Fujitsu Developer Suite

User Manual

Version 0.9.4.0

Fujitsu Semiconductor Europe GmbH

Graphics Competence Center

I. Introduction

• General Information

Please check the *Customer Information* section of this manual for further details about this version.

• Packages

Different packages are available supporting one or more of the following Chip Designs :

• **MB88F332 / MB88F333**	**(Indigo / IndigoL)**
• **MB88F334 / 5 / 6**	**(Indigo2N / Indigo2S / Indigo2)**
• **MB86298**	**(Ruby)**
• **MB86R02**	**(JadeD)**
• **MB86R11**	**(EmeraldL)**
• **MB86R12**	**(EmeraldP)**
• **MB8AC0440**	**(Triton)**
• **MB86R91**	**(ApCo)**

Depending on the Chip Designs there are one or more Connection Types possible :

• **SPI**	(Serial Peripheral Interface)
• **PCIe**	(Peripheral Component Interconnect Express)
• **JTAG**	(Joint Test Action Group)
• **ETHERNET**	(Ethernet with limited access via SSH connection)
• **E2IP**	(Special Ethernet Protocol based on UDP)

1. System Requirements

a. Software

Supported OS

Windows 2000 Professional, Service Pack 4
Windows XP Professional or Home Edition, Service Pack 2 or higher
Windows Vista™ (32 / 64 bit)
Windows 7 (32 / 64 bit)

Additional Requirements

Microsoft® .NET Framework 2.0
Full Administrator Rights
Local Installation

b. Hardware

Processor

32-bit / 64-bit Intel Pentium or equivalent Processor

Available Hard-Disk Space

600 MB for Program Files and Help
50 MB for Microsoft .NET Framework (when not already installed)

RAM

Minimum Requirement is 1024 MB (for program only)

Network

Network compatible

Resolution

Recommended Resolution is 1280 x 1024 or higher

2. Licence Agreement

3. Disclaimer

Because the Fujitsu Developer Suite will be continuously developed it can happen that this document can contain outdated, incomplete or incorrect information.
No liability or omissions can be accepted for any inaccuracy of this manual.
Fujitsu Semiconductor reserves the right to change the specification and contents of the hardware and software described in this manual at anytime without prior notice.
No part of this document may be reproduced and/or transmitted without the permission of Fujitsu Semiconductor.

4. Trademarks

Microsoft, Windows Vista, .NET and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

APIX is a registered trademark of Inova Semiconductors GmbH.

5. Installation Procedure

The installation files are as follows,

- **setup.exe**
- **FujitsuDeveloperSuiteSetup.msi**

For installation you can either double click the *setup.exe* file or you can select and right clicking the *FujitsuDeveloperSuiteSetup.msi* file to open the context menu and select the install item.

Please follow the installation procedure.

• Installation Information

Program Files Folder :

Fujitsu Semiconductor Europe GmbH - GCC/Fujitsu Developer Suite/

Subdirectories :

/user	main user directory
/user/xxx/binary	for binary dumps
/user/xxx/sequence	for register sequences
/user/xxx/image	for images
/user/xxx/source	for generated source code
/user/xxx/doc	latest target documentation

(xxx = target device, e.g. Indigo, Ruby, JadeD, EmeraldL, ...)

Program Menu Folder :

Fujitsu Semiconductor Europe GmbH - GCC > Fujitsu Developer Suite

File Links :

Fujitsu Developer Suite	main application
Fujitsu Developer Suite User Manual	direct link to the user manual
Fujitsu Developer Suite Release Notes	information about the current and previous release versions
Link to Fujitsu Developer Suite User Directory	link to the main user directory

6. Uninstall Software

Choose *Add/Remove Software* in the *Windows Control Panel*.
(Start Menu > Settings > Control Panel > Add/Remove Software)

Press the *Remove* button and follow the uninstall instructions.

7. Software Activation

For software activation please contact us.

8. Contact

Homepage :

http://www.fujitsu.com/emea/services/microelectronics/gdc/contact.html

E-Mail :

gdc_tech_support.fseu@de.fujitsu.com

9. Startup Information

- ## First Time Start

On all application starts it will be checked if the software is authorized.
If this is not the case an error message appear and the application stops processing.
When this happens please contact us.

Up to four instances can be opened at a time to get connected to different Fujitsu Devices.
Current limitation is that the two in parallel supported Fujitsu Devices are NOT allowed to be of the same Type.

On the first start there is no project file loaded and the application appears empty.
Load a chip specific project

Menu Bar :   File -> Open -> Project / Solution

and select the required file.
After the file is opened the *Register Debugger* as well as the other tools and features appear.

Because it is possible that more than one Connection Device (e.g. Aardvark on SPI) is available the corresponding Device List has to be refreshed.
After selecting the correct Connection Device from the list it is possible to establish a connection to the Fujitsu Target.

- ## Registry Settings

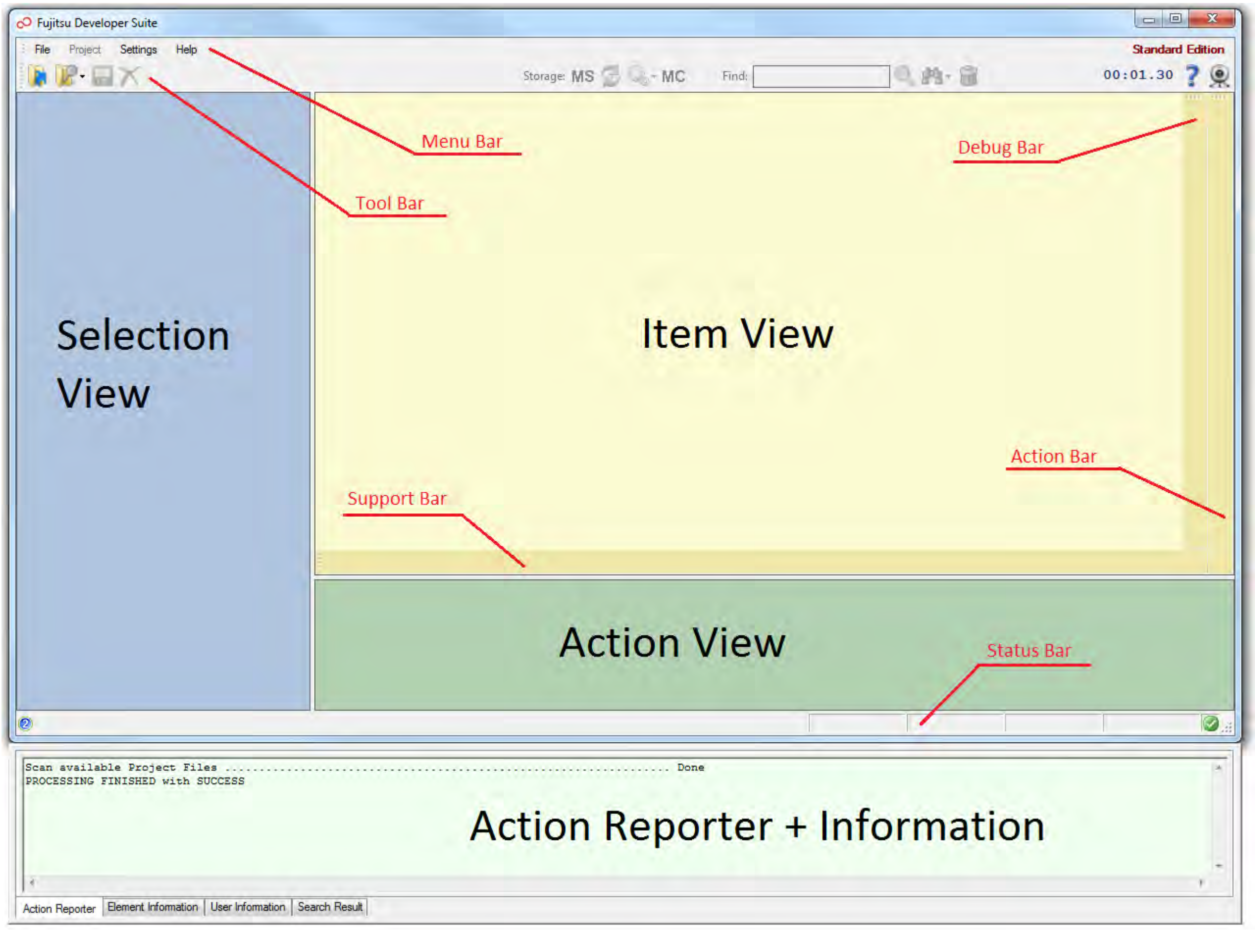
When the application starts the first time then default settings will be written into the *Windows Registry*.

Main Registry Path :

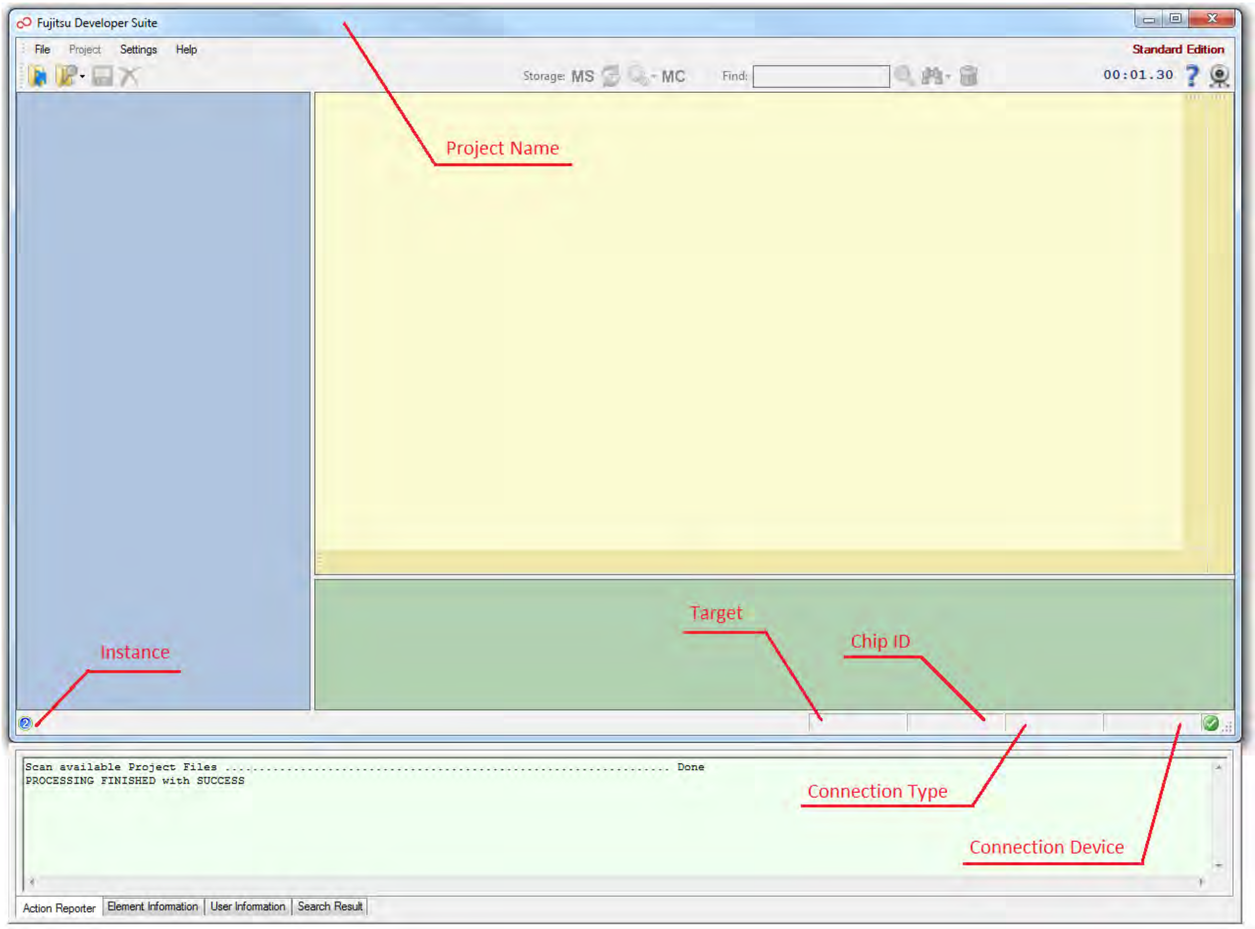
[HKEY_LOCAL_MACHINE]¥¥SOFTWARE¥¥
Fujitsu Semiconductor Europe GmbH - GCC¥¥Fujitsu Developer Suite

These values will be reset to default when a new release version is installed.

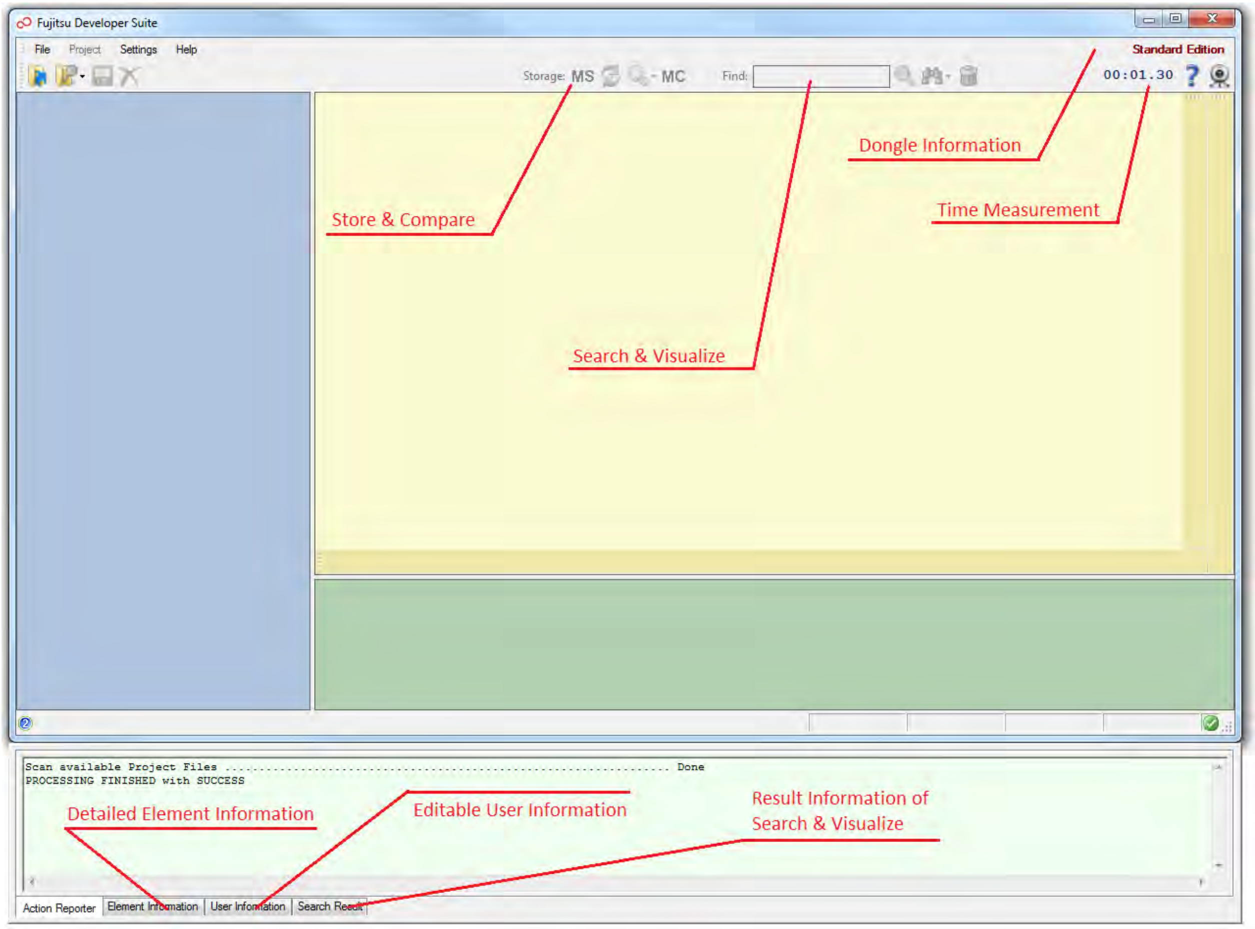
II. Overview



```
Scan available Project Files .............................................................. Done
PROCESSING FINISHED with SUCCESS
```

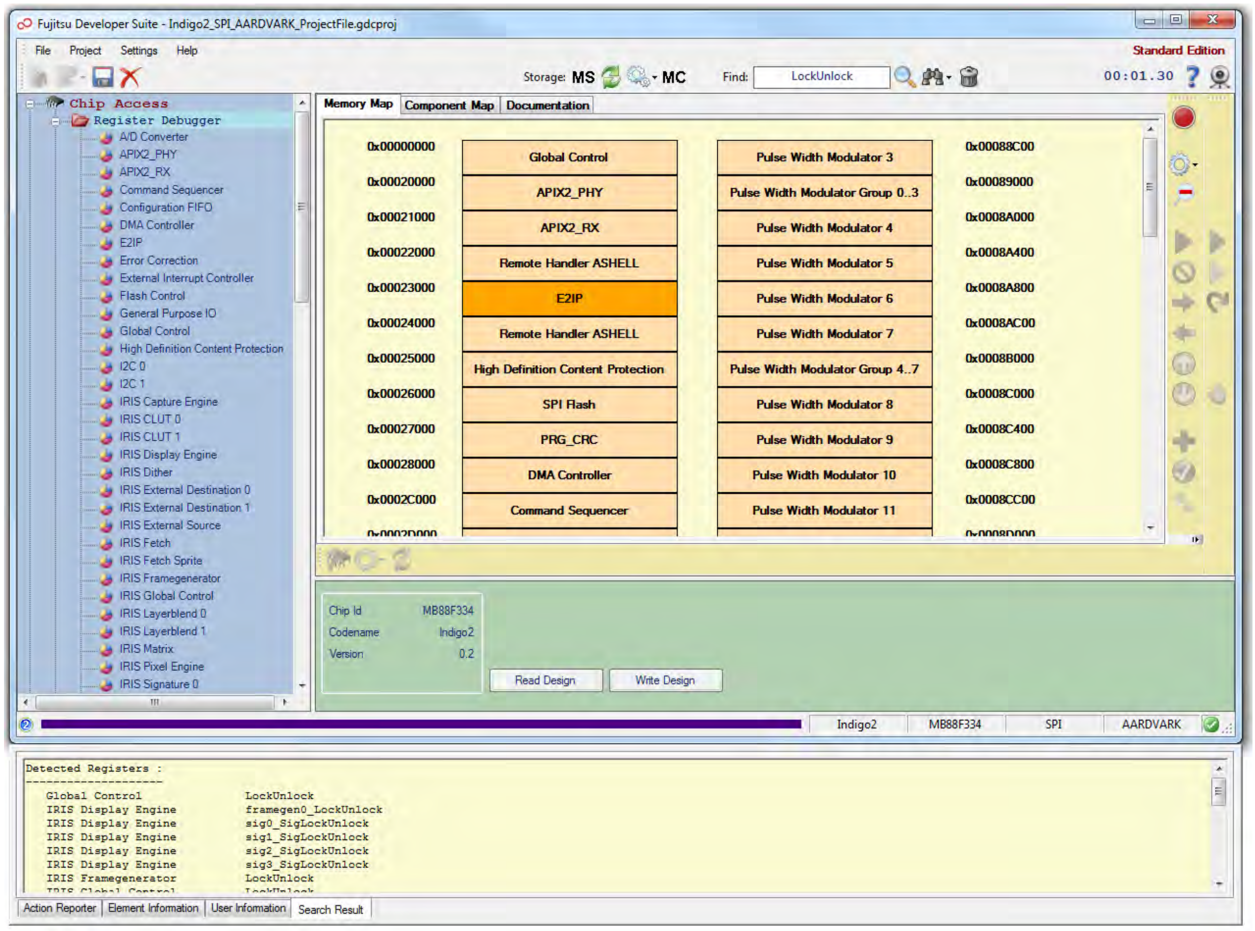


```
Scan available Project Files ........................................................... Done
PROCESSING FINISHED with SUCCESS
```



```
Scan available Project Files .......................................................... Done
PROCESSING FINISHED with SUCCESS
```

Example : Indigo 2



Chip Access
- Register Debugger
  - A/D Converter
  - APIX2_PHY
  - APIX2_RX
  - Command Sequencer
  - Configuration FIFO
  - DMA Controller
  - E2IP
  - Error Correction
  - External Interrupt Controller
  - Flash Control
  - General Purpose IO
  - Global Control
  - High Definition Content Protection
  - I2C 0
  - I2C 1
  - IRIS Capture Engine
  - IRIS CLUT 0
  - IRIS CLUT 1
  - IRIS Display Engine
  - IRIS Dither
  - IRIS External Destination 0
  - IRIS External Destination 1
  - IRIS External Source
  - IRIS Fetch
  - IRIS Fetch Sprite
  - IRIS Framegenerator
  - IRIS Global Control
  - IRIS Layerblend 0
  - IRIS Layerblend 1
  - IRIS Matrix
  - IRIS Pixel Engine
  - IRIS Signature 0

0x00000000	Global Control	Pulse Width Modulator 3	0x00088C00
0x00020000	APIX2_PHY	Pulse Width Modulator Group 0..3	0x00089000
0x00021000	APIX2_RX	Pulse Width Modulator 4	0x0008A000
0x00022000	Remote Handler ASHELL	Pulse Width Modulator 5	0x0008A400
0x00023000	E2IP	Pulse Width Modulator 6	0x0008A800
0x00024000	Remote Handler ASHELL	Pulse Width Modulator 7	0x0008AC00
0x00025000	High Definition Content Protection	Pulse Width Modulator Group 4..7	0x0008B000
0x00026000	SPI Flash	Pulse Width Modulator 8	0x0008C000
0x00027000	PRG_CRC	Pulse Width Modulator 9	0x0008C400
0x00028000	DMA Controller	Pulse Width Modulator 10	0x0008C800
0x0002C000	Command Sequencer	Pulse Width Modulator 11	0x0008CC00

Chip Id	MB88F334
Codename	Indigo2
Version	0.2

Indigo2 | MB88F334 | SPI | AARDVARK

```
Detected Registers :
--------------------
   Global Control             LockUnlock
   IRIS Display Engine        framegen0_LockUnlock
   IRIS Display Engine        sig0_SigLockUnlock
   IRIS Display Engine        sig1_SigLockUnlock
   IRIS Display Engine        sig2_SigLockUnlock
   IRIS Display Engine        sig3_SigLockUnlock
   IRIS Framegenerator        LockUnlock
```

1. Menu Bar

The *Menu Bar* represents the main interface for the user concerning the application.

a. File Menu

- ### Open

  - **Project / Solution**

    Allows opening a *Project / Solution File* of the following types,

    .gdcprojs
    encrypted standard project file which includes all relevant information of a project

    .gdcproj
    standard project file which includes all relevant information of a project

    .gdcdefproj
    project development type which require additional description .xml files

    It is only allowed to open a project when no other project is currently opened.

- ### Open / Scan

  - **Rescan Project**

    Start scanning for all Project Files located in the main directory of the application.

    .gdcprojs
    encrypted standard project file which includes all relevant information of a project

    .gdcproj
    standard project file which includes all relevant information of a project

    .gdcdefproj
    project development type which require additional description .xml files

  - **Entries ...**

    After successful scanning all detected Project Files will be listed for easy selection.

- ### Save Project

  Choose this menu item if the current open project should be saved.
  Only files can be stored with the extension .gdcproj or .gdcprojs because they include all required project information.

  It is only allowed to store a project when a project is already opened.

- ### Save Project As ...

Choose this menu item if it is required to store a special project configuration or setup under a different name.
Only files can be stored with the extension .gdcproj or .gdcprojs because they include all required project information.

It is only allowed to store a project when a project is already opened.

- **Close Project**

Close the current project.

It is only allowed to close a project when a project is already opened.

- **Exit**

Leave Application.

b. Project Menu

- **Remember Hardware Connection**

Saves the current *Hardware Connection Device Identifier / Serial Number* which was selected in the *Action Bar*.
Storing an identifier can be done for each project file separately.

When a project file is loaded where the *Hardware Connection Device Identifier* was stored before the connection can be established at once without re-scanning the bus for all available devices.

Unchecking the item in the *Menu Bar* will reset the stored identifier information.

This item is only available when a valid *Hardware Connection Device* was selected in the *Action Bar*.

c. Settings Menu

• Set As Startup Project ...

Set the current active project as *Startup Project*.
By doing this the project will be automatically loaded when the application is started.

The *Startup Project* will be reset to none when either the project file is not available, the current active project is manually closed or the menu item is unchecked.

This item is only available when a project is currently active.

• Fade-In on Start

Enable / Disable the *Fujitsu Logo* fade-in functionality.

• Auto-Project-Scan on Start

Enable / Disable the automatic project file scan functionality on application start.

• Ignore Startup Warning / Error Messages

When startup warning or error messages are displayed which are not relevant for further processing they can be disabled with this item.
(e.g. Font availability warnings)

• Prefer Standard Flashing when available

Typically on chip designs there is standard flashing available.
On specific chip designs there is an advanced flashing program available which will be preferred on default when it is detected - standard flashing is then not possible.
In some situations it is nevertheless required to prevent loading advanced flashing and and preferring standard flashing.
By enabling this item standard flashing will be preferred and advanced flashing will be suppressed.

• Suppress Default Chip Initialization on Connection when possible

When connecting to the target chip some typical initializations are done by the Developer Suite to setup the system - e.g. enable clocks, unlock all locked registers etc.
In some situations it may be required to suppress this initialization.
But be careful - it is possible that some parts of the chip cannot be accessed without initialization.
So when suppressing the initialization the user must ensure its own chip configuration e.g. by executing a user defined sequence in the Register Sequencer.

- **Always On Top**

  Activate / Deactivate the property which allows the application to always be on top of other applications.

d. Help Menu

- **Index**

  Open the *Index Page* of the *HTML Help File*.

- **Contents**

  Open the *Contents Page* of the *HTML Help File*.

- **About Fujitsu Developer Suite**

  Open the *About Box* with detailed information of the current version.

- **Check for Updates**

  Scanning the Fujitsu Server for the latest Fujitsu Developer Suite version.
  When a newer version is detected it can be downloaded and installed.

e. Edition Menu

On the right side of the menu bar the edition menu can be seen.
It depends on the purchased Version.

Basic Edition

Free Version.
Only basic memory/flash access is possible.
No other features available.

Standard Edition

Only available with USB Hardlock / Key.
Features available :

Register Debugger
Register Sequencer
Sequence Stacker
Memory Editor
Memory Dump
Image Manager
Font Manager

Depending on the target chip additional features are available as Plugin(s) allowing to access the hardware on a functional level to simplify usage and setup.
For more details please refer to the next chapters.

• License Information

Here all available chips are listed.
Depending on the license the current authorized chip names are checked.

• Update License

This item can only be selected when an USB Key is detected and a Standard or Professional Edition is available.
When a license update is required (Edition and/or Target Support) then please contact Fujitsu Semiconductor Europe.
A license update file (.v2c) will be sent which can be selected by this menu item to upgrade the license information in the Key.
After that please restart application to ensure that new license is detected correctly.

Remark :

- when removing USB Key while executing the application it will be stopped automatically and requests for the corresponding Key
- after changing / inserting Key with a different license please restart application to ensure that the correct license information is displayed

2. Tool Bar

The *Tool Bar* contains different controls which can be used directly without stepping through the *Menu Bar*.

The following controls are currently implemented,

Project

Open a *Project / Solution File* of type .gdcproj or .gdcdefproj.
It is only allowed to open a project when no other project is currently opened.

Scanning all *Project / Solution File* located in the main application folder of type .gdcprojs, .gdcproj or .gdcdefproj.
After scanning has been finished all detected projects will be listed under this toolbar item for easy selection.
Scanned items will also available in the Menu Bar.

Save the current active *Project / Solution* in a selectable or new file with the extension .gdcproj.
It is only allowed to store a project when a project is already opened.

Close the current active project.
It is only allowed to close a project when a project is already opened.

Value Storage

Storage :

Planned for easy up debugging and comparing register values.
A current Register Set - either a single IP or the whole chip can be stored in the "Stored Value" field in each page.
Then hardware changes can be forced by sequences or external triggers and the new values can be read back.
Now the values can be compared - that means "Register Value" and "Stored Value" will be compared and marked so that differences can be easily visualized .

MS Store current Register Debugger Set into the "Stored Value" field of the corresponding page.

Compare Values stored in the "Stored Value" field with the current values in the "Register Value" field of the corresponding Register Debugger page.

Set Options for the Compare between the "Register Value" and the "Stored Value".

MC

Clean the current marked registers from the last compare.

Search Register/Field

Find: c

Search for Register and Field Names.
Enter text that will be searched in either all registers and fields or the registers/fields of a specific component.
Press enter to start search through the register debugger elements.
All detected registers/fields will be marked with a specific color.
Furthermore the search results will be displayed in the Action Reporter, Search Page.

Press button to start search.

Press button to clean marked registers/field.

Contain different search options.
Start with ... :
Searching for content that starts with the entered text fragment.
Contain parts of ...
Searching for content where the entered text is find on any position.
Identical with ...
Searching for content which is identical to the entered text.

Other Helpers

00:10.05

Automatic Stopwatch.
Measures the time between specific hardware action tasks.
Format: **Minutes : Seconds . Milliseconds**
e.g.
Hardware Connection, Disconnection and Scan
Memory Block Read/Write
Flash Block Read/Write

?

Context Sensitive Help.
By pressing this button a control can be selected.
After selecting the corresponding page of the help file will be opened automatically when an article is available.

Snapshot / Capture Window.
This button will capture an image from the current application and save it to a user defined position and filename.

3. Status Bar

The *Status Bar* typically contains essential status and process information.
This bar is located on the bottom of the application.

Currently the following information is supported,

- **Last Hardware Access Action**

This icon will be displayed when no action was performed.

When this symbol appears then the last action, e.g. read / write access failed its execution.

When this symbol appears then the last action, e.g. read / write access was successfully executed.

- **Progress Status Information**

This icon will be displayed when scanning / checking is in progress.

If displayed the destination / target will be removed or erased.

This is a special icon that illustrates flashing of the *Flash Memory Chip*.

Furthermore the *Status Bar* contains information about the currently loaded Target, Chip ID, Connection Type and Connection Device.

4. Action Bar

The *Action Bar* supports the user with fast access to specific actions that can be done with the selected item in the *Selection View* and / or with the selected item in the *Item View*.

a. Always Available Controls

• Hardware Connection / Disconnection

When this symbol is highlighted on the connection button the application is currently connected to the application.
Press Button to disconnect from hardware.

If this symbol is highlighted on the connection button then the connection between application and hardware is not established.
Press Button to connect to target hardware.

Attention :
The connection to a Fujitsu Target can only be established when the corresponding Connection Device is selected (see below "Configure Hardware Connection")

• Configure Hardware Connection

Because more than one Connection Devices - which are used to establish a connection between the PC and the Fujitsu Target Device - can be available it is possible to scan the ports.
This can be done by refreshing the Device List.
All detected Devices will then be listed in the combo box with an Unique Identifier specifying the Device - typically the Serial Number.
Now the required Connection Device must be selected.

• Last Action Access Reporter

By pressing this icon the *Action Bar* will open a docked dialog with information to the last error that happened while accessing the hardware.
This symbol only appears when the *Last Action Access Dialog* is closed.

Pushing this button will close the currently open *Last Action Access Dialog*.
This symbol only appears when the *Last Action Access Dialog* is open.

b. Register Debugger Controls

By pressing this button all register of the current selected active (highlighted) item in either the *Selection View* or the *Item View* will be read from hardware.

That means when the active view is the

Selection View, and the selected item is the

Register Debugger

All registers of the current design (e.g. Indigo) will be read out from hardware.

Component / IP

All registers of the current component will be read out from hardware

Item View, and the selected item is a(n)

Address Block

All registers of the selected address block will be read out from hardware.

Attention :
Address Blocks are no longer supported from Version 1.0.1.0

Register

The required register will be read out.

Field

Here also the whole register will be read out, but only the concerning field value will be updated.
This is for storing and handling reasons.

By pressing this button all registers of the current selected active (highlighted) item in either the *Selection View* or the *Item View* will be written to hardware - similar to the previous button.

Not available.

No available.

This Button is available on the *Register Debugger* but is required for the *Register Sequencer*.
The selected (highlighted) register in the *Item View* will be added to the currently active register sequence.
When a Component/IP is selected in the *Selection View* then it is also possible to add all register of this component at once.

c. Register Sequencer Controls

Run / Play the complete register sequence which is currently selected in the *Selection View*.

Prevent the current selected item from execution.
When the item is selected with this icon it will be greyed out.

A register sequence consists of a special order of different items.
By pressing this button the currently selected register item in the *Item View* will be moved one step of the order upwards.
This is required for a proper sorting of the register sequence.

A register sequence consists of a special order of different items.
By pressing this button the currently selected register item in the *Item View* will be moved one step of the order downwards.
This is required for a proper sorting of the register sequence.

By pressing this icon a *User Defined Sequencer Item* will be inserted into the active sequence.

To copy a sequence item.
Places the copy directly below the original one.

This button will remove the currently selected register item in the *Item View* from the sequence.

d. Sequence Stacker Controls

▶	Run / Play the complete sequence stack which is currently selected in the *Selection View*.
🚫	Prevent the current selected sequence from execution. When the item is selected with this icon it will be greyed out.
⬆	A sequence stack consists of a special order of different sequences. By pressing this button the currently selected sequence in the *Item View* will be moved one step of the order upwards. This is required for properly sorting the register sequences.
⬇	A sequence stack consists of a special order of different sequences. By pressing this button the currently selected sequence in the *Item View* will be moved one step of the order downwards. This is required for properly sorting the register sequences.
❌	This button will remove the currently selected sequence in the *Item View* from the stack.
➕	By pressing this button a new sequence can be added / loaded into the current sequence stack.

5. Support Bar

The *Support Bar* supports the user with fast access to specific actions.

a. Memory / Flash Editor Controls

When this icon is highlighted on the button then the flash support of the *Memory Editor* is deactivated.

By pressing the button above this icon will appear that illustrates that the flash support in now enabled.

This button icon is only active when flash support is enabled. It offers menu items with some special flash options.

This button allows comparing the data that is currently present in the *Memory Editor Grid* with the content of the memory / flash located at the specified address.

b. Memory / Flash Dump Controls

When this icon is highlighted on the button then the flash support of the *Memory Dump Page* is deactivated.

By pressing the button above this icon will appear that illustrates that the flash support in now enabled.

This button icon is only active when flash support is enabled. It offers menu items with some special flash options.

This button allows comparing the data that is currently present in the *Memory Dump View* with the content of the memory / flash located at the specified address.

6. Debug Bar

The *Debug Bar* supports the user with fast access to specific actions.

a. Register Sequencer Controls

▶	Executes the sequence until the next breakpoint is detected or the end of the sequence is reached.
▷	This icon is used to execute the sequence step by step. *Single Step Mode.*
↻	Stops the current debugging and returns the debug cursor to the begin of the sequence.
✋	Toggle the breakpoint property of the selected item.

b. Sequence Stacker Controls

Executes the stack until the next breakpoint is detected or the end of the sequence stack is reached.

This icon is used to execute the sequence stack step by step. *Single Step Mode.*

Stops the current debugging and returns the debug cursor to the beginning of the stack.

Toggle the breakpoint property of the selected item.

7. Action Reporter Page

The *Action Reporter* gives a short overview of errors and/or warnings that happened on the last hardware action (like read, write ...) that was executed by the user.

If the last action was successfully then the window color will be displayed in light green without any messages on it.

When an hardware access error / warning occur then the window color will be light red with a detailed explanation of the failed action.

8. Element Information Page

The *Information Page* contains the same detailed information of e.g. Register or Register Field like the popup information displayed when hovering over such an Register Debugger Element.
The advantage of this information page is that it will be displayed permanent - that means as long as a specific Element is selected.
This is very helpful with big explanations or when changing values of an element in the Action View.

9. Selection View

The *Selection View* contains the main tools that are available with the current project / solution in combination with the available hardware design.

Currently the *Selection View* supports the following base items,

Chip Access

- **Register Debugger**
- **Register Sequencer**
- **Sequence Stacker**
- **Memory / Flash Editor**
- **Memory / Flash Dump**

Tools

- **Image Manager**
- **Font Manager**

For more details to the tool please refer to the corresponding chapters.

Depending on the supported Chip and Connection Type different additional Managers, Tools and Helpers are available.

10. Item View

The *Item View* contains specific available items corresponding to the selected / highlighted item in the *Selection View*.

For more details to the *Item View* please refer to the corresponding tool chapters.

11. Action View

The *Action View* contains specific available items corresponding to the selected / active item in the *Selection View* or in the *Item View*.

For more details to the *Action View* please refer to the corresponding tool chapters.

III. Register Debugger

Purpose

The main purpose of the *Register Debugger* is to manipulate and review hardware. Furthermore while development it can help to validate software and hardware design.

Hardware Cover

The *Register Debugger* cover all (or at least the essential) hardware IP's of a specific design and allows reading and writing of register and register sets.
Depending on the Package the following Designs are supported.

MB88F334 / Indigo2 Register Debugger supports the following IP's,

- A/D Converter Unit
- APIX2 PHY
- APIX2 RX
- Command Sequencer
- Configuration FIFO
- DMA Controller
- E2IP
- Error Correction
- External Interrupt Controller
- Flash Control
- General Purpose IO (GPIO)
- Global Control
- High Definition Content Protection (HDCP)
- I2C 0
- I2C 1
- IRIS Capture Engine
- IRIS CLUT 0
- IRIS CLUT 1
- IRIS Display Engine
- IRIS Dither
- IRIS External Destination 0
- IRIS External Destination 1
- IRIS External Source
- IRIS Fetch
- IRIS Fetch Spriet
- IRIS Framegenerator
- IRIS Global Control
- IRIS Layerblend 0
- IRIS Layerblend 1
- IRIS Matrix

- IRIS Pixel Engine
- IRIS Signature 0
- IRIS Signature 1
- IRIS Signature 2
- IRIS Signature 3
- IRIS Timing Controller
- LIN
- PRG CRC
- Pulse Width Modulator 0
- Pulse Width Modulator 1
- Pulse Width Modulator 2
- Pulse Width Modulator 3
- Pulse Width Modulator 4
- Pulse Width Modulator 5
- Pulse Width Modulator 6
- Pulse Width Modulator 7
- Pulse Width Modulator 8
- Pulse Width Modulator 9
- Pulse Width Modulator 10
- Pulse Width Modulator 12
- Pulse Width Modulator 13
- Pulse Width Modulator 14
- Pulse Width Modulator 15
- Pulse Width Modulator Global Control
- Pulse Width Modulator Group 0..3
- Pulse Width Modulator Group 4..7
- Pulse Width Modulator Group 8..11
- Pulse Width Modulator Group 12..15
- RBUS ECU
- Reload Timer 0
- Reload Timer 1
- Reload Timer 2
- Reload Timer 3
- Reload Timer 4
- Reload Timer 5
- Reload Timer 6
- Reload Timer 7
- Reload Timer 8
- Reload Timer 9
- Reload Timer 10
- Reload Timer 11
- Reload Timer 12
- Reload Timer 13
- Reload Timer 14
- Reload Timer 15
- Remote Handler ASHELL
- Remote Handler E2IP
- Sound Generator
- SPI External Devices

- SPI Flash
- Stepper Motor Control 0
- Stepper Motor Control 1
- Stepper Motor Control 2
- Stepper Motor Control 3
- Stepper Motor Control 4
- Stepper Motor Control 5
- Stepper Motor Control Trigger

MB88F332 / Indigo Register Debugger supports the following IP's,

- A/D Converter Unit
- Chip Control Unit
- Clock Modulator Unit
- Clock Synthesis Unit
- Color Lookup Table Unit
- Command Sequencer Unit
- Configuration FIFO Unit
- Display Controller
- Dithering Unit
- DMA Controller Unit
- External Interrupt 0 Unit
- External Interrupt 1 Unit
- General Purpose IO Unit
- I2C Unit
- Memory Interface Unit
- Pulse Generator Unit
- Reload Timer Unit
- Remote Handler Unit
- Run Length Decoder Unit
- Signature Unit
- Sound Generator Unit
- SPI Flash Unit
- Sprite Engine Control Unit
- Sprite Engine SAT Unit
- Sprite Engine Special Sprite Unit
- Stepper Motor Controller Unit
- Timing Controller Unit
- UART Unit

MB88F333 / IndigoL Register Debugger supports the following IP's,

- A/D Converter Unit
- Chip Control Unit
- Clock Modulator Unit
- Clock Synthesis Unit
- Color Lookup Table Unit
- Command Sequencer Unit
- Configuration FIFO Unit

- Display Controller
- Dithering Unit
- DMA Controller Unit
- External Interrupt 0 Unit
- External Interrupt 1 Unit
- General Purpose IO Unit
- I2C Unit
- Memory Interface Unit
- Pulse Generator Unit
- Reload Timer Unit
- Remote Handler Unit
- Run Length Decoder Unit
- Signature Unit
- Sound Generator Unit
- Sprite Engine Control Unit
- Sprite Engine SAT Unit
- Sprite Engine Special Sprite Unit
- Stepper Motor Controller Unit
- Timing Controller Unit
- UART Unit

MB86298 / Ruby Register Debugger supports the following IP's,

- ARGES Unit
- Capture Requester Unit
- Command Sequencer Unit
- Display 0 Control Unit
- Display 1 Control Unit
- Global Controller Unit
- GPIO Unit
- I2C Unit
- Interconnect Unit
- Interrupt Controller Unit
- Memory Controller Unit
- PCI Express Host Unit
- Pixel Blitter Unit
- SPI Unit
- Timer Unit
- Unified Shader Unit
- Video Capture 0 Unit
- Video Capture 1 Unit
- Video Capture 2 Unit
- Video Capture 3 Unit
- Write Back Unit

MB86R01 / JadeD Register Debugger supports the following IP's,

- A/D Converter 0
- A/D Converter 1

- APIX Interface
- CAN 0 Interface
- CAN 1 Interface
- Chip Control
- Clock Reset Generator
- Color Lookup Table 0
- Color Lookup Table 1
- DDR2 Controller
- Dithering Unit 0
- Dithering Unit 1
- DMA Controller
- External Bus Interface
- External Interrupt Controller
- General Purpose IO
- Graphics Display Controller
- I2C Unit 0
- I2C Unit 1
- I2S Unit
- Interrupt Request Controller 0
- Interrupt Request Controller 1
- Interrupt Request Controller 2
- MediaLB Interface
- Pulse Width Modulator 01
- Pulse Width Modulator 23
- Pulse Width Modulator 45
- Pulse Width Modulator 67
- Remap Boot Controller
- Run Length Decoder
- Signature Generator 0
- Signature Generator 1
- SPI Unit 0
- SPI Unit 1
- Spread Spectrum Control
- Timer
- Timing Controller
- UART Unit 0
- UART Unit 1
- UART Unit 2
- UART Unit 3
- UART Unit 4
- UART Unit 5

MB86R11 / EmeraldL Register Debugger supports the following IP's,

- CAN 0
- CAN 1
- Capture 0
- Capture 1
- Capture 2

- Capture 3
- Chip Control
- Clock Reset Generator 0
- Clock Reset Generator 1
- Display Controller 0
- Display Controller 1
- External IRQ 0
- External IRQ 1
- External Bus IF
- GPIO Unit
- HDMAC Unit
- I2C Unit 0
- I2C Unit 1
- I2C Unit 2
- I2C Unit 3
- I2C Unit 4
- I2S Unit 0
- I2S Unit 1
- I2S Unit 2
- I2S Unit 3
- Pixel Engine
- Power Management Unit
- Pule Width Modulator 0
- Pule Width Modulator 1
- Pule Width Modulator 2
- RLD
- SDIO 0
- SDIO 1
- SDIO 2
- Serial Flash Interface 0
- Serial Flash Interface 1
- Signature Unit 0
- Signature Unit 1
- Signature Unit 2
- Timing Controller
- UART 0
- UART 1
- UART 2
- UART 3
- UART 4
- UART 5
- USART 0
- USART 1
- USART 2
- USART 3
- USART 4
- USART 5

MB86R12 / EmeraldP Register Debugger supports the following IP's,

- APIX PHY
- APIX RX Link
- APIX TX Link Channel 0
- APIX TX Link Channel 1
- APIX TX Link Channel 2
- CAN 0
- CAN 1
- Capture 0
- Capture 1
- Capture 2
- Capture 3
- Chip Control
- Clock Reset Generator 0
- Clock Reset Generator 1
- DDR Controller
- Display Controller 0
- Display Controller 1
- Display Controller 2
- External IRQ 0
- External IRQ 1
- External Bus IF
- GPIO Unit
- HDMAC Unit
- I2C Unit 0
- I2C Unit 1
- I2C Unit 2
- I2C Unit 3
- I2C Unit 4
- I2S Unit 0
- I2S Unit 1
- I2S Unit 2
- I2S Unit 3
- Pixel Engine
- Power Management Unit
- Pule Width Modulator 0
- Pule Width Modulator 1
- Pule Width Modulator 2
- RLD
- SDIO 0
- SDIO 1
- SDIO 2
- Serial Flash Interface 0
- Serial Flash Interface 1
- Signature Unit 0
- Signature Unit 1
- Signature Unit 2
- Timing Controller
- UART 0
- UART 1

- UART 2
- UART 3
- UART 4
- UART 5
- USART 0
- USART 1
- USART 2
- USART 3
- USART 4
- USART 5

MB8AC0440/ Triton Register Debugger supports the following IP's,

- Capture 0
- Capture 1
- Capture 2
- Capture 3
- CCNT
- CmdSeq
- CRG11 0
- CRG11 1
- DDR Controller 0
- DDR Controller 1
- Display Controller 0
- Display Controller 1
- Ethernet MAC
- Ethernet MMC
- EXIRC
- GPIO
- HDMAC
- HS SPI
- I2C 0
- I2C 1
- I2S 0
- I2S 1
- I2S 2
- I2S 3
- IRIS BitBlend
- IRIS CLUT 0
- IRIS CLUT 1
- IRIS Display Config
- IRIS Dither
- IRIS ExtDst
- IRIS Fetch 0 RLD
- IRIS Fetch 1 LIGHT
- IRIS Fetch 2 ROT
- IRIS Fetch 3 WARP
- IRIS Fetch 4 LIGHT
- IRIS Framegenerator

- `IRIS Global Control`
- `IRIS HScaler`
- `IRIS Layer Blend 0`
- `IRIS Layer Blend 1`
- `IRIS Matrix 0`
- `IRIS Matrix 1`
- `IRIS Pixel Bus`
- `IRIS Rop`
- `IRIS Signature 0`
- `IRIS Store`
- `IRIS VScaler`
- `MediaLB Phy`
- `MediaLB Register`
- `Performance Measurement`
- `PMU`
- `PWM 0`
- `PWM 1`
- `SDIO 0`
- `Serial Flash 0`
- `Serial Flash 1`
- `Signature 0`
- `Signature 1`
- `TCON`
- `UART 0`
- `UART 1`
- `UART 2`
- `UART 3`
- `UART 4`
- `UART 5`
- `USART 0`
- `USART 1`
- `USART 2`
- `USART 3`
- `USART 4`
- `WDT B`

• Store/Load Configurations

When a special hardware configuration was made in the *Register Debugger* all the values can be stored in a *Project File* to be able to return to the last setup without changing all the registers again.

After loading a specific *Project File* the stored values can be applied to the hardware either separately *(Field, Register)* or as a complete set *(Component/IP,* complete *Design)*

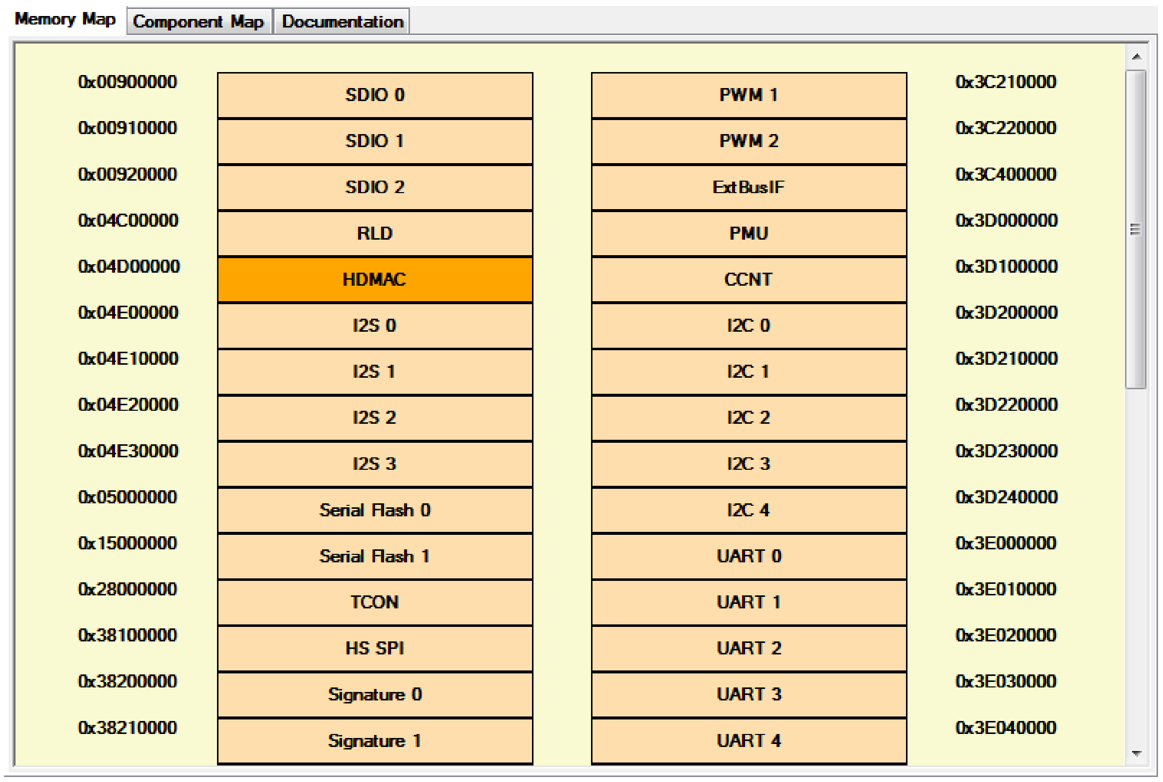
1. Memory / Component Map

The Register Debugger Page contains two helpful Maps that can be used for a better chip overview as well as a direct link into the corresponding chip components.

- **Memory Map**

  Address Map Overview.
  By pressing a component the corresponding IP in the Register Debugger will be opened.

  e.g. Emerald

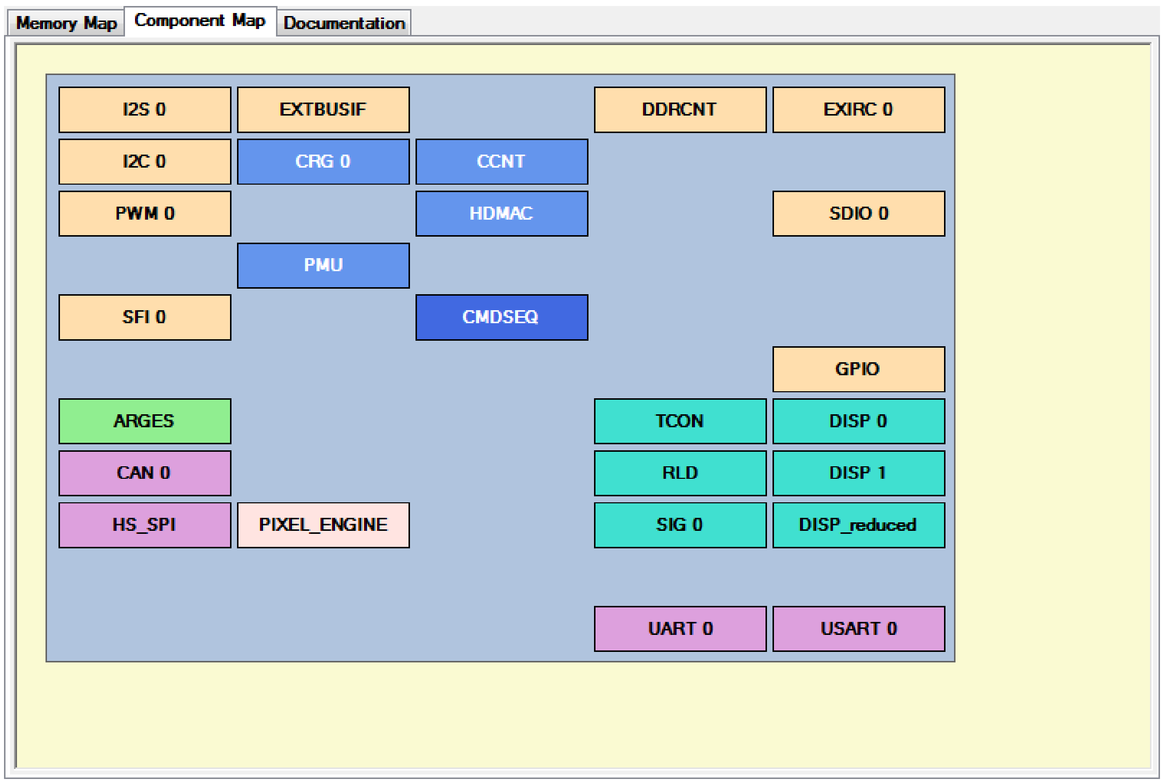


0x00900000	SDIO 0	PWM 1	0x3C210000
0x00910000	SDIO 1	PWM 2	0x3C220000
0x00920000	SDIO 2	ExtBusIF	0x3C400000
0x04C00000	RLD	PMU	0x3D000000
0x04D00000	HDMAC	CCNT	0x3D100000
0x04E00000	I2S 0	I2C 0	0x3D200000
0x04E10000	I2S 1	I2C 1	0x3D210000
0x04E20000	I2S 2	I2C 2	0x3D220000
0x04E30000	I2S 3	I2C 3	0x3D230000
0x05000000	Serial Flash 0	I2C 4	0x3D240000
0x15000000	Serial Flash 1	UART 0	0x3E000000
0x28000000	TCON	UART 1	0x3E010000
0x38100000	HS SPI	UART 2	0x3E020000
0x38200000	Signature 0	UART 3	0x3E030000
0x38210000	Signature 1	UART 4	0x3E040000

- **Component Map**

  Chip Layout Overview.
  By pressing a component the corresponding IP in the Register Debugger will be opened.

  e.g. Emerald



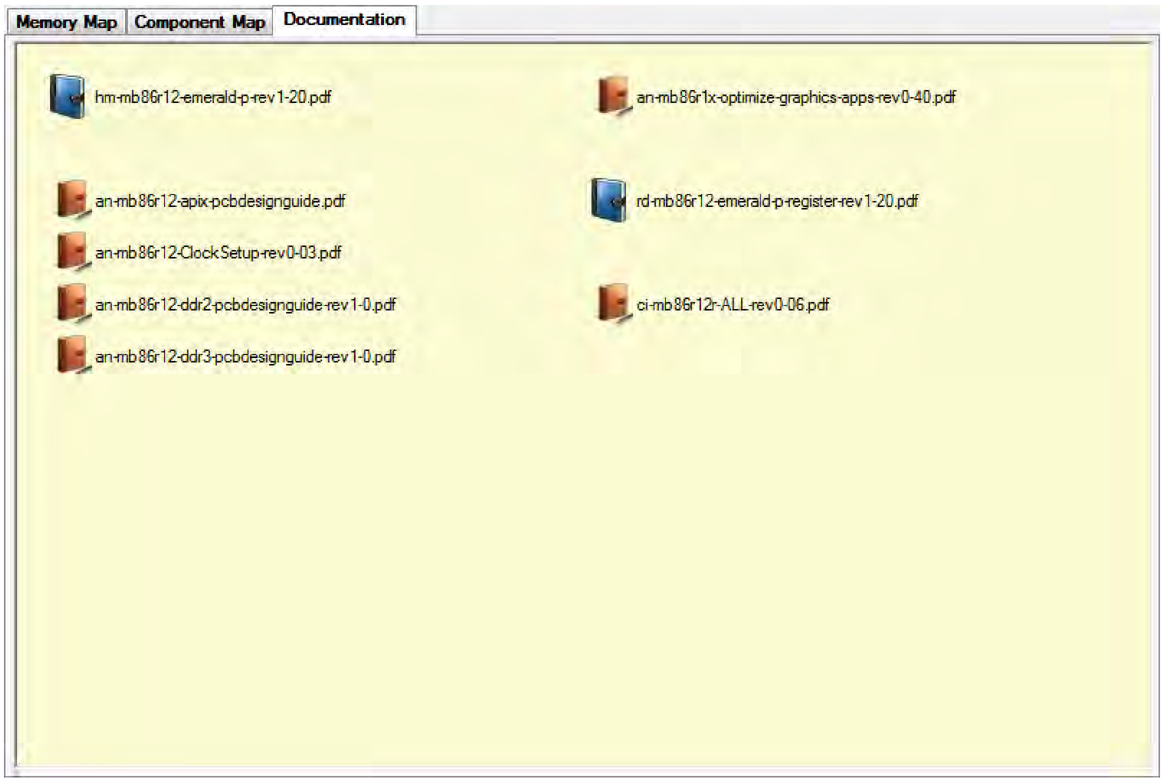
- **Documentation**

Contains latest manuals and application notes to the corresponding chip version in pdf format.

By pressing one of the documentation buttons the corresponding file will be opened and displayed to the user.

Remark : It is required that a PDF viewer is installed to read the documents.

e.g. Emerald



hm-mb86r12-emerald-p-rev1-20.pdf
an-mb86r1x-optimize-graphics-apps-rev0-40.pdf
an-mb86r12-apix-pcbdesignguide.pdf
rd-mb86r12-emerald-p-register-rev1-20.pdf
an-mb86r12-ClockSetup-rev0-03.pdf
an-mb86r12-ddr2-pcbdesignguide-rev1-0.pdf
ci-mb86r12r-ALL-rev0-06.pdf
an-mb86r12-ddr3-pcbdesignguide-rev1-0.pdf

2. Selection View

The *Selection View* contains the main tools that are available with the current *Project / Solution* in combination with the available hardware design.

Typically the *Selection View* contains the following sub-items for the *Register Debugger*,

- **Component**

  Each *Register Debugger* of a specific hardware design contains a set of available *Components / IP's*.
  The components are listed as sub-items of the register debugger item.

3. Item View

The *Item View* contains specific available items corresponding to the selected / highlighted item in the *Selection View*.

The following sections are displayed for the *Register Debugger* in the *Item View*.

Name Section

List names of the available *Address Blocks*, *Registers* and *Fields*.

- **Register**

  A *Component / IP* typically contains a set of *Register(s)* which are displayed as sub-item with the specified icon.

- **Register - Lock / Unlock Key**

  A special Register which is required to Lock / Unlock a set of other registers.
  For locking/unlocking a special key is required which must be programmed in this register.

- **Register - Lock / Unlock Key Status**

  A special Register displaying the status of the current Lock/ Unlock mechanism.

- **Field**

  A *Field* is a bit or a bit-field corresponding to the register which it holds.

Type Section

Describe the type of a field, that can be

• **R**	Read
• **W**	Write
• **RW**	Read / Write
• **RSVD**	Reserved
• **W1C**	Write Once
• **RW1C**	Read / Write Once
• **RWX**	Read / Write

Bit Offset Section

Describe the offset of the first bit of the field within the register.

Bit Width Section

Describe the width of the field.

Field Value Section

Value of the field. (decimal)

Register Value Section

Value of the Register. (Hexadecimal)

Error Section

When an error occurs on the last hardware access this section contains a status letter,

• **X**	Error
• **W**	Warning
• **I**	Information

Keyboard Controls

Backspace	Read current selected item
Enter	Select write edit field
Tab	Select next user control
Up / Down Arrow	Step to previous / next item
Left / Right Button	Expand / Collapse item
Any other Letter	Step to the next item that starts with this letter

4. Action View

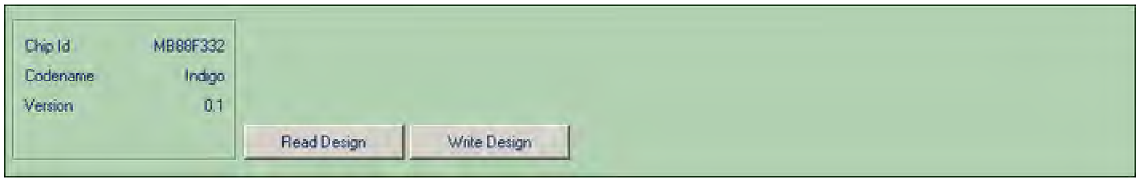
The *Action View* contains specific available items corresponding to the selected / active item in the *Selection View* or in the *Item View*.

Typically it contains some information about the item as well as some controls which offer special actions.

If a so-called *Field* item is selected in the *Item View*, then (depending on the field size) some advanced controls like *Edit Boxes*, *Sliders*, *List Views* ...appear and offer hardware read and write actions.

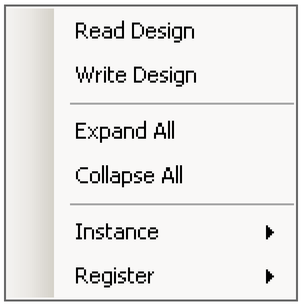
a. Design Page

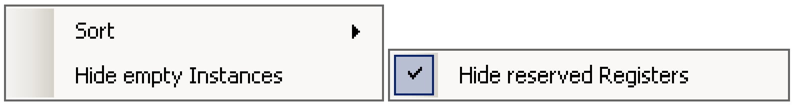
When the *Register Debugger* item is selected on the *Selection View* then relevant design information as well as some possible action controls will appear on the *Action View*.



Chip Id	MB88F332
Codename	Indigo
Version	0.1

A Context menu is also available by pressing the right mouse button on the Register Debugger item in the Selection View.





Expand All and *Collapse All* menu items expands/collapses all Register Items of the complete design.

There are also some display options available,

Instances

- *Sort* (to sort the instance list below the Register Debugger node)
  Mode -> by Name or Address
  Order -> normal or reverse order
- *Hide empty instances* (to prevent empty instances of being displayed)

Register

- *Hide reserved registers* (to prevent reserved registers of being displayed)

Read Design Button

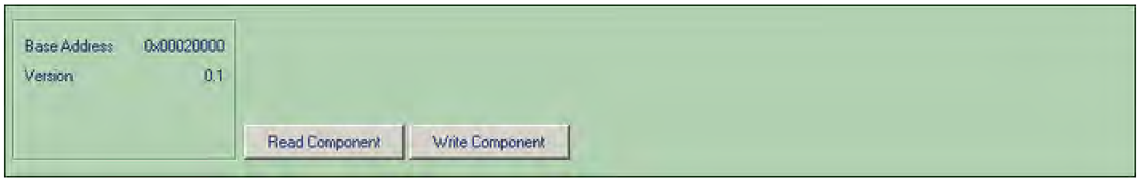
Read all registers of the current *Chip Design*.

Write Design Button

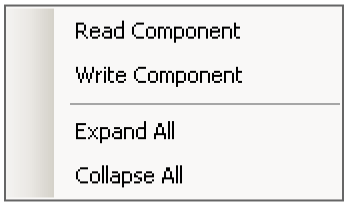
Write all registers of the current *Chip Design*.

b. Component Page

When a *Component* (sub-item of the *Register Debugger*) is selected on the *Selection View* then relevant *Component / IP* information as well as some possible action controls will appear on the *Action View*.


Base Address	0x00020000
Version	0.1

A Context menu is also available by pressing the right mouse button on a Component / IP Item in the Selection View.



Expand All and *Collapse All* menu item expands/collapses all Register Items of the current Component.

Read Component Button

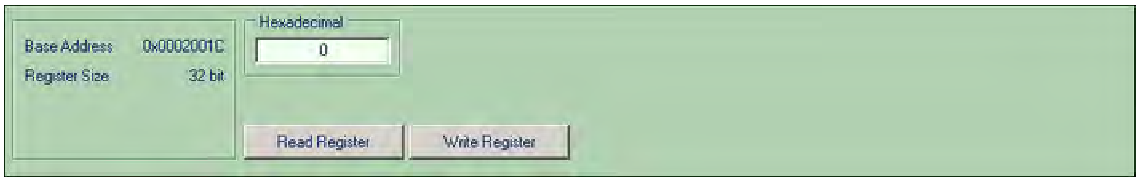
Read all registers of the current *Component / IP*.

Write Component Button

Write all registers of the current *Component / IP*.

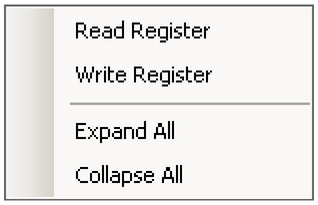
c. Register Page

When a *Register* is selected on the *Item View* then relevant information as well as some possible action controls will appear on the *Action View*.



Base Address 0x0002001C
Register Size 32 bit
Hexadecimal 0

A Context menu is also available by pressing the right mouse button on a Register Item.



Expand All and *Collapse All* menu item expands/collapses the Fields of the Register.

Read Register Button

Read current register.

Write Register Button

Write current register.

Hexadecimal Edit Field

To manipulate a register you can either press the *Return Key* or you can directly select the *Hexadecimal Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value, the *Enter Key* must be pressed to write the new value to the register.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

Element Edit Field

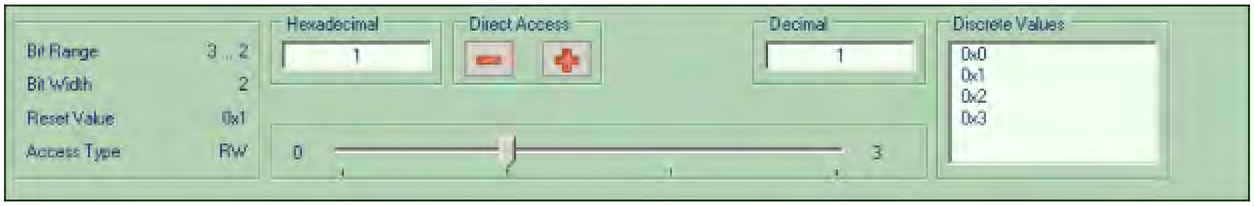
When there is a group of registers / register-sets which are followed by absolutely identical registers / register-sets then only the first will be displayed by default.
In other words such a register / register-set is considered as *Element* where the element with the ID = 0 is selected as default.
Only in this case the *Element Edit Field* appears.

If access to another similar register / register-set is required then the element number can be set to the one which should be displayed.

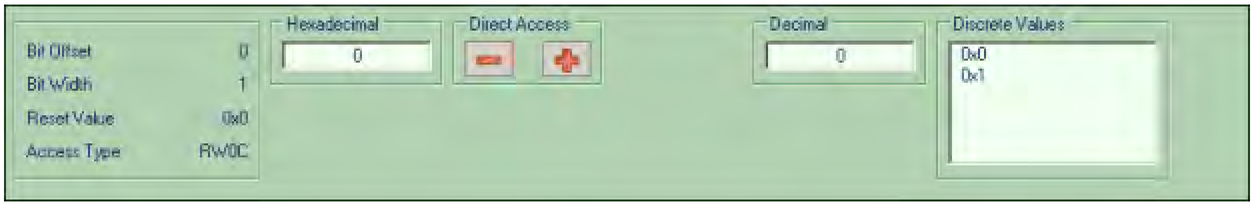
Don't forget to read out the register when the element number has been changed, otherwise the old content is still displayed.

d. Field Page

When a *Field* is selected on the *Item View* then relevant information as well as some possible action controls will appear on the *Action View*.



Bit Range	3 .. 2
Bit Width	2
Reset Value	0x1
Access Type	RW



Bit Offset	0
Bit Width	1
Reset Value	0x0
Access Type	RW0C

Hexadecimal Edit Field

To manipulate a *Register Field* you can either press the *Return Key* or you can directly select the *Hexadecimal Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value, the *Enter Key* must be pressed to write the new value to the register.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

Decimal Edit Field

To manipulate the *Register Field* with a decimal value you need to select the *Decimal Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value the *Enter Key* must be pressed to write the new value to the register.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

Field Value Slider

In some cases - that means when the bit width of a *Field* is between 2 and 8 - then an additional slider will appear to manipulate the *Register Field* easily by mouse.

Change the value with the slider - by pressing the left mouse button and dragging the mouse - as required.

To acknowledge the new value simply right clicking the mouse on the slider control.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

Field Entry List View

In some cases - that means when discrete values of a *Field* exists - then a *List View* will appear.
This list contains some discrete values which are allowed for that field and resembles some specific settings.

If one of these values are required select them with the left mouse button and acknowledge the change with the right mouse button.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

Direct Access Button

There are two direct access buttons available.

➕	Increases the current field value by one and automatically writes it to the hardware.
➖	Decreases the current field value by one and automatically writes it to the hardware.

IV. Register Sequencer

• Purpose

The *Register Sequencer* is required to create sequences of register access events.
These sequences are helpful to store and load special hardware configurations like the initialization of a LCD panel, the setup of a video mode and so on.

• Drag & Drop

It is possible to drag & drop *Sequence Files* into the application.
This can be done by selecting a sequence file in the window explorer and dragging the file onto the *Item View*.
This can only be done when the *Register Sequencer Item* is selected in the *Selection View*.
(Multiple File Selection)

1. Selection View

The *Selection View* contains the main tools that are available with the current *Project / Solution* in combination with the available hardware design.

Register sequences can be stored and loaded separately.

The *Register Sequencer* can contain none or more register sequences that describe a specific behavior - for example a special initialization sequence.

Typically the *Selection View* contains the following sub-items for the *Register Sequencer* :

- **Register Sequence**

  The *Register Sequencer* can contain none or more register sequences that describe a specific behavior.
  This symbol specifies an available register sequence.

- **Active Register Sequence**

  When a sub-item of the *Register Sequencer* has such a symbol it is marked as the *Active Register Sequence*.
  That means that it is possible to add registers from the *Register Debugger* to this sequence.

2. Item View

The *Item View* contains specific available items corresponding to the selected / highlighted item in the *Selection View*.

The following sections are displayed for the *Register Debugger* in the *Item View*.

Register Name Section

Identify the register by its name.

- **Write Element**

  Write the Register Value with the given Size to the specified Address.

- **Read Element**

  Read the Register Value with the given Size from the specified Address.

- **Poll for Read Element**

  Reads the Register Value with the given Size from the specified Address until,
  - User Condition is getting TRUE :
    Value & Mask == Mask
    This is useful to check Status Register Bits/Flags,
    e.g. Value=0x00AB, Mask=0x0800 -> FALSE -> read again ...
    e.g. Value=0x0FAB, Mask=0x0800 -> TRUE -> continue
  - the number of *Poll Cycles* are reached
  - an error occurs

- **Poll for Target Element**

  Reads the Register Value with the given Size from the specified Address until,
  - User Condition is getting TRUE :
    Value & Mask == Target
    This is useful to check Bits/Flags with highest flexibility.
  - the number of *Poll Cycles* are reached
  - an error occurs

- **Write Repeat Element**

  Write the Hexadecimal Value to the specified Address.
  This will be done until Repeat Count is reached.

- **Write Repeat Increment Element**

  Write the Hexadecimal Value to the specified Address.
  Now the Address will automatically incremented and the same Value will be written to the new Address.
  This will be done until Repeat Count is reached.

- **Compare Repeat Increment Element**

  Compare the Hexadecimal Value at the specified Address with the entered Value.
  Now the Address will automatically incremented and comparison will be done again until the Element Repeat Count is reached.
  When a mismatch is detected between the incremented address and the value the sequence will be interrupted.

- **Disabled Element**

  Execution suppressed.

- **Special Element**

  Marks a special Element like,
  - SW Delay Element
  - Indigo VSync Element

- **Disabled Special Element**

  Execution suppressed.

- **Element with Breakpoint**

  Marks an element of different types where a breakpoint is set.

- **Disabled Element with Breakpoint**

  Execution suppressed.

Component Name Section

Contain the name of the corresponding *Component / IP*.

Address Section

Represent the total address of the register.

Size Section

Describe the size of the register. (8, 16 or 32 Bit)

Register Value Section

Value of the register. (Hexadecimal)

Error Section

When an error occurs on the last hardware access then this section contains a status letter,

• **X**	Error
• **W**	Warning
• **I**	Information

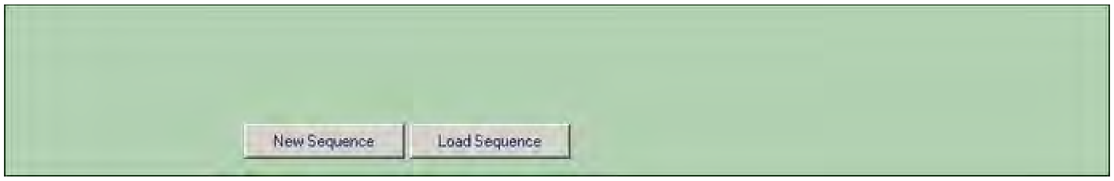
Keyboard Controls

r	Remove item
u	Move item up
d	Move item down
e	Toggle enable/disable item
t	Move item to Top
b	Move item to Bottom
c	Copy item
Enter	Select write edit field
Tab	Select next user control
Up / Down Arrow	Step to previous / next item

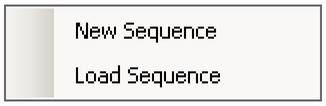
3. Action View

The *Action View* contains specific available items corresponding to the selected / active item in the *Selection View* or in the *Item View*.

Typically it contains some information about the item as well as some controls which offer special actions.



A Context menu is also available by pressing the right mouse button on the Register Sequencer in the Selection View.



New Sequence Button

Create a new and empty sequence as sub-item of the *Register Sequencer* on the *Selection View*.

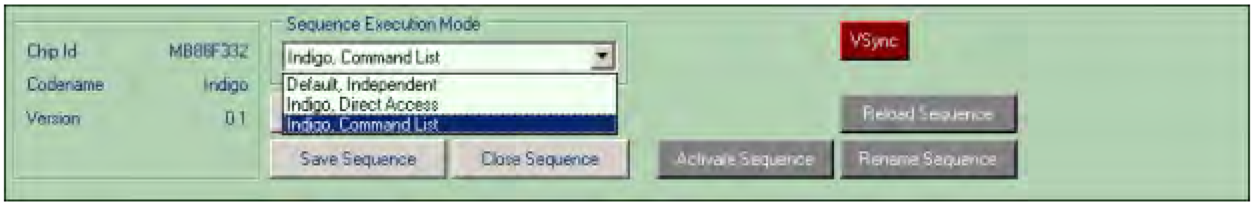
Load Sequence Button

Allows loading a register sequence that was prepared and stored before and add this sequence as sub-item of the *Register Sequencer*.

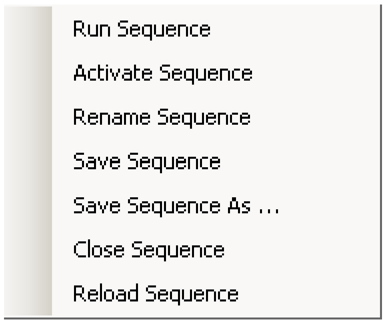
The typical file extension of a sequence is .gdcseq.

a. Sequence Page

When a *Sequence Node* (sub-item of the *Register Sequencer*) is selected on the *Selection View* then relevant sequence information as well as some action controls will appear on the *Action View*.



A Context menu is also available by pressing the right mouse button on a Register Sequencer Node item in the Selection View.



Save As ... Button

Press this button if it is required to store the selected register sequence.

The typical file extension of a sequence is .gdcseq.

A secured/protected sequence file which can only be read by Fujitsu Developer Suite, Fujitsu GDC Studio and Fujitsu GDC Player instances with the same Authorization / Encryption Code can be saved as .gdcseqsec.

If the "Indigo, Command List" mode is selected in the "Sequence Execution Mode" *Combo Box* then it is possible to store the sequence as .gdcicmd.

A human readable and easy to edit text file format is also available to store data, .par.
For more Information please refer to the *Par File Format* information page in the ***Customer Information*** section.

Save Sequence Button

Press this button if it is required to store the register sequence which was formerly loaded.

This sequence will be stored with the same extension as it was loaded.

Close Sequence Button

Closes the current selected register sequence and removes it from the *Selection View*.

Activate Sequence Button

Activates the current register sequence and marks it having the input focus.
Remarks : Register(s) will only be added to the current activated register sequence.

Rename Sequence Button

Allow the user to change the name of a register sequence.
This is essential for identifying the aim of such a sequence.

Reload Sequence Button

When a register sequence is already loaded but modified outside in an editor it can be reloaded by pressing this button.

Sequence Execution Mode Combo Box

There are different modes available to execute a sequence.

- **Default, Independent**
  This execution mode is not device specific.
  It executes the register sequence as they are listed in the *Item View* without any device specific options.
  Special device specific elements will be ignored in this execution mode.

- **Indigo, Direct Access**
  This is a specific operation mode for the Indigo device.

- **Indigo, Command List**
  This is a specific operation mode for the Indigo device.
  When executing the sequence in this mode a *Command List* will be generated and written into the *Command List Buffer*.
  After this the Indigo Command Sequencer will be triggered to force executing the command list at once.
  It allows adding a special *Command Sequencer Elements*.

Indigo2, Command List
This is a specific operation mode for the Indigo2 device.
When executing the sequence in this mode a *Command List* will be generated and written into the *Command List Buffer*.
After this the Indigo2 Command Sequencer will be triggered to force executing the command list at once.
It allows adding special *Command Sequencer Elements.*

Sw Delay Button

This special button is only available in the *Default, Independent Mode*.

By pressing this button a *Software Delay Element* is added which delays the execution of a sequence on the inserted position for a specified period of time.

Indigo Specific Elements

These special elements are only available for the Indigo device and the *Indigo, Command List Execution Mode*.

VSync Element Button

It forces the chip to wait until a *Vertical Sync* event occurs before continuing the *Command List Operation*.

Indigo2 Specific Elements

These special elements are only available for the Indigo2 device and the *Indigo2, Command List Execution Mode*.

Special Element ComboBox

WAIT Wait Element

This instruction performs a delay.
The number of microseconds can be specified by the Count operand.
Due to implementation issues, the overall delay can be larger (up to 3 microseconds) than the specified Count value but will never be shorter.

SWINT Software Interrupt Element

This instruction generates a pulse on swint_o output signal which should be connected to interrupt controller.

LABEL Label Element

Store current program counter address to EREG register.
This can be used for implementation of backward loops.

LOOP Loop Element

Continue execution at address stored in EREG register.
This can be used for implementation of backward loops.

JUMP Jump Element

Continue execution at provided Address.
This instruction is like a jump and won't return.

JUMP RELATIVE Jump Relative Element

Continue execution at provided distance.

WATCHDOG RESET Watchdog Reset

This instruction resets the watchdog timer.
It must be executed within a limited time given by the watchdog load register and the divider value.

WATCHDOG SET Watchdog Set

This instruction does the setup of the watchdog timer.
If Divider and Counter parameters are all '0', watchdog timer will be disabled, otherwise timer will be started with the specified values.
Doing a new WDS instruction while timer is running also starts the timer with the new values immediately

WRITE	Write Element

Write list of data to destination buffer.

DRGET	Data Register Get Element

Get data from address and store it in local DREG register.
8 bit, 16 bit and 32 bit transfers can be performed.
Address has to be aligned to the transfer size.

DRPUT	Data Register Put Element

Store data from local DREG register to Address.
8 bit, 16 bit and 32 bit transfers can be performed.
Address has to be aligned to the transfer size.

DRAND	Data Register And Element

Logical and of DREG content with value.

DROR	Data Register Or Element

Logical or of DREG content with value.

DRINVERT	Data Register Invert Element

Bitwise logical not of DREG content.

DRSHIFT LEFT	Data Register Shift Left Element

Logical shift left of DREG content.

DRSHIFT RIGHT	Data Register Shift Right Element

Logical shift right of DREG content.

DRADD	Data Register Add Element

Add value to DREG content.

DRCHECK	Data Register Check Element

Compare bits of DREG with provided Value and skip next instruction when result is equal.

ARGET	Address Register Get Element

Get data from address and store it in local AREG register.

ARGET INDIRECT	Address Register Get Indirect Element

Get data from address in AREG register and store it in local DREG register.
8 bit, 16 bit and 32 bit transfers can be performed.
AREG value has to be aligned to the transfer size.

ARPUT INDIRECT	Address Register Put Indirect Element

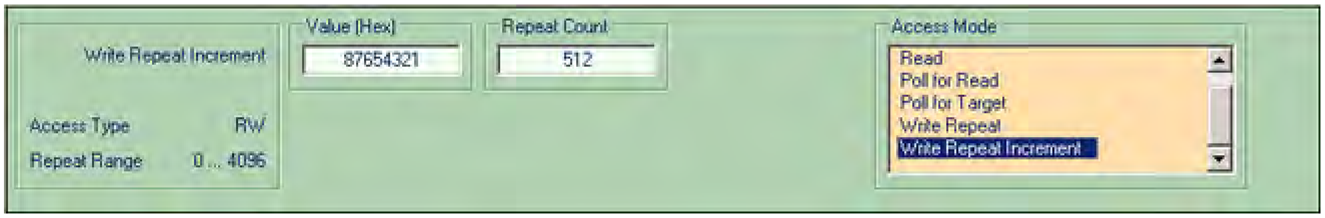
Store data from local DREG register to address in AREG register.
8 bit, 16 bit and 32 bit transfers can be performed.
AREG value has to be aligned to the transfer size.

END End Element

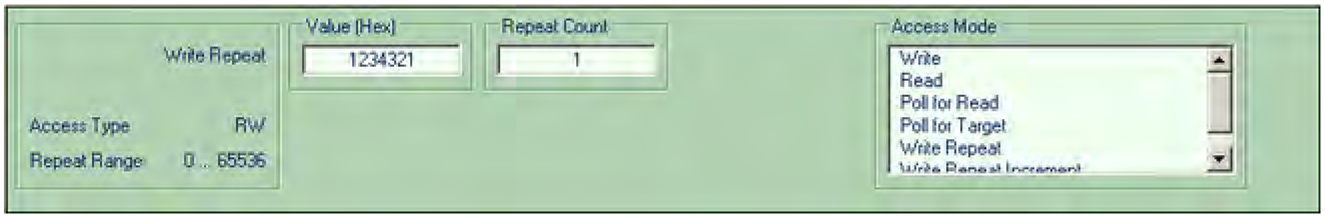
Stop execution of the current command.

b. Sequence Item Page

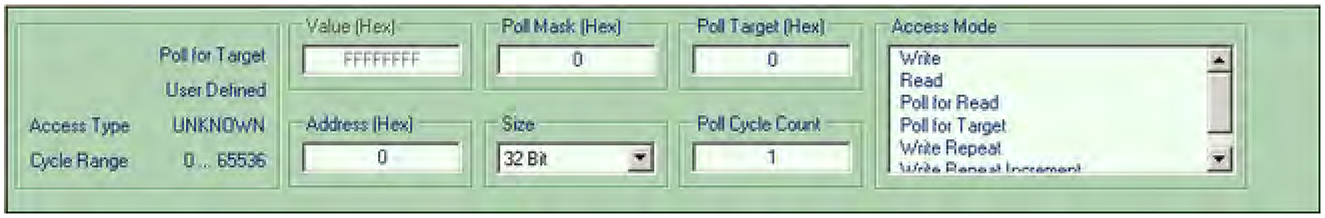
When a *Sequence Item* is selected on the *Item View* then relevant information as well as some possible action controls will appear on the *Action View*.



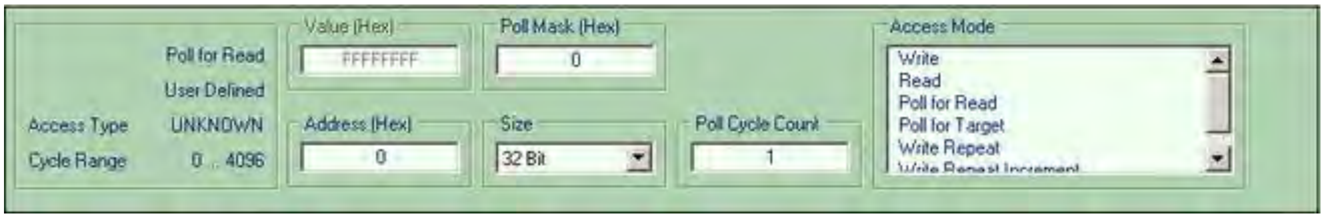
Write Repeat Increment
Access Type RW
Repeat Range 0 ... 4096
Value (Hex): 87654321
Repeat Count: 512
Access Mode: Read, Poll for Read, Poll for Target, Write Repeat, Write Repeat Increment



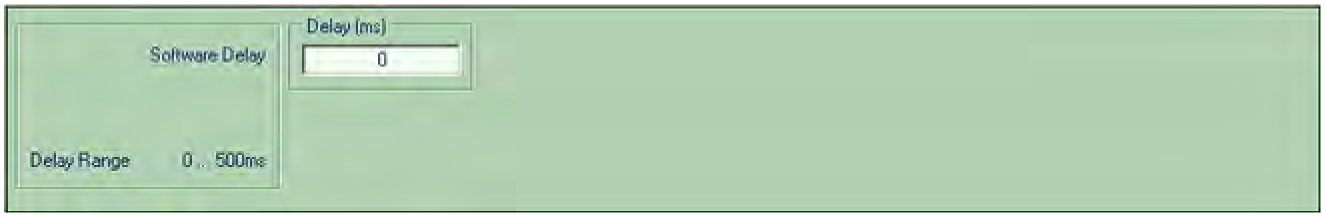
Write Repeat
Access Type RW
Repeat Range 0 .. 65536
Value (Hex): 1234321
Repeat Count: 1
Access Mode: Write, Read, Poll for Read, Poll for Target, Write Repeat, Write Repeat Increment



Poll for Target
User Defined
Access Type UNKNOWN
Cycle Range 0 .. 65536
Value (Hex): FFFFFFFF
Poll Mask (Hex): 0
Poll Target (Hex): 0
Address (Hex): 0
Size: 32 Bit
Poll Cycle Count: 1
Access Mode: Write, Read, Poll for Read, Poll for Target, Write Repeat, Write Repeat Increment



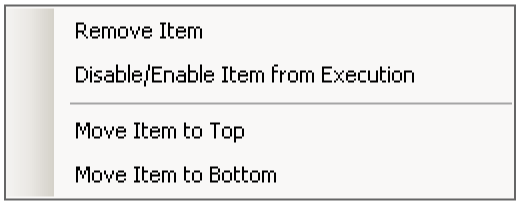
Poll for Read
User Defined
Access Type UNKNOWN
Cycle Range 0 .. 4096
Value (Hex): FFFFFFFF
Poll Mask (Hex): 0
Address (Hex): 0
Size: 32 Bit
Poll Cycle Count: 1
Access Mode: Write, Read, Poll for Read, Poll for Target, Write Repeat, Write Repeat Increment



Software Delay
Delay Range 0 .. 500ms
Delay (ms): 0

...

A Context menu is also available by pressing the right mouse button on a Register Sequencer Item in the Action View.



Value Edit Field

Depending on the selected item an *Edit Field* will appear which is prepared to enter hexadecimal values.

Typically this field will appear when a register value should be entered.

Special Function Elements,

Jump Element

> The entered value in the jump element will be used for checking and comparing the element with the value of the action that happened immediately before.

Indigo2

DREG AND Element

> Data Register & Value -> Data Register

DREG OR Element

> Data Register | Value -> Data Register

DREG ADD Element

> Data Register + Value -> Data Register

DREG CHECK Element

> Compare bits of Data Register with Value and skip next instruction when result is equal.
> if(DREG == Value) PC + sizeof(next instruction) -> PC

Write Element

> Value that should be written onto the given destination address.

To manipulate a register sequence item you can either press the *Return Key* or you can directly select the hexadecimal *Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value, the *Enter Key* must be pressed to accept the new value.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

Poll Cycle Count / Delay Edit Field / Repeat Count

Depending on the selected item an *Edit Field* appear which is prepared to enter decimal values.

Currently this field appears when the selected item is a,

SW Delay Element
The field represents the software delay in milliseconds.
Delay(ms).

Poll for Read Element
Here the the maximum allowed *Poll Cycles* to reach User Condition should be entered.
If User Condition is not reached within *Poll Cycles* processing of the *Poll for Read Element* will be stopped.
Poll Cycles.

Poll for Target Element
Here the the maximum allowed *Poll Cycles* to reach User Condition should be entered.
If User Condition is not reached within *Poll Cycles* processing of the *Poll for Read Element* will be stopped.
Poll Cycles.

Write Repeat Element
Here the field represents the number of *Write Repeats*.
Repeat Count.

Write Repeat Increment Element
Here the field represents the number of *Write + Address Increment Repeats*.
Repeat Count.

Jump Element
Here the maximum number of loops should be entered before escaping.
This is helpful for a check condition and to ensure that the loop will be left.

Special Function Elements,

Indigo2

Wait Element
Delay in microseconds.

Watchdog Set Element
Represents the divider value of the watchdog timer.

DREG SHIFT LEFT
Bit count to shift content of Data Register left.

DREG SHIFT RIGHT
Bit count to shift content of Data Register right.

To manipulate a register sequence item you can either press the *Return Key* or you can directly select the decimal *Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value, the *Enter Key* must be pressed to accept the new value.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

Access Mode List View

Each *Sequence Item* can be one of the following Access Modes,

Read

If selected the item will be a *Read Element*.
That means that from the given *Address* a value with the displayed *Size* will be read on execution.

Write

This is the default Access Mode when an item was newly added to the sequence.
On execution the *Register Value* will be written with the displayed *Size* to the given *Address*.
Write Element.

Write Field

This is a special write Access Mode allowing to only save a field of the register.
But be careful for this the accessed register must readable and writeable.
Furthermore the following rules must be followed :

0 <= fieldOffset < register width
1 <= fieldWidth < (register width - fieldOffset)
0 <= fieldValue < 2 ^ fieldWidth

On execution the *Register Value* will first be read out, modified with the field information an written back.
Write Field Element.

Read Field

This is a special write Access Mode allowing to read a field of the register.
The following rules must be followed :

0 <= fieldOffset < register width
1 <= fieldWidth < (register width - fieldOffset)

On execution the *Register Value* will be read out, masked, shifted and the result is part of the Value Edit Field.
Read Field Element.

Poll for Read

Poll for Read Element.
In this mode the Value with the specific Size will be read out from the defined *Address* until,

- User Condition is getting TRUE :
  Value & Mask == Mask
  This is useful to check Status Register Bits/Flags,
  e.g. Value=0x00AB, Mask=0x0800 -> FALSE -> read again ...
  e.g. Value=0x0FAB, Mask=0x0800 -> TRUE -> continue
- the number of *Poll Cycles* are reached
- an error occurs

Poll for Target

Poll for Target Element.
In this mode the Value with the specific Size will be read out from the defined *Address* until,

- User Condition is getting TRUE :
  Value & Mask == Target
  This is useful to check Bits/Flags with highest flexibility.
- the number of *Poll Cycles* are reached
- an error occurs

Write Repeat

When this item is executed the *Value* will be written with the displayed *Size* to the given

Address. This will be done multiple times until *Repeat Count* is reached.
Write Repeat Element.

Write Repeat Increment

On execution the *Value* will be written with the displayed *Size* to the given *Address*. Then the *Address* will be *automatically incremented* by the displayed *Size* and the same *Value* will be written again to the (now incremented) *Address*. This will be done multiple times until Repeat Count is reached.
Very useful to fill arrays with the same value.
Write Repeat Increment Element.

Label

A Label element can be set but does not have a direct special function.
It can only be the destination for jump loops.
For more information refer to the Jump.

Jump

The jump command evaluates the resulting Value of the action element immediately before the jump element itself.
When the value given on the jump was true then the jump will be executed.

jmpxx value maxretry

Where "value" is the value which will be compared to the previous value result.
"maxretry" is the maximum number of loops that are allowed before stopping the loop and was meant as escape possibility.

The following jump types are allowed an can be selected in a combo box,

Jump if Equal
The jump will be performed when the last operation value is equal to the jump value.

Jump if Not Equal
The jump will be performed when the last operation value is not equal to the jump value.

Jump if Greater
The jump will be performed when the last operation value is greater than the jump value.

Jump if Greater or Equal
The jump will be performed when the last operation value is greater or equal to the jump value.

Jump if Less
The jump will be performed when the last operation value is less than the jump value.

Jump if Less or Equal
The jump will be performed when the last operation value is less or equal to the jump value.

Jump on Success
The jump will be performed when the last operation was successful.

Jump if Warning Or Error
The jump will be performed when the last operation returned either with a warning or failed..

Example :

Jumping is useful for check routines where a specific value of a register is checked etc.

Label Element

Delay Element (e.g. 10ms)
Read Element (e.g. 8Bit on address 0xAABBCCDD, resulting value = 0x20)

Jump if Not Equal (e.g.jump value = 0x30, max loop = 20)

What happens :

1.the delay will be executed before start reading 8 bit from the specified address
2. on the jump element it will be compared if the last value that was read from address 0xAABBCCDD matches the corresponding condition with the jump value of 0x30.
Here a jump will be performed when these values differ.
3. As long as the read value stays 0x20 and the max loop counter was not reached the loop will be executed.
4. When now either the last read value is getting 0x30 or the loop counter was reached the loop will be left and the sequence will be continued.

If one of these values are required select them with the left mouse button and acknowledge the change with the right mouse button.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

Poll Mask Edit Field

Depending on the selected item an *Edit Field* appear which is prepared to enter hexadecimal values.

With this mask it is possible check the register value that was read from hardware for special bits.

Currently this field appears when the selected item is a,

Poll for Read Element

If the mask bits are detected the *Poll for Read Item* will be finished with TRUE user condition.
When the bits are not detected the register value will be read continuously until,

- user condition is getting TRUE - means mask bits detected
  Value & Poll Mask == Poll Mask
  e.g. Value=0x00AB, Mask=0x0800 -> FALSE -> read again ...
  e.g. Value=0x0FAB, Mask=0x0800 -> TRUE -> continue
- or *Poll Cycle Count* is reached - which leads to FALSE user condition
- or an hardware access error occurs - which also leads to FALSE user condition

Poll for Target Element

If the target is calculated and identical then the *Poll for Target Item* will be finished with TRUE user condition.
When the target is not detected the register value will be read continuously until,

- user condition is getting TRUE - means target detected
  Value & Poll Mask == Target
- or *Poll Cycle Count* is reached - which leads to FALSE user condition
- or an hardware access error occurs - which also leads to FALSE user condition

Poll Target Edit Field

This field is only available when the Access Mode was set to *Poll for Target Element*.

If the target is calculated and identical then the *Poll for Target Item* will be finished with TRUE user condition. When the target is not detected the register value will be read continuously until,

- user condition is getting TRUE - means target detected
  Value & Poll Mask == Target
- or *Poll Cycle Count* is reached - which leads to FALSE user condition
- or an hardware access error occurs - which also leads to FALSE user condition

Address Edit Field

This field is only available when a *User Defined Item / Element* was added by the user.

It this *Address Edit Field* the Register Address of the register sequence item can be changed.

Special Function Elements,

Indigo2

Jump Element
: Represents the address to jump from the current command sequence position.

Write Element
: Destination address of the command sequencer write operation.

DREG Get Element
: Address to load data into the Data Register.

DREG Put Element
: Address to store content of the Data Register.

AREG Get Element
: Address to load data into the Address Register.

Jump Relative Element
: Offset to jump from current position.
Each value will be interpreted as multiple of 4 byte.
PC + Offset * 4 -> PC

To manipulate a register sequence item you can either press the *Return Key* or you can directly select the hexadecimal *Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value, the *Enter Key* must be pressed to accept the new value.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

Jump Condition/Mode Combo Box

This box is only available when on the *Jump Element*.

It allows manipulating the jump condition of the loop to the last Label Element.

The following jump conditions are allowed an can be selected in the combo box,

Jump if Equal

The jump will be performed when the last operation value is equal to the jump value.

Jump if Not Equal

The jump will be performed when the last operation value is not equal to the jump value.

Jump if Greater

The jump will be performed when the last operation value is greater than the jump value.

Jump if Greater or Equal

The jump will be performed when the last operation value is greater or equal to the jump value.

Jump if Less

The jump will be performed when the last operation value is less than the jump value.

Jump if Less or Equal

The jump will be performed when the last operation value is less or equal to the jump value.

Jump on Success

The jump will be performed when the last operation was successful.

Jump if Warning Or Error

The jump will be performed when the last operation returned either with a warning or failed..

Register Size Combo Box

This box is only available when a *User Defined Item / Element* was added by the user.

It allows manipulating the Size of the Register Access.

The following register access sizes are available,

- 8 Bit
- 16 Bit
- 32 Bit

Field Offset

This field is only available when a *User Defined Item / Element* was added by the user.

Currently this field appears when the selected item is a,

> *Write Field Element*
>
> > This field represents the offset within the register where the register field starts.
> > Please ensure that the combination "Field Offset", "Field Width" and "Value" are correct and does not exceed maximum possible.

Special Function Elements,

> **Indigo2**
>
> > Watchdog Set Element
> >
> > > Represents the counter value of the watchdog timer.

To manipulate a register sequence item you can either press the *Return Key* or you can directly select the *Edit Field* by mouse.

When the *Edit Field* is selected and ready to be entered, the background color will be displayed in a light red color.

After entering the value, the *Enter Key* must be pressed to accept the new value.
If the new value is not acknowledged - either by leaving the control input or pressing the *Escape Key* - then the old value will appear again.

In both cases the input mode ends and the background color of the *Edit Field* is displayed in a light green color.

V. Sequence Stacker

• Purpose

The *Sequence Stacker* is required to create more complex sequence scenarios combining different register sequences to a complete action.
These stacks are helpful to store complete hardware configurations.

• Drag & Drop

It is possible to drag & drop *Sequence Stack Files* into the application.
This can be done by selecting a sequence stack file in the window explorer and dragging the file onto the *Item View*.
This can only be done when the *Sequence Stacker Item* is selected in the *Selection View*.
(Multiple File Selection)

1. Selection View

The *Selection View* contains the main tools that are available with the current *Project / Solution* in combination with the available hardware design.

Sequence stacks can be stored and loaded separately.

The *Sequence Stacker* can contain none or more sequence stacks that can describe a more complex scenario / behavior - combining multiple register sequences to a stack.

Typically the *Selection View* contains the following sub-items for the *Sequence Stacker* :

- **Sequence Stack**

  The *Sequence Stacker* can contain none or more sequence stacks.
  This symbol specifies an available sequence stack.

- **Active Sequence Stack**

  When a sub-item of the *Sequence Stacker* has such a symbol it is marked as the *Active Sequence Stack*.

2. Item View

The *Item View* contains specific available items corresponding to the selected / highlighted item in the *Selection View*.

The following sections are displayed for the *Register Debugger* in the *Item View*.

Name Section

Identify the filename of the loaded register sequence.

- **Standard Register Sequence Element**

  ●

  Represent a complete register sequence file which was loaded into the current sequence stack
  Typically the name of the register sequence file was displayed in this section.
  Furthermore when hovering over the filename information to the corresponding sequence file will be displayed.
  Remark :

  The single sequence items of the register sequence can also be displayed by expanding the tree.
  It is not possible to manipulate any sequence item.

- **Disabled Element**

  Execution of the disabled register sequence will be suppressed.

- **Element with Breakpoint**

  ●

  If a breakpoint was set to a loaded register sequence it will be marked with this icon.

- **Disabled Element with Breakpoint**

  ●

  Disabled item where a breakpoint was set.
  Execution suppressed.

Component Name Section

Contain the name of the corresponding *Component / IP*.
Only for information purposes to a specific sequence item - nothing can be changed.

Address Section

Represent the total address of the register.
Only for information purposes to a specific sequence item - nothing can be changed.

Size Section

Describe the size of the register. (8, 16 or 32 Bit)
Only for information purposes to a specific sequence item - nothing can be changed.

Register Value Section

Value of the register. (Hexadecimal)
Only for information purposes to a specific sequence item - nothing can be changed.

Error Section

When an error occurs on the last hardware access then this section contains a status letter,

• **X**	Error
• **W**	Warning
• **I**	Information

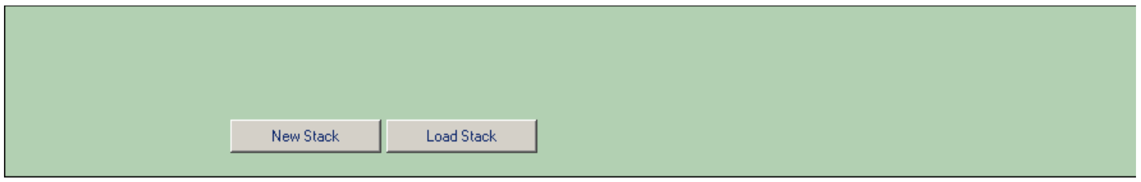
Keyboard Controls

r	Remove item
u	Move item up
d	Move item down
e	Toggle enable/disable item
t	Move item to Top
b	Move item to Bottom
Up / Down Arrow	Step to previous / next item
Right / Left Arrow	Expand / Collapse item

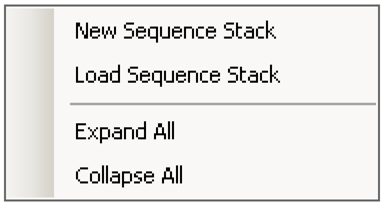
3. Action View

The *Action View* contains specific available items corresponding to the selected / active item in the *Selection View* or in the *Item View*.

Typically it contains some information about the item as well as some controls which offer special actions.



A Context menu is also available by pressing the right mouse button on the Register Sequencer in the Selection View.



New Stack Button

Create a new and empty sequence stack as sub-item of the *Sequence Stacker* on the *Selection View*.

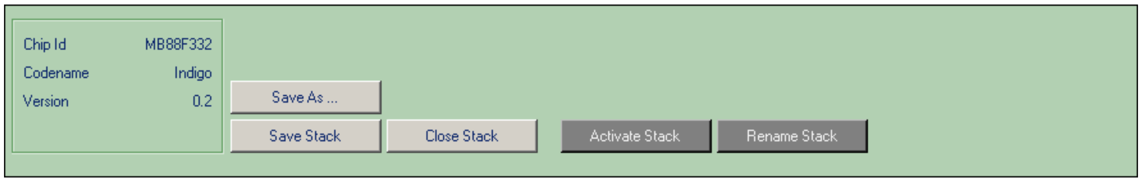
Load Stack Button

Allows loading a sequence stack that was prepared and stored before and add it as sub-item of the *Sequence Stacker*.

The typical file extension of a sequence stack is .gdcseqstack.

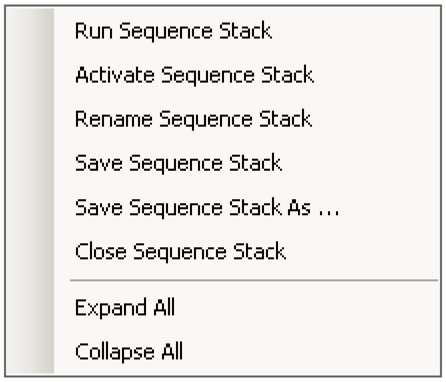
a. Stack Page

When a *Stack Node* (sub-item of the *Sequence Stacker*) is selected on the *Selection View* then relevant stack information as well as some action controls will appear on the *Action View*.



Chip Id	MB88F332
Codename	Indigo
Version	0.2

A Context menu is also available by pressing the right mouse button on a Sequence Stacker Node item in the Selection View.



Save As ... Button

Press this button if it is required to store the selected sequence stack.

The typical file extension of a sequence is .gdcseqstack.

Save Stack Button

Press this button if it is required to store the sequence stack which was formerly loaded.

This stack will be stored with the same extension as it was loaded.

Close Stack Button

Closes the current selected sequence stack and removes it from the *Selection View*.

Activate Stack Button

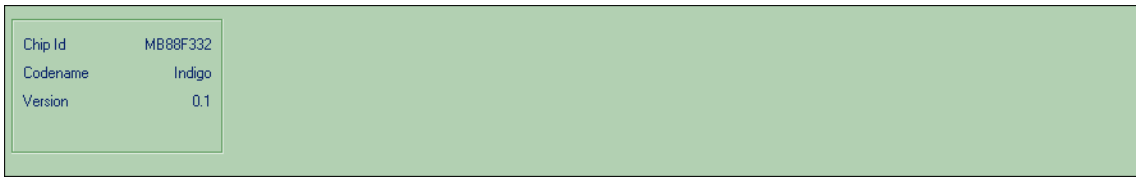
Activates the current sequence stack and marks it having the input focus.

Rename Stack Button

Allow the user to change the name of the sequence stack.
This is essential for identifying the intention of such a stack.

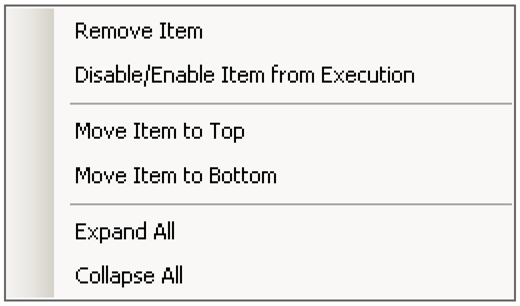
b. Stack Item Page

When a *Stack Item* is selected on the *Item View* then only information will be displayed showing the version number on which the register sequence was based on creation time.



Chip Id	MB88F332
Codename	Indigo
Version	0.1

A Context menu is also available by pressing the right mouse button on a register sequence in the Action View.



VI. Image Manager

- **Purpose**

One aim of the *Image Manager* is to support application development by converting image files into different pixel formats.
But the main purpose is the conversion of the image pixel data into a chip specific format and easily copying the converted pixel data as well as the Color Palette to the source code of the Target Application.

- **Drag & Drop**

It is possible to drag & drop an *Image File* into the application.
This can be done by selecting an imaging in the window explorer and dragging the file onto the *Item View*.
This can only be done when the *Image Manager Item* is selected in the *Selection View*.
(Single File Selection)

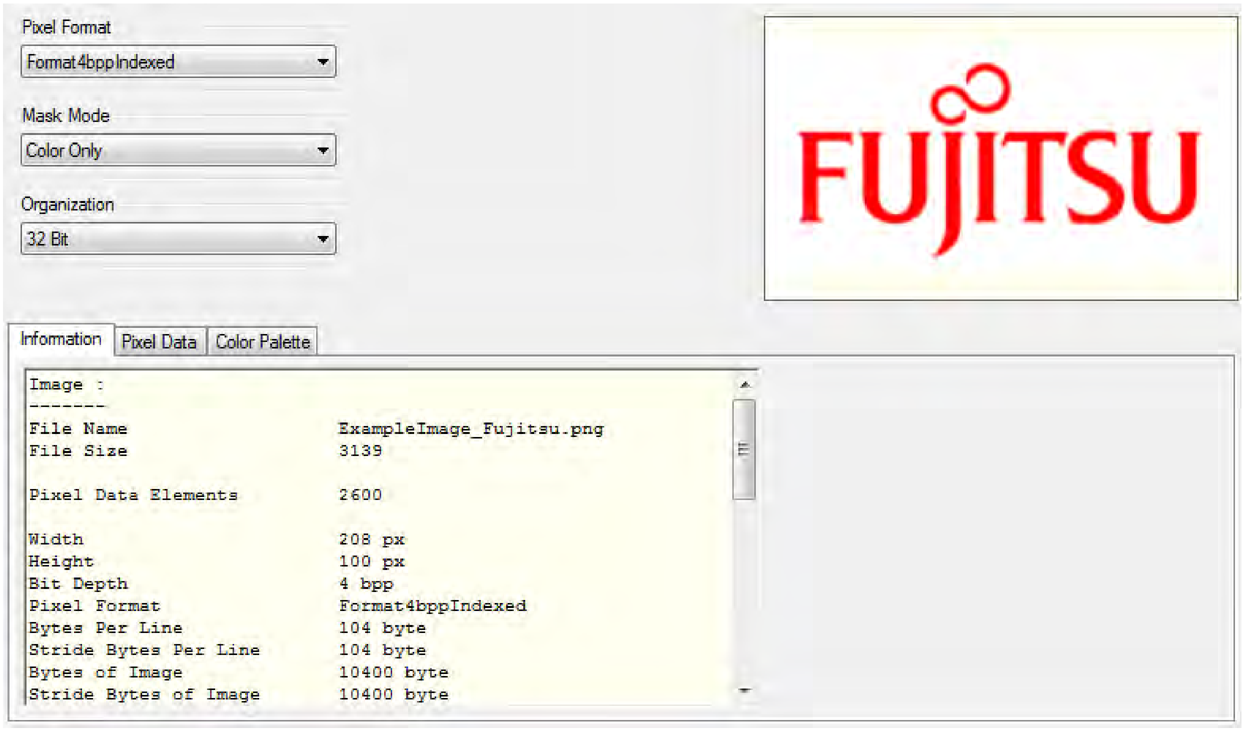
1. Item View

The *Item View* contains different user input controls which allows manipulation of the original image as well as text boxes that are prepared to display and extract image information.

a. Information Page

The first page that is visible is the *Information Page*.
It contains all relevant information of the image as well as of the color palette, when available.
The most essential ones to build a constant data array in the application code are the *Width*, *Height* and *Bit Depth* which represent the image dimensions and the number of *Pixel Data Elements* that a data array in the specified organization (see corresponding combo box) would require.

e.g.



```
Image :
-------
File Name                    ExampleImage_Fujitsu.png
File Size                    3139

Pixel Data Elements          2600

Width                        208 px
Height                       100 px
Bit Depth                    4 bpp
Pixel Format                 Format4bppIndexed
Bytes Per Line               104 byte
Stride Bytes Per Line        104 byte
Bytes of Image               10400 byte
Stride Bytes of Image        10400 byte
```

b. Pixel Data Page

This page contains the core pixel data of the image.
By changing the data organization style (see corresponding combo box) it can be selected how the data is getting organized.
It could either be 8, 16, 24 or 32 bit width data elements.

For some chip versions one or more specific styles are offered which re-organizes the pixel data / elements in a way which best match the hardware requirements.

Indigo2 / MB88F334

• "Indigo2 32 Bit - ARGB"	Arranging the pixel data as they are required.
• "Indigo2 32 Bit - ABGR"	Arranging the pixel data as they are required.
• "Indigo2 32 Bit - RGBA"	Arranging the pixel data as they are required.
• "Indigo2 32 Bit RLD - ARGB"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.
• "Indigo2 32 Bit RLD 2 - ABGR" other RLD	Other RLD compression algorithm.

Indigo / MB88F332, IndigoL / MB88F333

• "Indigo 32 Bit"	Arranging the pixel data as they are required.
• "Indigo 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.

Ruby / MB86298

• "Ruby 32 Bit - ARGB"	Only available for 32 Bit formats. Arranging the pixel data as ARGB.
• "Ruby 32 Bit - ABGR"	Only available for 32 Bit formats. Arranging the pixel data as ABGR.
• "Ruby 32 Bit - RGBA"	Only available for 32 Bit formats. Arranging the pixel data as RGBA.

JadeD / MB86R02

• "JadeD 32 Bit"	Arranging the pixel data as they are required.
• "JadeD 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.

EmeraldL / MB86R11
EmeraldP / MB86R12

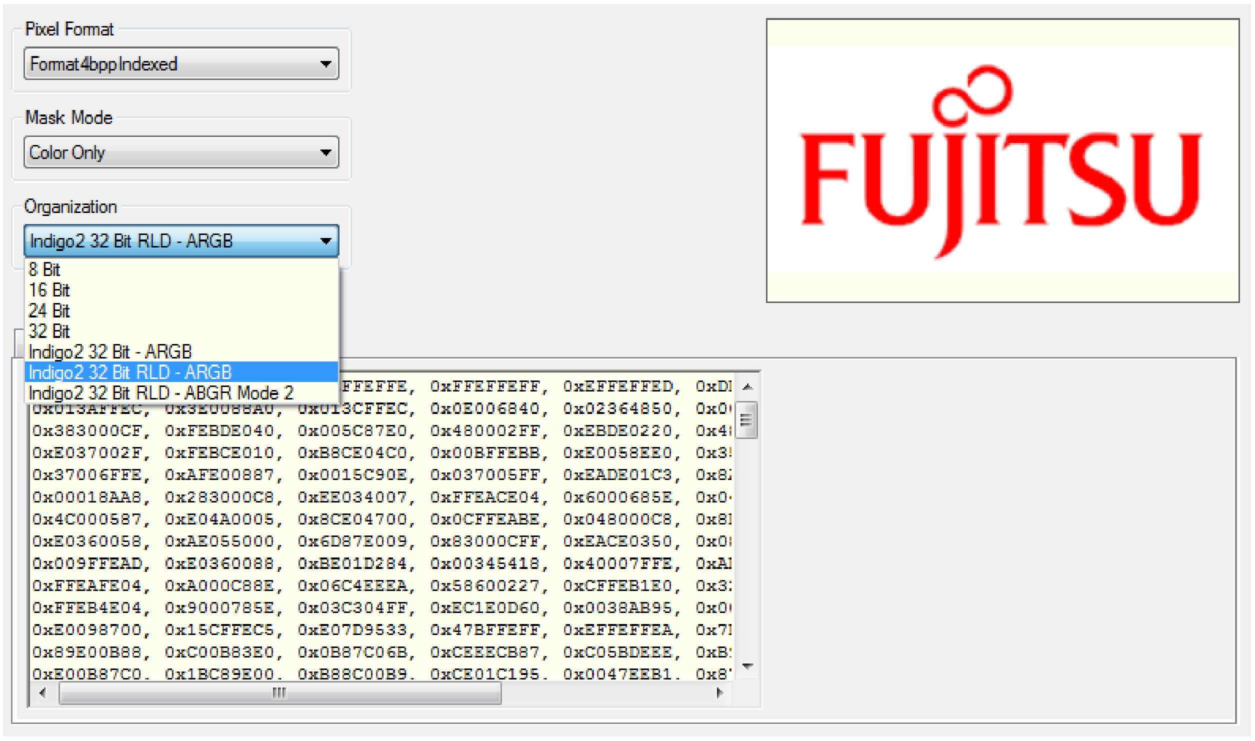
• "Emerald 32 Bit - ARGB"	Only available for 32 Bit formats. Arranging the pixel data as ARGB.
• "Emerald 32 Bit - ABGR"	Only available for 32 Bit formats. Arranging the pixel data as ABGR.
• "Emerald 32 Bit - RGBA"	Only available for 32 Bit formats. Arranging the pixel data as RGBA.
• "Emerald 32 Bit - ARGB666UC24To18"	Special Format converting 24RGB images into 18Bit images. Typically required for special Panels.

• "Emerald 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.

Triton / MB8AC0440

- "Triton 32 Bit - ARGB" Arranging the pixel data as they are required.
- "Triton 32 Bit - ABGR" Arranging the pixel data as they are required.
- "Triton 32 Bit - RGBA" Arranging the pixel data as they are required.
- "Triton 32 Bit RLD - ARGB"
  Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.
- "Triton 32 Bit RLD 2 - ABGR" other RLD
  Other RLD compression algorithm.

e.g.


Pixel Format: Format4bppIndexed

Mask Mode: Color Only

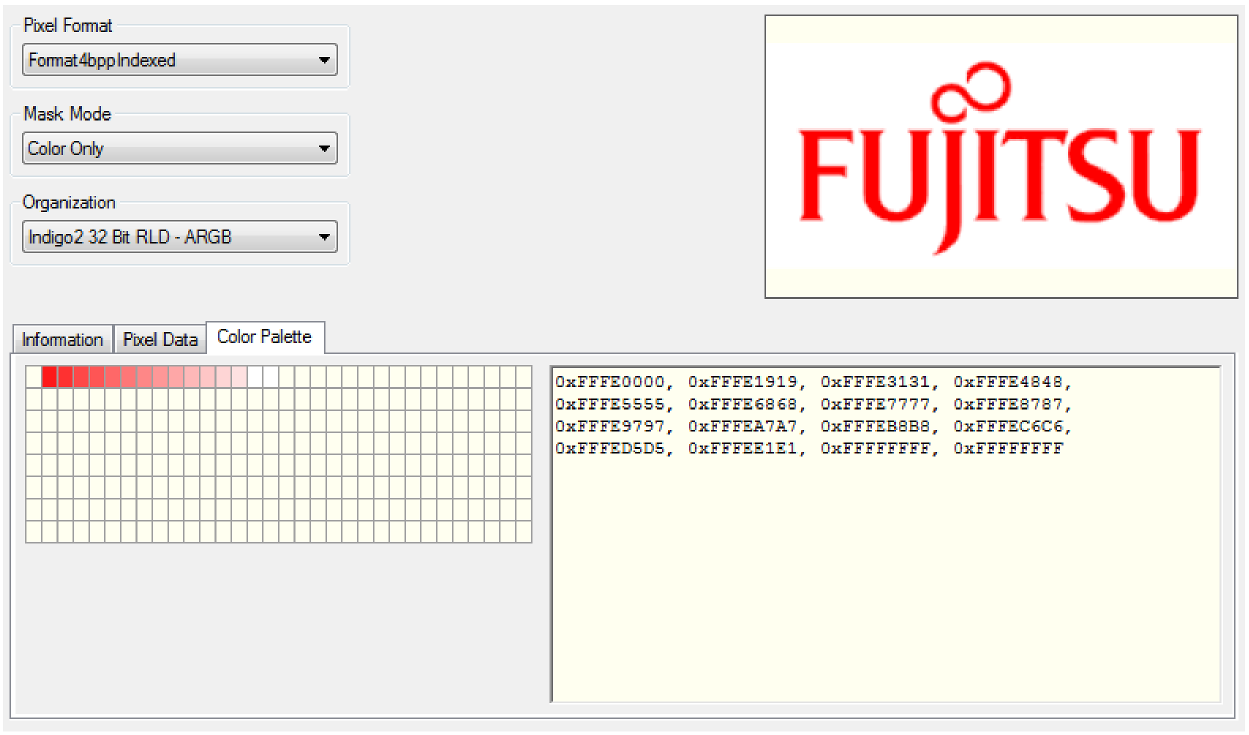
Organization: Indigo2 32 Bit RLD - ARGB

```
                        FFEFFE,  0xFFEFFEFF,  0xEFFEFFED,  0xD
0x013AFFEC,  0x3E0088A0,  0x013CFFEC,  0x0E006840,  0x02364850,  0x0
0x383000CF,  0xFEBDE040,  0x005C87E0,  0x480002FF,  0xEBDE0220,  0x4
0xE037002F,  0xFEBCE010,  0xB8CE04C0,  0x00BFFEBB,  0xE0058EE0,  0x3
0x37006FFE,  0xAFE00887,  0x0015C90E,  0x037005FF,  0xEADE01C3,  0x8
0x00018AA8,  0x283000C8,  0xEE034007,  0xFFEACE04,  0x6000685E,  0x0
0x4C000587,  0xE04A0005,  0x8CE04700,  0x0CFFEABE,  0x048000C8,  0x8
0xE0360058,  0xAE055000,  0x6D87E009,  0x83000CFF,  0xEACE0350,  0x0
0x009FFEAD,  0xE0360088,  0xBE01D284,  0x00345418,  0x40007FFE,  0xA
0xFFEAFE04,  0xA000C88E,  0x06C4EEEA,  0x58600227,  0xCFFEB1E0,  0x3
0xFFEB4E04,  0x9000785E,  0x03C304FF,  0xEC1E0D60,  0x0038AB95,  0x0
0xE0098700,  0x15CFFEC5,  0xE07D9533,  0x47BFFEFF,  0xEFFEFFEA,  0x7
0x89E00B88,  0xC00B83E0,  0x0B87C06B,  0xCEEECB87,  0xC05BDEEE,  0xB
0xE00B87C0,  0x1BC89E00,  0xB88C00B9,  0xCE01C195,  0x0047EEB1,  0x8
```

c. Color Palette Page

If a color palette is available on the image then the elements will be visualized on this page.
The grid which is located on the left side of the page represents the corresponding color of the 32 bit hexadecimal value displayed in the text box on the right side.
By copying this color information into the target application a *Color Lookup Table* could be build up.

e.g.



Pixel Format: Format4bppIndexed

Mask Mode: Color Only

Organization: Indigo2 32 Bit RLD - ARGB

```
0xFFFE0000, 0xFFFE1919, 0xFFFE3131, 0xFFFE4848,
0xFFFE5555, 0xFFFE6868, 0xFFFE7777, 0xFFFE8787,
0xFFFE9797, 0xFFFEA7A7, 0xFFFEB8B8, 0xFFFEC6C6,
0xFFFED5D5, 0xFFFEE1E1, 0xFFFFFFFF, 0xFFFFFFFF
```

Pixel Format Combo Box

When loading an image file, this combo box contains the pixel format of the current image as well as some target pixel formats to which this image can be transformed to.
The first entry in the box always represents the pixel format of the original image.

Organization Combo Box

This *Combo Box* allows the user to reorganize the pixel data into a style that is comfortable for the target application.
The data can be organized as a 8, 16, 24 or 32 bit width data array.

For some chip versions one or more specific styles are offered which re-organizes the pixel data / elements in a way which best match the hardware requirements.

Indigo2 / MB88F334

"Indigo2 32 Bit - ARGB"	Arranging the pixel data as they are required.
"Indigo2 32 Bit - ABGR"	Arranging the pixel data as they are required.
"Indigo2 32 Bit - RGBA"	Arranging the pixel data as they are required.
"Indigo2 32 Bit RLD - ARGB"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.
"Indigo2 32 Bit RLD 2 - ABGR" other RLD	Other RLD compression algorithm.

Indigo / MB88F332, IndigoL / MB88F333

"Indigo 32 Bit"	Arranging the pixel data as they are required.
"Indigo 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.

Ruby / MB86298

"Ruby 32 Bit - ARGB"	Only available for 32 Bit formats. Arranging the pixel data as ARGB.
"Ruby 32 Bit - ABGR"	Only available for 32 Bit formats. Arranging the pixel data as ABGR.
"Ruby 32 Bit - RGBA"	Only available for 32 Bit formats. Arranging the pixel data as RGBA.

JadeD / MB86R02

"JadeD 32 Bit"	Arranging the pixel data as they are required.
"JadeD 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.

EmeraldL / MB86R11

"Emerald 32 Bit - ARGB"	Only available for 32 Bit formats. Arranging the pixel data as ARGB.
"Emerald 32 Bit - ABGR"	Only available for 32 Bit formats. Arranging the pixel data as ABGR.
"Emerald 32 Bit - RGBA"	Only available for 32 Bit formats. Arranging the pixel data as RGBA.
"Emerald 32 Bit - ARGB666UC24To18"	Special Format converting 24RGB images into 18Bit images. Typically required for special Panels.
"Emerald 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.

EmeraldP / MB86R12

• "Emerald 32 Bit - ARGB"	Only available for 32 Bit formats. Arranging the pixel data as ARGB.
• "Emerald 32 Bit - ABGR"	Only available for 32 Bit formats. Arranging the pixel data as ABGR.
• "Emerald 32 Bit - RGBA"	Only available for 32 Bit formats. Arranging the pixel data as RGBA.
• "Emerald 32 Bit - ARGB666UC24To18"	Special Format converting 24RGB images into 18Bit images. Typically required for special Panels.
• "Emerald 32 Bit RLD"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.
• "Emerald 32 Bit RLD Mode 2"	Other RLD compression algorithm.

Triton / MB8AC0440

• "Triton 32 Bit - ARGB"	Arranging the pixel data as they are required.
• "Triton 32 Bit - ABGR"	Arranging the pixel data as they are required.
• "Triton 32 Bit - RGBA"	Arranging the pixel data as they are required.
• "Triton 32 Bit RLD - ARGB"	Encoding the image pixel data with an RLD algorithm and arranging the pixel data as they are required.
• "Triton 32 Bit RLD 2 - ABGR" other RLD	Other RLD compression algorithm.

Notes :

Some combinations of the *Pixel Format Combo Box* and the *Organization Combo Box* do not offer results.
The reason is that these special combinations for the loaded Image leads to an invalid format, pixel data alignment or similar which is not supported by the the corresponding hardware.

Mask Mode Combo Box

This combo box allows the user to choose between different masks that will be applied to the color values of the image.
Depending on the loaded picture as well as the selected pixel format one or more of the mask modes will become available.

- **Original** Mode
  This is only available for the original image. That means when the pixel format of the original image is selected in the *Pixel Format Combo Box* - which is always the first one in the list.
  When choosing this option only the original data without any masking or conversion will be displayed on the different pages.

- **Color Only** Mode
  This option only considers the color values R, G, B of an image and ignores the Alpha.
  The original image will first be converted into a 32bit ARGB image - this must be done to get an exact position-to-color related copy - especially for indexed formats.
  Afterwards the alpha channel will be removed before the manipulated image will be inserted into the converter where the destination pixel format will be applied.
  Finally the resulting image will be displayed.

- **Alpha Only** Mode
  This option only considers the alpha channel of the image and ignores the color values R, G, B.
  The original image will first be converted into a 32bit ARGB image - this must be done to get an exact position-to-color related copy - especially for indexed formats.
  Afterwards all color channels will set to the same value of the alpha channel to get a greyscaled copy of the image that represents the alpha channel.
  This copy will then be inserted into the converter where it is transformed into another greyscaled copy with the destination pixel format.
  At the end the converted output will then be manipulated to get back the alpha channel only while resetting the color values to 0 (black).
  The user can change the default color of 0 (black) by means of the *Alpha Mask Color Selection Box*.
  The colors on the grid of the *Color Palette Page* represent the alpha values as greyscaled palette, whereas the values in the text box are the correct ones.

- **Inverted Alpha Only** Mode
  Same as *Alpha Only Mode* but with inverted alpha map.

- **Alpha + Color** Mode
  It is similar to the *Original* mode with the difference that it is only available for the 32bit ARGB pixel format - independent if it is the same format than the original.
  Furthermore the displayed information and data was the result of the converter.

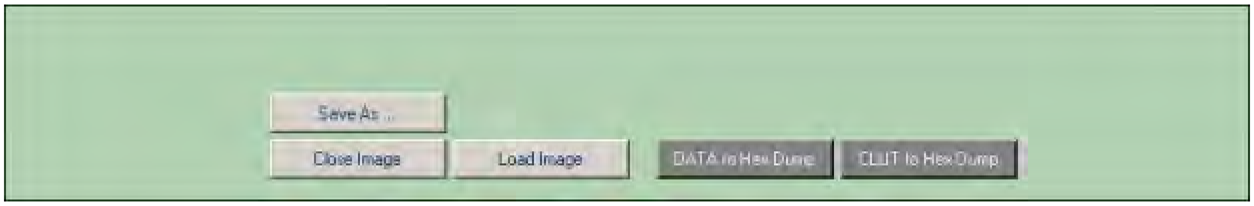
Alpha Mask Color Selection

This button only appears when in the *Mask Mode Combo Box* one of the *Alpha Mask Modes* is selected. It allows changing the basic color of the alpha image.

The default color is black.

2. Action View

When the *Image Manager* item is selected on the *Selection View* then relevant action controls will appear on the *Action View*.



Load Image Button

This button offers a dialog box which allows opening an image of the following file types,

• bmp	Bitmap
• png	Portable Network Graphics
• tiff	Tagged Image File Format
• jpeg	Joint Photographic Expert Group
• gif	Graphics Interchange Format

The supported pixel formats are,

- 1 bpp indexed
- 4 bpp indexed
- 8 bpp indexed
- 16 bpp
- 24 bpp
- 32 bpp

Close Image Button

By pressing this button the opened image will be closed.

Save As ... Button

1. Source Code Generator

Source Code of the currently loaded image can be automatically generated.

This is especially helpful when creating an application that should contain the *Pixel Data*, the *CLUT Data* (when available) and all required definitions to access and use them.

The output format depends on the *Pixel Format* and the *Organization* that are selected in the corresponding combo box.

By pressing this button a *Dialog Box* appear which allows to create or select a source code file of the following types,

• .c	(Standard C, Code File)
• .h	(Header File)

It does not matter if a **.c** file or a **.h** file is selected/entered, because always both files with the specified name will be generated.
If the specified filename already exists it will be overwritten - so please ensure that this is wanted, or enter a different name.

All generated definitions and arrays are in upper case.
Furthermore the following information of the original image is used to generate an unique identifier for the source code,

• NAMEOFIMAGE	Name of the Image
• WIDTHOFIMAGE	Width of the Image
• HEIGHTOFIMAGE	Height of Image
• BITDEPTHOFIMAGE	Bit Depth of Image

The syntax of the definitions and arrays are as follows,
NAMEOFIMAGE_WIDTHOFIMAGE_HEIGHTOFIMAGE_BITDEPTHOFIMAGE_XXX
whereas XXX can be,

• WIDTH	Image width definition
• HEIGHT	Image height definition
• BITDEPTH	Image bit depth definition
• DATA_SIZE	Size of elements in the data array definition
• DATA[]	Data array that contains pixel data
• CLUT_SIZE	Size of the CLUT array definition
• CLUT	CLUT array that contains color information

The data types will be interpreted depending on the selected organization,

• 8 Bit	uint8
• 16Bit	uint16
• 24 Bit	uint24 (should not be used)
• 32 Bit	uint32

The generated header file includes a "portable.h" file which is only a placeholder.
This file is not provided by Fujitsu Semiconductor because it is destination platform dependent and must contain the interpretation of the mentioned data types.

2. Binary File

The current loaded image can also be stored as binary file.

This is only possible when one of the 32 bit output organization items was selected in the Organization Combo Box.
When a Color Lookup Table is available then a second File will automatically generated with the suffix _CLUT.

The typical file extension for binary files is .gdc32dat.

File Information :

.gdc32dat

Is a standard binary file in which the values are stored as UInt32.
e.g. when two 32 bit values are displayed as,

0x12345678 0x0ABCDEF0

then the binary output file contains the following byte stream,
(reviewed by an Hex Viewer/Editor)

78 56 34 12 F0 DE BC 0A

When the image has an indexed pixel format, then two binary files will be stored.
To the selected or entered output filename the following endings will be added,

Pixel Data	-	*UserFilename*_PixelData.gdc32dat
Color Palette	-	*UserFilename*_ColorPalette.gdc32.dat

DATA to Hex Dump Button

By pressing this button the converted image data will directly be copied into the Memory / Hex Dump page.

CLUT to Hex Dump Button

By pressing this button the converted image color lookup table (if available) will directly be copied into the Memory/Hex Dump page.

VII. Font Manager

- ## Purpose

The *Font Manager* supports the application development with sprites that represents either single letters, numbers and so on or more complex text.

It allows selecting a font type in a specific *Size* and *Text Color*, rendering the entered text onto an empty image with a chosen *Background Color* and save it into a File.
Different *Rendering Modes* are supported to achieve an optimal and smooth effect on the display.

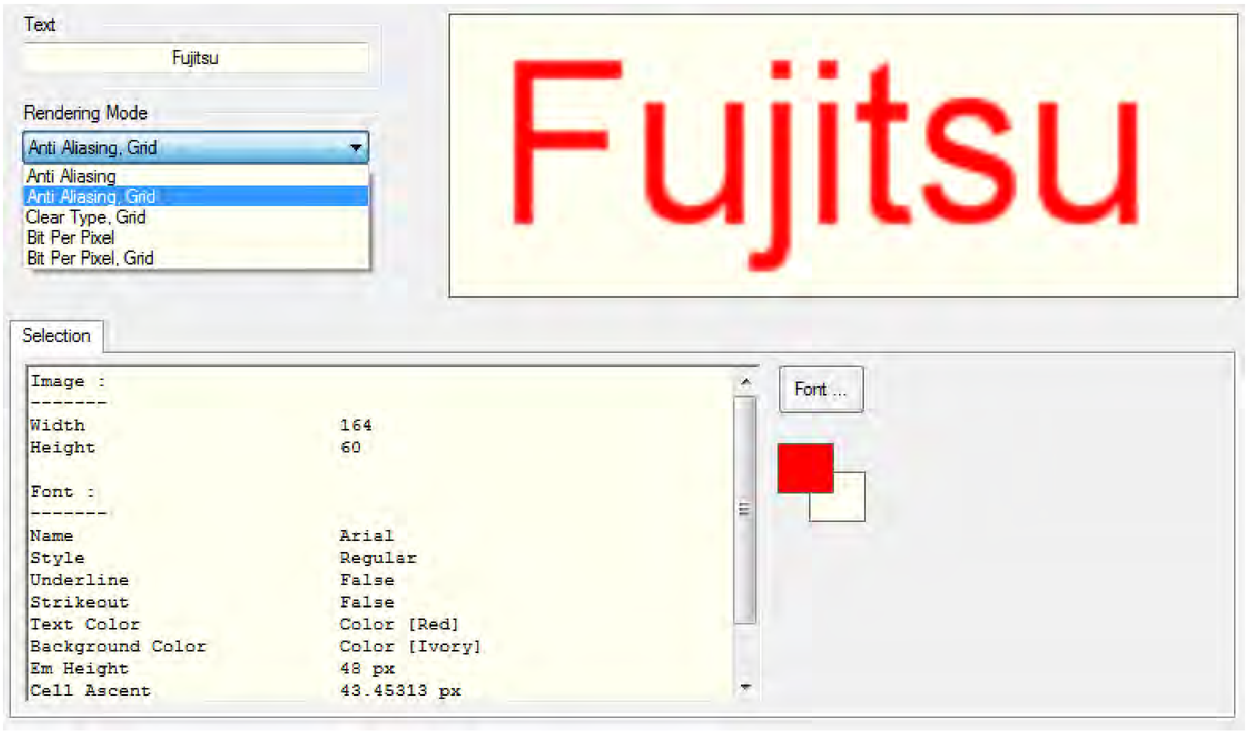
1. Item View

The *Item View* contains different user input controls which supports font selection options as well as different "Text to Bitmap" conversion possibilities.

a. Selection Page

The *Selection Page* is currently the only page available for the *Font Manager*.
On the left side a text box is located which contain relevant information of the selected font as well as the expected dimension of the output image.
The right side contains some controls that allow changing the *Font Style*, *Text Color* and *Background Color* of the destination image.

e.g



```
Image :
-------
Width                          164
Height                         60

Font :
-------
Name                           Arial
Style                          Regular
Underline                      False
Strikeout                      False
Text Color                     Color [Red]
Background Color               Color [Ivory]
Em Height                      48 px
Cell Ascent                    43.45313 px
```

Text Edit Field

Enter the letter, number or complex text to be rendered.

Rendering Mode Combo Box

This *Combo Box* offers all supported modes that are available to render the specific text onto an empty image.
Currently the following rendering modes are supported,

- Anti Aliasing
- Anti Aliasing (Grid)
- Clear Type (Grid)
- Bit Per Pixel
- Bit Per Pixel (Grid)

Font ... Button

By pressing this button a dialog box appears which allows selecting the required font style as well as the font size.

All fonts that are currently part of the *Operating System* are available.

If a new special font should be supported then first copy the font file into the default
font directory - typically located in C:¥WINDOWS¥Fonts.

Foreground Color Selection

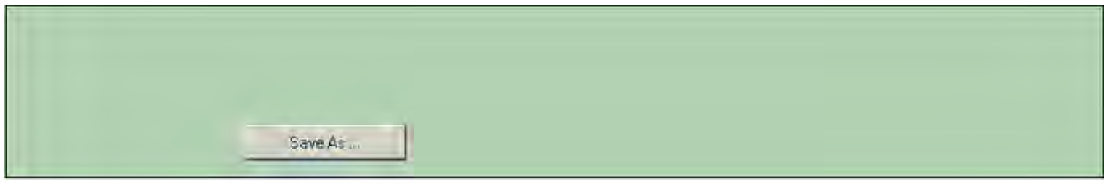
By pressing the left-topmost section beneath the *Font ...* button a dialog box appear that allows selecting the foreground / text color for the selected font.

Background Color Selection

By pressing the right-bottommost section beneath the *Font ...* button a dialog box appear that allows selecting the background color of the target image.

2. Action View

When the *Font Manager* item is selected on the *Selection View* then relevant action controls will appear on the *Action View*.



Save As ... Button

Opens a dialog box which allows selecting a path and a filename to store the entered text in the specified font and rendering mode as a 32 bit image.

The image can be saved in one of the following file formats,

• bmp	Bitmap
• png	Portable Network Graphics
• tiff	Tagged Image File Format
• jpeg	Joint Photographic Expert Group
• gif	Graphics Interchange Format

VIII. Memory / Flash Editor

- **Purpose**

  The Aim of the *Memory / Flash Editor Page* is the support of debugging and validation for both hardware and software - e.g. dumping the memory content to check proper sprite loading.
  But it is also possible to manipulate data in memory by reading / writing either single *Memory Items* (4 Byte each) or complete *Memory Blocks* from / to a specified address.

- **How to,**

  1. **Manipulate Memory Data :**

     Before manipulating memory items the required block must be first read out.
     For this the offset as well as the number of required items must be entered.
     By pressing the *Read Memory Block* button the information will be read out from hardware into the virtual *Item View*.
     Then it is possible to manipulate the items. Attention - Only Virtual.
     When editing is finished it must be written back to the destination / target memory of the real hardware by pressing the corresponding write button.
     The destination offset must not be the same address then reading it in - this will give more flexibility in the manipulation.

1. Item View

The *Item View* of the *Memory / Flash Editor Page* consists of a grid that will display the 32 bit values when dumping a memory or register area.

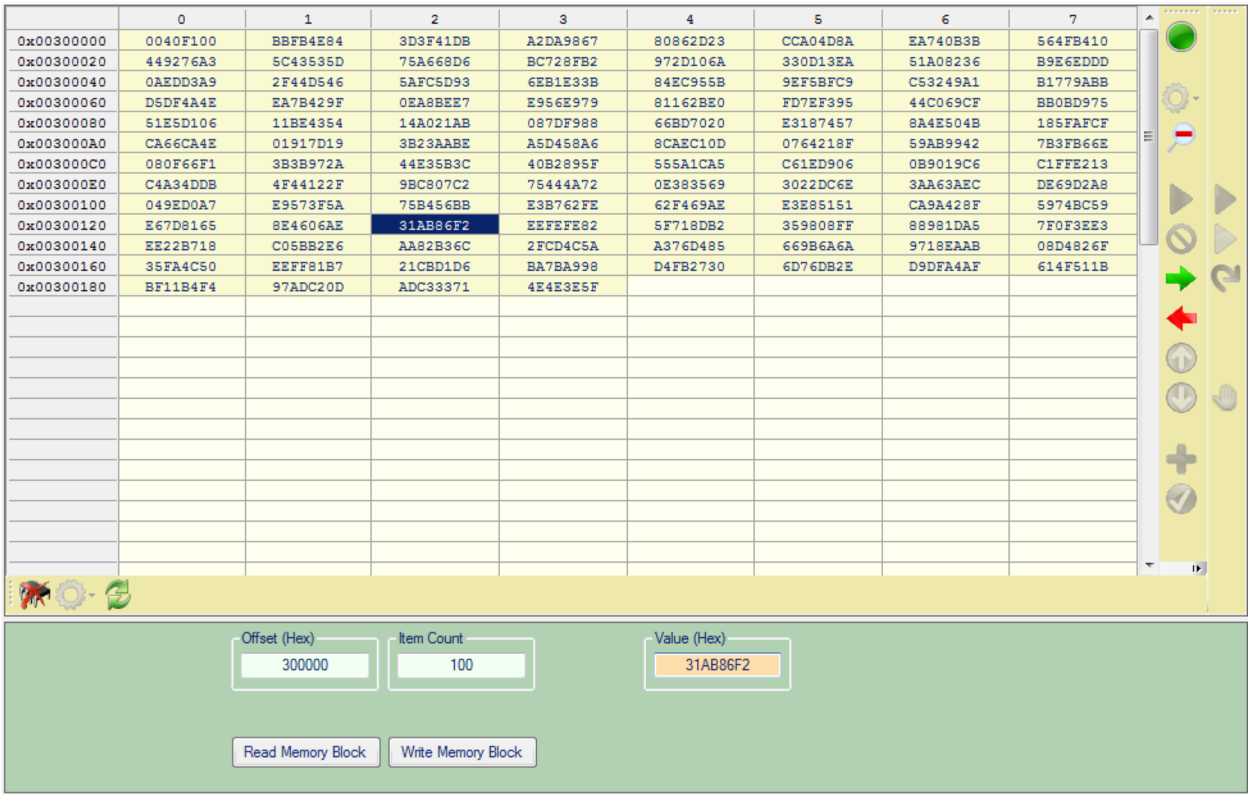
On this page the content of memory or register is always displayed as 32bit values.
That means e.g. when two 32 bit values are displayed as,

0x12345678 0x0ABCDEF0

then memory contains the following byte stream :

78 56 34 12 F0 DE BC 0A

e.g.

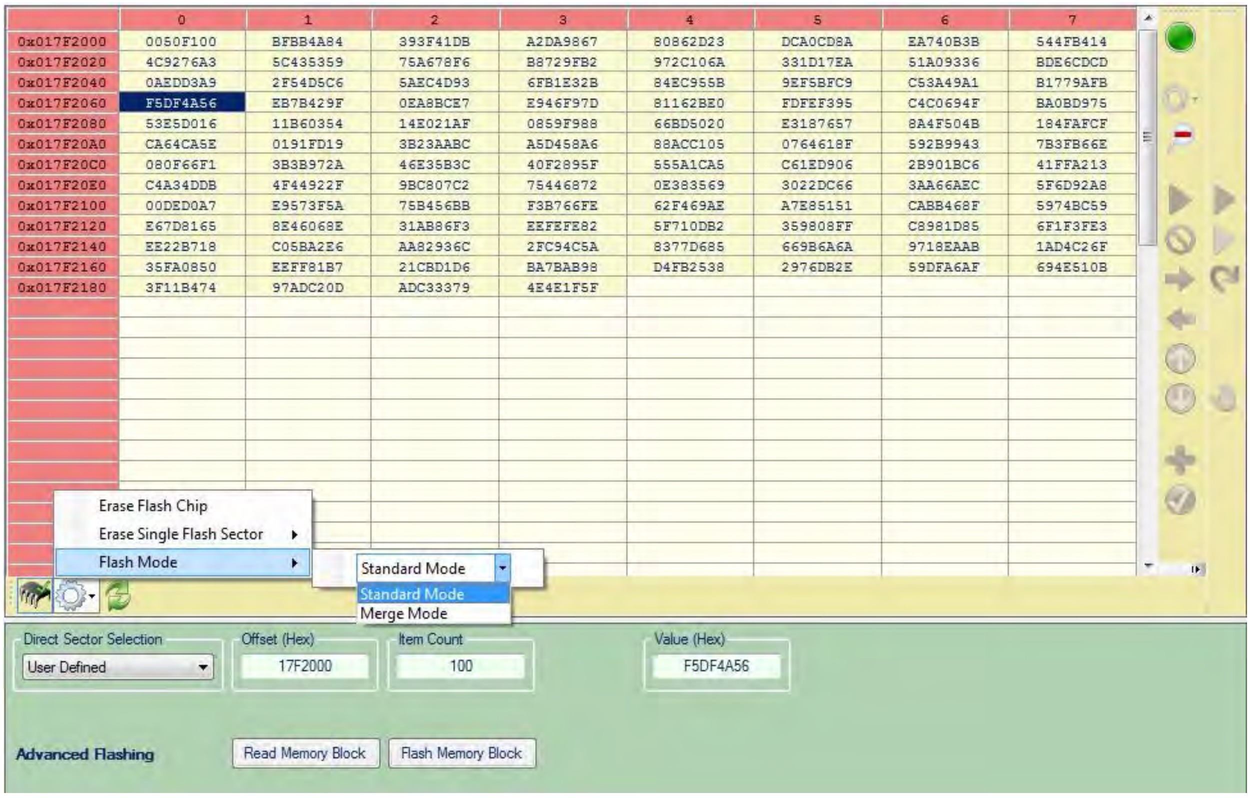


	0	1	2	3	4	5	6	7
0x00300000	0040F100	BBFB4E84	3D3F41DB	A2DA9867	80862D23	CCA04D8A	EA740B3B	564FB410
0x00300020	449276A3	5C43535D	75A668D6	BC728FB2	972D106A	330D13EA	51A08236	B9E6EDDD
0x00300040	0AEDD3A9	2F44D546	5AFC5D93	6EB1E33B	84EC955B	9EF5BFC9	C53249A1	B1779ABB
0x00300060	D5DF4A4E	EA7B429F	0EA8BEE7	E956E979	81162BE0	FD7EF395	44C069CF	BB0BD975
0x00300080	51E5D106	11BE4354	14A021AB	087DF988	66BD7020	E3187457	8A4E504B	185FAFCF
0x003000A0	CA66CA4E	01917D19	3B23AABE	A5D458A6	8CAEC10D	0764218F	59AB9942	7B3FB66E
0x003000C0	080F66F1	3B3B972A	44E35B3C	40B2895F	555A1CA5	C61ED906	0B9019C6	C1FFE213
0x003000E0	C4A34DDB	4F44122F	9BC807C2	75444A72	0E383569	3022DC6E	3AA63AEC	DE69D2A8
0x00300100	049ED0A7	E9573F5A	75B456BB	E3B762FE	62F469AE	E3E85151	CA9A428F	5974BC59
0x00300120	E67D8165	8E4606AE	31AB86F2	EEFEFE82	5F718DB2	359808FF	88981DA5	7F0F3EE3
0x00300140	EE22B718	C05BB2E6	AA82B36C	2FCD4C5A	A376D485	669B6A6A	9718EAAB	08D4826F
0x00300160	35FA4C50	EEFF81B7	21CBD1D6	BA7BA998	D4FB2730	6D76DB2E	D9DFA4AF	614F511B
0x00300180	BF11B4F4	97ADC20D	ADC33371	4E4E3E5F				

Offset (Hex): 300000
Item Count: 100
Value (Hex): 31AB86F2

When Flash Support is available and enabled then it will be additionally visualized that each following access concerns the Flash Memory.

e.g.



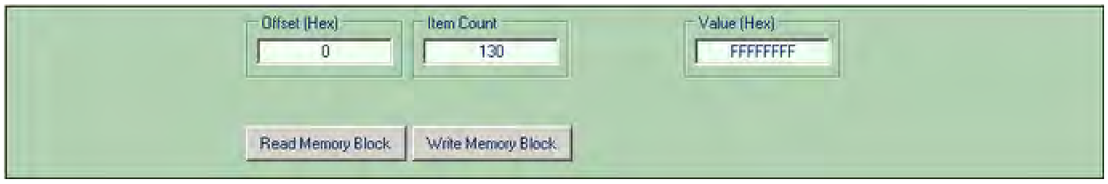
	0	1	2	3	4	5	6	7
0x017F2000	0050F100	BFBB4A84	393F41DB	A2DA9867	80862D23	DCA0CD8A	EA740B3B	544FB414
0x017F2020	4C9276A3	5C435359	75A678F6	B8729FB2	972C106A	331D17EA	51A09336	BDE6CDCD
0x017F2040	0AEDD3A9	2F54D5C6	5AEC4D93	6FB1E32B	84EC955B	9EF5BFC9	C53A49A1	B1779AFB
0x017F2060	F5DF4A56	EB7B429F	0EA8BCE7	E946F97D	81162BE0	FDFEF395	C4C0694F	BA0BD975
0x017F2080	53E5D016	11B60354	14E021AF	0859F988	66BD5020	E3187657	8A4F504B	184FAFCF
0x017F20A0	CA64CA5E	0191FD19	3B23AABC	A5D458A6	88ACC105	0764618F	592B9943	7B3FB66E
0x017F20C0	080F66F1	3B3B972A	46E35B3C	40F2895F	555A1CA5	C61ED906	2B901BC6	41FFA213
0x017F20E0	C4A34DDB	4F44922F	9BC807C2	75446872	0E383569	3022DC66	3AA66AEC	5F6D92A8
0x017F2100	00DED0A7	E9573F5A	75B456BB	F3B766FE	62F469AE	A7E85151	CABB468F	5974BC59
0x017F2120	E67D8165	8E46068E	31AB86F3	EEFEFE82	5F710DB2	359808FF	C8981D85	6F1F3FE3
0x017F2140	EE22B718	C05BA2E6	AA82936C	2FC94C5A	8377D685	669B6A6A	9718EAAB	1AD4C26F
0x017F2160	35FA0850	EEFF81B7	21CBD1D6	BA7BAB98	D4FB2538	2976DB2E	59DFA6AF	694E510B
0x017F2180	3F11B474	97ADC20D	ADC33379	4E4E1F5F				

Direct Sector Selection: User Defined
Offset (Hex): 17F2000
Item Count: 100
Value (Hex): F5DF4A56

2. Action View

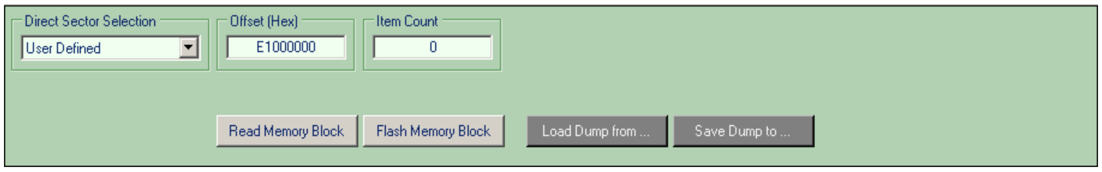
When the *Memory / Flash Editor* item is selected on the *Selection View* then relevant action controls will appear on the *Action View*.

Memory Mode :



Offset (Hex): 0
Item Count: 130
Value (Hex): FFFFFFFF

Flash Mode :



Direct Sector Selection: User Defined
Offset (Hex): E1000000
Item Count: 0

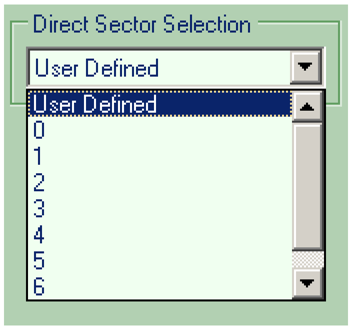
Offset Edit Field

This field allows entering the base offset address (hexadecimal) of the edit field.

The corresponding base address for the entered offset will typically be 0x00000000.
If flash support is enabled the base address will automatically be set to the base address of the of the first flash memory sector to guarantee proper access.

Flash Information :

When flashing is enabled a Sector Selection combo box is available allowing to directly select one of the available sectors.
When a sector is selected the corresponding target address will be automatically entered into the Offset edit field.



Remark :

When entering an offset / address directly it must be ensured that the address is 4 byte aligned - otherwise it will be corrected automatically.

To accept this value press the *Return Key*, otherwise the old value will return.

Items Edit Field

Here the number of items (decimal) should be entered that needs to be dumped.
Currently it is limited to 512 items.
An item - which is the smallest unit to dump - is always 32 bit (4 byte) width.

To accept this value press the *Return Key*, otherwise the old value will return.

Value Edit Field

When in the *Item View* a valid field is selected, then it is possible to enter a new 32 bit value.
To accept this value press the *Return Key*, otherwise the old value will return.

The value will only be changed / manipulated on the visualization and not directly in hardware.
If you want to store the edited Item(s) in hardware please use
either,

- Read / Write controls of the *Action Bar*
  - which allows manipulating the current selected item/field only
  - effective for only few changes

or,

- Read Memory Block / Write Memory Block Buttons controls of the *Action View*
  - which allows to store the complete manipulated *Memory Block* at once

Read Memory Block Button

By pressing this button the entered number of items will be read out from the specified address.

Information :

Depending on the hardware connection speed and the number of items to read this can take some time.

Write / Flash Memory Block Button

By pressing this button all values which are currently available in the *Item View* will be written back to the hardware.

If flash memory support is enabled a state machine will perform some checks / actions before finally writing to the flash memory.
This mechanism is implemented to guarantee an optimal life-time of the flash.
Currently two different Flash Modes are available that can be selected with the "Flash Options" Button of the Support Bar.

Standard Mode

- **Erasing** corresponding sectors.
  The sectors that correspond to the offset and modified data block will be erased.
- **Flashing** corresponding sectors.
  Writing the updated local data storage to the flash memory.

Merge Mode

- **Checking** destination flash memory block if already empty or containing valid data.
  If already empty writing can be performed at once.
- **Comparing** destination flash memory block and edited/manipulated memory block.
  If identical nothing must be done.
- **Saving** current destination flash memory block to local memory.
  To prevent overwriting already valid content of the corresponding flash memory sectors, they will be saved.
- **Modifying** local memory to update the content.
  Writing the modified memory block to the local memory storage of the flash memory.
- **Erasing** corresponding sectors.
  The sectors that correspond to the offset and modified data block will be erased.
- **Flashing** corresponding sectors.
  Writing the updated local data storage to the flash memory.

Information :

Depending on the hardware connection speed and the number of items to write this can take some time.

IX. Memory / Flash Dump

• Purpose

The Aim of the *Memory / Flash Dump* is like the *Memory / Flash Editor Page* the support of debugging and validation for hardware and software.
In comparison to the *Memory / Flash Editor* it is possible to dump much more memory into a *Rich Text Box* and easily copy and store the dump in a file.

• Drag & Drop

It is possible to drag & drop *Binary Files* into the application.
This can be done by selecting a binary file in the window explorer and dragging the file onto the *Item View*.
This can only be done when the *Memory / Flash Dump Item* is selected in the *Selection View*.
(Single File Selection)

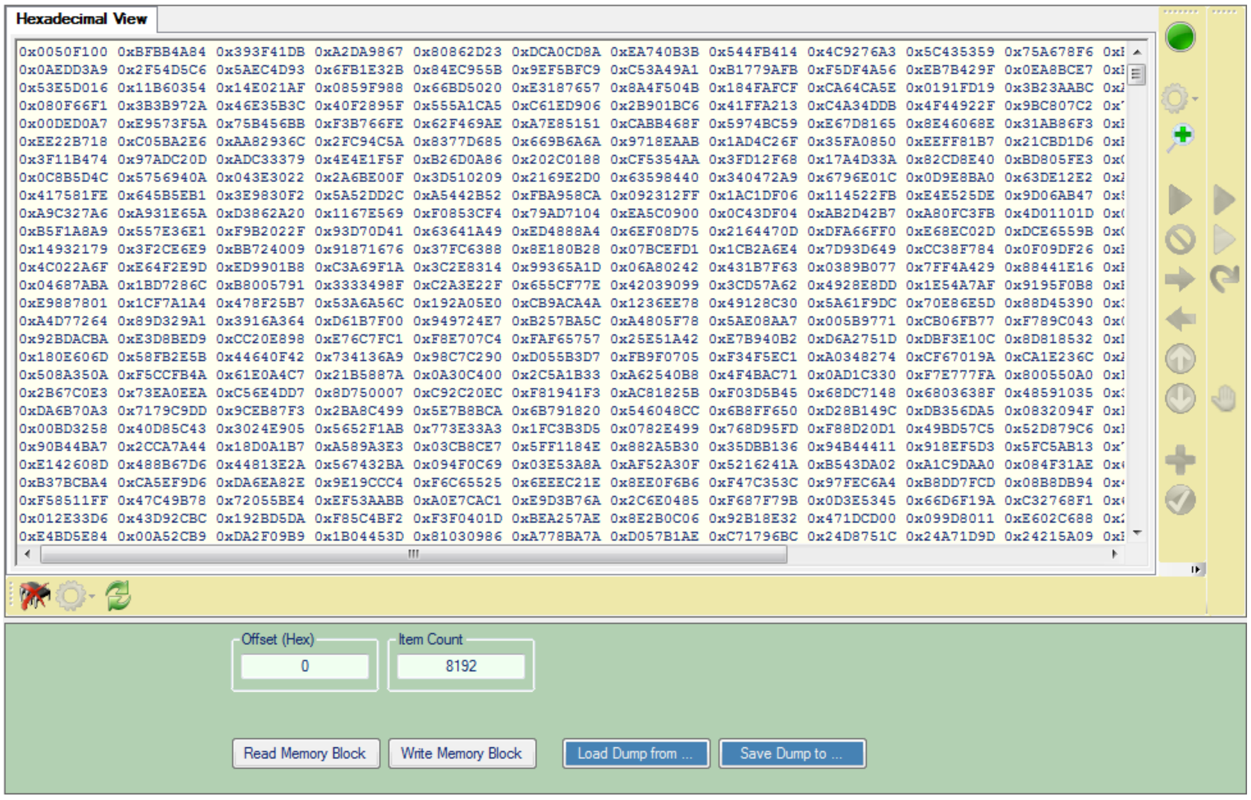
1. Item View

The *Item View* of the *Memory / Flash Dump Page* consists of a *Rich Text Box* that will display the 32 bit values when dumping a memory area.

On this page the content of memory / register or file is always displayed as 32bit values.
That means e.g. when on the *Hex Dump Page* two 32 bit values are displayed as,
0x12345678 0x0ABCDEF0
then the binary file as well as memory contains the following byte stream :
78 56 34 12 F0 DE BC 0A

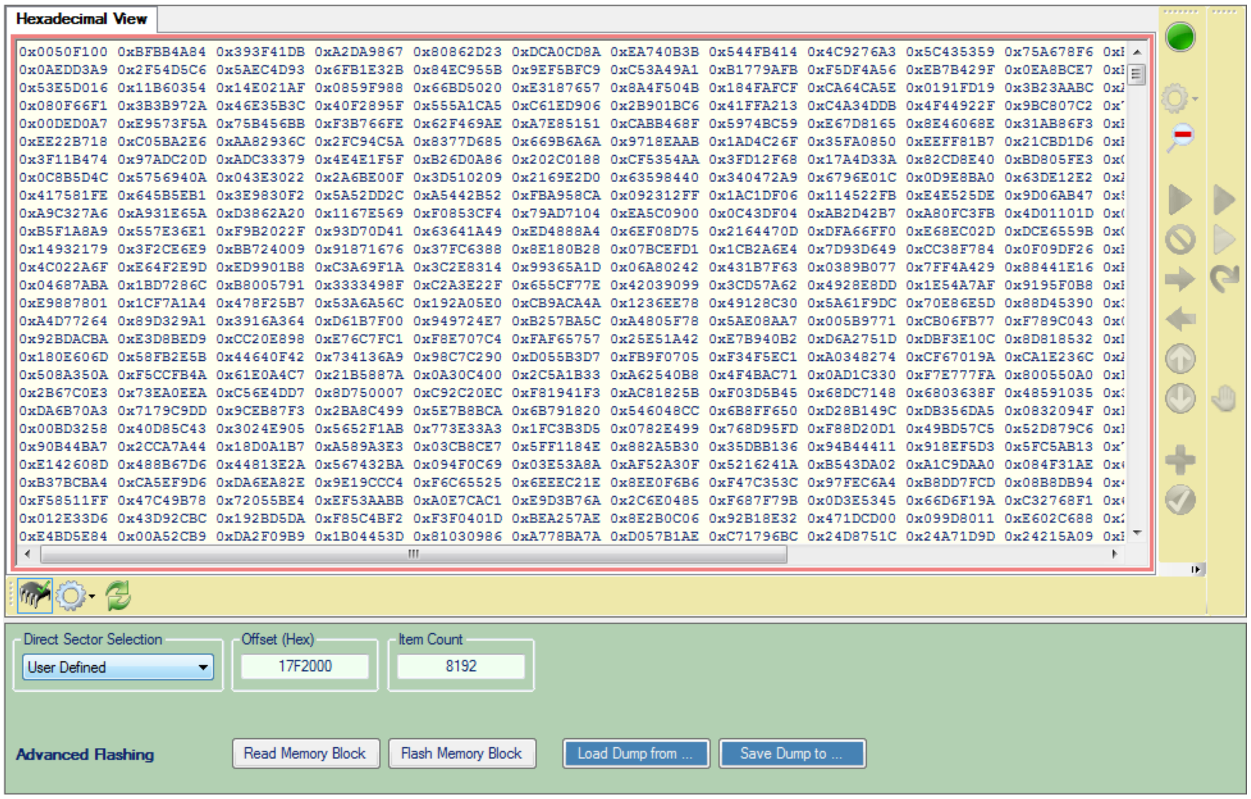
The next screen-shot visualizes the *Item View* when flash support is available and activated.
In case of deactivated flash support the orange border is not present.

e.g.



```
0x0050F100 0xBFBB4A84 0x393F41DB 0xA2DA9867 0x80862D23 0xDCA0CD8A 0xEA740B3B 0x544FB414 0x4C9276A3 0x5C435359 0x75A678F6
0x0AEDD3A9 0x2F54D5C6 0x5AEC4D93 0x6FB1E32B 0x84EC955B 0x9EF5BFC9 0xC53A49A1 0xB1779AFB 0xF5DF4A56 0xEB7B429F 0x0EA8BCE7
0x53E5D016 0x11B60354 0x14E021AF 0x0859F988 0x66BD5020 0xE3187657 0x8A4F504B 0x184FAFCF 0xCA64CA5E 0x0191FD19 0x3B23AABC
0x080F66F1 0x3B3B972A 0x46E35B3C 0x40F2895F 0x555A1CA5 0xC61ED906 0x2B901BC6 0x41FFA213 0xC4A34DDB 0x4F44922F 0x9BC807C2
0x00DED0A7 0xE9573F5A 0x75B456BB 0xF3B766FE 0x62F469AE 0xA7E85151 0xCABB468F 0x5974BC59 0xE67D8165 0x8E46068E 0x31AB86F3
0xEE22B718 0xC05BA2E6 0xAA82936C 0x2FC94C5A 0x8377D685 0x669B6A6A 0x9718EAAB 0x1AD4C26F 0x35FA0850 0xEEFF81B7 0x21CBD1D6
0x3F11B474 0x97ADC20D 0xADC33379 0x4E4E1F5F 0xB26D0A86 0x202C0188 0xCF5354AA 0x3FD12F68 0x17A4D33A 0x82CD8E40 0xBD805FE3
0x0C8B5D4C 0x5756940A 0x043E3022 0x2A6BE00F 0x3D510209 0x2169E2D0 0x63598440 0x340472A9 0x6796E01C 0x0D9E8BA0 0x63DE12E2
0x417581FE 0x645B5EB1 0x3E9830F2 0x5A52DD2C 0xA5442B52 0xFBA958CA 0x092312FF 0x1AC1DF06 0x114522FB 0xE4E525DE 0x9D06AB47
0xA9C327A6 0xA931E65A 0xD3862A20 0x1167E569 0xF0853CF4 0x79AD7104 0xEA5C0900 0x0C43DF04 0xAB2D42B7 0xA80FC3FB 0x4D01101D
0xB5F1A8A9 0x557E36E1 0xF9B2022F 0x93D70D41 0x63641A49 0xED4888A4 0x6EF08D75 0x2164470D 0xDFA66FF0 0xE68EC02D 0xDCE6559B
0x14932179 0x3F2CE6E9 0xBB724009 0x91871676 0x37FC6388 0x8E180B28 0x07BCEFD1 0x1CB2A6E4 0x7D93D649 0xCC38F784 0x0F09DF26
0x4C022A6F 0xE64F2E9D 0xED9901B8 0xC3A69F1A 0x3C2E8314 0x99365A1D 0x06A80242 0x431B7F63 0x0389B077 0x7FF4A429 0x88441E16
0x04687ABA 0x1BD7286C 0xB8005791 0x3333498F 0xC2A3E22F 0x655CF77E 0x42039099 0x3CD57A62 0x4928E8DD 0x1E54A7AF 0x9195F0B8
0xE9887801 0x1CF7A1A4 0x478F25B7 0x53A6A56C 0x192A05E0 0xCB9ACA4A 0x1236EE78 0x49128C30 0x5A61F9DC 0x70E86E5D 0x88D45390
0xA4D77264 0x89D329A1 0x3916A364 0xD61B7F00 0x949724E7 0xB257BA5C 0xA4805F78 0x5AE08AA7 0x005B9771 0xCB06FB77 0xF789C043
0x92BDACBA 0xE3D8BED9 0xCC20E898 0xE76C7FC1 0xF8E707C4 0xFAF65757 0x25E51A42 0xE7B940B2 0xD6A2751D 0xDBF3E10C 0x8D818532
0x180E606D 0x58FB2E5B 0x44640F42 0x734136A9 0x98C7C290 0xD055B3D7 0xFB9F0705 0xF34F5EC1 0xA0348274 0xCF67019A 0xCA1E236C
0x508A350A 0xF5CCFB4A 0x61E0A4C7 0x21B5887A 0x0A30C400 0x2C5A1B33 0xA62540B8 0x4F4BAC71 0x0AD1C330 0xF7E777FA 0x800550A0
0x2B67C0E3 0x73EA0EEA 0xC56E4DD7 0x8D750007 0xC92C20EC 0xF81941F3 0xAC81825B 0xF03D5B45 0x68DC7148 0x6803638F 0x48591035
0xDA6B70A3 0x7179C9DD 0x9CEB87F3 0x2BA8C499 0x5E7B8BCA 0x6B791820 0x546048CC 0x6B8FF650 0xD28B149C 0xDB356DA5 0x0832094F
0x00BD3258 0x40D85C43 0x3024E905 0x5652F1AB 0x773E33A3 0x1FC3B3D5 0x0782E499 0x768D95FD 0xF88D20D1 0x49BD57C5 0x52D879C6
0x90B44BA7 0x2CCA7A44 0x18D0A1B7 0xA589A3E3 0x03CB8CE7 0x5FF1184E 0x882A5B30 0x35DBB136 0x94B44411 0x918EF5D3 0x5FC5AB13
0xE142608D 0x488B67D6 0x44813E2A 0x567432BA 0x094F0C69 0x03E53A8A 0xAF52A30F 0x5216241A 0xB543DA02 0xA1C9DAA0 0x084F31AE
0xB37BCBA4 0xCA5EF9D6 0xDA6EA82E 0x9E19CCC4 0xF6C65525 0x6EEEC21E 0x8EE0F6B6 0xF47C353C 0x97FEC6A4 0xB8DD7FCD 0x08B8DB94
0xF58511FF 0x47C49B78 0x72055BE4 0xEF53AABB 0xA0E7CAC1 0xE9D3B76A 0x2C6E0485 0xF687F79B 0x0D3E5345 0x66D6F19A 0xC32768F1
0x012E33D6 0x43D92CBC 0x192BD5DA 0xF85C4BF2 0xF3F0401D 0xBEA257AE 0x8E2B0C06 0x92B18E32 0x471DCD00 0x099D8011 0xE602C688
0xE4BD5E84 0x00A52CB9 0xDA2F09B9 0x1B04453D 0x81030986 0xA778BA7A 0xD057B1AE 0xC71796BC 0x24D8751C 0x24A71D9D 0x24215A09
```

e.g.



```
0x0050F100 0xBFBB4A84 0x393F41DB 0xA2DA9867 0x80862D23 0xDCA0CD8A 0xEA740B3B 0x544FB414 0x4C9276A3 0x5C435359 0x75A678F6
0x0AEDD3A9 0x2F54D5C6 0x5AEC4D93 0x6FB1E32B 0x84EC955B 0x9EF5BFC9 0xC53A49A1 0xB1779AFB 0xF5DF4A56 0xEB7B429F 0x0EA8BCE7
0x53E5D016 0x11B60354 0x14E021AF 0x0859F988 0x66BD5020 0xE3187657 0x8A4F504B 0x184FAFCF 0xCA64CA5E 0x0191FD19 0x3B23AABC
0x080F66F1 0x3B3B972A 0x46E35B3C 0x40F2895F 0x555A1CA5 0xC61ED906 0x2B901BC6 0x41FFA213 0xC4A34DDB 0x4F44922F 0x9BC807C2
0x00DED0A7 0xE9573F5A 0x75B456BB 0xF3B766FE 0x62F469AE 0xA7E85151 0xCABB468F 0x5974BC59 0xE67D8165 0x8E46068E 0x31AB86F3
0xEE22B718 0xC05BA2E6 0xAA82936C 0x2FC94C5A 0x8377D685 0x669B6A6A 0x9718EAAB 0x1AD4C26F 0x35FA0850 0xEEFF81B7 0x21CBD1D6
0x3F11B474 0x97ADC20D 0xADC33379 0x4E4E1F5F 0xB26D0A86 0x202C0188 0xCF5354AA 0x3FD12F68 0x17A4D33A 0x82CD8E40 0xBD805FE3
0x0C8B5D4C 0x5756940A 0x043E3022 0x2A6BE00F 0x3D510209 0x2169E2D0 0x63598440 0x340472A9 0x6796E01C 0x0D9E8BA0 0x63DE12E2
0x417581FE 0x645B5EB1 0x3E9830F2 0x5A52DD2C 0xA5442B52 0xFBA958CA 0x092312FF 0x1AC1DF06 0x114522FB 0xE4E525DE 0x9D06AB47
0xA9C327A6 0xA931E65A 0xD3862A20 0x1167E569 0xF0853CF4 0x79AD7104 0xEA5C0900 0x0C43DF04 0xAB2D42B7 0xA80FC3FB 0x4D01101D
0xB5F1A8A9 0x557E36E1 0xF9B2022F 0x93D70D41 0x63641A49 0xED4888A4 0x6EF08D75 0x2164470D 0xDFA66FF0 0xE68EC02D 0xDCE6559B
0x14932179 0x3F2CE6E9 0xBB724009 0x91871676 0x37FC6388 0x8E180B28 0x07BCEFD1 0x1CB2A6E4 0x7D93D649 0xCC38F784 0x0F09DF26
0x4C022A6F 0xE64F2E9D 0xED9901B8 0xC3A69F1A 0x3C2E8314 0x99365A1D 0x06A80242 0x431B7F63 0x0389B077 0x7FF4A429 0x88441E16
0x04687ABA 0x1BD7286C 0xB8005791 0x3333498F 0xC2A3E22F 0x655CF77E 0x42039099 0x3CD57A62 0x4928E8DD 0x1E54A7AF 0x9195F0B8
0xE9887801 0x1CF7A1A4 0x478F25B7 0x53A6A56C 0x192A05E0 0xCB9ACA4A 0x1236EE78 0x49128C30 0x5A61F9DC 0x70E86E5D 0x88D45390
0xA4D77264 0x89D329A1 0x3916A364 0xD61B7F00 0x949724E7 0xB257BA5C 0xA4805F78 0x5AE08AA7 0x005B9771 0xCB06FB77 0xF789C043
0x92BDACBA 0xE3D8BED9 0xCC20E898 0xE76C7FC1 0xF8E707C4 0xFAF65757 0x25E51A42 0xE7B940B2 0xD6A2751D 0xDBF3E10C 0x8D818532
0x180E606D 0x58FB2E5B 0x44640F42 0x734136A9 0x98C7C290 0xD055B3D7 0xFB9F0705 0xF34F5EC1 0xA0348274 0xCF67019A 0xCA1E236C
0x508A350A 0xF5CCFB4A 0x61E0A4C7 0x21B5887A 0x0A30C400 0x2C5A1B33 0xA62540B8 0x4F4BAC71 0x0AD1C330 0xF7E777FA 0x800550A0
0x2B67C0E3 0x73EA0EEA 0xC56E4DD7 0x8D750007 0xC92C20EC 0xF81941F3 0xAC81825B 0xF03D5B45 0x68DC7148 0x6803638F 0x48591035
0xDA6B70A3 0x7179C9DD 0x9CEB87F3 0x2BA8C499 0x5E7B8BCA 0x6B791820 0x546048CC 0x6B8FF650 0xD28B149C 0xDB356DA5 0x0832094F
0x00BD3258 0x40D85C43 0x3024E905 0x5652F1AB 0x773E33A3 0x1FC3B3D5 0x0782E499 0x768D95FD 0xF88D20D1 0x49BD57C5 0x52D879C6
0x90B44BA7 0x2CCA7A44 0x18D0A1B7 0xA589A3E3 0x03CB8CE7 0x5FF1184E 0x882A5B30 0x35DBB136 0x94B44411 0x918EF5D3 0x5FC5AB13
0xE142608D 0x488B67D6 0x44813E2A 0x567432BA 0x094F0C69 0x03E53A8A 0xAF52A30F 0x5216241A 0xB543DA02 0xA1C9DAA0 0x084F31AE
0xB37BCBA4 0xCA5EF9D6 0xDA6EA82E 0x9E19CCC4 0xF6C65525 0x6EEEC21E 0x8EE0F6B6 0xF47C353C 0x97FEC6A4 0xB8DD7FCD 0x08B8DB94
0xF58511FF 0x47C49B78 0x72055BE4 0xEF53AABB 0xA0E7CAC1 0xE9D3B76A 0x2C6E0485 0xF687F79B 0x0D3E5345 0x66D6F19A 0xC32768F1
0x012E33D6 0x43D92CBC 0x192BD5DA 0xF85C4BF2 0xF3F0401D 0xBEA257AE 0x8E2B0C06 0x92B18E32 0x471DCD00 0x099D8011 0xE602C688
0xE4BD5E84 0x00A52CB9 0xDA2F09B9 0x1B04453D 0x81030986 0xA778BA7A 0xD057B1AE 0xC71796BC 0x24D8751C 0x24A71D9D 0x24215A09
```

Direct Sector Selection: User Defined

Offset (Hex): 17F2000

Item Count: 8192

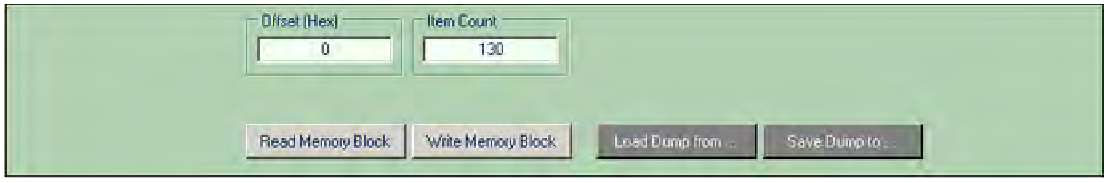
a. Hex Dump Page

This Page is is currently the only one available for the Hex Dump Item.

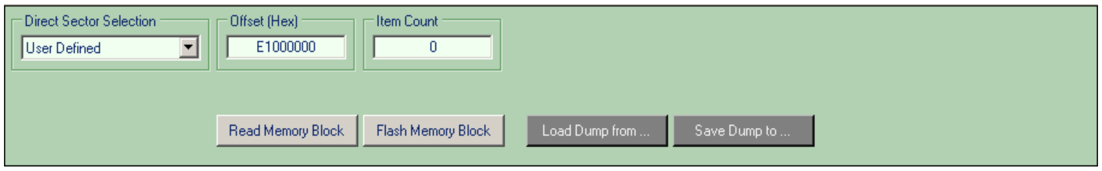
2. Action View

When the *Memory / Flash Dump* item is selected on the *Selection View* then relevant action controls will appear on the *Action View*.

Memory Mode :


Offset (Hex): 0
Item Count: 130

Flash Mode :


Direct Sector Selection: User Defined
Offset (Hex): E1000000
Item Count: 0

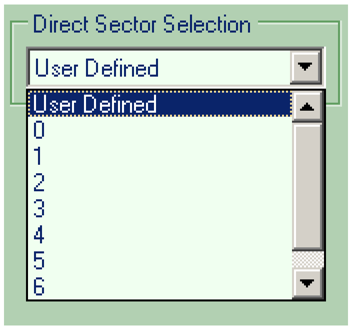
Offset Edit Field

This field allows entering the offset address (hexadecimal) of the memory dump.

The corresponding base address for the entered offset will typically be 0x00000000.
If flash support is enabled the base address will automatically be set to the base address of the of the first flash memory sector to guarantee proper access.

Flash Information :

When flashing is enabled a Sector Selection combo box is available allowing to directly select one of the available sectors.
When a sector is selected the corresponding target address will be automatically entered into the Offset edit field.



Remark :

When entering an offset / address directly it must be ensured that the address is 4 byte aligned - otherwise it will be corrected automatically.
To accept this value press the *Return Key*, otherwise the old value will return.

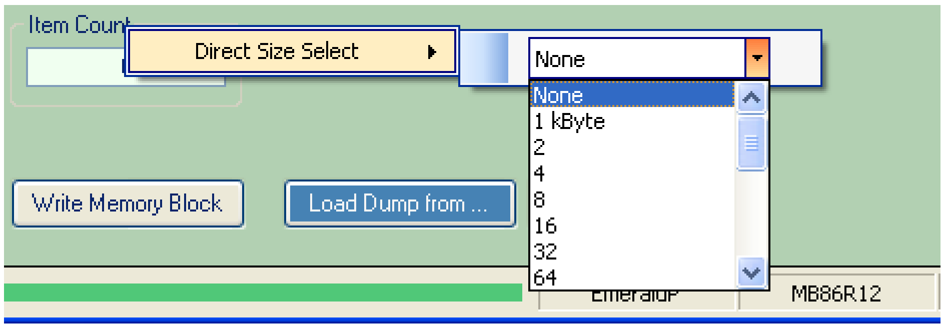
Items Edit Field

Here the number of items (decimal) should be entered that needs to be dumped.
Currently it is limited to 67108864 items. (equals 256MB)
An item - which is the smallest unit to dump - is always 32 bit (4 byte) width.

To accept this value press the *Return Key*, otherwise the old value will return.

Context Menu Helper Function :

For helping selecting a size directly a context menu is available which will be displayed when right clicking on the "Item Count" text.
Here a list with some predefined memory size are available that can be selected directly.
After selecting an equivalent number of items are entered into the field which must be accepted by pressing the return key afterwards.



Read Memory Block Button

By pressing this button the entered number of items will be read out from the specified address.

Information :

Depending on the hardware connection speed and the number of items to read this can take some time.

Write / Flash Memory Block Button

By pressing this button all values which are currently available in the *Item View* will be written back to the hardware.

If flash memory support is enabled a state machine will perform some checks / actions before finally writing to the flash memory.
This mechanism is implemented to guarantee an optimal life-time of the flash.
Currently two different Flash Modes are available that can be selected with the "Flash Options" Button of the Support Bar.

Standard Mode

- **Erasing** corresponding sectors.
  The sectors that correspond to the offset and modified data block will be erased.
- **Flashing** corresponding sectors.
  Writing the updated local data storage to the flash memory.

Merge Mode

- **Checking** destination flash memory block if already empty or containing valid data.
  If already empty writing can be performed at once.
- **Comparing** destination flash memory block and edited/manipulated memory block.
  If identical nothing must be done.
- **Saving** current destination flash memory block to local memory.
  To prevent overwriting already valid content of the corresponding flash memory sectors, they will be saved.
- **Modifying** local memory to update the content.
  Writing the modified memory block to the local memory storage of the flash memory.
- **Erasing** corresponding sectors.
  The sectors that correspond to the offset and modified data block will be erased.
- **Flashing** corresponding sectors.
  Writing the updated local data storage to the flash memory.

Information :

Depending on the hardware connection speed and the number of items to write this can take some time.

Load Dump from ... Button

Press this button if loading and converting a file to a valid hex dump is required.
The loaded content of the file will then be displayed in the *Item View*.

Currently supported are the following file formats,

• .gdc32dat	Standard hex dump file
• .bin	Standard hex dump file
• .mhx	Standard hex dump file
• .gdcseq	Standard register sequence file (Indigo only)
• .gdcicmd	Special Indigo sequence file (command list)

General Information :

All sequence files will be read out and interpreted so that the resulting hex values represent a valid command list.

File Information :

.gdc32dat
.bin

Is a standard binary file in which the values are stored as UInt32.
E.g. when on the *Hex Dump Page* two 32 bit values are displayed as,

0x12345678 0x0ABCDEF0

then the binary output file contains the following byte stream :

78 56 34 12 F0 DE BC 0A

.mhx

File format with a code redundancy check.
Typically storing data as 8 bit values.
Currently only supported for data content which are multiples of 32 bit.

.par

This is a human readable and easy to edit text file format which can also be used in the Register Sequencer.
For more Information please refer to the *Par File Format* information page in the ***Customer Information*** section.

Save Dump to ... Button

With this button it is possible to store the hex dump that was read in from memory/registers or from flash memory into a binary file.

The typical file extension for binary files is .gdc32dat or .bin.

File Information :

.gdc32dat
.bin

Is a standard binary file in which the values are stored as UInt32.
E.g. when on the *Hex Dump Page* two 32 bit values are displayed as,

0x12345678 0x0ABCDEF0

then the binary output file contains the following byte stream,
(reviewed by an Hex Viewer/Editor)

78 56 34 12 F0 DE BC 0A

.mhx

File format with a code redundancy check.
Typically storing data as 8 bit values.
Currently only supported for data content which are multiples of 32 bit.

.par

This is a human readable and easy to edit text file format which can also be used in the Register Sequencer.
For more Information please refer to the *Par File Format* information page in the ***Customer Information*** section.

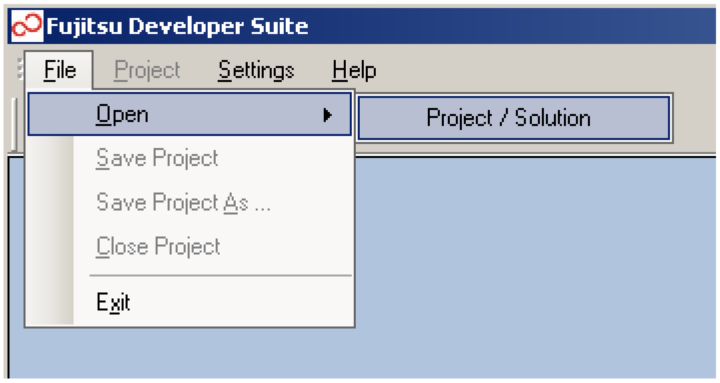
X. How to ...

1. Startup

1. Start Application

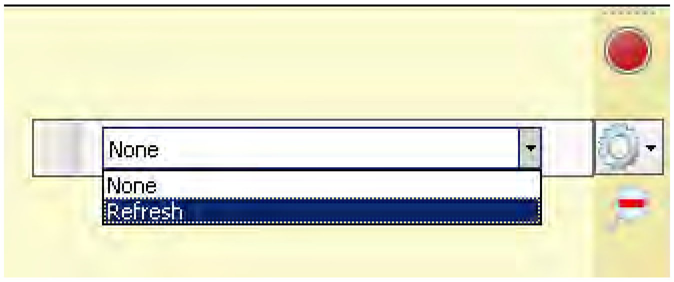
2. Select Project File

Corresponding to the required Target and Connection.



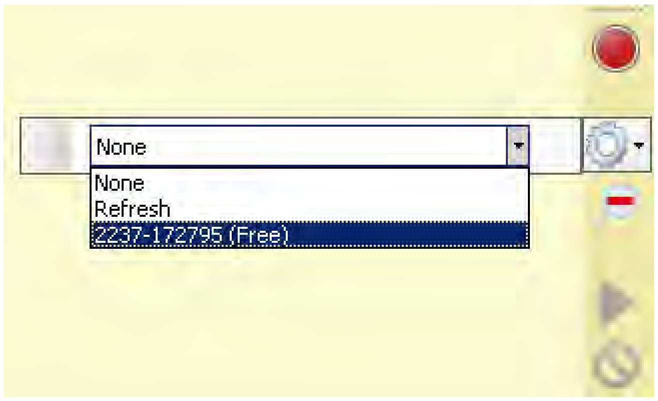
3. Scan for target connection devices

Not available for Ethernet.



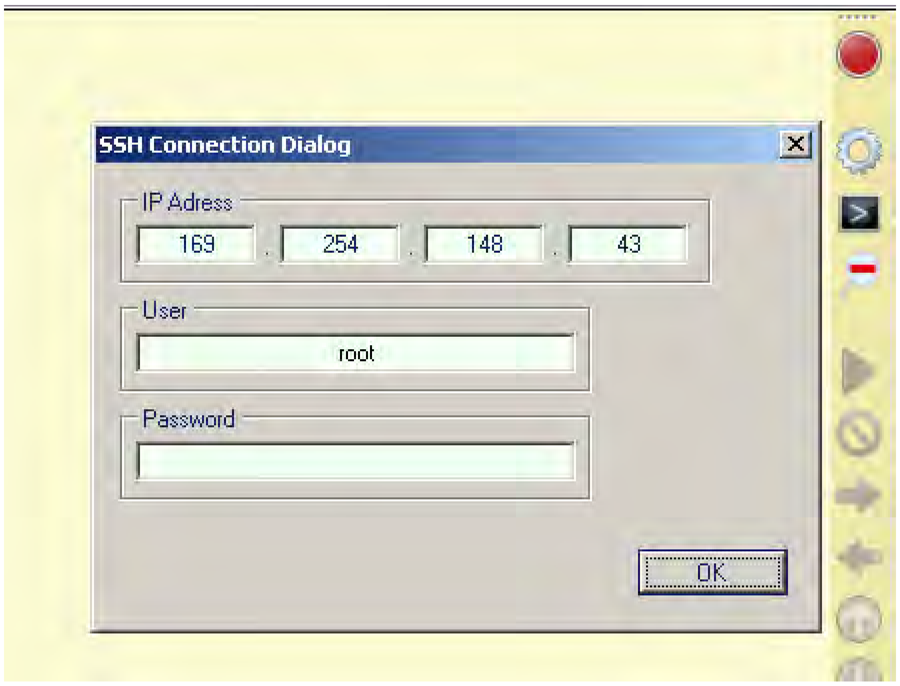
4a. Select required target connection device

Not available for Ethernet.
Example for SPI, Aardvark.
Multiple Devices in List possible.


None
None
Refresh
2237-172795 (Free)

4b. Configure Ethernet connection

Not available for SPI.
Not available for JTAG.
Typically it is predefined and ready for beeing used with the Fujitsu Linux BSP.



SSH Connection Dialog

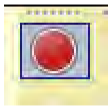
IP Adress: 169 . 254 . 148 . 43

User: root

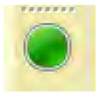
Password:

5. Establish the connection

Ensure that the target is properly connected to the connection device.
Press the button below to establish the connection.



If the target has established a connection and is well initialized the button change its color and signalizes connection status.



FUJITSU

2. Flashing

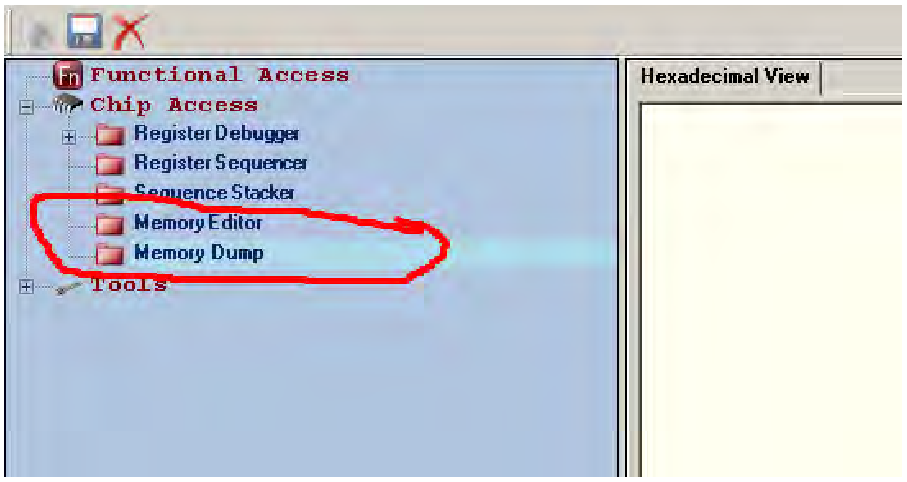
Remarks

- Switching to Flash Operation is only possible when a connection to the target device is already established.
  Please check the **How to ... Startup** section for more information.

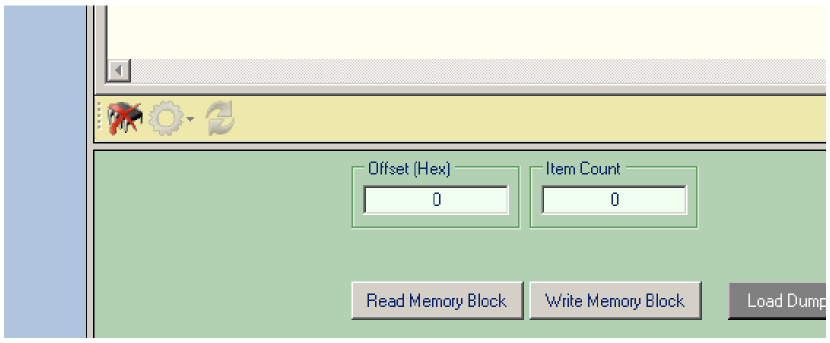
- Flashing is supported on the
  Memory / Flash Editor Page
  Memory Flash Dump Page
  of the application.

- Not all target chips support flashing. (e.g. Ruby)
- Not all connection types support flashing. (e.g. Ethernet)

1. Select the page that should be used for flashing



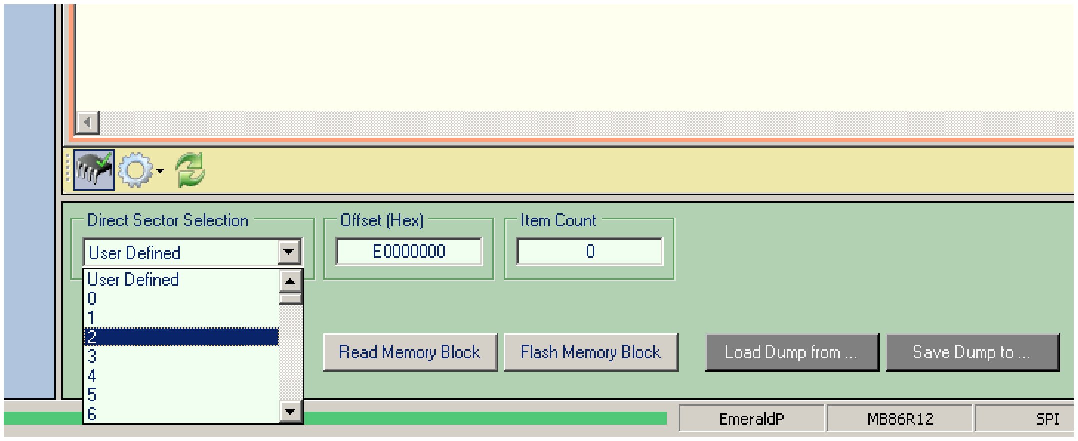
2. Press the Flash Enable/Disable Button in the Support Bar



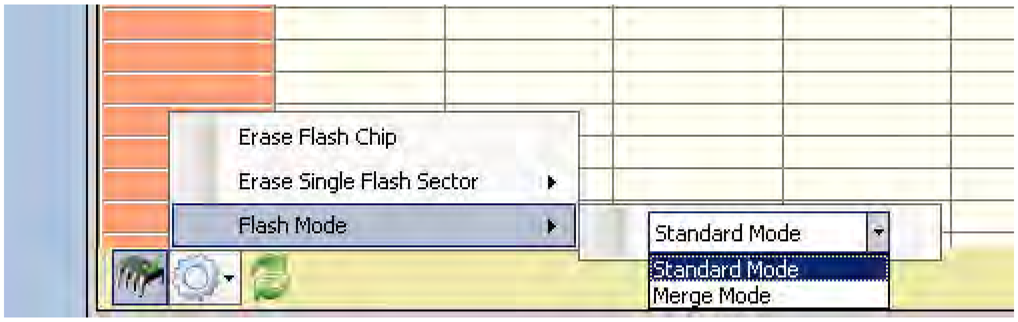
Flash Mode disabled.

Flash Mode enabled.

3. Select a flash sector as base for operation



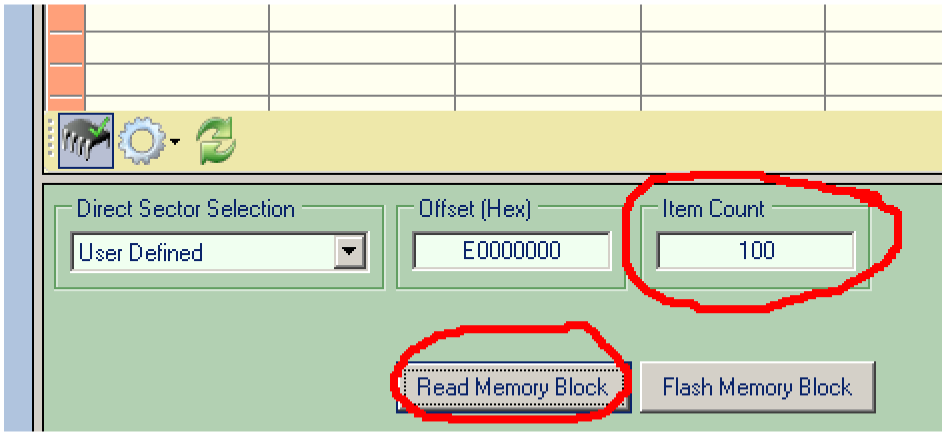
4. Select the required Flash Mode



5. Flash Editor

a. Read a Flash Block

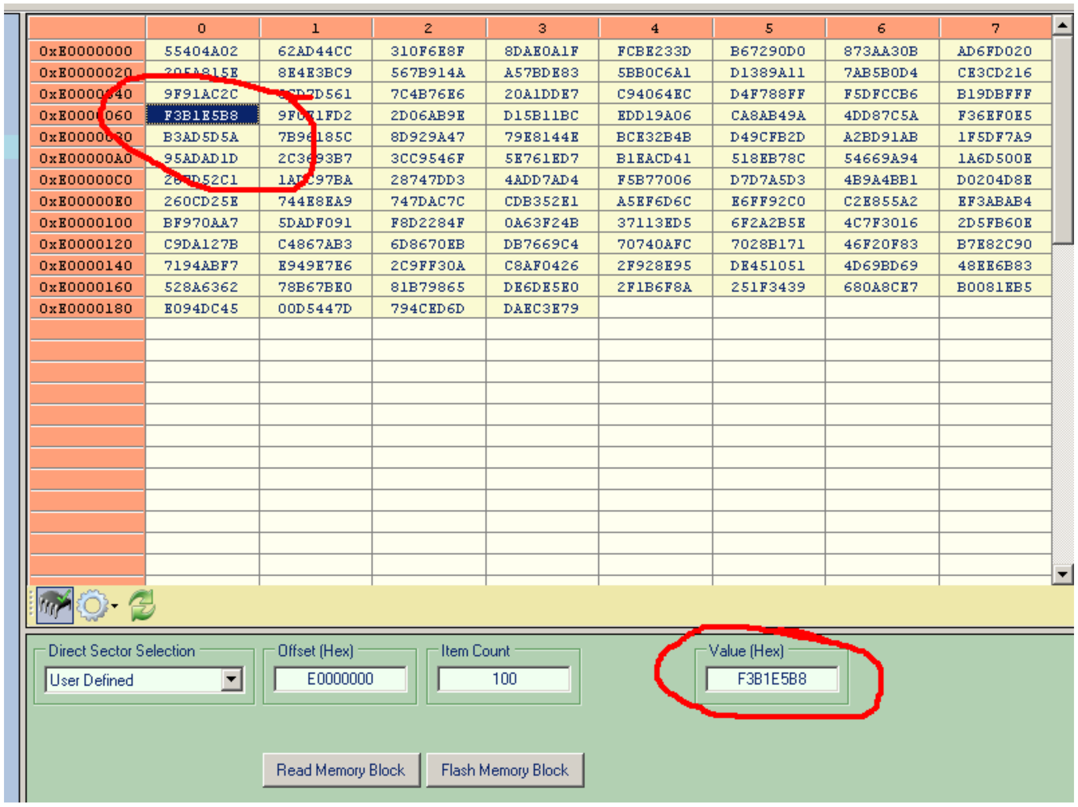
Enter number of items to read.
An item is 4Byte.
A maximum of 512 items can be displayed at once.



Direct Sector Selection	Offset (Hex)	Item Count
User Defined	E0000000	100

b. Modify entries of the Flash Block

After Reading the required entries are displayed in the Grid View.
Select an entry in the view and press enter to manipulate the Data.

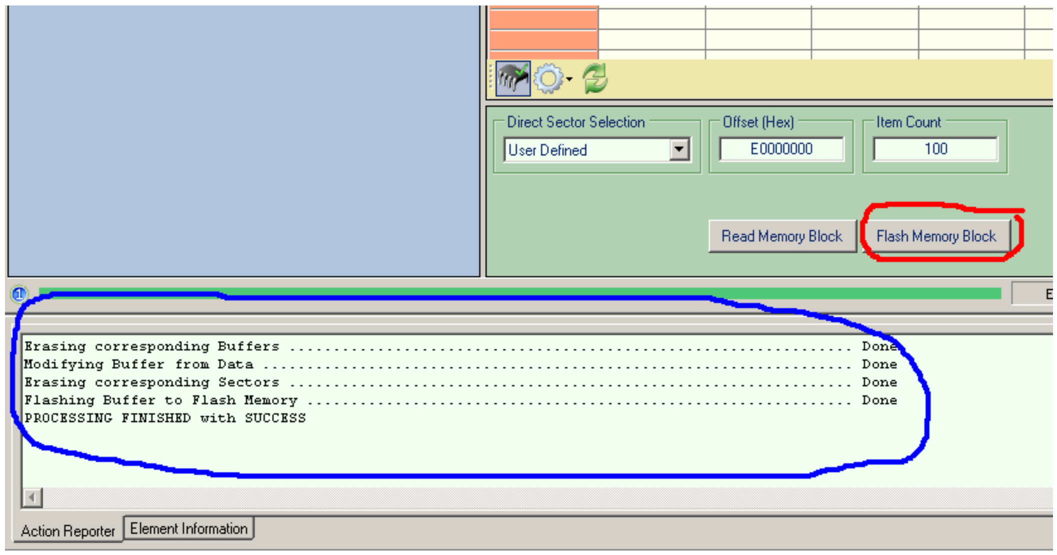


	0	1	2	3	4	5	6	7
0xE0000000	55404A02	62AD44CC	310F6E8F	8DAE0A1F	FCBE233D	B67290D0	873AA30B	AD6FD020
0xE0000020	305A815E	8E4E3BC9	567B914A	A57BDE83	5BB0C6A1	D1389A11	7AB5B0D4	CE3CD216
0xE0000040	9F91AC2C	[illegible]	7C4B76E6	20A1DDE7	C94064EC	D4F788FF	F5DFCCB6	B19DBFFF
0xE0000060	F3B1E5B8	[illegible]	2D06AB9E	D15B11BC	EDD19A06	CA8AB49A	4DD87C5A	F36EF0E5
0xE0000080	B3AD5D5A	[illegible]	8D929A47	79E8144E	BCE32B4B	D49CFB2D	A2BD91AB	1F5DF7A9
0xE00000A0	95ADAD1D	[illegible]	3CC9546F	5E761ED7	B1EACD41	518EB78C	54669A94	1A6D500E
0xE00000C0	[illegible]	[illegible]	28747DD3	4ADD7AD4	F5B77006	D7D7A5D3	4B9A4BB1	D0204D8E
0xE00000E0	260CD25E	744E8EA9	747DAC7C	CDB352E1	A5EF6D6C	E6FF92C0	C2E855A2	EF3ABAB4
0xE0000100	BF970AA7	5DADF091	F8D2284F	0A63F24B	37113ED5	6F2A2B5E	4C7F3016	2D5FB60E
0xE0000120	C9DA127B	C4867AB3	6D8670EB	DB7669C4	70740AFC	7028B171	46F20F83	B7E82C90
0xE0000140	7194ABF7	E949E7E6	2C9FF30A	C8AF0426	2F928E95	DE451051	4D69BD69	48EE6B83
0xE0000160	528A6362	78B67BE0	81B79865	DE6DE5E0	2F1B6F8A	251F3439	680A8CE7	B0081EB5
0xE0000180	E094DC45	00D5447D	794CED6D	DAEC3E79				

Direct Sector Selection	Offset (Hex)	Item Count	Value (Hex)
User Defined	E0000000	100	F3B1E5B8

c. Write Flash and check Reporter

Flash the data content in the grid view with the number of items in the item count field to the entered flash address.



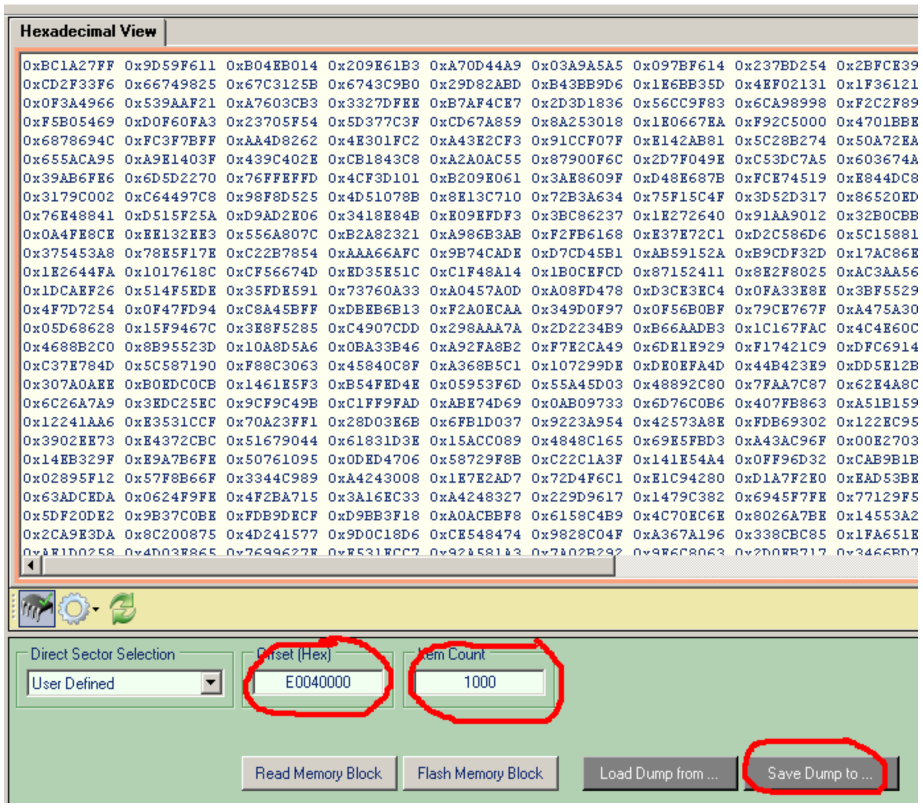
```
Erasing corresponding Buffers ................................................................ Done
Modifying Buffer from Data ................................................................... Done
Erasing corresponding Sectors ................................................................ Done
Flashing Buffer to Flash Memory .............................................................. Done
PROCESSING FINISHED with SUCCESS
```

6. Flash Dump

Different Flash File Formats will be supported that can be read into the hex view.

a. Read Flash and Save.

Enter Flash Address directly into the edit box or choose one of the predefined sectors with the Sector Selection Combo Box.
Enter the number of items to be read and start reading the Flash Block.



```
0xBC1A27FF 0x9D59F611 0xB04EB014 0x209E61B3 0xA70D44A9 0x03A9A5A5 0x097BF614 0x237BD254 0x2BFCE39
0xCD2F33F6 0x66749825 0x67C3125B 0x6743C9B0 0x29D82ABD 0xB43BB9D6 0x1E6BB35D 0x4EF02131 0x1F36121
0x0F3A4966 0x539AAF21 0xA7603CB3 0x3327DFEE 0xB7AF4CE7 0x2D3D1836 0x56CC9F83 0x6CA98998 0xF2C2F89
0xF5B05469 0xD0F60FA3 0x23705F54 0x5D377C3F 0xCD67A859 0x8A253018 0x1E0667EA 0xF92C5000 0x4701BBE
0x6878694C 0xFC3F7BFF 0xAA4D8262 0x4E301FC2 0xA43E2CF3 0x91CCF07F 0xE142AB81 0x5C28B274 0x50A72EA
0x655ACA95 0xA9E1403F 0x439C402E 0xCB1843C8 0xA2A0AC55 0x87900F6C 0x2D7F049E 0xC53DC7A5 0x603674A
0x39AB6FE6 0x6D5D2270 0x76FFEFFD 0x4CF3D101 0xB209E061 0x3AE8609F 0xD48E687B 0xFCE74519 0xE844DC8
0x3179C002 0xC64497C8 0x98F8D525 0x4D51078B 0x8E13C710 0x72B3A634 0x75F15C4F 0x3D52D317 0x86520ED
0x76E48841 0xD515F25A 0xD9AD2E06 0x3418E84B 0xE09EFDF3 0x3BC86237 0x1E272640 0x91AA9012 0x32B0CBB
0x0A4FE8CE 0xEE132EE3 0x556A807C 0xB2A82321 0xA986B3AB 0xF2FB6168 0xE37E72C1 0xD2C586D6 0x5C15881
0x375453A8 0x78E5F17E 0xC22B7854 0xAAA66AFC 0x9B74CADE 0xD7CD45B1 0xAB59152A 0xB9CDF32D 0x17AC86E
0x1E2644FA 0x1017618C 0xCF56674D 0xED35E51C 0xC1F48A14 0x1B0CEFCD 0x87152411 0x8E2F8025 0xAC3AA56
0x1DCAEF26 0x514F5EDE 0x35FDE591 0x73760A33 0xA0457A0D 0xA08FD478 0xD3CE3EC4 0x0FA33E8E 0x3BF5529
0x4F7D7254 0x0F47FD94 0xC8A45BFF 0xDBEB6B13 0xF2A0ECAA 0x349D0F97 0x0F56B0BF 0x79CE767F 0xA475A30
0x05D68628 0x15F9467C 0x3E8F5285 0xC4907CDD 0x298AAA7A 0x2D2234B9 0xB66AADB3 0x1C167FAC 0x4C4E60C
0x4688B2C0 0x8B95523D 0x10A8D5A6 0x0BA33B46 0xA92FA8B2 0xF7E2CA49 0x6DE1E929 0xF17421C9 0xDFC6914
0xC37E784D 0x5C587190 0xF88C3063 0x45840C8F 0xA368B5C1 0x107299DE 0xDE0EFA4D 0x44B423E9 0xDD5E12B
0x307A0AEE 0xB0EDC0CB 0x1461E5F3 0xB54FED4E 0x05953F6D 0x55A45D03 0x48892C80 0x7FAA7C87 0x62E4A8C
0x6C26A7A9 0x3EDC25EC 0x9CF9C49B 0xC1FF9FAD 0xABE74D69 0x0AB09733 0x6D76C0B6 0x407FB863 0xA51B159
0x12241AA6 0xE3531CCF 0x70A23FF1 0x28D03E6B 0x6FB1D037 0x9223A954 0x42573A8E 0xFDB69302 0x122EC95
0x3902EE73 0xE4372CBC 0x51679044 0x61831D3E 0x15ACC089 0x4848C165 0x69E5FBD3 0xA43AC96F 0x00E2703
0x14EB329F 0xE9A7B6FE 0x50761095 0x0DED4706 0x58729F8B 0xC22C1A3F 0x141E54A4 0x0FF96D32 0xCAB9B1B
0x02895F12 0x57F8B66F 0x3344C989 0xA4243008 0x1E7E2AD7 0x72D4F6C1 0xE1C94280 0xD1A7F2E0 0xEAD53BE
0x63ADCEDA 0x0624F9FE 0x4F2BA715 0x3A16EC33 0xA4248327 0x229D9617 0x1479C382 0x6945F7FE 0x77129F5
0x5DF20DE2 0x9B37C0BE 0xFDB9DECF 0xD9BB3F18 0xA0ACBBF8 0x6158C4B9 0x4C70EC6E 0x8026A7BE 0x14553A2
0x2CA9E3DA 0x8C200875 0x4D241577 0x9D0C18D6 0xCE548474 0x9828C04F 0xA367A196 0x338CBC85 0x1FA651E
```

Direct Sector Selection: User Defined

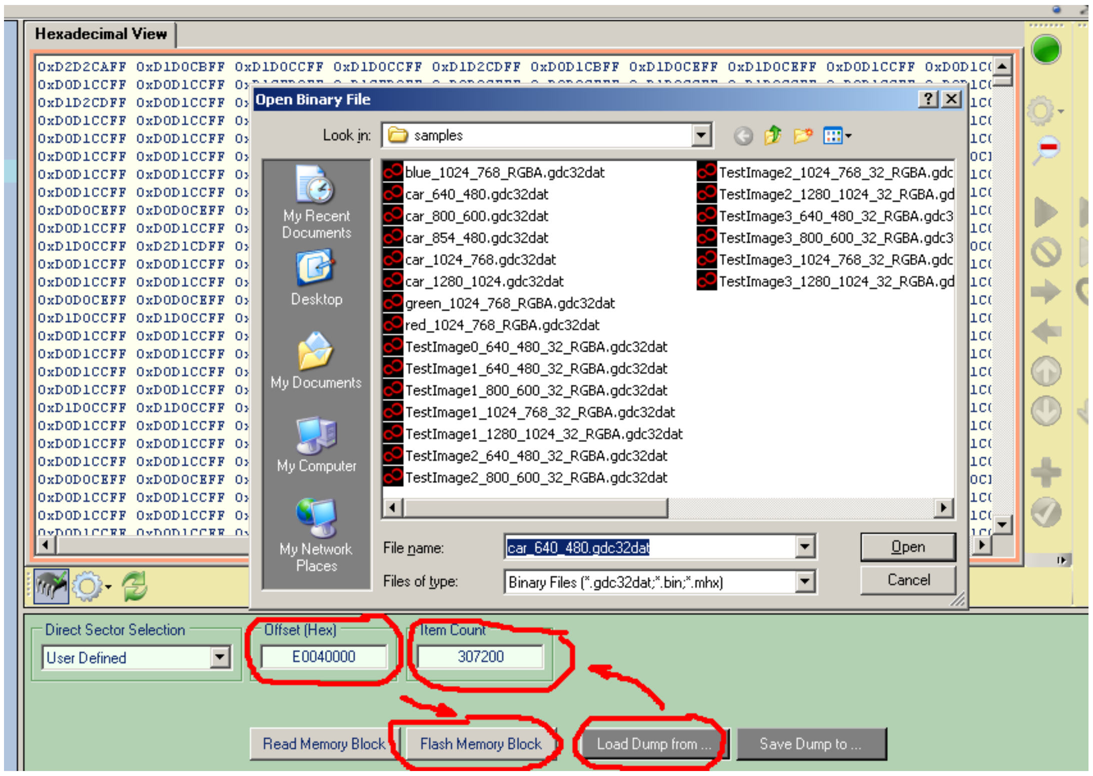
Offset (Hex): E0040000

Item Count: 1000

After reading the flash dump can be saved into a file.

b. Read File and Flash

Enter Flash Address directly into the edit box or choose one of the predefined sectors with the Sector Selection Combo Box.
Enter the number of items to be read and start reading the Flash Block.



```
0xD2D2CAFF 0xD1D0CBFF 0xD1D0CCFF 0xD1D0CCFF 0xD1D2CDFF 0xD0D1CBFF 0xD1D0CEFF 0xD1D0CEFF 0xD0D1CCFF 0xD0D1C
0xD0D1CCFF 0xD0D1CCFF
0xD1D2CDFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D0CEFF 0xD0D0CEFF
0xD0D1CCFF 0xD0D1CCFF
0xD1D0CCFF 0xD2D1CDFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D0CEFF 0xD0D0CEFF
0xD1D0CCFF 0xD1D0CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD1D0CCFF 0xD1D0CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D0CEFF 0xD0D0CEFF
0xD0D1CCFF 0xD0D1CCFF
0xD0D1CCFF 0xD0D1CCFF
```

Look in: samples

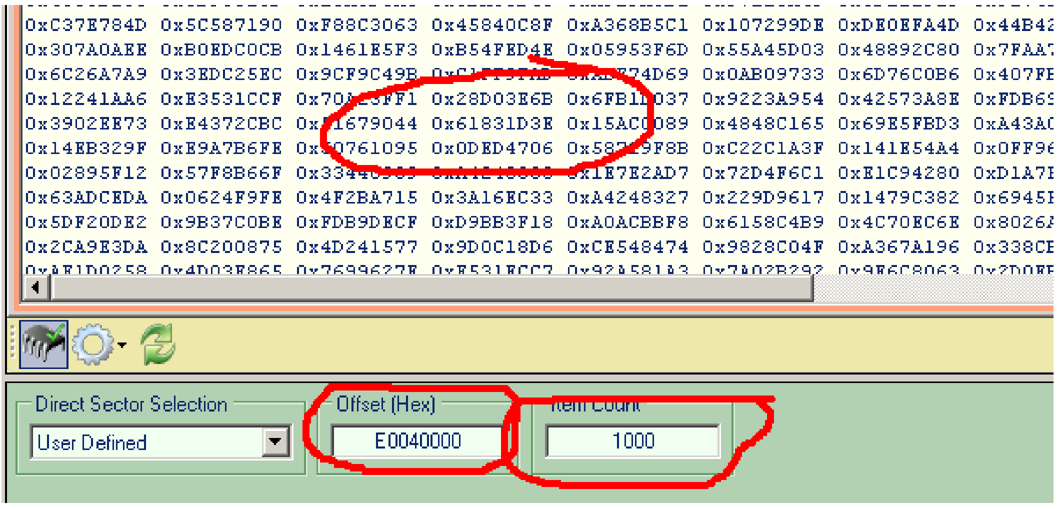
blue_1024_768_RGBA.gdc32dat
car_640_480.gdc32dat
car_800_600.gdc32dat
car_854_480.gdc32dat
car_1024_768.gdc32dat
car_1280_1024.gdc32dat
green_1024_768_RGBA.gdc32dat
red_1024_768_RGBA.gdc32dat
TestImage0_640_480_32_RGBA.gdc32dat
TestImage1_640_480_32_RGBA.gdc32dat
TestImage1_800_600_32_RGBA.gdc32dat
TestImage1_1024_768_32_RGBA.gdc32dat
TestImage1_1280_1024_32_RGBA.gdc32dat
TestImage2_640_480_32_RGBA.gdc32dat
TestImage2_800_600_32_RGBA.gdc32dat
TestImage2_1024_768_32_RGBA.gdc
TestImage2_1280_1024_32_RGBA.gd
TestImage3_640_480_32_RGBA.gdc3
TestImage3_800_600_32_RGBA.gdc3
TestImage3_1024_768_32_RGBA.gdc
TestImage3_1280_1024_32_RGBA.gd

File name: car_640_480.gdc32dat
Files of type: Binary Files (*.gdc32dat;*.bin;*.mhx)

Direct Sector Selection: User Defined
Offset (Hex): E0040000
Item Count: 307200

7. Compare with flash

Compare a number of items (entered in Item Count) from the data content in the hex dump view with flash memory starting at the entered flash address.

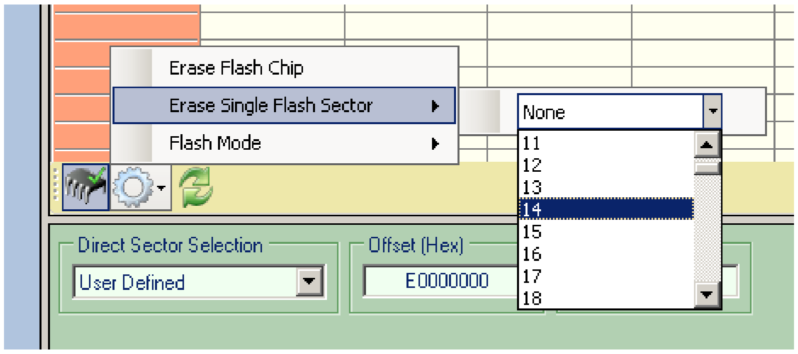


```
0xC37E784D 0x5C587190 0xF88C3063 0x45840C8F 0xA368B5C1 0x107299DE 0xDE0EFA4D 0x44B42
0x307A0AEE 0xB0EDC0CB 0x1461E5F3 0xB54FED4E 0x05953F6D 0x55A45D03 0x48892C80 0x7FAA7
0x6C26A7A9 0x3EDC25EC 0x9CF9C49B                       74D69 0x0AB09733 0x6D76C0B6 0x407FE
0x12241AA6 0xE3531CCF 0x70   FF1 0x28D03E6B 0x6FB1  037 0x9223A954 0x42573A8E 0xFDB69
0x3902EE73 0xE4372CBC 0x 1679044 0x61831D3E 0x15AC 089 0x4848C165 0x69E5FBD3 0xA43AC
0x14EB329F 0xE9A7B6FE 0x 0761095 0x0DED4706 0x58  9F8B 0xC22C1A3F 0x141E54A4 0x0FF96
0x02895F12 0x57F8B66F 0x33                   x1E7E2AD7 0x72D4F6C1 0xE1C94280 0xD1A7E
0x63ADCEDA 0x0624F9FE 0x4F2BA715 0x3A16EC33 0xA4248327 0x229D9617 0x1479C382 0x6945E
0x5DF20DE2 0x9B37C0BE 0xFDB9DECF 0xD9BB3F18 0xA0ACBBF8 0x6158C4B9 0x4C70EC6E 0x8026A
0x2CA9E3DA 0x8C200875 0x4D241577 0x9D0C18D6 0xCE548474 0x9828C04F 0xA367A196 0x338CE
```

Direct Sector Selection: User Defined
Offset (Hex): E0040000
Item Count: 1000

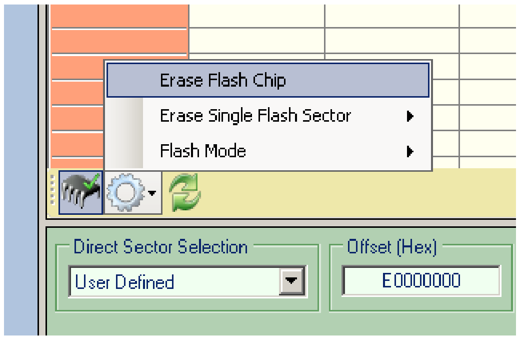
8. Erase flash sector

Erase a single flash sector.



9. Erase flash chip

Erase the complete flash chip.



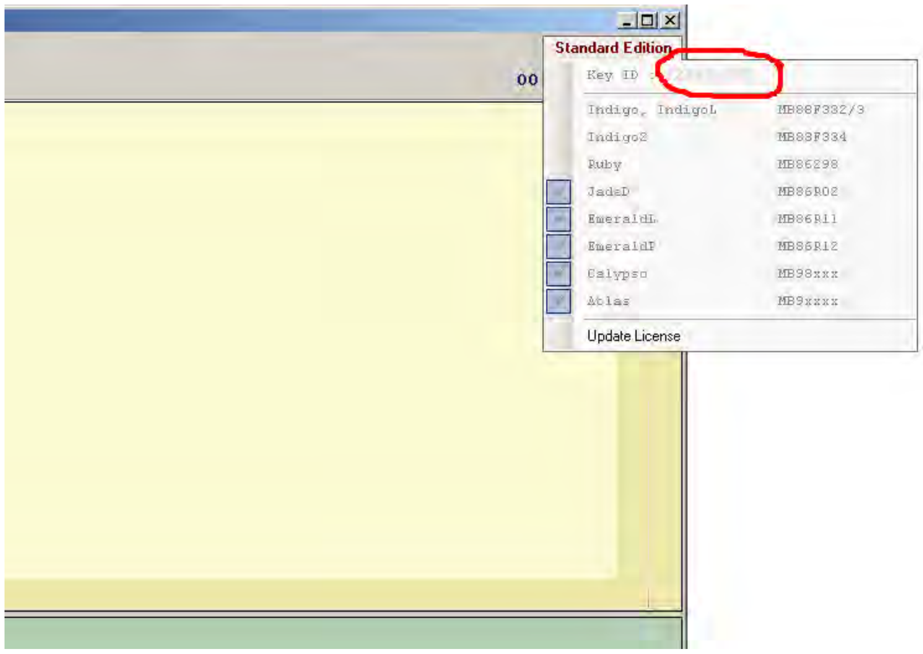
3. Dongle License Update / Upgrade

Remarks

- If the Dongle should be upgraded please follow the steps below.

1. Find out the ID of the Dongle to update

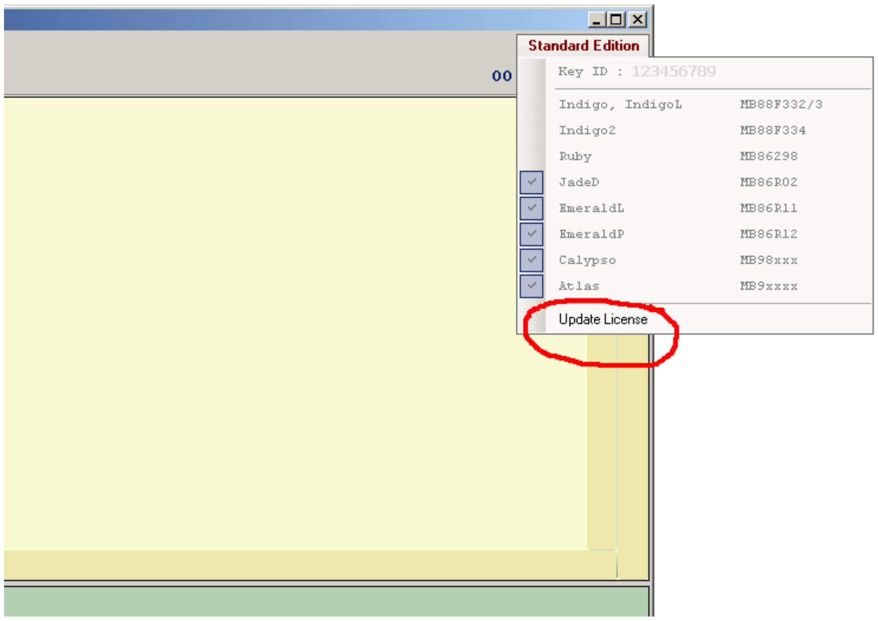
For this Start the Fujitsu Developer Suite and enter the menu item located on the upper right side in the menu bar.



Read out the complete number directly behind the Key ID item and send it directly to your Fujitsu contact person.

2. Update License

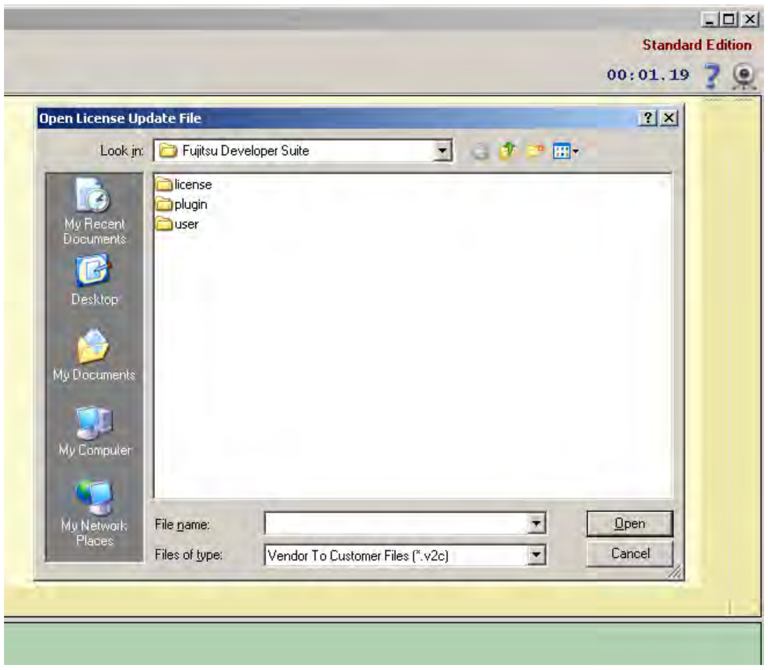
Fujitsu will then internally update your License Information and send an update file back to you. (.v2c)
Now start the Fujitsu Developer Suite again and press the Update License button, see below.



Key ID : 123456789

Indigo, IndigoL	MB88F332/3
Indigo2	MB88F334
Ruby	MB86298
✓ JadeD	MB86R02
✓ EmeraldL	MB86R11
✓ EmeraldP	MB86R12
✓ Calypso	MB98xxx
✓ Atlas	MB9xxxx

Select the received file (.v2c) and press Open.
The update starts immediately.



After the Dongle has been successfully updated close the Fujitsu Developer Suite and start it again.
Now you can check your new license information by reviewing the supported Chips (see Image on top of this article)

Special Modules ...

4. Emerald - Auto Update

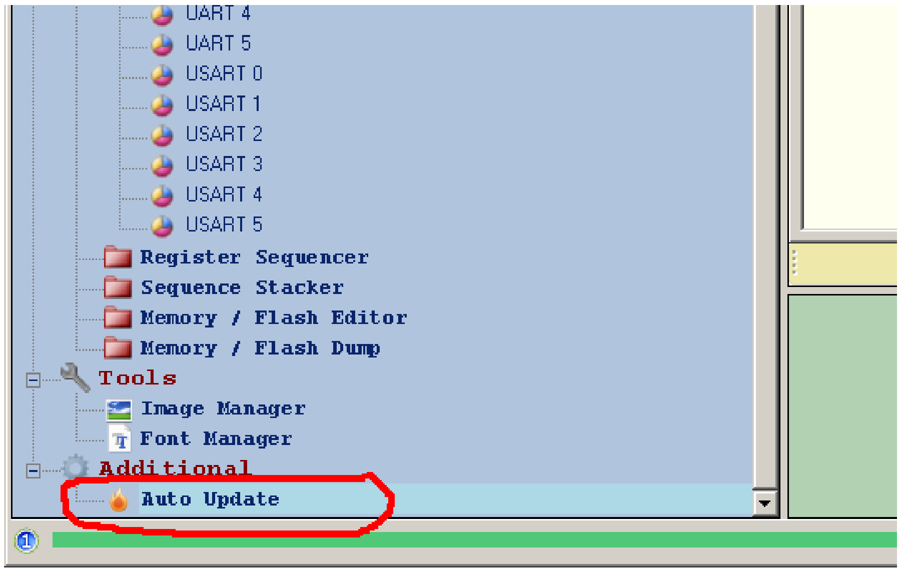
Updating Linux BSP on Emerald Systems

Remarks

- only available for SSH connection type
- working with
  1. Emerald L, ES2
  2. Emerald P, all versions

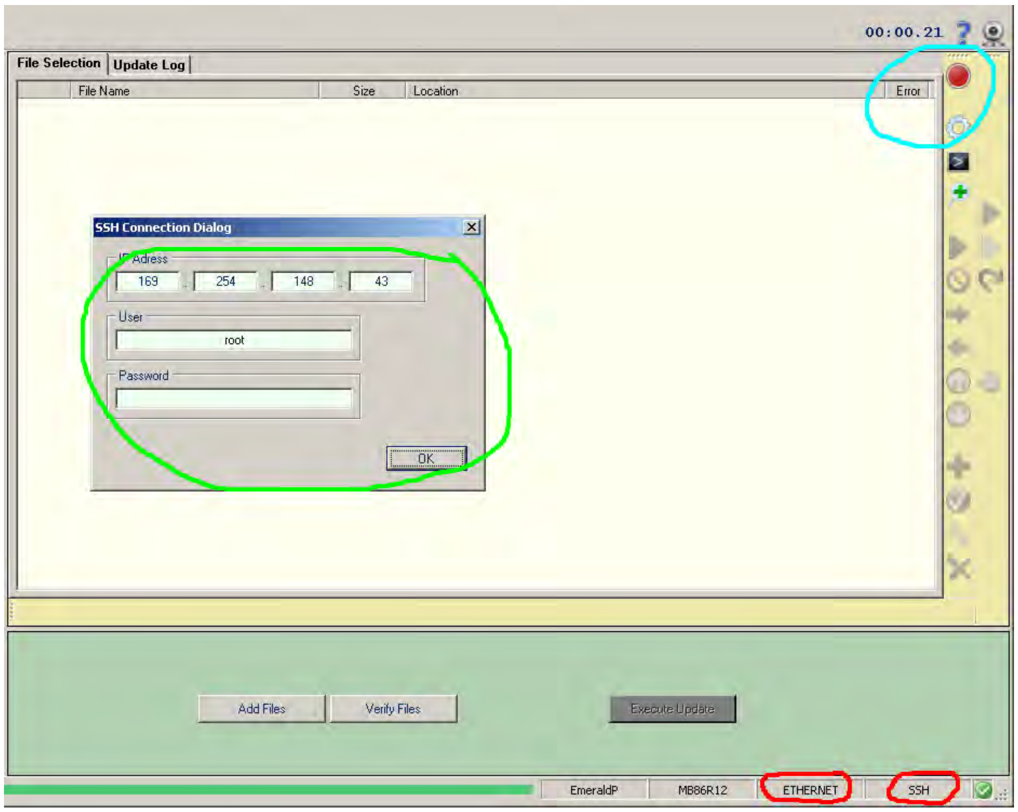
1. Select Auto Update Feature Page

Select the "Auto Update" page in the selection view.
Attention: It is only available for emerald targets with the SSH connection type.



2. Configure and establish SSH / Ethernet connection

Setup the SSH / Ethernet connection IP Address, User Name as well as the Password which is identical to your Linux system.



SSH Connection Dialog

IP Adress: 169 . 254 . 148 . 43

User: root

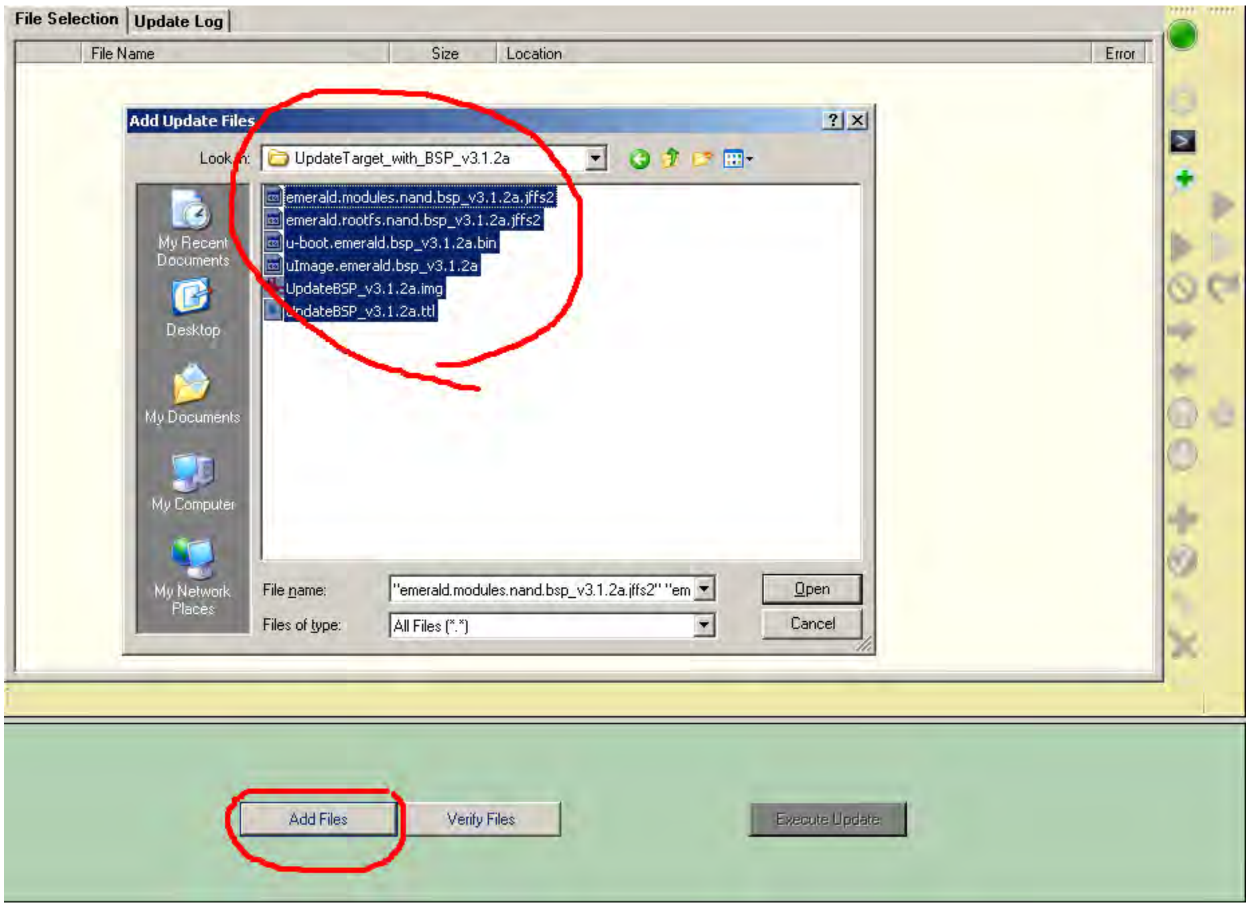
Password:

OK

When the connection information are entered correctly it is possible to establish a SSH connection to the target system by pressing the Hardware Connect / Disconnect button as usual.

3. Add update files

Press the "Add Files" button or drag and drop the required Linux BSP files into the "File Selection" view of the Auto Update page.



emerald.modules.nand.bsp_v3.1.2a.jffs2
emerald.rootfs.nand.bsp_v3.1.2a.jffs2
u-boot.emerald.bsp_v3.1.2a.bin
uImage.emerald.bsp_v3.1.2a
UpdateBSP_v3.1.2a.img
UpdateBSP_v3.1.2a.ttl

File name: "emerald.modules.nand.bsp_v3.1.2a.jffs2" "em

Files of type: All Files (*.*)

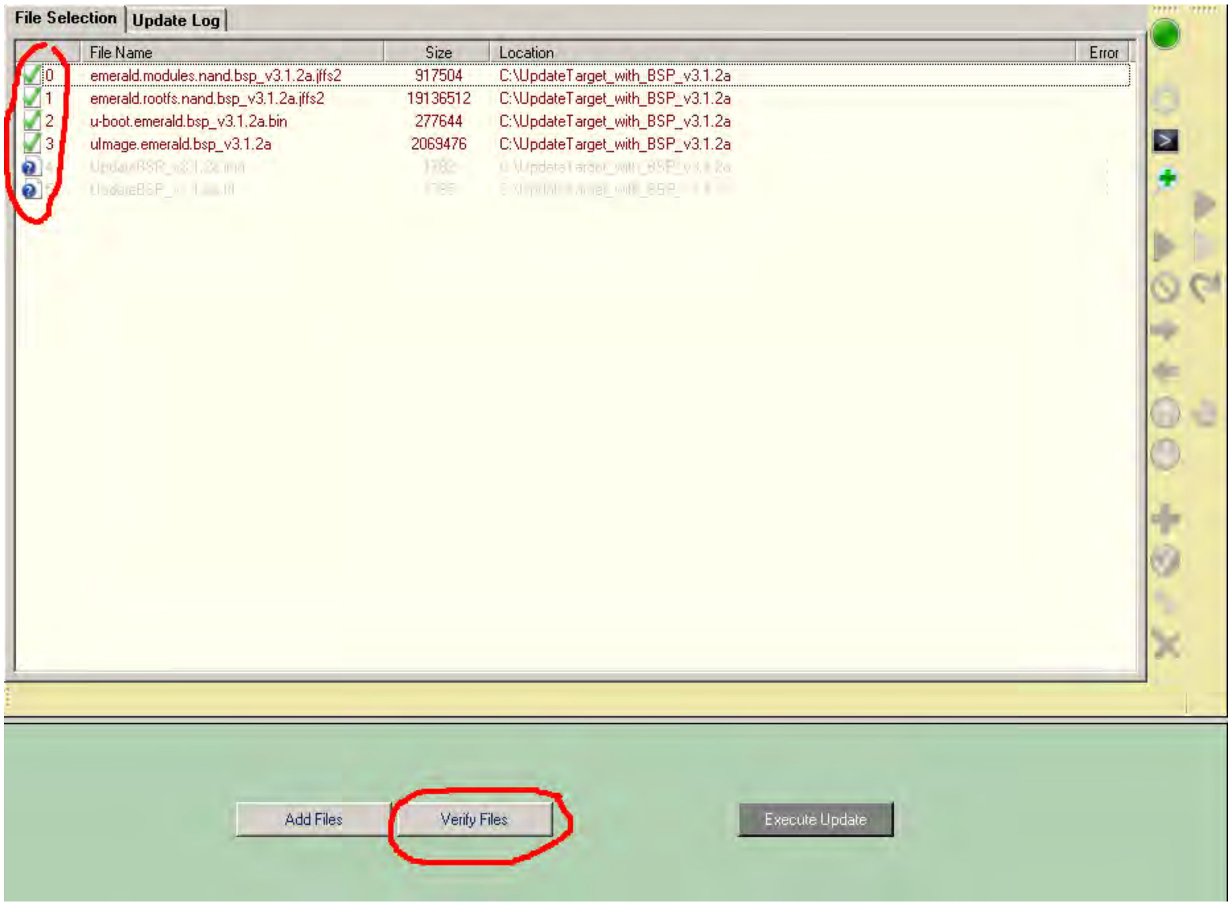
4. Verify files

When files are added it is required to start file verification - otherwise updating the system cannot be executed.

While verification it will be checked for the following attributes to ensure update safety :

1. file types (e.g. uboot, rootsfs, etc.)
2. identical BSP versions
3. availability and accessability
4. size limitations
5. redundant files

etc.



	File Name	Size	Location	Error
✓ 0	emerald.modules.nand.bsp_v3.1.2a.jffs2	917504	C:\UpdateTarget_with_BSP_v3.1.2a	
✓ 1	emerald.rootfs.nand.bsp_v3.1.2a.jffs2	19136512	C:\UpdateTarget_with_BSP_v3.1.2a	
✓ 2	u-boot.emerald.bsp_v3.1.2a.bin	277644	C:\UpdateTarget_with_BSP_v3.1.2a	
✓ 3	uImage.emerald.bsp_v3.1.2a	2069476	C:\UpdateTarget_with_BSP_v3.1.2a	
? 4	UpdateBSP_v3.1.2a.ini	1782	C:\UpdateTarget_with_BSP_v3.1.2a	
? 5	[illegible]	1785	[illegible]	

Redundant and all invalid files will be marked / disabled in the view and ignored while processing.

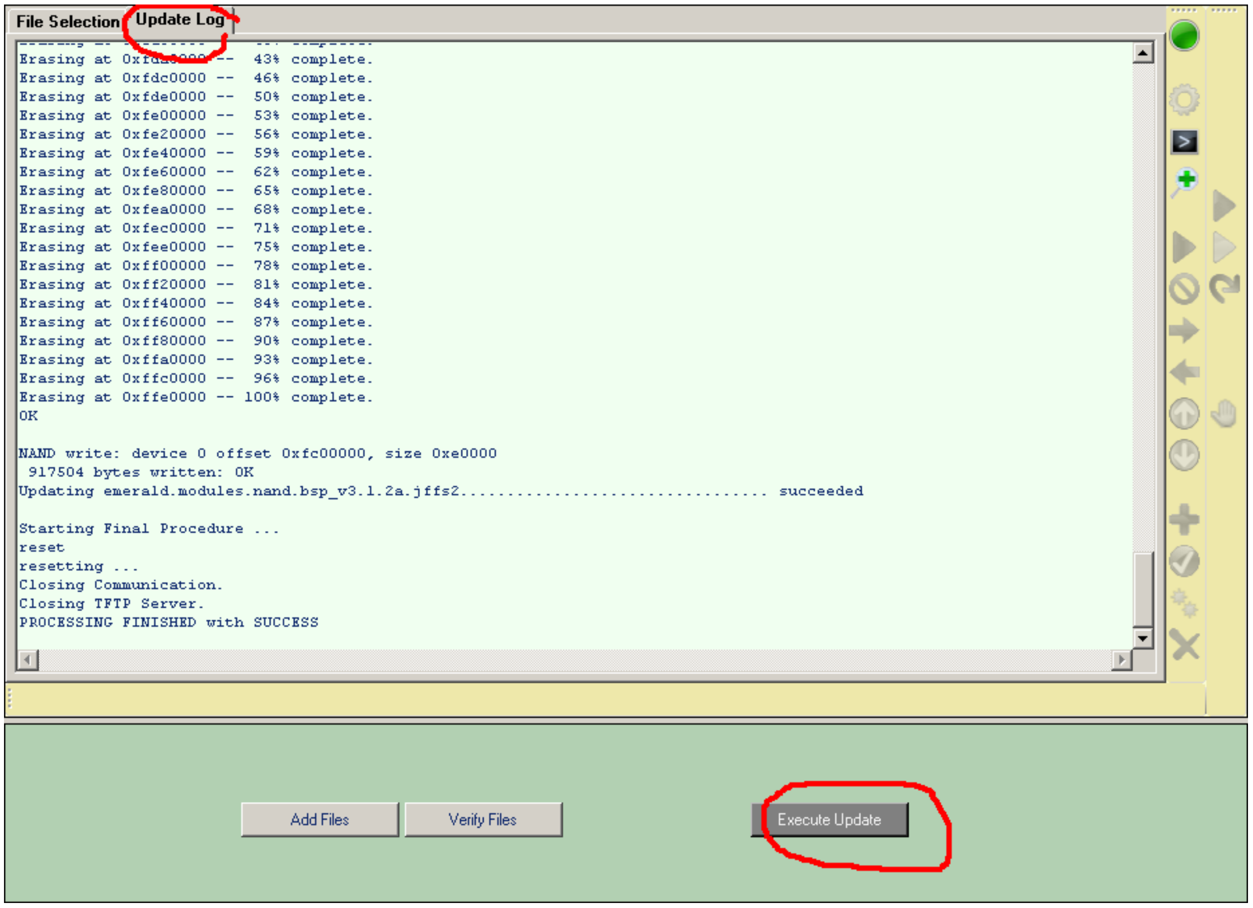
Only if valid files are detected - the update process can be started.

5. Execute update

This is only possible when valid files are detected / verified in the "File Selection" Tab of the "Auto Update".

By pressing the "Execute Update" button on the lower right, the "Update Log" page will be automatically selected and processing the update is started.

In the log box detailed information about the current process are available.



```
Erasing at 0xfda0000 --  43% complete.
Erasing at 0xfdc0000 --  46% complete.
Erasing at 0xfde0000 --  50% complete.
Erasing at 0xfe00000 --  53% complete.
Erasing at 0xfe20000 --  56% complete.
Erasing at 0xfe40000 --  59% complete.
Erasing at 0xfe60000 --  62% complete.
Erasing at 0xfe80000 --  65% complete.
Erasing at 0xfea0000 --  68% complete.
Erasing at 0xfec0000 --  71% complete.
Erasing at 0xfee0000 --  75% complete.
Erasing at 0xff00000 --  78% complete.
Erasing at 0xff20000 --  81% complete.
Erasing at 0xff40000 --  84% complete.
Erasing at 0xff60000 --  87% complete.
Erasing at 0xff80000 --  90% complete.
Erasing at 0xffa0000 --  93% complete.
Erasing at 0xffc0000 --  96% complete.
Erasing at 0xffe0000 -- 100% complete.
OK

NAND write: device 0 offset 0xfc00000, size 0xe0000
 917504 bytes written: OK
Updating emerald.modules.nand.bsp_v3.1.2a.jffs2................................ succeeded

Starting Final Procedure ...
reset
resetting ...
Closing Communication.
Closing TFTP Server.
PROCESSING FINISHED with SUCCESS
```

At the end it will be reported if the update has been succeeded or failed.

Information: The target system will be rebooted automatically.

5. Emerald - Display / Panel Manager

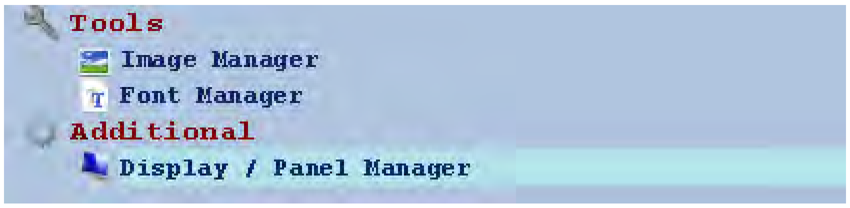
Main intention is to setup, configure and test new panels / display resolutions on the target chip.
After successful validation with test patterns timing register values as well as layer settings can be stored as .par file to be able to reimport settings with the Register Sequencer.
Test Images can also be stored as source files, images and binary files for further usage in programs or for reusing it in the Developer Suite (Image Manager, Hex Dump pages).

Remarks

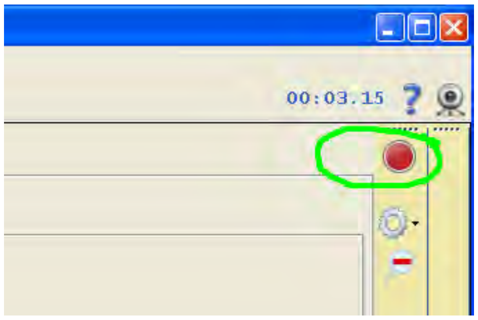
- available for JTAG and SPI connection type
- working with
  1. Emerald L, ES2
  2. Emerald P, all versions

1. Select Display / Panel Manager Feature Page

Select the "Display / Panel Manager" page in the selection view.



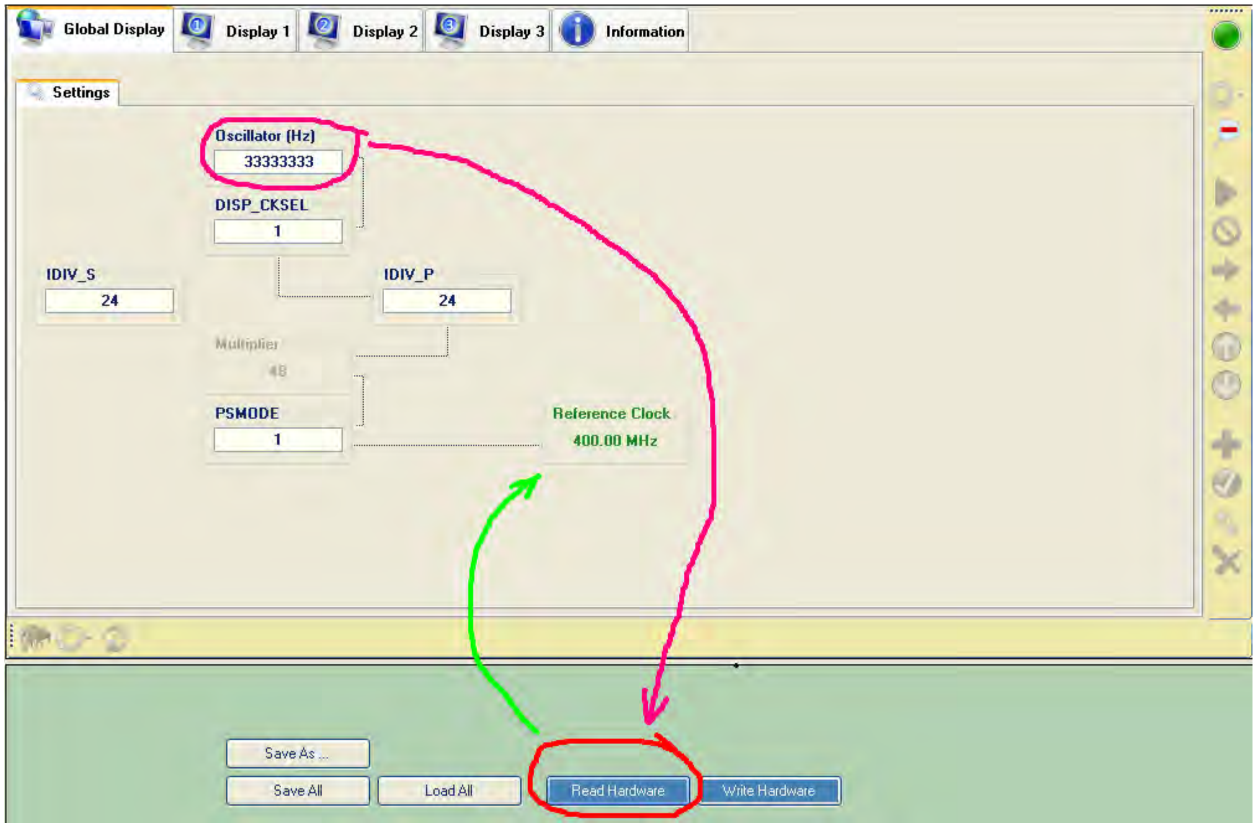
2. Connect to Target Device



3. Collect Device Information

The fist step is to ensure that the oscillation frequency is set to the correct value. This is the only thing that cannot be extracted from hardware.

After checking the oscillator read out current global hardware settings.
By doing this registers from different modules will be read out and the corresponding Reference Clock will be calculated from this values.
The Reference Clock is the base for the Display Clock which is required for further calculations.



Oscillator (Hz): 33333333
DISP_CKSEL: 1
IDIV_S: 24
IDIV_P: 24
Multiplier: 48
PSMODE: 1
Reference Clock: 400.00 MHz

Remarks :
On this page it is possible to store all timing settings from all pages into either a .gdcfunc file (which is also loadable on this page) or into a .par file which can be used in the Register Sequencer.

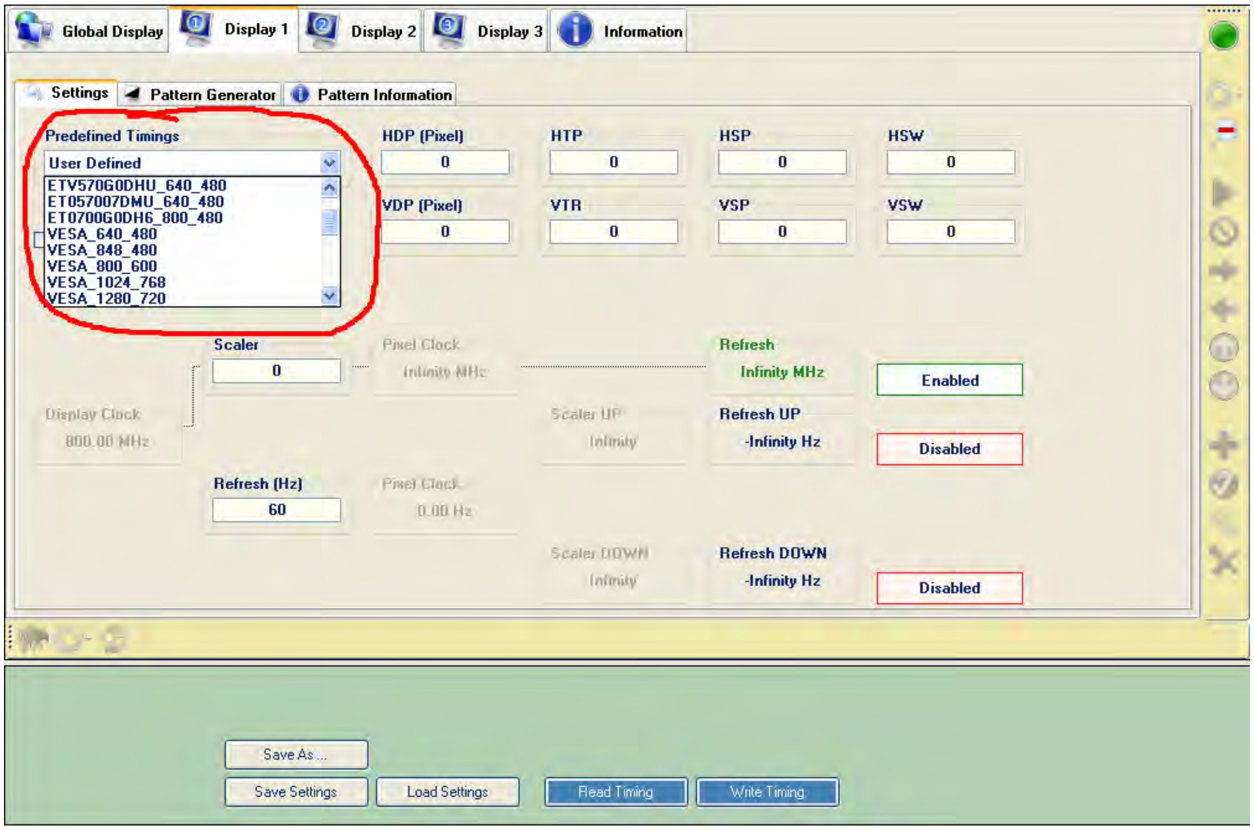
3. Display Timing selection and setup

a.

Select the Tab page that represents the display output that should be used.

On the Settings sub-tab page some predefined timings can be found containing panels as well as a few VESA standard timings.

By selecting one of the predefined settings the corresponding timing values will be automatically entered into the text boxes.



Predefined Timings: User Defined

ETV570G0DHU_640_480
ET057007DMU_640_480
ET0700G0DH6_800_480
VESA_640_480
VESA_848_480
VESA_800_600
VESA_1024_768
VESA_1280_720

HDP (Pixel)	HTP	HSP	HSW
0	0	0	0
VDP (Pixel)	VTR	VSP	VSW
0	0	0	0

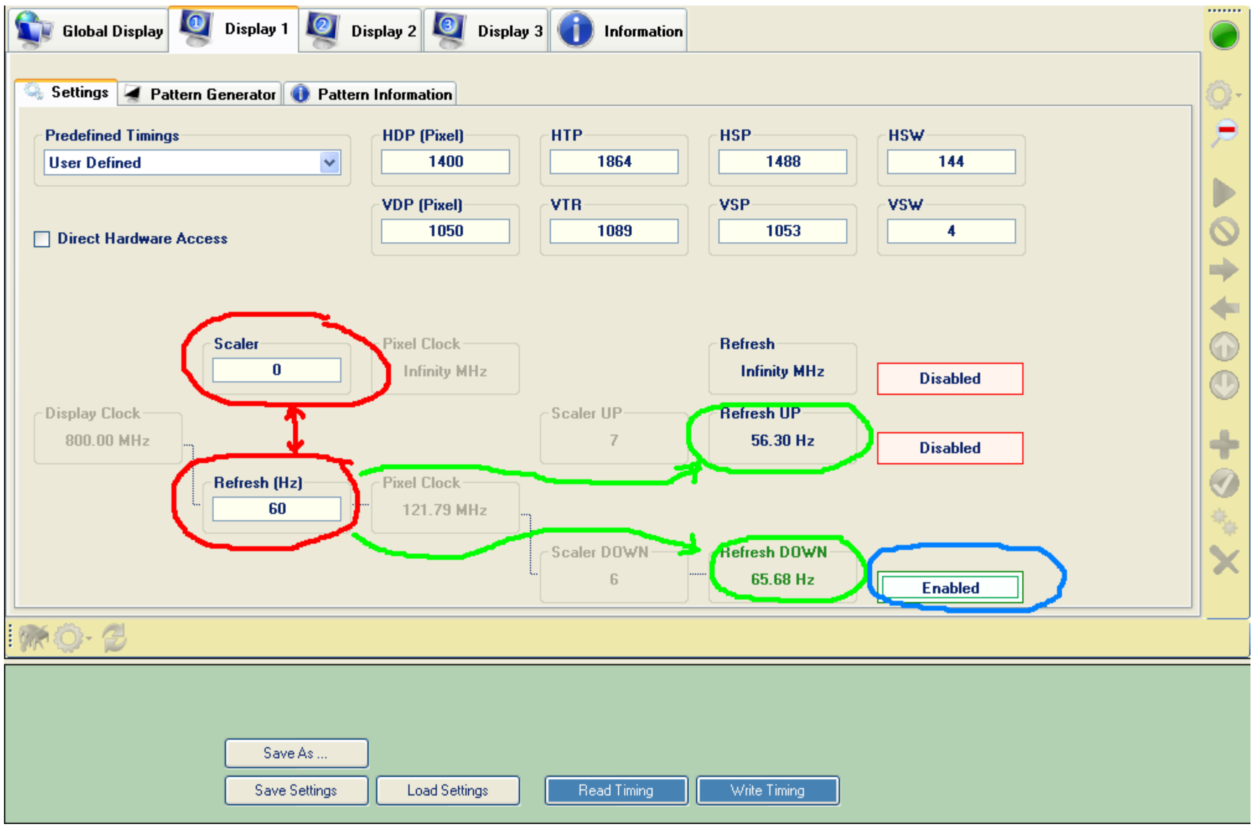
Scaler: 0
Pixel Clock: Infinity MHz
Refresh: Infinity MHz — Enabled
Display Clock: 800.00 MHz
Scaler UP: Infinity
Refresh UP: -Infinity Hz — Disabled
Refresh (Hz): 60
Pixel Clock: 0.00 Hz
Scaler DOWN: Infinity
Refresh DOWN: -Infinity Hz — Disabled

Afterwards change the timing settings so that they best matches the used panel.
For timing details please refer to the user manual of the connected panel / display.

b.

There are 2 possibilities to continue with the timing setup.
Either set Scaler value (not recommended) or enter the required Refresh Rate directly (which is the easiest way to setup the target timing).

Typically it is not possible to match the required Refresh Rate exactly.
Because of this two possible Scaler values will be calculated automatically that resulting in two Refresh Rates that are closest to the required Refresh Rate.
One calculated value is higher, the other value is smaller than the given value.
Now the user can "Enable" the Refresh Path that is closest to the required one or that can be manipulated in a way that is compatible with the target panel.



Predefined Timings: User Defined

HDP (Pixel)	HTP	HSP	HSW
1400	1864	1488	144

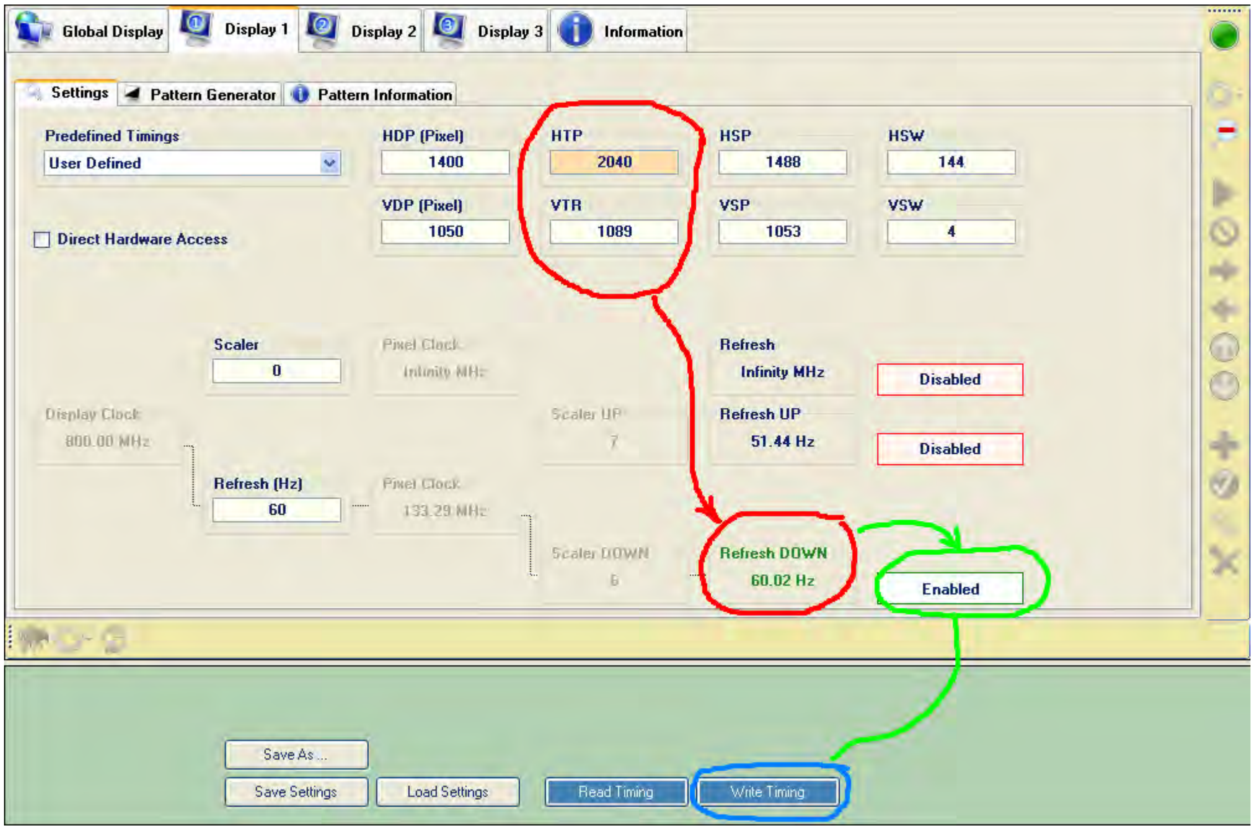
VDP (Pixel)	VTR	VSP	VSW
1050	1089	1053	4

Direct Hardware Access

Scaler: 0
Pixel Clock: Infinity MHz
Refresh: Infinity MHz — Disabled
Display Clock: 800.00 MHz
Scaler UP: 7
Refresh UP: 56.30 Hz — Disabled
Refresh (Hz): 60
Pixel Clock: 121.79 MHz
Scaler DOWN: 6
Refresh DOWN: 65.68 Hz — Enabled

c.

To get close to the required Refresh Rate change Horizontal and Vertical total pixels / raster timing values.
Please check the user manual of the connected panel for allowed timing and Refresh Rate ranges.



Predefined Timings: User Defined

☐ Direct Hardware Access

HDP (Pixel)	HTP	HSP	HSW
1400	2040	1488	144
VDP (Pixel)	**VTR**	**VSP**	**VSW**
1050	1089	1053	4

Scaler: 0
Pixel Clock: Infinity MHz
Refresh: Infinity MHz — Disabled
Display Clock: 800.00 MHz
Scaler UP: 7
Refresh UP: 51.44 Hz — Disabled
Refresh (Hz): 60
Pixel Clock: 133.29 MHz
Scaler DOWN: 6
Refresh DOWN: 60.02 Hz — Enabled

When all settings are done check that the path is still "Enabled" and write timing to the target hardware.

Remarks :

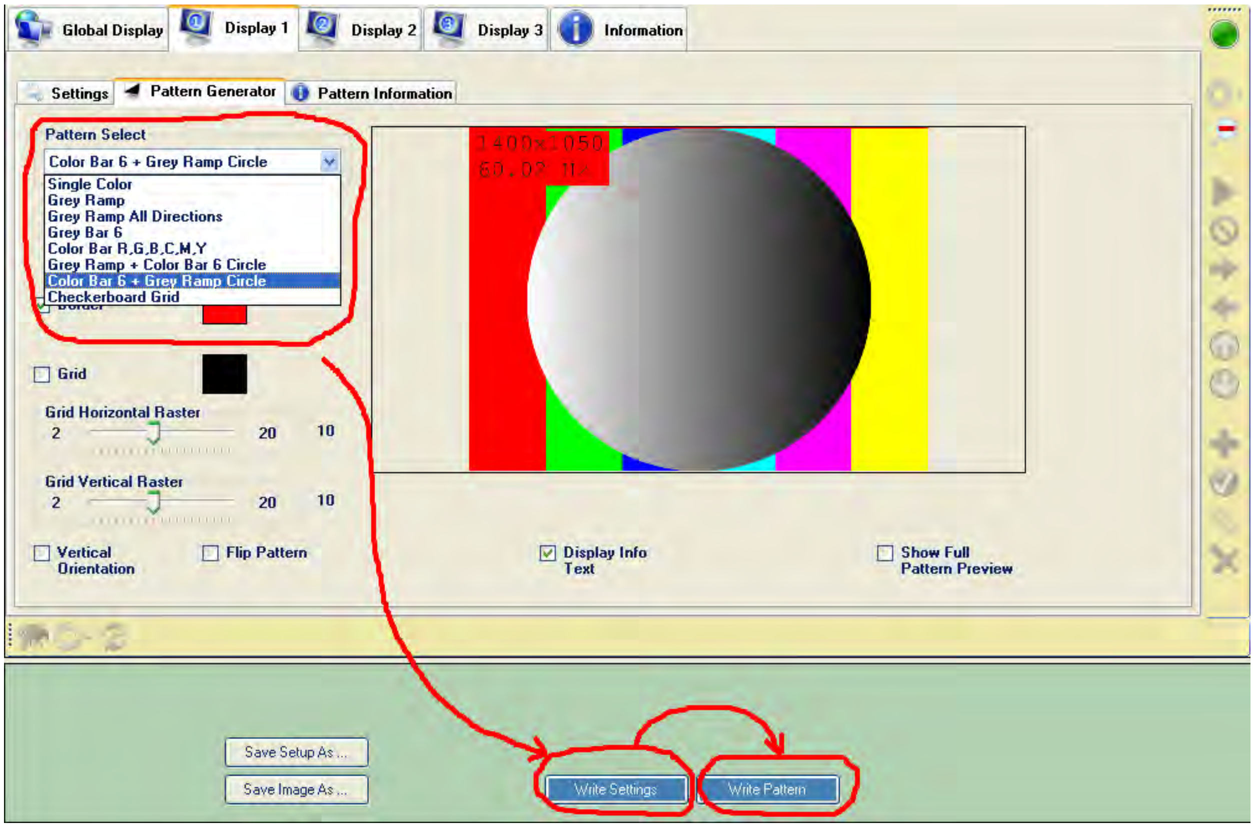
On this page it is also possible to store the timing settings for the selected display port into either a .gdcfunc file (which is also loadable on this page) or into a .par file which can be used in the Register Sequencer.

If a valid timing setup is already located in the hardware (e.g. already setup by Register Sequencer or by Software) the corresponding timing / register values can be read out from hardware by pressing the "Read Timing" button.

4. Pattern Generator

For test the timing changes made under 3) a Pattern Generator is available.

There is a set of predefined patterns that will be calculated and drawn corresponding to the Active Area of the Settings page.
That means when you change resolution, refresh rate etc. the pattern will be generated automatically.
Furthermore some pattern features can be set that supports timing tests like a border, grid, orientation, display timing info etc.



After pattern was setup do the following,

a. Write Settings

By pressing this button the corresponding layers of the display output will be setup automatically to prepare a framebuffer like environment.

b. Write Pattern

Here the pattern displayed in the preview box will be copied to the target memory / framebuffer.

Attention :

It is required that the display memory was initialized before with a proper setup.
The display memory can differ depending on the customer.
For the Fujitsu validation boards there is a sequence available that can be executed before in the Register Sequencer.

c. Adapt Timing

Finally step back to the settings page and correct the values as required.

It is now possible to enable the "Direct Hardware Access" check box that enables the possibility that each change in the display timing will directly be written into the corresponding register.
This allows adjusting the display live on the target.

Remarks :

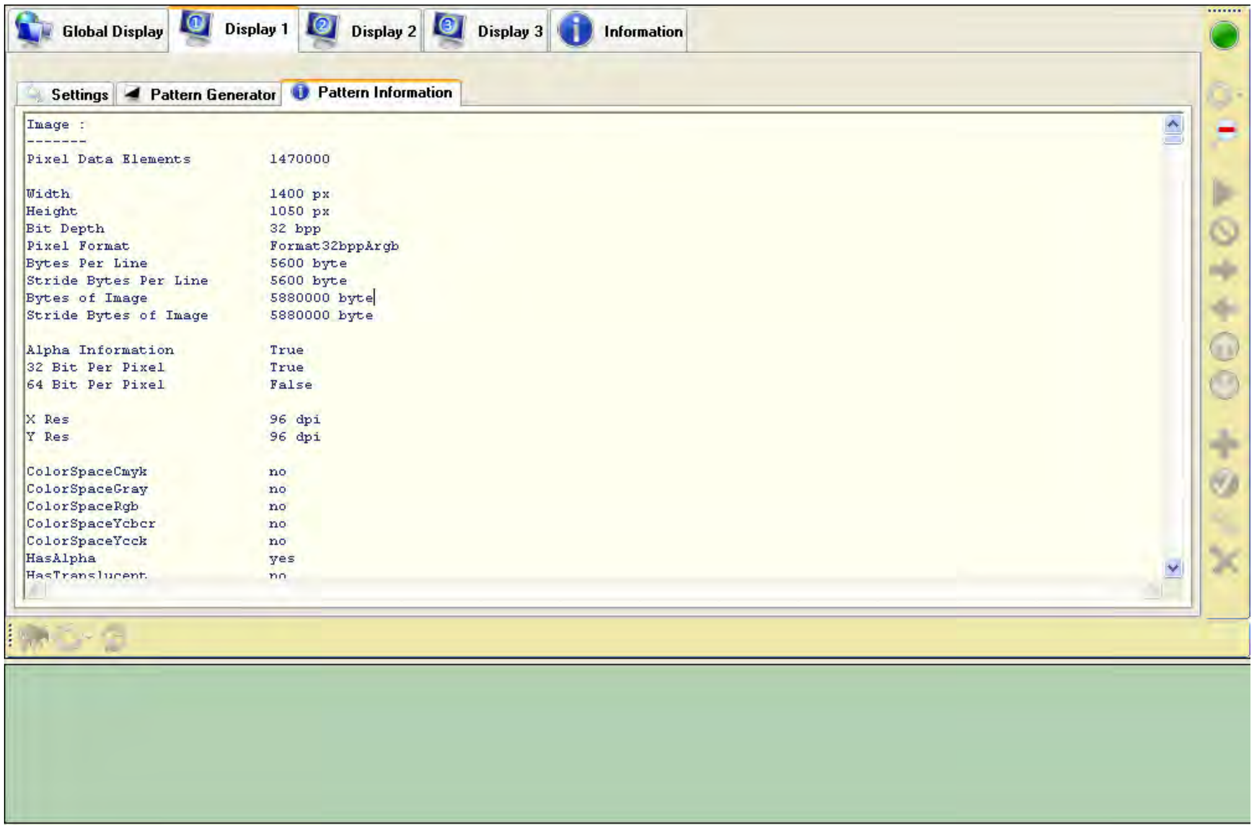
On this page it is also possible to store the generated pattern image either as an image file (for further usage in the Image Manager), as a binary file (for later usage in the Memory / Flash Dump page) or as source code file for usage in a target application.

Furthermore the Layer Settings can be stored as .par file for further usage in the Register Sequencer..

5. Other pages

a. Pattern Information Page

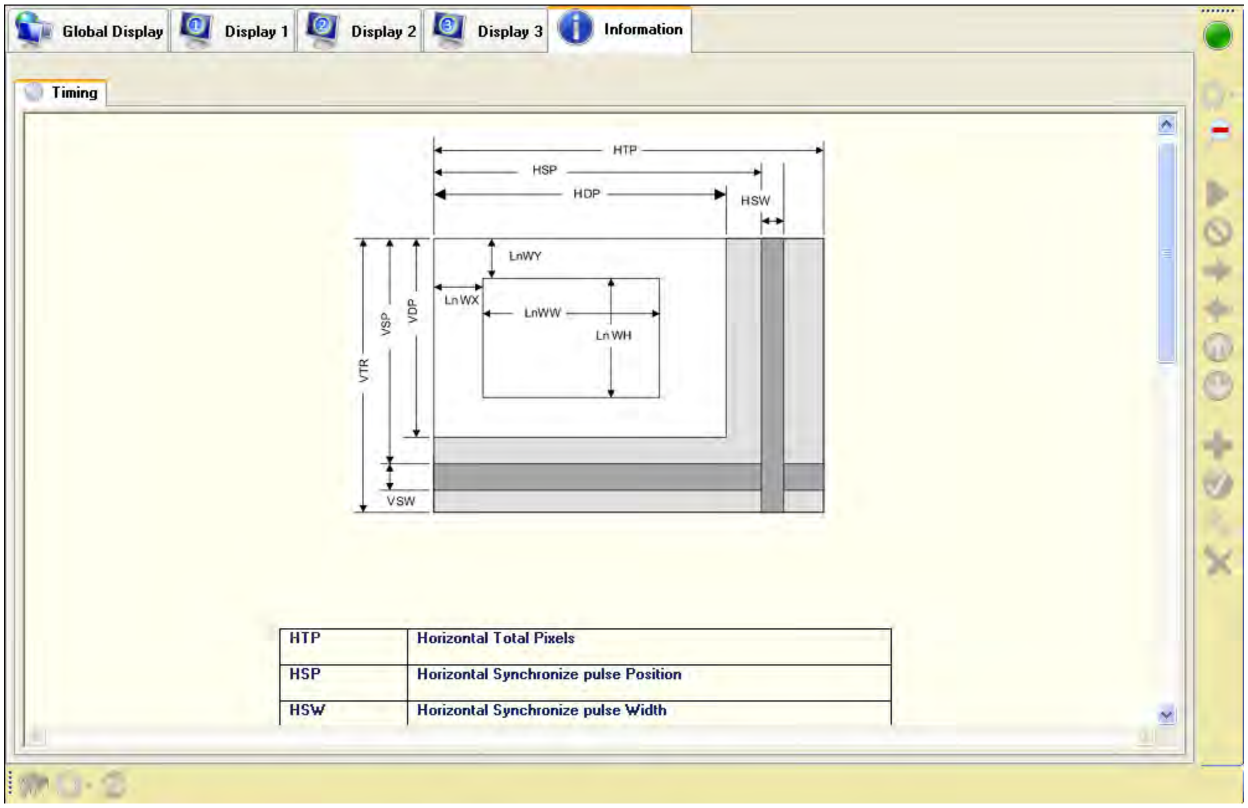
This page contains detailed information about the generated Pattern.
Furthermore it contains the complete Pixel Data information - ready to be copied directly into source code.



```
Image :
-------
Pixel Data Elements          1470000

Width                        1400 px
Height                       1050 px
Bit Depth                    32 bpp
Pixel Format                 Format32bppArgb
Bytes Per Line               5600 byte
Stride Bytes Per Line        5600 byte
Bytes of Image               5880000 byte
Stride Bytes of Image        5880000 byte

Alpha Information            True
32 Bit Per Pixel             True
64 Bit Per Pixel             False

X Res                        96 dpi
Y Res                        96 dpi

ColorSpaceCmyk               no
ColorSpaceGray               no
ColorSpaceRgb                no
ColorSpaceYcbcr              no
ColorSpaceYcck               no
HasAlpha                     yes
HasTranslucent               no
```

b. Global Information Page

This page contains some information to display / panel timings as well as supporting the user to read out the correct timing values from the panel specification.



HTP	Horizontal Total Pixels
HSP	Horizontal Synchronize pulse Position
HSW	Horizontal Synchronize pulse Width

6. Indigo2 - Display / Panel Manager

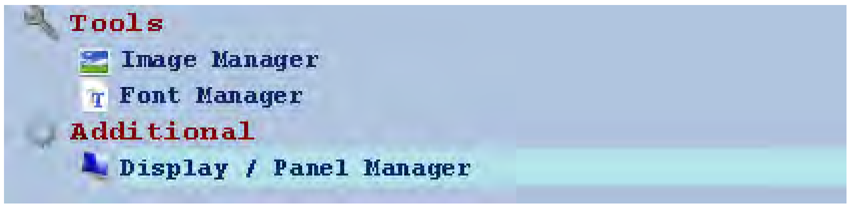
Main intention is to setup, configure and test new panels / display resolutions on the target chip.
After successful validation with test patterns timing register values, clock settings and iris settings can be stored in .par files to be able to reimport settings with the Register Sequencer.
Because of less SRAM memory the simulated test patterns are a combination with iris framegenerator specific actions, iris fetch and iris sprite engine setup etc.
Nevertheless the test images can be stored as source files, images and binary files for further usage in programs or when more memory is available externally.

Remarks

- available for SPI (Aardvark, FTDI) and Ethernet (E2IP) connection type
- working with Indigo2

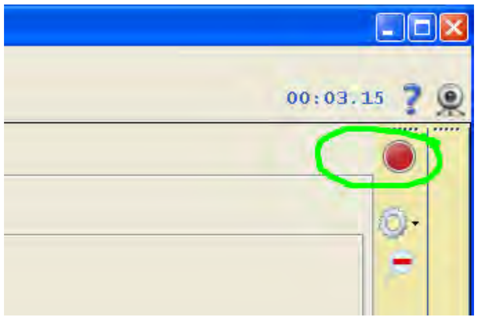
1. Select Display / Panel Manager Feature Page

Select the "Display / Panel Manager" page in the selection view.



2. Connect to Target Device

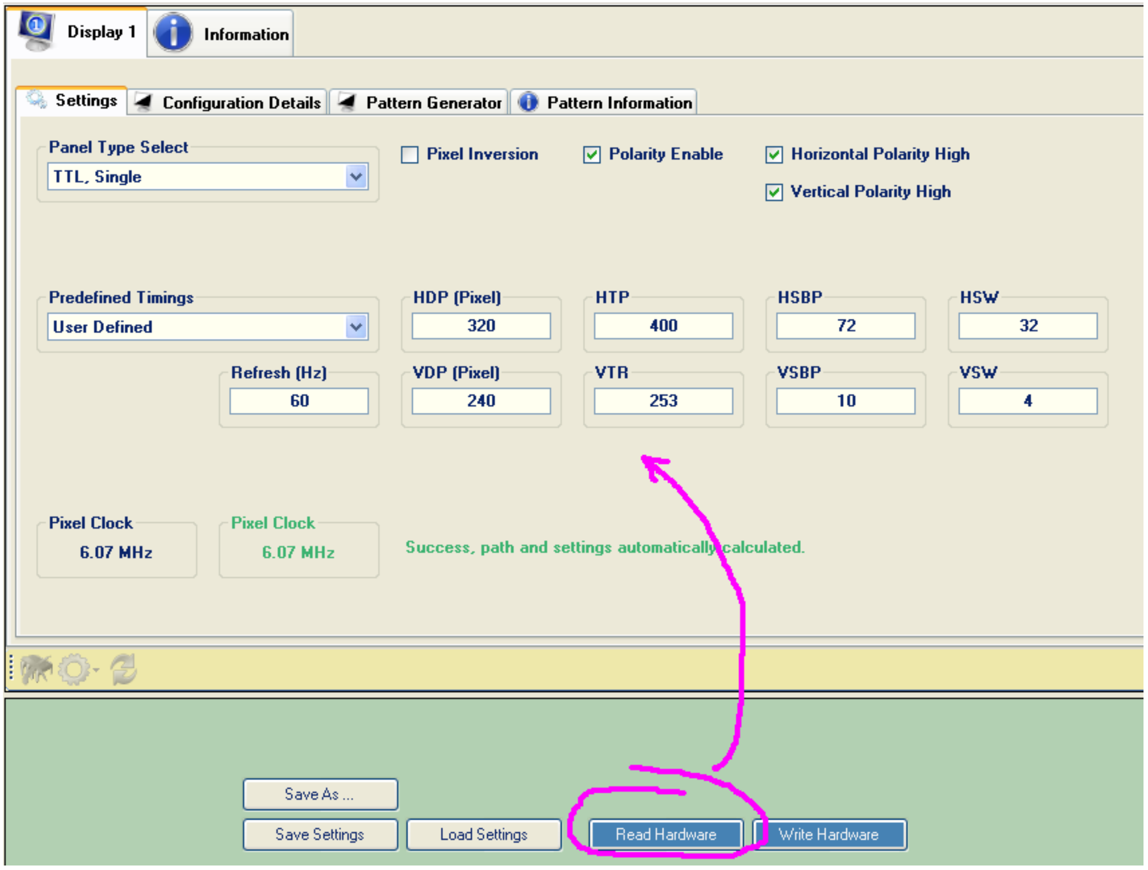
Depending on the connection type the connection devices must be scanned or configured before a connection can be established with the button below.



3. Collect Device Information

If it is required for user information purpose the current settings located in the Hardware can be read out by pressing the Read Hardware Button.

For setting up a new panel it is not required to read out information before.



Panel Type Select: TTL, Single
Pixel Inversion [ ]
Polarity Enable [✓]
Horizontal Polarity High [✓]
Vertical Polarity High [✓]

Predefined Timings: User Defined

	HDP (Pixel)	HTP	HSBP	HSW
	320	400	72	32
Refresh (Hz)	VDP (Pixel)	VTR	VSBP	VSW
60	240	253	10	4

Pixel Clock: 6.07 MHz
Pixel Clock: 6.07 MHz

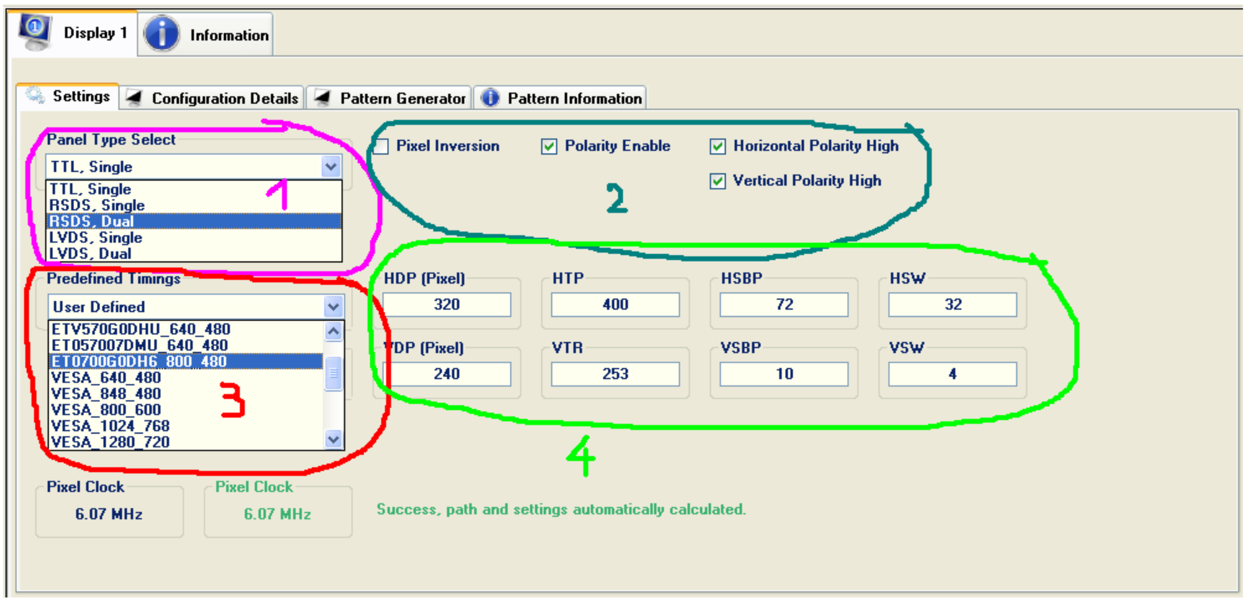
Success, path and settings automatically calculated.

Remarks :

After reading the settings the upper fields will be updated to the current values.

4. Setup Panel Information and Timing

1. Choose the required Panel Type
2. Setup Panel Polarity and if Pixel Inversion is required
3. Select one of the Predefined Timings or one which is close to the required Panel Timing
4. Fine tune the Panel Timing corresponding to the Display Specification



Panel Type Select: TTL, Single

TTL, Single
RSDS, Single
RSDS, Dual
LVDS, Single
LVDS, Dual

Pixel Inversion [ ] Polarity Enable [✓] Horizontal Polarity High [✓] Vertical Polarity High [✓]

Predefined Timings: User Defined

ETV570G0DHU_640_480
ET057007DMU_640_480
ET0700G0DH6_800_480
VESA_640_480
VESA_848_480
VESA_800_600
VESA_1024_768
VESA_1280_720

HDP (Pixel)	HTP	HSBP	HSW
320	400	72	32
VDP (Pixel)	**VTR**	**VSBP**	**VSW**
240	253	10	4

Pixel Clock: 6.07 MHz
Pixel Clock: 6.07 MHz

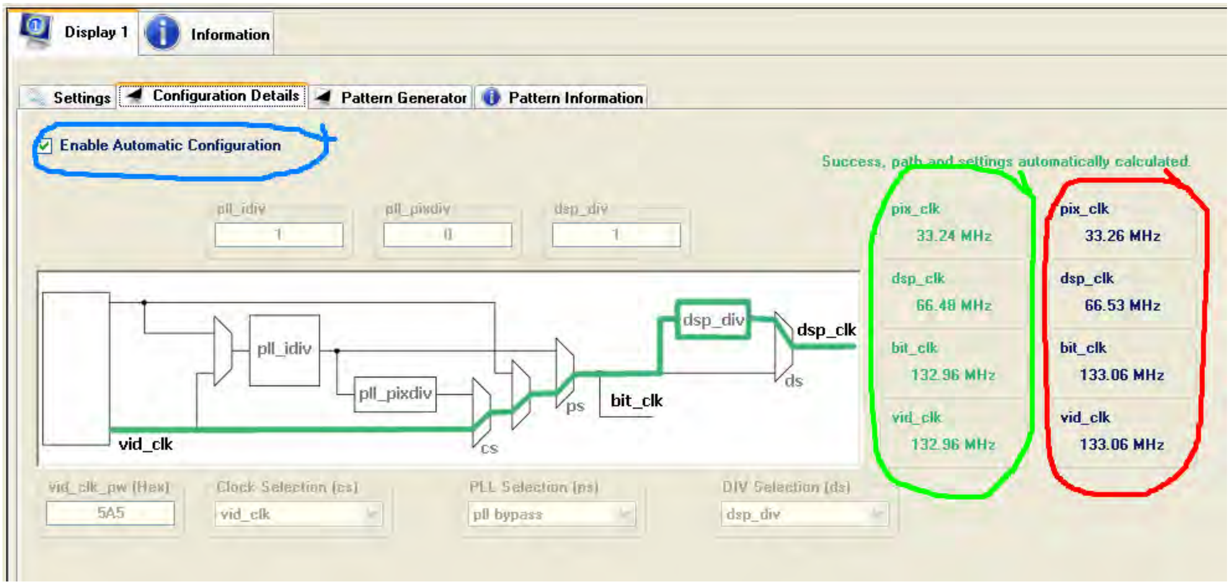
Success, path and settings automatically calculated.

After configuration the required Pixel Clock as well as the real Pixel Clock that can be adjusted / is adjusted in the Clock Path will be displayed.
Furthermore a short message will appear with success or failed information.

5. Clock Path Setup

a. Automatic Mode

It is recommended to use the automatic setup by enabling the "Enable Automatic Configuration" check box.
When it is enabled the best path trough the Clock Path as well as the best settings for the PLL and the different Dividers will be calculated automatically.
This will be done in a way to get the best and closest timing to the calculated reference values.



☑ Enable Automatic Configuration

Success, path and settings automatically calculated.

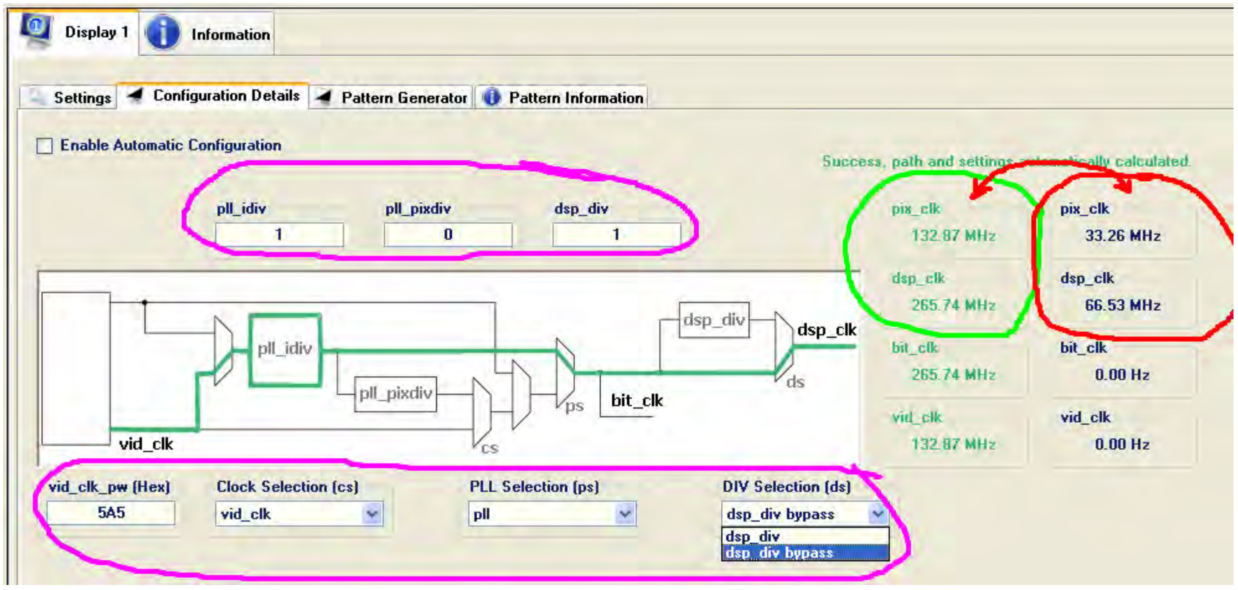
pll_idiv	pll_pixdiv	dsp_div
1	0	1

pix_clk 33.24 MHz	pix_clk 33.26 MHz
dsp_clk 66.48 MHz	dsp_clk 66.53 MHz
bit_clk 132.96 MHz	bit_clk 133.06 MHz
vid_clk 132.96 MHz	vid_clk 133.06 MHz

vid_clk_pw (Hex)	Clock Selection (cs)	PLL Selection (ps)	DIV Selection (ds)
5A5	vid_clk	pll bypass	dsp_div

b. Manual mode

When the automatic clock setting is disabled the user can choose its favorite path by its own.
After changing the Path, the PLL or the Dividers the **real adjusted clock** will be automatically calculated and displayed - corresponding to its path.
Right beside the current clock setting the **required clock values** are found.
Now choose your path and set the dividers until the real adjusted clocks are close to the required clocks.
But please be aware to check the different requirements and limitations.



☐ Enable Automatic Configuration

Success, path and settings automatically calculated.

pll_idiv	pll_pixdiv	dsp_div
1	0	1

pix_clk 132.87 MHz	pix_clk 33.26 MHz
dsp_clk 265.74 MHz	dsp_clk 66.53 MHz
bit_clk 265.74 MHz	bit_clk 0.00 Hz
vid_clk 132.87 MHz	vid_clk 0.00 Hz

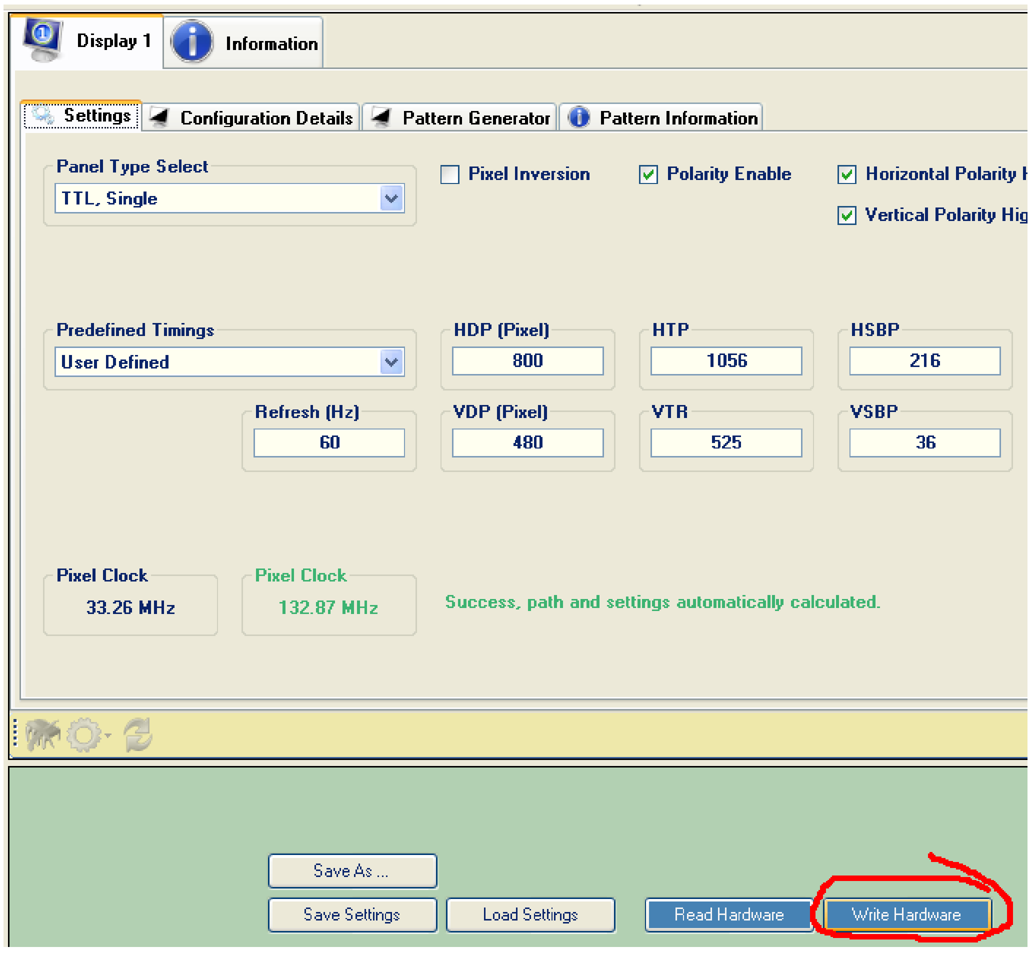
vid_clk_pw (Hex)	Clock Selection (cs)	PLL Selection (ps)	DIV Selection (ds)
5A5	vid_clk	pll	dsp_div bypass

dsp_div
dsp_div bypass

6. Write Hardware

When all timing settings are made and the real pixel clock is as required all values can be written to the Hardware by pressing the "Write Hardware" button.

After that the display/panel is setup and configured.
If a test image is required please continue on step 6.



Panel Type Select: TTL, Single
Pixel Inversion: ☐
Polarity Enable: ✓
Horizontal Polarity H: ✓
Vertical Polarity Hig: ✓

Predefined Timings: User Defined
Refresh (Hz): 60

HDP (Pixel)	HTP	HSBP
800	1056	216

VDP (Pixel)	VTR	VSBP
480	525	36

Pixel Clock: 33.26 MHz
Pixel Clock: 132.87 MHz

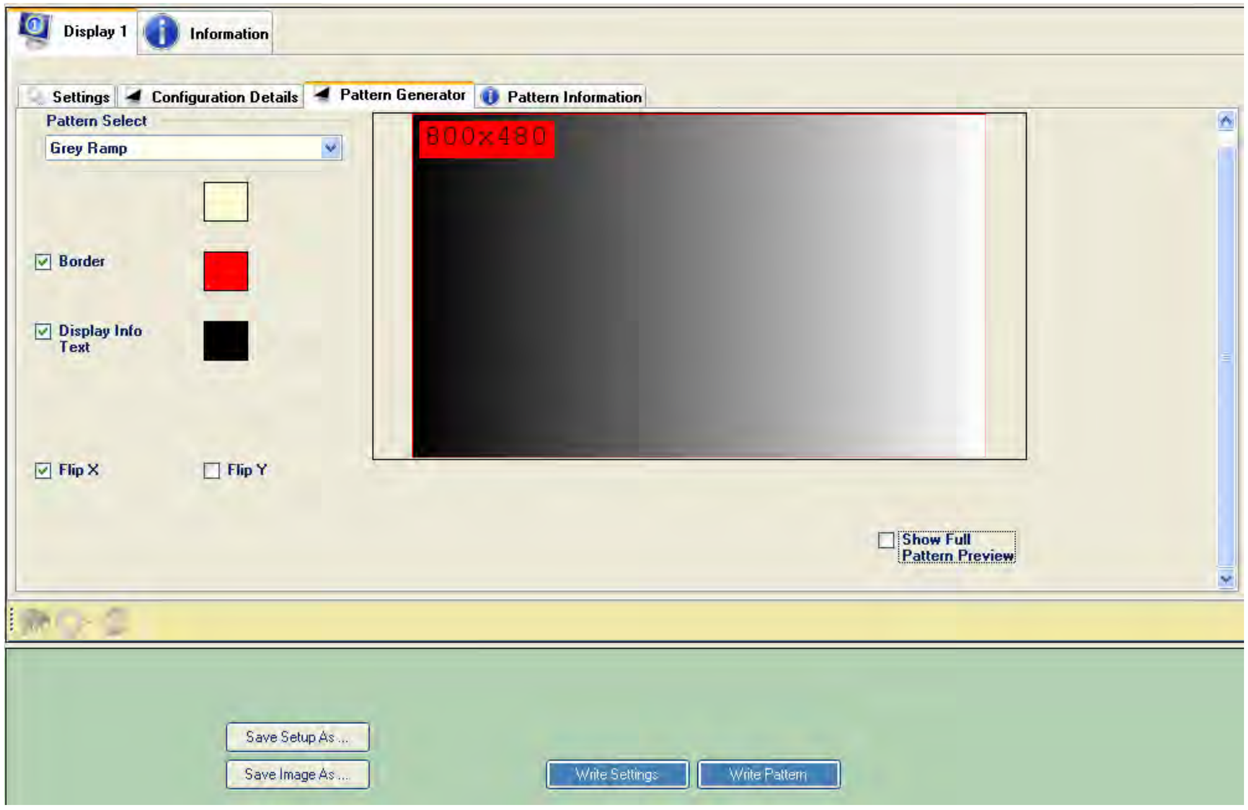
Success, path and settings automatically calculated.

7. Test Image Generator

To test timing changes made a Test Image Generator is available.
There are a few test images as well as some features to produce and apply such a test pattern.

Because of the small Indigo2 memory the test image is typically not a full calculated 32bit image.
The image is more a combination of different Iris features and path settings to get a displayed pattern similar to the full image - also the fetch sprite unit is used for this purpose.

Please be aware that the Iris will be setup and current values are lost after applying the test pattern.



Pattern Select: Grey Ramp

☑ Border

☑ Display Info Text

☑ Flip X

☐ Flip Y

800x480

☐ Show Full Pattern Preview

After pattern was setup do the following,

a. Write Pattern

By pressing this button the required pattern - or at least parts of it will be transferred to SRAM for further usage.

b. Write Settings

Here the Iris and its components will be setup to display the required image.

8. Save / Load Settings

To store current timing setup as well as the complete test image configuration a "Save As ..." and a "Save Settings" button is available below.

Typical storage format is .gdcfuncs.
Nevertheless from the complete configuration a par file can be generated that can be further processed in the Register Sequencer.

For loading a previous stored setup the stored .gdcfuncs file can be loaded.

9. Save Test Image

To save the complete Iris configuration required to setup the test image please press the "Save Setup As ..." button.

The "Save Image As ..." button will save the Image as it is displayed in the preview box.
Attention:

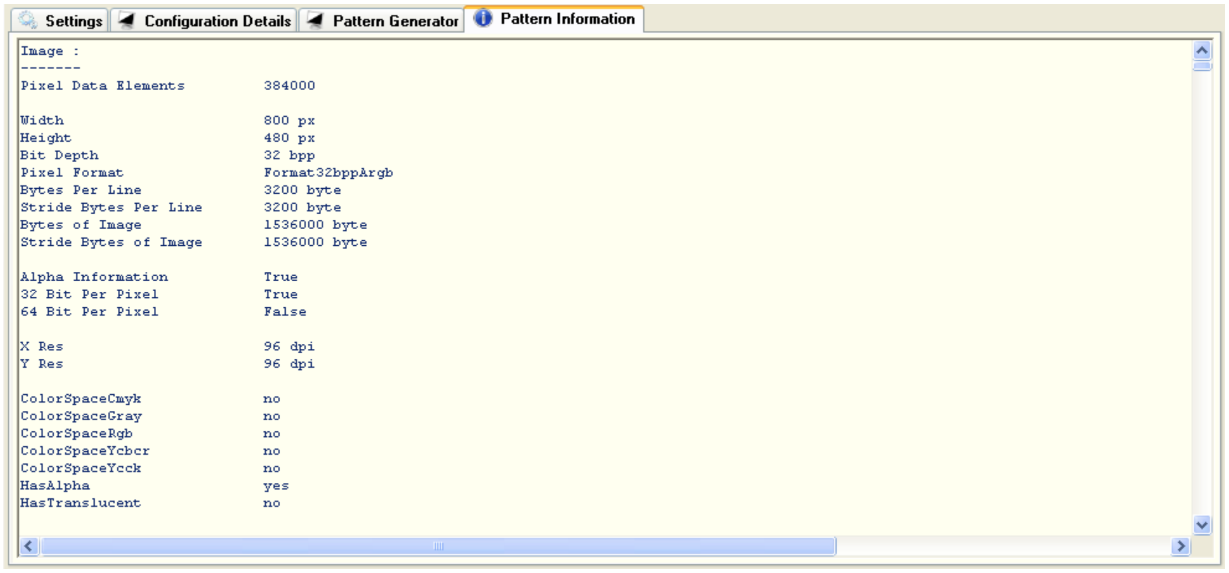
> the image will be stored as full image and is - or can be different than the real memory content of the Indigo2.
> This is because of the limited SRAM memory.
> Nevertheless for further processing or usage in a different context the full image can be of interest.

10. Other pages

a. Pattern Information Page

This page contains detailed information about the Image that is displayed in the preview box.
Attention:

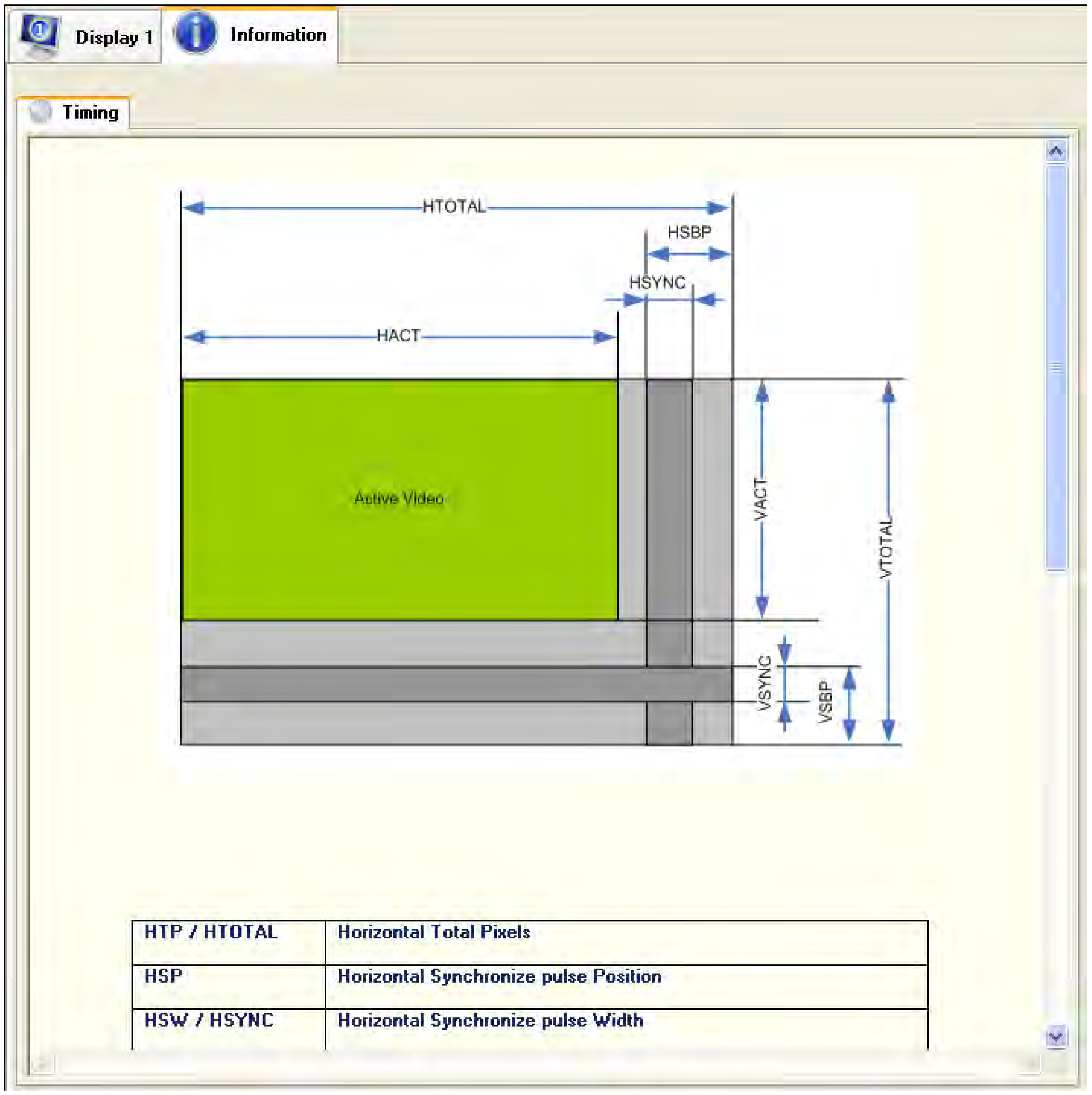
> the image will be stored as full image and is - or can be different than the real memory content of the Indigo2.
> This is because of the limited SRAM memory.
> Nevertheless for further processing or usage in a different context the full image can be of interest.



```
Image :
-------
Pixel Data Elements         384000

Width                       800 px
Height                      480 px
Bit Depth                   32 bpp
Pixel Format                Format32bppArgb
Bytes Per Line              3200 byte
Stride Bytes Per Line       3200 byte
Bytes of Image              1536000 byte
Stride Bytes of Image       1536000 byte

Alpha Information           True
32 Bit Per Pixel            True
64 Bit Per Pixel            False

X Res                       96 dpi
Y Res                       96 dpi

ColorSpaceCmyk              no
ColorSpaceGray              no
ColorSpaceRgb               no
ColorSpaceYcbcr             no
ColorSpaceYcck              no
HasAlpha                    yes
HasTranslucent              no
```

b. Global Information Page

This page contains some information to display / panel timings as well as supporting the user to read out the correct timing values from the panel specification.



HTP / HTOTAL	Horizontal Total Pixels
HSP	Horizontal Synchronize pulse Position
HSW / HSYNC	Horizontal Synchronize pulse Width

7. Indigo2 - Signature Manager

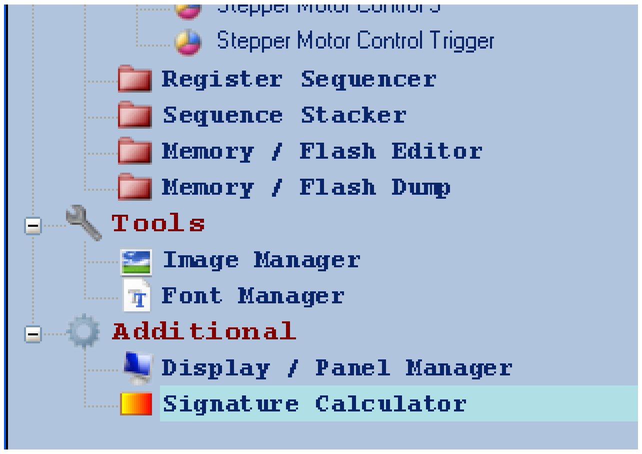
Main intention is to easily setup signature unit of the Indigo2.
Furthermore the signature information of a selected image area will be calculated and displayed for further usage in an application.

Remarks

- available for SPI (Aardvark, FTDI) and Ethernet (E2IP) connection type
- working with Indigo2

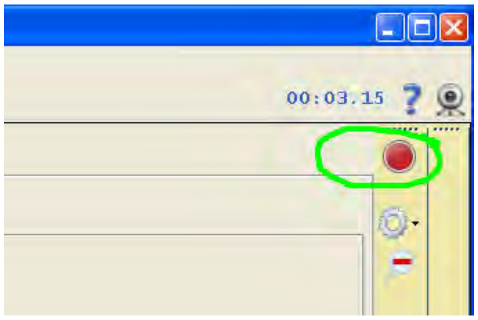
1. Select Display / Panel Manager Feature Page

Select the "Display / Panel Manager" page in the selection view.



2. Connect to Target Device

Depending on the connection type the connection devices must be scanned or configured before a connection can be established with the button below.



3. Global Signature Page

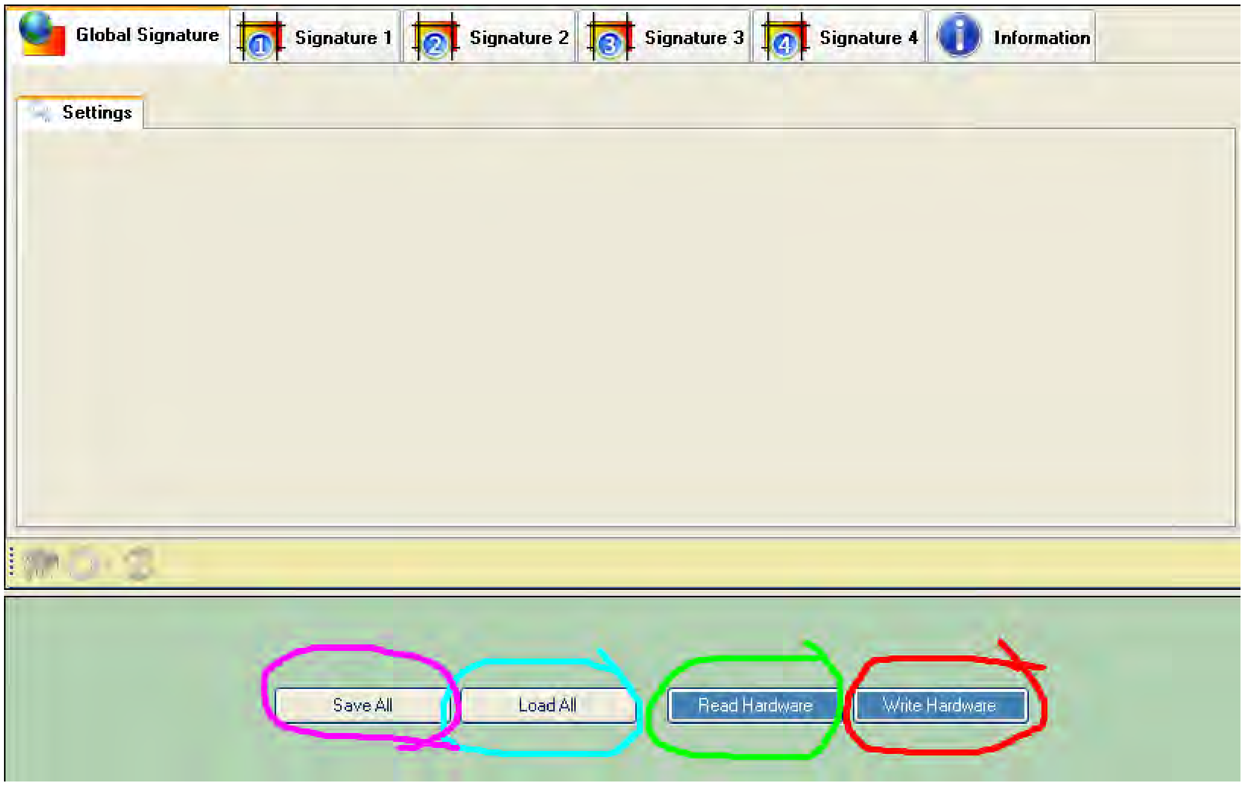
The main page is the so called global page which allows to configure global settings for the corresponding hardware if they are available.
In case of the Indigo2 signature unit there are currently no global settings supported that are equal for all available units.

Read Hardware
: Read all global values as well as the configuration values of the different units at once.

Write Hardware
: Write all global values as well as the configuration values of the different units at once.

Save All
: Save the settings of the complete Signature Manager including the settings of the different signature units.

Load All
: Load the settings of the complete Signature Manager including the settings of the different signature units.



Remarks :
On this page it is possible to store all timing settings from all pages into either a .gdcfunc file (which is also loadable on this page) or into a .par file which can be used in the Register Sequencer.

4. Signature Unit Setup

a. Configuration

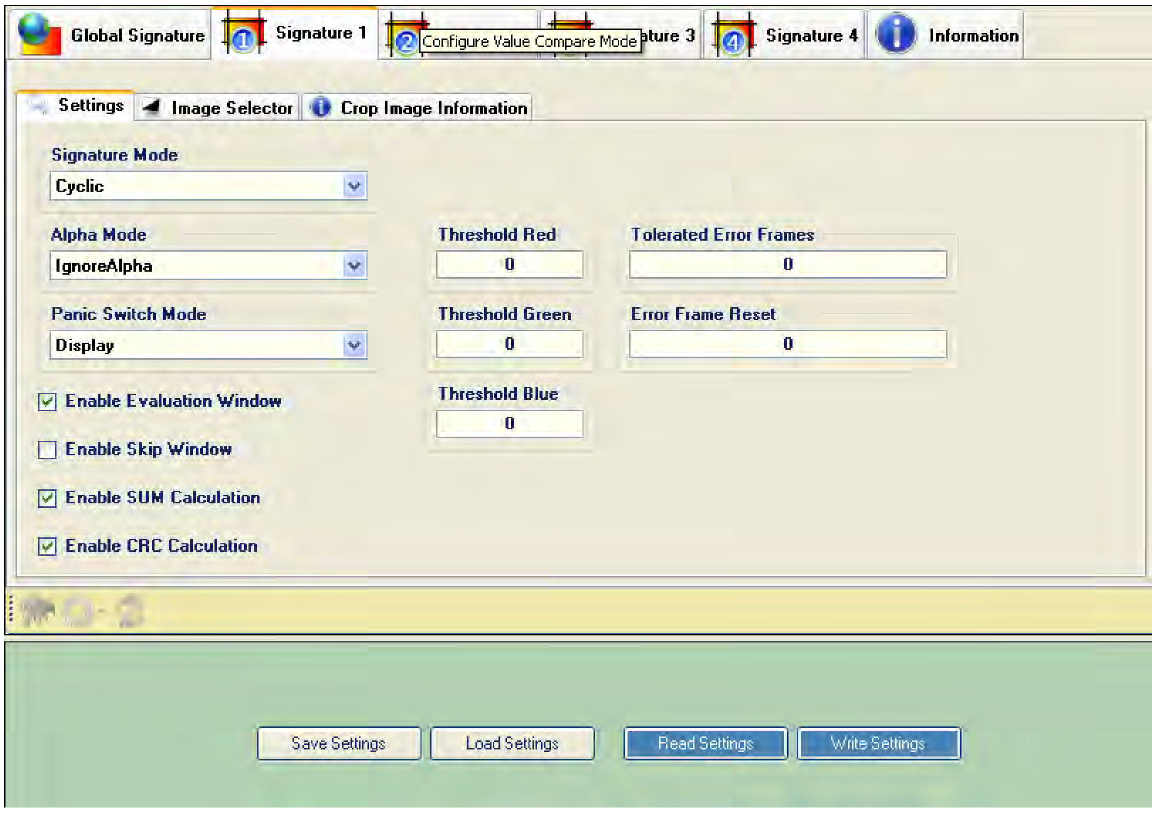
Select and configure the required settings for a specific signature unit on the corresponding page.

Read Settings
Read the hardware settings.
A saved configuration can be loaded into an other unit.

Write Settings
Write hardware settings.
A saved configuration can be loaded into an other unit.

Save Settings
Save single settings either as .par file or .gdcfunc file.

Load Settings
Load single settings from .gdcfunc file.



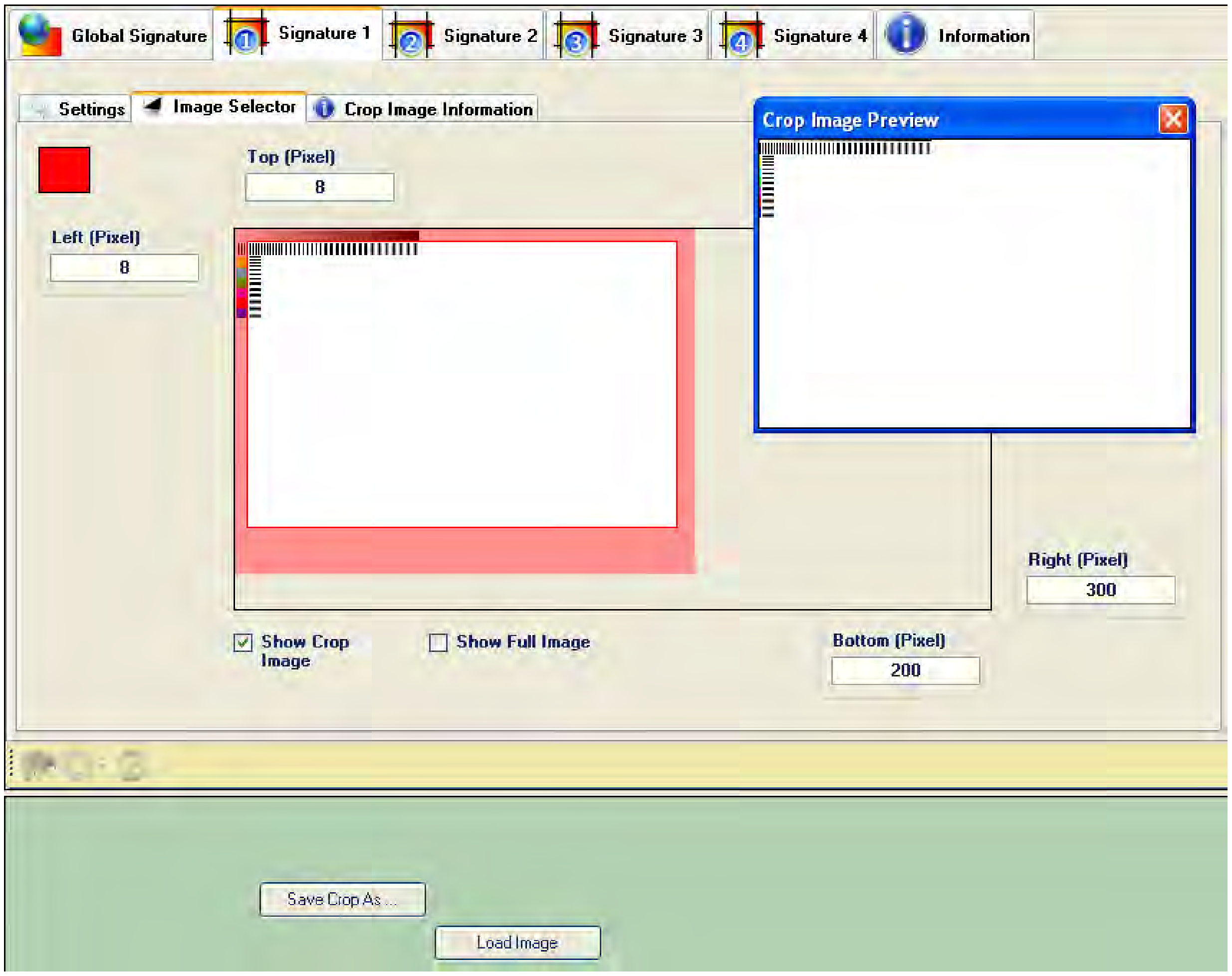
Signature Mode: Cyclic
Alpha Mode: IgnoreAlpha
Panic Switch Mode: Display
Threshold Red: 0
Threshold Green: 0
Threshold Blue: 0
Tolerated Error Frames: 0
Error Frame Reset: 0
[✓] Enable Evaluation Window
[ ] Enable Skip Window
[✓] Enable SUM Calculation
[✓] Enable CRC Calculation

b. Image

First step is to load the required image or pattern from a file into the preview window.

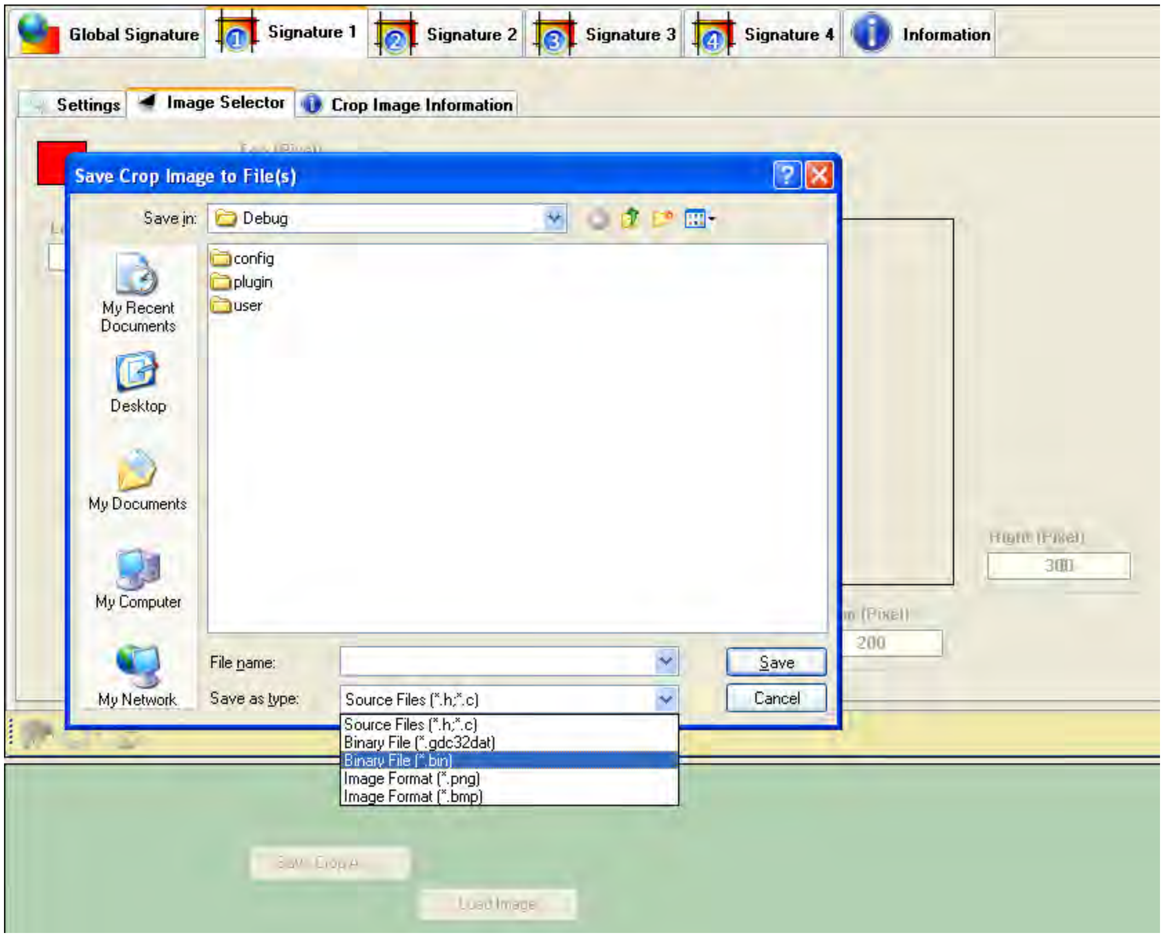
Then the essential signature window can be defined by entering the crop values in the Left, Top, Right and Bottom field - in Pixel.
The resulting crop window is visible while the invalid area will be marked out in a selectable color - partly transparent.
The crop image can also be displayed separately in an extra window when required.

The crop window is base for the signature calculations.



Top (Pixel) 8
Left (Pixel) 8
Right (Pixel) 300
Bottom (Pixel) 200

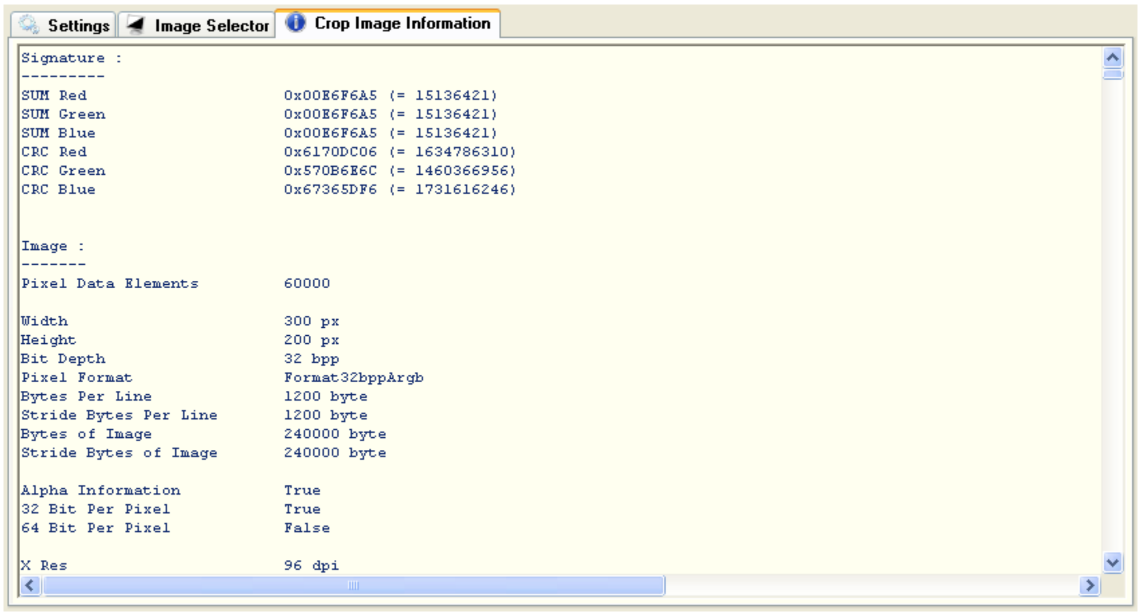
The crop image can separately be stored in different formats for further processing or usage in applications.
Typically as source files, binary file or image file.



Source Files (*.h;*.c)
Binary File (*.gdc32dat)
Binary File (*.bin)
Image Format (*.png)
Image Format (*.bmp)

c. Information

Signature information as well as crop image information will be displayed on the last tab page.
For Indigo2 the output pixel data is typically arranged in an RGBA array.



```
Signature :
---------
SUM Red                     0x00E6F6A5 (= 15136421)
SUM Green                   0x00E6F6A5 (= 15136421)
SUM Blue                    0x00E6F6A5 (= 15136421)
CRC Red                     0x6170DC06 (= 1634786310)
CRC Green                   0x570B6E6C (= 1460366956)
CRC Blue                    0x67365DF6 (= 1731616246)


Image :
-------
Pixel Data Elements         60000

Width                       300 px
Height                      200 px
Bit Depth                   32 bpp
Pixel Format                Format32bppArgb
Bytes Per Line              1200 byte
Stride Bytes Per Line       1200 byte
Bytes of Image              240000 byte
Stride Bytes of Image       240000 byte

Alpha Information           True
32 Bit Per Pixel            True
64 Bit Per Pixel            False

X Res                       96 dpi
```

8. Indigo / Indigo L Command Sequencer Support

The Indigo / Indigo L contains a command sequencer that can execute sequences of code stored in memory/flash.
The command sequencer can be used to trigger such sequences in certain situation like chip reset, signature unit trigger etc.

The Fujitsu Developer Suite contain support to create such command sequences out of the Register Sequencer.

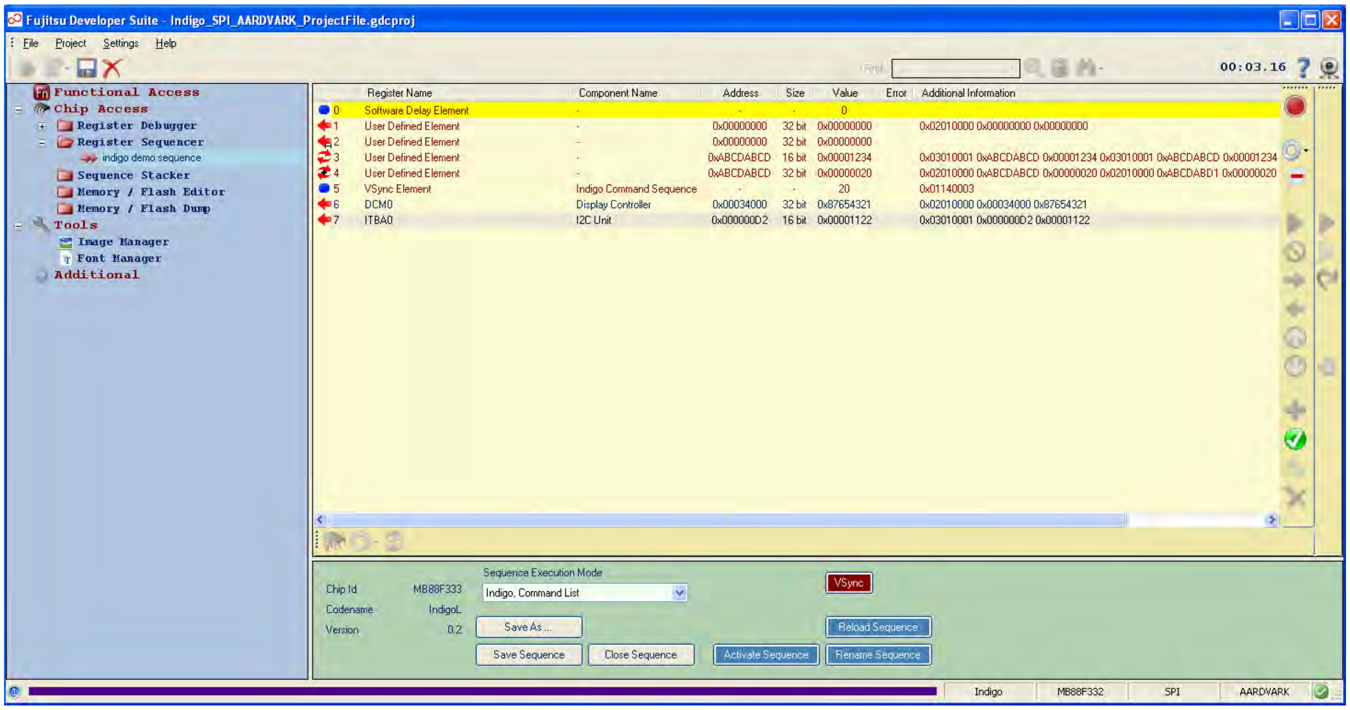
Please refer to the Register Debugger and the Register Sequencer section to get basic information on how to,

- create a new Register Sequence
- activate a specific Register Sequence as preparation for execution
- add registers from the Register Debugger onto the active Register Sequence
- create user defined elements in the Register Sequencer

Please refer also to the Customer Information section, Register Sequence Execution Modes.

1. Configure Register Sequence

Select and create a proper Register Sequence,


Fujitsu Developer Suite - Indigo_SPI_AARDVARK_ProjectFile.gdcproj

File Project Settings Help

00:03.16

- Functional Access
- Chip Access
  - Register Debugger
  - Register Sequencer
    - indigo demo sequence
  - Sequence Stacker
  - Memory / Flash Editor
  - Memory / Flash Dump
- Tools
  - Image Manager
  - Font Manager
- Additional

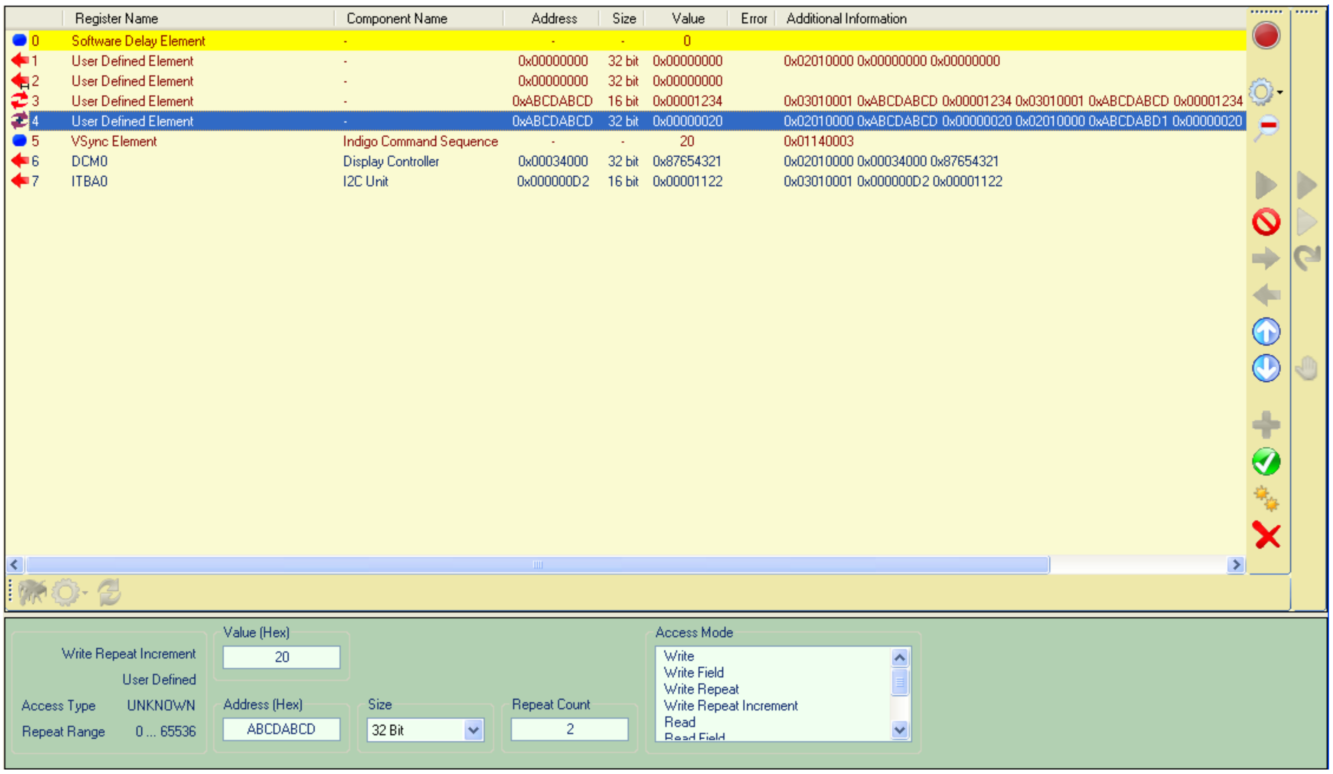
	Register Name	Component Name	Address	Size	Value	Error	Additional Information
0	Software Delay Element	-	-	-	0		
1	User Defined Element	-	0x00000000	32 bit	0x00000000		0x02010000 0x00000000 0x00000000
2	User Defined Element	-	0x00000000	32 bit	0x00000000		
3	User Defined Element	-	0xABCDABCD	16 bit	0x00001234		0x03010001 0xABCDABCD 0x00001234 0x03010001 0xABCDABCD 0x00001234
4	User Defined Element	-	0xABCDABCD	32 bit	0x00000020		0x02010000 0xABCDABCD 0x00000020 0x02010000 0xABCDABD1 0x00000020
5	VSync Element	Indigo Command Sequence	-	-	20		0x01140003
6	DCM0	Display Controller	0x00034000	32 bit	0x87654321		0x02010000 0x00034000 0x87654321
7	ITBA0	I2C Unit	0x000000D2	16 bit	0x00001122		0x03010001 0x000000D2 0x00001122

Chip Id	MB88F333
Codename	IndigoL
Version	0.2

Sequence Execution Mode: Indigo, Command List

VSync | Save As ... | Reload Sequence | Save Sequence | Close Sequence | Activate Sequence | Rename Sequence

Indigo | MB88F332 | SPI | AARDVARK


	Register Name	Component Name	Address	Size	Value	Error	Additional Information
0	Software Delay Element	-	-	-	0		
1	User Defined Element	-	0x00000000	32 bit	0x00000000		0x02010000 0x00000000 0x00000000
2	User Defined Element	-	0x00000000	32 bit	0x00000000		
3	User Defined Element	-	0xABCDABCD	16 bit	0x00001234		0x03010001 0xABCDABCD 0x00001234 0x03010001 0xABCDABCD 0x00001234
4	User Defined Element	-	0xABCDABCD	32 bit	0x00000020		0x02010000 0xABCDABCD 0x00000020 0x02010000 0xABCDABD1 0x00000020
5	VSync Element	Indigo Command Sequence	-	-	20		0x01140003
6	DCM0	Display Controller	0x00034000	32 bit	0x87654321		0x02010000 0x00034000 0x87654321
7	ITBA0	I2C Unit	0x000000D2	16 bit	0x00001122		0x03010001 0x000000D2 0x00001122

Write Repeat Increment

User Defined

Access Type	UNKNOWN
Repeat Range	0 ... 65536

Value (Hex): 20

Address (Hex): ABCDABCD

Size: 32 Bit

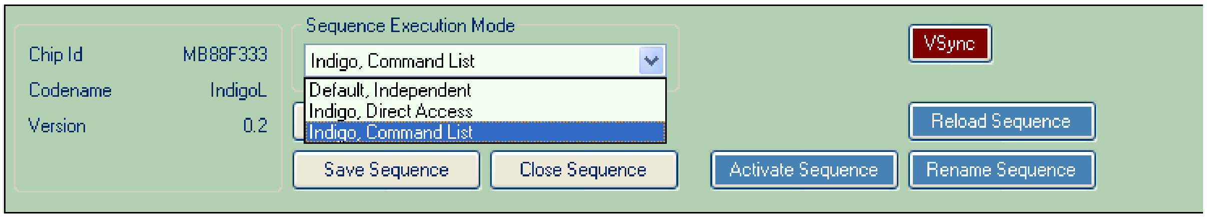
Repeat Count: 2

Access Mode:
- Write
- Write Field
- Write Repeat
- Write Repeat Increment
- Read
- Read Field

2. Sequence Execution Mode

By default the Sequence Execution Mode is set to Default / Chip Independent which means that all sequence items will be executed as single transfers and corresponding to the action. In this mode only standard actions will be available like Read, Write, Write Field etc.

So the next thing is to ensure that the Sequence Execution Mode is set to Command List.



Chip Id	MB88F333
Codename	IndigoL
Version	0.2

Sequence Execution Mode: Indigo, Command List

Only in this mode Command Sequencer specific items appear that can be used to continue with the Chip Specific sequence.

Remark :

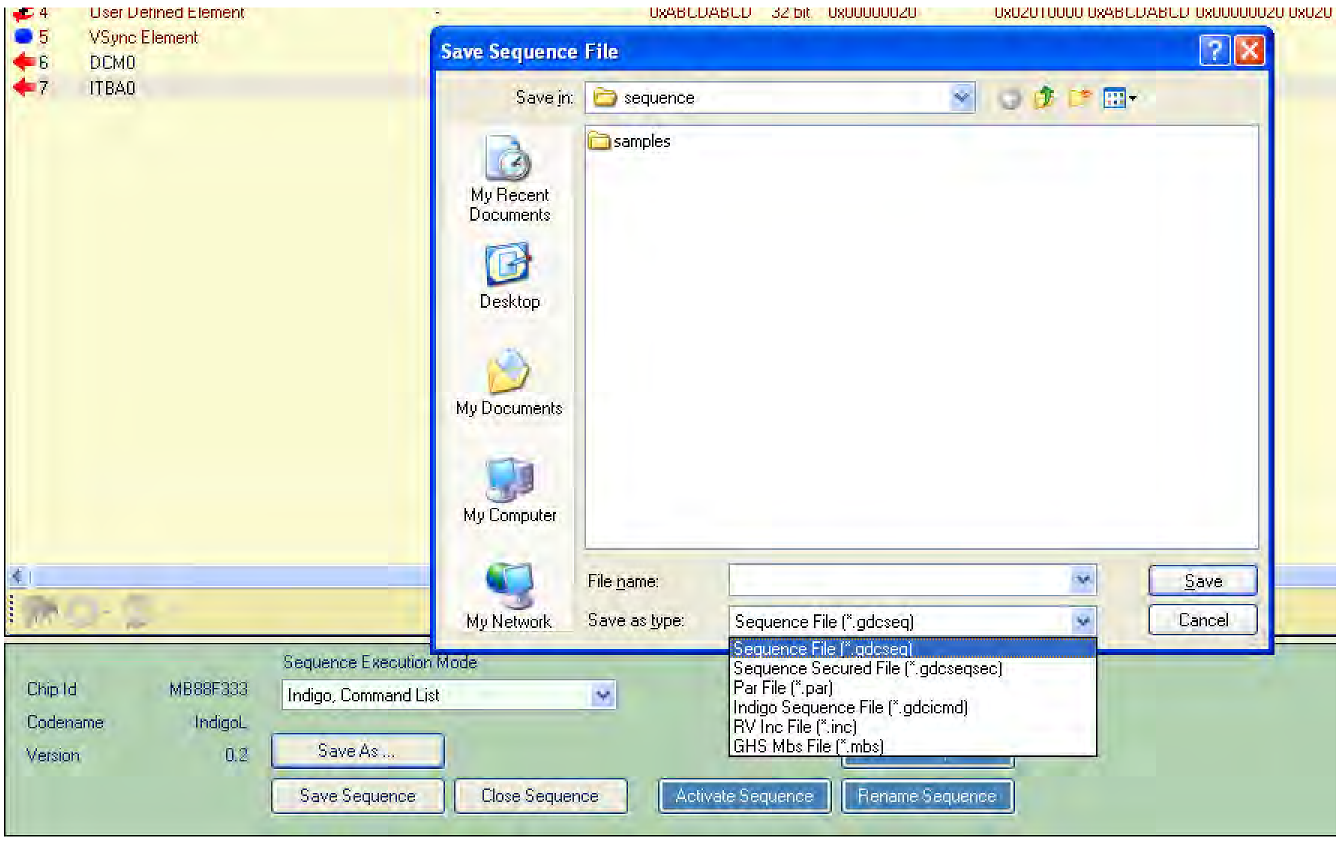
In the Command List mode standard actions like Read, Write etc. will be converted into command sequences - but be careful because some standard actions cannot be converted when there is no equal action available in the command sequencer.
All convertible actions (resulting in a valid command sequence) are visualized in the "Additional Information" column of the Register Sequencer, all non convertible commands contain no information and will be ignored when executing the sequence in this mode.

3. Save Sequence

The best method is to save the sequence as standard Sequence File (.gdcseq) or as Indigo Sequence File (.gdcicmd).

Remark :

Currently there is no difference between the storage option (.gdcseq) and (.gdcicmd). Sequences can also be saved in other formats like the .par format which is better human readable - but this format will currently not interpret actions into command sequences.



4 User Defined Element - 0xABCDABCD 32 bit 0x00000020 0x02010000 0xABCDABCD 0x00000020 0x020
5 VSync Element
6 DCM0
7 ITBA0

Save Sequence File

Save in: sequence

samples

File name:

Save as type: Sequence File (*.gdcseq)

Sequence File (*.gdcseq)
Sequence Secured File (*.gdcseqsec)
Par File (*.par)
Indigo Sequence File (*.gdcicmd)
RV Inc File (*.inc)
GHS Mbs File (*.mbs)

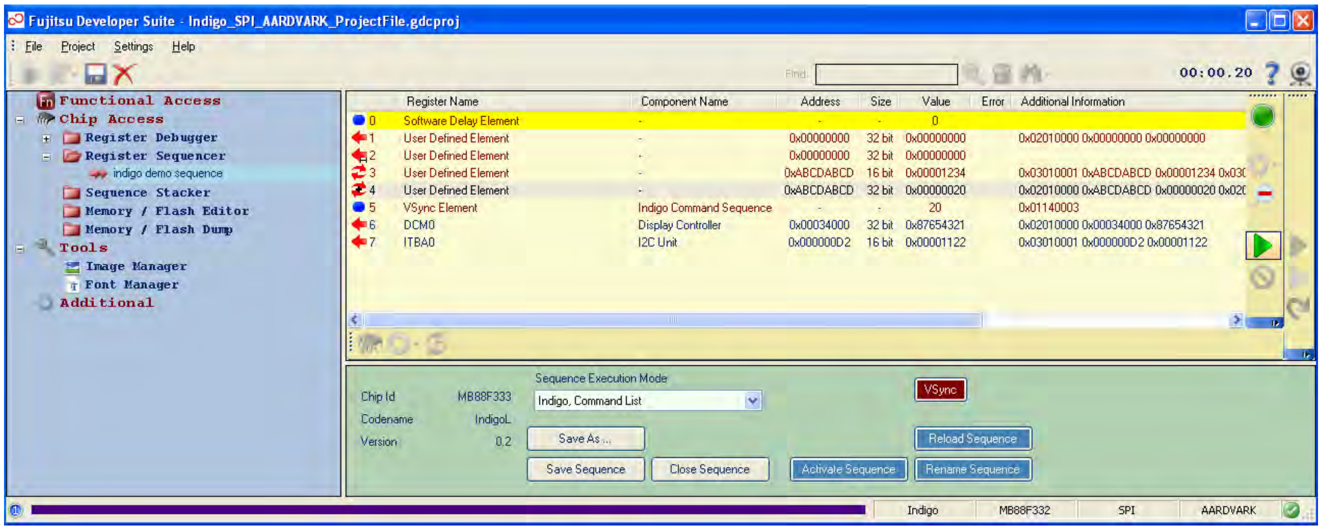
Chip Id MB88F333
Codename IndigoL
Version 0.2

Sequence Execution Mode: Indigo, Command List

4. Execute Sequence

The sequence can also be executed at once as command sequence.
For this do the following steps,

a. connect to the target
b. ensure that the execution sequence is the active one
c. ensure that the Sequence Execution is set to Command List
d. press the play button



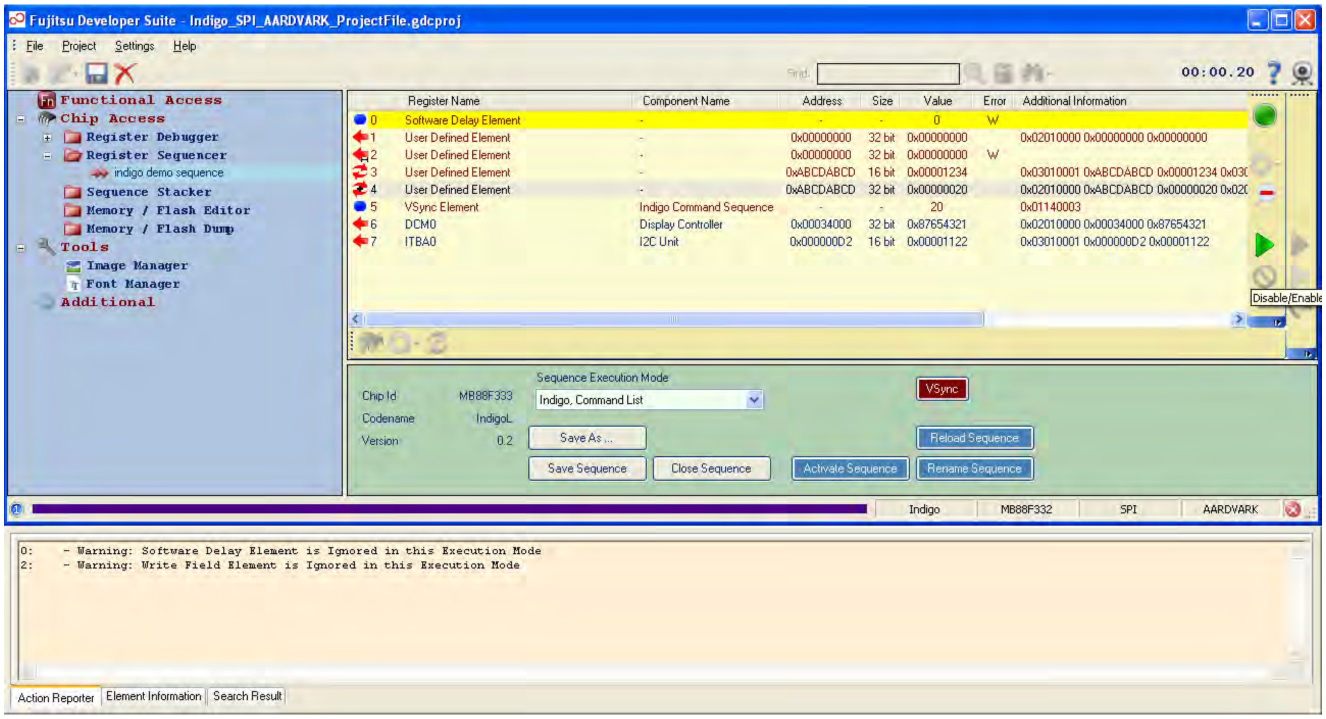
Fujitsu Developer Suite - Indigo_SPI_AARDVARK_ProjectFile.gdcproj

	Register Name	Component Name	Address	Size	Value	Error	Additional Information
0	Software Delay Element	-	-	-	0		
1	User Defined Element	-	0x00000000	32 bit	0x00000000		0x02010000 0x00000000 0x00000000
2	User Defined Element	-	0x00000000	32 bit	0x00000000		
3	User Defined Element	-	0xABCDABCD	16 bit	0x00001234		0x03010001 0xABCDABCD 0x00001234 0x030
4	User Defined Element	-	0xABCDABCD	32 bit	0x00000020		0x02010000 0xABCDABCD 0x00000020 0x020
5	VSync Element	Indigo Command Sequence	-	-	20		0x01140003
6	DCM0	Display Controller	0x00034000	32 bit	0x87654321		0x02010000 0x00034000 0x87654321
7	ITBA0	I2C Unit	0x000000D2	16 bit	0x00001122		0x03010001 0x000000D2 0x00001122

Sequence Execution Mode: Indigo, Command List

Chip Id	MB88F333
Codename	IndigoL
Version	0.2

Non convertible actions will be ignored on execution and a warning message will be displayed.



Fujitsu Developer Suite - Indigo_SPI_AARDVARK_ProjectFile.gdcproj

	Register Name	Component Name	Address	Size	Value	Error	Additional Information
0	Software Delay Element	-	-	-	0	W	
1	User Defined Element	-	0x00000000	32 bit	0x00000000		0x02010000 0x00000000 0x00000000
2	User Defined Element	-	0x00000000	32 bit	0x00000000	W	
3	User Defined Element	-	0xABCDABCD	16 bit	0x00001234		0x03010001 0xABCDABCD 0x00001234 0x030
4	User Defined Element	-	0xABCDABCD	32 bit	0x00000020		0x02010000 0xABCDABCD 0x00000020 0x020
5	VSync Element	Indigo Command Sequence	-	-	20		0x01140003
6	DCM0	Display Controller	0x00034000	32 bit	0x87654321		0x02010000 0x00034000 0x87654321
7	ITBA0	I2C Unit	0x000000D2	16 bit	0x00001122		0x03010001 0x000000D2 0x00001122

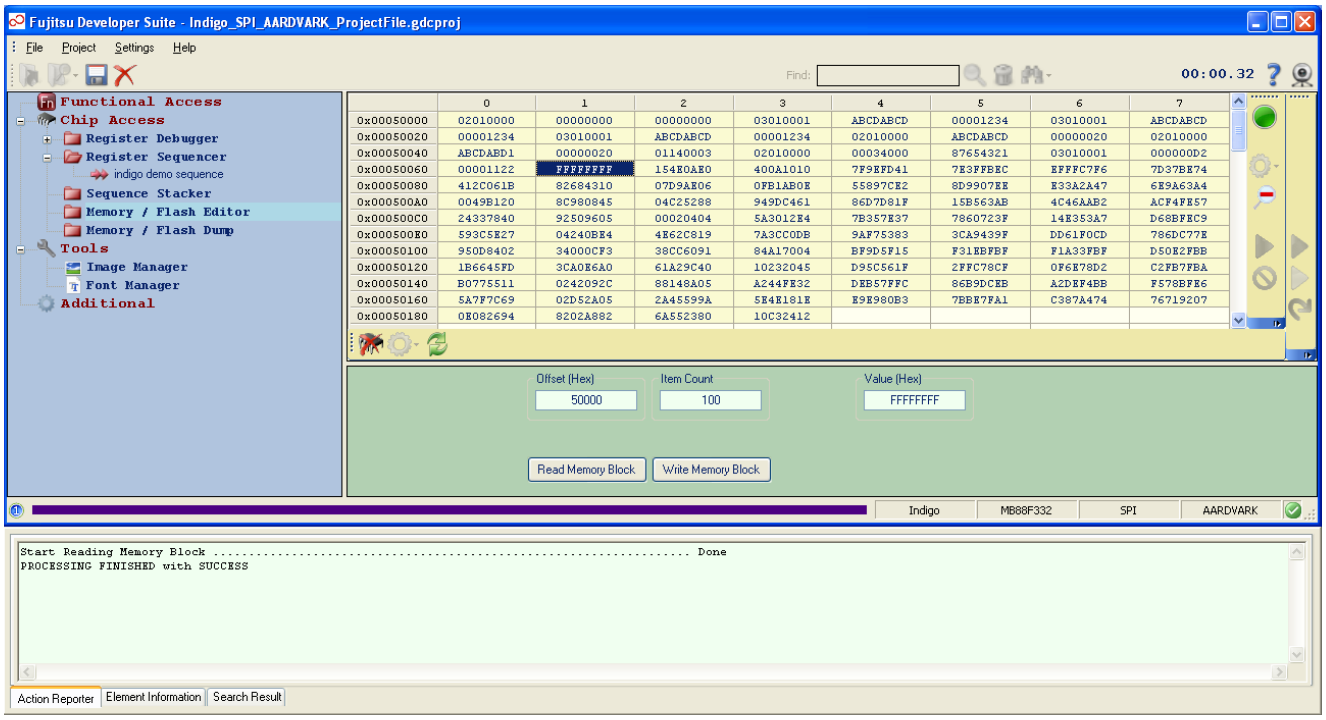
Sequence Execution Mode: Indigo, Command List

Chip Id	MB88F333
Codename	IndigoL
Version	0.2

```
0:    - Warning: Software Delay Element is Ignored in this Execution Mode
2:    - Warning: Write Field Element is Ignored in this Execution Mode
```

After pressing the play button all actions will be converted into a command sequence and copied into memory (here SRAM at address 0x50000).

The Command Sequencer will then be programmed to execute the sequence on that address.



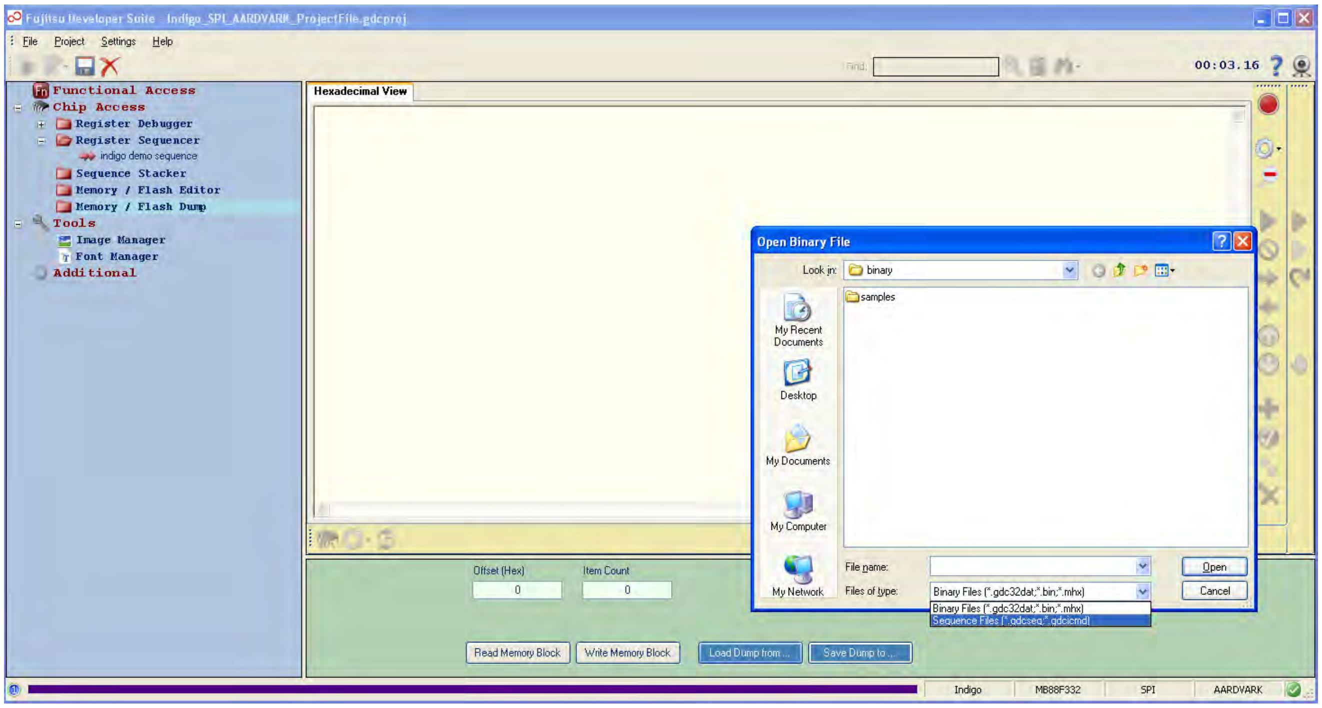
	0	1	2	3	4	5	6	7
0x00050000	02010000	00000000	00000000	03010001	ABCDABCD	00001234	03010001	ABCDABCD
0x00050020	00001234	03010001	ABCDABCD	00001234	02010000	ABCDABCD	00000020	02010000
0x00050040	ABCDABD1	00000020	01140003	02010000	00034000	87654321	03010001	000000D2
0x00050060	00001122	FFFFFFFF	154E0AE0	400A1010	7F9EFD41	7E3FFBEC	EFFFC7F6	7D37BE74
0x00050080	412C061B	82684310	07D9AE06	0FB1AB0E	55897CE2	8D9907EE	E33A2A47	6E9A63A4
0x000500A0	0049B120	8C980845	04C25288	949DC461	86D7D81F	15B563AB	4C46AAB2	ACF4FE57
0x000500C0	24337840	92509605	00020404	5A3012E4	7B357E37	7860723F	14E353A7	D68BFEC9
0x000500E0	593C5E27	04240BE4	4E62C819	7A3CC0DB	9AF75383	3CA9439F	DD61F0CD	786DC77E
0x00050100	950D8402	34000CF3	38CC6091	84A17004	BF9D5F15	F31EBFBF	F1A33FBF	D50E2FBB
0x00050120	1B6645FD	3CA0E6A0	61A29C40	10232045	D95C561F	2FFC78CF	0F6E78D2	C2FB7FBA
0x00050140	B0775511	0242092C	88148A05	A244FE32	DEB57FFC	86B9DCEB	A2DEF4BB	F578BFE6
0x00050160	5A7F7C69	02D52A05	2A45599A	5E4E181E	E9E980B3	7BBE7FA1	C387A474	76719207
0x00050180	0E082694	8202A882	6A552380	10C32412				

Offset (Hex)	Item Count	Value (Hex)
50000	100	FFFFFFFF

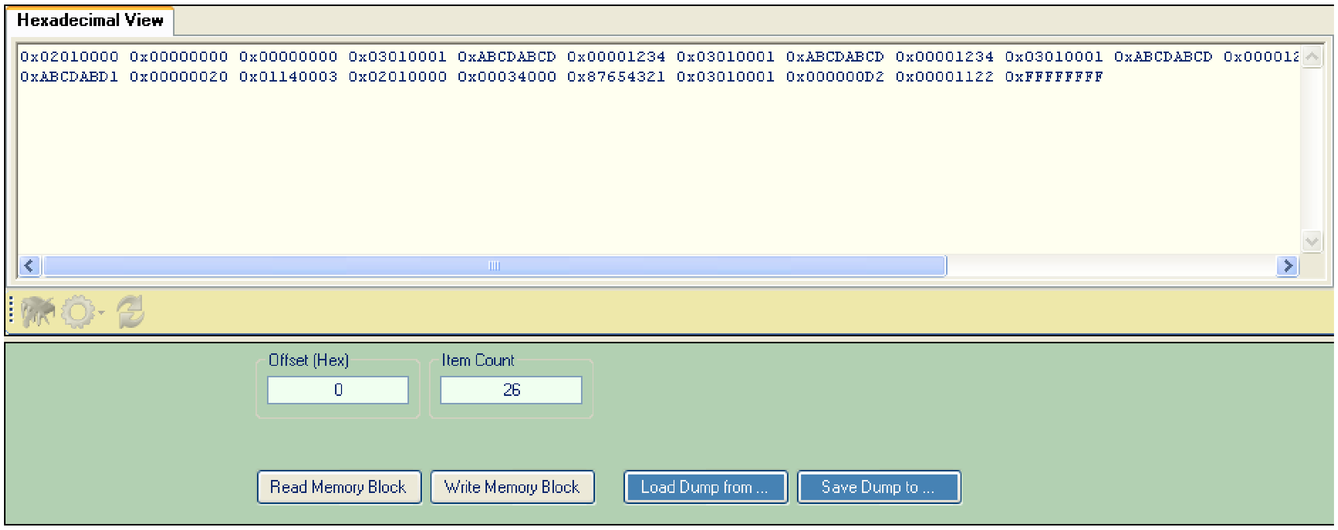
```
Start Reading Memory Block .................................................................. Done
PROCESSING FINISHED with SUCCESS
```

5. Load Sequence into Dump

Mostly it is useful to program the sequence into flash memory for certain actions like startup or some triggered events.
For this the saved sequence (.gdcseq or .gdcicmd) can be loaded into the Memory/Flash Dump Page.



By loading the sequence the content will automatically converted into an equal command sequence - which will then be displayed as 32bit stream.
This stream can then be handled as all other binary streams and either be copied into memory/flash or saved as binary and/or other file types.



```
0x02010000 0x00000000 0x00000000 0x03010001 0xABCDABCD 0x00001234 0x03010001 0xABCDABCD 0x00001234 0x03010001 0xABCDABCD 0x000012
0xABCDABD1 0x00000020 0x01140003 0x02010000 0x00034000 0x87654321 0x03010001 0x000000D2 0x00001122 0xFFFFFFFF
```

6. Supported Commands - Direct

Write 8bit
Write 16bit
Write 32bit
VSync, wait for

7. Supported Commands - Standard, Converted

Write
Write Repeat
Write Repeat Increment

9. Indigo 2 Command Sequencer Support

The Indigo2 contains a command sequencer that can execute sequences of code stored in memory/flash. The command sequencer can be used to trigger such sequences in certain situation like chip reset, signature unit trigger etc.

The Fujitsu Developer Suite contain support to create such command sequences out of the Register Sequencer.

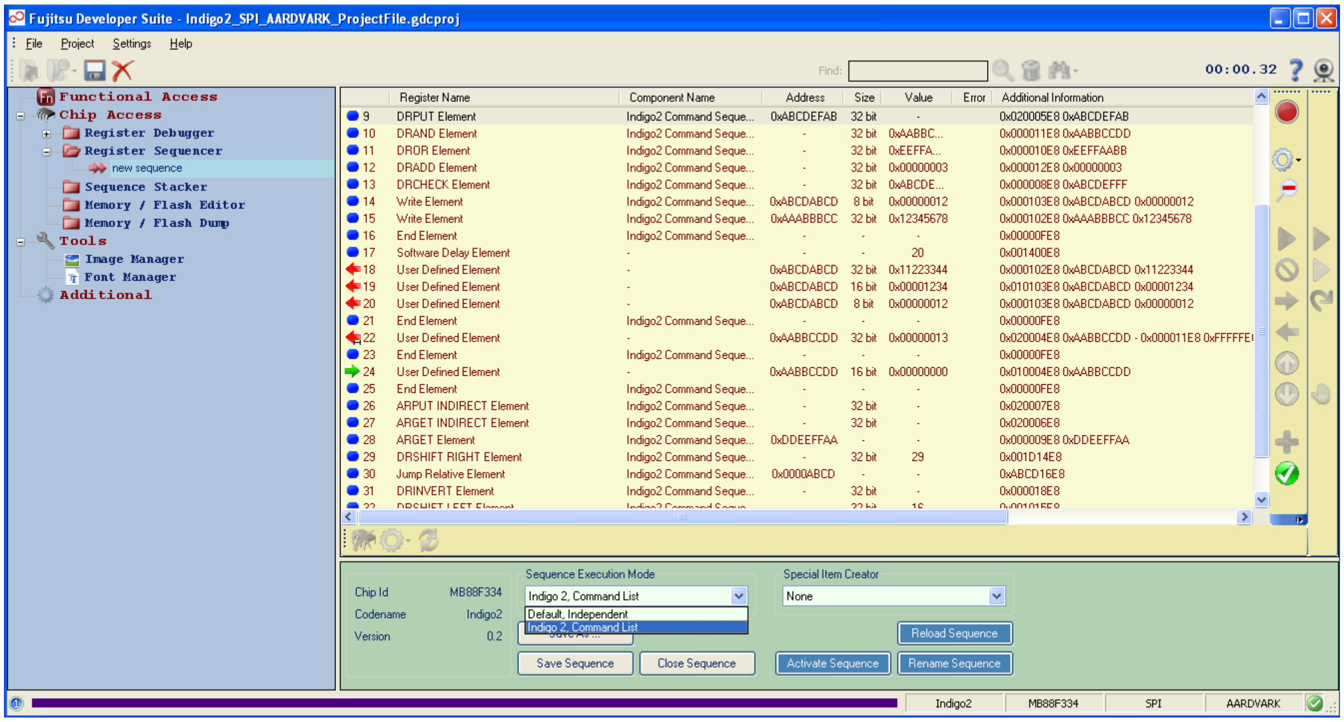
Please refer to the Register Debugger and the Register Sequencer section to get basic information on how to,

- create a new Register Sequence
- activate a specific Register Sequence as preparation for execution
- add registers from the Register Debugger onto the active Register Sequence
- create user defined elements in the Register Sequencer

Please refer also to the Customer Information section, Register Sequence Execution Modes.

1. Configure Register Sequence

Select and create a proper Register Sequence,



	Register Name	Component Name	Address	Size	Value	Error	Additional Information
9	DRPUT Element	Indigo2 Command Seque...	0xABCDEFAB	32 bit	-		0x020005E8 0xABCDEFAB
10	DRAND Element	Indigo2 Command Seque...	-	32 bit	0xAABBC...		0x000011E8 0xAABBCCDD
11	DROR Element	Indigo2 Command Seque...	-	32 bit	0xEEFFA...		0x000010E8 0xEEFFAABB
12	DRADD Element	Indigo2 Command Seque...	-	32 bit	0x00000003		0x000012E8 0x00000003
13	DRCHECK Element	Indigo2 Command Seque...	-	32 bit	0xABCDE...		0x000008E8 0xABCDEFFF
14	Write Element	Indigo2 Command Seque...	0xABCDABCD	8 bit	0x00000012		0x000103E8 0xABCDABCD 0x00000012
15	Write Element	Indigo2 Command Seque...	0xAAABBBCC	32 bit	0x12345678		0x000102E8 0xAAABBBCC 0x12345678
16	End Element	Indigo2 Command Seque...	-	-	-		0x00000FE8
17	Software Delay Element	-	-	-	20		0x001400E8
18	User Defined Element	-	0xABCDABCD	32 bit	0x11223344		0x000102E8 0xABCDABCD 0x11223344
19	User Defined Element	-	0xABCDABCD	16 bit	0x00001234		0x010103E8 0xABCDABCD 0x00001234
20	User Defined Element	-	0xABCDABCD	8 bit	0x00000012		0x000103E8 0xABCDABCD 0x00000012
21	End Element	Indigo2 Command Seque...	-	-	-		0x00000FE8
22	User Defined Element	-	0xAABBCCDD	32 bit	0x00000013		0x020004E8 0xAABBCCDD - 0x000011E8 0xFFFFFE
23	End Element	Indigo2 Command Seque...	-	-	-		0x00000FE8
24	User Defined Element	-	0xAABBCCDD	16 bit	0x00000000		0x010004E8 0xAABBCCDD
25	End Element	Indigo2 Command Seque...	-	-	-		0x00000FE8
26	ARPUT INDIRECT Element	Indigo2 Command Seque...	-	32 bit	-		0x020007E8
27	ARGET INDIRECT Element	Indigo2 Command Seque...	-	32 bit	-		0x020006E8
28	ARGET Element	Indigo2 Command Seque...	0xDDEEFFAA	-	-		0x000009E8 0xDDEEFFAA
29	DRSHIFT RIGHT Element	Indigo2 Command Seque...	-	32 bit	29		0x001D14E8
30	Jump Relative Element	Indigo2 Command Seque...	0x0000ABCD	-	-		0xABCD16E8
31	DRINVERT Element	Indigo2 Command Seque...	-	32 bit	-		0x000018E8

Chip Id	MB88F334
Codename	Indigo2
Version	0.2

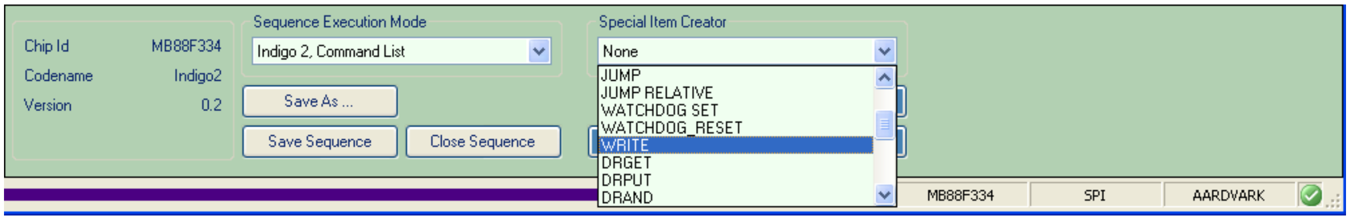
Sequence Execution Mode: Indigo 2, Command List
- Default, Independent
- Indigo 2, Command List

Special Item Creator: None

2. Sequence Execution Mode

By default the Sequence Execution Mode is set to Default / Chip Independent which means that all sequence items will be executed as single transfers and corresponding to the action. In this mode only standard actions will be available like Read, Write, Write Field etc.

So the next thing is to ensure that the Sequence Execution Mode is set to Command List.



Only in this mode Command Sequencer specific items appear that can be used to continue with the Chip Specific sequence.

Remark :

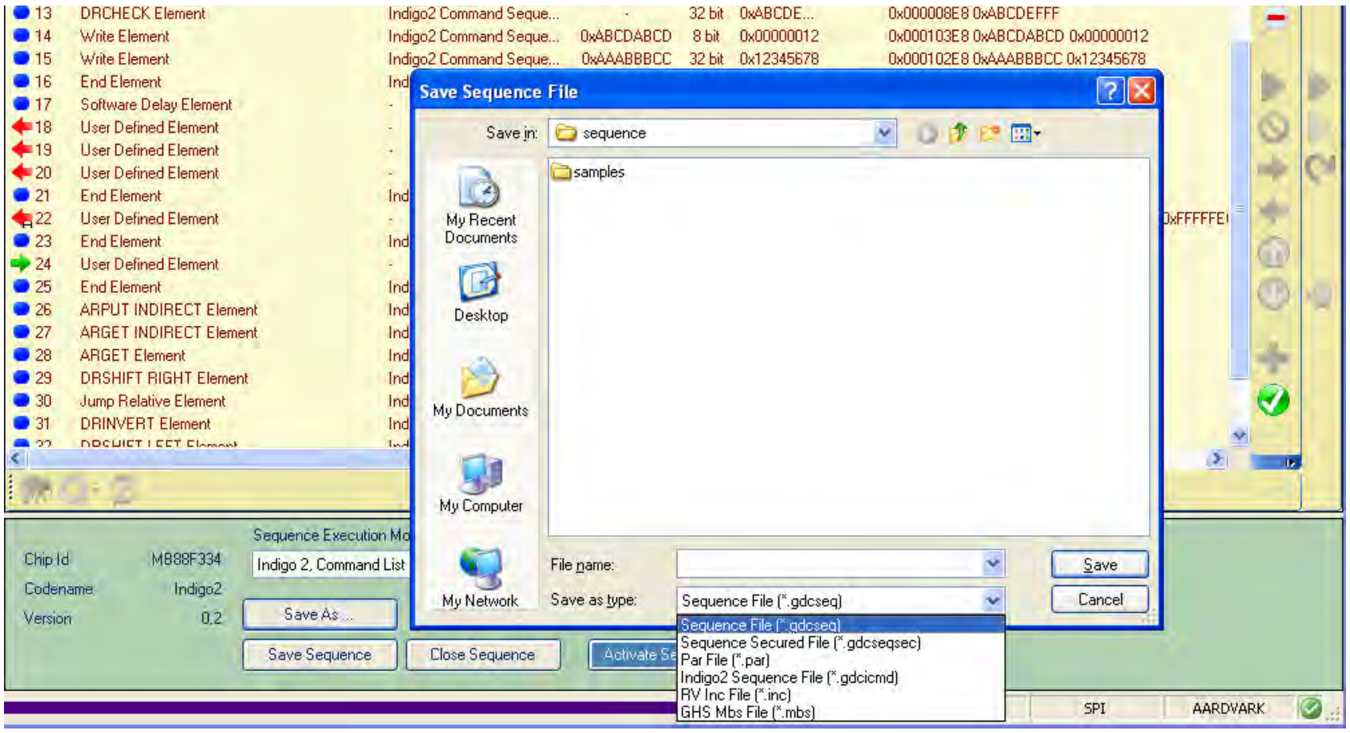
In the Command List mode standard actions like Read, Write etc. will be converted into command sequences - but be careful because some standard actions cannot be converted when there is no equal action available in the command sequencer.
All convertible actions (resulting in a valid command sequence) are visualized in the "Additional Information" column of the Register Sequencer, all non convertible commands contain no information and will be ignored when executing the sequence in this mode.

3. Save Sequence

The best method is to save the sequence as standard Sequence File (.gdcseq) or as Indigo2 Sequence File (.gdcicmd).

Remark :

Currently there is no difference between the storage option (.gdcseq) and (.gdcicmd). Sequences can also be saved in other formats like the .par format which is better human readable - but this format will currently not interpret actions into command sequences.



13	DRCHECK Element	Indigo2 Command Seque...	-	32 bit	0xABCDE...	0x000008E8 0xABCDEFFF
14	Write Element	Indigo2 Command Seque...	0xABCDABCD	8 bit	0x00000012	0x000103E8 0xABCDABCD 0x00000012
15	Write Element	Indigo2 Command Seque...	0xAAABBBCC	32 bit	0x12345678	0x000102E8 0xAAABBBCC 0x12345678
16	End Element	Ind				
17	Software Delay Element	-				
18	User Defined Element	-				
19	User Defined Element	-				
20	User Defined Element	-				
21	End Element	Ind				
22	User Defined Element	-				0xFFFFFE
23	End Element	Ind				
24	User Defined Element	-				
25	End Element	Ind				
26	ARPUT INDIRECT Element	Ind				
27	ARGET INDIRECT Element	Ind				
28	ARGET Element	Ind				
29	DRSHIFT RIGHT Element	Ind				
30	Jump Relative Element	Ind				
31	DRINVERT Element	Ind				

Chip Id MB88F334
Codename Indigo2
Version 0.2

Sequence Execution Mo
Indigo 2, Command List

Save Sequence File

Save in: sequence

samples

File name:

Save as type: Sequence File (*.gdcseq)

- Sequence File (*.gdcseq)
- Sequence Secured File (*.gdcseqsec)
- Par File (*.par)
- Indigo2 Sequence File (*.gdcicmd)
- RV Inc File (*.inc)
- GHS Mbs File (*.mbs)

4. Execute Sequence

The sequence can also be executed at once as command sequence.
For this do the following steps,

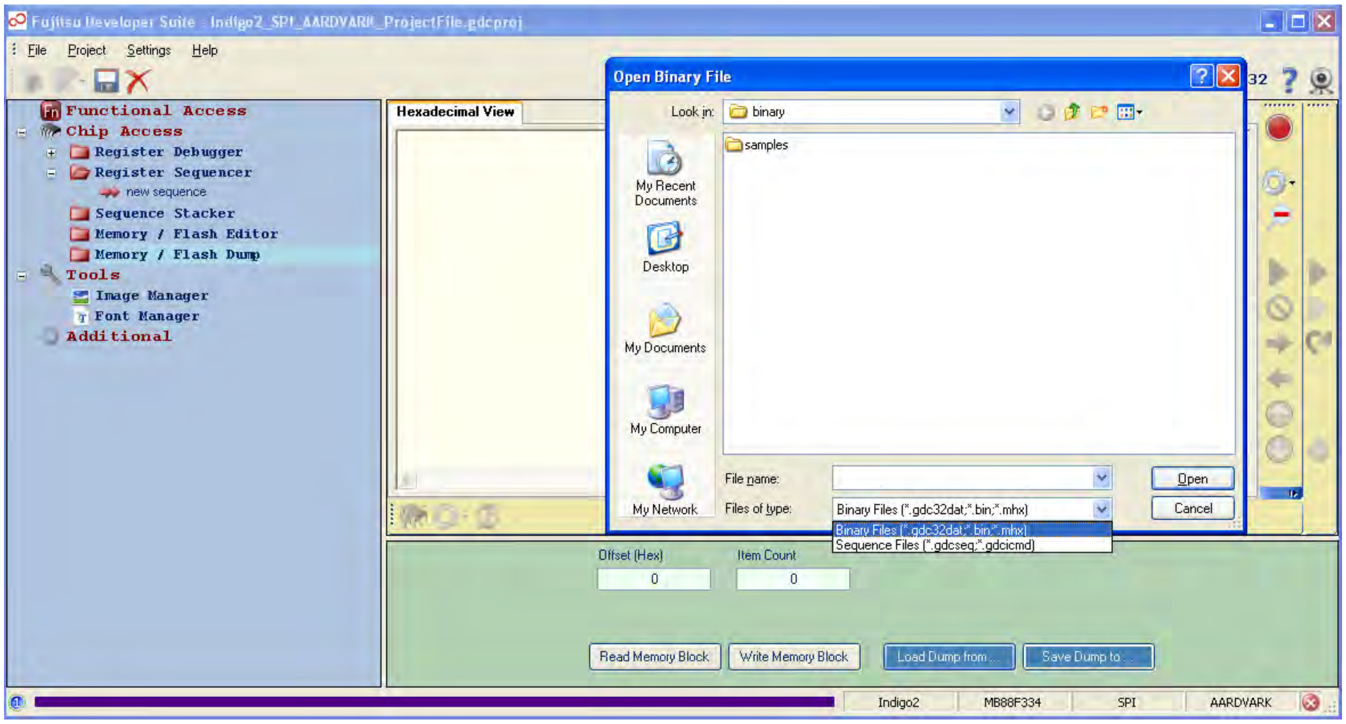
a. connect to the target
b. ensure that the execution sequence is the active one
c. ensure that the Sequence Execution is set to Command List
d. press the play button

Non convertible actions will be ignored on execution and a warning message will be displayed.

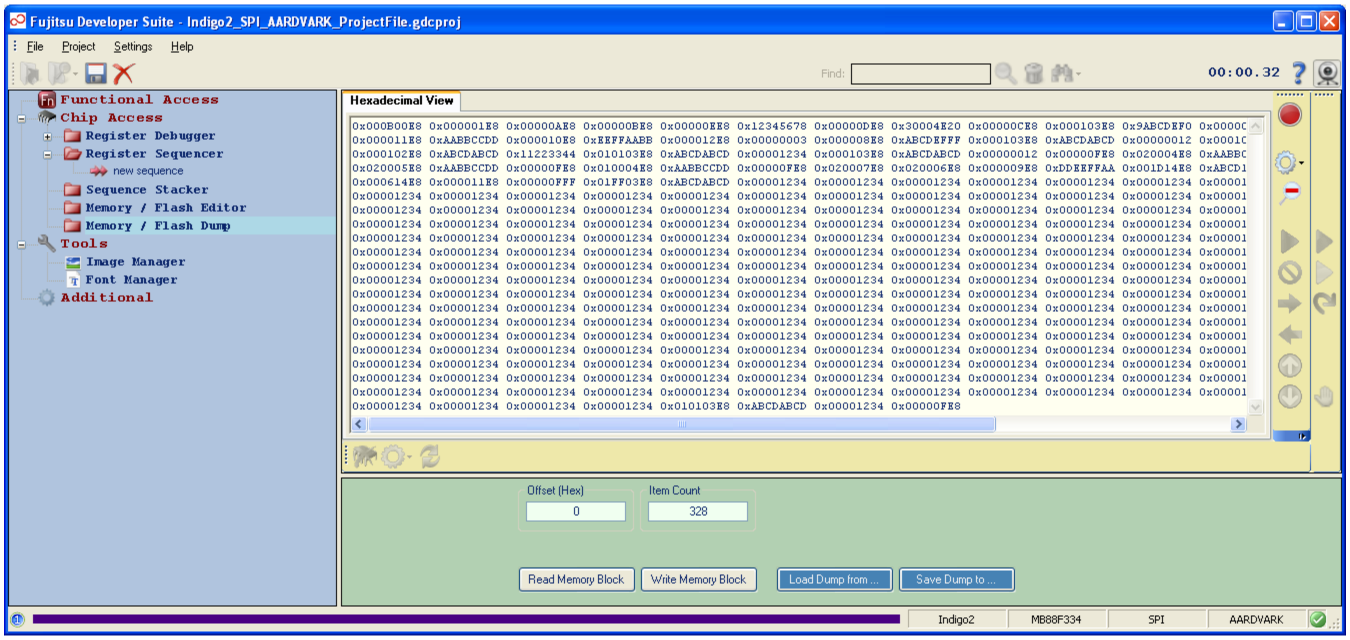
After pressing the play button all actions will be converted into a command sequence and copied into memory.
The Command Sequencer will then be programmed to execute the sequence on that address.

5. Load Sequence into Dump

Mostly it is useful to program the sequence into flash memory for certain actions like startup or some triggered events.
For this the saved sequence (.gdcseq or .gdcicmd) can be loaded into the Memory/Flash Dump Page.



By loading the sequence the content will automatically converted into an equal command sequence - which will then be displayed as 32bit stream.
This stream can then be handled as all other binary streams and either be copied into memory/flash or saved as binary and/or other file types.



```
0x000B00E8 0x000001E8 0x00000AE8 0x00000BE8 0x00000EE8 0x12345678 0x00000DE8 0x30004E20 0x00000CE8 0x000103E8 0x9ABCDEF0 0x0000C
0x000011E8 0xAABBCCDD 0x000010E8 0xEEFFAABB 0x000012E8 0x00000003 0x000008E8 0xABCDEFFF 0x000103E8 0xABCDABCD 0x00000012 0x0001C
0x000102E8 0xABCDABCD 0x11223344 0x010103E8 0xABCDABCD 0x00001234 0x000103E8 0xABCDABCD 0x00000012 0x00000FE8 0x020004E8 0xAABBC
0x020005E8 0xAABBCCDD 0x00000FE8 0x010004E8 0xAABBCCDD 0x00000FE8 0x020007E8 0x020006E8 0x000009E8 0xDDEEFFAA 0x001D14E8 0xABCD1
0x000614E8 0x000011E8 0x00000FFF 0x01FF03E8 0xABCDABCD 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001234 0x00001
0x00001234 0x00001234 0x00001234 0x00001234 0x010103E8 0xABCDABCD 0x00001234 0x00000FE8
```

Offset (Hex): 0
Item Count: 328

6. Supported Commands - Direct

WAIT	Wait Element

This instruction performs a delay.
The number of microseconds can be specified by the Count operand.
Due to implementation issues, the overall delay can be larger (up to 3 microseconds) than the specified Count value but will never be shorter.

SWINT Software Interrupt Element

This instruction generates a pulse on swint_o output signal which should be connected to interrupt controller.

LABEL Label Element

Store current program counter address to EREG register.
This can be used for implementation of backward loops.

LOOP Loop Element

Continue execution at address stored in EREG register.
This can be used for implementation of backward loops.

JUMP Jump Element

Continue execution at provided Address.
This instruction is like a jump and won't return.

JUMP RELATIVE Jump Relative Element

Continue execution at provided distance.

WATCHDOG RESET Watchdog Reset

This instruction resets the watchdog timer.
It must be executed within a limited time given by the watchdog load register and the divider value.

WATCHDOG SET Watchdog Set

This instruction does the setup of the watchdog timer.
If Divider and Counter parameters are all '0', watchdog timer will be disabled, otherwise timer will be started with the specified values.
Doing a new WDS instruction while timer is running also starts the timer with the new values immediately

WRITE Write Element

Write list of data to destination buffer.

DRGET Data Register Get Element

Get data from address and store it in local DREG register.
8 bit, 16 bit and 32 bit transfers can be performed.
Address has to be aligned to the transfer size.

DRPUT Data Register Put Element

Store data from local DREG register to Address.
8 bit, 16 bit and 32 bit transfers can be performed.
Address has to be aligned to the transfer size.

DRAND Data Register And Element

Logical and of DREG content with value.

DROR Data Register Or Element
Logical or of DREG content with value.

DRINVERT Data Register Invert Element
Bitwise logical not of DREG content.

DRSHIFT LEFT Data Register Shift Left Element
Logical shift left of DREG content.

DRSHIFT RIGHT Data Register Shift Right Element
Logical shift right of DREG content.

DRADD Data Register Add Element
Add value to DREG content.

DRCHECK Data Register Check Element
Compare bits of DREG with provided Value and skip next instruction when result is equal.

ARGET Address Register Get Element
Get data from address and store it in local AREG register.

ARGET INDIRECT Address Register Get Indirect Element
Get data from address in AREG register and store it in local DREG register.
8 bit, 16 bit and 32 bit transfers can be performed.
AREG value has to be aligned to the transfer size.

ARPUT INDIRECT Address Register Put Indirect Element
Store data from local DREG register to address in AREG register.
8 bit, 16 bit and 32 bit transfers can be performed.
AREG value has to be aligned to the transfer size.

END End Element
Stop execution of the current command.

7. Supported Macro Commands - Standard, Converted

Write
Write Field
Write Repeat
Write Repeat Increment
Read
Read Field
Delay

ATTENTION : ongoing

XI. Troubleshooting ...

1. mscoree.dll

Problem :

... the dynamic link library mscoree.dll could not be found ...

When this error message appears then the required *.NET Framework* is not installed on the target computer.

Solution :

Please install the *.NET Framework 2.0* on the target computer.
Then connect to the *Microsoft Update Server* to get the latest updates and fixes.

2. Security Warning

Problem :

... The publisher could not be verified. Are you sure you want to run this software ? ...

When an error dialog appears that contains the above mentioned message then the application was probably started from a network device.
After selecting the "**Run**" button the application crashes.

Solution :

The application can only be installed and used on a *Local Computer*.
Executing the application on a network device will lead to a security warning and / or crash.

3. Installation

Problem :

... Unable to install because a newer version of this product is already installed. ...

When this dialog appears then there is already an instance installed on the target computer which is newer than the version which should be installed.

Solution :

Normally it is useful to install always the latest version of the application.
Nevertheless if it is required to install a previous version then the current one must be uninstalled first.
This can be done in the **Control Panel -> Add or Remove Software**, selecting the application and pressing the "**Remove**" button.

4. Flash Problems

Problem :

It is not possible to properly Write to ... or Read from ... the Internal Flash Memory.

Root Cause / Solution :

1. Loss of connection

It is possible that the connection to the target device over the USB to SPI/JTAG (PC connection) is lost.

This can be checked by opening,

Indigo2 :
the *Chip Control Unit* page in the *Register Debugger*, selecting the *ChipInfo* register - Address: 0x00000000

Indigo :
the *Chip Control Unit* page in the *Register Debugger*, selecting the *ChipInfo* register - Address: 0x10000

JadeD :
the *Chip Control Unit* page in the *Register Debugger*, selecting the *CCID* register - Address: 0xFFF42000

Ruby :
the *Global Controller Unit* page in the *Register Debugger*, selecting the *CHIP INFO* register - Address: 0x30020050

EmeraldL :
the *Chip Control Unit* page in the *Register Debugger*, selecting the *CINFO* register - Address: 0x3D100000

EmeraldP :
the *Chip Control Unit* page in the *Register Debugger*, selecting the *CINFO* register - Address: 0x3D100000

Triton :
the *Chip Control Unit* page in the *Register Debugger*, selecting the *CINFO* register - Address: 0x3D100000

and reading the content.

When reading happens without any problems then the *Action Reporter* docking dialog will be highlighted in a light green color without any messages on it.
If reading fails because of connection problems then the Action Reporter docking dialog appears in a light red color with some error messages attached.

Solution :

- close the application
- check the connection to the USB to SPI/JTAG device
- check the connection from the USB to SPI/JTAG device to the target board
- check the power supply

- open the application and check connection again

2. No Flash available

Attention : Some Chip Designs does not have Flash Memory.

If having an Indigo FPGA Evaluation Board then NO REAL FLASH is available.
In the FPGA Version the *Flash Memory* will be simulated by means of *RAM*.
Because the application is expecting *Flash Memory* instead of *RAM* it is using special flash commands for reading / writing onto it.

Solution :

- disable the flash support
- as Offset (Hex) enter the base address of the *Internal Flash Memory* (e.g. A0000 on Indigo)
- now it is possible to read / write to this simulated flash area like onto every memory / register area

5. Connection / Disconnection Problems

Problem :

It is not possible to connect / disconnect the ... properly with / from the target device.

Root Cause / Solution :

1. Connection / Disconnection Button fail

- Before a connection can taken place the Device Ports must be scanned and the required Connection Device have to be selected.
  This can be done with the "Configure Hardware Connection" Button immediately below the Connection/Disconnection Button.

2. Conflicting Applications

When using another tool or application beside the Fujitsu Developer Suite which also accessing the communication interface of the same target device it is possible that these applications conflicting each other.

Solution :

- close all open applications that are accessing the communication interface of the same target device (e.g. SPI, JTAG)
- now open only those application that is required for the current usage

6. Unknown Error Or Exception

Attention : Expert Users Only !
All other Users please contact us.

Problem :

Unknown Error Or Exception.

Root Cause / Solution :

On any other unknown error or exception the following can be done.
Read carefully BEFORE executing the following steps.

1. Enter Registry :
   - Windows Start Menu -> Run
   - in the text field enter : **regedit**
     (press return afterwards)
2. Remove the Registry Entry :
   - step through the registry tree and select the following Key / Entry :
     [HKEY_LOCAL_MACHINE]¥¥SOFTWARE¥¥Fujitsu Semiconductor Europe GmbH - GCC¥¥Fujitsu Developer Suite
   - now delete the Registry Key / Entry - and only this one

Attention : Any other deleted Key / Entry in the Registry can harm your PC !

3. Close the Registry and Restart.
   The Application will now set all internal values to default and add cleaned information to the Registry automatically.

If the Error or Exception still occurs please contact us.

XII. Customer Information

1. Register Sequence Execution Modes

For some devices (e.g. Indigo, Indigo2) the Fujitsu Developer Suite supports two different modes of executing register sequences. Depending on the mode the behavior can be different.
Please also refer to the corresponding "How to ..." sections,
"Indigo - Command Sequencer Support"
"Indigo2 - Command Sequencer Support"
as well as to the Register Sequencer sections,
Indigo Specific
Indigo2 Specific

Default, Independent Mode

This mode is the standard mode which accesses the registers directly as they are listed in the sequence.
Each sequence item action will be performed separately and at once depending on the speed and behavior of the current connection type (e.g. SPI, Ethernet).
On transmission error the sequence will be stopped immediately, whilst on success the sequence will be performed until finished.

For executing the register sequence in this mode please ensure that the
“Sequence Execution Mode” is set to **Default, Independent**.

Command List / Command Sequencer Mode

This mode is a chip specific mode.
All sequence items will be internally converted into real command sequencer code.
Then the complete command stream will automatically being copied into the SRAM memory of the Indigo2.
Finally the Command Sequencer will be set up to execute the complete sequence at once.

When this mode is selected additional command sequence macros that are directly supported by the Command Sequencer will be offered as register sequence items.
But please keep in mind that those chip specific macros can only being executed in this mode.

For executing the register sequence in this mode please ensure that the
“Sequence Execution Mode” is set to e.g. **Indigo2, Command List**.

Remark :
This execution mode is typically needed by customers for testing real chip behavior.
Furthermore it can be used for converting existing register sequences into real command sequence code and storing it as “.bin” file as preparation for flashing it into memory.

2. Initialization Sequence

By default the Fujitsu Developer Suite needs to setup certain registers to have full access to the complete target device. For this reason an initialization sequence will be performed automatically when connecting to the chip/board.

This may implies that debugging may behave differently compared to the standalone operation of the device. For testing customer developed startup code or specific command sequences it is recommended to disable this automatic initialization on connection.

This can be done in the following way,

Disconnect from the current device or ensure being disconnected.

Enable the following item which can be found in the “Settings” menu of the Fujitsu Developer Suite menu bar,

"Suppress Default Chip Initialization on Connection when possible"

Now all new established connections will be performed without the default chip initialization.
But please keep in mind that it is possible that not all registers can be accessed because of missing unlock commands or similar initialization actions.
Nevertheless it is always possible to return to the standard behavior when disabling the upper mentioned menu item.

3. Emerald High Speed Flashing / Advanced Flashing

EmeraldL ES2
EmeraldP

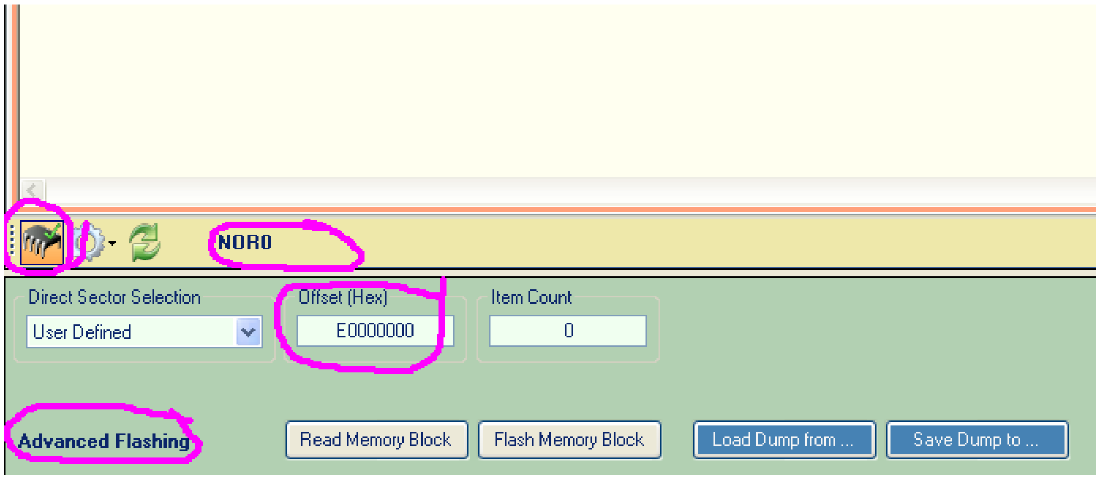
This flash programmer is based on an executable binary which is downloaded to target memory.
For a proper execution it is required that some basic chip initializations as well as memory initializations are done before.

The corresponding initialization sequence - that will be automatically executed before flashing is possible - is stored in a special chip dependent par file,

/user/EmeraldL/sequence/flash/**mb86r11_flash_init.par**
/user/EmeraldP/sequence/flash/**mb86r12_flash_init.par**

Remark :
The mentioned/delivered sequences are prepared for the Fujitsu Development Boards only.
If a customer is using a special design which is not compatible to the reference design (e.g. different memory type, pinmux setting etc.) it is required to change/adapt the initialization file accordingly.

Check :
If DDR initialization was correct after stepping into flash mode you will see an image similar to the below one - please check the marked positions.
Attention: "NOR0" and/or "E0000000" can differ depending on the Flash Type


NOR0

Direct Sector Selection: User Defined
Offset (Hex): E0000000
Item Count: 0

4. Indigo2 High Speed Flashing / Advanced Flashing

Indigo2

This advanced flash programmer is based on extended command sequencer features.

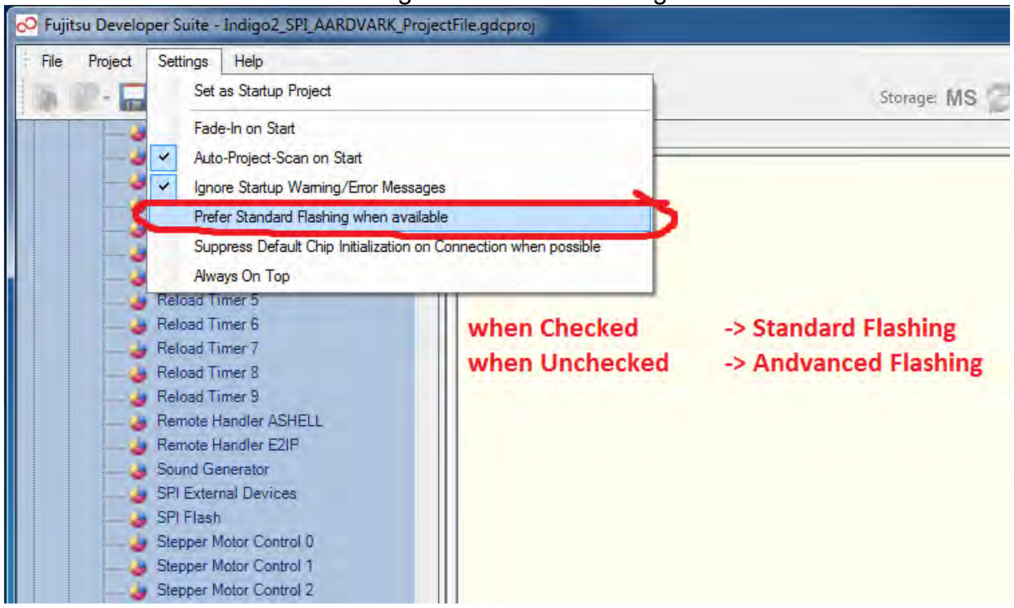
Remark :
To ensure that Advanced Flashing is valid the following two points should be checked,

1. "Prefer Standard Flashing when available" must be disabled in the Menu
2. When flashing is enabled and the Fujitsu Developer Suite is connected to the target device, "Advanced Flashing" must be visible.

Typical measured Flashing Time with the SPI/Aardvark connection.
Attention:
Connection Speed strongly depends on board design and electrical stability.
Values are valid for the Fujitsu Promotion Board as well as for the Fujitsu Validation Board.

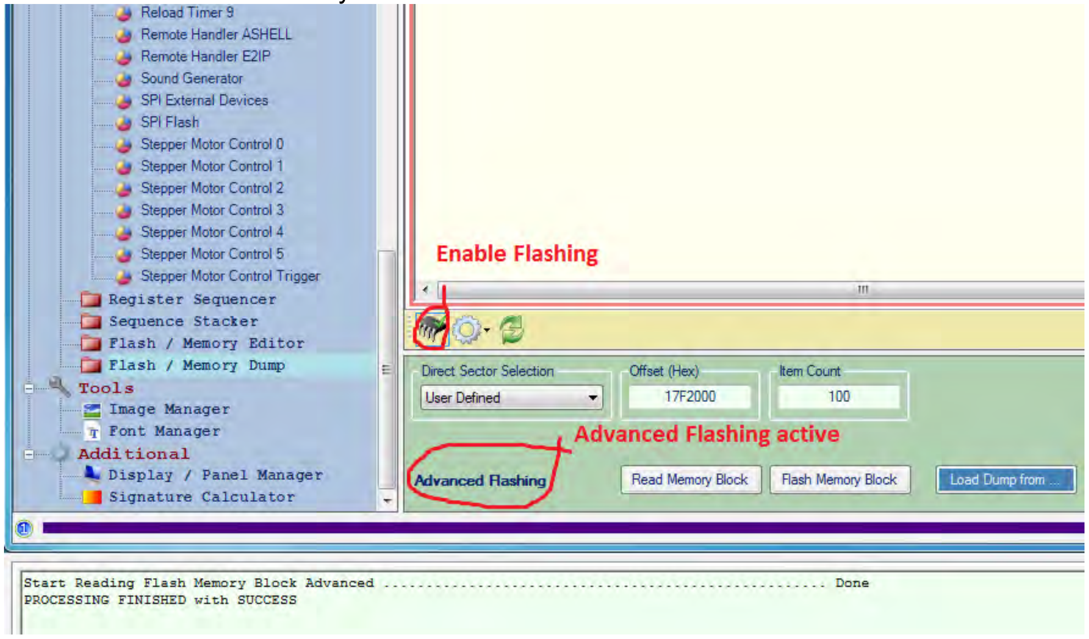
	SPI/Aardvark	SPI/Aardvark
	Standard Flashing	**Advanced Flashing**
Flash Write - 32kByte	~ 2.34 minutes	~ 4.5 seconds
Flash Read - 32kByte	~ 4.8 seconds	~ 4.8 seconds

Enable Advanced Flashing or Standard Flashing



when Checked -> Standard Flashing
when Unchecked -> Andvanced Flashing

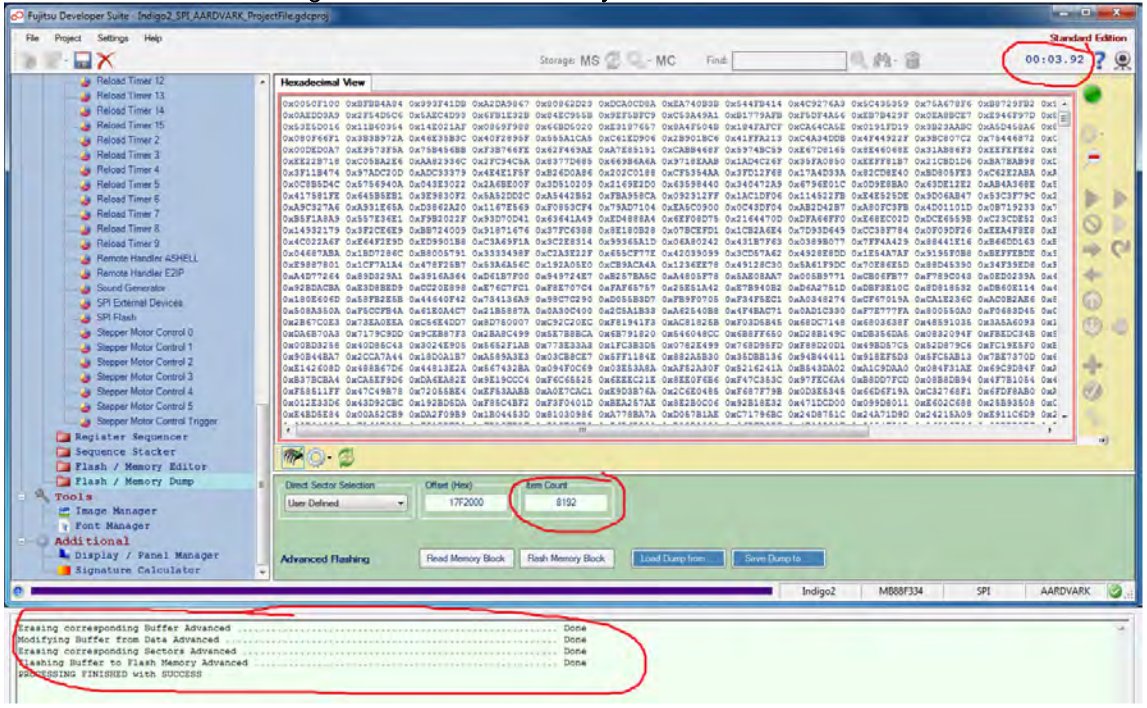
Switch Flash/Memory Editor to Flash Mode



Direct Sector Selection: User Defined
Offset (Hex): 17F2000
Item Count: 100

```
Start Reading Flash Memory Block Advanced .............................................. Done
PROCESSING FINISHED with SUCCESS
```

Advanced Flashing - Sector 0 Write - 32kByte



Direct Sector Selection: User Defined
Offset (Hex): 17F2000
Item Count: 8192

```
Erasing corresponding Buffer Advanced ......................................... Done
Modifying Buffer from Data Advanced ........................................... Done
Erasing corresponding Sectors Advanced ........................................ Done
Flashing Buffer to Flash Memory Advanced ...................................... Done
PROCESSING FINISHED with SUCCESS
```

5. Connection Device

SPI

- **Totalphase, Aardvark SPI Device**

  To ensure that the Aardvark Device is working properly
  it is required to update the **Firmware** of the Aardvark Device
  to Version **3.50**, or higher – compatible versions only.
  Furthermore it is necessary to install the latest **USB drivers**
  which are **v2.10, or later**.

  To update the firmware as well as the latest drivers
  please visit the Homepage of Total Phase, Inc. under
  www.totalphase.com and download the required software.

  It has been detected that the transfer rate of the Aardvark Device
  can get about 3 times faster when connecting the Aardvark over a
  powered USB Hub to the PC.

  This connection type is not available for all targets.

- **Future Technology Device International, FT4232 SPI Device**

  To ensure that the FTDI SPI Device is working properly
  please ensure using the latest **Driver v2.08.14 WHQL or higher**.

  To update to the latest drivers please visit the Homepage at
  www.ftdichip.com and download the required software.

  This connection type is not available for all targets.

JTAG

- **Segger, JLINK JTAG Device**

  To ensure that JLINK Device is working properly
  please ensure using the latest **Driver v4.32 or higher**.

  To update to the latest drivers please visit the Homepage
  of SEGGER Microcontroller GmbH under
  www.segger.com and download the required software.

  This connection type is not available for all targets.

ETHERNET (SSH)

- **Fujitsu Linux BSP**

For using the Ethernet connection for a specific target device it is required that a Linux OS is running on the target platform.
Typically there is a Fujitsu Linux BSP available which is prepared for this connection.
Communication over SSH/SCP.

This connection type is not available for all targets.

ETHERNET (E2IP Protocol)

Special protocol type based on the UDP protocol.

This connection type is currently only available for Indigo2.

6. Par File Format

The Par File Format is a standard text file containing human readable data which reflects actions that can be executed by the Register Sequencer.
Each line can contain one command (see syntax) or can be empty - but no two commands are allowed to be in the same line.

Syntax:

- **Installation Information**

  The first two lines must contain target device identifiers to ensure accessing the correct device.
  Currently they must be one of the following,

  Ruby :
  # MB86298
  # Ruby

  Indigo :
  # MB88F332
  # Indigo

  IndigoL :
  # MB88F333
  # IndigoL

  JadeD :
  # MB86R01
  # JadeD

  EmeraldL :
  # MB86R11
  # EmeraldL

  EmeraldP :
  # MB86R12
  # EmeraldP

  Indigo2 :
  # MB88F334
  # Indigo2

  Triton :
  # MB8AC0440
  # Triton

  ApCo :
  # MB86R091
  # ApCo

- **Comment**

  Any comment within the file starts with a hash followed by a space,

  e.g.
  # comment ...

- **Write Element**

  For writing data different access types are allowed,

  32 bit :
  w addr data
  16 bit :
  w16 addr data
  8 bit :
  w8 addr data

  Where "addr" means the address to write "data" to.
  All strings are separated by a space.
  Both values can be either decimal (no prefix) or hexadecimal (prefix: 0x).

- **Write Field Element**

  For writing field data different access types are allowed,

  32 bit :
  wf addr fieldOffset fieldWidth fieldValue
  16 bit :
  wf16 fieldOffset fieldWidth fieldValue
  8 bit :
  wf8 fieldOffset fieldWidth fieldValue

  Where "fieldOffset" means the bit offset within the register
  (0 <= fieldOffset < register width).
  "fieldWidth" means the width of the required field in bits.
  (1 <= fieldWidth < (register width - fieldOffset))
  "fieldValue" is the value of the field itself
  (0 <= fieldValue < 2 ^ fieldWidth)
  All strings are separated by a space.
  All values can be either decimal (no prefix) or hexadecimal (prefix: 0x).

- **Read Element**

  For reading data different access types are allowed,

  32 bit :
  r addr
  16 bit :
  r16 addr
  8 bit :
  r8 addr

  Where "addr" means the address to read from.
  All strings are separated by a space.

The address can be either decimal (no prefix) or hexadecimal (prefix: 0x).

- **Read Field Element**

For reading field data different access types are allowed,

32 bit :
    rf addr fieldOffset fieldWidth
16 bit :
    rf16 addr fieldOffset fieldWidth
8 bit :
    rf8 addr fieldOffset fieldWidth

Where "fieldOffset" means the bit offset within the register
(0 <= fieldOffset < register width).
"fieldWidth" means the width of the required field in bits.
(1 <= fieldWidth < (register width - fieldOffset))
All strings are separated by a space.
All values can be either decimal (no prefix) or hexadecimal (prefix: 0x).

- **Poll for Read Element**

For polling data different access types are allowed,

32 bit :
    p addr mask mask count
16 bit :
    p16 addr mask mask count
8 bit :
    p8 addr mask mask count

All strings are separated by a space.
All values can either be entered decimal (no prefix) or hexadecimal (prefix: 0x).
For more details to the *Poll for Read Element* please refer to the ***Register Sequencer*** chapter.

- **Poll for Target Element**

For polling data different access types are allowed,

32 bit :
    p addr target mask count
16 bit :
    p16 addr target mask count
8 bit :
    p8 addr target mask count

All strings are separated by a space.
All values can either be entered decimal (no prefix) or hexadecimal (prefix: 0x).
For more details to the *Poll for Read Element* please refer to the ***Register Sequencer*** chapter.

- **Write Repeat Element**

For writing data multiple times to the same address different access types are allowed,

32 bit :

f addr data count

16 bit :

f16 addr data count

8 bit :

f8 addr data count

All strings are separated by a space.
All values can either be entered decimal (no prefix) or hexadecimal (prefix: 0x).
For more details to the *Write Repeat Element* please refer to the ***Register Sequencer*** chapter.

- **Write Repeat Increment**

For writing data multiple times to an address with autoincrement the following access types are allowed,

32 bit :

a addr data count

16 bit :

a16 addr data count

8 bit :

a8 addr data count

All strings are separated by a space.
All values can either be entered decimal (no prefix) or hexadecimal (prefix: 0x).
For more details to the *Write Repeat Increment Element* please refer to the ***Register Sequencer*** chapter.

- **Label**

A Label element can be set but does not have a direct special function.
It can only be the destination for jump loops.
For more information refer to the Jump.

- **Jump Element**

The jump command evaluates the resulting Value of the action element immediately before the jump element itself.
When the value given on the jump was true then the jump will be executed.

jmpxx value maxretry

Where "value" is the value which will be compared to the previous value result.
"maxretry" is the maximum number of loops that are allowed before stopping the loop and was meant as escape possibility.

The following jump types are allowed,

jmpe
Jump if Equal
The jump will be performed when the last operation value is equal to the jump value.
jmpne
Jump if Not Equal
The jump will be performed when the last operation value is not equal to the jump value.
jmpg

Jump if Greater
The jump will be performed when the last operation value is greater than the jump value.
jmpge
Jump if Greater or Equal
The jump will be performed when the last operation value is greater or equal to the jump value.
jmpl
Jump if Less
The jump will be performed when the last operation value is less than the jump value.
jmple
Jump if Less or Equal
The jump will be performed when the last operation value is less or equal to the jump value.
jmps
Jump on Success
The jump will be performed when the last operation was successful.
jmpwoe
Jump if Warning Or Error
The jump will be performed when the last operation returned either with a warning or failed.

- **Software Delay Element**

To insert a software delay.

d microseconds

All strings are separated by a space.
All values can either be entered decimal (no prefix) or hexadecimal (prefix: 0x).
For more details to the *Software Delay Element* please refer to the ***Register Sequencer*** chapter.

- **Indigo Special Command Sequencer Elements**

VSync Element
swivsync

This element should only be used for Indigo par files.
For more details to the *Indigo VSync Element* please refer to the ***Register Sequencer*** chapter.

- **Indigo2 Special Command Sequencer Elements**

Wait Element
i2wait cycles

Software Interrupt Element
i2swint

Label Element
i2label

Loop Element
i2loop

Jump Element

i2jump addr

Jump Relative Element

i2jumpr offset

Watchdog Reset

i2wdr

Watchdog Set

i2wds divider counter

Write Element

i2write size addr data

Data Register Get Element

i2drget size addr

Data Register Put Element

i2drput size addr

Data Register And Element

i2drand data

Data Register Or Element

i2dror data

Data Register Invert Element

i2drinvert

Data Register Shift Left Element

i2drshiftl value

Data Register Shift Right Element

i2drshiftr value

Data Register Add Element

i2dradd data

Data Register Check Element

i2drcheck data

Address Register Get Element

i2arget addr

Address Register Get Indirect Element

i2argetindirect size

Address Register Put Indirect Element

i2arputindirect size

End Element

i2end

These elements should only be used for Indigo2 par files.
For more details to the please refer to the ***Register Sequencer*** chapter.

- **End**

  To signalize the end of the valid par file section.

  e

XIII. Release Notes

Attention :

1. **Please ensure that the software protection dongle is removed while installation**

2. **Connection Device Information**

   a. Totalphase / Aardvark SPI
   To ensure that the Aardvark Device is working properly
   it is required to update the **Firmware** of the Aardvark Device
   to Version **3.50**, or higher – compatible versions only.
   Furthermore it is necessary to install the latest **USB drivers**
   which are **v2.10, or later**.

   To update the firmware as well as the latest drivers
   please visit the Homepage of Total Phase, Inc. under
   www.totalphase.com and download the required software.

   It has been detected that the transfer rate of the Aardvark Device
   can get about 3 times faster when connecting the Aardvark over a
   powered USB Hub to the PC.

   b. Segger / JLINK JTAG
   To ensure that JLINK Device is working properly
   please ensure using the latest **Driver v4.32 or higher**.

   To update to the latest drivers please visit the Homepage
   of SEGGER Microcontroller GmbH under
   www.segger.com and download the required software.

   c. Future Technology Device International, FT4232 SPI Device
   To ensure that the FTDI SPI Device is working properly
   please ensure using the latest **Driver v2.08.14 WHQL** or higher.

   To update to the latest drivers please visit the Homepage at
   www.ftdichip.com and download the required software.

Version : 0.9.4.0
Date : 15-Mar-13

Added Functionality :

- **General :**
  + In case of the Basic Edition – Standard Flashing will automatically being enabled

+ Additional Fujitsu Developer Suite documentation added, easy accessible via Register Debugger -> Documentation Tab

- **Indigo2**
  + High Speed Flashing - Advanced Flasher Plugin implemented
  + Panel Manager
    TCON Driver programming added with fixed values to simplify panel setup
    RSDS Panel support implemented
    Test Image Generator will now also save SRAM content into the .par file
    Test Image Generator extended for new features
    Text Size of Image Generator adjusted to completely fit into SRAM
  + Project Files and Register Descriptions updated
  + Flashing changed from problematic ECC Off Mode to ECC On Mode
  + Initialization sequence extended, raise clock settings, E2IP unlock
  + Menu Item, "Settings->Suppress default chip initialization" is now also available for accessing via SPI/Aardvark device
  + strange lock/unlock behavior will now also being considered by the initialization sequence - if already unlocked no further unlock will be done
  + new sample sequences added demonstrating safer locking/unlocking chip by Command Sequencer
  Attention: Ongoing - not finished.
- **Triton :**
  + New chip available with different connection types
  + Flash Access improved for Serial Flash
  Attention: Ongoing - not finished.
- **Emerald :**
  + Flash Access improved for Serial Flash
- **ApCo :**
  + New chip available with different connection types
  Attention: Ongoing - not finished.

Fixed Issues :

- Indigo2 :
  + Command Sequencer Execution Mode corrected for SPI/Aardvark connection type
  + Mapbit arrangement corrected which could lead to wrong colors in specific situations

Known Issues :

- Indigo2 :
  + Writing the Lock/Unlock Register with an unexpected value will lead to a transmission error on SPI/Aardvark connection only

----

Version : 0.9.3.0
Date : 11-Jan-13

Added Functionality :

General

- User Comments possible on Components/Registers and Fields in the Register Debugger.
- User Comments possible on Register Sequences, Sequence Items and Stacker Items/Subitems.
- Indigo2 :
  + Updating Register and Field information
  + Flashing implemented, SPI/Aardvark, SPI/FTDI and Ethernet/E2IP
  + Command Sequence Interpreter implemented with direct storage option
  + Binary to Command Sequence Analyzer implemented for interpreting flash content
  Attention: Ongoing - not finished.

- Triton :
  + New chip available with different connection types
  Attention: Ongoing - not finished.

Fixed Issues :

Known Issues :

Version : 0.9.2.0
Date : 13-Jul-12

Added Functionality :

General

- Saving data/binary content in .mhx format is now possible
- Checking and downloading latest versions from Fujitsu Homepage is now possible over the Help Menu item "Check for Updates"
- The Register Sequencer now contains a "label" and a "jump" element to allow simple loops in the sequence with different jump options.
- Additional Register Sequencer Element added - Read Field
- The Register Debugger contains the functionality to Save the current Register Set.
  This is helpful for comparing the stored set with e.g. a new loaded register set.
- Simple search mechanism for scanning all components for specific register and field names - different search modes
- Memory and Flash Dump Page now contains a context menu on the Items box allowing to directly select some predefined sizes
- Indigo2
  + Basic support for Command Sequencer implemented
  + Command List Generator implemented
  + new connection type Ethernet/E2IP available
  + new connection type SPI/FTDI with Advanced Protocol available
  Attention: Ongoing - not finished.

Fixed Issues :

- Register entries are fixed, extended or added for the following chips,
  EmeraldL / P
  Indigo2
  JadeD
- Advanced Flasher:
  + reading sector content with a length not matching the sector size leads to reading 0 values at the end -> fixed
  + erasing flash can be stopped by a too short timeout -> fixed
  + sector number and sizes will be evaluated dynamically instead of fixed table
  + automatic detection and initialization of 32/16 bit flash types
  + initialization optimization and reduction

Version : 0.9.1.0
Date : 16-Mar-12

Added Functionality :

General

- Comparing of a Memory Block with the content located at a specified address is now possible in the Memory / Flash Dump and the Memory / Flash Editor.
  (Previously this was limited to Flash functionality)
- The Register Sequencer now contains an "compare with auto increment" element allowing to compare regions of memory for a specific value.
  This is intended e.g. for Memory Tests in combination with the write repeat increment element
- The register debugger now contains special icons displaying lock/unlock key registers as well as lock/unlock status registers
- Suppressing chip initialization when connecting to target is now possible by selecting an item in the Settings menu - currently not supported by all chips
- Some special pages are integrated for supporting the user :
  a. Emerald - Auto Update Page -> to comfortable updating the Fujitsu Linux BSP
  b. Display / Panel Manager -> to comfortable adjusting the display / panel timing
  Both are currently only available for the Emerald chip.
  For more information please refer to the "How to ..." section.
-

Fixed Issues :
-

Known Issues :

Version : 0.9.0.8
Date : 12-Dec-11

Added Functionality :
General
- Updating User Manuals and Application Notes for all supported Chip Designs

Fixed Issues :
- Emerald Sector Table of Advanced Flash Programmer extended

Known Issues :

Version : 0.9.0.7
Date : 09-Dec-11

Added Functionality :
General
- Startup Warnings/Errors can be disabled in the Settings Menu (e.g. Font Warnings)
- New setting available to prefer Standard Flashing on startup instead of Advanced Flashing
- Current active Flashing Program will be displayed (Standard / Advanced)
- Current selected Flash Type (Advanced Flashing) will be displayed in the Support Bar

Fixed Issues :
- Advanced Flashing Program failed on initialization -> fixed
- Sector Table on Advanced Flashing contains wrong startup values -> fixed

----

Version : 0.9.0.6

Date **: 11-Nov-11**

Added Functionality :

- **High speed flashing** implemented for EmeraldL ES2 and EmeraldP
  Remark : Only available for access over JTAG (Segger)
- **Auto project scanner** implemented
  Scanning, detecting and displaying all available projects in the main directory.
  Easy selecting via toolbar or menu bar.
- **Progress bar**
  Flashing content will now be visualized with a progress bar.
  Sector - from … to … progress.

Version **: 0.9.0.5**
Date **: 06-Oct-11**

Added Functionality :
EmeraldL ES2 support

Version **: 0.9.0.4**
Date **: 21-Sep-11**

Added Functionality :
General

- Improved Flash Modes implemented
- Optimizing internal Memory Management
- Register Sequences corrected for EmeraldL and EmeraldP
- Init Sequences for EmeraldL and EmeraldP optimized

Version **: 0.9.0.3**
Date **: 02-Sep-11**

Added Functionality :
General

- Minimize, Maximize and Context Sensitive Help Button moved
- Snapshot implemented to extract an image from the current application view
- Application Exit implemented in the File Menu
- Stopwatch implemented starting automatically on time consuming hardware actions
- Memory Editor / Dump
  Now contains separate sector selection combo box allowing to select a sector directly.
  By doing this the sector start address will automatically beein entered into the address bar.
- 4 instances can be opened in parallel
- 64 Bit OS supported

EmeraldL / EmeraldP

- JTAG support for Segger JTAG JLINK device integrated.
- SPI support for FTDI device integrated.

Fixed Issues :
Smaller fixes done.

Known Issues :

Version : 0.9.0.2
Date : 01-July-11

Added Functionality :

General

- Sequence Stacker
  New main feature added which allows stacking created sequences to a sequence stack.
  This sequence stack can be used to combine sequences to more complex operations.
  Furthermore debug support on sequence level is available with breakpoints, single step, play, play to breakpoint, ...
  Detailed error reporting as well as arranging sequences within the stack is also possible.
- Documentation Page
  Page displayed on the Register Debugger offering latest manuals, application notes ... to the corresponding chip.
- Register Sequencer
  New User Defined item added allowing to write a register field instead of the whole register.
  Be careful, Register must be Read and Writeable.

Fixed Issues :

Smaller fixes done.

Known Issues :

Version : 0.9.0.1
Date : 02-May-11

Added Functionality :

General

Initial Version

Fixed Issues :

Known Issues :
