Using TWR-56f8400 board with FreeMASTER via USBTAP interface

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool that can display multiple variables changing over time on an oscilloscope-like display, or show the data in text form.

FreeMASTER installer is available at:

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FREEMASTER&fpsp=1&tab=Design_Tools_Tab

Besides the FreeMASTER tool, user also needs to install its communication driver which implements the serial interface between the target application and the host PC.

The serial interface covers the UART/SCI and CAN communication for all supported devices,

the EOnCE/JTAG communication for 56F8xxx family and the USB interface for some platforms.

FreeMASTER communication driver is available at the same link in above.

Taking TWR-56F8400 as an example, this document shows how to create a project in CodeWarrior v10.6 and get its global data visualization in FreeMaster through USBTAP.

This example shows three global variables (test8, test16, test32) in 'Variable Watch Window' of FreeMASTER , displays the changing curve of test8 in scope, and displays test16 as curve in Recorder. When the application receives the command 0x1 from FreeMASTER, it re-initializes test8 to 128. The steps for creating CW and FreeMASTER project are listed as below:

----

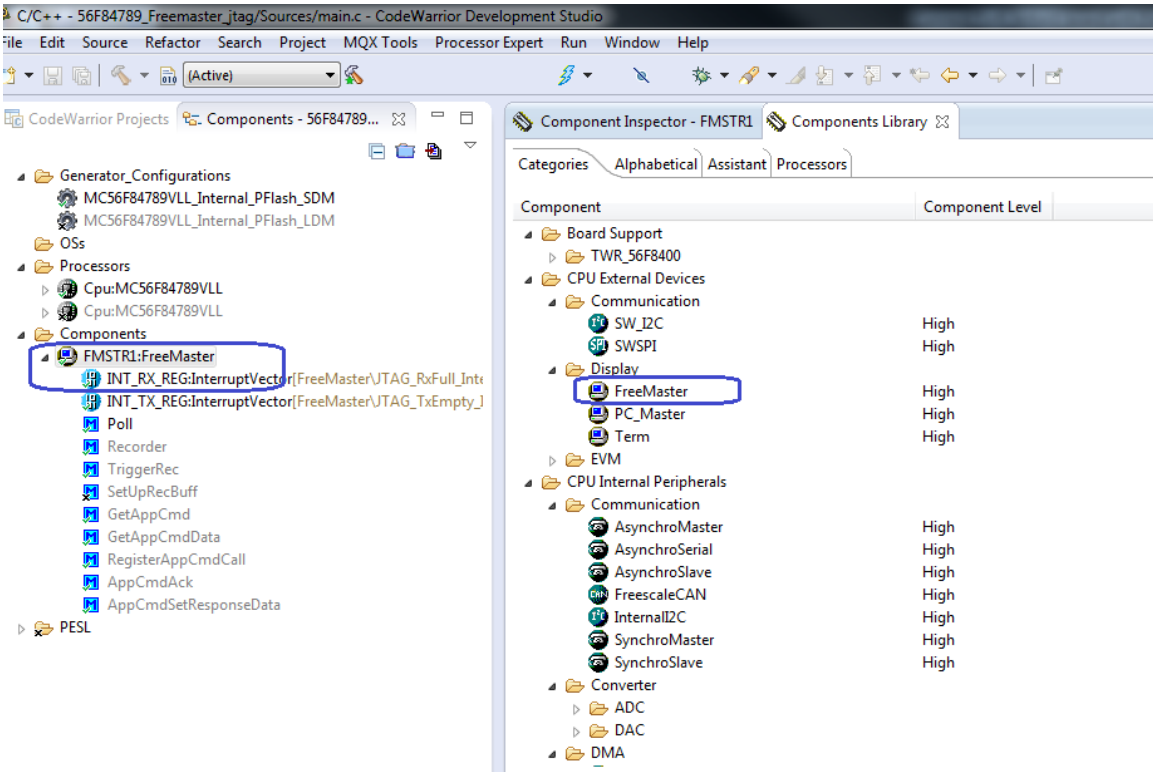
1. Create CodeWarrior Project

1.1 Add PE FreeMaster component

When creating the application project, please enable the Processor Expert option. The FreeMaster & PC_Master component included in PE makes it very easy to use FreeMASTER in applications.

Drag FreeMaster Component into the project which is under Component Categories 'CPU External Device -> Dispaly'.

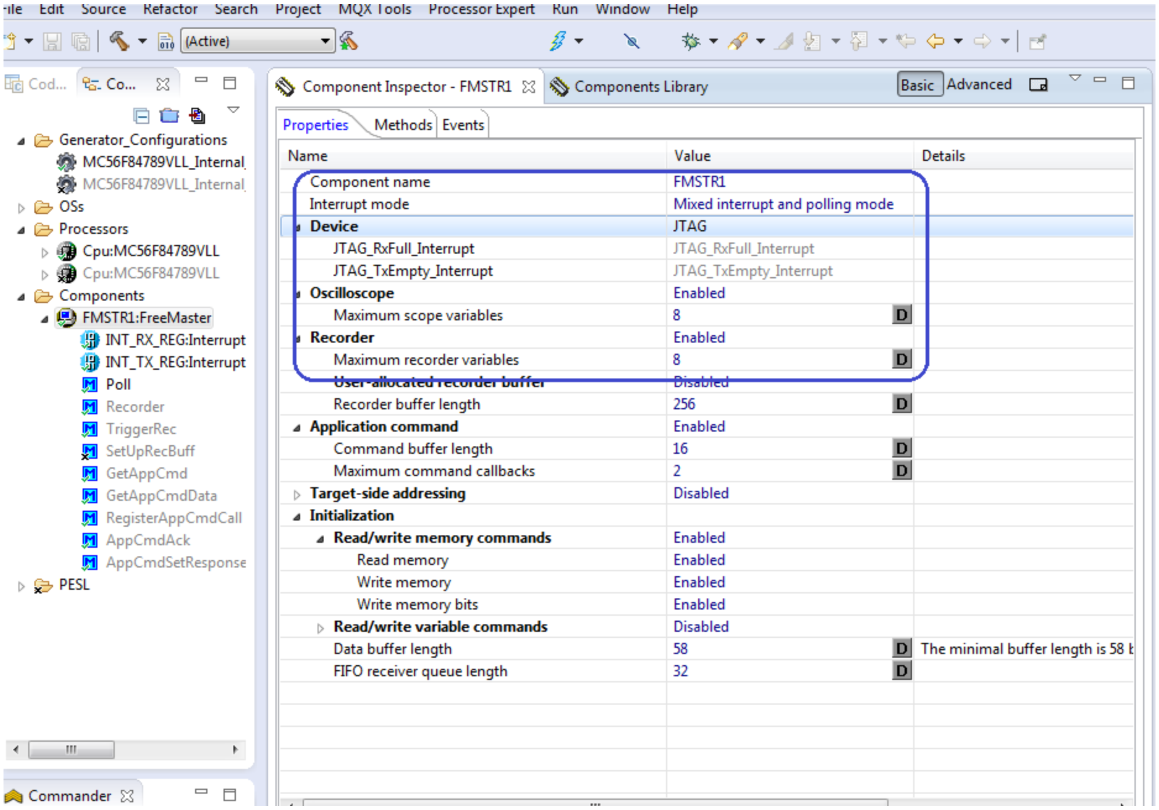
PC_Master component is listed next to Freemaster. User can choose either of them if he would use SCI connection. For JTAG connection, user can only choose FreeMaster.



The picture below shows a typical configuration in the Component Inspector. It sets the 'Device' as 'JTAG', the interrupt mode can be completely 'Interrupt-Driven', or 'Mixed Interrupt and Polling Modes', or 'Completely Poll-driven'.

The feature "Oscilloscope" is to display real-time graph of variable values. It enables up to eight variables to be read at once on a single request from the host. This is an optional feature.

The feature "Recorder" defines a standard way of how to select up to eight variables on the target whose values are periodically recorded into dedicated on-board memory buffer. After the data sampling is stopped, the data buffer is downloaded to the host and displayed as a graph. This is also an optional feature and may be disabled if Recorder is not needed in the application.



Name	Value	Details
Component name	FMSTR1	
Interrupt mode	Mixed interrupt and polling mode	
Device	JTAG	
JTAG_RxFull_Interrupt	JTAG_RxFull_Interrupt	
JTAG_TxEmpty_Interrupt	JTAG_TxEmpty_Interrupt	
Oscilloscope	Enabled	
Maximum scope variables	8 D	
Recorder	Enabled	
Maximum recorder variables	8 D	
User-allocated recorder buffer	Disabled	
Recorder buffer length	256 D	
Application command	Enabled	
Command buffer length	16 D	
Maximum command callbacks	2 D	
Target-side addressing	Disabled	
Initialization		
Read/write memory commands	Enabled	
Read memory	Enabled	
Write memory	Enabled	
Write memory bits	Enabled	
Read/write variable commands	Disabled	
Data buffer length	58 D	The minimal buffer length is 58 b
FIFO receiver queue length	32 D	

The help document for Freemaster bean is under:

C:\Freescale\CW MCU v10.6\MCU\ProcessorExpert\DOCs\FreescaleSDK\Beans\FreeMaster

1.2 Call FreeMaster functions in application.

Call FMSTR_Poll() during the "idle" time in the main application loop. This function initializes internal variables of the FreeMASTER driver and enables the communication interface.

Call FMSTR1_TriggerRec()during the "idle" time in the main application loop.   A typical place to call the recorder routine is at a timer or PWM interrupt, but it can be anywhere the user wants the variable values to be sampled. The attached example project makes a FMSTR_Recorder call in the main application loop.

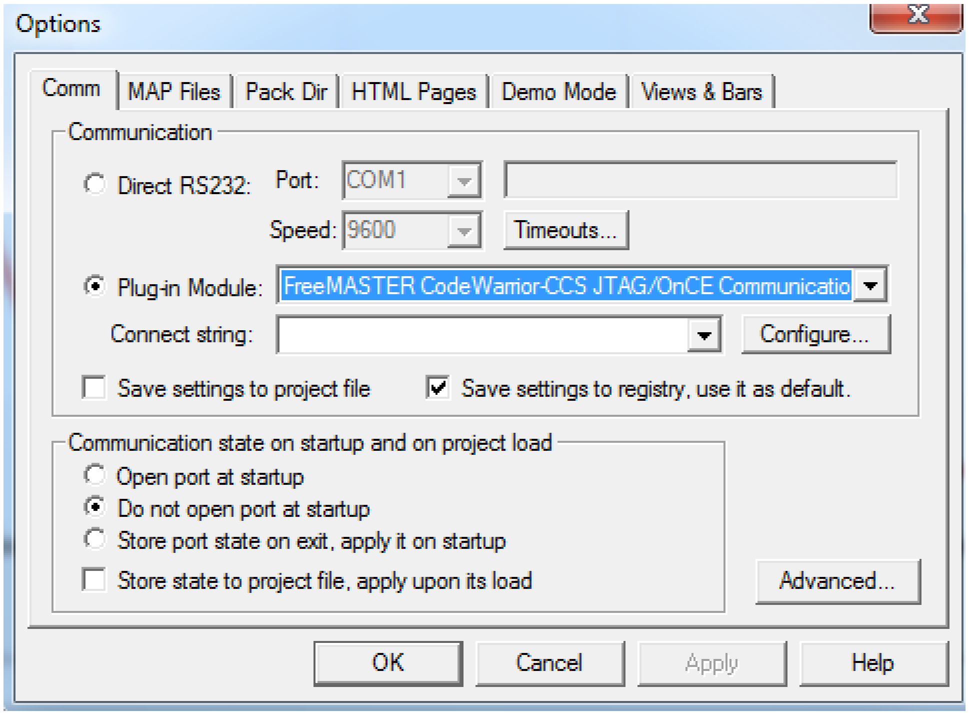
In the beginning of main application, call FMSTR_RegisterAppCmdCall()to register callback handler for application Command 0x01. And in the callback function, re-initialize test8 to 128.

In the main application loop, call FMSTR_GetAppCmd() to check if a new command has been received. If yes, and the command has been processed, the application calls FMSTR_AppCmdAck() to response to host.

2. Create FreeMASTER Project

Open FreeMASTER software, and click on ‘File’ Menu-> New Project, then open its option settings from ’Project’ Menu-> Options.

Under ‘Comm’ tab, enable ‘Plug-in Module’, and choose ‘FreeMASTER CodeWarrior-CCS JTAG/OnCE Communication Plugin’.



Options

Communication

○ Direct RS232: Port: COM1 Speed: 9600

● Plug-in Module: FreeMASTER CodeWarrior-CCS JTAG/OnCE Communicatio

Connect string:

□ Save settings to project file ☑ Save settings to registry, use it as default.

Communication state on startup and on project load

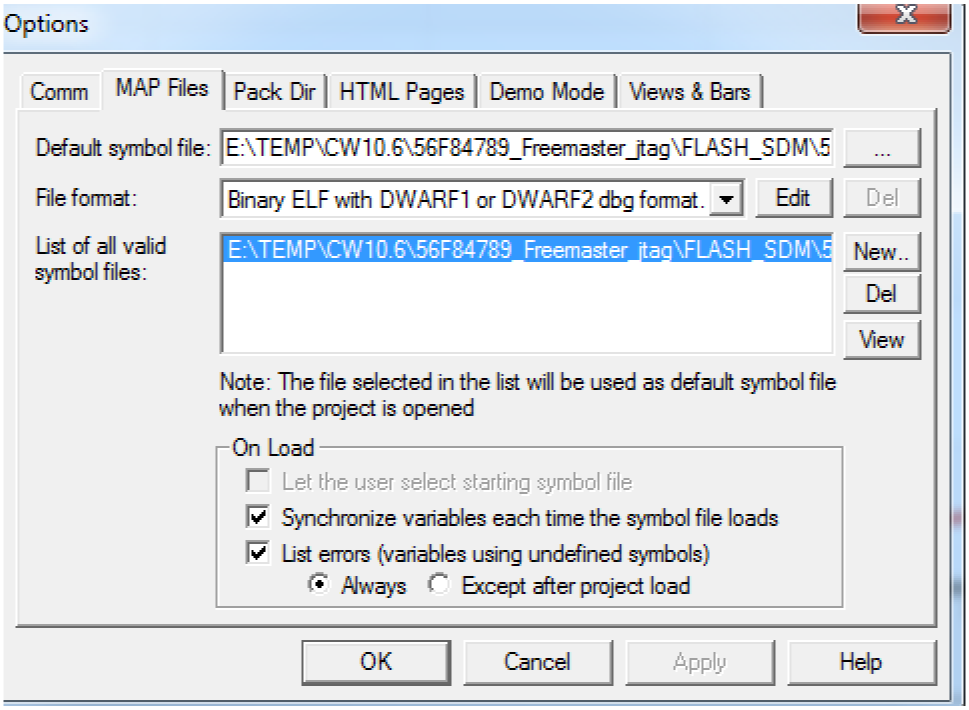
○ Open port at startup

● Do not open port at startup

○ Store port state on exit, apply it on startup

□ Store state to project file, apply upon its load

Under ‘MAP Files’ tab, choose the ELF file of CodeWarrior project, and set File format as ‘Binary ELF with DWARF1or DWARF2 dbg format’.



Default symbol file: E:\TEMP\CW10.6\56F84789_Freemaster_jtag\FLASH_SDM\5

File format: Binary ELF with DWARF1 or DWARF2 dbg format.

List of all valid symbol files: E:\TEMP\CW10.6\56F84789_Freemaster_jtag\FLASH_SDM\5

Note: The file selected in the list will be used as default symbol file when the project is opened

On Load

□ Let the user select starting symbol file

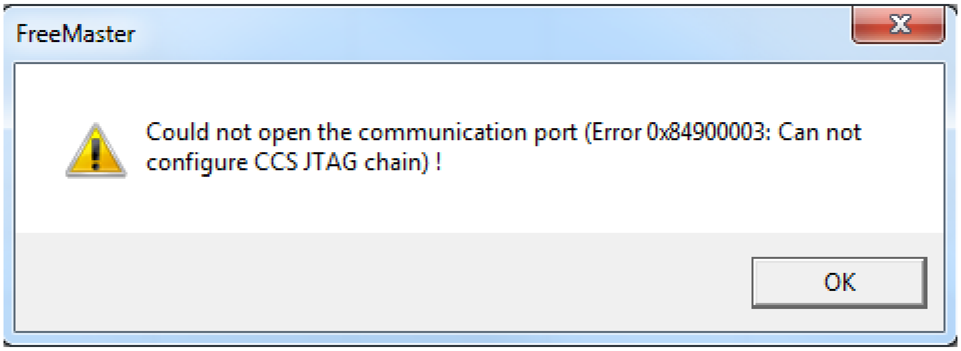
☑ Synchronize variables each time the symbol file loads

☑ List errors (variables using undefined symbols)

● Always ○ Except after project load

3. Connect with the Board

After creating the project, click on the red “Stop/Start” button or ‘File’ menu-> Start Communication. If an error popped up as below, user needs to manually open and configure CCS.

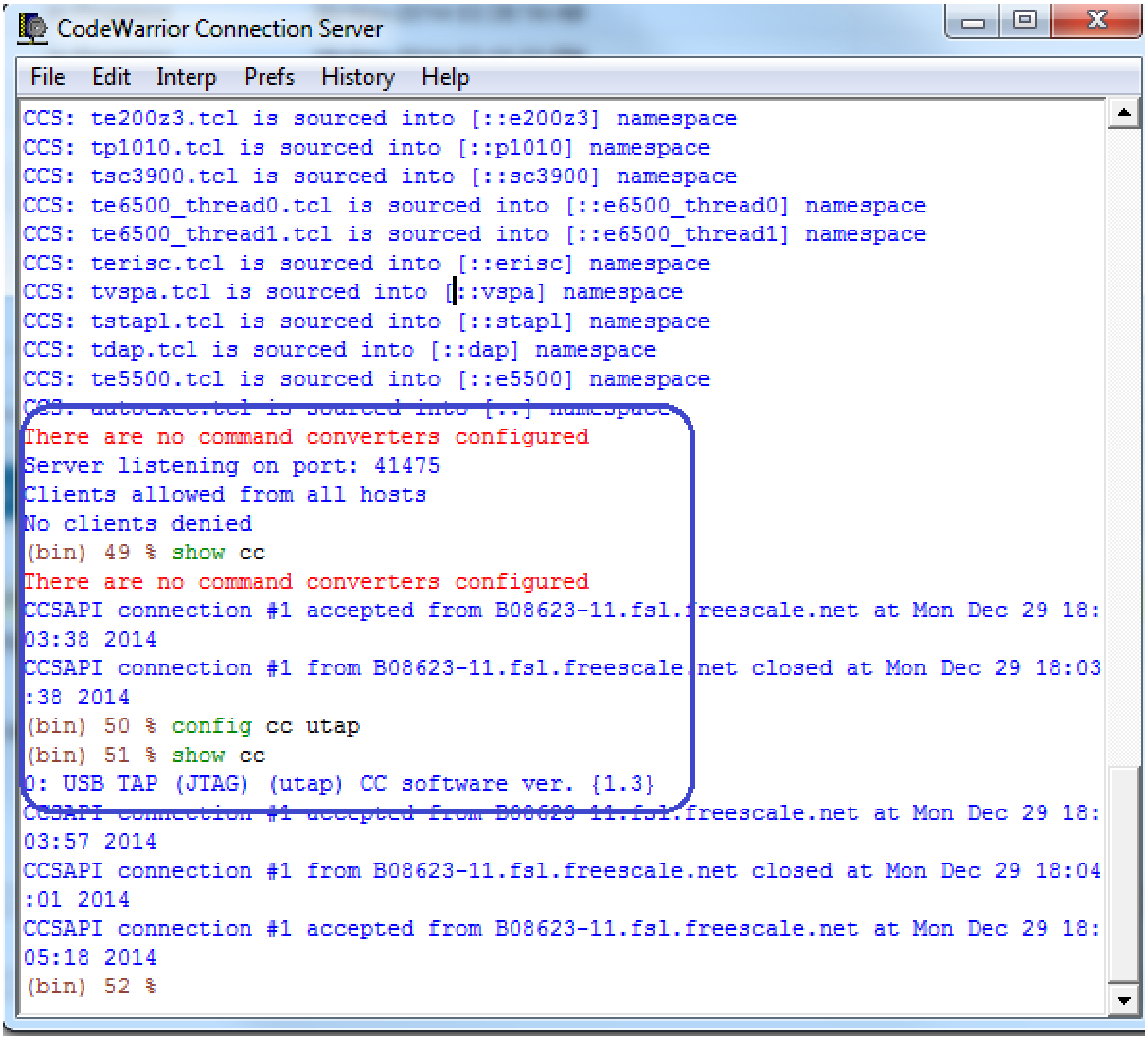


Could not open the communication port (Error 0x84900003: Can not configure CCS JTAG chain) !

FreeMASTER make of CCS to handle the communication, so CCS must be running when connect a FreeMASTER project.

The CCS.exe is under:
C:\Freescale\CW MCU v10.6\MCU\ccs\bin

Please double click ‘CCS.exe’ to open it, and on the console, key in the command “config cc utap”, then click on the red “Stop/Start” button to restart the communication.



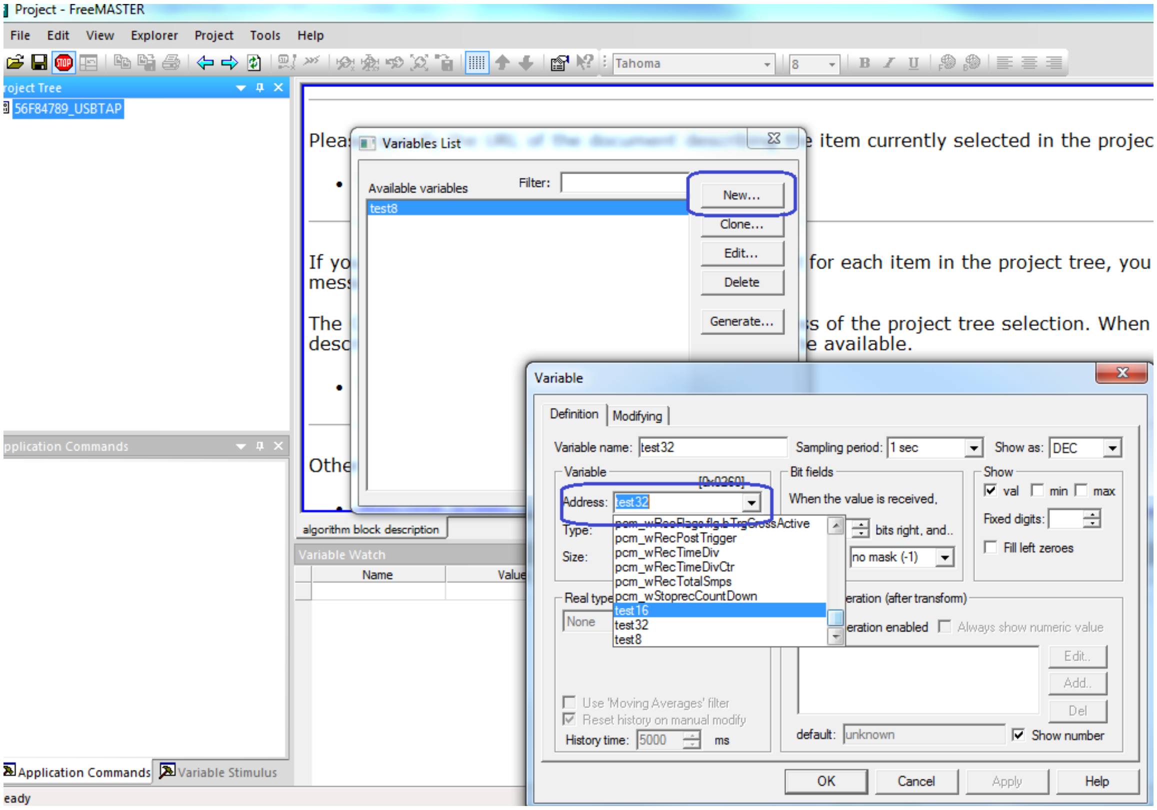
```
CCS: te200z3.tcl is sourced into [::e200z3] namespace
CCS: tp1010.tcl is sourced into [::p1010] namespace
CCS: tsc3900.tcl is sourced into [::sc3900] namespace
CCS: te6500_thread0.tcl is sourced into [::e6500_thread0] namespace
CCS: te6500_thread1.tcl is sourced into [::e6500_thread1] namespace
CCS: terisc.tcl is sourced into [::erisc] namespace
CCS: tvspa.tcl is sourced into [::vspa] namespace
CCS: tstapl.tcl is sourced into [::stapl] namespace
CCS: tdap.tcl is sourced into [::dap] namespace
CCS: te5500.tcl is sourced into [::e5500] namespace
CCS: autoexec.tcl is sourced into [::] namespace
There are no command converters configured
Server listening on port: 41475
Clients allowed from all hosts
No clients denied
(bin) 49 % show cc
There are no command converters configured
CCSAPI connection #1 accepted from B08623-11.fsl.freescale.net at Mon Dec 29 18:
03:38 2014
CCSAPI connection #1 from B08623-11.fsl.freescale.net closed at Mon Dec 29 18:03
:38 2014
(bin) 50 % config cc utap
(bin) 51 % show cc
0: USB TAP (JTAG) (utap) CC software ver. {1.3}
CCSAPI connection #1 accepted from B08623-11.fsl.freescale.net at Mon Dec 29 18:
03:57 2014
CCSAPI connection #1 from B08623-11.fsl.freescale.net closed at Mon Dec 29 18:04
:01 2014
CCSAPI connection #1 accepted from B08623-11.fsl.freescale.net at Mon Dec 29 18:
05:18 2014
(bin) 52 %
```

4. Add monitor for variables

4.1 Add variables for monitor and data visualization

Please click on Project Menu-> Variables, the on the Variable List, please click “New…” button to add the Variables.

On the view of ‘Variables’, select the variable in ‘Address’ pull-down menu, and select for Type, Size, Sampling Period...



Plea… item currently selected in the projec

If yo… mess… for each item in the project tree, you

The… desc… s of the project tree selection. When … e available.

Othe

Available variables: test8

Filter:

Variable name: test32

Sampling period: 1 sec

Show as: DEC

Variable

[0x0260]

Address: test32

Type:

Size:

pcm_wRecFlags.flg.bTrgCrossActive
pcm_wRecPostTrigger
pcm_wRecTimeDiv
pcm_wRecTimeDivCtr
pcm_wRecTotalSmps
pcm_wStoprecCountDown
test16
test32
test8

Bit fields

When the value is received,

bits right, and..

no mask (-1)

Show

☑ val ☐ min ☐ max

Fixed digits:

☐ Fill left zeroes

Real type

None

eration (after transform)

eration enabled ☐ Always show numeric value

☐ Use 'Moving Averages' filter

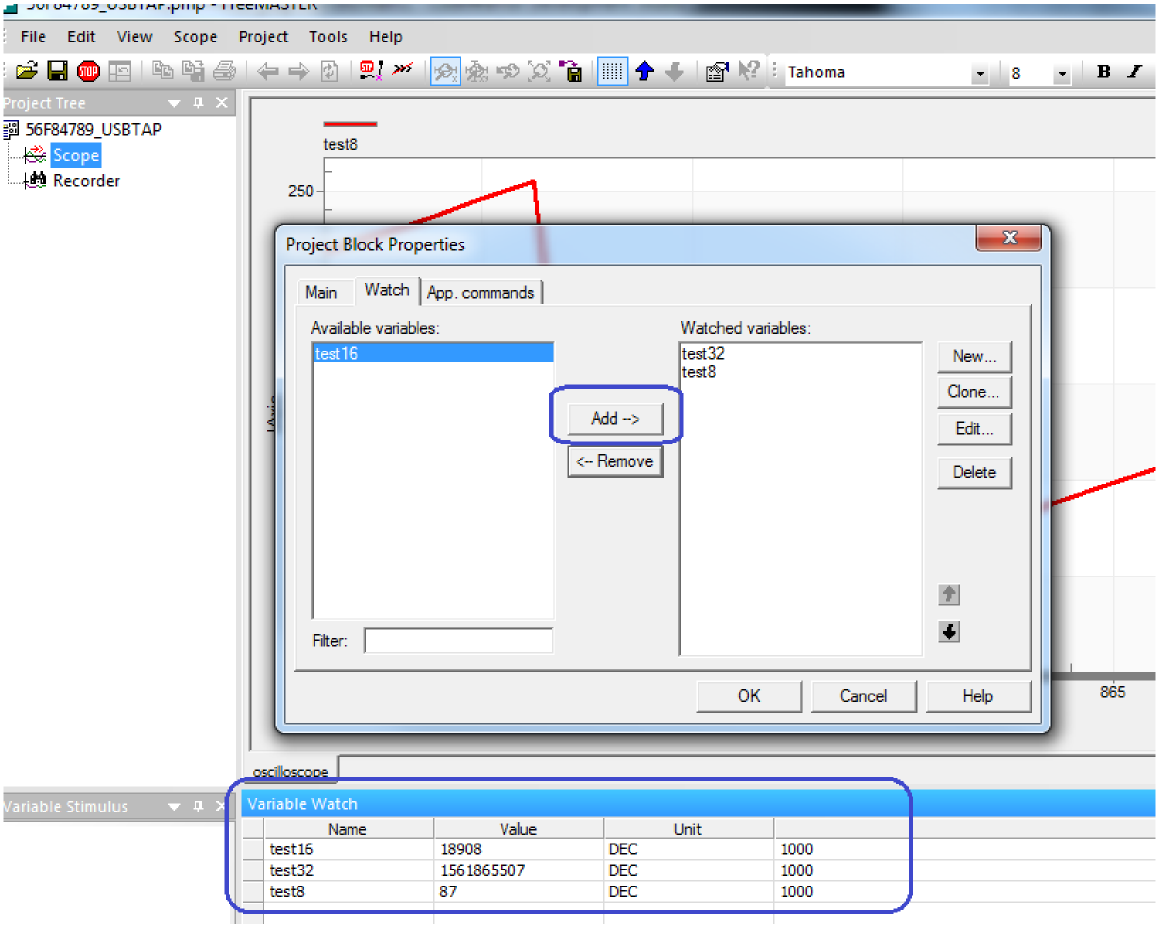
☑ Reset history on manual modify

History time: 5000 ms

default: unknown

☑ Show number

To add the above variables into ‘Variable Watch’ window, click on the blank line in ‘Variable Watch’ window, and select ‘Watch Properties’, then under ‘Watch’ tab, add the variables listed on the left column to the right.

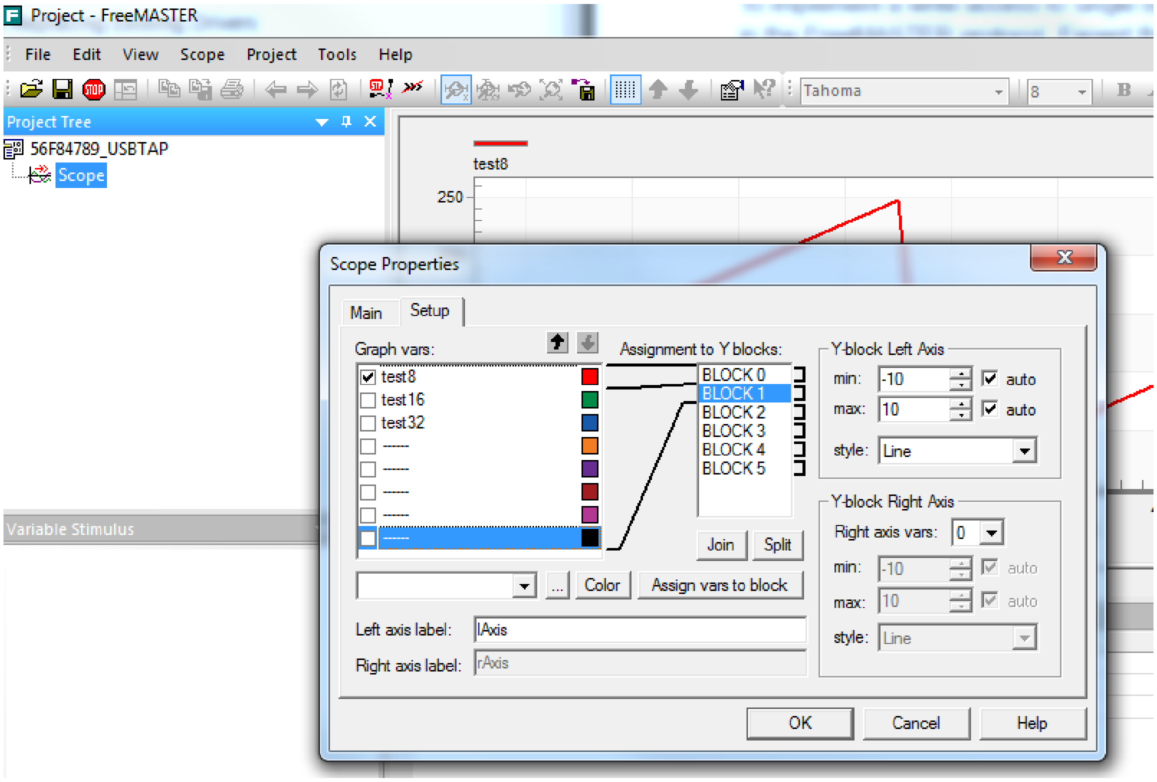


Name	Value	Unit	
test16	18908	DEC	1000
test32	1561865507	DEC	1000
test8	87	DEC	1000

4.2 Create Scope

Right click on the Project Name under ‘Project Tree’ view and select ‘Create Scope’.

On the Scope Properties, configure the parameters as user needs under ‘Main’ tab. Add the variables that user wants to monitor under ‘Setup’ tab, set their colors and axis as below:



Graph vars:

☑ test8 🟥
☐ test16 🟩
☐ test32 🟦
☐ ------
☐ ------
☐ ------
☐ ------
☐ ------

Left axis label: lAxis

Right axis label: rAxis

Y-block Left Axis

min: -10 ☑ auto

max: 10 ☑ auto

style: Line

Y-block Right Axis

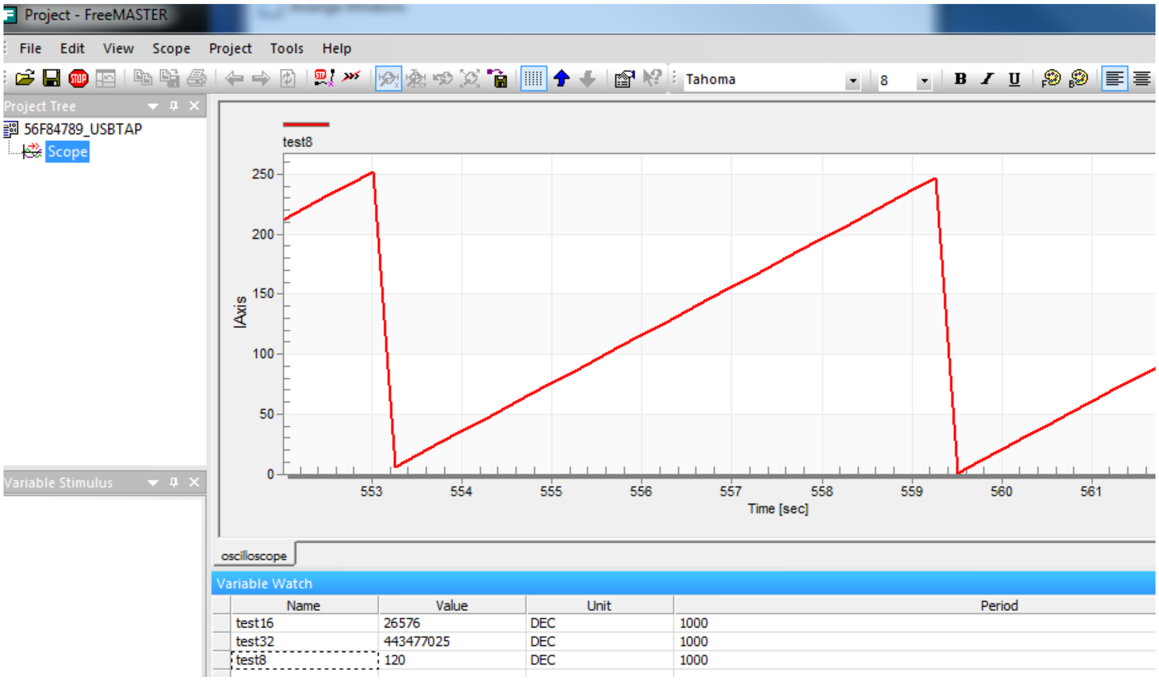
Right axis vars: 0

min: -10 ☑ auto

max: 10 ☑ auto

style: Line

Then click OK, the variable changing will display in Scope Window.



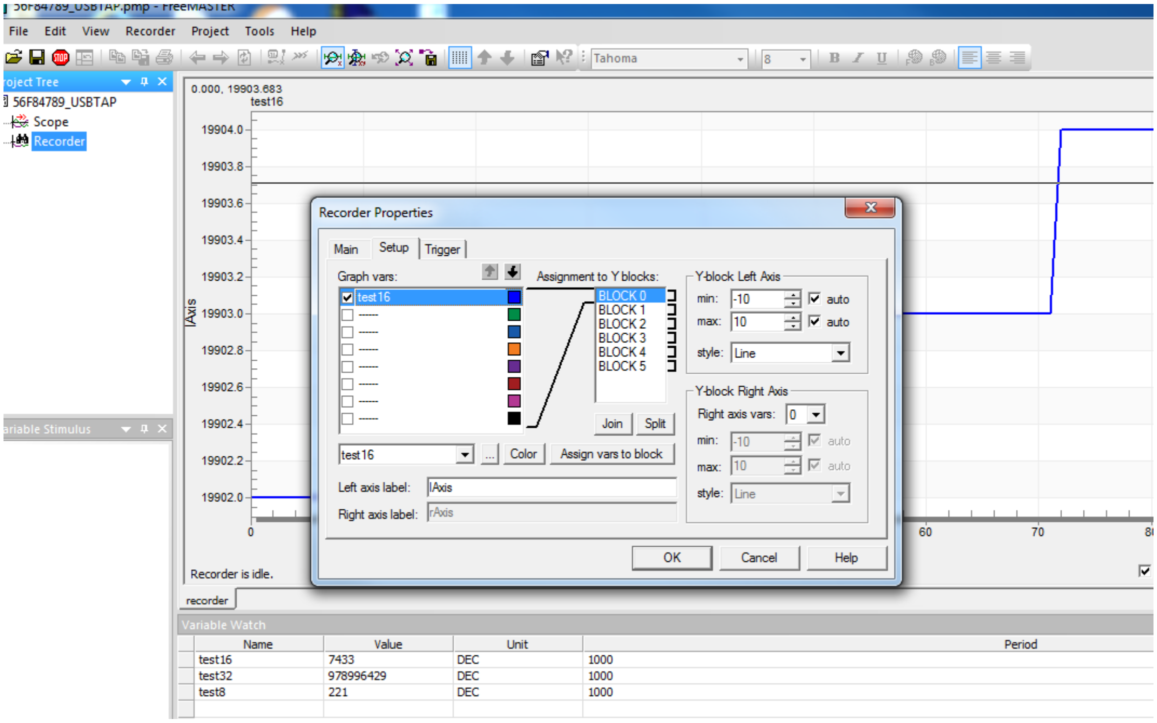
Variable Watch

Name	Value	Unit	Period
test16	26576	DEC	1000
test32	443477025	DEC	1000
test8	120	DEC	1000

4.3 Create Recorder

Right click on the Project Name under ‘Project Tree’ view and select ‘Create Recorder’.

On the Scope Properties, configure the parameters as user needs under ‘Main’ tab. Add the variables that user wants to monitor under ‘Setup’ tab, set their colors and axis as below:



Graph vars: ☑ test16

Y-block Left Axis
min: -10 ☑ auto
max: 10 ☑ auto
style: Line

Y-block Right Axis
Right axis vars: 0
min: -10 ☑ auto
max: 10 ☑ auto
style: Line

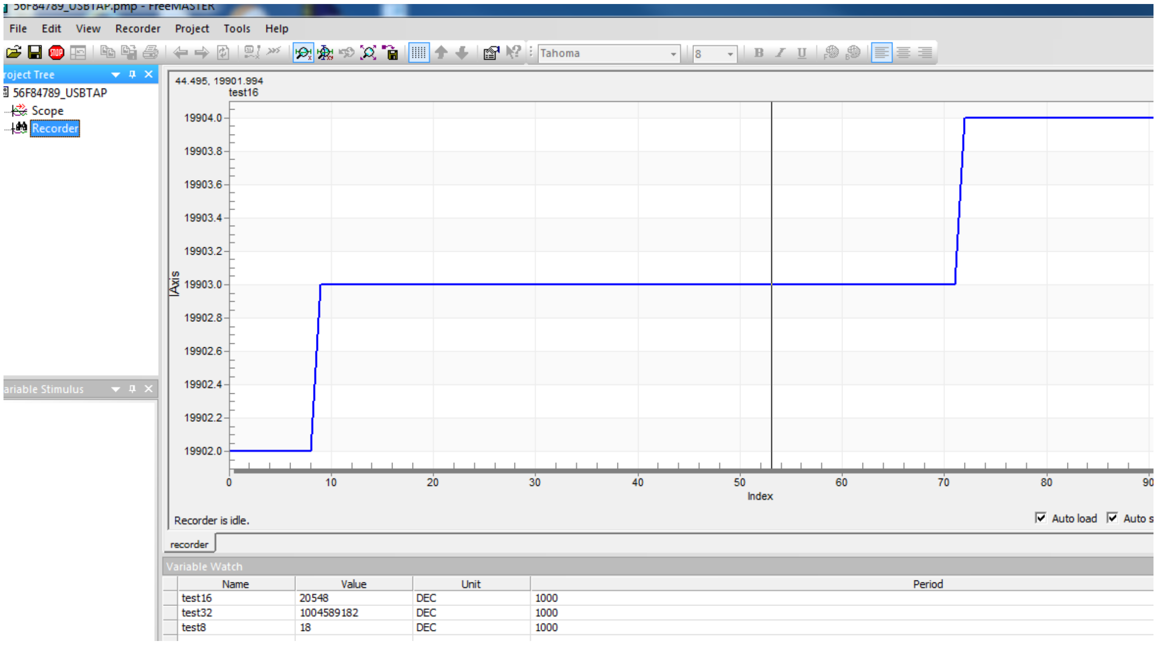
test16

Left axis label: lAxis
Right axis label: rAxis

Variable Watch

Name	Value	Unit	Period
test16	7433	DEC	1000
test32	978996429	DEC	1000
test8	221	DEC	1000

Then click OK, the variable changing will display in Recorder Window.

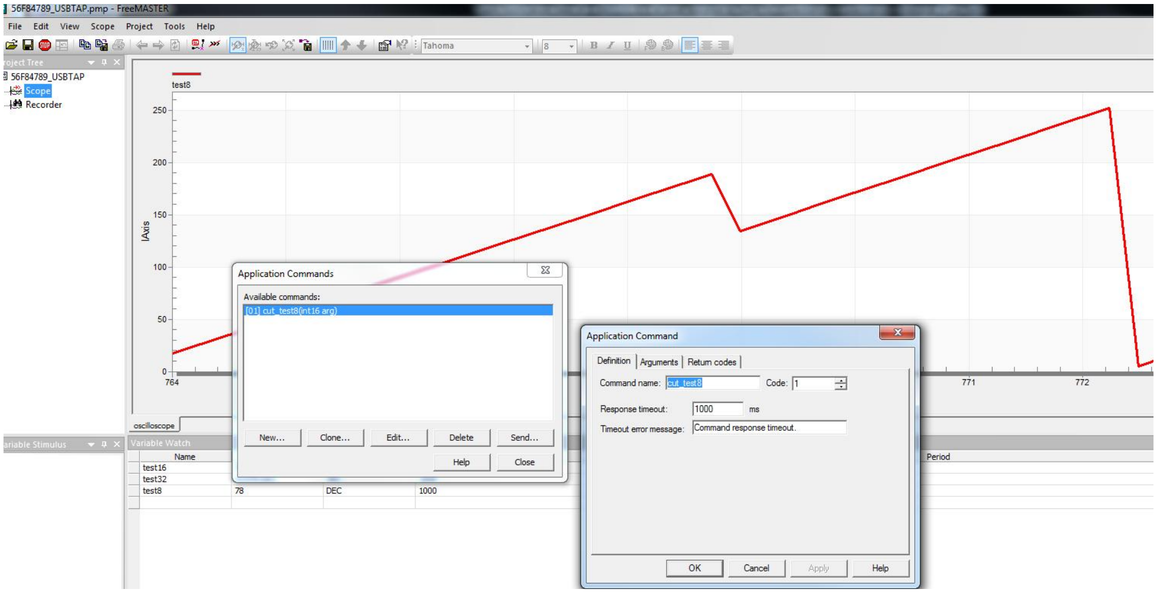


Variable Watch

Name	Value	Unit	Period
test16	20548	DEC	1000
test32	1004589182	DEC	1000
test8	18	DEC	1000

4.4 Add Command

Right click on the ‘Project’ Menu-> Command to create a new application command. The fill the name, code and data type for the command. This example uses the command code 0x01 to force the application to re- initializes ‘test8’.



Application Commands

Available commands:
[01] cut_test8(int16 arg)

Name			
test16			
test32			
test8	78	DEC	1000

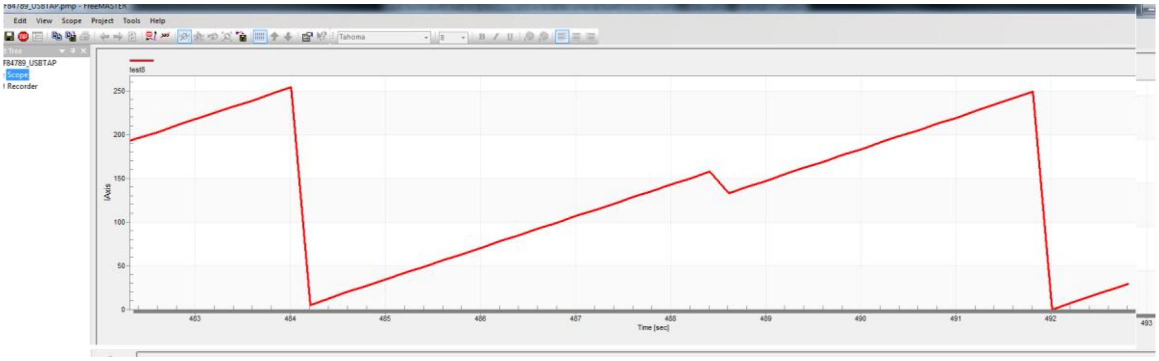
Application Command

Command name: cut_test8 Code: 1

Response timeout: 1000 ms

Timeout error message: Command response timeout.

After click on the ‘Send’ button in ‘Application Command’ window, the scope shows that ‘test8’ restart its increasing from 128.



The attached file includes both CodeWarrior and FreeMASTER project.