System Console におけるダッシュボードの作成手順とサンプル紹介

Ver.18.0

2019 年 11 月 Rev.1

System Console におけるダッシュボードの作成手順とサンプル紹介

目次

本書をお読みになる前に

この資料の内容は 2019 年 9 月現在のものです。

この資料で紹介しているソフトウェアやハードウェア、操作手順などは、指定バージョンやデバイス等以外でも共通のものもありますが、一部については共通にならないものもありますので、ご注意ください。

文書中の記号

ⓘ **Note**	補足情報などを記載しています。
Ⓟ **Point**	重要なポイントを記載しています。
参考	理解を深めるため、参考となる資料やサイトを紹介しています。
⚠ **注記**	この資料の中では具体的には触れませんが、必要となる知識や情報を記載しています。
⃠ **禁止**	注意点や、してはいけないことを記載しています。

文中の表記

下線	クリックする事で、資料中の別の章や、外部のサイトにジャンプします。
太字斜体	画面の操作をする際の、メニューやウィンドウなどに表示されている文字を示しています。
xxxxxxx↵	入力するコマンド文字列を示しています。
網掛け	使用するツールを示しています。

1. はじめに

Intel® Quartus® Prime には JTAG 経由でデバッグが行える System Console という機能があります。

System Console には GUI のようにグラフィカルに操作することのできるダッシュボードという機能があります。

この資料では、ダッシュボードの基本的な作成手順とサンプルを紹介しています。

2. System Console についてと実装方法

2-1. System Console について

System Console は JTAG を経由したデバッグを行うことができる機能です。

JTAG 経由で System Console に接続されているペリフェラルに対するレジスタアクセスや、JTAG マスターのクロックの確認等の機能を有しており、様々な操作が可能です。

> **参考:**
>
> - System Console 関する情報については、以下の資料が参考になります。
>   - 『System Console User Guide』
>   - 『Analyzing and Debugging Designs with System Console』

> Ⓟ Point:
>
> デバッグの際に簡単なレジスタアクセスであれば、Nios® II を実装しなくても System Console で行うことができます。

2-2. System Console を使用するために必要なペリフェラル

System Console は Platform Designer に以下のいずれかのペリフェラルを含めることで使用することができます。

- JTAG to Avalon Master Bridge (JTAG Master)
- Nios® II Processor with JTAG Debug Module
- USB Debug Master
- Avalon-ST (Streaming) JTAG Interface
- JTAG UART
- Ethernet component

本資料で提供するサンプルでは JTAG Master を使用している為、JTAG Master を用いた例を紹介します。

2-3. System Console を使用するためのデザイン作成

まず始めに JTAG Master をデザインに含めるために、Platform Designer の IP Catalog にて **JTAG to Avalon Master Bridge** と検索し選択します（図 2-1 参照）。

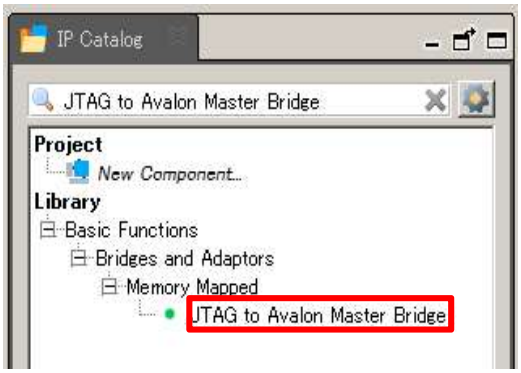


図 2-1 JTAG to Avalon Master Bridge の検索と選択

Platform Designer の System Contents タブでは、JTAG Master の入出力ピンを適切に接続する必要があります。

JTAG Master の入出力ピンの概要を以下に記載します。

- clk: クロック入力ピン
- clk_reset: リセット入力ピン
- master: Avalon Memory Mapped Master ピン。操作したい Slave のペリフェラルを選択してください
- master_reset: Reset Output ピン。接続先のペリフェラルを System Console のコマンドでリセットを発行することができます

図 2-2 に JTAG Master の Slave として、On-Chip Memory と PIO を含んだ Platform Designer の接続を紹介しますので参考にしてください。

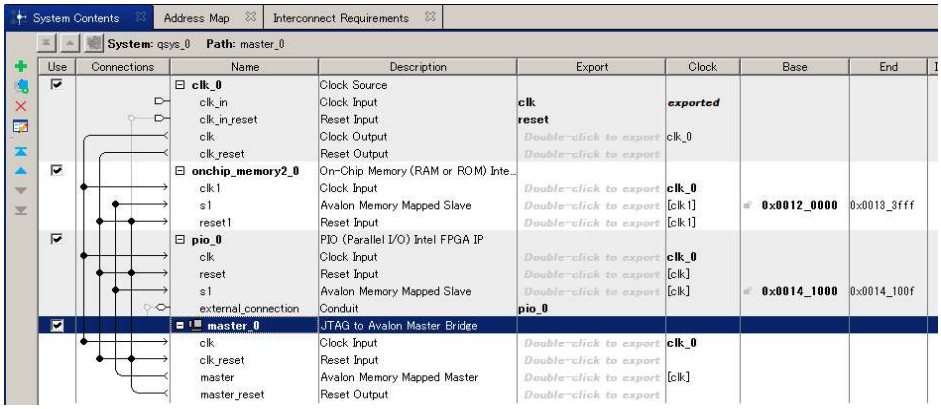


図 2-2 JTAG Master 接続例

作成した Platform Designer を組み込んで Quartus Prime でコンパイルすることによりデザインの作成は完了です。

2-4. System Console の立ち上げ方

System Console の立ち上げ方法は下記 3 通りです。

- Quartus Prime メニューバーから起動する方法
- Platform Designer から起動する方法
- Nios® II Command Shell から起動する方法

それぞれの起動方法についてこの後紹介します。

2-4-1. Quartus Prime から起動する方法

Quartus Prime メニューバーの Tools から System Debugging Tools ⇒ System Console を選択します（図 2-3 参照）。

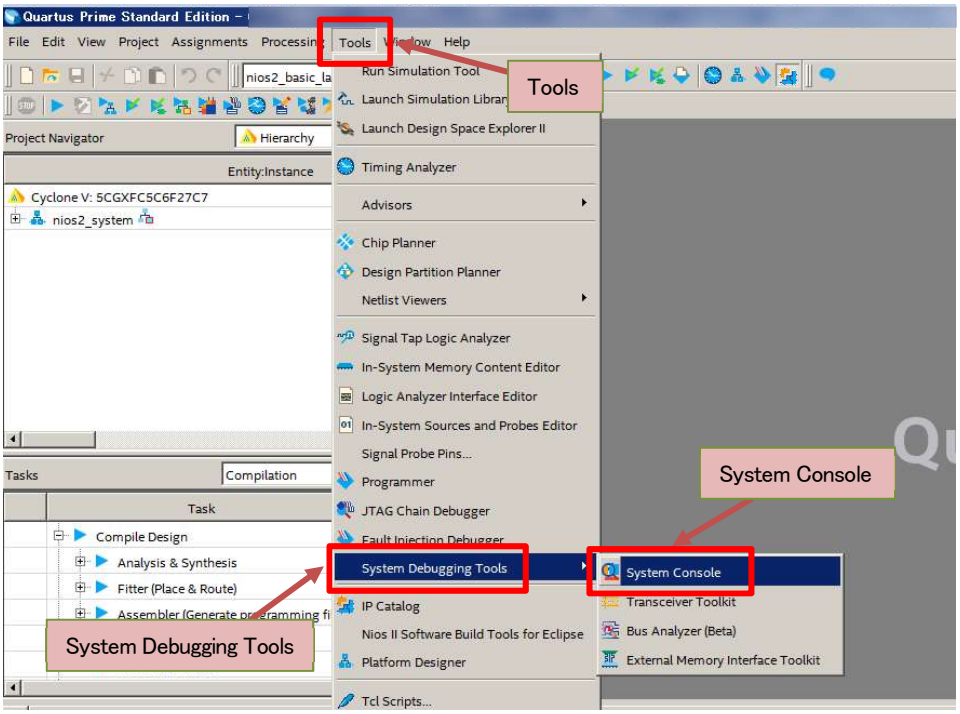


図 2-3 Quartsu Prime から System Console を起動

2-4-2. Platform Designer から起動する方法

Platform Designer メニューバーの Tools から System Console を選択します（図 2-4 参照）。

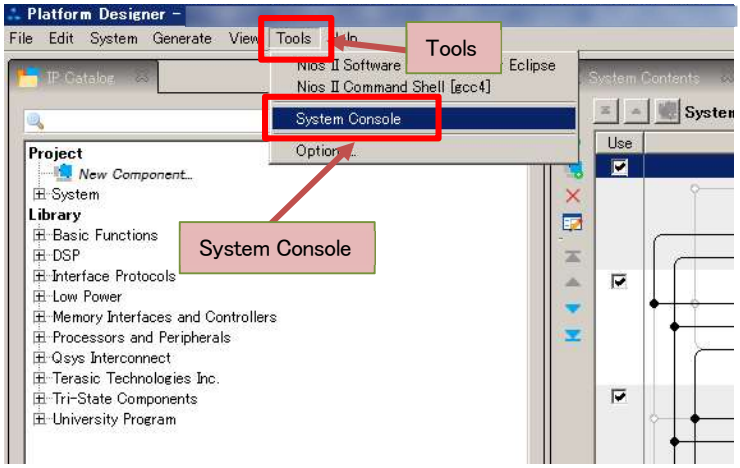


図 2-4 Platform Designer から System Console を起動

2-4-3. Nios II Command Shell から起動する方法

Nios® II Command Shell 起動し System-console.exe と入力します（図 2-5 参照）。

図 2-5 Nios® II Command Shell から System Console を起動

2-5. FPGA をコンフィグレーション

System Console でデバッグ操作を行う前に FPGA をコンフィグレーションする必要があります。

Programmer を用いてコンフィグレーションを行うことも可能ですが、本資料では System Console を用いたコンフィグレーションを紹介します。

まずはコンフィグレーションするデバイスに対するサービスパスを System Console 上で get_service_paths コマンドを用いて確認し指定します。

その後 device_download_sof コマンドでコンフィグレーションを行います。

以下にコマンドの使用例を記載しますのでご参考にしてください。

```
$ get_service_paths device
$ set d_path [lindex [get_service_paths device] 0]
$ device_download_sof $d_path test.sof
```

コンフィグレーションが完了したら System Console で各種コマンドを実行しデバッグ操作を行います。

以下によく使用するコマンドを記載しますのでご参考にしてください。

- 使用されるコマンドの例
  - get_service_paths: 使用可能なサービスパス一覧を表示します
  - open_service: サービスをオープンします
  - close_service: サービスをクローズします
  - jtag_debug_loop: JTAG チェーンで指定した値でループバックできるか確認します
  - jtag_debug_sample_clock: クロックをサンプリングします
  - jtag_debug_reset_system: Platform Designer で Avalon Master の master_reset ポートに接続されているペリフェラルに対しリセットを発行します
  - master_read_8/master_read_16/master_read_32: 指定したアドレスの値を Read します。master_read_8/master_16/master_32 はそれぞれ 8-bit/16-bit/32-bit で Read 可能です
  - master_write_8/master_write_16/master_write_32: 指定したアドレスに値を Write します。Master_write_8/master_write_16/master_write_32 はそれぞれ 8-bit/16-bit/32-bit で Write 可能です

参考:

- System Console で使用できるコマンドに関する情報については、以下の資料が参考になります。
  - 『System Console User Guide』

    Console Command の章をご覧ください。

3. ダッシュボードについて

“2. System Console についてと実装方法” の章で System Console の説明をしましたが、その都度コマンドを入力するのは面倒です。

そこでお勧めしたいのがダッシュボードという機能です。

ダッシュボードは事前に Tcl Script でボタンや LED 等の ウィジェットとコマンドを紐づけて操作用の GUI を作成することができる機能です。

一度 Tcl Script を記述するだけで、コマンド入力をその都度行う必要が無くなり、さらに視覚的に分かりやすいので操作性が向上します。

都度行っていたコマンド入力をどのように省略できるか以下で説明します。

図 3-1 は System Console でコマンド入力し、0x0000000 に 1 を write し、0x0000000 に 0 を write する例です。

この場合、入力しているコマンドは以下の 4 つです。

```
$ set master_path [lindex [get_service_paths master] 0]
$ open_service master $master_path
$ master_write_8 $master_path 0x00000000 1
$ master_write_8 $master_path 0x00000000 0
```

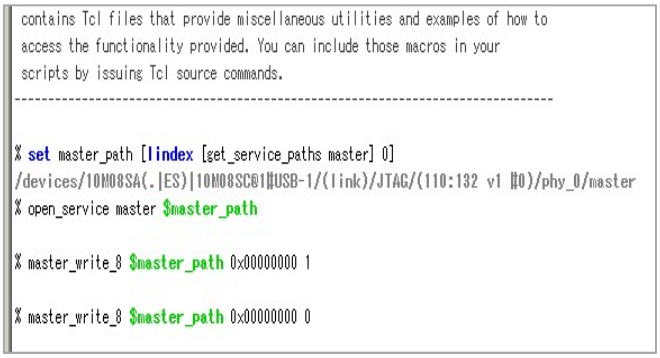


図 3-1 コマンド手入力により System Console を操作

比較的簡単なコマンドですが、これだけでも長いコマンドが 4 行必要になります。

次に上記のコマンドをダッシュボードで作成し、実行する例を示します（図 3-2）。

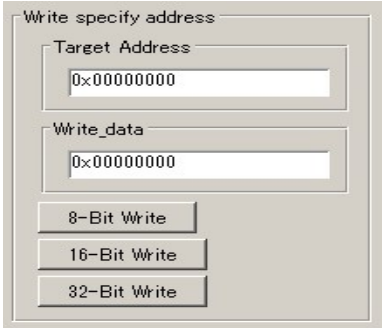


図 3-2 ダッシュボードを用いて System Console を操作

図 3-2 の ダッシュボードでは、下記の 2 コマンドは Tcl Script 内に既に記載しているので、起動すると自動で実行されます。

```
$ set master_path [lindex [get_service_paths master] 0]
$ open_service master $master_path
```

下記のコマンドについては、GUI の Target Address ボックスに 0x0000000、Write Data ボックスに 1 を入力し、8-Bit Write ボタンをクリックするだけで実行されます。

```
$ master_write_8 $master_path 0x00000000 1
```

下記のコマンドについても同様に、GUI の Target Address ボックスに 0x0000000、Write Data ボックスに 0 を入力し、8-Bit Write ボタンをクリックするだけで実行されます。

```
$ master_write_8 $master_path 0x00000000 0
```

4. ウィジェットの種類

ダッシュボードで使用できるウィジェットは複数あります。

先ほど登場した 図 3-2 の GUI では、次のようにグループ、ボタン、ラベルの 4 種類のウィジェットを用いて作成しました。

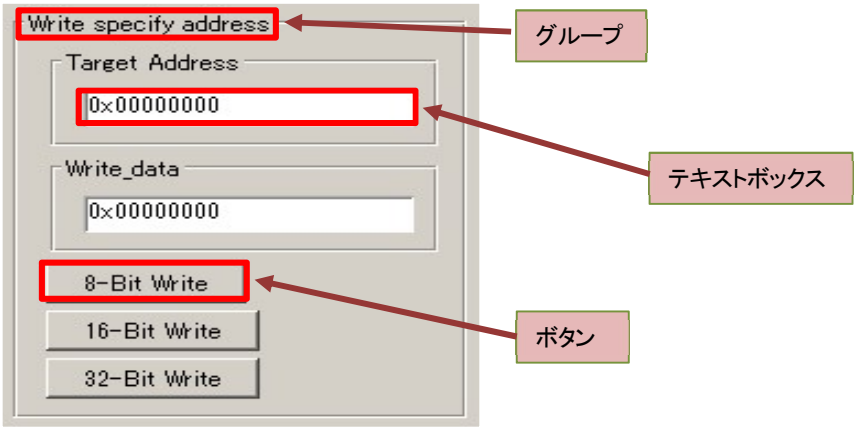


図 4-1 ウィジェットの使用例

> Ⓟ Point:
>
> 複数のウィジェットを組み合わせて、ダッシュボードを作成することにより、操作性が向上します。

ダッシュボードには 図 3-2 で紹介したグループ、テキストボックス、ボタン以外にも下記便利なウィジェットがあります。

- LED

図 4-2 LED

- ファイル選択ボタン

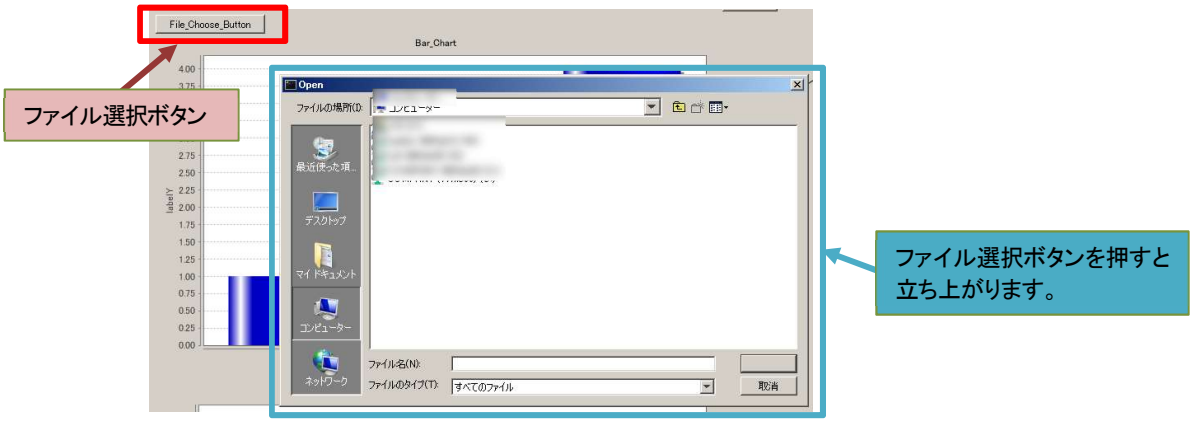


図 4-3 ファイル選択ボタン

・ ラベル

図 4-4 ラベル

・ 棒グラフ

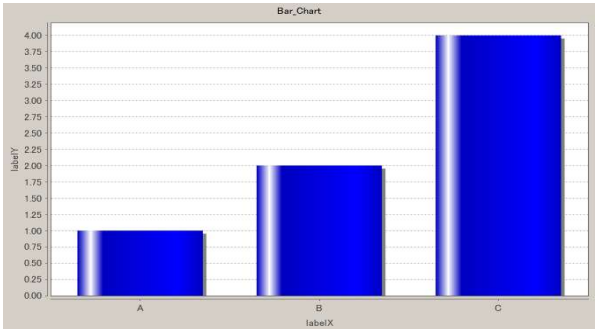


図 4-5 棒グラフ

・ ダイアル

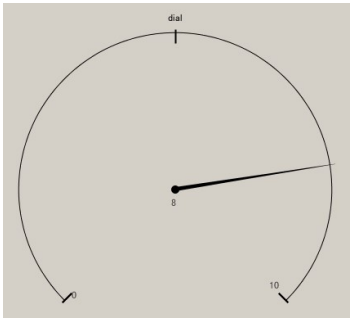


図 4-6 ダイアル

・ 折れ線グラフ

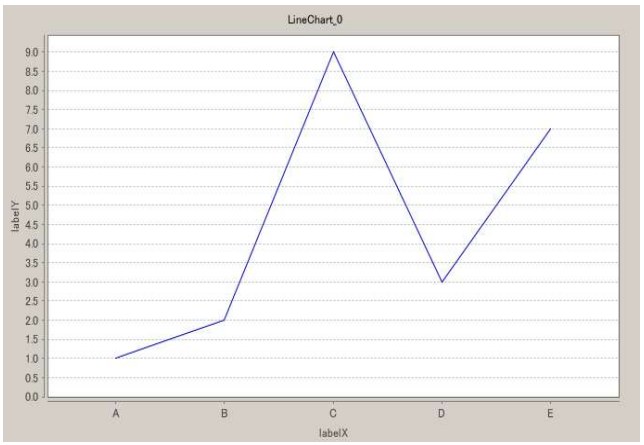


図 4-7 折れ線グラフ

5. ダッシュボード構成のための Tcl Script 作成手順

ダッシュボード構成の作成手順概要は、下記の通りです。

① ダッシュボードの作成 / 設定

② 各種ウィジェットの追加/設定

Ex. Button、LED、テキストボックス

③ イベントの追加

Ex. 図 3-2 の場合、各種ボタンを押したときにバックグランドでどのような動作をするかの定義

以下、順番に説明します。

5-1. ダッシュボードの作成/設定

- 使用するサービスをオープン

  まず始めにサービスをオープンする必要があります。オープンの際には “add_service dashboard” コマンドを用います。

  以下に例を記載しますので参考にしてください。

```
$ set dash [add_service dashboard dashboard_example “Dashboard Example” “Tools/Example”]↵
```

- ダッシュボードを System Console に表示

  ダッシュボードを System console に表示するために、“dashboard_set_property” コマンドを用います。

  以下に例を記載しますので参考にしてください。

```
$ dashboard_set_property $dash self visible true
```

5-2. ウィジェットの追加/設定

ウィジェットの追加について説明する前に、ダッシュボード内にはグループという概念があり、ウィジェットはグループ内に作成します。ダッシュボード作成時にはデフォルトで “self” というトップのグループが生成されます。

“self” がルートのグループとなり、グループを新たに作成する際には、“self” 以下に作成します。

図 5-1 の例では、”self” の下に “Write specify address” というグループを作成し、指定のアドレスに対し Write する機能をまとめています。

また、“Write specify address” の下に “Target Address” と “Write data” の 2 つが存在しています。

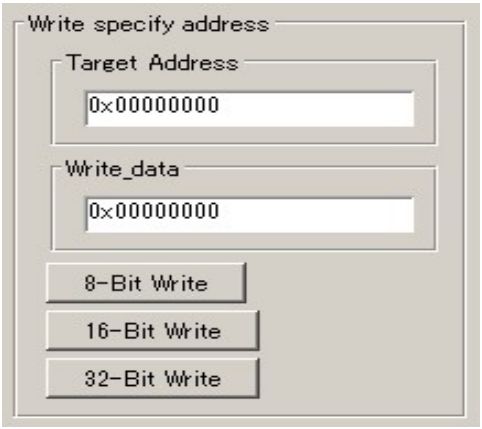


図 5-1 グループの使用例

尚、グループもウィジェットの 1 つです。

“self” 以下にグループを作成する際には、次の「ウィジェットの追加」を参照してください。

- ウィジェットの追加

  ウィジェットの追加には dashboard_add コマンドを使用します。

  ```
  $ dashboard_add <サービス> <ウィジェットにつける名前> <ウィジェットの種類> <ウィジェットを挿入するグループ>
  ```

  以下にテキストボックスを挿入する際の例を示します。

  ```
  $ dashboard_add $dash_path target_address_text text target_address
  ```

  - サービス: $dash_path
  - ウィジェットに付ける名前: target_address_text
  - ウィジェットを挿入するグループ: target_address

- ウィジェットの設定

ウィジェットを作成したら、各ウィジェットに対してオプションで設定が可能になります。

テキストボックス用ウィジェットであれば、初期表示の設定や、テキストボックスの幅の設定ができ、グループ用ウィジェットであれば、グループウィジェット内で 1 行内にいくつのウィジェットを配置するかの設定ができます。

ウィジェットの設定は dashboard_add_property コマンドを使用します。

```
$ dashboard_add_property <サービス> <対象のウィジェット> <設定>
```

以下にテキストボックス用ウィジェットを用いた際の設定例を示します。

```
$ dashboard_add $dash_path target_address_text text target_address
$ dashboard_set_property dash_path target_address_text editable true
$ dashboard_set_property $dash_path target_address_text text $target_address_d
```

この例では

`dashboard_add $dash_path target_address_text text target_address`

でテキストボックス用ウィジェットを作成し、

`dashboard_set_property $dash_path target_address_text editable true`

`dashboard_set_property $dash_path target_address_text address_input_text`

でウィジェットの設定を行っています。

`dashboard_set_property $dash_path target_address_text editable true`

はテキストボックスのテキストを編集可能として設定しています。

`dashboard_set_property $dash_path target_address_text $target_address_d`

はテキストボックスの初期値として $target_address_d を与えおり、ダッシュボード起動時にデフォルトで入力されている値になります。

5-3. イベントの追加

図 3-2 の場合、8-bit write というボタンをクリックした際に、master_write_8 コマンドが実行されていました。このように何かをトリガとして起動するコマンドを、事前にイベントとして登録しておくことができます。

- イベントの定義

  イベントは proc コマンドを用いて定義します。

  記述方法は下記の通りです。

```
$ proc <イベント名> { <コール元のサービス> <引数> }
```

  以下に例を示します。

```
$ proc toggle_led {dash_path 1}{
…イベントの内容を記載
}
```

  - イベント名: toggle_led
  - コール元のサービス: dash_path
  - 引数: 1

- イベントのコール

  ほとんどのケースでイベントをコールするのは、ボタンをクリックした際になるかと思います。その為、ボタンをクリックした際にイベントをコールする方法に絞って説明します。

  図 3-2 の場合では、8-bit write のボタンをクリックすることによってイベントをコールしていました。

  ボタン・ウィジェットに dashboard_set_property コマンドを用いて、イベントをコールするための設定を付加します。

```
$ dashboard_set_property <サービス> <ボタン・ウィジェットにつけた名前> onClick {<イベント名> <サービス> <引数>}
```

  以下に例を示します。

```
$ dashboard_set_property $dash_path button_0 onClick {toggle_led $dash_path 1}
```

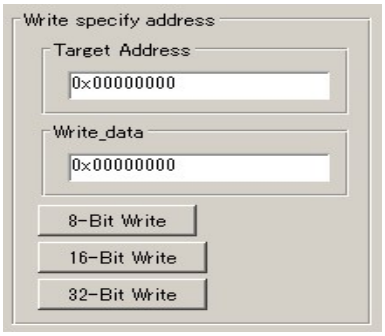
  - サービス(ボタン・ウィジェット作成の際に指定した): $dash_path
  - ボタン・ウィジェットにつけた名前: button_0
  - イベント名: toggle_led
  - サービス: 使用するサービス
  - 引数: 1

6. System Console 上における Tcl Script 実行方法

Tcl script ファイルを実行する際には、System Console 上で下記コマンドを入力して起動します。

```
$ source <Tcl Script ファイル名>
```

作成したダッシュボード GUI が表示されます。



7. 図 3-2 の記述内容

5 章 ではダッシュボードの作成手順を一通り説明しました。

この章では、一連の流れとして 図 3-2 でも使用した以下のダッシュボードの記述方法について説明します。

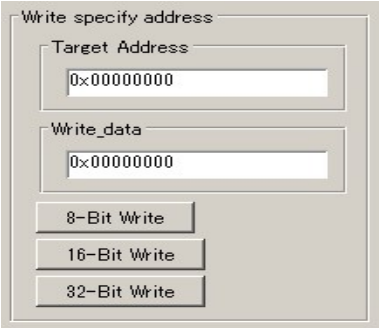


図 7-1 作成するダッシュボード

まず始めにダッシュボードの追加/設定を行います。

```
# Create DashBoard
$ set dash_path [add_service dashboard sys_dash “LED Control” “Tools/LED Control”]
$ dashboard_set_property $dash_path self visible true
```

次にグループと LED とボタン（イベントのコール設定も含む）のウィジェットを追加/設定します。

```
## Create Widget
# Create Group
$ dashboard_add $dash_path led_button_group group self
$ dashboard_set_property $dash_path led_button_group title "led_button"
$ dashboard_set_property $dash_path led_button_group itemsPerRow 1

# Create LED
$ dashboard_add $dash_path led_0 led led_button_group
$ dashboard_set_property $dash_path led_0 color red
$ dashboard_set_property $dash_path led_0 text "LED"

# Create Button
$ dashboard_add $dash_path button_0 button led_button_group
$ dashboard_set_property $dash_path button_0 text "on/off"
```

最後に、イベントの内容を記載します。

```
## Event
$ proc toggle_led { dash_path } {
$ set master_path [lindex [get_service_paths master] 1]
$ open_service master $master_path

# current led check
$ set led_read [master_read_8 $master_path 0x00000 1]

# led controle
$ if { $led_read == 1 } {
$  dashboard_set_property $dash_path led_0 color green
$  master_write_8 $master_path 0x00000  0
  }
$ if { $led_read == 0 } {
$  dashboard_set_property $dash_path led_0 color red
$  master_write_8 $master_path 0x00000  1
  }
$ close_service master $master_path
}
```

8. サンプル紹介

弊社で作成したダッシュボードのサンプルを紹介します（図 8-1）。

本 Tcl Script サンプルは、こちら からダウンロード可能です。

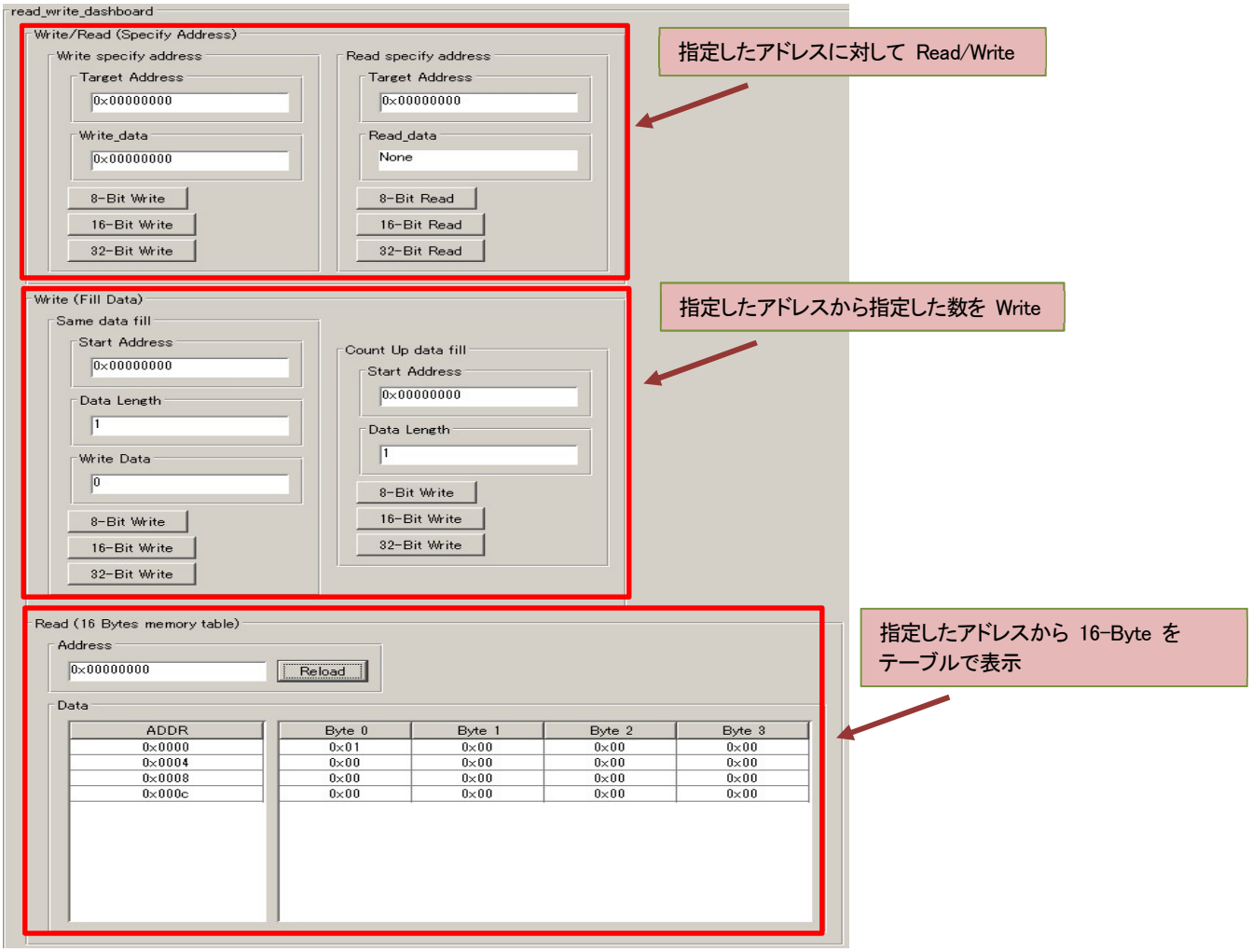


図 8-1 ダッシュボードのサンプルのユーザー・インターフェイス

> Ⓟ Point:
>
> Read/Write は 8-bit、16-bit、32-bit のいずれにも対応しています。

8-1. サンプルの機能

① 指定したアドレスに対して Read/Write

レジスターマップには Read Only や Write Only の場合があります。そのような際にこの機能は便利です。

② 指定したアドレスから指定したデータ数を Write

指定したアドレスから指定した数を Write することができます。

Write するデータ値は、指定した固定値か、+1 のインクリメント・データの数字を選択することができます。

On-Chip RAM メモリのデータを埋めたい場合等に便利です。

③ 指定したアドレスから 16-Byte をテーブルで表示

指定したアドレスから一度に多くの範囲のデータを確認できます。

On-Chip RAM のデータを一度に広い範囲で確認したい際に便利です。

それぞれの機能について、ダッシュボード上のユーザー・インターフェイスを紹介します。

8-2. 指定したアドレスに対して Read/Write

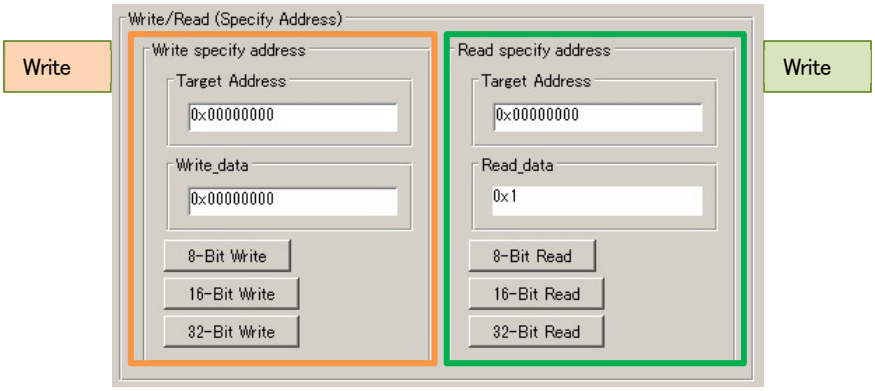


図 8-2 指定したアドレスに対して Read/Write

Write は Target Address、Write data ボックスを設定後、8-bit/16-bit/32-bit Write のいずれかのボタンをクリックして実行します。

Read は Target Address ボックスを設定し、8-bit/16-bit/32bit Read のいずれかのボタンをクリックして実行します。Read したデータは Read data で確認できます。

8-3. 指定のアドレスから指定した数を Write

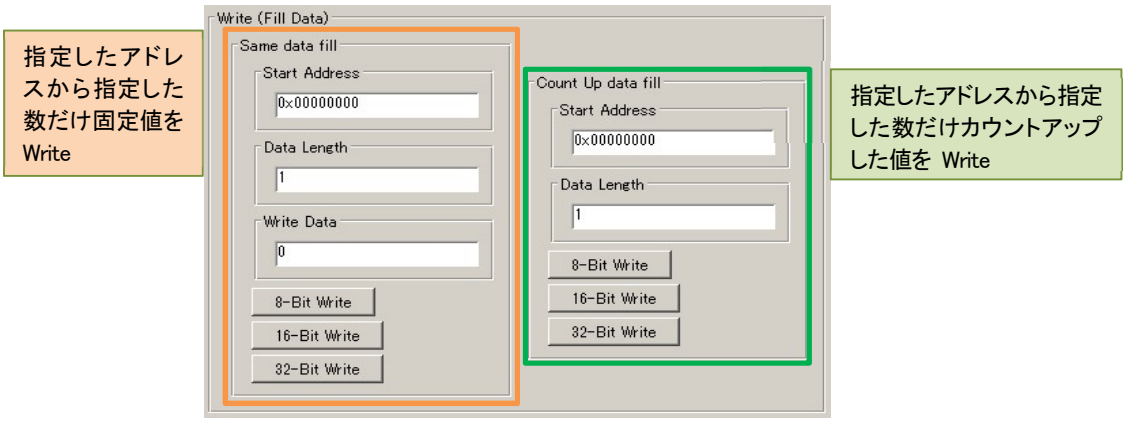


図 8-3 指定したアドレスから指定した数を Write

図 8-3 の左側は、Start Address ボックスで先頭アドレス、Data Length ボックスで Write する数、Write Data ボックスで書き込む固定値データを指定し、8-bit/16-bit/32-bit のいずれかのボタンをクリックすることで Write を実行します。

図 8-3 の右側は、Start Address ボックスで先頭アドレス、Data Length ボックスで Write する数を指定し、8-bit/16-bit/32-bit のいずれかのボタンをクリックすることで Write を実行します。データは 0 から +1 のインクリメント・データカウントアップが Write されます。

8-4. 指定したアドレスから 16-Byte をテーブルで表示

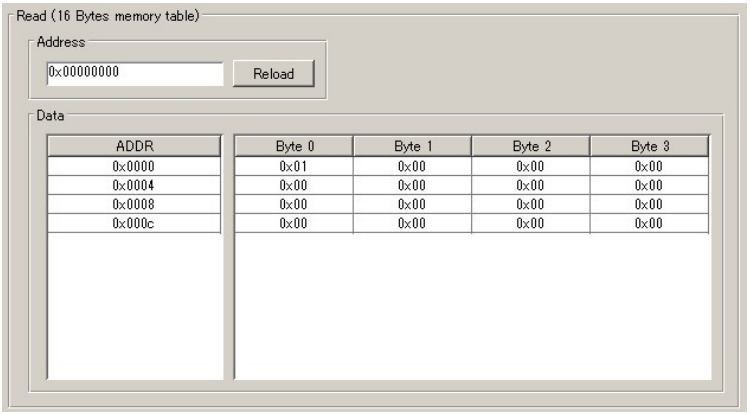


図 8-4 指定したアドレスから 16Byte をテーブルで表示

Address ボックスに先頭アドレスを設定し、Reload ボタンをクリックすると、テーブルに 16-Byte 分のデータが一覧で表示されます。

改版履歴

Revision	年月	概要
1	2019 年 11 月	初版