol $ulcd( ) {
200:            .l.c move [eval .l.c create line $lcdshape($symbol) -fill grey -width 2 -tags lcd
¥
201:                ] $lcdoffset 0
202:        }
203:        incr lcdoffset 40
```

LCD の描画（全て消灯状態）

```
204:    }
205:}
206:
207:
208:
209:labelframe .l -text "LCD Display" -font *-ansi-bold-r-*-12-*
210:pack .l -side top -fill both -expand 1 -ipadx 5 -ipady 5 -pady 5 -padx 5
211:canvas .l.c -height 55 -width 330
212:
213:labelframe .l2 -text "Current Setting" -font *-ansi-bold-r-*-12-*
214:pack .l2 -side top -fill both -expand 1  -padx 5
215:foreach j {1 2 3} {frame .l2.f$j ; pack .l2.f$j -side left -fill x -expand 1}
216:pack .l2.f2 -side left -expand 0
217:label .l2.f1.l0 -text "          "
218:label .l2.f1.l1 -text "Duty Cycle" -font *-ansi-normal-r-*-10-*
219:label .l2.f1.l2 -text "Segments" -font *-ansi-normal-r-*-10-*
220:label .l2.f1.l3 -text "LCD Drive Power Supply" -font *-ansi-normal-r-*-10-*
221:label .l2.f1.l4 -text "LCD Controller/Driver Used" -font *-ansi-normal-r-*-10-*
222:label .l2.f1.l5 -text "LCD RAM data displayed" -font *-ansi-normal-r-*-10-*
223:label .l2.f1.l6 -text "Standby Mode" -font *-ansi-normal-r-*-10-*
224:label .l2.f1.l7 -text "          "
225:foreach j {0 1 2 3 4 5 6 7} {pack .l2.f1.l$j -side top -padx 5 -anchor w}
226:
227:label .l2.f2.l0 -text "           "
228:pack .l2.f2.l0 -side top  -padx 5
229:
230:foreach j {1 2 3 4 5 6} {
231:    label .l2.f2.l$j -text "          :" -font *-ansi-normal-r-*-10-*
232:    pack .l2.f2.l$j -side top  -padx 5
233:}
234:
235:label .l2.f2.l7 -text "           "
236:pack .l2.f2.l7 -side top  -padx 5
237:
238:for {set i 0 } {$i<=7} {incr i} {
239:    label .l2.f3.l$i -text "                                                    "
240:    pack .l2.f3.l$i -side top  -padx 5
241:}
242:
243:frame .f
244:pack .f -side top
245:set stop 0
246:button .f.start -text "Start" -command start -width 8 -font *-ansi-bold-r-*-12-*
247:button .f.stop -text "Stop" -command {set stop 1} -width 8 -font *-ansi-bold-r-*-12-*
248:pack .l.c
249:pack .f.start .f.stop -side left -anchor center  -padx 2 -pady 2
250:
251:showLCD
```

トップレベルウィンドウの作成

③LCD 制御レジスタの読み込み

下記の処理により、LCD 制御レジスタ（LPCR）からデューティ比を読み込みます。このとき、HEW の“md（memory_display）”コマンドを使用して、ターゲットから LCD 制御レジスタの値を読み込みます。

```
294:#################################################################################
295:# Proc        : checkDutyCycle
296:# Arg         : -
297:# Return: -
298:# Function: Check duty cycle
299:
300:proc checkDutyCycle {} {
301:    global LPCR
302:
303:    set fd [open $::env(TMP)/log.txt {RDWR TRUNC}]
304:    close $fd
```

LPCR レジスタの値の読み込み

```
305:
306:    puts [MD $LPCR 1]
307:
308:    set fd [open $::env(TMP)/log.txt RDWR]
309:    set b ""
310:    seek $fd 0 start
311:    set formatted [format "%08X" $LPCR]
312:    while {$b!=$formatted} {
313:        set e [gets $fd]
314:        set d [split $e ""]
315:        set b ""
316:        for {set j 0} {$j<8} {incr j} {append b [lindex $d $j]}
317:    }
318:    set ar [split $e]
319:    set content [lindex $ar 2]
320:
321:    global dutyCycle
322:    set r [format "%02X" [expr 0x$content & 0xc0 ] ]
323:    switch -- $r {
324:        00 {set dutyCycle Static}
325:        40 {set dutyCycle "1/2"}
326:        80 {set dutyCycle "1/3"}
327:        C0 {set dutyCycle "1/4"}
328:    }
329:    .l2.f3.l1 configure -fg black -text $dutyCycle
330:    if {$dutyCycle!="1/4"} {.l2.f3.l1 configure -fg red}
331:    close $fd
332:    return 1
333:}
```

デューティ比の取得

トップレベルウィンドウの更新

同様に checkSegment()、checkPSW()、checkACT()、checkDISP()、checkStbyMode()により、ターゲットからセグメント設定、LCD 駆動電源制御の状態、LCD コントローラ／ドライバ動作制御の状態、LCD 表示データ制御の状態、スタンバイモードの状態を読み込みます。

④LCD の描画処理

下記は、ターゲットから 8 桁分の LCD RAM の値を読み込む処理です。LCD 制御レジスタと同様、HEW の md コマンドを使用します。読み込んだ LCD RAM の値は、変数 ram_val に格納します。

```
670:############################################################
671:# Proc         : read_LCD
672:# Arg         : -
673:# Return: -
674:# Function: read the LCD RAM
675:
676:proc read_LCD {} {
677:    global LPCR
678:    global LCR
679:    global LCR2
680:    global LCD_RAM
681:    global env
682:    global ram_val
683:
684:    #clear the console log file
685:    set fd [open $env(TMP)/log.txt {RDWR TRUNC CREAT}]
686:    close $fd
687:
688:    #read content of every digit in respective LCD_RAM address
689:    for {set i 0} {$i<8} {incr i} {
690:        set formatted [format "%08X" [expr (2*$i)+$LCD_RAM]]
691:        if [catch {puts [MD H'$formatted H'2 word]}] {tk_messageBox -icon error -title HEW
            -message "Error in getting memory content. Please make sure your entry is valid and
            device is set properly!" ; return}
```

LCD RAM データ読み込み

```
692:
693:        set fd [open $env(TMP)/log.txt {RDWR}]
694:        set b ""
695:        seek $fd 0 start
696:        while {$b!=$formatted} {
697:            set e [gets $fd]
698:            set d [split $e ""]
699:            set b ""
700:            for {set j 0} {$j<8} {incr j} {append b [lindex $d $j]}
701:        }
702:        close $fd
703:        set ram_val($i) [lindex $e 1]
704:    }
705:}
```

読み込んだデータを ram_val に格納

下記は、トップレベルウィンドウの LCD を再描画する処理です。上記の処理により読み込んだ LCD RAM のパターンデータをもとに、トップレベルウィンドウに LCD の表示状態を再描画していきます。

```
581:#################################################################################
582:# Proc        : display
583:# Arg         : -
584:# return: -
585:# Function: get which side(s) of the LCD are on
586:
587:proc display {} {
588:
589:    global ram_val
590:    global onLCD
591:
592:    read_LCD
593:
594:    for {set i 0} {$i<8} {incr i} {set onLCD($i) ""}
595:
596:    for {set i 0} {$i<8} {incr i} {
597:        if {[expr 0x0100 & 0x$ram_val($i)]=="0x0100"} {lappend onLCD($i) "a"}
598:        if {[expr 0x0200 & 0x$ram_val($i)]=="0x0200"} {lappend onLCD($i) "b"}
599:        if {[expr 0x0400 & 0x$ram_val($i)]=="0x0400"} {lappend onLCD($i) "c"}
600:        if {[expr 0x8000 & 0x$ram_val($i)]=="0x8000"} {lappend onLCD($i) "d"}
601:        if {[expr 0x4000 & 0x$ram_val($i)]=="0x4000"} {lappend onLCD($i) "e"}
602:        if {[expr 0x2000 & 0x$ram_val($i)]=="0x2000"} {lappend onLCD($i) "f"}
603:        if {[expr 0x0001 & 0x$ram_val($i)]=="0x0001"} {lappend onLCD($i) "g"}
604:        if {[expr 0x0010 & 0x$ram_val($i)]=="0x0010"} {lappend onLCD($i) "h"}
605:        if {[expr 0x0020 & 0x$ram_val($i)]=="0x0020"} {lappend onLCD($i) "i"}
606:        if {[expr 0x0002 & 0x$ram_val($i)]=="0x0002"} {lappend onLCD($i) "j"}
607:        if {[expr 0x0040 & 0x$ram_val($i)]=="0x0040"} {lappend onLCD($i) "k"}
608:        if {[expr 0x0004 & 0x$ram_val($i)]=="0x0004"} {lappend onLCD($i) "l"}
609:        if {[expr 0x0008 & 0x$ram_val($i)]=="0x0008"} {lappend onLCD($i) "m"}
610:        if {[expr 0x0080 & 0x$ram_val($i)]=="0x0080"} {lappend onLCD($i) "n"}
611:        if {[expr 0x0800 & 0x$ram_val($i)]=="0x0800"} {lappend onLCD($i) "o"}
612:    }
613:
614:    display_LCD
615:}
616:
617:#############################################################
618:# Proc        : display_LCD
619:# Arg         : width, default is 8 digit
620:# Return: -
621:# Function: Displays the LCD according to LCD_RAM value
622:
623:proc display_LCD {{width 8}} {
624:    global lcdshape
625:    global onLCD
626:    global offLCD
627:    global Static
628:    global half
```

ram_val に格納された LCD RAM の値から、点灯している LCD のセグメント（a～o）を調べる

```
629:    global oneThird
630:    global dutyCycle
631:    global seg1to4
632:    global seg5to8
633:    global seg9to12
634:    global seg13to16
635:    global seg17to20
636:    global seg21to24
637:    global seg25to28
638:    global seg29to32
639:    set lcdoffset 280
640:
641:    update
642:
643:    for {set i 0} {$i<8} {incr i} {
644:        set digitOn 0
645:
646:        switch -- $lcdoffset {
647:            280 {set digitOn $seg1to4}
648:            240 {set digitOn $seg5to8}
649:            200 {set digitOn $seg9to12}
650:            160 {set digitOn $seg13to16}
651:            120 {set digitOn $seg17to20}
652:            80 {set digitOn $seg21to24}
653:            40 {set digitOn $seg25to28}
654:            0 {set digitOn $seg29to32}
655:        }
656:
657:        if [info exists onLCD($i)] {
658:            if {$digitOn} {
659:                foreach symbol $onLCD($i) {
660:                    .l.c move [eval .l.c create line $lcdshape($symbol) -fill black -width 2
                        -tags lcd ] $lcdoffset 0
661:                }
662:            }
663:        }
664:        incr lcdoffset -40
665:    }
666:
667:    update
668:}
```

点灯している LCD のセグメント（a～o）を描画

⑤メイン処理

トップレベルウィンドウの「Start」をクリックすると、下記のメイン処理が呼ばれます。LCD 制御レジスタの値をターゲットから読み込んだあと、LCD が表示状態になっている場合には、LCD RAM に格納されたパターンデータをもとにトップレベルウィンドウに LCD を再描画します。この処理を、「Stop」がクリックされるまで繰り返します。

```
253:################################################
254:# Proc        : start
255:# Arg         : -
256:# return: -
257:# Function: start the polling of register values
258:
259:proc start {} {
260:    global stop
261:    set stop 0
262:    global dutyCycle
263:    global seg1to4
264:    global seg5to8
265:    global seg9to12
266:    global seg13to16
267:    global seg17to20
268:    global seg21to24
```

```
269:    global seg25to28
270:    global seg29to32
271:    global segText
272:    global PSW
273:    global ACT
274:    global DISP
275:    global stby
276:    global stop
277:
278:    while {!$stop} {
279:        checkDutyCycle
280:        checkSegment
281:        checkPSW
282:        checkACT
283:        checkDISP
284:        checkStbyMode
285:
286:        showLCD
287:        update
288:        if {$dutyCycle=="1/4" && $PSW && $ACT && $DISP && !$stby} {display}
289:
290:        sleep (100)
291:    }
292:}
```

LCD 制御レジスタの取得

LCD の再描画

2.3. メモリ検証サンプル

メモリ検証サンプルは、モトローラ S 形式ファイル(以下 mot ファイルという)をターゲットにダウンロードし、さらに、ダウンロードしたメモリの内容と mot ファイルが一致しているかどうか検証するサンプルです。

(1) 使用方法

①HEW ワークスペースのオープン

“C¥WorkSpace¥MemoryVerify_sample¥MemoryVerify_sample.hws”を HEW で開きます。

②メモリ検証スクリプトファイルの読み込み

HEWメニューバー[表示->TCLツールキット]よりTCLツールキットを立ち上げ、メモリ検証用のスクリプトファイル“C:¥WorkSpace¥MemoryVerify_sample¥MemoryVerify_sample.tcl”を開きます。図 5は、メモリ検証サンプルの起動直後の画面です。

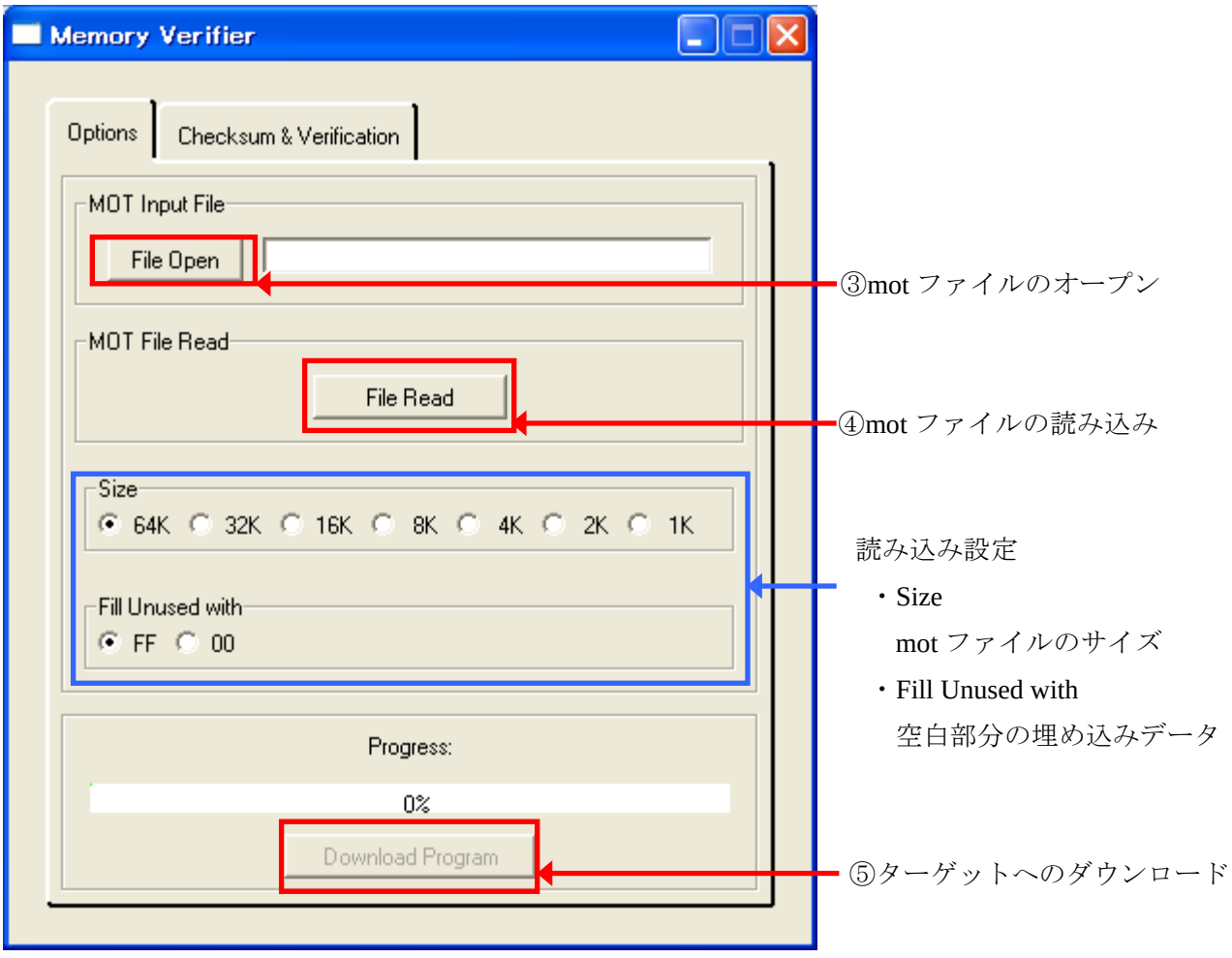


図 5 Memory Verifier 画面 Options タブ

③mot ファイルのオープン

図 5の「File Open」をクリックし、“MomoryVerify_Sample¥Debug¥MemoryVerify_sample.mot”を開いてください。

④mot ファイルの読み込み

図 5の「File Read」をクリックすると、motファイルを読み込みます。motファイルの読み込みが完了すると、図 6のメッセージが表示され、図 5の「Download Program」がクリックできるようになります。

図 6　mot ファイル読み込み完了

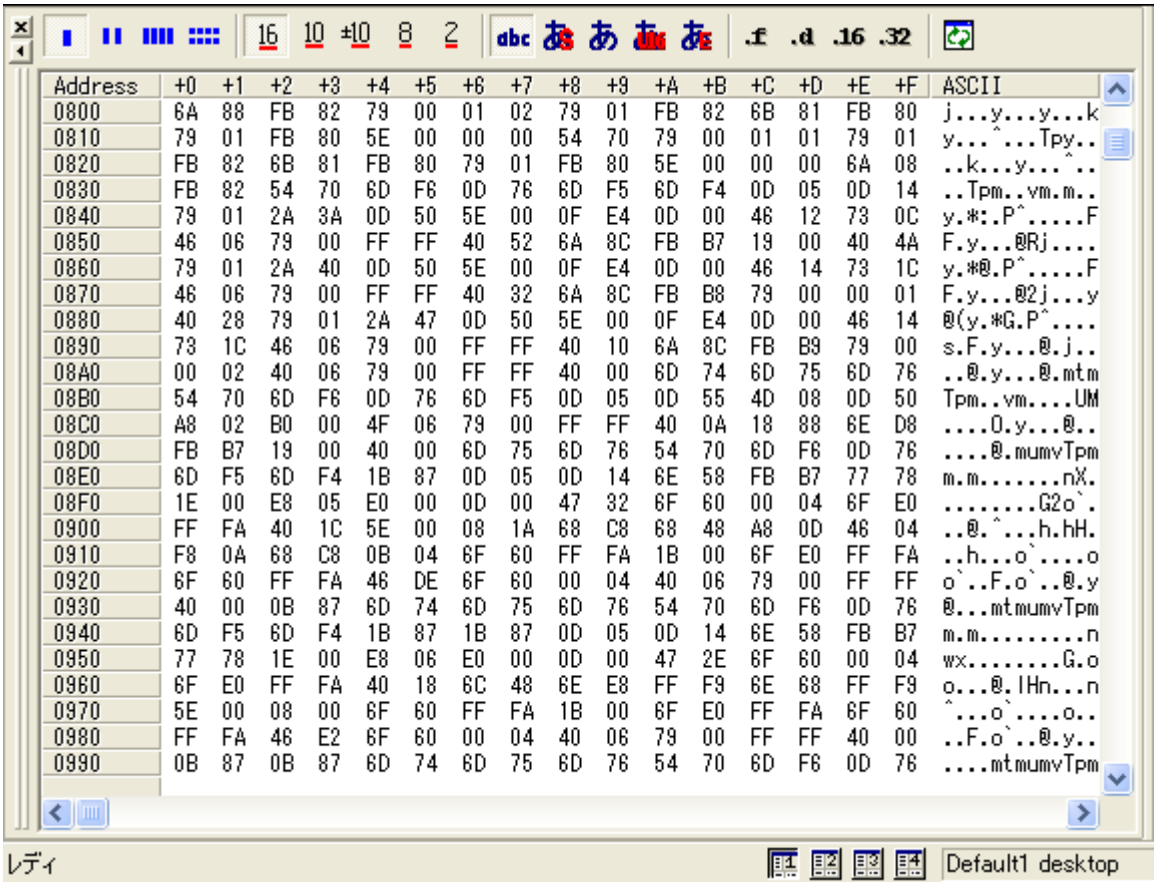
なお、図 5の読み込み設定は、必要に応じて設定を変更してください。「Size」には、ターゲットのROMサイズを指定します。「Fill Unused with」には、このあとのターゲットへのダウンロード処理において、motファイルにデータのない領域を 0xFFで埋めるか、0x00 で埋めるか選択します。なお、「Size」に指定した値よりも大きなサイズのmotファイルが指定された場合、図 7に示すエラーが表示されます。

図 7　Size の設定値によるエラー表示

⑤ターゲットへのダウンロード

図 5の「Download Program」をクリックすると、あらかじめ読み込んでおいたmotファイルのデータをターゲットにダウンロードします。ダウンロードが完了すると、図 8に示すメッセージが表示されます。また、図 9のように、HEWのメモリウィンドウ上でもターゲットプログラムがダウンロードされたことが確認できます。

図 8　ターゲットへのダウンロード完了



Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F	ASCII
0800	6A	88	FB	82	79	00	01	02	79	01	FB	82	6B	81	FB	80	j...y...y...k
0810	79	01	FB	80	5E	00	00	00	54	70	79	00	01	01	79	01	y...^...Tpy..
0820	FB	82	6B	81	FB	80	79	01	FB	80	5E	00	00	00	6A	08	..k...y...^...
0830	FB	82	54	70	6D	F6	0D	76	6D	F5	6D	F4	0D	05	0D	14	..Tpm..vm.m..
0840	79	01	2A	3A	0D	50	5E	00	0F	E4	0D	00	46	12	73	0C	y.*:.P^.....F
0850	46	06	79	00	FF	FF	40	52	6A	8C	FB	B7	19	00	40	4A	F.y...@Rj....
0860	79	01	2A	40	0D	50	5E	00	0F	E4	0D	00	46	14	73	1C	y.*@.P^.....F
0870	46	06	79	00	FF	FF	40	32	6A	8C	FB	B8	79	00	00	01	F.y...@2j...y
0880	40	28	79	01	2A	47	0D	50	5E	00	0F	E4	0D	00	46	14	@(y.*G.P^....
0890	73	1C	46	06	79	00	FF	FF	40	10	6A	8C	FB	B9	79	00	s.F.y...@.j..
08A0	00	02	40	06	79	00	FF	FF	40	00	6D	74	6D	75	6D	76	..@.y...@.mtm
08B0	54	70	6D	F6	0D	76	6D	F5	0D	05	0D	55	4D	08	0D	50	Tpm..vm....UM
08C0	A8	02	B0	00	4F	06	79	00	FF	FF	40	0A	18	88	6E	D8	....O.y...@..
08D0	FB	B7	19	00	40	00	6D	75	6D	76	54	70	6D	F6	0D	76	....@.mumvTpm
08E0	6D	F5	6D	F4	1B	87	0D	05	0D	14	6E	58	FB	B7	77	78	m.m.......nX.
08F0	1E	00	E8	05	E0	00	0D	00	47	32	6F	60	00	04	6F	E0	........G2o`.
0900	FF	FA	40	1C	5E	00	08	1A	68	C8	68	48	A8	0D	46	04	..@.^...h.hH.
0910	F8	0A	68	C8	0B	04	6F	60	FF	FA	1B	00	6F	E0	FF	FA	..h...o`....o
0920	6F	60	FF	FA	46	DE	6F	60	00	04	40	06	79	00	FF	FF	o`..F.o`..@.y
0930	40	00	0B	87	6D	74	6D	75	6D	76	54	70	6D	F6	0D	76	@...mtmumvTpm
0940	6D	F5	6D	F4	1B	87	1B	87	0D	05	0D	14	6E	58	FB	B7	m.m.........n
0950	77	78	1E	00	E8	06	E0	00	0D	00	47	2E	6F	60	00	04	wx........G.o
0960	6F	E0	FF	FA	40	18	6C	48	6E	E8	FF	F9	6E	68	FF	F9	o...@.lHn...n
0970	5E	00	08	00	6F	60	FF	FA	1B	00	6F	E0	FF	FA	6F	60	^...o`....o..
0980	FF	FA	46	E2	6F	60	00	04	40	06	79	00	FF	FF	40	00	..F.o`..@.y..
0990	0B	87	0B	87	6D	74	6D	75	6D	76	54	70	6D	F6	0D	76	....mtmumvTpm

図 9　ターゲットへのダウンロード結果(HEW メモリウィンドウ)

⑥チェックサムの表示とメモリ検証

チェックサムの表示とメモリ検証は、「Checksum & Verification」タブで行います。

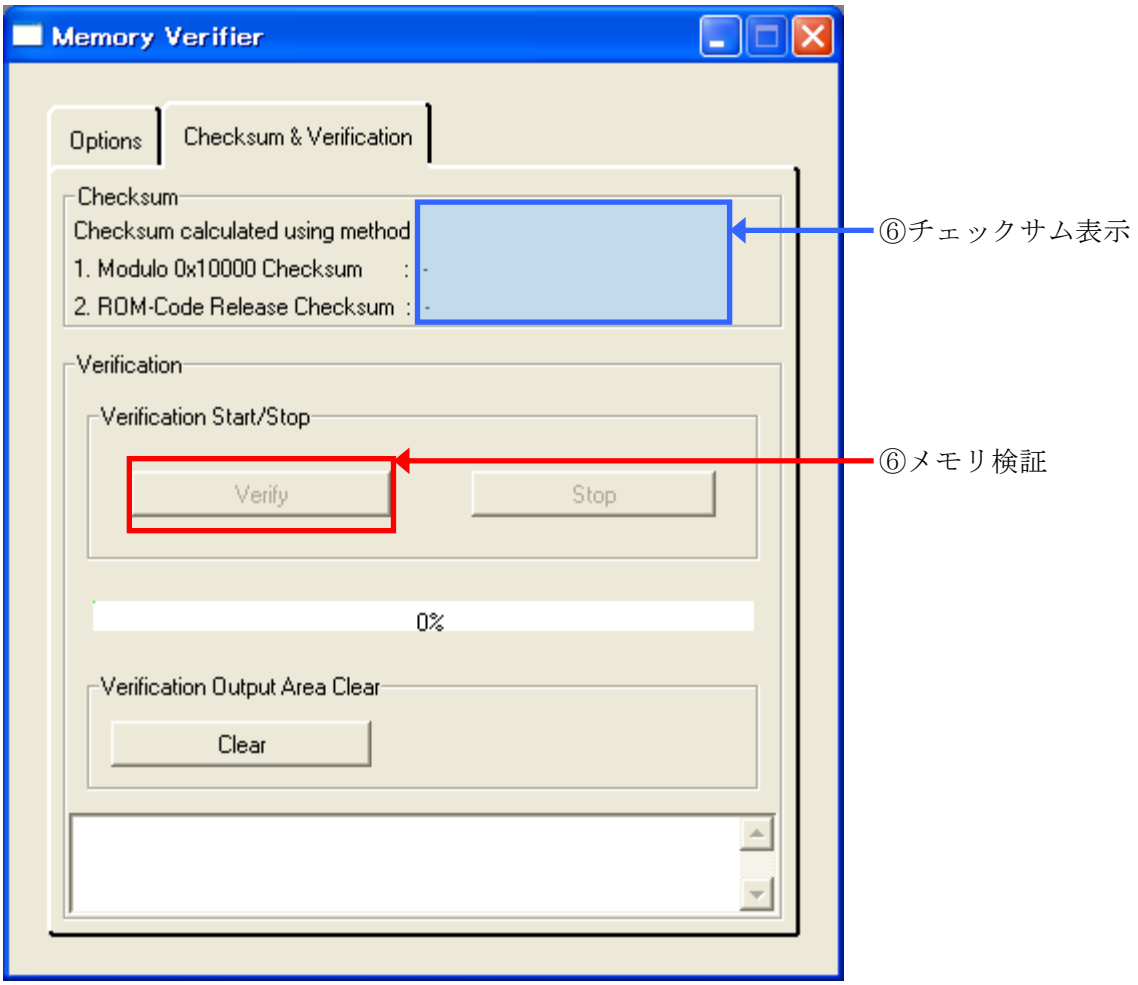


図 10　Memory Verifier 画面 - Checksum & Verification タブ

・チェックサムの表示

mot ファイルを読み込むと、mot ファイルから計算したチェックサムが「Checksum & Verification」タブの「Checksum」に表示されます。

・メモリ検証

「Checksum & Verification」タブの「Verify」をクリックし、ターゲットのメモリ内容と mot ファイル間の差分をチェックします。

差分がない場合（正常終了）は、図 11、図 12のように差分がなかった旨が表示されます。

差分がある場合（メモリへの書き込み異常など）は、図 13、図 14のように差分の詳細が表示されます。例えば、アドレス 0x852 番地において、motファイルのデータ 0x79 に対して、ターゲットのメモリが 0x87 と、差分の内容が表示されます。

なお、差分結果をクリアするには、「Checksum & Verification」タブの「Clear」をクリックしてください。また、メモリ検証を中断する場合は、「Stop」をクリックしてください。

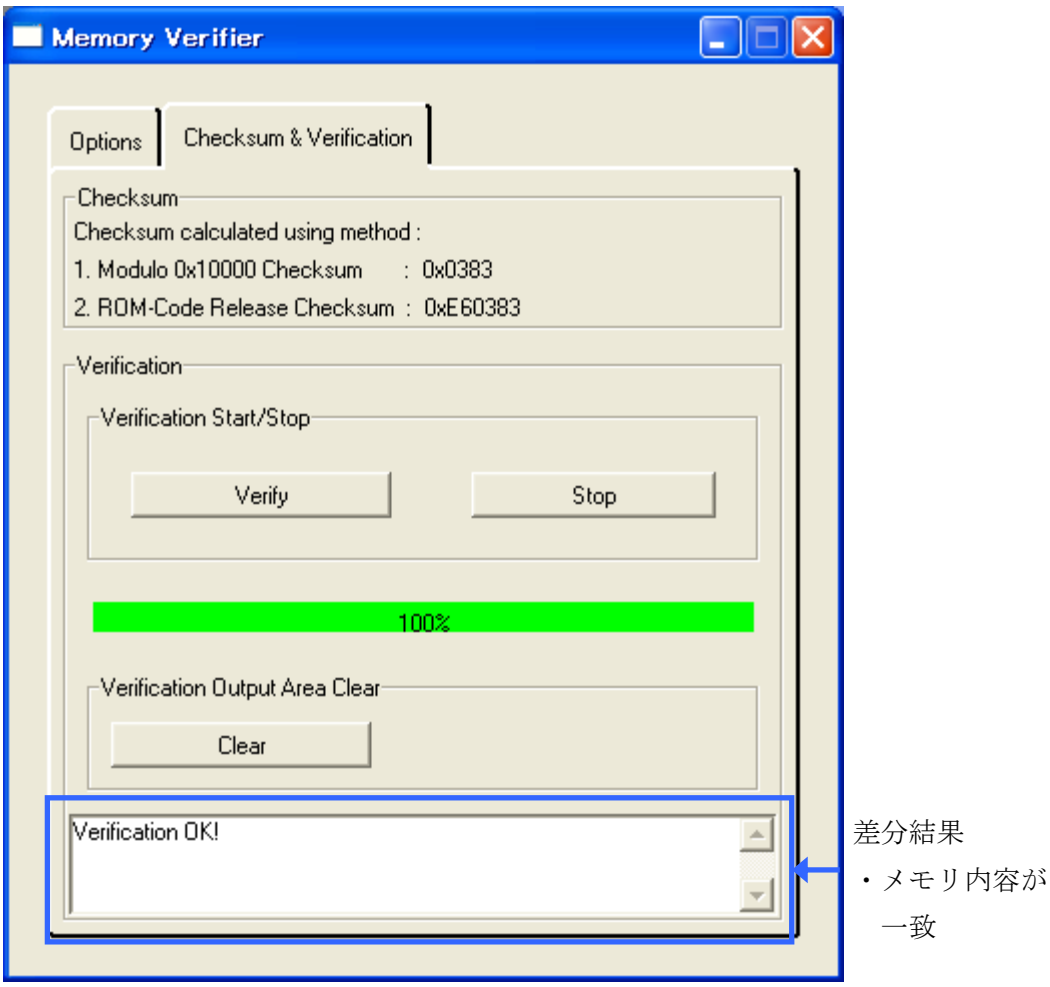


図 11　メモリ検証で差分が見つからなかった場合

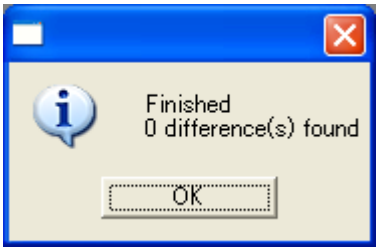


図 12　メモリ検証で差分が見つからなかった場合

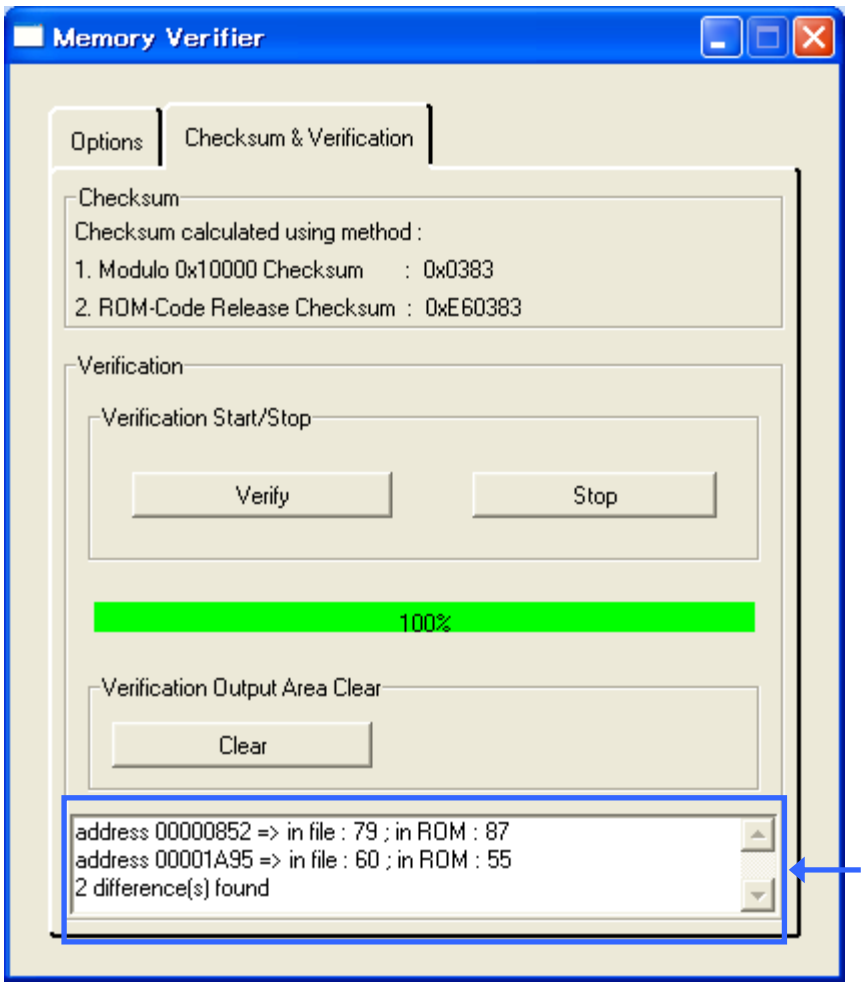


図 13　メモリ検証で差分が見つかった場合

図 14　メモリ検証で差分が見つかった場合

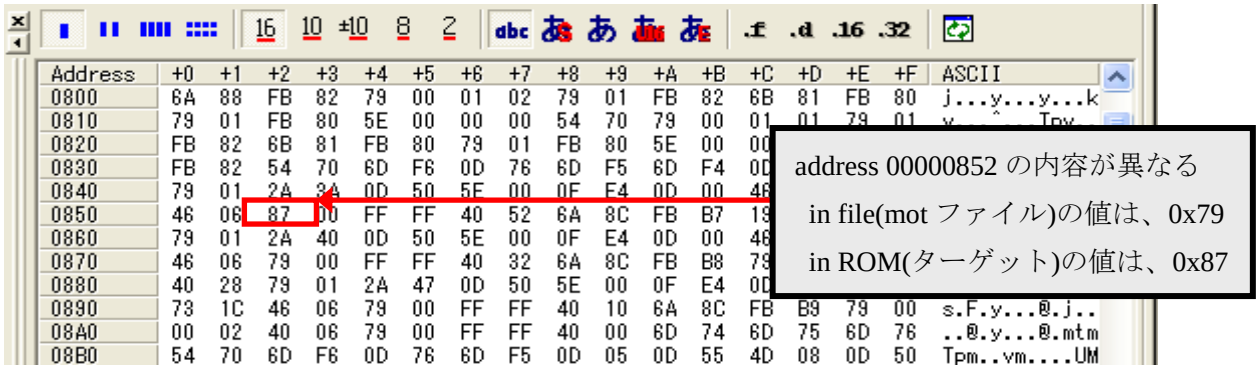


図 15　メモリ不一致箇所

(2) Tcl/Tk スクリプトの説明

Tcl/Tk スクリプトについて説明します。メモリ検証サンプルの Tcl/Tk スクリプトには、以下の処理が含まれています。

- トップレベルウィンドウの作成
- mot ファイルの読み込み（「File Read」クリック時の処理）
- ターゲットへのダウンロード（「Download Program」クリック時の処理）
- メモリの検証（「Verify」クリック時の処理）

①トップレベルウィンドウの作成

下記の処理により、トップレベルウィンドウを作成します。なお、タブやプログレスバーを作成するための汎用的な処理は、別のスクリプトファイル（notedef.tcl）に記述しています。本アプリケーションノートでは、notedef.tcl についての説明は省略します。

```
 1:#load the notebook and progress bar widget
 2:source notedef.tcl
 3:
 4:#############################################################
 5:# The following code implements the GUI
 6:#
 7:
 8:wm title . "Memory Verifier"
 9:
10:#decide whether to hide the "detail view" tab
11:Notebook:create 370 380 .n -pages {Options "Checksum & Verification"} -pad 20
12:pack .n -fill both -expand 1
13:set w [Notebook:frame .n Options]
14:set u [Notebook:frame .n "Checksum & Verification"]
15:
16:#############################################################
17:# NOTEBOOK Checksum & Verification TAB
18:#
19:
20:#//////////////////////////////
21:# LabelFrame
22:labelframe $u.lf1 -text Checksum
23:label $u.lf1.l1 -text "Checksum calculated using method :"
24:label $u.lf1.l2 -text "1. Modulo 0x10000 Checksum       :  -"
25:label $u.lf1.l3 -text "2. ROM-Code Release Checksum  :  -"
26:pack $u.lf1.l1 $u.lf1.l2 $u.lf1.l3 -side top -anchor nw
27:pack $u.lf1 -side top -fill both -expand 1 -pady 5 -padx 5
28:
29:#//////////////////////////////
30:# LabelFrame
31:labelframe $u.lf2 -text Verification
32:
33:frame      $u.lf2.f
34:
35:labelframe $u.lf2.1 -text "Verification Start/Stop"
36:
37:button     $u.lf2.1.b1 -state disabled -text "Verify" -command {verify} -width 20
38:global stop
39:set stop 0
40:button     $u.lf2.1.b2 -state disabled -text "Stop" -command {set stop 1} -width 20
41:
42:progressbar_create $u.lf2.fb green
43:set pc 0
44:progressbar_value $u.lf2.fb $pc
45:
46:labelframe $u.lf2.2 -text "Verification Output Area Clear"
```

「Checksum & Verification」タブの作成

```
 47:
 48:button     $u.lf2.2.b1 -text "Clear" -command {$u.lf2.st delete 0 end} -width 20
 49:
 50:listbox    $u.lf2.st -height 100 -yscrollcommand "$u.lf2.st.y set"
 51:scrollbar $u.lf2.st.y -command "$u.lf2.st yview"
 52:
 53:pack $u.lf2      -side top -fill both -expand 1 -pady 5 -padx 5
 54:pack $u.lf2.f    -side top
 55:
 56:#verify, stop
 57:pack $u.lf2.1    -expand 1 -side top -fill both -padx 10 -pady 10
 58:pack $u.lf2.1.b1 $u.lf2.1.b2 -anchor nw -fill both -side left -padx 20 -pady 20
 59:#pack $u.lf2.1.b2 -anchor nw -side left -pady 5
 60:
 61:#progressbar
 62:pack $u.lf2.fb   -expand yes -fill both -padx 10 -pady 10
 63:
 64:#clear
 65:pack $u.lf2.2    -expand 1 -side top -fill both -padx 10 -pady 10
 66:pack $u.lf2.2.b1 -anchor nw -side left -fill both -padx 10 -pady 10
 67:
 68:#listbox
 69:pack $u.lf2.st $u.lf2.st.y -side top -expand 0 -fill x
 70:
 71:Notebook:raise.page .n 0
 72:
 73:#################################################################
 74:# NOTEBOOK Options TAB
 75:#
 76:
 77:#//////////////////////////////
 78:# Frame
 79:frame $w.f -bd 2 -relief groove
 80:pack  $w.f -fill both -expand 1 -padx 5 -pady 5
 81:
 82:# labelframe setup
 83:labelframe $w.f.1 -text "MOT Input File"
 84:labelframe $w.f.2 -text "MOT File Read"
 85:labelframe $w.f.3 -text "Size"
 86:labelframe $w.f.4 -text "Fill Unused with"
 87:
 88:# MOT Input File
 89:frame  $w.f.1.f0
 90:button $w.f.1.f0.b0 -text "File Open" -command {
 91:    set inputFile [tk_getOpenFile -defaultextension ".mot"]
 92:} -width 10
 93:entry  $w.f.1.f0.e0 -textvariable inputFile -width 230
 94:
 95:# MOT File READ CLOSE
 96:frame  $w.f.2.f0
 97:button $w.f.2.f0.b1 -text "File Read" -command {startRead} -width 15
 98:
 99:# Size
100:set size "64k"
101:radiobutton $w.f.3.b1 -text "64K" -variable size -value "64k"
102:radiobutton $w.f.3.b2 -text "32K" -variable size -value "32k"
103:radiobutton $w.f.3.b3 -text "16K" -variable size -value "16k"
104:radiobutton $w.f.3.b4 -text " 8K" -variable size -value "8k"
105:radiobutton $w.f.3.b5 -text " 4K" -variable size -value "4k"
106:radiobutton $w.f.3.b6 -text " 2K" -variable size -value "2k"
107:radiobutton $w.f.3.b7 -text " 1K" -variable size -value "1k"
108:
109:# Fill Unused with
110:set fill "FF"
111:radiobutton $w.f.4.b1 -text "FF"   -variable fill -value "FF"
112:radiobutton $w.f.4.b2 -text "00"   -variable fill -value "00"
113:
114:
```

「Options」タブの作成

```
115:pack $w.f.1 -expand 1 -fill both -padx 5 -pady 5
116:pack $w.f.2 -expand 1 -fill both -padx 5 -pady 5
117:pack $w.f.3 -expand 1 -fill both -padx 10 -pady 10
118:pack $w.f.4 -expand 1 -fill both -padx 10 -pady 10
119:
120:pack $w.f.1.f0 -side top -padx 10 -pady 10
121:pack $w.f.1.f0.b0 -anchor nw -side left -padx 5
122:pack $w.f.1.f0.e0 -anchor nw -side left -padx 5
123:
124:pack $w.f.2.f0 -side top -padx 10 -pady 10
125:pack $w.f.2.f0.b1 -anchor nw -side left -padx 5
126:
127:pack $w.f.3.b1 -side left
128:pack $w.f.3.b2 -side left
129:pack $w.f.3.b3 -side left
130:pack $w.f.3.b4 -side left
131:pack $w.f.3.b5 -side left
132:pack $w.f.3.b6 -side left
133:pack $w.f.3.b7 -side left
134:
135:pack $w.f.4.b1 -side left
136:pack $w.f.4.b2 -side left
137:
138:#////////////////////////////
139:# Frame
140:frame $w.f2 -bd 2 -relief groove
141:
142:label $w.f2.lProg -text Progress:
143:pack $w.f2.lProg -side top -anchor center  -fill x -pady 5
144:
145:#Progress Bar Set
146:progressbar_create $w.f2.fb green
147:pack $w.f2.fb -expand yes -fill both -padx 10 -pady 10
148:set pc2 0
149:progressbar_value $w.f2.fb $pc2
150:pack $w.f2.fb  -side top -anchor center  -pady 2
151:
152:button $w.f2.b -text "Download Program" -state disabled -command {download} -width 20
153:pack $w.f2.b  -side top -anchor center -pady 5 -padx 5
154:pack $w.f2 -fill both -expand 1 -padx 5 -pady 5 -side top
```

②mot ファイルの読み込み

図 5の「File Read」をクリックすると、下記の処理が呼ばれます。motファイルに書かれているテキストデータを読み込み、ターゲットへのダウンロードデータとして変数contextなどに格納します。

```
157:#################################################################
158:# Proc        : startRead
159:# Arg         : -
160:# Return      : -
161:# Function    : Read the .mot file
162:proc startRead {} {
163:    . configure -cursor watch
164:    upvar #0 inputFile inputFile
165:    upvar #0 w w
166:    upvar #0 u u
167:    upvar #0 fill fill
168:    upvar #0 size size
169:
170:    global fileFormat
171:    global byteUsed
172:    global lastFilled
173:    global showDetail
174:
175:    set byteUsed 0
176:
177:    $w.f2.b configure -state disabled
```

「Download Program」、「Verify」、「Stop」ボタンの無効化

```
178:    $u.lf2.1.b1 configure -state disabled
179:    $u.lf2.1.b2 configure -state disabled
180:
181:    global pc2
182:
183:    catch {
184:        set pc2 0
185:        progressbar_value $w.f2.fb $pc2
186:    }
187:
188:    if {$inputFile==""} {tk_messageBox -message "No file entered" ; return}
189:    if {[catch {set fd [open "$inputFile" RDONLY]}]} {tk_messageBox -message "Filename
        $inputFile not found" ; return}
190:
191:    global size2
192:    switch -- $size {
193:        "64k" {set size2 0xffff}
194:        "32k" {set size2 0x7fff}
195:        "16k" {set size2 0x3fff}
196:        "8k" {set size2 0x1fff}
197:        "4k" {set size2 0xfff}
198:        "2k" {set size2 0x7ff}
199:        "1k" {set size2 0x3ff}
200:    }
201:
202:    #fill the whole array
203:    global content
204:    global filled
205:    for {set i 0} {$i<=$size2} {incr i} {
206:        set content($i) $fill
207:        set filled($i) 0
208:    }
209:
210:    #get the header
211:    set header [gets $fd]
212:
213:    incr pc2 25 ; update
214:    progressbar_value $w.f2.fb $pc2
215:    update
216:
217:    while {![eof $fd]} {
218:        set temp 0
219:        while {[read $fd 1]!="S"} {}
220:
221:        set data [gets $fd]
222:        set dataList [split $data ""]
223:        set format [lindex $dataList 0]
224:
225:        set temp "[lindex $dataList 1][lindex $dataList 2]"
226:        set byteCount 0x$temp
227:
228:
229:        set reachEnd 0
230:        switch -- $format {
231:            1 { set numbOfAddByte 2 ;set fileFormat $format}
232:            2 { set numbOfAddByte 3 ;set fileFormat $format}
233:            3 { set numbOfAddByte 4 ;set fileFormat $format}
234:            7 {
235:                set temp2 [gets $fd]
236:                set end "S$format$temp$temp2"
237:                set reachEnd 1
238:                incr pc2 25 ; update
239:                progressbar_value $w.f2.fb $pc2
240:              }
241:            8 {
242:                set temp2 [gets $fd]
243:                set end "S$format$temp$temp2"
244:                set reachEnd 1
```

mot ファイルの読み込み

データフォーマット（S1,S2,S3,S7,S8,S9）の解析

```
245:                incr pc2 25 ; update
246:                progressbar_value $w.f2.fb $pc2
247:              }
248:            9 {
249:                set temp2 [gets $fd]
250:                set end "S$format$temp$temp2"
251:                set reachEnd 1
252:                incr pc2 25 ; update
253:                progressbar_value $w.f2.fb $pc2
254:              }
255:        }
256:
257:        if {!$reachEnd} {
258:            if [catch {set numbOfDataByte [expr $byteCount-$numbOfAddByte-0x01]}] {
259:                tk_messageBox -message "Error in reading file, possibly cause by incorrect
                    file format." -icon error
260:                . configure -cursor ""
261:                set pc2 0
262:                progressbar_value $w.f2.fb $pc2
263:                return
264:            }
265:            set temp ""
266:            if {$format==1} {
267:                for {set i 3} {$i<=6} {incr i} {append temp [lindex $dataList $i]}
268:            } ¥
269:            elseif {$format==2} {
270:                for {set i 3} {$i<=8} {incr i} {append temp [lindex $dataList $i]}
271:            } ¥
272:            else {
273:                for {set i 3} {$i<=10} {incr i} {append temp [lindex $dataList $i]}
274:            }
275:            set address 0x$temp
276:            set addCopy [format "%01i" $address]
277:
278:            set dataByte ""
279:            if {$format==1} {
280:                set lastDataBit [expr 6+(2*$numbOfDataByte)]
281:                for {set i 7} {$i<=$lastDataBit} {incr i} {
282:                    append dataByte [lindex $dataList $i]
283:                }
284:            } ¥
285:            elseif {$format==2} {
286:                set lastDataBit [expr 8+(2*$numbOfDataByte)]
287:                for {set i 9} {$i<=$lastDataBit} {incr i} {
288:                    append dataByte [lindex $dataList $i]
289:                }
290:            } else {
291:                set lastDataBit [expr 10+(2*$numbOfDataByte)]
292:                for {set i 11} {$i<=$lastDataBit} {incr i} {
293:                    append dataByte [lindex $dataList $i]
294:                }
295:            }
296:            set b [split $dataByte ""]
297:            set j 0
298:            for {set i 0} {$i<$numbOfDataByte} {incr i } {
299:                if {$addCopy>$size2} {
300:                    tk_messageBox -message "Some part of the content allocated outside the
                        size specified.¥nPlease try again with larger size!" ¥
301:                            -icon warning
302:                    . configure -cursor ""
303:                    set pc2 0
304:                    progressbar_value $w.f2.fb $pc2
305:                    return
306:                }
307:                set content($addCopy) "[lindex $b $j][lindex $b [expr $j+1]]"
308:                set filled($addCopy) 1
309:                set lastFilled $addCopy
310:                incr byteUsed
```

1行分のデータレコードの解析

```
311:                incr addCopy
312:                incr j 2
313:            }
314:        }
315:        if {$reachEnd} {break;}
316:    }
317:
318:    close $fd
319:    incr pc2 25
320:    progressbar_value $w.f2.fb $pc2
321:    update
322:
323:    $w.f2.b configure -state normal
324:    $u.lf2.1.b1 configure -state normal
325:    $u.lf2.1.b2 configure -state normal
326:
327:    calcChecksum
328:    incr pc2 25
329:    progressbar_value $w.f2.fb $pc2
330:    update
331:    . configure -cursor ""
332:    tk_messageBox -message "Finished"
333:}
```

「Download Program」、「Verify」、「Stop」ボタンの有効化

チェックサムの計算

下記の処理により、mot ファイルの各レコードのチェックサムを計算し、結果を表示します。

```
448:#############################################################################################################
##################
449:# Proc         : calcChecksum
450:# Arg          : -
451:# Return       : -
452:# Function     : to calculate the checksum of all bytes in the file (rom code release checksu)
453:#
454:proc calcChecksum {} {
455:    upvar #0 size size
456:    global content
457:    upvar #0 fileFormat fileFormat
458:    upvar #0 w w
459:    global pc
460:
461:    global size2
462:
463:    set checksum 0
464:    for {set i 0} {$i<=$size2} {incr i} {
465:        set a 0x$content($i)
466:        set checksum [expr $checksum+$a]
467:    }
468:
469:    set checksum2 [format "%01X" $checksum]
470:    set checksum1 [expr $checksum%0x10000]
471:    set checksum1 [format "%04X" $checksum1]
472:
473:    upvar #0 u u
474:    $u.lf1.l2 configure -text "1. Modulo 0x10000 Checksum       :  0x$checksum1"
475:    $u.lf1.l3 configure -text "2. ROM-Code Release Checksum  :  0x$checksum2"
476:}
477:
478:#make the main window not resizable either in x or y direction
479:wm resizable . 0 0
```

チェックサムの計算

③ターゲットへのダウンロード

「Download Program」をクリックすると、下記の処理が呼ばれます。mot ファイルから読み込んだデータをターゲットにダウンロードします。このとき、HEW の”mf（memory_fill）”コマンドを使用

します。

```
414:#################################################################
415:# Proc        : download
416:# Arg         : -
417:# Return      : -
418:# Function    : to download the content of file to physical memory
419:#
420:proc download {} {
421:    . configure -cursor watch
422:    global content
423:    global filled
424:    global size2
425:    global lastFilled
426:    upvar #0 w w
427:    global pc2
428:
429:    set pc2 0
430:    progressbar_value $w.f2.fb $pc2
431:    for {set i 0} {$i<=$lastFilled} {incr i} {
432:        if {$filled($i)} {
433:            set formatted [format "%08X" $i]
434:            set formatted1 [format "%08X" [expr 1+$i]]
435:            if [catch {mf H'$formatted H'$formatted1 H'$content($i)}] {tk_messageBox
                -message "Communication error. Please make sure the device is connected to HEW!"
                -title HEW -icon error ; . configure -cursor "" ; return}
436:        }
437:        if {$i!=0}  {
438:            if {![expr $i%($lastFilled/5)]} {
439:                incr pc2 20 ; update
440:                progressbar_value $w.f2.fb $pc2
441:            }
442:        }
443:    }
444:    . configure -cursor ""
445:    tk_messageBox -message "Finished!"
446:}
```

HEW の mf コマンドを使用して、
mot ファイル内容をターゲットにダウンロード

④メモリ検証(Verify)

「Verify」をクリックすると、下記の処理が呼ばれます。ターゲットのメモリを読み込みながら、あらかじめ mot ファイルから読み込んでおいた値と比較します。このとき、HEW の“md（memory_display）”コマンドを使用して、ターゲットのメモリを読み込みます。比較した結果、差分があった場合は、そのアドレスと値を表示します。

```
335:#################################################################
336:# Proc        : verify
337:# Arg         : -
338:# Return      : -
339:# Function    : Verify content of file with physical memory
340:#
341:proc verify {} {
342:    . configure -cursor watch
343:    global content
344:    global filled
345:    global size2
346:    upvar #0 u u
347:    upvar #0  w w
348:    global pc
349:
350:    global env
351:    global stop
352:    set stop 0
353:    global lastFilled
354:
```

```
355:    set count 0
356:    $u.lf2.st delete 0 end
357:    set pc 0
358:    progressbar_value $u.lf2.fb $pc
359:
360:    set noOfLines [expr $lastFilled/16]
361:
362:    for {set i 0} {$i<=$noOfLines} {incr i} {
363:        if {$stop} {
364:            set answer [tk_messageBox -type yesno -message "Are you sure to stop
                verification?" -icon warning]
365:            if {$answer=="yes"} {
366:                . configure -cursor ""
367:                tk_messageBox -message "Verification stopped by user"
368:                set pc 0 ; return
369:                progressbar_value $u.lf2.fb $pc
370:            }
371:        }
372:        catch {if {$i!=0} {if {![expr $i%($noOfLines/4)]} {incr pc 20 ; update ;
            progressbar_value $u.lf2.fb $pc}}}
373:        if {$noOfLines<4} {incr pc 60 ; update ; progressbar_value $u.lf2.fb $pc}
374:        set formatted [format "%08X" [expr 16*$i]]
375:        set fd [open $env(TMP)/log.txt {RDWR CREAT TRUNC}]
376:        close $fd
377:        if [catch {puts [md H'$formatted H'10]}] {tk_messageBox -message "Communication
            error. Please make sure the device is connected to HEW!" -title HEW -icon error ;
            . configure -cursor "" ; return}
378:        set fd [open $env(TMP)/log.txt RDWR]
379:        set b ""
380:        seek $fd 0 start
381:        while {$b!=$formatted} {
382:            set e [gets $fd]
383:            set d [split $e ""]
384:            set b ""
385:            for {set j 0} {$j<8} {incr j} {append b [lindex $d $j]}
386:        }
387:        set ar [split $e]
388:
389:        set start [expr 16*$i]
390:        for {set j 0} {$j<16} {incr j} {
391:            set k [expr $start+$j]
392:            if {$filled($k)} {
393:                set data [lindex $ar [expr $j+2]]
394:                #puts "$content($k)  $data"
395:                if {$data!=$content($k)} {
396:                    $u.lf2.st insert end "address [format "%08X" $k] => in file : $content($k) ;
                    in ROM : $data¥n"
397:                    incr count
398:                }
399:            }
400:        }
401:        close $fd
402:        update
403:    }
404:
405:    if {$count==0} {$u.lf2.st insert end "Verification OK!"} ¥
406:    else {$u.lf2.st insert end "¥n$count difference(s) found"}
407:
408:    incr pc 20 ; update
409:    progressbar_value $u.lf2.fb $pc
410:    . configure -cursor ""
411:    tk_messageBox -message "Finished¥n$count difference(s) found"
412:}
```

HEW の md コマンドで、ターゲットのメモリを読み込む

ターゲットのメモリと mot ファイルの内容を比較

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ

http://japan.renesas.com/

お問合せ先

http://japan.renesas.com/inquiry

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2011.01.11	—	初版発行
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-
-	-	-	-

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA機器、通信機器、計測機器、AV機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口 http://www.renesas.com

※営業お問合せ窓口の住所・電話番号は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス販売株式会社　〒100-0004　千代田区大手町2-6-2（日本ビル） (03)5201-5307

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：http://japan.renesas.com/inquiry

Colophon 1.0