

KITPSCDEBUG evaluation board

Tracer for MC33816 and PT2000



Figure 1. KITPSCDEBUGEVM

Contents

1 Important notice.....	3
2 Getting started.....	4
2.1 Kit contents/packing list.....	4
2.2 Jump start.....	4
2.3 Required equipment and software.....	4
2.4 System requirements.....	4
3 Understanding the system.....	5
3.1 Block diagram.....	5
4 Getting to know the hardware.....	6
4.1 Board overview.....	6
4.2 Board description.....	6
5 Installing the software and setting up the hardware.....	7
5.1 Installing SPIGen freeware on the computer.....	7
5.2 Installing MC33PT2000 Developer Studio on the computer.....	8
5.3 Configuring the hardware.....	9
5.4 Step-by-step setup instructions for the tracer using the PT2000 IDE.....	10
5.5 Tracer waveform with the AN5186 example.....	17
6 Updating the tracer firmware.....	18
7 Troubleshooting.....	19
8 References.....	20
8.1 Support.....	20
8.2 Warranty.....	20
9 Revision history.....	21

1 Important notice

NXP provides the enclosed product(s) under the following conditions:

This evaluation kit is intended for use of ENGINEERING DEVELOPMENT OR EVALUATION PURPOSES ONLY. It is provided as a sample IC pre-soldered to a printed circuit board to make it easier to access inputs, outputs, and supply terminals. This EVB may be used with any development system or other source of I/O signals by simply connecting it to the host MCU or computer board via off-the-shelf cables. This EVB is not a Reference Design and is not intended to represent a final design recommendation for any particular application. Final device in an application will be heavily dependent on proper printed circuit board layout and heat sinking design as well as attention to supply filtering, transient suppression, and I/O signal quality.

The goods provided may not be complete in terms of required design, marketing, and or manufacturing related protective considerations, including product safety measures typically found in the end product incorporating the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge. In order to minimize risks associated with the customers applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards. For any safety concerns, contact NXP sales and technical support services.

Should this evaluation kit not meet the specifications indicated in the kit, it may be returned within 30 days from the date of delivery and will be replaced by a new kit.

NXP reserves the right to make changes without further notice to any products herein. NXP makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typical", must be validated for each customer application by customer's technical experts.

NXP does not convey any license under its patent rights nor the rights of others. NXP products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the NXP product could create a situation where personal injury or death may occur.

Should the buyer purchase or use NXP products for any such unintended or unauthorized application, the buyer shall indemnify and hold NXP and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that NXP was negligent regarding the design or manufacture of the part. NXP™ and the NXP logo are trademarks of NXP Semiconductors. All other product or service names are the property of their respective owners.

© 2016 NXP B.V.

2 Getting started

2.1 Kit contents/packing list

The KITPSCDEBUGEVM contents include:

- Assembled and tested evaluation board in an anti-static bag
- Quick Start Guide, Analog Tools
- Warranty card
- USB cable to connect Tracer board to computer

2.2 Jump start

NXP's analog product development boards help to easily evaluate NXP products. These tools support analog mixed signal and power solutions including monolithic ICs using proven high-volume SMARTMOS mixed signal technology, and system-in-package devices utilizing power, SMARTMOS and MCU dies. NXP products enable longer battery life, smaller form factor, component count reduction, ease of design, lower system cost and improved performance in powering state of the art systems.

- Go to www.nxp.com/KITPSCDEBUG
- Review the Tool Summary Page
- Look for



Jump Start Your Design

- Download documents, software, and other information

Once the files are downloaded, review the user guide in the bundle. The user guide includes setup instructions, BOM and schematics. Jump start bundles are available on each tool summary page with the most relevant and current information. The information includes everything needed for design.

2.3 Required equipment and software

To use this kit, you need:

- PT2000 or MC33816 EVB, or ECU using MC33816 or PT2000
- Coaxial cable to connect tracer to DBG pin on EVB
- PT2000 last IDE version for PT2000 or MC33816
- SPIGen 7.0 or greater www.nxp.com/spigen

2.4 System requirements

The kit requires the following to function properly with the software:

- USB-enabled PC with Windows® 7 or higher

3 Understanding the system

The PT2000 and MC33816 Developer Studio (IDE) software, and the KITPSCDEBUGEVM are used to configure, receive, and decode the trace codes coming from the device under test. The Tracer board only needs two pins (DBG and GND) to receive the trace code coming from the PT2000 or MC33816. This document explains the steps to correctly use the KITPSCDEBUGEVM.

3.1 Block diagram

The high level system block diagram (Figure 2) outlines the way the NXP tracer is used.



Figure 2. EVB block diagram

3.1.1 Device features

This debug board features the following NXP products:

Table 1. Device Features

Device	Description	Features
PT2000 MC33816	Programmable Solenoid Controller Tracer	DBG pin of each devices MC33816 and PT2000 used to make a trace of the code

4 Getting to know the hardware

4.1 Board overview

The KITPSCDEBUGEVM is an easy-to-use circuit board allowing the user to debug the microcode programmed in the PT2000 or MC33816. A PC communicates to the tracer through a USB port. The NXP PT2000/MC33816 Developer Studio (IDE) is used to configure the device under test and also the tracer (see tracer tab). The SPIGen program (version 7.0 and above) with a KL25Z on the MC33816/PT2000 EVB provides the user interface to the PT2000 SPI port and allows the user to start the trace on the IC (similar things can be done by an MCU directly on the ECU).

4.2 Board description

The KITPSCDEBUGEVM is the interface between the PT2000/MC33816 and the device under test. LEDs are used to let the user know which step they are in or if the RAM is full.



Figure 3. Board description

Table 2. Board description

Name	Description
USB connector	Use to communicate between the PT2000/MC33816 IDE and tracer
Coaxial connector	To be connected to the DBG pin and GND. A BNC Jack to Hook Clip can be used for this purpose (ex: Pomona 3788)
Green LED	LED is ON when calibration sequence is received, therefore the tracer board is ready to receive the trace codes sequence
Yellow LED	LED is ON when tracer RAM is full
Red LED	LED is ON when tracer is waiting for calibration pulses

5 Installing the software and setting up the hardware

5.1 Installing SPIGen freeware on the computer

The latest version of SPIGen is designed to run on Windows 7 or later operating systems. To install the software, go to www.nxp.com/analogtools and select the kit. Click on the link to open the corresponding Tool Summary page. Look for “Jump Start Your Design”. Download the SPIGen software as well as the associated configuration file to the computer desktop.

Run the install program from the desktop. The Installation Wizard conducts the rest of the process.

The GUI is shown in Figure 4. The text at the top is the name of the configuration file loaded. The left side panel displays grouped user interfaces. The interfaces in the pre-installed PT2000 folder pertain specifically to the board under discussion. The process of loading the configuration file has assigned a list of “Extra Pins”, as well as a list of “Quick Commands”, all of which are board-specific.

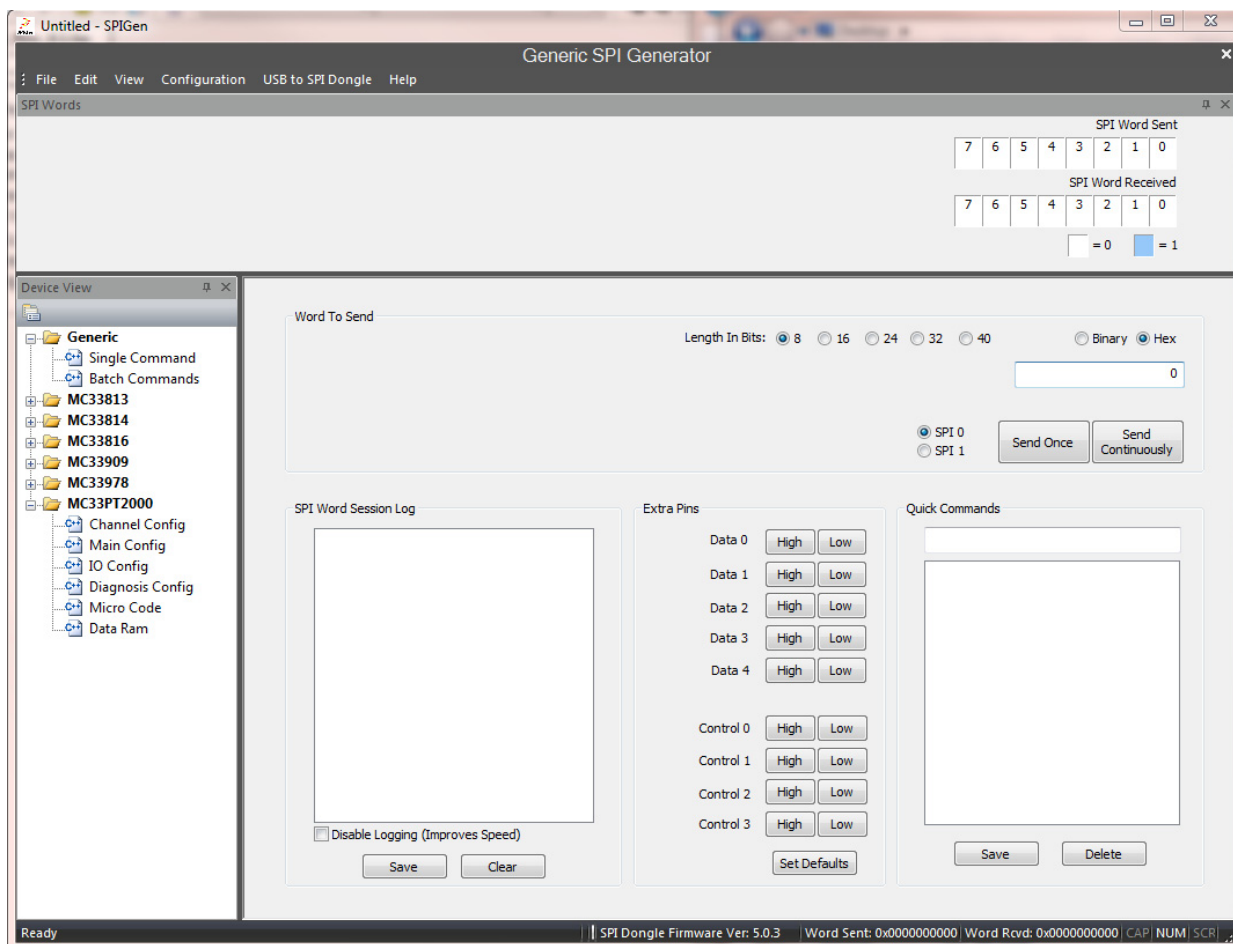


Figure 4. SPIGen GUI

5.2 Installing MC33PT2000 Developer Studio on the computer

The MC33PT2000 Developer Studio is compatible with Microsoft Windows 7 and later operating systems, and runs on both 32-bit and 64-bit versions of Windows. Download the installer from the PT2000 web site and run it on the computer to install the program.

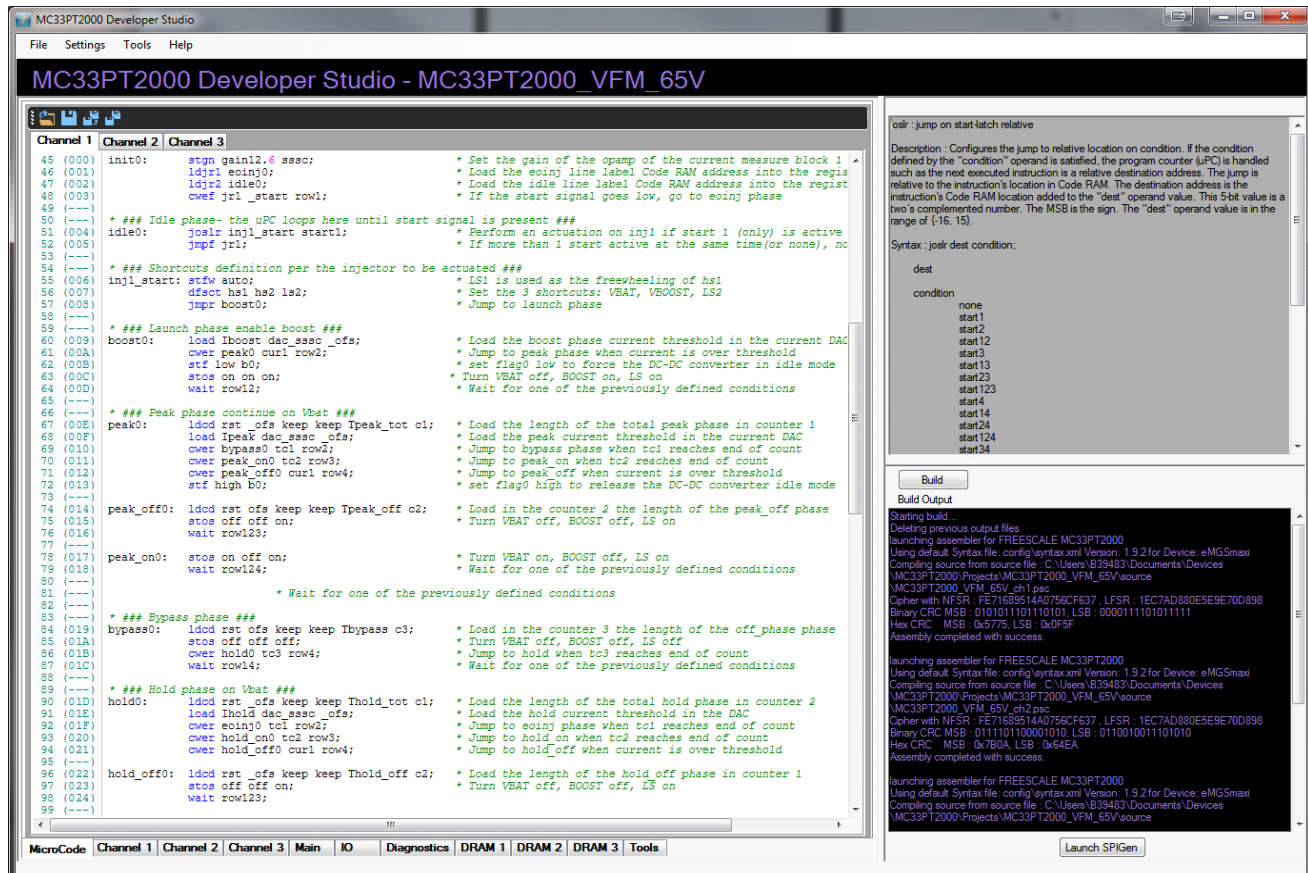


Figure 5. Developer Studio GUI

5.3 Configuring the hardware

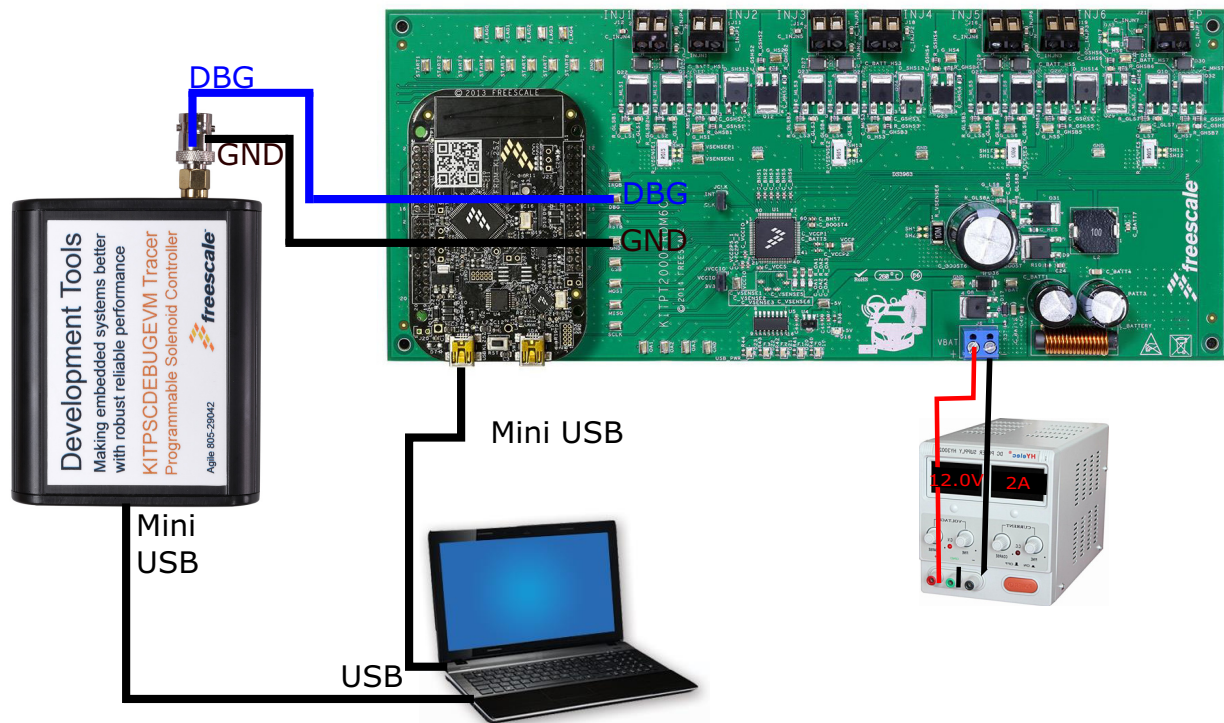


Figure 6. KITPSCDEBUGEVM board setup

5.4 Step-by-step setup instructions for the tracer using the PT2000 IDE

There are three steps when doing a trace. First prepare the PT2000 or MC33816 device under test, configure the tracer, and then enable the trace. The example focuses only on the PT2000, but the setup is similar for the MC33816. Only the SPI register addresses are different. Before proceeding to the next step, the MC33816/PT2000 microcode needs to be written and built in the IDE, and the device needs to be flashed. Refer to the IDE user guide for more information.

5.4.1 PT2000/MC33816 trace description

The trace sequence is composed of five steps:

1. **Calibrate:** Communication between the PT2000/MC33816 and the tracer is asynchronous, since no clock line is shared. To allow a good synchronization between the tracer and the device under test, as soon as the trace is enabled, a burst of eight clock cycles (with a frequency half of the internal cksys) is sent to the DBG pin. Following this, the DBG pin is kept low for at least one ck clock period. When this calibration is done, the Green LED on the tracer is ON.
2. **Start:** The trace operation consists of reconstructing the execution path from the codes the PT2000/MC33816 transmitted to the tracer. However, these codes only describe the variation of the uPC value. To obtain the actual execution path, the trace operation must start from a point in the code known to the tracer. This is done by setting the trace_start SPI register using the IDE or SPIGEN. If the trace operation is enabled and the trace unit is in idle state, when the uPC value of the selected microcore reaches the start, the trace is sent through the DBG pin.
3. **Trace:** Each cycle transmits a 4-bit code value identifying which path was taken by the code execution
4. **Stop:** The trace operation is not meant to last indefinitely. It is possible to define a “stop” address using the SPI register trace_stop. If during the precedent phase (trace) the uPC reaches the stop address, the trace_unit goes to the following phase. If the stop is never reached, it is possible to turn OFF the trace by setting the trace enable bit to 0.
5. **Post Trigger:** It possible after the trace stop is reached to capture the trace operation for a fixed number of ck clock cycles. After this time has elapsed, the trace_unit goes to the idle state. If the post trigger bits are all set to 1, the trace stops only when the bit trace enable will be set to 0.

5.4.2 Step 1: PT2000/MC33816 trace configuration using IDE

This chapter explains how to configure the trace using the PT2000 IDE.

The user first needs to open an actual project with the microcode already written. Use the example already available on the web.

In this example, [AN5186](#) is used. The trace is only done on the channel 1 microcore 0, with a start point at the boost phase (code line 48h) and a stop point when end of injection (code line 71h) is reached.

```
(---) *****
(---) *                                     BOOST PHASE                                     *
(---) *****
(---) * ### Launch phase enable boost ###
(048) boostT0:  load Iboost dac_sssc_ofs;          * Load the boost phase current threshold in the current DAC
(049)          ldcd rst_ofs keep keep injMaxIBoost c3; * Start Boost Counter in case Iboost never reached
(04A)          ldcd rst_ofs keep keep injMinIBoost c2; * Start Boost Counter in case Iboost is reached too fast
(04B)          stf low BstFlag;                    * Set flag0 high to stop the DC-DC converter
(04C)          stos on on on;                       * Turn VBAT off, BOOST on, LS on
(04D)          endiags on on on on;                 * Enable auto diag
(---)
(04E)          cwer boostT1 tc2 row5;                * After injMinBoost go to Boost T2
(04F)          cwer boost_err0 ocur row2;            * Jump to error in case current is reached before the end of injMinBoost
(050)          wait row125;
(---)
(051) boostT1:  cwer peak0 ocur row2;                * Jump to peak phase when current is over threshold
(052)          cwer boost_err0 tc3 row5;            * Define Wait Table if actuation longer than injMaxGuard go to eoinj
(053)          wait row125;
(---)
(054) boost_err0: stf high BstFlag;                  * set flag0 high to release DCDC regulation
(055)          req1 0;                               * Go to software subroutine is fault detected in Boost phase, did not reach Iboost on ti
```

Figure 7. Boost phase as a start point for the trace (line 48h)

```

(---) *****
(---) *                               END OF INJECTION PHASE                               *
(---) *****
(---) * ### End of injection phase ###
(---)
(071) eoinj0:  stos off off off;          * Turn VEAT off, BOOST off, LS off
(072)          endiags diagoff;        * Disable auto diag
(073)          bias all off;          * Enable all biasing structures, kept ON even during actuation
(074)          stf high BstFlag;      * set flag0 to high to release the DC-DC converter idle mode
(075)          jmpf jr2;              * Go back to idle mode
(---) * ### End of Channel 1 - uCore0 code ###
(---)

```

Figure 8. End of injection phase as a stop point for the trace (line 71h)

When the microcore and start/stop point for the tracer are selected, specify them using the SPI registers:

- trace_start (address 1AAh): in this case 0x48
- trace_stop (address 1ABh): in this case 0x71
- trace_config (1ACh): where the microcore used for the trace and the value of the post trigger is selected.
In this example: 0x3 (uc select = 0 and post trigger length = 3 to capture until line (74h))

Figure 9 shows the setting used for this example.

trace_start - 1AAh															set all		clear all		0x0048									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
			Re-served							start address																		
trace_stop - 1ABh															set all		clear all		0x0071									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
			Re-served							stop address																		
trace_config - 1ACh															set all		clear all		0x0003									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
Trace enable				Re-served				uc select				Post trigger length																

Figure 9. PT2000 SPI registers to configured for the trace

Configuration of the device under test is now done. The same process can be done using an MCU or SPIGEN instead of the IDE. The trace execution is not started, since it is now necessary to configure the KITPSCDEBUG.

5.4.3 Step 2: KITPSCDEBUG configuration using IDE

For this configuration, the tab “Tracer” is used.

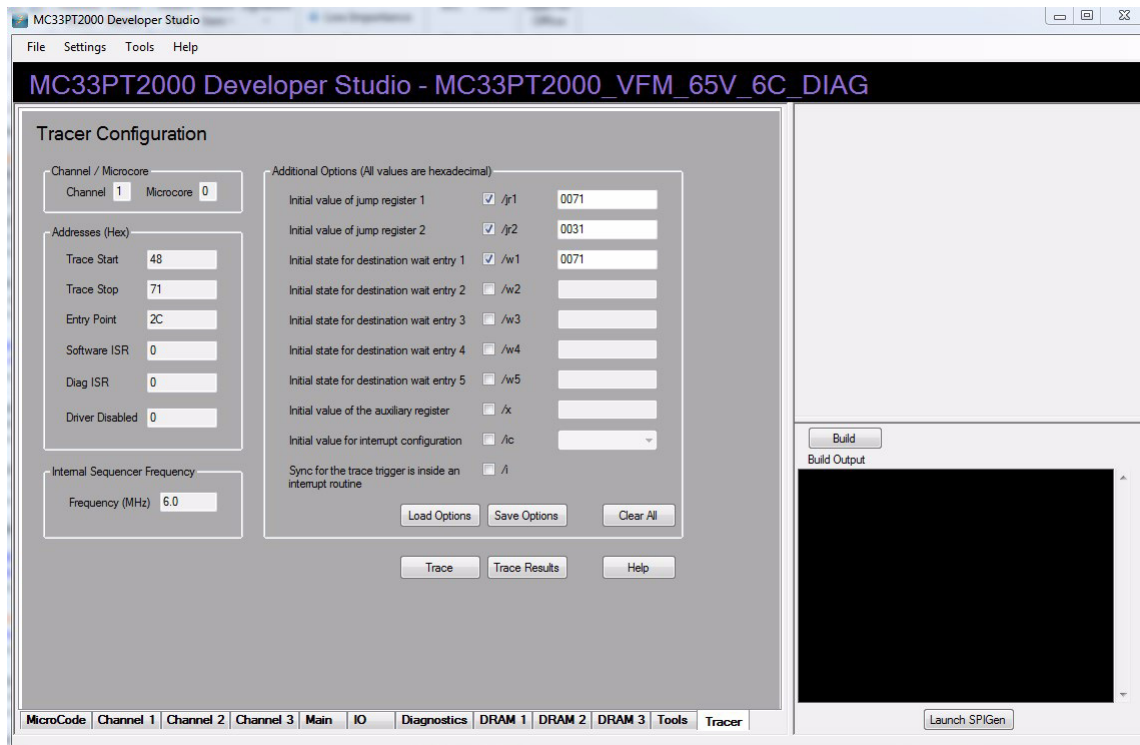


Figure 10. PT2000 trace tab configuration

As shown in Figure 10, the configuration on the left side is read only and comes directly from the previous register setting. However, the right column still needs to be configured.

If the trace is not starting at the first line of the code, it is necessary to tell the trace where the uPC jumps. For example, during the initialization phase (see Figure 11), jump1, jump2, and wait 1 are defined. Since the trace only starts during the boost phase on line 48h, it is mandatory to tell the tracer where to jump, if those conditions are met.

```
(---) * ## Initialization phase ##
(02C) init0:  stgn gain8.68 sssc;          * Set the gain of the opamp of the current measure block 1
(02D)         ldjr1 eoinj0;              * Load the eoinj line label Code RAM address into the register jr1
(02E)         ldjr2 idle0;              * Load the idle line label Code RAM address into the register jr2
(02F)         stirq high;                * If the start signal goes low, go to eoinj phase
(030)         cwef jr1_start row1;
```

Figure 11. Initialization phase wait and jump initialization

The following configuration is then necessary:

Figure 12. Tracer configuration for the AN5186 example

Once settings are complete, saving the option for reuse is possible. If this configuration is not set, an error will occur during the trace.

5.4.3.1 Options description

The following descriptions are options which can be sent to the tracer, depending on the trace start position in the code.

- /i: To be specified only if the sync for the trace trigger is inside an interrupt routine.
- /wX wait DestX: Specify the initial state for the destination wait entry X. If not specified, it starts as invalid at the beginning of the decoding operation.
- /jr1 jumpReg1: Specify the initial value of the jump register 1. It must be between 0 and 1022. If not specified, it starts as invalid at the beginning of the decoding operation.
- /jr2 jumpReg2: Specify the initial value of the jump register 2. It must be between 0 and 1022. If not specified, it starts as invalid at the beginning of the decoding operation.
- /x auxReg: Specify the initial value of the auxiliary register. It must be between 0 and 1022. If not specified, it starts as invalid at the beginning of the decoding. This option is needed if for example the trace is started inside a subroutine and the rfs instruction is executed.
- /ic conf: Specify the initial value for the interrupt configuration. The possible values are none, restart, or continue.

5.4.4 Step 3: Calibration and trace enable

When the two previous steps are complete, start the KITPSCDEBUG by Clicking on Trace in the IDE. Microcode is compiled and the following command window should appear.

```

C:\windows\system32\cmd.exe
microcode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_
diag_ch1.bin
Writing hexadecimal file:
c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr
ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_
diag_ch1.hex
Writing Trace 32 file:
c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr
ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_
diag_ch1.t32
Writing Trace metadata:
c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr
ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_
diag_ch1.trace.dat
Microcode metadata read from c:\users\b16868\desktop\home_driver\psc\2-emgs3325\
6-software\microcode\ide microcode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\
source\mc33pt2000_vfm_65v_6c_diag_ch1.trace.dat
Device under test is pt2000
Saturn tracer board detected
HW version 1.0
Reading data from port
Press 'q' to abort operation at any time
Trace Start
FW version 2.0

```

Figure 13. Trace command window

The red LED on the KITPSCDEBUG should be ON. This means the KIT is waiting for the calibration sequence. Since the device should already have been flashed, either by the MCU or by SPIGEN, it is now necessary to set the bit `trace_enable` to 1.

If the post trigger bits are all set to 1 the trace does not stop automatically, but only when the trace enable bit is set to 0 using SPIGEN.

Note: The same thing can happen if the code does not go to the `trace_stop` line. It can be the case when a diagnostic error is detected and the code goes to interrupt and not to the end of injection. So to stop the trace when it is not reaching the `trace_stop`, write 0 to the `trace_enable` bit.

In some cases, the trace data and the trace output file can be very big on the computer. If Notepad cannot open the trace output because of the file size, use a freeware application such as HJ-Split to split the file into smaller files that can be viewed more easily.

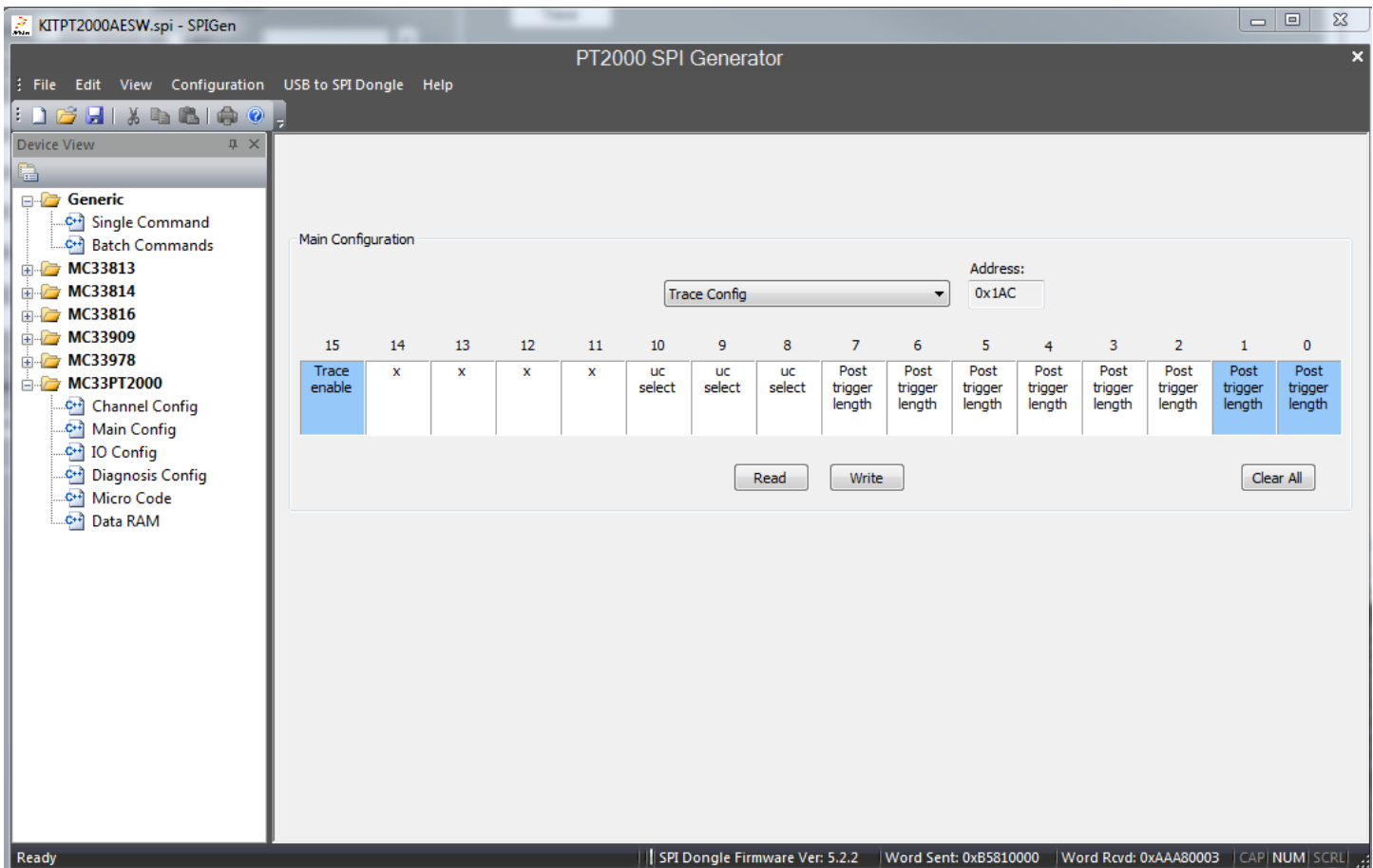


Figure 14. Trace enable bit set to 1 to start calibration sequence

When this trace enable is set to 1, the green LED on the tracer is ON (Red LED stays ON). This means the tracer now waits for the data coming from the DBG pin. This happens as soon as the code reaches the trace start point.

In this example, it happens as soon as the uPC goes to Boost phase on microcore 0 channel 1, corresponding to the time where injector 1 turns ON. A pulse on start 1 is needed to start the trace. Results after a pulse on start 1 should be as in Figure 15.

```

C:\windows\system32\cmd.exe
6-software\microcode\ide microcode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\
source\mc33pt2000_vfm_65v_6c_diag_ch1.trace.dat
Device under test is pt2000
Saturn tracer board detected
HW version 1.0
Reading data from port
Press 'q' to abort operation at any time
Trace Start
FW version 2.0
First Data received
306
Trace Stop

Acquisition Complete
Decoding content of file c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-so
ftware\microcode\ide microcode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\trac
er\stream.dat
Decoding stream recorded from Saturn 1.0 tracer board
Tracer board FW version 2.0
Clock cycles acquired in the trace operation: 4903
100%
Decoding operation complete
Are you sure you want to exit [Y]?
  
```

Figure 15. Trace command window after execution

Type a “y” command on the keyboard to close the trace command window to go back to the PT2000/MC33816 IDE.

5.4.5 Step 4: Analyzing trace results

Clicking on Trace Results on the IDE automatically opens the folder where the trace is stored. The trace output file name is the same as the microcode file with a .trace.res extension. Use notepad or notepad++ to open the .trace.res file. If several traces are done for the same project, they will be appended in the same output file (.trace.res).

5.4.5.1 Trace output file description

TIME	Code Line (hex)	Num CLK Cycles	Microcode
time 20.67us	81	1	boostt1: cwer peak0 ocur row2; * jump to peak phase when current is over threshold
time 20.83us	82	1	cwer boost_err0 tc3 row5; * define wait table if actuation longer than injmaxguard go to eoinj
time 21.00us	83	1213	wait row125;

Figure 16. Trace output file example

In the Figure 16 example, see how the trace output file is generated. On the left side, the first time where the instruction is executed, 20.67μs is shown. The next column specifies the microcode line number, 81 in this case. The third column corresponds to the number of clk cycle spent by the microcore in this instruction. For example, it took 1213 ck cycles to reach a wait 125 (in this case wait 5).

5.4.5.2 Trace output file with AN5186 example

With the example of AN5186 and the parameters selected previously, the output results should look like the one in Figure 17.

TIME	Code Line (hex)	Num CLK Cycles	Microcode
time 0.00us	048	1	boostt0: load iboost dac_sssc_ofs; * load the boost phase current threshold in the current dac
time 0.17us	049	1	lddc rst_ofs keep keep injmaxtboost c3; * start boost counter in case iboost never reached
time 0.33us	04A	1	lddc rst_ofs keep keep injmintboost c2; * start boost counter in case iboost is reached too fast
time 0.50us	04B	1	stf low bstflag; * set flag0 high to stop the dc-dc converter
time 0.67us	04C	1	stos on on on; * turn vbat off, boost on, ls on
time 0.83us	04D	1	endiaga on on on; * enable auto diag
time 1.00us	04E	1	cwer boostt1 tc2 row5; * after injminboost go to boost t2
time 1.17us	04F	1	cwer boost_err0 ocur row2; * jump to error in case current is reached before the end of injminboost
time 1.33us	050	116	wait row125;
time 20.67us	051	1	boostt1: cwer peak0 ocur row2; * jump to peak phase when current is over threshold
time 20.83us	052	1	cwer boost_err0 tc3 row5; * define wait table if actuation longer than injmaxguard go to eoinj
time 21.00us	053	1179	wait row125;
time 217.50us	056	1	peak0: store cnt3 openinj_record ofs; * store opening time in the data in dram with offset to separate inj1 and 2
time 217.67us	057	1	stf high bstflag; * set flag0 high to release the dc-dc converter idle mode
time 217.83us	058	1	lddc rst_ofs keep keep tpeak_tot c1; * load the length of the total peak phase in counter 1
time 218.00us	059	1	load ipeak dac_sssc_ofs; * load the peak current threshold in the current dac
time 218.17us	05A	1	cwer bypass0 tc1 row2; * jump to bypass phase when tc1 reaches end of count
time 218.33us	05B	1	cwer peak_on0 tc2 row3; * jump to peak_on when tc2 reaches end of count
time 218.50us	05C	1	cwer peak_off0 ocur row4; * jump to peak_off when current is over threshold
time 218.67us	05D	1	peak_off0: lddc rst_ofs keep keep tpeak_off c2; * load in the counter 2 the length of the peak_off phase
time 218.83us	05E	1	stos off on off; * turn vbat off, boost off, ls on
time 219.00us	05F	120	wait row123;
time 239.00us	060	1	peak_on0: stos on on off; * turn vbat on, boost off, ls on
time 239.17us	061	1	wait row124; * wait for one of the previously defined conditions

Figure 17. Trace output part 1 boost and peak phase

time 905.67us	06B	1	hold_off0: lddc rst_ofs keep keep thold_off c2; * load the length of the hold_off phase in counter 2
time 905.83us	06C	1	stos off on off; * turn vbat off, boost off, ls on
time 906.00us	06D	240	wait row123;
time 946.00us	06E	1	hold_on0: stos on on off; * turn vbat on, boost off, ls on
time 946.17us	06F	68	wait row124; * wait for one of the previously defined conditions
time 957.50us	071	1	eoinj0: stos off off off; * turn vbat off, boost off, ls off
time 957.67us	072	1	endiaga diagoff; * disable auto diag
time 957.83us	073	1	bias all off; * enable all biasing structures, kept on even during actuation
time 958.00us	074	1	stf high bstflag; * set flag0 to high to release the dc-dc converter idle mode

Figure 18. Trace output part 2 hold and end of injection phase

As shown in Figure 18, when the stop trace reaches line 71h, the tracer still saves three more lines corresponding to the value of the post length trigger register (refer to Figure 9).

5.5 Tracer waveform with the AN5186 example

Figure 19 is a scope shot example with an injector connected to INJ1. The expected behavior is when the Start1 pin rise injection goes to Boost phase until the boost current is reached, then jumps to peak phase, then bypass phase, and finally to hold phase, until the start pin goes low or after a 10 ms timeout. With the above settings for the trace, expect the trace to follow the start 1 pulse.

NOTE: When executing the trace, it is recommended NOT to connect a scope probe on the DBG, since it can disturb the 24/48 MHz trace signal.

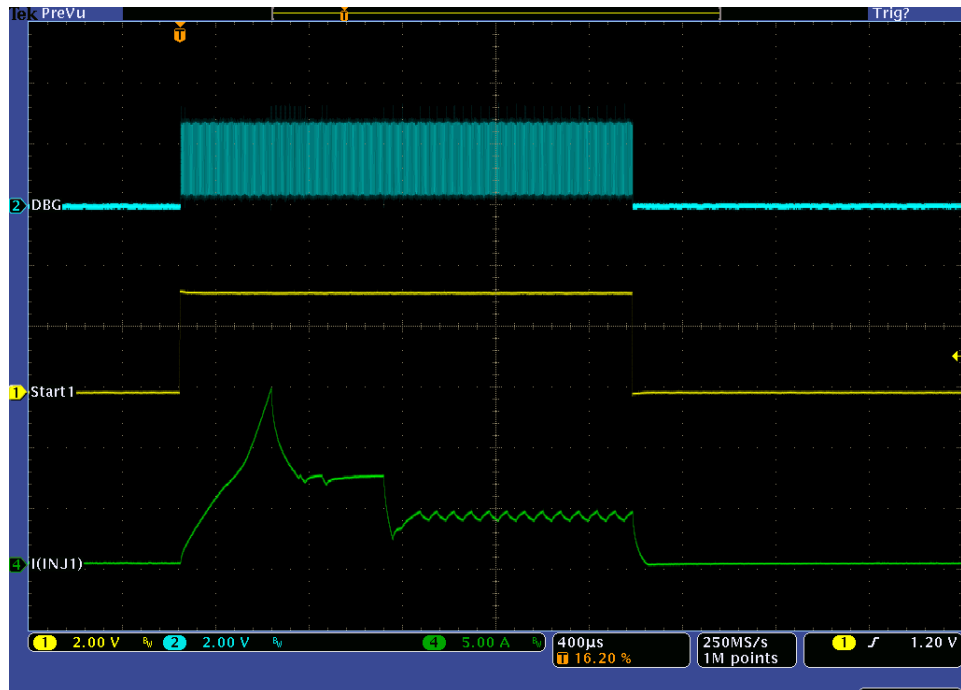
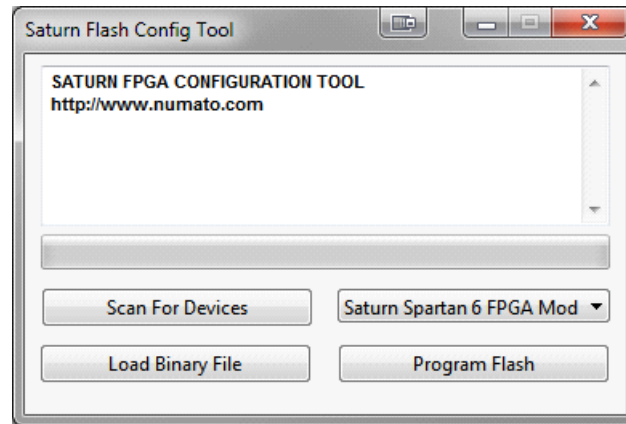


Figure 19. Trace scope shot

6 Updating the tracer firmware

The tool chain is continuously improving, so check to see if the tracer firmware may need to be updated. The Saturn Flash Config software is used to perform the update. It is included in the PT2000/MC33816 IDE installation folder (The default path is C:\Program Files\NXP\PT2000DevStudio\SaturnFlashConfig).



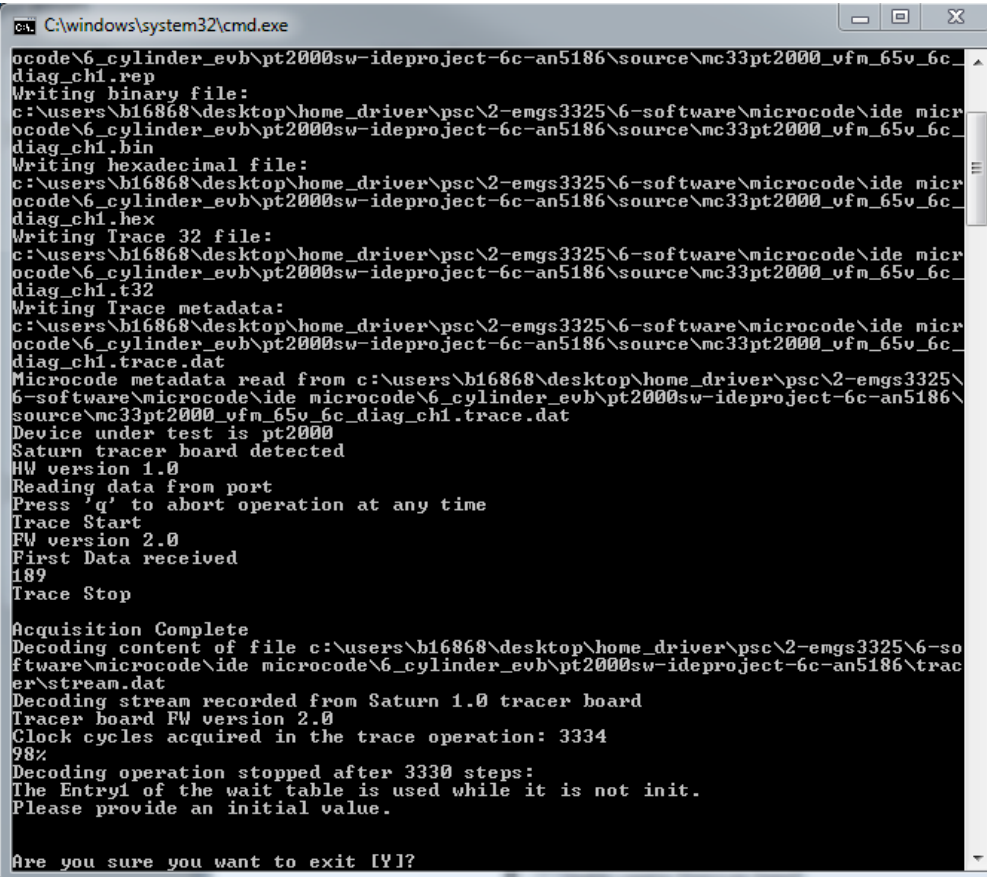
If an update of the Tracer firmware is required, the following steps are necessary to re-flash the KITPSCDEBUEVM:

- Connect the Tracer to a computer USB port (same as in a normal setup)
- Double click on SaturnFlashConfig.exe (located in the PT2000 or MC33816 IDE installation folder)
- Click on Load Binary File
- Select the file AsicTracer.bin included in the same folder as the Saturn Flash Config Tool
- Click on Program Flash
- The latest version of the tracer firmware is loaded

Note: If the computer being used has USB 3.0, the Saturn Flash Config Tool only works on Windows 8 and later versions of the operating system.

7 Troubleshooting

Table 3. Troubleshooting

Problem	Possible Solution
<p>The Entry1 of the wait table is used while it is not in init</p>	 <pre> C:\windows\system32\cmd.exe ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_ diag_ch1.rep Writing binary file: c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_ diag_ch1.bin Writing hexadecimal file: c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_ diag_ch1.hex Writing Trace 32 file: c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_ diag_ch1.t32 Writing Trace metadata: c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-software\microcode\ide micr ocode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\source\mc33pt2000_vfm_65v_6c_ diag_ch1.trace.dat Microcode metadata read from c:\users\b16868\desktop\home_driver\psc\2-emgs3325\ 6-software\microcode\ide microcode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\ source\mc33pt2000_vfm_65v_6c_diag_ch1.trace.dat Device under test is pt2000 Saturn tracer board detected HW version 1.0 Reading data from port Press 'q' to abort operation at any time Trace Start FW version 2.0 First Data received 189 Trace Stop Acquisition Complete Decoding content of file c:\users\b16868\desktop\home_driver\psc\2-emgs3325\6-so ftware\microcode\ide microcode\6_cylinder_evb\pt2000sw-ideproject-6c-an5186\trac er\stream.dat Decoding stream recorded from Saturn 1.0 tracer board Tracer board FW version 2.0 Clock cycles acquired in the trace operation: 3334 98% Decoding operation stopped after 3330 steps: The Entry1 of the wait table is used while it is not in init. Please provide an initial value. Are you sure you want to exit [Y]? </pre> <p>In this case, the option configuration of the tracer is not done properly, it needs to specify the line where wait 1 goes. Refer to section 5.4.3, Step 2: KITPSCDEBBUG configuration using IDE, Page 12.</p>
Illegal code	Make sure no scope probes are connected to the DBG pin
Unlimited trace	trace_stop is never reached so the KITPSCDEBBUG acquires data until the RAM is full. In this case write the trace_enable bit to 0 by SPI.

8 References

The following are URLs for information on related NXP products and application solutions:

NXP.com Support Pages	Description	URL
KITPSCDEBUGEVM	Tool Summary Page	http://www.nxp.com/KITPSCDEBUGEVM
PT2000	Product Summary Page	http://www.nxp.com/PT2000
PT2000-IDEUG	User Guide	http://www.nxp.com/files/analog/doc/user_guide/PT2000-IDEUG.pdf
Analog Home Page		http://www.nxp.com/analog
Automotive Home Page		http://www.nxp.com/automotive

8.1 Support

Visit www.nxp.com/support for a list of phone numbers within your region.

8.2 Warranty

Visit www.nxp.com/warranty to submit a request for tool warranty.

9 Revision history

Revision	Date	Description of changes
1.0	10/2015	• Initial release
2.0	11/2015	• Added section Updating the tracer firmware on page 18
	8/2016	• Updated to NXP document form and style

How to Reach Us:**Home Page:**[NXP.com](http://www.nxp.com)**Web Support:**<http://www.nxp.com/support>

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no expressed or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation, consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by the customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:

<http://www.nxp.com/terms-of-use.html>.

NXP, the NXP logo, Freescale, the Freescale logo, and SMARTMOS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. All rights reserved.

© 2016 NXP B.V.

Document Number: KTPSCDEBBUGUG

Rev. 2.0

8/2016



Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

NXP:

[KITPSCDEBUGEVM](#)