



HiChannel
User Guide

Issue	00B01
Date	2014-05-15

Copyright © HiSilicon Technologies Co., Ltd. 2014. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

Trademarks and Permissions



HISILICON, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

HiSilicon Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <http://www.hisilicon.com>

Email: support@hisilicon.com



About This Document

Purpose

This document describes how to use the channel debugging tool HiChannel.

Related Versions

The following table lists the product versions related to this document.

Product Name	Version
Hi3712	V100
Hi3136	V100
Hi3137	V100


Intended Audience

This document is intended for:





- Technical support engineers
- Software development engineers
- Chip development engineers

Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 DANGER	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.



Symbol	Description
 WARNING	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
 CAUTION	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 TIP	Provides a tip that may help you solve a problem or save time.
 NOTE	Provides additional information to emphasize or supplement important points in the main text.

Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

Issue 00B01 (2014-05-15)

This issue is the first draft release.



Contents

About This Document.....	iii
1 Overview.....	1
1.1 Introduction to the HiChannel.....	1
1.2 Preparing the Environment.....	1
1.3 Installing the USB Driver.....	2
2 Main GUI.....	5
3 Menus and Toolbar.....	8
3.2 Menu Items	9
3.2.1 File	9
3.2.2 Script.....	10
3.2.3 Setting	11
3.3 Tools.....	12
3.3.1 Clearing Chip Information	12
3.3.2 Connection Manager	12
4 Main Functions.....	15
4.1 Register	15
4.1.1 Segment Information Area	16
4.1.2 Register Information Area.....	17
4.1.3 Shortcut Menu in the Register Information Area	18
4.2 Button.....	20
4.3 Indicator	21
4.4 Trace.....	22
4.5 Monitor.....	23
4.6 Universal Read/Write	24
4.7 I2C Set.....	25
4.8 Log	27
5 Configuration File.....	28
5.1 Global.....	28
5.2 Button.....	29
5.3 Indicator	29
5.4 Trace.....	30



5.5 Monitor.....	30
5.6 I2C Set.....	31
5.7 Advance.....	31
5.8 Segment.....	31
5.9 Bit Register.....	32
5.10 Word Register.....	32
6 C Script.....	34
6.1 Dynamic Library Interfaces	34
6.1.2 Requirements on Return Values of Functions Bound to Controls.....	36
6.1.3 Prerequisite	38
6.1.4 I2C Read/Write	38
6.2 Description of Script Functions.....	40
6.3 Instance	40
7 Advanced Functions	44
7.1 Error Rate Monitor.....	44
7.2 C/N Monitor	45
7.3 Constellation View	46
7.4 Spectrum View	47
7.5 Time Domain View	48
7.6 Data File Format	51
7.6.1 Conventions	51
7.6.2 Instance	51
8 Functions of the Customer edition	52
8.1 Indicator	52
8.2 Button.....	52
8.3 Trace.....	53
8.4 Monitor.....	53
8.5 Segment.....	54
8.6 Operation Procedure.....	55
8.7 Spectrum View	56
8.8 Constellation View	57
8.9 Time Domain View	58
9 Precautions	60
10 FAQs	61
10.2 What Do I Do If "Invalid Word Register content : bit count error at line xx" Is Displayed When the Configuration File Is Being Loaded?	61
10.3 What Do I Do If "Segment name in the file is not exist in the register area" Is Displayed After Write by file Is Selected from the Shortcut Menu of the Register Pane?	62
10.4 What Do I Do If "please check register type at first line" Is Displayed After Write by file Is Selected from the Shortcut Menu of the Register Pane?	62



10.5 What Do I Do If the Register List Is Not Read or Is Read Incompletely When the Register Type Is Word Register But No Error Information Is Displayed When Write by file Is Selected?	63
--	----



Figures

Figure 1-1 Opening the driver file.....	2
Figure 1-2 Installing the driver.....	3
Figure 1-3 Information indicating that the driver is successfully installed.....	3
Figure 1-4 Checking the Device Manager.....	4
Figure 2-1 HiChannel GUI.....	5
Figure 3-1 Menu items of the HiChannel professional edition	8
Figure 3-2 Menu items of the HiChannel customer edition	8
Figure 3-3 File menu	9
Figure 3-4 Configuring export information.....	10
Figure 3-5 Advance menu	10
Figure 3-6 Script menu.....	11
Figure 3-7 Setting menu.....	11
Figure 3-8 Changing Timer period.....	11
Figure 3-9 Clearing chip information.....	12
Figure 3-10 Setting the DUT network.....	12
Figure 3-11 Setting the USB connection	13
Figure 3-12 Setting the command for starting the board program.....	14
Figure 4-1 Register pane	16
Figure 4-2 Selecting a segment	16
Figure 4-3 Adding a segment	17
Figure 4-4 Modifying a segment	17
Figure 4-5 Delete confirmation	17
Figure 4-6 Register information area.....	18
Figure 4-7 Editing a register value	18
Figure 4-8 Shortcut menu.....	19
Figure 4-9 Adding a register.....	19



Figure 4-10 Button pane.....	20
Figure 4-11 Configuring a button.....	20
Figure 4-12 Setting parameters	21
Figure 4-13 Indicator pane	22
Figure 4-14 Setting an indicator	22
Figure 4-15 Trace pane.....	23
Figure 4-16 Trace Event Set.....	23
Figure 4-17 Monitor pane.....	23
Figure 4-18 Selecting a function	24
Figure 4-19 Universal Read Write pane	25
Figure 4-20 I2C Set pane.....	26
Figure 4-21 Specifying the function.....	26
Figure 4-22 Log pane	27
Figure 7-1 Error Rate Monitor	45
Figure 7-2 CN Monitor.....	46
Figure 7-3 Constellation View.....	47
Figure 7-4 Spectrum View.....	48
Figure 7-5 Time Domain View.....	49
Figure 8-1 Indicator.....	52
Figure 8-2 Button	53
Figure 8-3 Trace	53
Figure 8-4 Monitor	54
Figure 8-5 Register.....	54
Figure 8-6 Overall GUI	56
Figure 8-7 Spectrum view	57
Figure 8-8 Constellation view	58
Figure 8-9 Time domain view	58



1 Overview

1.1 Introduction to the HiChannel

The HiChannel is software on the PC for debugging the HiSilicon channel chips. It is divided into the customer edition and professional edition.

The customer edition is designed for customers and technical support engineers of HiSilicon. It has the following functions:

- Supports connection to the target board over the Ethernet or USB port.
- Allows you to monitor the chip running status by using the **Indicator**, **Trace**, and **Monitor** panes.
- Allows you to check and edit registers of channel chips, facilitating channel fault identification.
- Allows you to observe the bit error rate (BER), carrier-to-noise ratio (C/N), constellation, spectrum, and time domain in graphics.

The professional edition is dedicated for the HiSilicon development engineers. It has the following functions:

- Supports connection to the target board over the Ethernet or USB port.
- Supports the development and debugging of application-specific integrated circuit (ASIC) chips or field-programmable gate array (FPGA) that contains channels.
- Allows you to customize the operations corresponding to controls in the **Button**, **Indicator**, **Trace**, and **Monitor** panes by writing C scripts based on the target chip.
- Allows you to create the configuration file and C script dedicated for the target chip (or FPGA) quickly and load the files at any time.
- Allows you to check and edit registers of channel chips.
- Allows you to customize the mode for checking and editing registers.
- Allows you to observe the BER, C/N, constellation, spectrum, and time domain in graphics.

1.2 Preparing the Environment

Before you use the HiChannel, perform the following steps:



Step 1 Copy **HiTool-X.X.X.zip** and **jre-6u1-windows-i586-p-s.rar** (in `SDK_DIR/tools/windows/HiTool[jre]`) in the SDK to a local disk drive on a PC that runs Windows 7 or Windows XP.

JRE 1.6 or later must be preinstalled. For the Windows OS, **jre-6u1-windows-i586-p-s.rar** must be installed; otherwise, the program cannot run.

Step 2 Decompress **HiTool-X.X.X.zip**, and double-click **HiTool_vX_X_X.exe**.

Step 3 Connect the network cable to the board, and configure the IP address for the board. Ensure that the PC that runs the HiChannel can be pinged on the board. If the USB-to-I²C adapter is used, connect the USB port of the adapter to that of the PC, and connect the I²C port of the adapter to that of the target board. Before using the USB-to-I²C adapter, you need to install the USB driver. For details, see section 1.3 "Installing the USB Driver."

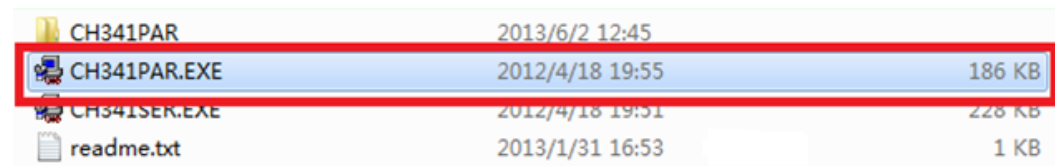
----End

1.3 Installing the USB Driver

The HiChannel enables direct access to the I²C interface of the target board over the USB port of the PC by using the USB-to-I²C adapter. The decoding chip, serial port, and Ethernet port are not required, which facilitates the debugging of the board. Before using the USB-to-I²C adapter, install the driver on the PC by performing the following steps:

Step 1 Double-click **CH341PAR.EXE** in the toolkit, as shown in Figure 1-1.

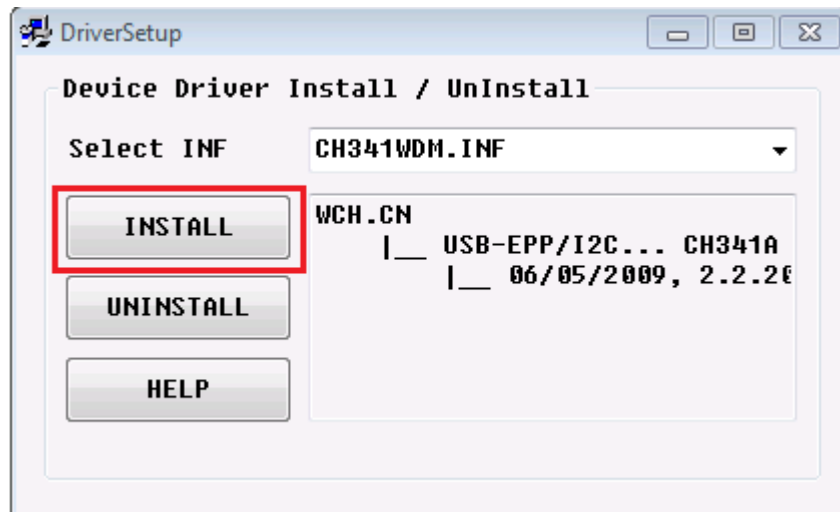
Figure 1-1 Opening the driver file



Step 2 Click **INSTALL**.

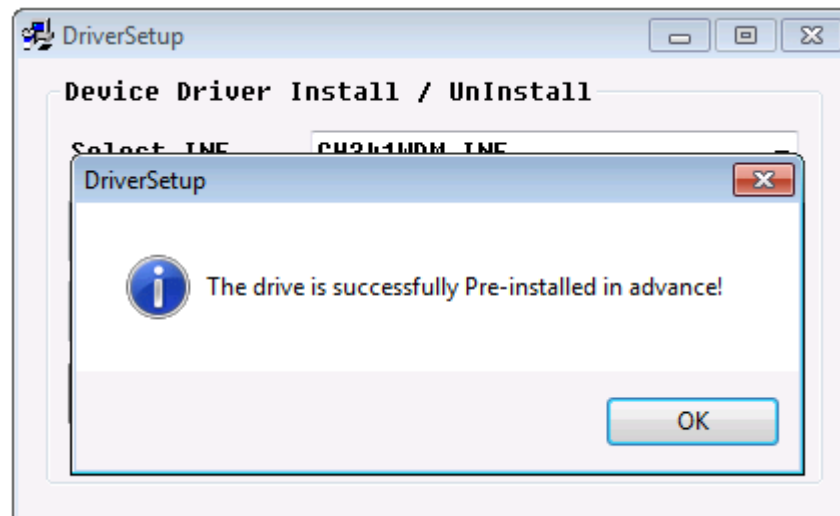


Figure 1-2 Installing the driver



Step 3 Wait until the installation is complete. The information indicating that the driver is successfully installed is displayed, as shown in [Figure 1-3](#).

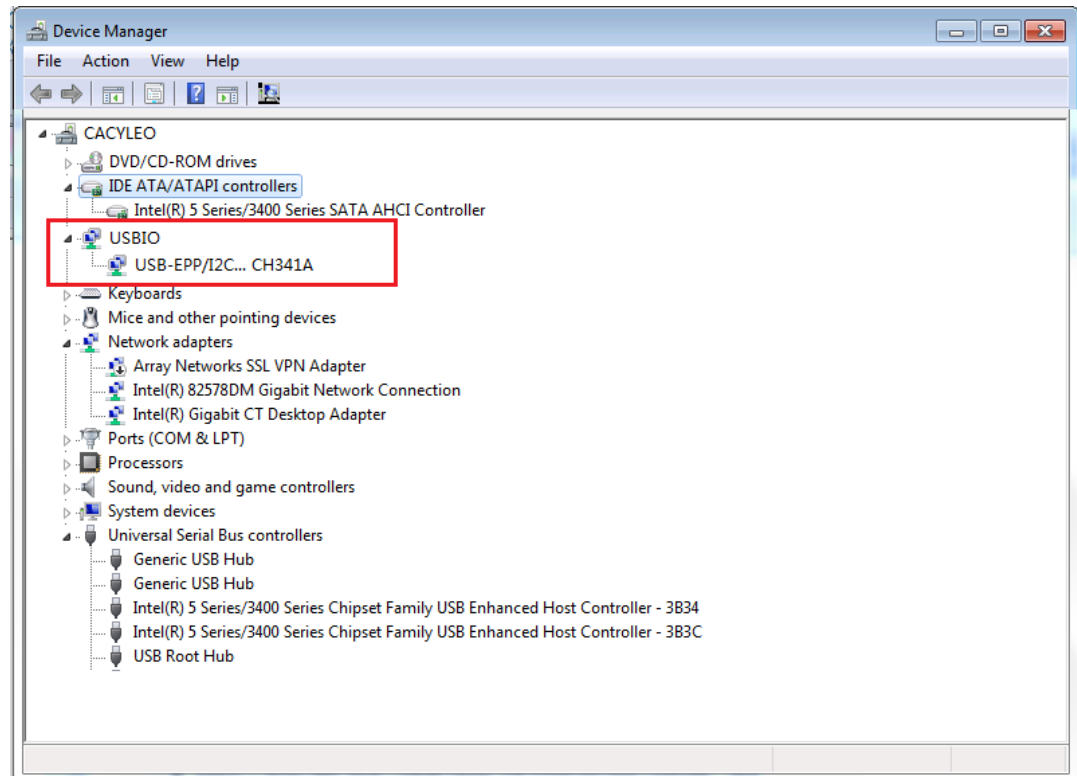
Figure 1-3 Information indicating that the driver is successfully installed



Step 4 Connect the USB-to-I²C adapter to the PC, and open **Device Manager** to check whether **USB-EPP/I2C...CH341A** is displayed under **USBIO**. If yes, the driver is successfully installed and can be used.



Figure 1-4 Checking the Device Manager



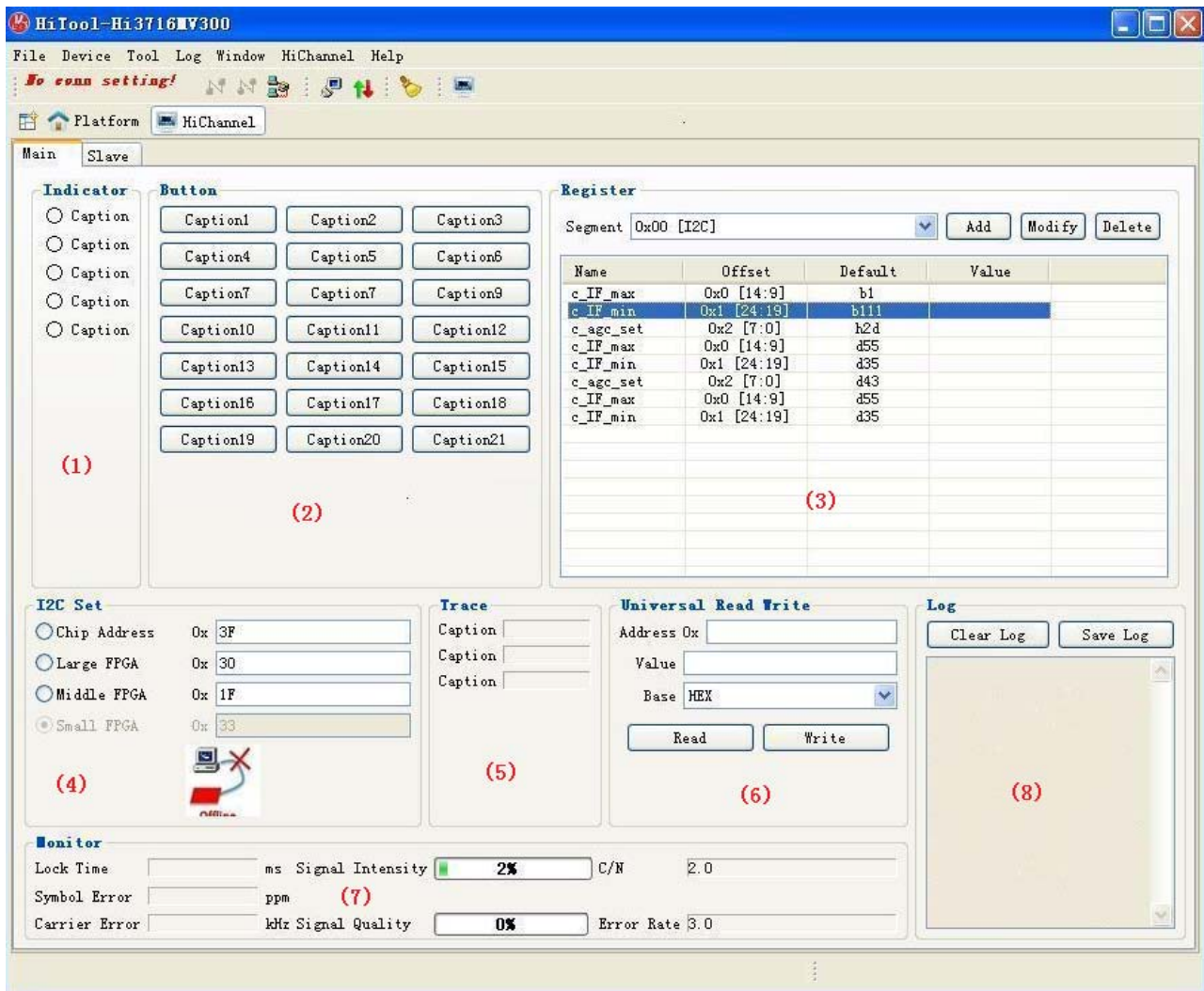
----End



2 Main GUI

Figure 2-1 shows the main GUI of the HiChannel.

Figure 2-1 HiChannel GUI





The main GUI includes the **Main** and **Slave** tab pages, and the layout of these two pages are the same. Typically the **Main** tab page is used to configure and debug chips, large FPGA, and medium FPGA, and the **Slave** tab page is used to configure and debug the small FPGA (which implements additional functions for the FPGA).



NOTE

The FPGA-related functions are designed for HiSilicon development engineers and can be ignored by other users. The large, medium, and small FPGAs on the FPGA platform each has its own component address. The register addresses for the large and medium FPGAs are allocated in a unified manner, that is, duplicate addresses are not allowed. The configuration files for the two tab pages are independent of each other.

The tab pages consist of the following panes:

- **Indicator**
See area (1) in [Figure 2-1](#). The HiChannel periodically calls the C script functions, obtains the return values of the functions, and dynamically changes the colors of the indicators according to the return values.
- **Button**
See area (2) in [Figure 2-1](#). Clicking a button in the **Button** pane triggers the execution of a C script function. Mapping between buttons and executed C script functions can be configured by users of the professional edition.
- **Register**
See area (3) in [Figure 2-1](#). The **Register** pane is used to add, modify, or delete register partitions and registers, and save and import register files by using the buttons and shortcut menu.
- **I2C set**
See area (4) in [Figure 2-1](#). The **I2C Set** pane is used to set I²C addresses for components and display the I²C bus status in real time.
- **Trace**
See area (5) in [Figure 2-1](#). The HiChannel periodically calls the C script functions, obtains the return values of the functions, and displays the values in the text box of the **Trace** pane.
- **Universal Read Write**
See area (6) in [Figure 2-1](#). You can read/write values from/to registers at specific addresses in the **Universal Read Write** pane.
- **Monitor**
See area (7) in [Figure 2-1](#). The HiChannel periodically calls the C script functions, obtains the return values of the functions, and displays the values in the text box and progress bar of the **Monitor** pane. The **Monitor** pane displays the following monitoring information:
 - C/N
 - Error Rate
 - Signal Intensity
 - Signal Quality
 - Lock Time
 - Symbol Error
 - Carrier Error
- **Log**



See area (8) in [Figure 2-1](#). The **Log** pane displays operation-related information and stores or clears logs.

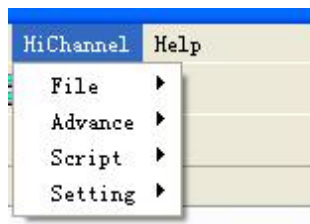


3 Menus and Toolbar

Menu items of the HiChannel are in the **HiChannel** menu of the menu bar.

The professional edition contains the **File**, **Advance**, **Script**, and **Setting** menu items, as shown in [Figure 3-1](#).

Figure 3-1 Menu items of the HiChannel professional edition





The customer edition contains the **Advance** and **Setting** menu items, as shown in [Figure 3-2](#).

Figure 3-2 Menu items of the HiChannel customer edition



The following two shortcut buttons in the toolbar are dedicated for the HiChannel:

-  : clears chip information (visible in only the professional edition)
-  : connection manager

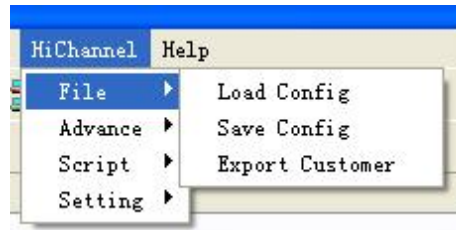


3.2 Menu Items

3.2.1 File

The **File** menu is used to operate configuration file and is visible in only the professional edition. See [Figure 3-3](#).

Figure 3-3 File menu



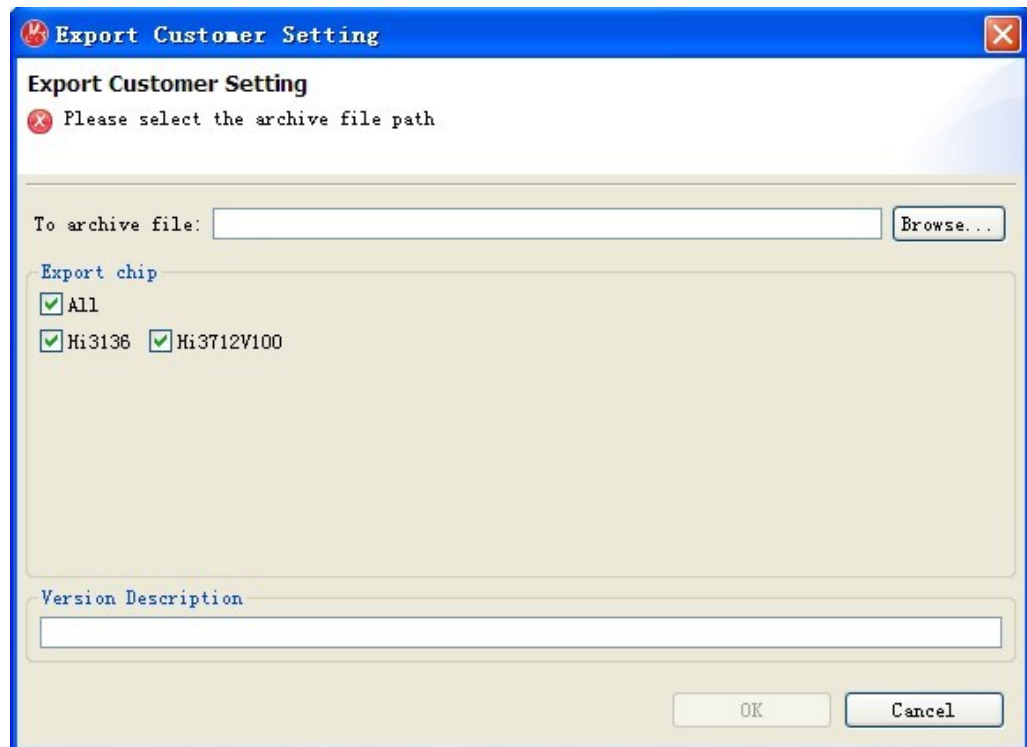
- **Load Config**
Loads the configuration file. This item is used when the configuration file needs to be updated manually or switched. The **Main** and **Slave** tab pages have separate configuration files (the formats are the same).
 - If a configuration file is loaded on the **Main** tab page, all fields in the configuration file take effect.
 - If a configuration file is loaded in the **Slave** tab page, the **Global**, **Advance**, and **Setting** fields do not take effect.



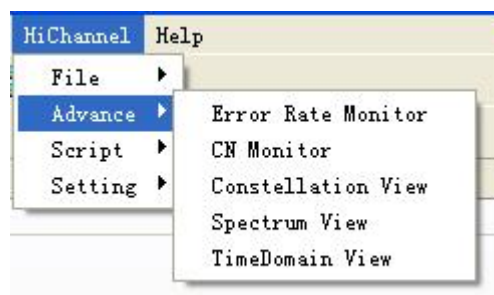
NOTE

It is highly recommended that the postfixes **_m** and **_s** be added to the configuration files for the **Main** and **Slave** tab pages respectively to facilitate identification.

- **Save Config**
Saves the configuration file. A configuration file is generated according to the current configurations, which contains configuration information on only the current page.
- **Export Customer**
Exports the HiChannel package of the customer edition. You can use this package to create the HiChannel of the customer edition by using the HiPublisher. The configuration files for multiple chips can be exported to the same tool package.
After choosing **Export Customer**, specify the save path, select the chips, enter description information (optional), and click **OK**. See [Figure 3-4](#).

Figure 3-4 Configuring export information

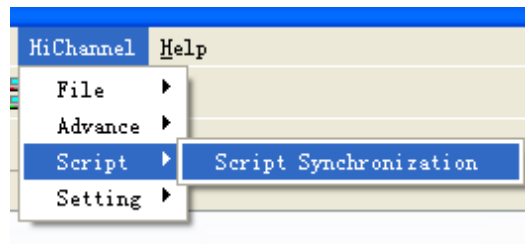
- **Advance**
Implements some advanced display functions, including displaying the BER, C/N, constellation, spectrum, and time domain information in graphics.
See [Figure 3-5](#). For details about this menu, see chapter 7 "[Advanced Functions](#)."

Figure 3-5 Advance menu

3.2.2 Script

The **Script** menu item is used to synchronize the script file. It is visible in only the professional edition. See [Figure 3-6](#).

Figure 3-6 Script menu

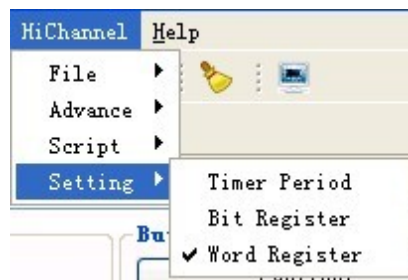


When the script file is modified externally, choosing **Script Synchronization** loads the script file again.

3.2.3 Setting

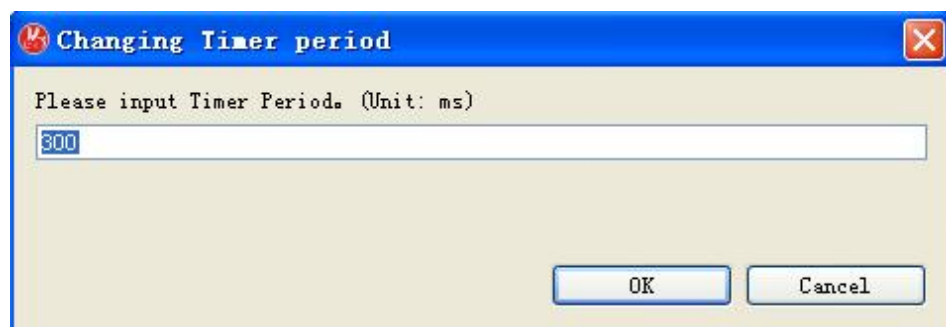
The **Setting** menu is used to configure the HiChannel. See [Figure 3-7](#).

Figure 3-7 Setting menu



- **Timer Period**
Time Period is used to set the automatic refresh cycle for the **Indicator**, **Trace**, and **Monitor** panes. The unit is ms. See [Figure 3-8](#).

Figure 3-8 Changing Timer period



- **Bit Register**
Registers in the register information area are defined, displayed, and operated by bit. When **Bit Register** is selected, ✓ is displayed close to **Bit Register**. You can select only one of **Bit Register** and **Word Register**.

- Word Register

Registers in the register information area are defined, displayed, and operated by word (or byte if the register bit width is 8 bits). When **Word Register** is selected, ✓ is displayed close to **Word Register**. You can select only one of **Bit Register** and **Word Register**. This option applies to the register list file generated for the driver.

3.3 Tools

3.3.1 Clearing Chip Information


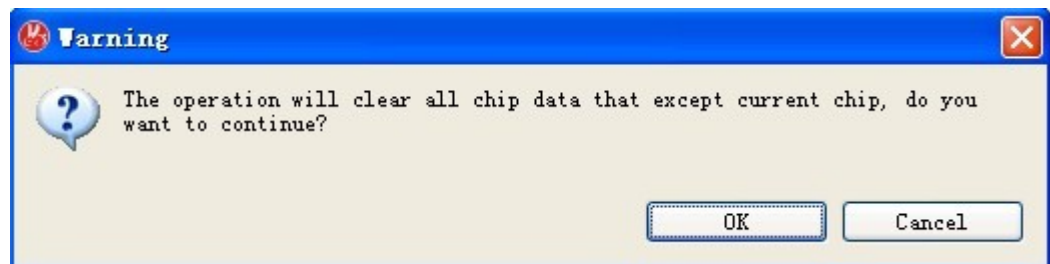

This function is used to clear the configuration information of all switched chips except the current chip. It is available in only the professional edition. Click , and click OK in the displayed dialog box. Then the related data is cleared. See [Figure 3-9](#).

Figure 3-9 Clearing chip information

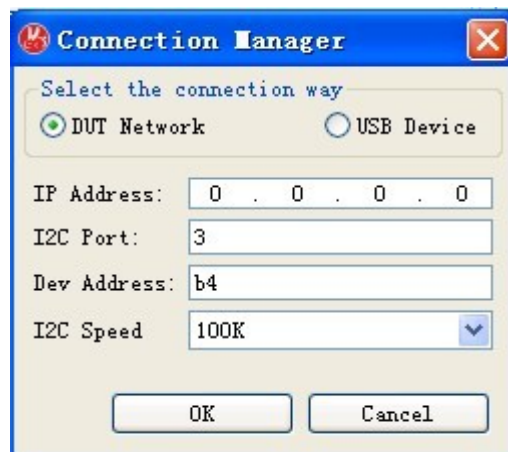


3.3.2 Connection Manager

The connection manager is used to set the DUT network or the USB connection. Click  on the toolbar to open the **Connection Manager** dialog box, select **DUT Network or USB Device**, and then set the parameters accordingly.

- Set the DUT network, as shown in [Figure 3-10](#).

Figure 3-10 Setting the DUT network



- **IP Address:** IP address for the target board
- **I2C Port:** port ID of the I²C component to be debugged on the target board
- **Dev Address:** address for the I²C component to be debugged on the target board
- **I2C Speed:** speed of the I²C component to be debugged on the target board, in kHz
- Set the USB connection, including the USB speed and device address. See [Figure 3-11](#).

Figure 3-11 Setting the USB connection



- **I2C Speed:** speed of the I²C component to be debugged on the target board, in kHz
- **Dev Address:** address for the I²C component to be debugged on the target board

After setting the parameters, click **OK**. Then the HiChannel interacts with the target board based on the configuration information. If the connection fails, an error message is displayed; if the connection is successfully set up, the HiChannel updates the chip status such as the signal intensity.



CAUTION

If the DUT network is used, the HiChannel sends a command for starting the board program first. This command can be configured in Preferences, and it is **soc_server** by default. See [Figure 3-12](#).



Figure 3-12 Setting the command for starting the board program





4 Main Functions

The main GUI of the HiChannel consists of the following panes:

- Register
- Button
- Indicator
- Trace
- Monitor
- Universal Read Write
- I2C Set
- Log

4.1 Register

[Figure 4-1](#) shows the **Register** pane, which is divided into the segment information area (area 1) and register information area (area 2). On the **Register** pane, you can add, modify, or delete register segments and registers, and save and import register files by using the buttons and shortcut menu. The display mode of registers in the **Register** pane is determined by **Bit Register** or **Word Register** in the **Setting** menu. [Figure 4-1](#) shows the **Bit Register** mode.

Figure 4-1 Register pane

[illegible]

4.1.1 Segment Information Area

Click the **Segment** drop-down list and select a segment (with the corresponding base address), as shown in [Figure 4-2](#). The registers in this segment are then displayed in the register information area. The last item in the drop-down list is **All**, indicating that all registers will be displayed.

Figure 4-2 Selecting a segment

Register

Segment 0x00 [I2C]

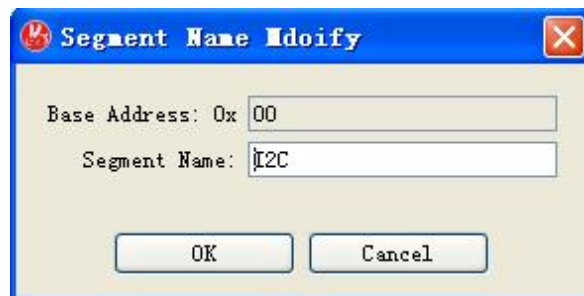
0x00 [I2C]
0x10 [MAN]

The **Add**, **Modify**, and **Delete** buttons in the segment information area allow you to add, modify, and delete segments. The **All** segment always exists (does not need to be added) and cannot be deleted. It corresponds to registers in all other segments.

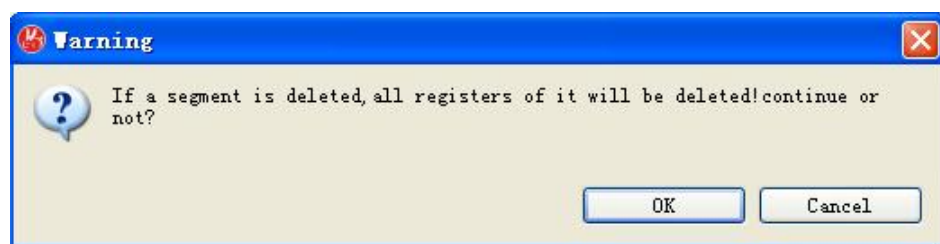
- **Add:** Clicking the **Add** button displays the dialog box shown in [Figure 4-3](#). Enter the base address and segment name, and click **OK**. A segment is added.

Figure 4-3 Adding a segment

- **Modify:** Clicking the **Modify** button displays the dialog box shown in [Figure 4-4](#). You can then modify the base address and segment name.

Figure 4-4 Modifying a segment

- **Delete:** Clicking the **Delete** button deletes the current segment and registers in the segment. To prevent a segment from being deleted by mistake, the system displays a confirmation message shown in [Figure 4-5](#).

Figure 4-5 Delete confirmation

4.1.2 Register Information Area

[Figure 4-6](#) shows the register information area, which contains the following information:

- **Name:** register name
- **Offset:** offset address for a register based on the segment address
- **Default:** default value
- **Value:** current value



The default value and current value are displayed in the specified mode, including the decimal, binary, and hexadecimal modes (identified by using the prefixes d, b, and h).

When **Segment** is set to **All**, all registers are listed, and the offset addresses become absolute addresses.

Figure 4-6 Register information area

Name	Offset	Default	Value
c_IF_max	0 [9:14]	55	0
c_IF_min	1 [19:24]	35	0
c_agc_set	2 [0:7]	43	26
c_IF_max	0 [9:14]	55	0
c_IF_min	1 [19:24]	35	0
c_agc_set	2 [0:7]	43	26
c_IF_max	0 [9:14]	55	0
c_IF_min	1 [19:24]	35	0

- To select a register, click the register row. The row turns blue, indicating that it is selected.
- To read a register, double-click the register row. The current values are displayed.
- To write to a register, click the **Value** column of the register, enter a value, and click anywhere but the current cell, as shown in [Figure 4-7](#).

Figure 4-7 Editing a register value

Name	Offset	Default	Value
c_IF_max	0 [9:14]	55	0
c_IF_min	1 [19:24]	35	0
c_agc_set	2 [0:7]	43	26
c_IF_max	0 [9:14]	55	0
c_IF_min	1 [19:24]	35	0
c_agc_set	2 [0:7]	43	26
c_IF_max	0 [9:14]	55	0
c_IF_min	1 [19:24]	35	0



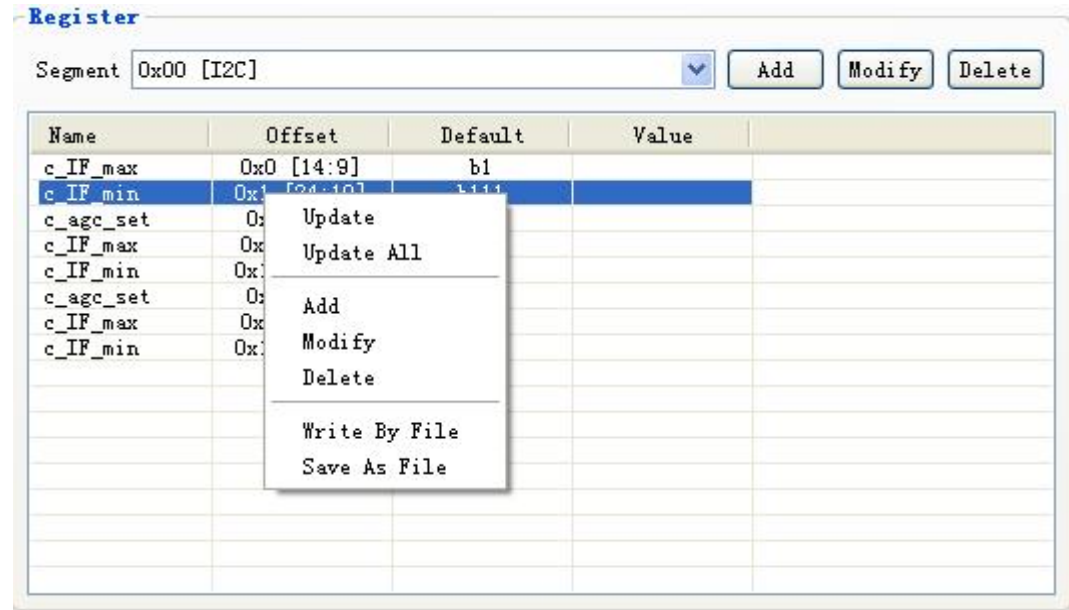
NOTE

Reading registers in this document indicates that the HiChannel reads register values from the target board and displays the values on its GUI.

4.1.3 Shortcut Menu in the Register Information Area

Select a register in the register information area. The register row turns blue. Right-click the register row. The shortcut menu shown in [Figure 4-8](#) is displayed. When **Segment** is set to **All**, the options **Add**, **Modify**, and **Delete** in the shortcut menu are unavailable (dimmed).

Figure 4-8 Shortcut menu



- **Update:** Reads the value of the selected register.
- **Update All:** Reads the values of all registers listed in the register information area.
- **Add:** Adds a register. Set **RegName**, **Offset** (hexadecimal), **Bit Range** (decimal, from high to low), **Radix** (hexadecimal), and **Default**, as shown in [Figure 4-9](#).

Figure 4-9 Adding a register



- **Modify:** Modifies a register. The displayed dialog box for modifying a register is similar to that shown in [Figure 4-9](#), except that the parameter values already exist.
- **Delete:** Deletes the selected register. There will be no confirmation message.

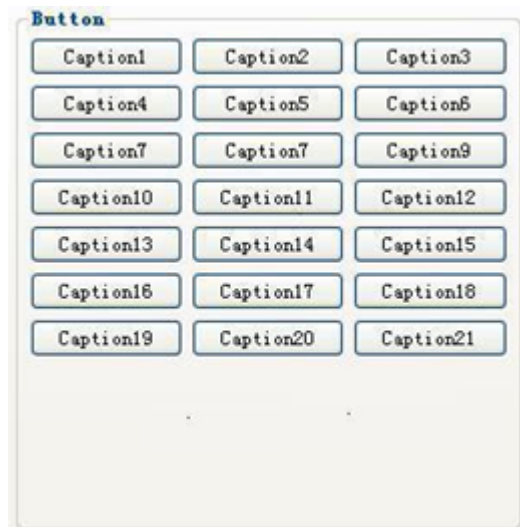


- **Save As File:** Saves the information in the current register information area to a file. The file can be specified.
- **Write by File:** Imports the register data in a file to the register information area.

4.2 Button

Figure 4-10 shows the **Button** pane.

Figure 4-10 Button pane



- For users of the customer edition, the function of each button in the **Button** pane is fixed. For details, see section 8.2 "Button."
- For users of the professional edition, the function of each button in the **Button** pane can be customized by configuring the corresponding C script function. Clicking the button implements the function defined in the corresponding C script function.

Right-clicking a button displays the dialog box shown in Figure 4-11. You can then configure the function of the button.

Figure 4-11 Configuring a button





- **Button:** button ID. You can configure different buttons by selecting them.
- **Caption:** character string to be displayed on the button (indicating the function of the button)
- **Function:** C script function, which can be selected from the drop-down list
- **ParamNum:** number of input parameters, 0 by default. If the value is 0, **Default** and **Remind** are unavailable.



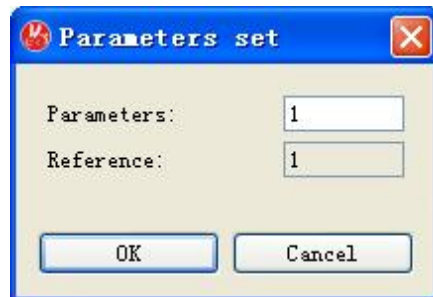
NOTE

The HiChannel does not parse the parameters. It only transmits the parameters in the format of character strings to the script function.

- **Default:** default values of input parameters, separated by using commas (,).
- **Remind:** input parameters
- **Show Return Value:** The return value of the C script function is displayed if this option is selected.
- **Uninstall:** uninstalls a specific button

Click a configured button, and enter parameters in the displayed dialog box if the parameter number is not 0. Separate multiple parameters by using commas (,). As shown in [Figure 4-12](#), the data displayed in the **Reference** text box is the character strings entered in the **Remind** text box when the button is configured. If the parameters have default values, the default values are also displayed.

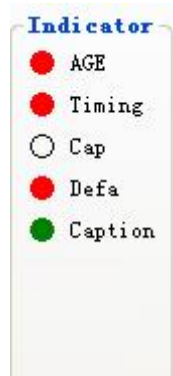
Figure 4-12 Setting parameters



4.3 Indicator

[Figure 4-13](#) shows the **Indicator** pane.

Figure 4-13 Indicator pane



- For users of the customer edition, the function of each indicator in the **Indicator** pane is fixed. For details, see section 8.1 "Indicator."
- For users of the professional edition, the function of each indicator in the **Indicator** pane can be customized by configuring the corresponding C script function. The system calls the C script function periodically and displays the indicator in different colors according to the return value of the function.

To set the function of an indicator, right-click the characters on the right of the indicator. A dialog box shown in Figure 4-14 is displayed.

Figure 4-14 Setting an indicator



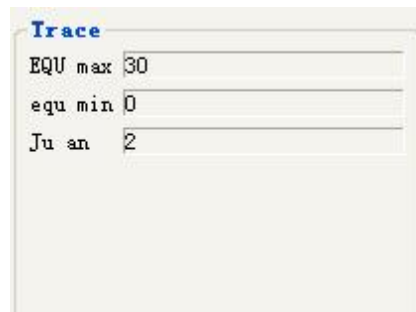
- **Lamp**: indicator ID. You can configure different indicators by selecting them.
- **Caption**: character string to be displayed on the right of the indicator (indicating the function of the indicator)
- **Lamp Mode**: display mode of the indicator. You can select **Gray+Green** (corresponding to value 0 and value greater than or equal to 1) or **Gray+Green+Red** (corresponding to values 0, 1, and value greater than or equal to 2).
- For details about others, see section 4.2 "Button."

4.4 Trace

Figure 4-15 shows the **Trace** pane.



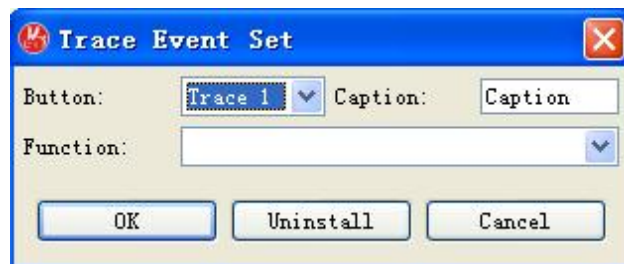
Figure 4-15 Trace pane



- For users of the customer edition, the function of each trace item is fixed. For details, see section 8.3 "Trace."
- For users of the professional edition, each trace item can be customized by configuring the corresponding C script function. The system calls the C script function periodically and displays the returned character strings in the **Trace** pane.

Right-clicking a text box (but not right on the text) in the **Trace** pane displays the dialog box shown in Figure 4-16. You can then set the trace function.

Figure 4-16 Trace Event Set



- **Trace**: trace text box ID. You can configure different text boxes by selecting them.
- **Caption**: character string to be displayed on the left of the text box (indicating the function of the trace item)
- For details about others, see section 4.2 "Button."

4.5 Monitor

Figure 4-17 shows the **Monitor** pane.

Figure 4-17 Monitor pane





- Lock Time
- Symbol Error
- Carrier Error
- Signal Intensity (0–100)
- Signal Quality (0–100)
- C/N
- Error Rate

For users of the customer edition, items in the **Monitor** pane are fixed. For details, see section [8.4 "Monitor."](#)

For users of the professional edition, each item in the **Monitor** pane can be customized by configuring the corresponding C script function. The system calls the C script function periodically and displays the returned character strings or progress.

Right-clicking the display box close to each monitoring item displays the dialog box shown in [Figure 4-18](#). You can select a C script function from the **Function** drop-down list.

Figure 4-18 Selecting a function



- **Function:** C script function, which can be selected from the drop-down list. If the bound function does not exist in the C script specified in the configuration file, the selected item is empty.
- **Uninstall:** leaves the bound function empty

4.6 Universal Read/Write

[Figure 4-19](#) shows the **Universal Read/Write** pane. **DEC**, **HEX**, and **BIN** in the **Radix** drop-down list indicate decimal, hexadecimal, and binary respectively.

Figure 4-19 Universal Read Write pane

The image shows a software window titled "Universal Read Write". It contains three input fields: "Address 0x" with the value "46", "Value" with the value "27", and "Radix" with a dropdown menu showing "HEX". Below these fields are two buttons: "Read" and "Write".

- **Read:** Enter the register address for the I²C component in the **Address** text box (hexadecimal), and click **Read**. The register value is read from the target board and is displayed in the **Value** text box based on the selected numeral system in **Radix**.
- **Write:** Enter the register address, enter a value in the **Value** text box based on the selected numeral system in **Radix**, and click **Write**.



CAUTION

- The address length cannot be greater than the value of **AddressBytes** in the configuration file.
- The length of the value in the **Value** text box is determined by **RegisterBytes** in the configuration file and the **Radix** value. If **Radix** is **HEX**, the value length cannot be greater than 2 x **RegisterBytes**; if **Radix** is **BIN**, the value length cannot be greater than 8 x **RegisterBytes**.

4.7 I2C Set

Figure 4-20 shows the **I2C Set** pane, which is used to set the I²C component address and monitor the I²C connection status.

Figure 4-20 I2C Set pane


The four radio buttons are used to specify the I²C slave component and its address (hexadecimal). **Chip Address** is used for communications with the ASIC chips, **Large FPGA** and **Middle FPGA** are used to debug and test the FPGA version, and **Small FPGA** is used to control the auxiliary functions of the FPGA platform. Any of the first three options can be selected on the **Main** tab page, and the last one is always used on the **Slave** page.

NOTE

The FPGA-related functions are designed for HiSilicon development engineers and can be ignored by other users. The large, medium, and small FPGAs on the FPGA platform each has its own component address. The register addresses for the large and medium FPGAs are allocated in a unified manner, that is, duplicate addresses are not allowed. The configuration files for the two tab pages are independent of each other.

The icon below the options indicates whether the I²C communication is normal. If it is normal, **Online** (in green) is displayed; otherwise, **Offline** (in red) is displayed. Users of the professional edition can set the C script function. The HiChannel calls the bound function periodically and checks whether the communication is normal based on the function execution result (the communication is normal if the result is greater than or equal to 1). You can right-click the status icon to select the C script function. See [Figure 4-21](#).

Figure 4-21 Specifying the function

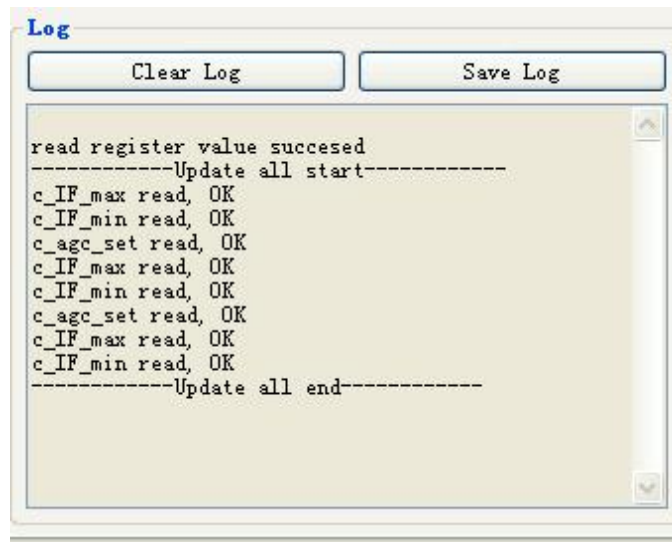

- **Function:** specifies the C script function (select it from the drop-down list). If the bound function does not exist in the C function script specified in the configuration file, the selected option is empty.
- **Uninstall:** leaves the bound function empty



4.8 Log

Figure 4-22 shows the **Log** pane, which displays various operation information, including the normal and error information of reading and writing registers in the **Register** and **Universal Read Write** panes, normal and error information of button operations in the **Button** pane, and error information of operations in the **Indicator**, **Monitor**, and **Trace** panes.

Figure 4-22 Log pane



- **Clear Log**: clears log information in the display area.
- **Save Log**: saves log information in the display area to a specified .txt file.



5 Configuration File



NOTE

This chapter describes the format of the configuration file, which requires attention of users of only the professional edition.

The configuration file consists of the following fields:

- [Global]
- [Button]
- [Indicator]
- [Trace]
- [Monitor]
- [I2C Set]
- [Advance]
- [Segment]
- [Bit Register]
- [Word Register]

The configuration files for the **Main** and **Slave** tab pages are independent and need to be configured separately. However, the file formats are the same.



CAUTION

If a row starts with a semicolon (;), the row is commented out.

5.1 Global



NOTE

In this section, the character strings after "=" are example values that need to be replaced with actual values, and the contents after "/" are description information, which is not required in practice.

The **Global** field stores the global configuration information.



```
[Global] //Field identifier
Chip=12 //Chip name. Characters after "=" are displayed on the top of the window.
Version=0.1 //Version. Characters after "=" are displayed after the chip name on the top of the window.
AddrBytes=1 //Number of bytes in the chip address. Set it to 1 for the 8-bit address line and 4 for the 32-bit address line. It ranges from 1 to 4.
OffsetBits=4 //Number of bits of the offset address. It ranges from 1 to 4.
RegisterBytes=1 //Number of bytes of the register. Set it to 1 for the 8-bit width and 4 for the 32-bit width. It ranges from 1 to 4.
CSRun=E:\HiChannel\demo-script|appl#E:\HiChannel\demo-script2|app2# //List of paths for C script files. Separate the paths by using "#". Data after the file paths is the name of the executable file defined after joint compilation. The file name must be unique. If there is no script file path, you must add the "|" identifier.
CSIncludes=E:\HiChannel\header1|E:\HiChannel\header2|#E:\HiChannel\header2|# //Path of the dependent header file of the C script. Separate the paths by using "#". If multiple header files are stored in the same path for the C script, separate the header file paths by using "|". If there is no header file, you must add the "|" identifier.
```

5.2 Button

The **Button** field corresponds to the **Button** pane on the GUI and stores the configuration information of each button.

```
[Button] //Field identifier
Button 1=FunctionName|Caption|Paramnum|param1,param2|Remind|isReturnValue
//Configuration of the first button
...
```

- **FunctionName:** C script function name specified in the **Function** drop-down list during configuration. The format is *Application name#function name*.
- **Caption:** content in the **Caption** text box in the **Button** pane
- **Paramnum:** number of parameters in the **ParamNum** drop-down list on the GUI. If it is not specified, the value is 0 by default.
- **param1,param2:** parameters in the **Default** text box on the GUI, separated by commas (,)
- **Remind:** content in the **Remind** text box on the GUI, for entering prompt messages
- **isReturnValue:** **Show Return Value** check box on the GUI
 - If the value is **true**, the check box is selected.
 - If the value is **false**, the value is not selected.

5.3 Indicator

The **Indicator** field corresponds to the **Indicator** pane on the GUI and stores the configuration information of each indicator.



```
[Indicator] //Field identifier  
Indicator 1=FunctionName|Caption|Type //Configuration of the first indicator  
Indicator 2=FunctionName|Caption|Type //Configuration of the second indicator  
Indicator 3=FunctionName|Caption|Type //Configuration of the third indicator  
...
```

- **FunctionName:** C script function name specified in the **Function** drop-down list during configuration. The format is *Application name#function name*.
- **Caption:** content in the **Caption** text box
- **Type:** value selected from the **Lamp Mode** drop-down list on the GUI

5.4 Trace

The **Trace** field corresponds to the **Trace** pane on the GUI and stores the configuration information of each trace item.

```
[Trace] //Field identifier  
Trace 1=FunctionName|Caption //Configuration of the first trace item  
Trace 2=FunctionName|Caption //Configuration of the second trace item  
Trace 3=FunctionName|Caption //Configuration of the third trace item  
...
```

- **FunctionName:** C script function name specified in the **Function** drop-down list during configuration. The format is *Application name#function name*.
- **Caption:** content in the **Caption** text box

5.5 Monitor

The **Monitor** field corresponds to the **Monitor** pane on the GUI and stores the configuration information of each monitoring item.

```
[Monitor] //Field identifier  
Lock Time=FunctionName //Lock time  
Symbol Error=FunctionName //Symbol rate error  
Carrier Error=FunctionName //Carrier frequency error  
Signal Intensity=FunctionName //Signal strength  
Signal Quality=FunctionName //Signal quality  
C/N=FunctionName //C/N  
Error Rate=FunctionName //Error rate
```

FunctionName: C script function name specified in the **Function** drop-down list during configuration. The format is *Application name#function name*.



5.6 I2C Set

The **I2C Set** field corresponds to the **I2C Set** pane on the GUI and stores the configuration information of each I²C component.

```
[I2C Set] //Field identifier
Chip Address=0x3F //I2C address for the chip
Large FPGA=0x30 //I2C address for the large FPGA
Middle FPGA=0x1F //I2C address for the middle FPGA
Small FPGA=0x33 //I2C address for the little FPGA
I2C Communication=FunctionName //Communication detection script function (Application name#function name)
Selection Item=Small FPGA //Selected item
```

The component addresses **Chip Address**, **Large FPGE**, **Middle FPGA**, and **Small FPGA** in the **I2C Set** field are stored in hexadecimal format.

5.7 Advance

The **Advance** field stores some advanced configuration items.

```
[Advance] //Field identifier
Spectrum Function=FunctionName //C script function selected for spectrum observation (Application name#function name)
Timer Period=300 //Timer cycle, in ms
I2C Speed=400 //I2C rate, in kHz
Auto Update=Yes //Reserved field
Stop Auto Update=No //Reserved field
Bit Register=Yes //Bit register mode. Yes indicates that Bit Register on the GUI is selected after the configuration file is loaded, and No indicates that Bit Register on the GUI is deselected after the configuration file is loaded. Bit Register and Word Register cannot be Yes at the same time.
Word Register=No //Word register mode. Yes indicates that Word Register on the GUI is selected after the configuration file is loaded, and No indicates that Word Register on the GUI is deselected after the configuration file is loaded. Bit Register and Word Register cannot be Yes at the same time.
```

5.8 Segment

The **Segment** field stores the segment configuration information in the **Register** pane.

```
[Segment] //Field identifier
Number=16 //This number must be consistent with the number of segment base addresses listed below.
//The following lists the segment base addresses. The format is as follows: segment base address (hexadecimal)=segment name.
```




```

00=I2C
10=MAN
20=AGC
...
F0=RSD

```

The number of listed segment base addresses must be the same as that specified in the **Number** field.

5.9 Bit Register

The **Bit Register** field stores configuration information of bit registers. The start bit and end bit of this type of registers can be any number from 0 to 31. The registers are listed by segment. The segment base address is listed first (which is defined in the **Segment** field), then registers that belong to the segment are listed. A blank row is left between every two segments.

```

[Bit Register]    //Field identifier

[00]             //[Hexadecimal base address]. The base address must be defined in the Segment field. Otherwise,
it is invalid.

Number=8          //Number of registers in this segment (decimal)

00[14: 9]=6'd55    | c_IF_max      //[Offset bit field address]=[Number of bits]'[Numeral
system][Default value] |[Register name]
01[24:19]=6'd35    | c_IF_min
02[ 7: 0]=8'd43    | c_agc_set
...

[10]             //[Hexadecimal base address]. The base address must be defined in the Segment field. Otherwise,
it is invalid.

Number=4          //Number of registers in this segment (decimal)

00[14: 9]=6'd55    | c_cbs_test    //[Offset bit field address]=[Number of bits]'[Numeral
system][Default value] |[Register name]
01[24:19]=6'd35    | c_cbs_XXX

```

5.10 Word Register

The **Word Register** field stores configuration information of word registers. The start bit and end bit of this type of registers are fixed at (8 x **RegisterBytes** – 1:0), and the number of bits is fixed at 8 x **RegisterBytes**. **RegisterBytes** is described in the **Global** field. The registers are listed by segment. The segment base address is listed first (which is defined in the **Segment** field), then registers that belong to the segment are listed. A blank row is left between every two segments.

```

[Word Register]   //Field identifier

[30]             //[Base address for the segment to which the register belongs. The base address must be defined in
the Segment field. Otherwise, it is invalid.

Number=8          //Number of registers in this segment (decimal)

```



00[7:0]=8'd55 | c_IF_max // [Offset bit field address]=[Number of bits]'[Numeral
system][Default value] |[Register name]

The offset bit field address is relative to the specific base address. Its format is OffsetAddr[HighBit, LowBit]. **OffsetAddr** is hexadecimal, and **HighBit** and **LowBit** are decimal. For example, if the base address is 0x20, and the offset bit field address is 0xF[5, 2], it indicates bit 5 to bit 2 in the address 0x2F.



6 C Script

This chapter describes how to use the dynamic library interfaces provided by the HiChannel when users of the professional edition write the C scripts bound to controls on the GUI. The interfaces are related to board operations.

6.1 Dynamic Library Interfaces

Socket Connection

- `int InitSock(char* SevIP, int SevPort)`
Sets up a socket connection.
The parameters are described as follows:
 - **char* SevIP**: IP address for the server end
 - **int SevPort**: port ID of the server endThe return value 0 indicates success, and other values indicate error code.
- `int DeinitSock()`
Ends a socket connection.
The return value 0 indicates success.
- `int openi2c()`
Opens the I²C port.
The return value 0 indicates success, and other values indicate error code.
- `int closei2c()`
Closes the I²C port.
The return value 0 indicates success, and other values indicate error code.
- `seti2cspeed(int i2cPort, int i2cSpeed)`
Sets the I²C port and speed.
The parameters are described as follows:
 - **int i2cPort**: I²C port ID
 - **int i2cSpeed**: I²C speedThe return value 0 indicates success, and other values indicate error code.



- `int ReadregMulti(int i2cport, char idevaddr, int iregaddr, int iregaddrcount, char *buffer, int len)`
Reads registers.
The parameters are described as follows:
 - **int i2cPort:** I²C port ID
 - **char idevaddr:** device address
 - **int iregaddr:** register address
 - **int iregaddrcount:** number of bytes of the register address
 - **char *buffer:** start address for the values to be read
 - **int len:** number of registers to be readThe return value 0 indicates success, and other values indicate error code.
- `WriteregMulti(int i2cport, char idevaddr, int iregaddr, int iregaddrcount, char *data, int len)`
Writes to registers.
The parameters are described as follows:
 - **int i2cPort:** I²C port ID
 - **char idevaddr:** device address
 - **int iregaddr:** register address
 - **int iregaddrcount:** number of bytes of the register address
 - **char *data:** start address for the data to be written
 - **int len:** number of registers to be writtenThe return value 0 indicates success, and other values indicate error code.
- `int ReadregPeriod(int i2cport, char idevaddr, int iregaddr, int iregaddrcount, char *buffer, int iregbytes, int count, int period)`
Reads a register periodically.
The parameters are described as follows:
 - **int i2cPort:** I²C port ID
 - **char idevaddr:** device address
 - **int iregaddr:** register address
 - **int iregaddrcount:** number of bytes of the register address
 - **char *buffer:** start address for the value to be read
 - **int iregbytes:** number of bytes of the register
 - **int count:** read count
 - **int period:** read cycle, in μ sThe return value 0 indicates success, and other values indicate error code.



CAUTION

- `DeinitSock()` and `InitSock()` must be used in pairs. Otherwise, other operations may be affected.



- When ReadregPeriod is called, **count*period** must be less than 4000000 because the socket read timeout period is set to 4s in the dynamic library. If it is greater than 4000000, dll returns -7.

USB Connection

- USBIO_OpenDevice(int deviceId)
Opens the USB device.
The parameters are described as follows:
int deviceId: device ID. **0** corresponds to the first device.
The device handle is returned. It is invalid if an error occurs.
- USBIO_CloseDevice(int deviceId)
Closes the USB device.
The parameters are described as follows:
int deviceId: device ID. **0** corresponds to the first device.
- USBIO_SetStream(int deviceId, int mode)
Sets the serial port stream mode.
The parameters are described as follows:
 - **int deviceId**: device ID. **0** corresponds to the first device.
 - **int mode**: specified mode
 - Bits 0 to 1: I²C port speed/SCL frequency. **00** indicates low speed/20 kHz, **01** indicates standard speed/100 kHz (default value), **10** indicates fast speed/400 kHz, and **11** indicates high speed/750 kHz.
 - Bit 2: SPI I/O pins. **0** (default value) indicates single input single output (D3 clock/D5 output and D7 input), and **1** indicates dual inputs and dual outputs (D3 clock/D4 and D5 outputs/D6 and D7 inputs).
 - Bit 7: bit sequence in SPI bytes. **0** indicates lower bits first, and **1** indicates upper bits first.
 - Other bits are reserved and set to 0.
- USBIO_StreamI2C(int deviceId, int writeLength, char * writeBuffer, int readLength, char * readBuffer)
Processes I²C data streams.
The parameters are described as follows:
 - **int deviceId**: device ID. **0** corresponds to the first device.
 - **int writeLength**: number of data bytes to be written
 - **char * writeBuffer**: pointer to a buffer for storing data to be written. The first byte specifies the I²C device address and read/write direction.
 - **int readLength**: number of data bytes to be read
 - **char * readBuffer**: pointer to a buffer. The read data is returned.

6.1.2 Requirements on Return Values of Functions Bound to Controls

The C script functions bound to controls must be displayed in the following format:

```
if(No exception is returned during the entire process
```



```
printf("error=%s,return=%f","ok",0.0);
elseif(error 1)
printf("error=%s,return=%d","err 1, failed", 1);
elseif(error 2)
printf("error=%s,return=%d","err 2, failed", 2);
```

- **error**: describes the execution status information. If the status information contains **err** or **failed**, the function execution fails; otherwise, the execution is successful.
- **return**: describes the actual return value. The HiChannel parses the actual return value only when the returned status is success.

When you write the C script functions that are bound to controls, pay attention to the requirements on the return values of functions for each control:

- Button: not restricted
- Indicator: The return value must be 0 or a positive integer.
- Trace: not restricted
- I2C Set: The return value must be an integer.
- Monitor:
 - Lock Time: The return value must be 0 or a positive integer.
 - Symbol Error: The return value must be a floating-point number or an integer. Note that the floating-point number is rounded off to five decimal places.
 - Carrier Error: The return value must be a floating-point number or an integer. Note that the floating-point number is rounded off to five decimal places.
 - Signal Intensity: The return value must be an integer. If the return value is smaller than 0, it is displayed on the progress bar as 0%; if the return value is greater than 100, it is displayed on the progress bar as 100%.
 - Signal Quality: The return value must be an integer. If the return value is smaller than 0, it is displayed on the progress bar as 0%; if the return value is greater than 100, it is displayed on the progress bar as 100%.
 - C/N: The return value must be a floating-point number or an integer.
 - Error Rate: The return value must be a floating-point number or an integer.
- Spectrum:

When data in the function is read to the spectrum view, the function return value defined in the C script must meet the following requirements:

- It is a character string that consists of two-dimensional coordinates. The points are separated by using a pound sign (#).
- The X and Y coordinates of a point are separated by using a comma (,).
- The X and Y coordinates must be integers or floating-point numbers.

The format is as follows: {x1,y1}#{x2,y2}#{x3,y3}#.... For example, {3,4}#{5,6}#{7,8}#.

The reference sample is as follows:

```
#include "stdio.h"
#define length 300
int fftdata(int argc, char ** argv )
{
    int doubleArray[2][length];
```



```
int i;  
int j;  
for(i = 0; i < length; i++){  
doubleArray[0][i] = i;  
doubleArray[1][i] = i + 1;  
}  
printf("\n");
```



CAUTION

"error=%s,return=" must be outside the **for** cycle.

```
printf("error=%s,return=", "ok");  
for(j = 0; j < (length- 1); j++){  
printf("{%d,%d}#",doubleArray[0][j],doubleArray[1][j]);  
}  
return 0;  
}
```

6.1.3 Prerequisite

The prerequisite is as follows:

- The debug agent service has been started on the board.
- For the network connection:
 - **IP Address** in the DUT network configuration must match the IP address for the board.
 - **I2C Port**, **Dev Address**, and **I2C Speed** in the DUT network configuration must match the configuration of the board.
- For the USB connection:
Dev Address and **I2C Speed** in the USB device configuration must match the configuration of the board.

6.1.4 I2C Read/Write

The C script calls the dll dynamic library to set up the communication for I²C read and write.

Socket Connection

If the DUT network and I²C rate are configured correctly, the following seven parameters are transferred to the C script function by default when the script function is executed:

- IP address for the board, for example, **192.168.0.1**
- Server port (fixed at 4000)
- I²C port (decimal integer), for example, **3**



- I²C rate (The unit is not transferred. For example, if the rate is set to 100K, only 100 is transferred.)
- Device address (The hexadecimal device address is converted into the decimal data and then transferred. For example, if the device address is set to b6, it is converted into the decimal 182 and then transferred.)
- Function name
- Connection mode identifier (–1 indicates USB, and 0 indicates socket.)

int functionName(int argc, char ** argv)

argc indicates the number of parameters, and **argv** stores all parameter values.

When the I²C read/write operation is to be performed, parameters are obtained based on the index. The parameter index starts from 1.

- **argv[1]**: board IP address (character string)
- **argv[2]**: server port (integer)
- **argv[3]**: I²C port (integer)
- **argv[4]**: I²C rate (integer)
- **argv[5]**: device address (integer)
- **argv[6]**: (Param1) If multiple parameters are configured for the button, they can be obtained in sequence.
- **argv[7]**: (Param2) If multiple parameters are configured for the button, they can be obtained in sequence.
- **argv[argc-3]**: (Param n) If multiple parameters are configured for the button, they can be obtained in sequence.
- **argv[argc-2]**: name of the function to be called (without #)
- **argv[argc-1]**: connection mode

USB Connection

If the USB connection is configured correctly, the following four parameters are transmitted to the C script function by default when the script function is executed:

- USB rate
- Device address
- Function name
- Connection mode identifier (–1 indicates USB, and 0 indicates socket.)

int functionName(int argc, char ** argv)

argc indicates the number of parameters, and **argv** stores all parameter values.

When the I²C read/write operation is to be performed, parameters are obtained based on the index. The parameter index starts from 1.

- **argv[1]**: USB rate (integer)
- **argv[2]**: device address (integer)
- **argv[3]**: (Param1) If multiple parameters are configured for the button, they can be obtained in sequence.



- **argv[4]**: (Param2) If multiple parameters are configured for the button, they can be obtained in sequence.
- **argv[argc-3]**: (Param n) If multiple parameters are configured for the button, they can be obtained in sequence.
- **argv[argc-2]**: name of the function to be called (without #)
- **argv[argc-1]**: connection mode



CAUTION

All parameters are transmitted to the C script as character strings. Therefore, the parameters need to be converted based on their types when being used. Otherwise, a timeout occurs when the program loads the parameters, and the program does not respond.

6.2 Description of Script Functions

- Note the following when using the functions:
 - For the socket connection in normal status, InitSock and DeinitSock must be called each time a single register is read/written, and InitSock and DeinitSock need to be called only before the first register is read/written and after the last register is read/written respectively when registers are read/written in batches.
 - For the USB connection, USBIO_StreamI2C can be used to perform read/write operations.
- Functions in the C script can be called by each other. However, the syntax must comply with the C language standard.



CAUTION

- If some functions in the C script are for internal use only, **static** must be marked before the corresponding functions in case that the functions are called externally. There can be multiple functions in a C script.
- The names of functions in the same C script file path cannot be duplicate.
- In the C script, if a defined function is to be called by other C script functions but not displayed in the function drop-down list of the GUI, you need to add **inner_** before the corresponding function name. For example, int inner_compile(int argc, char**argv).

6.3 Instance



CAUTION

This instance is used as only reference for writing C script functions and cannot be used directly.

When I²C read/write is required, a header file must be created to define the calling functions. The following is an instance of the header file:

```
#define CHIP_PORT atoi(argv[3]) //For the DUT network connection, argv[3]
defines the I2C port.
#define CHIP_ADDR atoi(argv[5])//For the DUT network connection, argv[5]
defines the device address.
#define USB_ADDR atoi(argv[2]) //For the USB connection, argv[2] defines
the USB device address.
#define USB_SPEED atoi(argv[1]) //For the USB connection, argv[1] defines
the USB rate.
//g_connect defines whether the connection is socket connection or USB
connection. 0 indicates the USB connection, and non-0 indicates the
socket connection.
#define INIT_CONNECT if (g_connect) {\
InitSock(argv[1],atoi(argv[2])); \
}else{\
if(USBIO_OpenDevice(0) == INVALID_HANDLE_VALUE ){ \
printf("error=%s,return=%d","err openererror,failed",-1); \
return; \
} \
USBIO_SetStream(0,USB_SPEED);\
}
#define DEINIT_CONNECT if (g_connect) {\
DeinitSock();\
}else{\
USBIO_CloseDevice(0);\
}
#define CHIP_MRD(REG_ADDR, P_RBUF, RLEN) if (g_connect)
{ ReadregMulti((CHIP_PORT), (CHIP_ADDR), (REG_ADDR), 1, ((char
*)(P_RBUF)), (RLEN)); \
} else {\
unsigned char P_WBUF[2];\
P_WBUF[0] = USB_ADDR;\
P_WBUF[1] = REG_ADDR;\
USBIO_StreamI2C(0,2,(PVOID)pSend,RLEN,(PVOID)(P_RBUF));\
}
#define CHIP_MWR(REG_ADDR, P_WBUF, WLEN) if (g_connect)
{ WriteregMulti((CHIP_PORT), (CHIP_ADDR), (REG_ADDR), 1, ((char
```



```
*)(P_WBUF)), (WLEN));\n}\nelse {\nunsigned char P_RBUF;\nchar *p = (char *)malloc((WLEN)+2);\nchar *pw =(char *) (P_WBUF);\nint i = 0;\np[0]= USB_ADDR;\np[1]= REG_ADDR;\nfor (i=0;i<(WLEN);i++) {\np[i+2] = pw[i];\n};\nUSBIO_StreamI2C(0,2+(WLEN),(PVOID)p,0,(PVOID)(&cRecv));\n}
```

After the header file is defined, it must be introduced at the beginning of the C script. See the following instance:

```
#include "stdio.h"\n#include "string.h"\n#include self-defined header file\n#include "USBIOX.H" //Mandatory\nextern int g_connect; //Mandatory\nint on_line(int argc, char ** argv)\n{\nINIT_CONNECT;\nunsigned char chip_id[4];\nunsigned char on_line;\nCHIP_MRD(0x6f, &chip_id, 4);\non_line = (chip_id[2]==0x36) && (chip_id[3]==0x31);\nif(on_line) {\nprintf("error=%s,return=%d","ok",1); }\nelse {\nprintf("error=%s,return=%d","ok",0); }\nDEINIT_CONNECT;\n}\n//Internal private function, identified by "static"\nstatic int demo_m2( int argc, char ** argv )\n{\nint i = 0;\nprintf("demo_m2.....");\nreturn 0;\n}\n//If the function is to be called externally and not displayed in the\nfunction list of the GUI, add "inner_".
```



```
int inner_compile( int argc, char ** argv )
{
    int i = 0;
    printf("I'm inner function");
    return 0;}
```



7 Advanced Functions

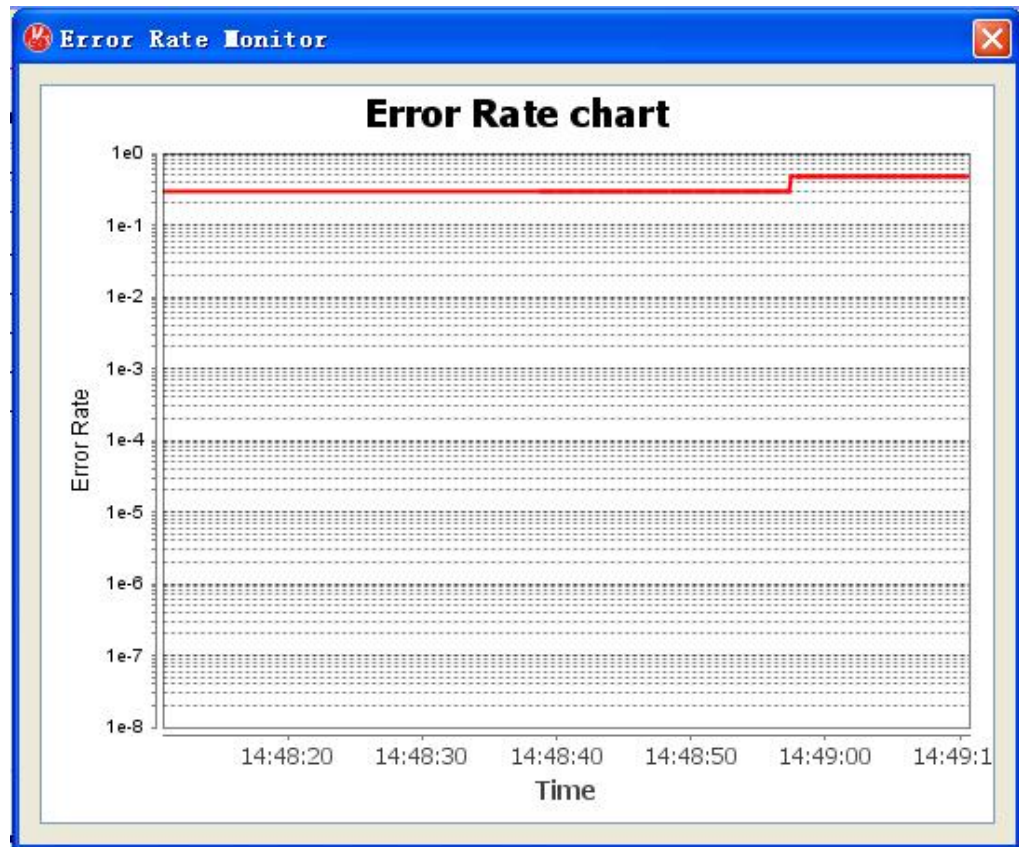
The **Advance** menu contains the following items:

- Error Rate Monitor
- CN Monitor
- Constellation View
- Spectrum View
- Time Domain View

7.1 Error Rate Monitor

[Figure 7-1](#) shows **Error Rate Monitor**.

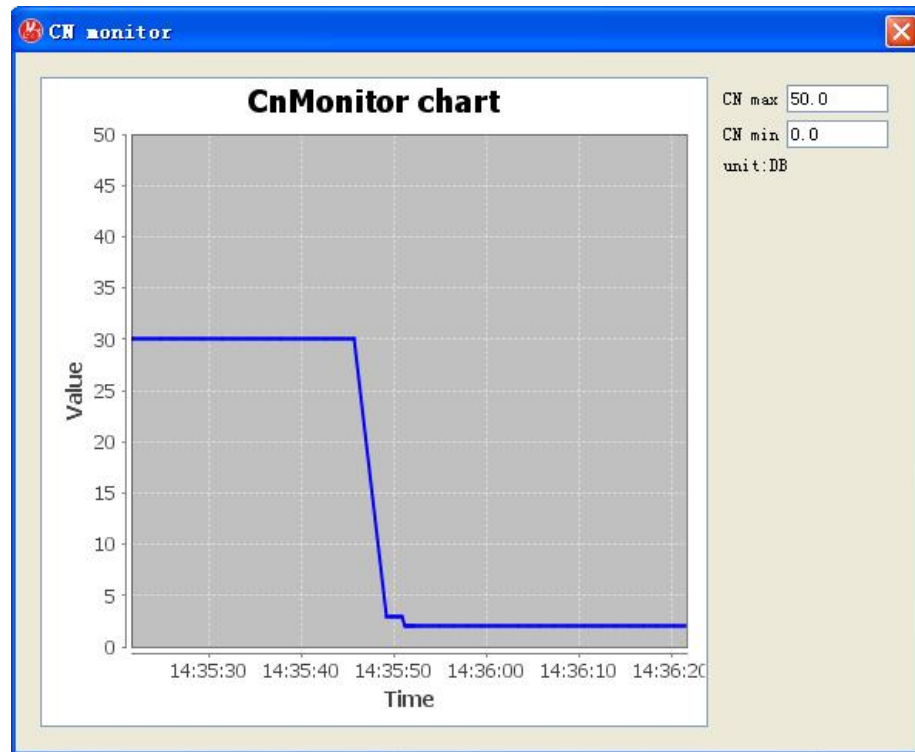
Figure 7-1 Error Rate Monitor



Data in the **Error Rate Monitor** is the same as that of **Error Rate** in the **Monitor** pane, that is, the data comes from the same function, and the error type is the same. The latest 300 error rates are displayed. One error rate indicates the error rate in one timing period. The monitoring stops after automatic refresh stops.

7.2 C/N Monitor

Figure 7-2 shows **CN Monitor**.

Figure 7-2 CN Monitor

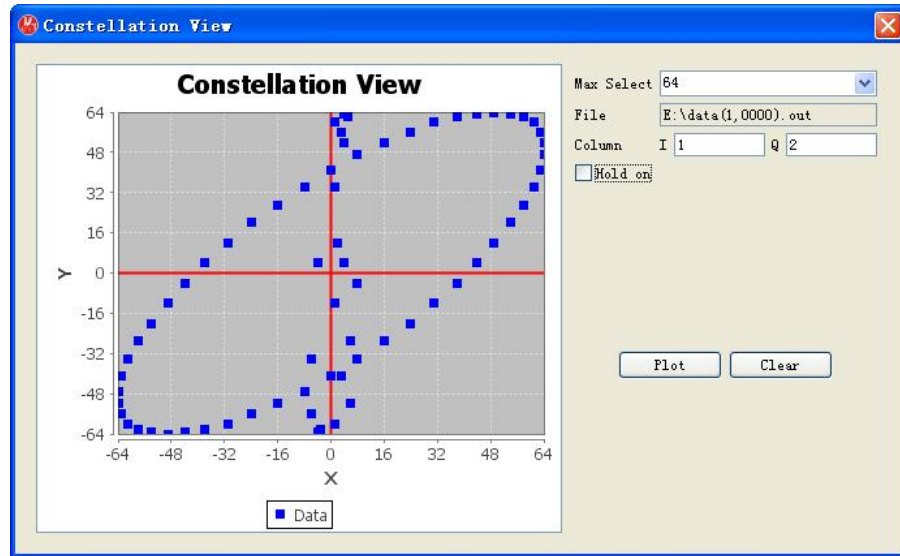
The C/N data is the same as C/N in the **Monitor** pane, that is, the data comes from the same function. The latest 300 data items are displayed. The monitoring stops after automatic refresh stops. If you click the **CN max** text box and enter a maximum C/N scale, or click the **CN min** text box and enter a minimum C/N scale, the previously recorded C/N values are still valid and displayed in the right positions based on the new configuration.

7.3 Constellation View

Figure 7-3 shows **Constellation View**.



Figure 7-3 Constellation View

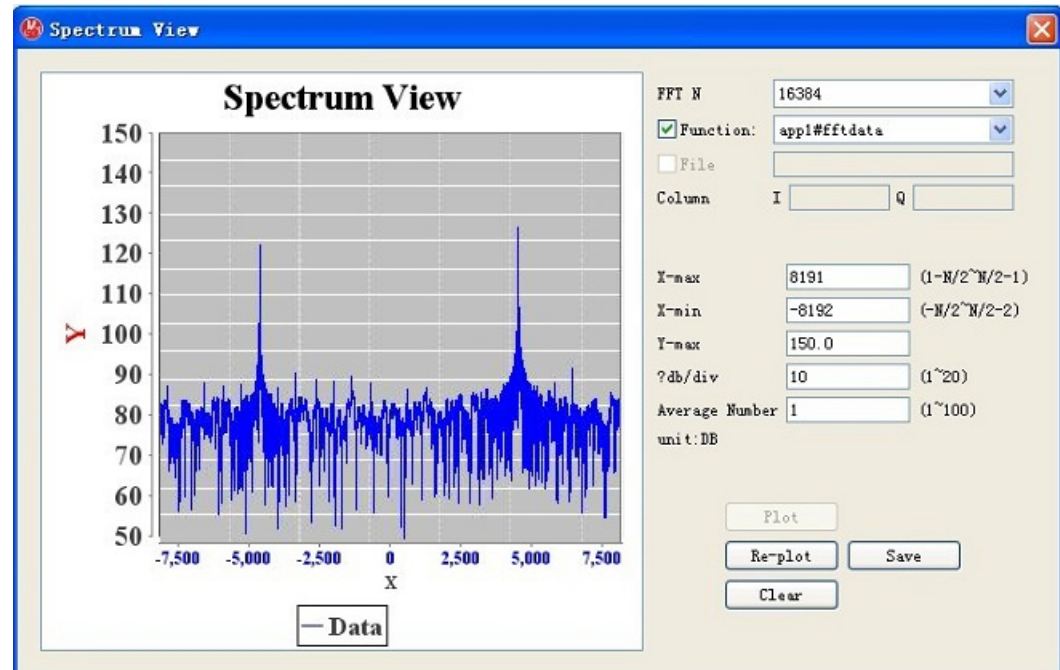


Select the maximum value from the **Max Select** drop-down list, select a file, and enter the **I** and **Q** values. Then you can click **Plot** to display the data or click **Clear** to clear the displayed data. If **Hold on** is selected, data is displayed on the original diagram; otherwise, data is displayed on a new diagram.

7.4 Spectrum View

Figure 7-4 shows the **Spectrum View**.

Figure 7-4 Spectrum View



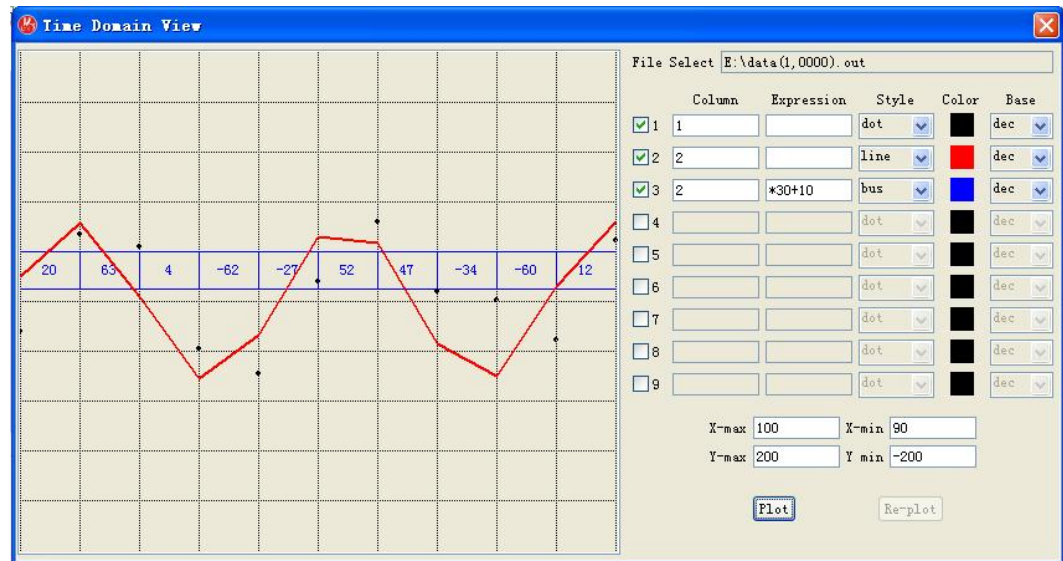
- The data source can be a specified data file or be generated by a C script function. Select **Function** or **File**. The deselected option is dimmed and unavailable.
 - If **Function** is selected, select a function from the drop-down list on the right.
 - If **File** is selected, click the text box on the right, select a file, and enter the column IDs of I and Q in the file in the text boxes next to **Column**.
- Click the **X-max** text box, and enter the maximum scale of the X axis ($1-N/2-N/2-1$, integer).
- Click the **X-min** text box, and enter the minimum scale of the X axis ($-N/2-N/2-2$, integer).
- Click the **Y-max** text box, and enter the maximum scale of the Y axis (floating point number).
- Click the **?dB/div** text box, and enter a value in a grid (1–20, integer).
- Click the **Average Number** text box, and enter the average times (1–100, integer).
- Click **Plot** to draw the spectrum diagram.
- Click **Re-plot** to redraw the diagram based on the new configurations (the file is not read). The configurations do not take effect after modification if **Plot** or **Re-plot** is not clicked.
- Click **Clear** to clear the spectrum data in the diagram.
- Click **Save** to save the graphics to a specified file.

7.5 Time Domain View

Figure 7-5 shows the **Time Domain View**.



Figure 7-5 Time Domain View



Click the **File Select** text box, and select a file. Select the signals to be displayed (at most nine signals can be selected). \checkmark indicates selected, and signals can be selected repeatedly. The five items (**Column**, **Expression**, **Style**, **Color**, and **Base**) of the selected signals can be configured. If a signal is not selected, the corresponding items cannot be configured.

- Click the text box in the **Column** column, and enter the column ID of the signal in the file.
- Click the text box in the **Expression** column, and enter the expression. If no calculation is involved, enter **+0**. In the expression, multiplication is used to adjust the signal swing, and subtraction is used to adjust the signal vertical position. You can configure the expressions properly so that multiple signals do not overlap each other on the diagram.
- Click the drop-down list in the **Style** column to select a display style (**bus**, **dot**, or **line**).
- Click the **Color** column to select a display color.
- Click the drop-down list in the **Base** column to select a numeral system (**dec**, **hex**, or **bin**, indicating decimal, hexadecimal, or binary).
- Click the **X-max** text box, and specify the end point for display (the specified point is displayed).
- Click the **X-min** text box, and specify the start point for display (the specified point is displayed).
- Click the **Y-max** text box, and enter the maximum scale of the Y axis (floating point number).
- Click the **Y-min** text box, and enter the minimum scale of the Y axis (floating point number).
- Click the **Plot** button to start drawing.
- Click the **Re-plot** button to redraw the diagram based on new configurations (the file is not read). If **Plot** or **Re-plot** is not clicked after **Y-max** or **Y-min** is modified, the modification does not take effect.
 - If **Style** is set to **dot**, each data corresponds to a point, and the Y coordinate is the calculation result of the expression after the value read from the file is put into the expression.



- If **Style** is set to **line**, a line that connects all dots is displayed, but the dots are not separately displayed.
- If **Style** is set to **bus**, the value of a point (original data read from the file that is not calculated by using the expression, displayed in the numeral system specified by **Base**) displayed in the grid. If the amount of data to be displayed is too large to be displayed in the grid, the data that exceeds the grid is not displayed. The top edge of the grid is the calculation result of putting 1 into the expression, and the bottom edge of the grid is that of putting 0 into the expression. The grid between point x and point x+1 displays the content of point x.



CAUTION

The expression contains only +, -, *, /, or numbers. The first character must be a computing sign, and the expression must comply with the rules of elementary arithmetic. For example, *30-10.



7.6 Data File Format

This section describes the format conventions of files that need to be loaded for the spectrum view, constellation view, and time domain view.

7.6.1 Conventions

- The number of columns in a row must be specified in the first row in the format of "Column:2". **Column** is not case-sensitive, the column quantity is a decimal number (greater than 0), and they are separated by a colon (:).
- Columns are separated by commas (.). For example, **-1,2** indicates that **-1** belongs to the first column, **2** belongs to the second column, and there are a total of two columns in the row.
- After the column quantity is specified in the first row, the quantity of columns in each row must be greater than or equal to the defined quantity. Otherwise, data in the row is invalid, and the coordinates in the row are not displayed in the view.
- The coordinate data must be positive/negative decimal integers or floating point numbers.

7.6.2 Instance

```
Column:2  
-1,2  
2,3  
2,5  
4,
```

Note that the column quantity of the last row is inconsistent with that specified in the first row (which is 2), and therefore the coordinate in this row is not displayed in the view.

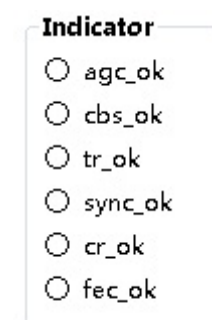


8 Functions of the Customer edition

8.1 Indicator

Figure 8-1 shows the **Indicator** pane.

Figure 8-1 Indicator



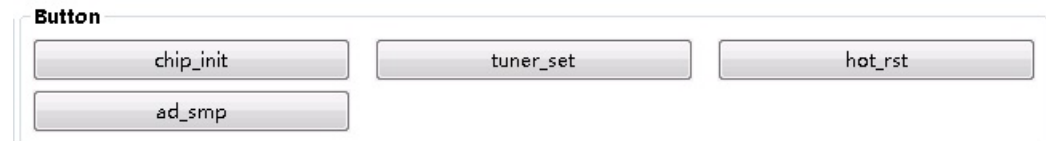
- **agc_ok**: If it is green, the AGC is locked; if it is red, the AGC is unlocked.
- **cbs_ok**: If it is green, blind scanning is successful; if it is red, blind scanning fails.
- **tr_ok**: If it is green, the timer is locked; if it is red, the timer is unlocked.
- **sync_ok**: If it is green, the S2 signal frame synchronization is successful; if it is red, the synchronization fails.
- **cr_ok**: If it is green, the carrier is locked; if it is red, the carrier is unlocked.
- **fec_ok**: If it is green, error correction decoding is successful; if it is red, error correction decoding fails.

8.2 Button

Figure 8-2 shows the Button pane.



Figure 8-2 Button



- **chip_init**: Initializes the chip. It requires no input parameter and has no return value.
- **tuner_set**: Initializes the tuner and configures the carrier center frequency and tuner filter bandwidth. The input parameters are as follows:
 - Tuner I²C channel ID
 - Tuner I²C component address
 - Tuner type (0-AVL2012, other types have not been added)
 - Tuner carrier center frequency (MHz)
 - Tuner low-pass filter bilateral bandwidth (MHz). It has no return value.
- **hot_rst**: Soft-resets the chip. It requires no input parameter and has no return value.
- **ad_smp**: Implements the embedded data sampling function of the chip. The input value is 1. (**0** indicates that the AD output data is sampled, and **1** indicates that the equalized output data is sampled.) Data is output to the file **hi_ad_smp**, and there are a total of 2048 sampling points.

8.3 Trace

Figure 8-3 shows the **Trace** pane.

Figure 8-3 Trace



Signal Basic: After the received signals are locked, basic information is displayed, including the standard (DVB-S2/DVB-S/DirectTV), modulation mode, bit rate, code length, and whether there is pilot and constant coding and modulation (CCM) information. If the signals are not locked, **unknown standard** is displayed.

8.4 Monitor

Figure 8-4 shows the **Monitor** pane.



Figure 8-4 Monitor

Monitor			
Lock Time	0 ms	Signal Intensity	7%
Symbol Error	0.00000 kHz	C/N	0.0
Carrier Error	0.00000 kHz	Signal Quality	0%
		Error Rate	0.5

- **Lock Time:** duration from chip reset to fec_ok (in ms). The data is updated after each reset.
- **Symbol Error:** error between the symbol rate of the current signal and that obtained by blind scanning (in kHz). A positive value indicates that the current symbol rate is greater.
- **Carrier Error:** error between the carrier center frequency of the current signal and that obtained by blind scanning (in kHz). A positive value indicates that the current carrier frequency is higher.
- **Signal Intensity:** signal strength
- **Signal Quality:** signal quality (not implemented yet)
- **C/N:** carrier-to-noise ratio (in dB), valid only after the signal is locked (**-10.0** is displayed if the signal is not locked)
- **Error Rate:** bit error rate, valid only after the signal is locked (**0.5** is displayed if the signal is not locked)

8.5 Segment

Figure 8-5 shows the **Register** pane.

Figure 8-5 Register

Register			
Segment	ALL		
	<div>Add Modify Delete</div>		
Name	0xb0 [DSEC]		
	0xa0 [CBS]		
DSEC_AD	0x90 [OUTP]		
DSEC_DA	0x80 [FEC]		
DSEC_RA	0x70 [EQU]		
DSEC_RA	0x60 [SYNC]		
DSEC_RA	0x50 [CR]		
TX_CTRL	0x40 [TR]		
RX_CTRL	0x20 [MAN]		
DSEC_EN	0x30 [AGC]		
	ALL		
RX_STATE	0xB8 [7:0]	h0	
INT_STATE	0xB9 [7:0]	h0	
FC_MAX_RELIAB...	0xA0 [7:0]	he4	
FS_SPAN	0xA1 [7:0]	h5	
AMP_MIN_FS	0xA7 [7:0]	h67	
CBS_CTRL_RDAD...	0xA8 [7:0]	h80	

- **MAN:** chip status and reset control register group
- **AGC:** AGC control and status register group



- **TR**: timing recovery control and status register group
- **CR**: carrier recovery control and status register group
- **SYNC**: frame synchronization control and status register group
- **EQU**: equalization control and status register group
- **FEC**: error correction decoding control and status register group
- **OUTP**: TS interface control register group
- **CBS**: blind scanning control and status register group
- **DSEC**: iseqc control and status register group

8.6 Operation Procedure

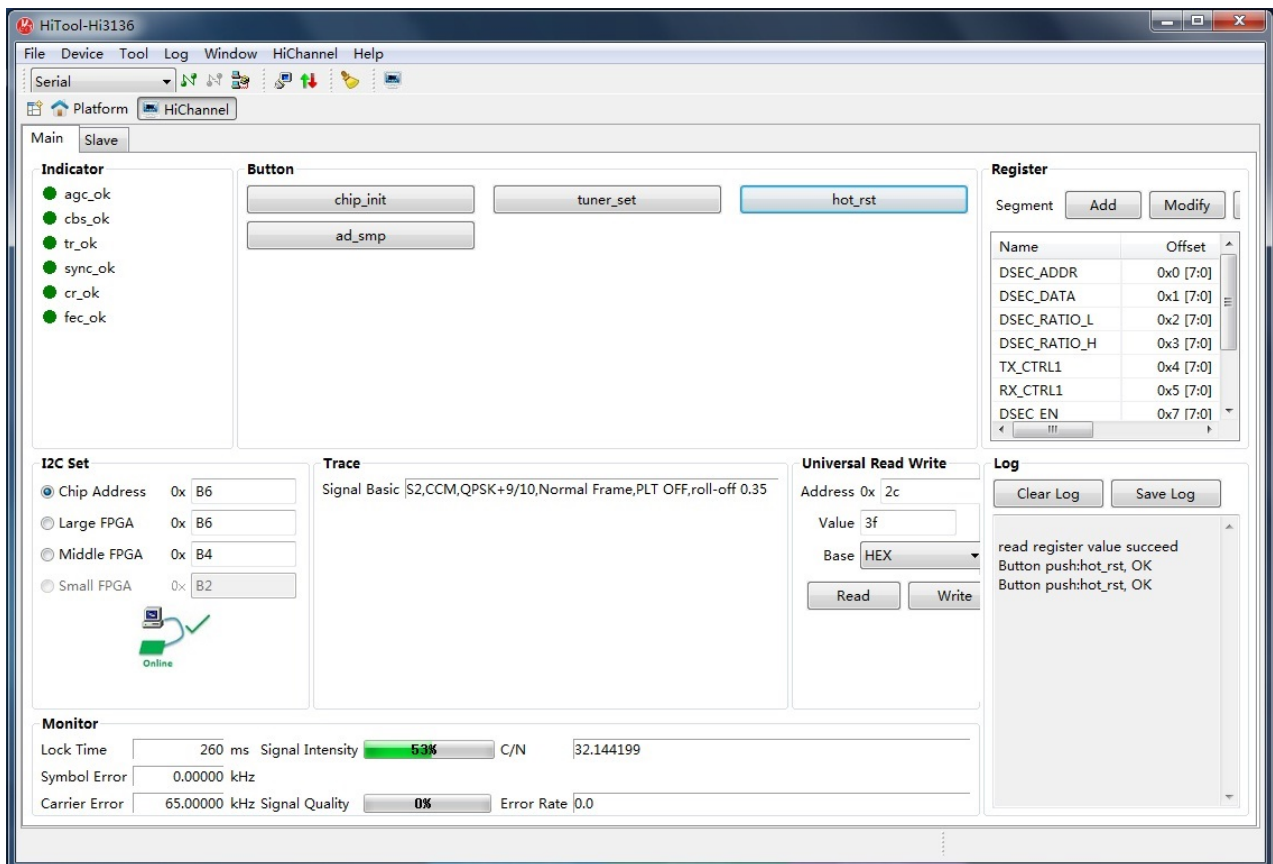
To use the HiChannel, perform the following steps:

- Step 1** Connect the signal cable, power cable, serial cable, and network cable, and power on the board.
- Step 2** Set the IP address for the board over the serial port.
- Step 3** Open the HiChannel, click the icon for Connection Manager in the toolbar, and set the board IP address, chip I²C channel ID, I²C component address, and I²C communication rate (kHz) for **DUT Network**. For details, see section 3.2.2 "Script."
- Step 4** Click the **chip_init** button.
- Step 5** Click the **tuner_set** button, and specify the tuner I²C channel ID, I²C component address, tuner type, carrier center frequency, and low-pass filter bilateral bandwidth based on actual configurations.
- Step 6** Click the **hot_rst** button.

If the signal quality meets the chip reception requirements, indicators in the **Indicator** pane turn green (sync_ok does not turn green for the DVB-S/DirecTV standard), the **Trace** pane displays basic information about the signals, and the **Monitor** pane displays the lock time, symbol rate error, carrier frequency error, signal strength, C/N, and BER. Figure 8-6 shows the overall GUI.



Figure 8-6 Overall GUI



Then you can read/write to registers or sample data by clicking the **ad_smp** button as required. You can also choose **HiChannel > Advance** to display the spectrum, constellation, or time domain view.

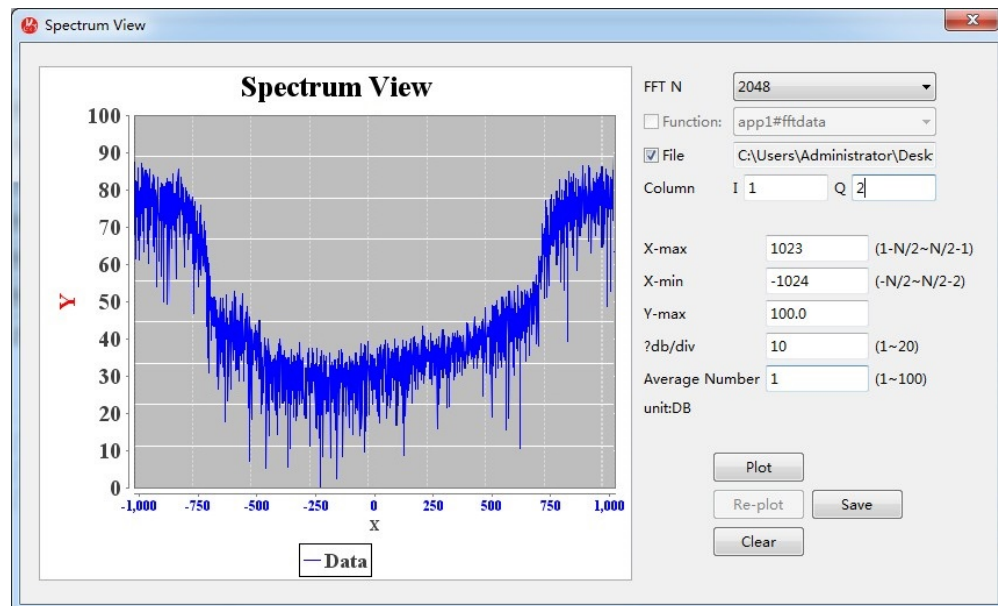
----End

8.7 Spectrum View

Choosing **HiChannel > Advance > Spectrum View** displays the spectrum data. [Figure 8-7](#) shows the spectrum of input signals sampled by clicking the **ad_smp** button. The signal symbol rate is 30 Mbaud, and the sampling rate is 125 MHz.



Figure 8-7 Spectrum view



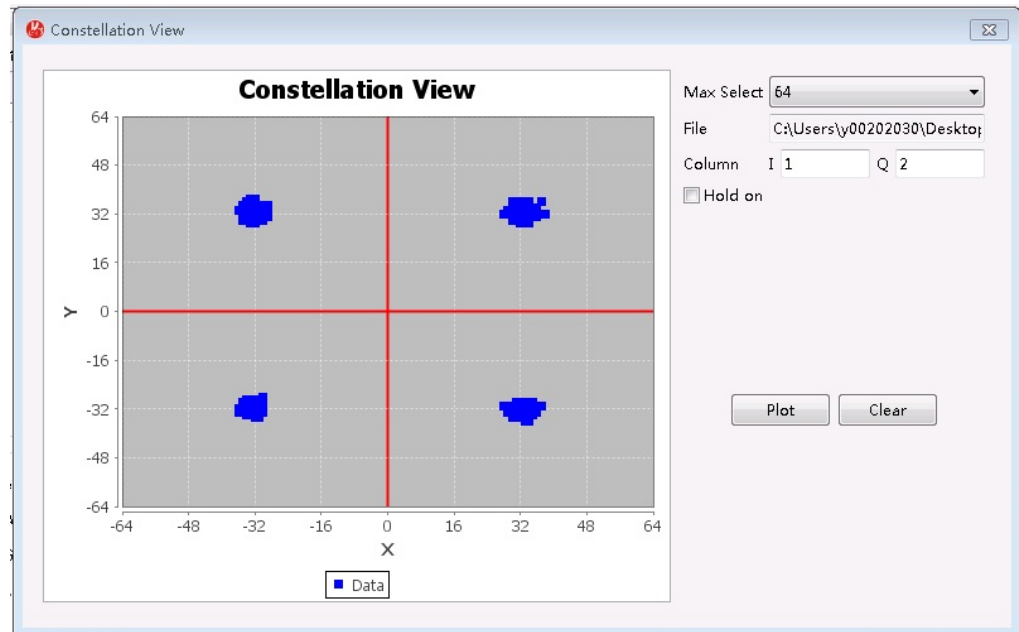
You can change parameters such as the FFT point quantity, X and Y axis ranges, and average times by specifying parameters on the right.

8.8 Constellation View

Choosing **HiChannel > Advance > Constellation View** displays the constellation data. [Figure 8-8](#) shows the constellation of equalized output signals sampled by clicking the **ad_smp** button. The signals are quadrature phase shift keying (QPSK) signals, and the signal-to-noise ratio (SNR) is about 30 dB.



Figure 8-8 Constellation view



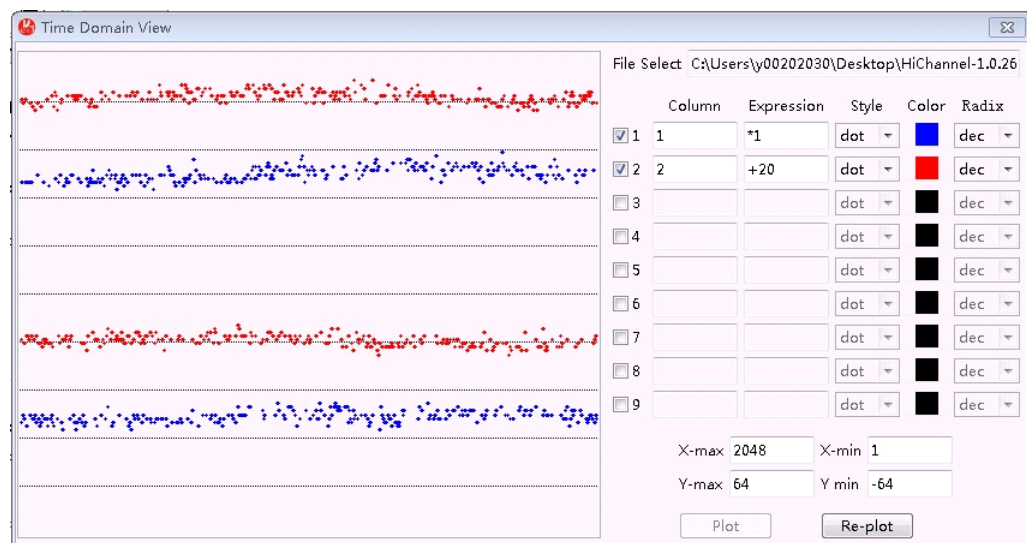
You can change the X and Y axis ranges by specifying parameters on the right.

8.9 Time Domain View

Choosing **HiChannel > Advance > Time Domain View** displays the time domain data.

[Figure 8-9](#) shows the time domain waveform of equalized output signals sampled by clicking the **ad_smp** button. The signals are QPSK signals, and the SNR is about 30 dB.

Figure 8-9 Time domain view





You can control the scaling and offset of input data and change the display type and color by specifying parameters on the right.



9 Precautions

- The configuration file must be in the specific format. Otherwise, it cannot be loaded or a running error may occur. You are advised not to modify the configuration file manually, unless you are very familiar with the configuration file format.
- If a C script file contains syntax errors, it fails to be loaded, and the errors are reported when you load the file.
- As the C script uses the internal compilation operating mode (after the C script is loaded successfully, it is directly compiled as executable code and executed in the HiChannel), misoperations during the running of the C script (for example, operate an empty pointer) may result in an unpredictable error or even system crash.



10 FAQs

What Do I Do If "Invalid Word Register bit area info\ : start bit must be xx at line xx" Is Displayed?

Problem Description

When the configuration file is being loaded, "Invalid Word Register bit area info\ : start bit must be xx at line xx" is displayed.

Cause Analysis

The configuration of the **Word Register** field in the configuration file is incorrect. The start bit and end bit are fixed at [8*RegisterBytes-1:0] when the word register is defined, which is related to the value of **RegisterBytes** in the **Global** field.

Solution

If the value of **RegisterBytes** in the **Global** field is 1, the start bit and end bit are fixed at [7:0] in the **Word Register** field.

10.2 What Do I Do If "Invalid Word Register content : bit count error at line xx" Is Displayed When the Configuration File Is Being Loaded?

Problem Description

"Invalid Word Register content : bit count error at line xx" is displayed when the configuration file is being loaded.

Cause Analysis

The **Word Register** settings are incorrect in the configuration file. The number of bits after the offset bit field address is incorrect, for example, [7:0]=32. In this case, an error occurs.



Solution

The correct configuration is [7:0]=8.

10.3 What Do I Do If "Segment name in the file is not exist in the register area" Is Displayed After Write by file Is Selected from the Shortcut Menu of the Register Pane?

Problem Description

"Segment name in the file is not exist in the register area" is displayed after **Write by file** from the shortcut menu of the **Register** pane is selected.

Cause Analysis

The segment name specified in the file does not exist in the current segment address list of the **Register** pane.

Solution

Add the corresponding segment in the **Register** pane and perform the operation again.

10.4 What Do I Do If "please check register type at first line" Is Displayed After Write by file Is Selected from the Shortcut Menu of the Register Pane?

Problem Description

"please check register type at first line" is displayed after **Write by file** from the shortcut menu of the **Register** pane is selected.

Cause Analysis

The register type specified in the first row of the file to be written is inconsistent with the type of registers in the **Register** pane.

Solution

Change the register type in the **Setting** menu, or change the register type in the file to the type of registers in the **Register** pane (you are advised not to use the latter method unless you are familiar with the file format).



10.5 What Do I Do If the Register List Is Not Read or Is Read Incompletely When the Register Type Is Word Register But No Error Information Is Displayed When Write by file Is Selected?

Problem Description

When the register type is **Word Register**, select **Write by file** from the shortcut menu of the **Register** pane. No error information is displayed. However, the register list is not read or is read incompletely.

Cause Analysis

The register definition in the written file mismatches the type of registers in the **Register** pane.

Solution

Ensure that the start bit and end bit of the offset bit field address for all registers in the read target file is $8 * \text{RegisterBytes} - 1$. For example, if the current **RegisterBytes** is **1**, the offset address must be **[7:0]=8** in the target file.



A

Acronyms and Abbreviations

A

AGC automatic gain control

B

Bin binary

C

CCM constant coding and modulation

D

Dec decimal

DDR double data rate

F

FPGA field programmable gate array

FFT fast Fourier transform

H

Hex hexadecimal

HDMI high definition multimedia interface

I

I2C inter-integrated circuit

Q

QPSK quadrature phase shift keying

S



SNR signal-to-noise ratio