AN10863

LPC17xx example description

Rev.02 — 04 Dec 2009 **Application note**

Document information

Info	Content
Keywords	LPC17xx, MCB1700
Abstract	Describe the example code to test each peripheral function of LPC17xx on Keil MCB1700 version 1.0

founded by Philips

Revision history

Rev	Date	Description
02	20091204	Secondary version
01	20090828	Initial version.

Contact information

For additional information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

This document will describe the example code to test each peripheral function of LPC17xx on Keil MCB1700 version 1.0 board.

2. Additional condition

2.1 Serial display

All the example codes require UART for display, and must be configured with the following condition:

Common communication is UART0 port on MCB1700 ver.1 with parameter below:

- 115200 bps
- 8 databit
- None parity
- 1 Stopbit
- No flow control
- Software used on window to communicate via UART is HyperTerminal configured with parameters configured above:

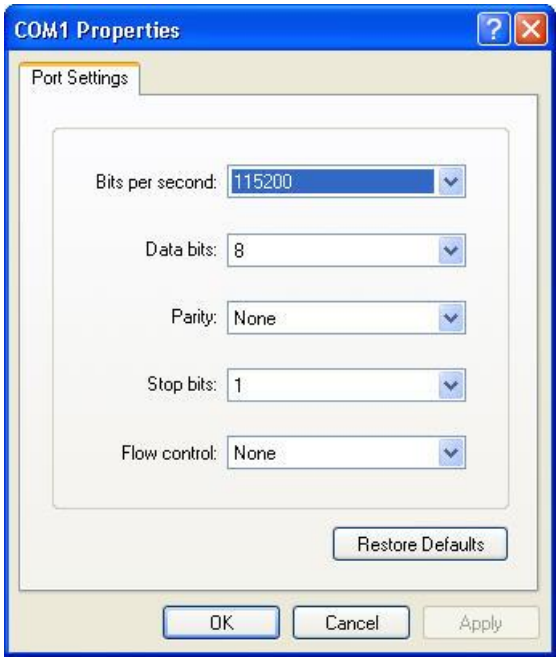

Fig 1. Com port setting on Window

Note: DO NOT USE HYPERTERMINAL if the content to display is larger than the space of hyper terminal window (white space), otherwise some characters will be lost (this is not an error of the UART driver). Error is as shown in Fig 2 when running the DMA example code.

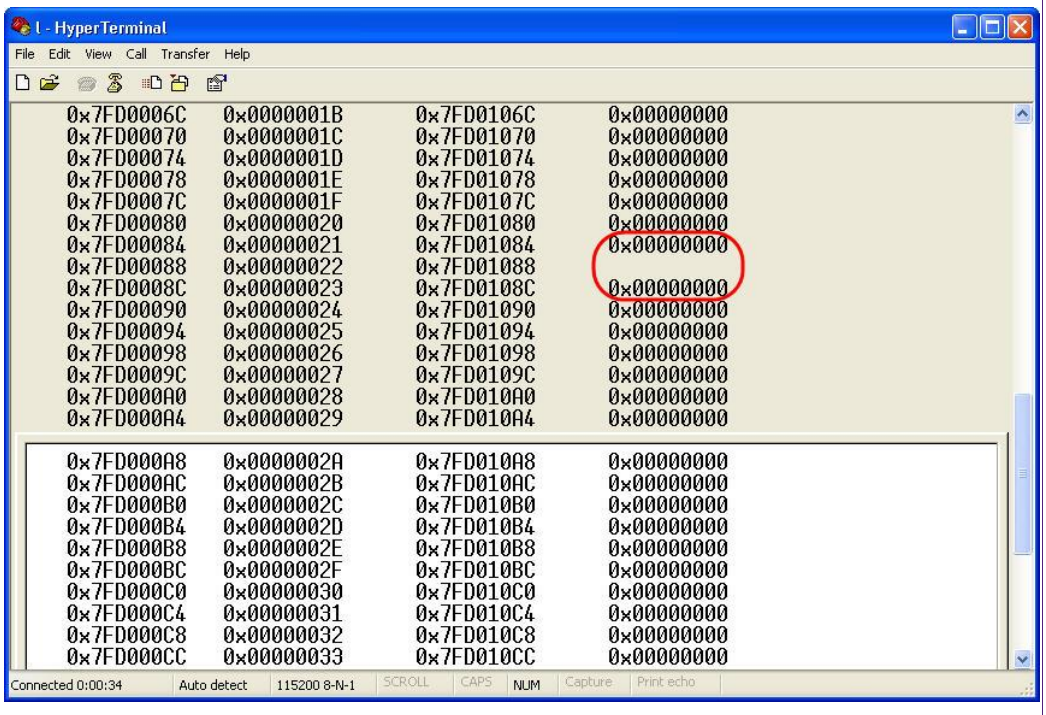

Fig 2. Some characters are lost

The solution to this is to use another communication tool, such as Flash magic, or TeraTerm.

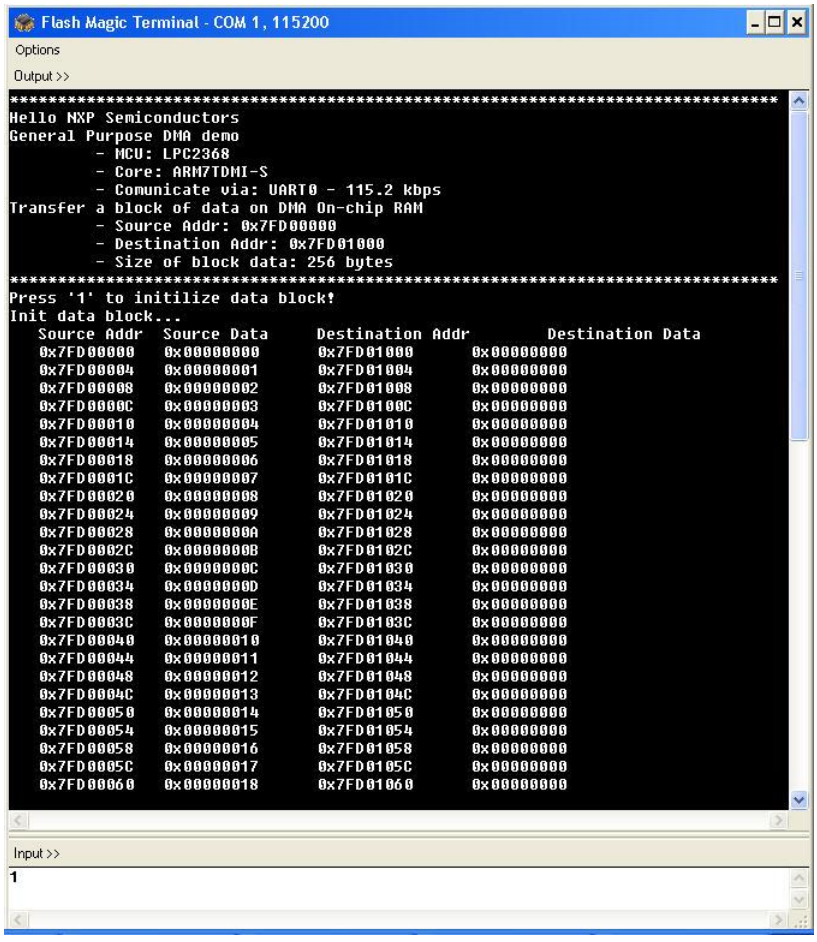

Fig 3. Terminal on Flash magic is smooth but response time of display is slow.

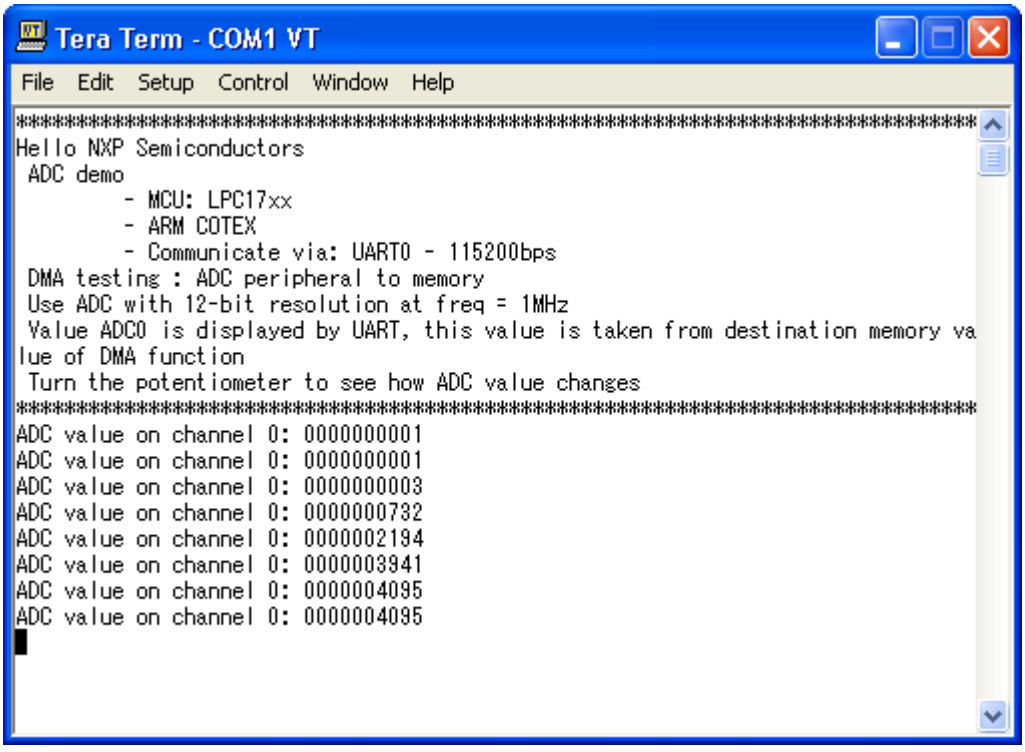

Fig 4. Teraterm IDE

All jumpers on MCB1700 board version 1.0 must be default as follows unless recommended in each example hardware requirement (please see file abtract.txt in each example for more detail):

- VBUS: power supply via USB port – Must always be ON.
- VDDIO jumper: ON.
- VDDREG jumper: ON.
- RST jumper: when using ISP with FlashMagic, OFF is normal operation
- ISP jumper: when using ISP with FlashMagic, OFF is normal operation

Note: UART0 function could not operate normally when either RST jumper or ISP jumper are ON.

2.2 Additional files requirement

In order to use serial display via UART0 port, two files below should be included together in each example (except for UART example):

- debug_frmwrk.h
- debug_frmwrk.c

A driver library configuration file must be also included in each example to enable/disable specified peripheral driver:

- lpc17xx_libcfg.h

2.3 Running mode

Each example can be used to run in both two supported mode: RAM mode and ROM (FLASH) mode unless recommended in Software Configuration → Running mode.

2.3.1 RAM mode

All files in each example must be built to .elf file, this file will be loaded in to RAM through a debugger tool before running.

2.3.2 ROM (FLASH) mode:

All files in each example must be built to .hex file; this file will be burned in to ROM (FLASH) memory through an external tool (i.e. Flash Magic…) before running.

Please refer to the "LPC17xx_SoftwareDevelopmentToolchain" document for more details.

3. UART polling example

3.1 Purpose

This is a simple UART example in polling mode

UART0 – 9600bps – 8 data bit – No parity – 1 stop bit – No flow control.

3.2 Hardware configuration

Please see abstract.txt file for more details.

3.3 Software configuration

Required files

uart_polling_test.c

Running mode

Please see abstract.txt file for more details.

3.4 Procedure

Please see abstract.txt file for more details.

After restart, the welcome screen will be like this:

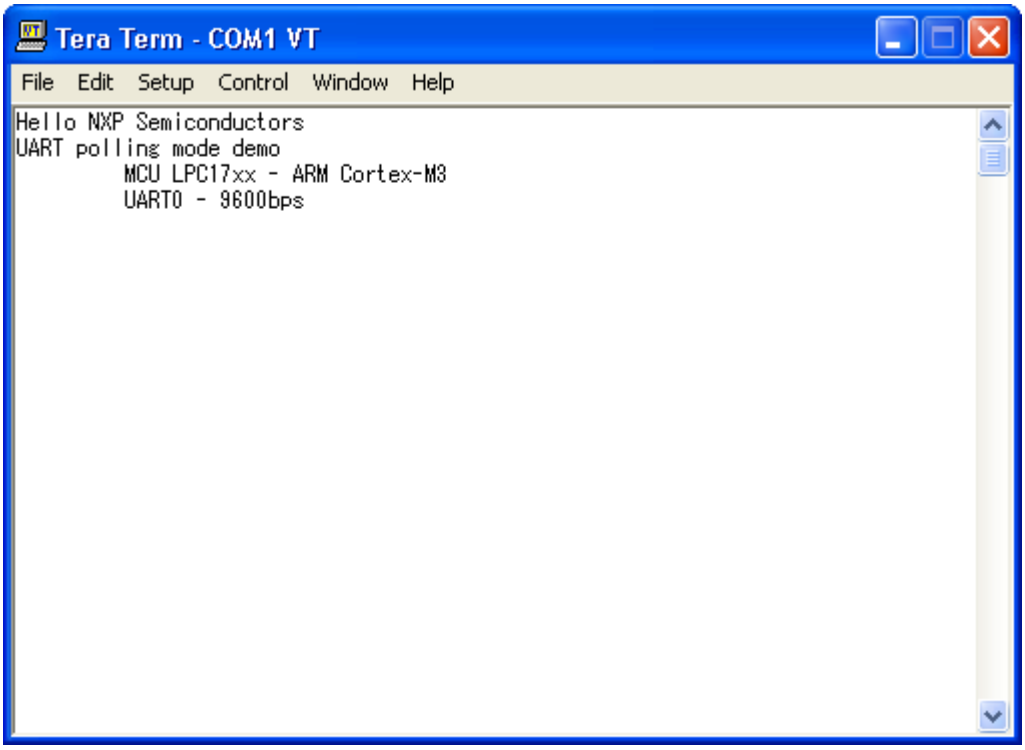

Fig 5. UART Polling demo

4. UART auto baud rate example

4.1 Purpose

This is a simple UART example using auto baud rate mode

4.2 Hardware configuration

Please see abstract.txt file for more details.

4.3 Software configuration

Required files

uart_autobaudrate_test.c

Running mode

Please see abstract.txt file for more details.

4.4 Procedure

- Open serial terminal application.
- Choose desired baud rate.
- Type ‘A’ or ‘a’ to start autobaudrate mode.

Please see abstract.txt file for more details.

The welcome screen will be like this:

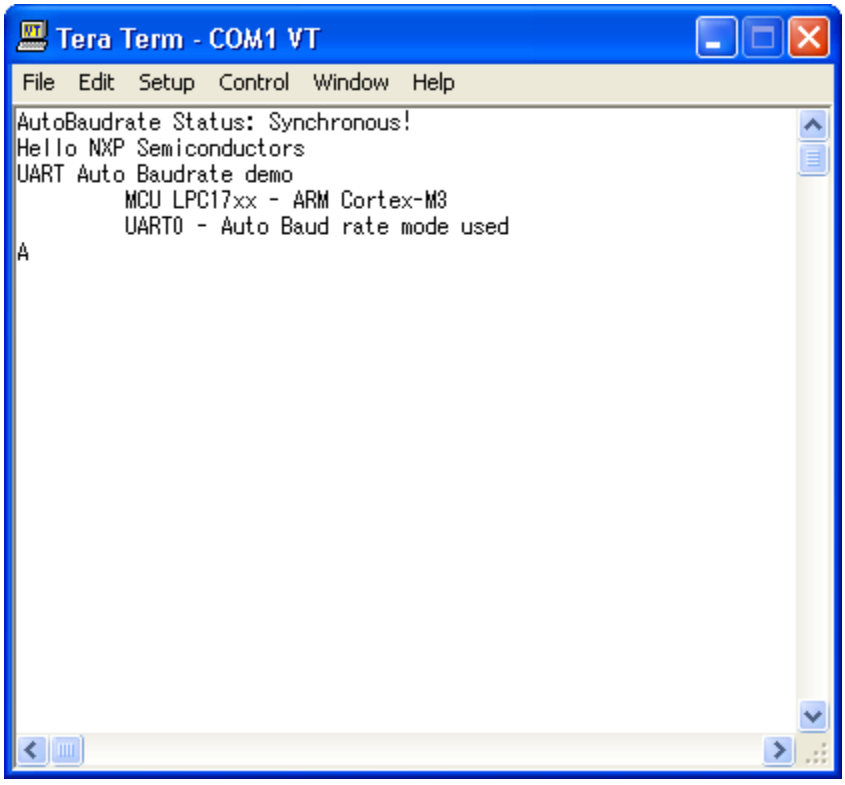

Fig 6. UART auto baudrate demo

5. UART DMA Example

5.1 Purpose

This is a simple UART example in DMA mode.

UART0 – 9600bps – 8 data bit – No parity – 1 stop bit – No flow control.

5.2 Hardware configuration

Please see abstract.txt file for more details.

5.3 Software configuration

Required files

uart_dma_test.c

Running mode

Please see abstract.txt file for more details.

5.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

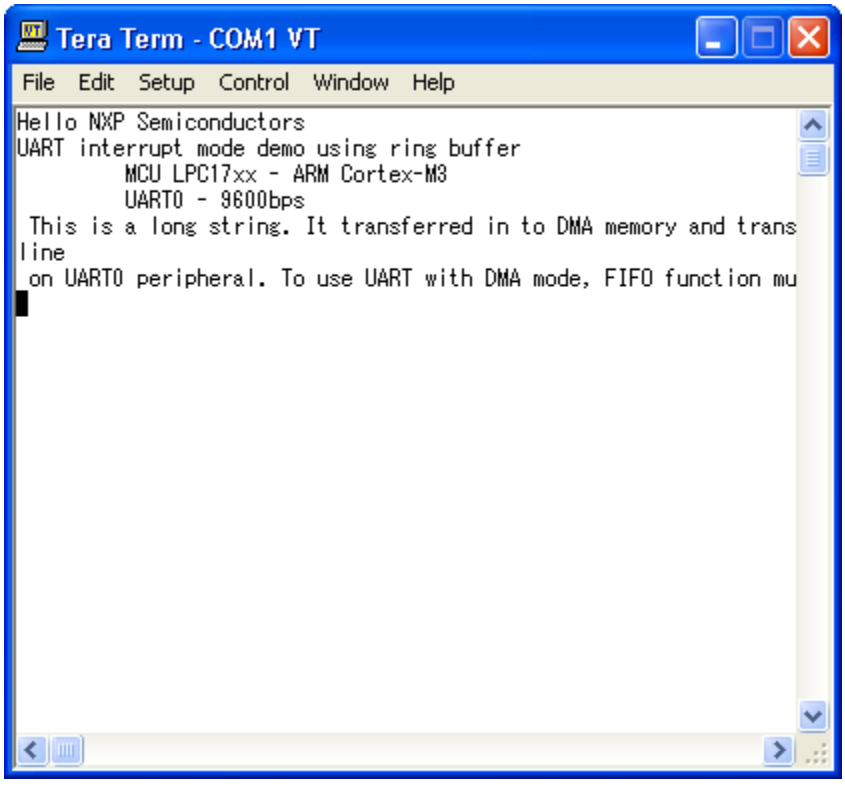

Fig 7. UART DMA demo

6. UART Interrupt example

6.1 Purpose

This is a simple UART example in interrupt mode

UART0 – 9600bps – 8 data bit – No parity – 1 stop bit – No flow control.

6.2 Hardware configuration

Please see abstract.txt file for more details.

6.3 Software configuration

Required files

uart_interrupt_test.c

Running mode

Please see abstract.txt file for more details.

6.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

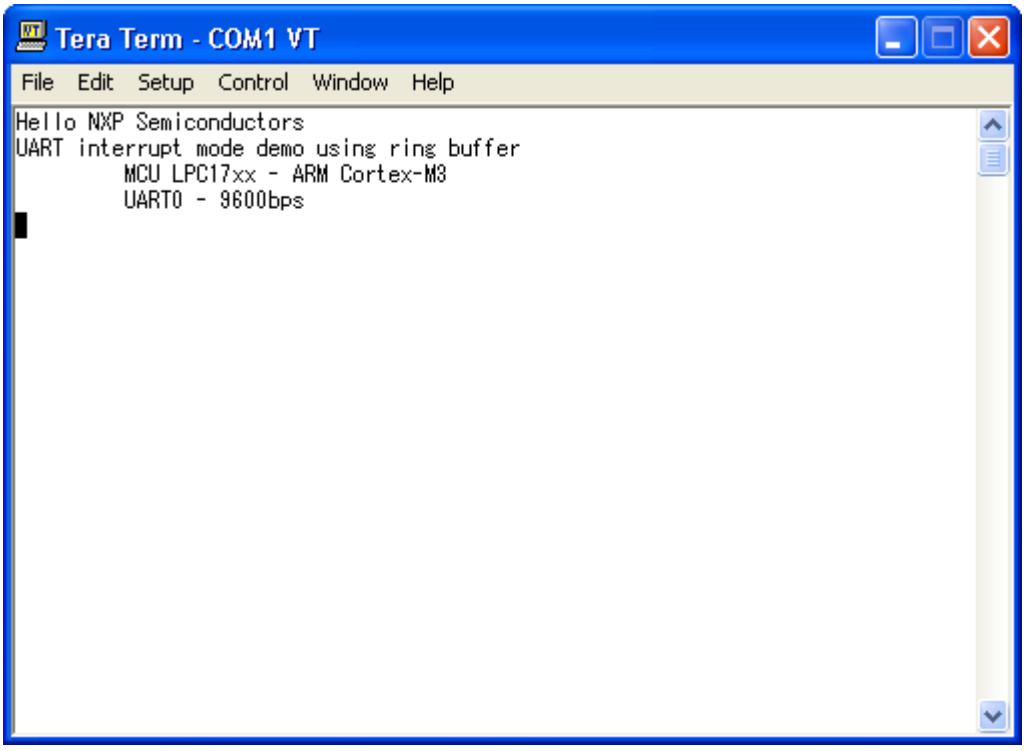

Fig 8. UART Interrupt demo

7. UART Full modem example

7.1 Purpose

This is a simple UART example using UART1 with Full modem mode

7.2 Hardware configuration

Please see abstract.txt file for more details.

7.3 Software configuration

Required files

uart_fullmodem_test.c

Running mode

Please see abstract.txt file for more details.

7.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

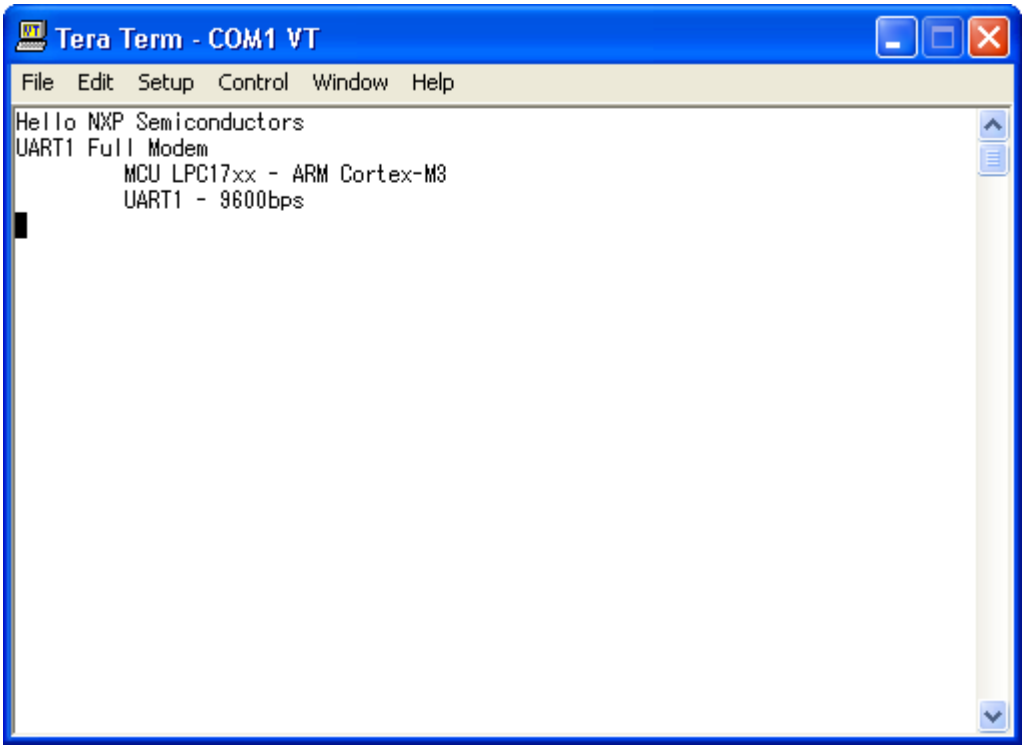

Fig 9. UART Full Modem demo

8. UART RS485 master example

8.1 Purpose

This example is used to test RS485 functionality on UART1

In this case, RS485 function on UART1 acts as Master mode on RS485 bus.

8.2 Hardware configuration

Please see abstract.txt file for more details.

8.3 Software configuration

Required files

rs485_master.c

Running mode

Please see abstract.txt file for more details.

8.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

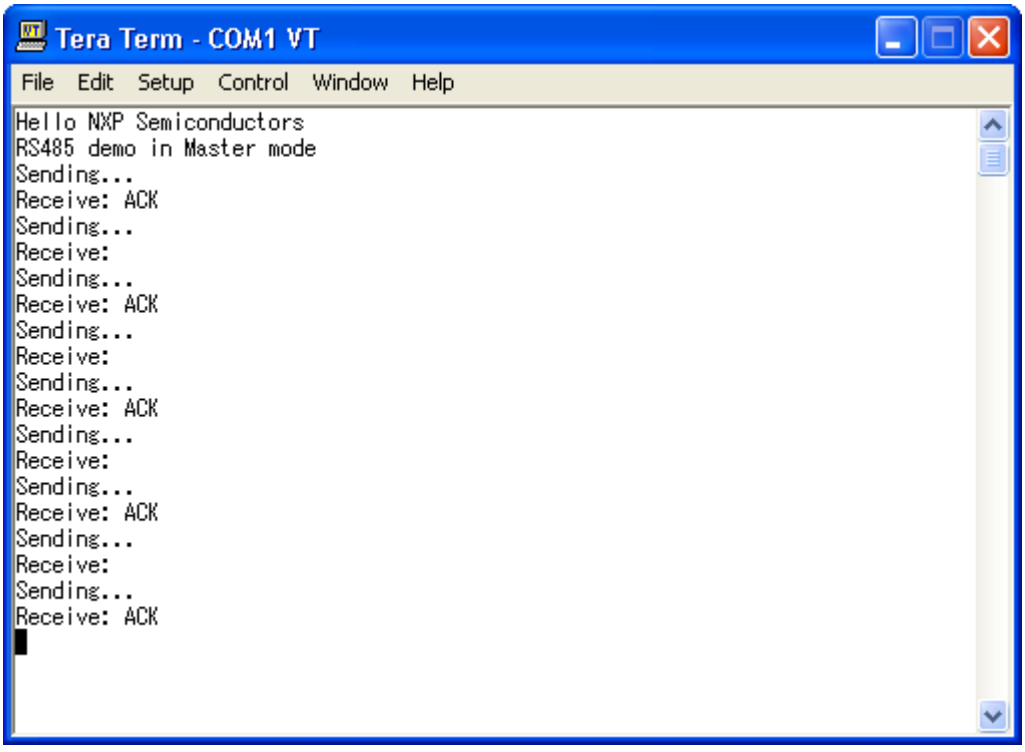

Fig 10. UART master

9. UART RS485 slave example

9.1 Purpose

This example is used to test RS485 functionality on UART1

In this case, RS485 function on UART1 acts as slave mode on RS485 bus.

9.2 Hardware configuration

Please see abstract.txt file for more details.

9.3 Software configuration

Required files

rs485_slave.c

Running mode

Please see abstract.txt file for more details.

9.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

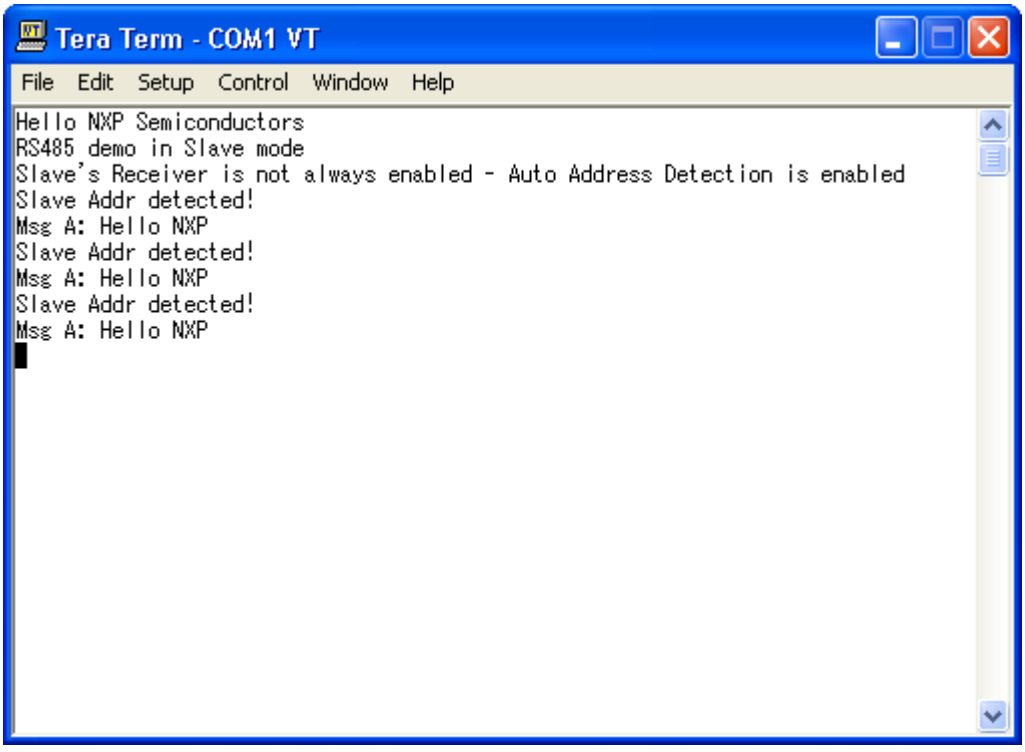

Fig 11. UART master

10. ADC polling mode example

10.1 Purpose

This is a simple example for A/D conversion in polling mode

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this conversion.

10.2 Hardware configuration

Please see abstract.txt file for more details.

10.3 Software configuration

Required files

adc_poll.c

Running mode

Please see abstract.txt file for more details.

10.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

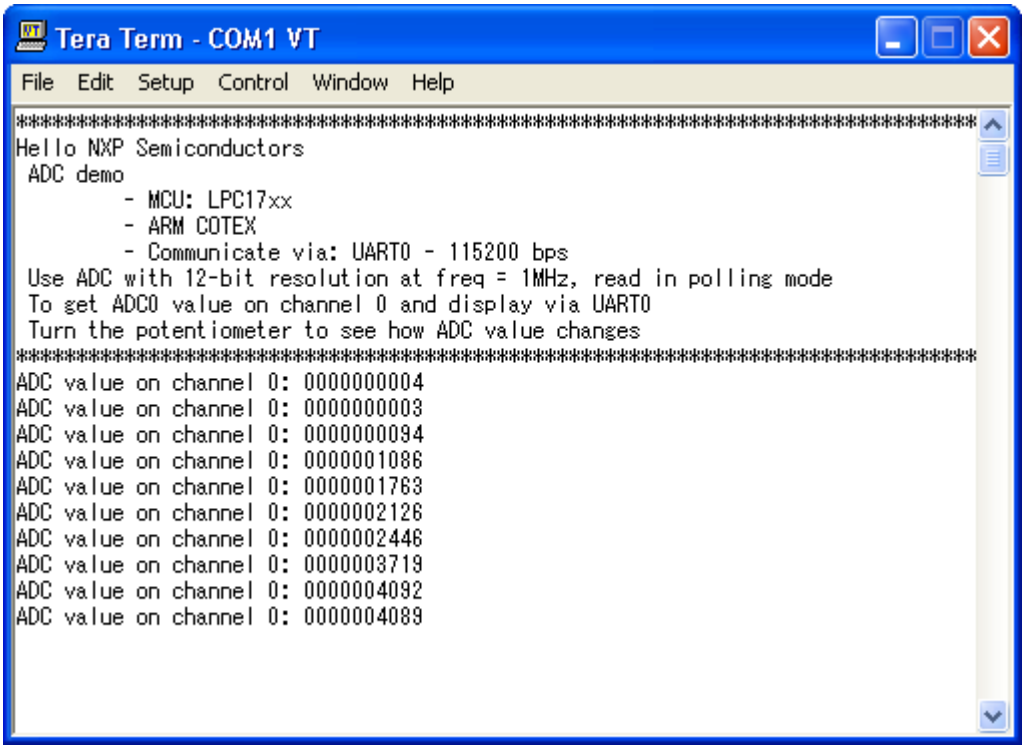

Fig 12. ADC polling mode demo

11. ADC Interrupt example

11.1 Purpose

This is a simple example for A/D conversion with interrupt mode

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this conversion.

11.2 Hardware configuration

Please see abstract.txt file for more details.

11.3 Software configuration

Required files

adc_interrupt_test.c

Running mode

Please see abstract.txt file for more details.

11.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

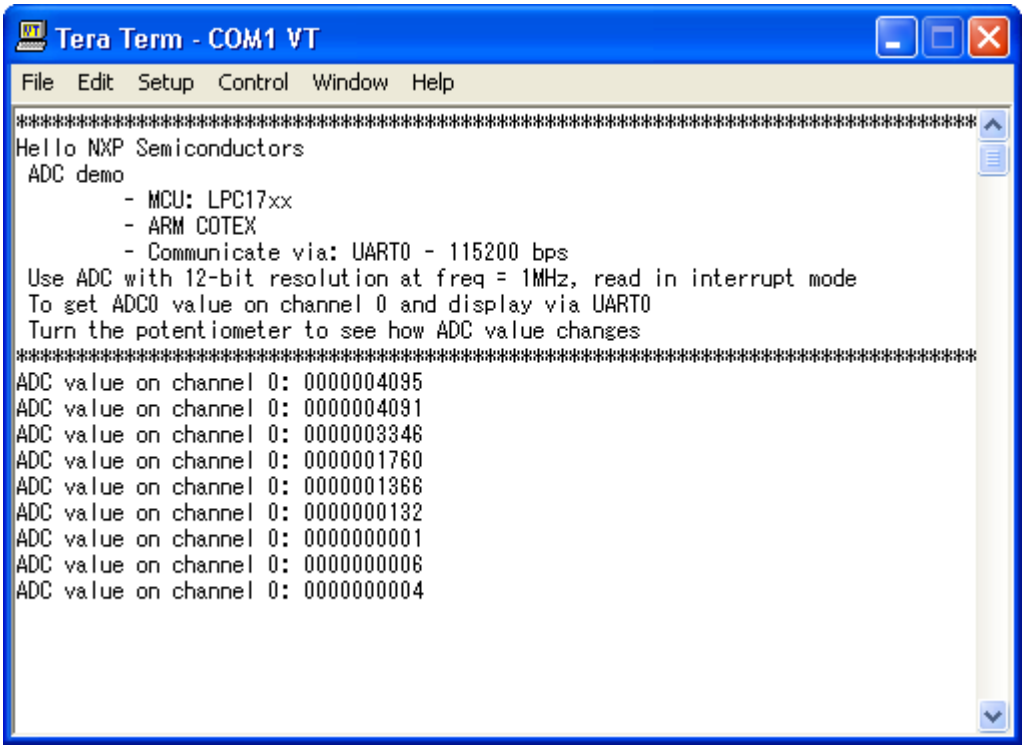

Fig 13. ADC interrupt demo

12. ADC DMA example

12.1 Purpose

This is DMA example apply for transfer ADC peripheral to memory

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this transfer.

12.2 Hardware configuration

Please see abstract.txt file for more details.

12.3 Software configuration

Required files

adc_dma_test.c

Running mode

Please see abstract.txt file for more details.

12.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

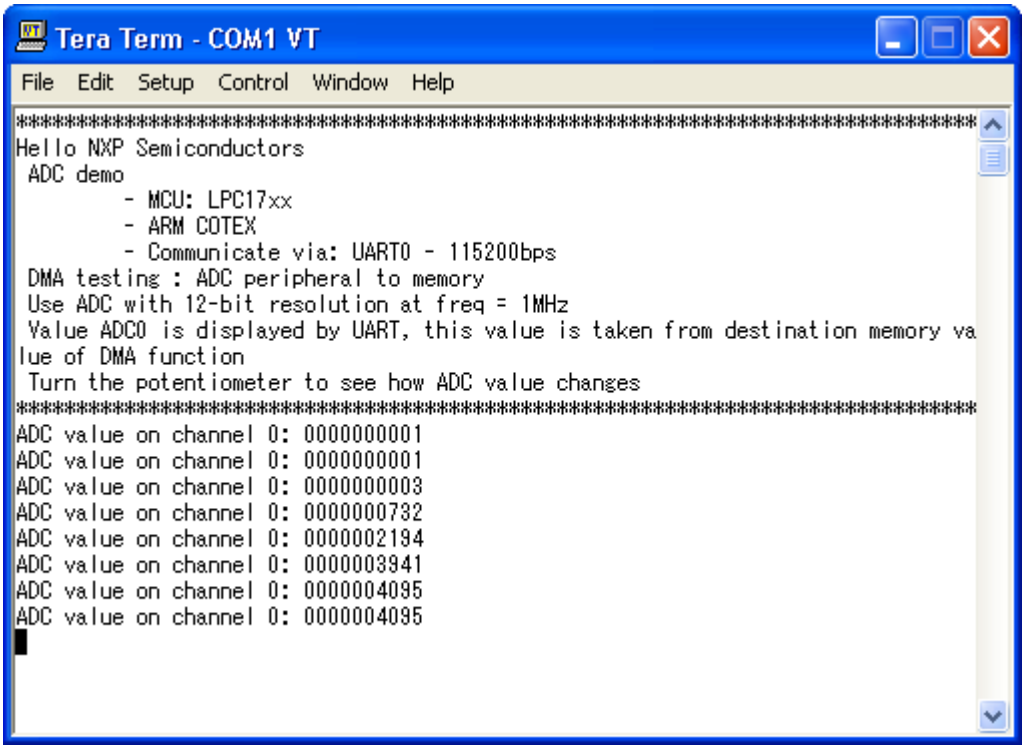

Fig 14. ADC DMA demo

13. CAN test Bypass mode example

13.1 Purpose

Use 2 CAN peripheral: CAN1 and CAN2 in the same board to test Bypass mode

We send infinite message to CAN2, the message ID and its data will be increased continuously after each transfer.

13.2 Hardware configuration

Pin 2 of CAN1 connects to Pin 2 of CAN2

Pin 7 of CAN1 connects to Pin 7 of CAN2

13.3 Software configuration

Required files

can_test_bypass_mode.c

Running mode

Default

13.4 Procedure

After reset, the welcome screen appears like this:

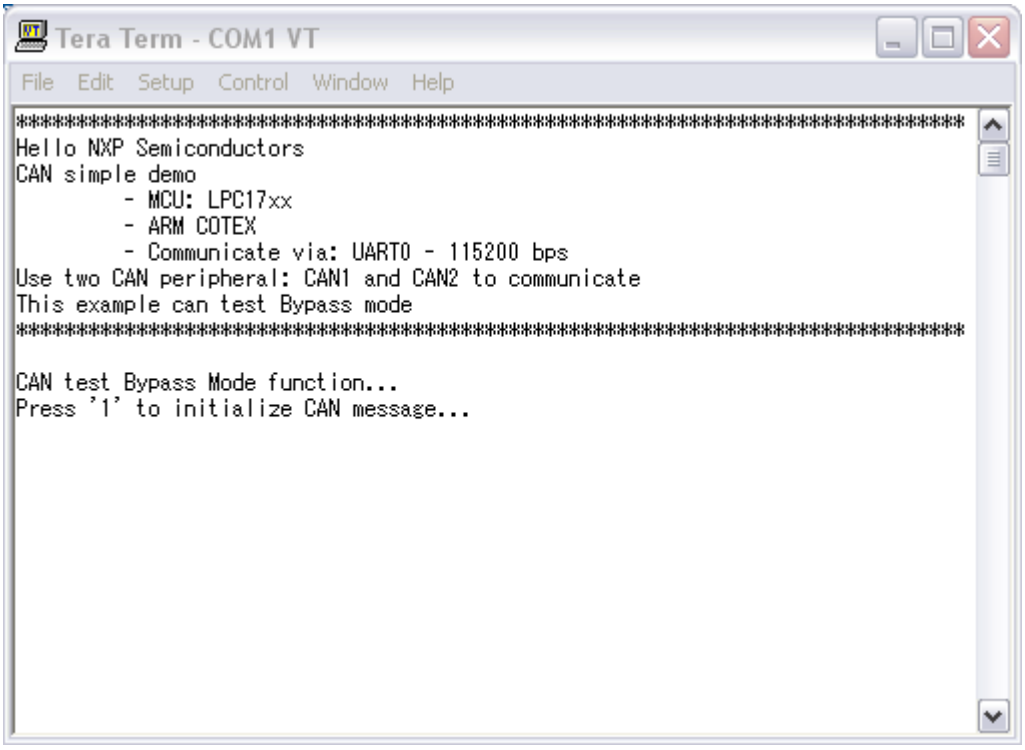

Fig 15. CAN test Bypass mode welcome screen

Press '1' to initialize transmit message…

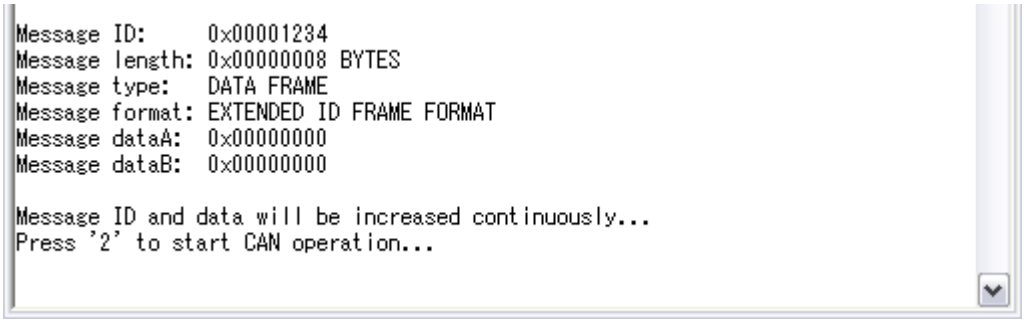

Fig 16. CAN transmit message is initialized

Press '2' to start CAN operation. Received messages will be displayed like this:

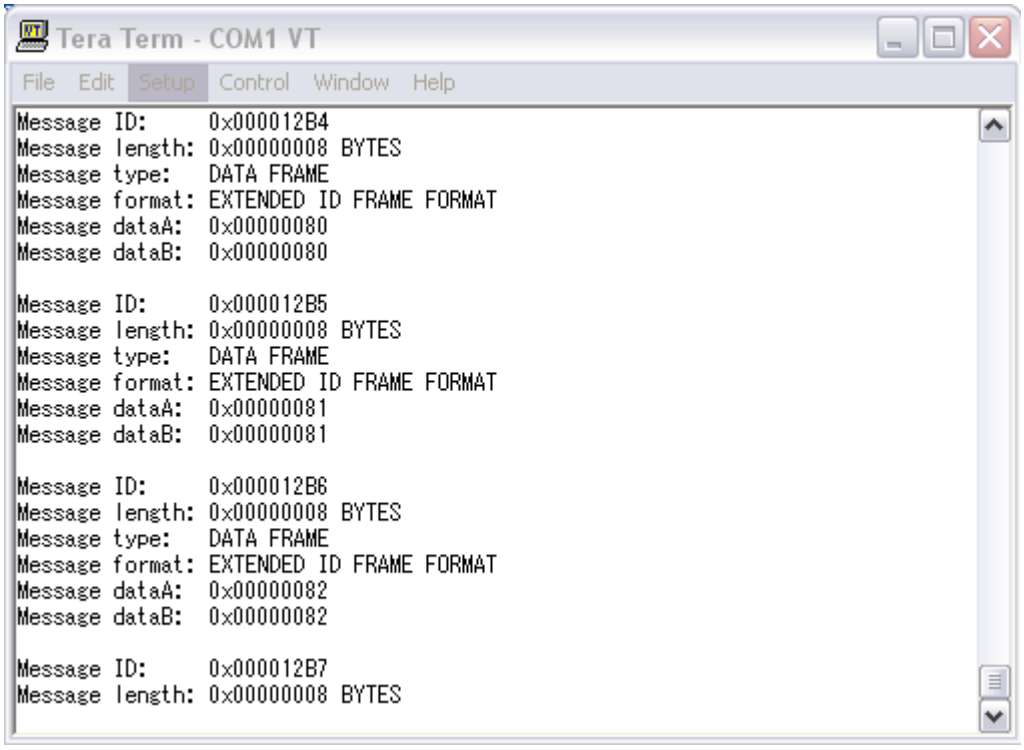

Fig 17. Display received messages

14. CAN test Acceptance Filter mode example

14.1 Purpose

Use 2 CAN channels CAN1 and CAN2 in the same board to test full Acceptance Filter mode. It supports FullCAN mode and uses both Explicit and Group ID Frame Format.

14.2 Hardware configuration

Port 2 of CAN1 connects to Port 2 of CAN2

Port 7 of CAN1 connects to Port 7 of CAN2

14.3 Software configuration

Required files

can_test_AFLUT.c

Running mode

Default

14.4 Procedure

After reset, the welcome screen appears like this:

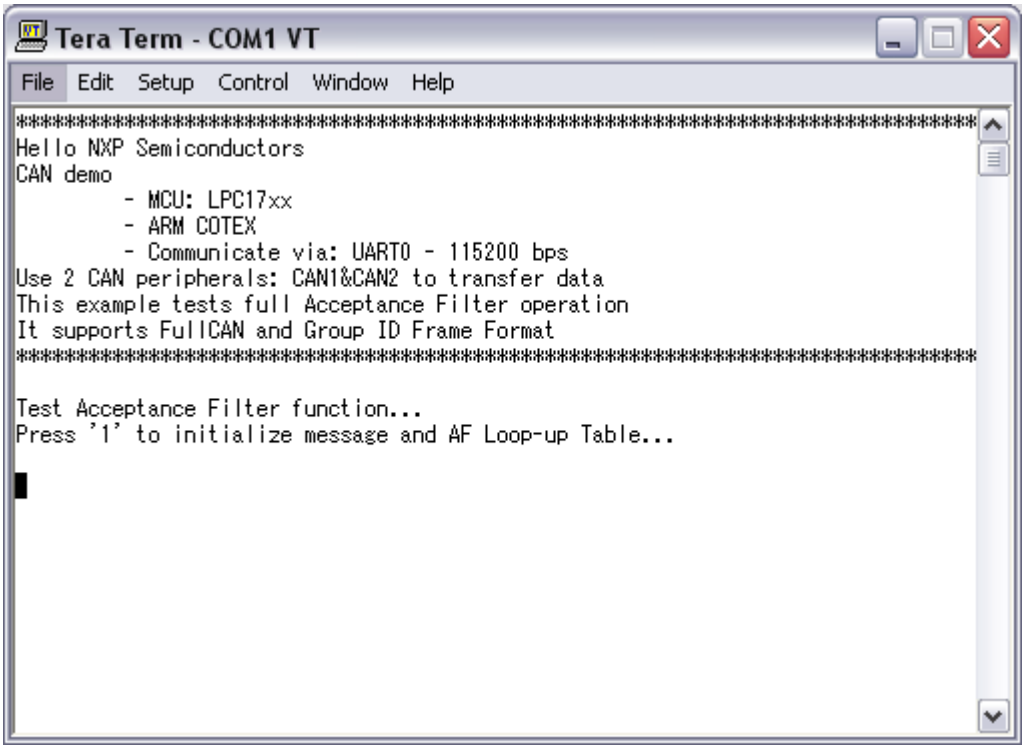

Fig 18. CAN test full AF mode welcome screen

Press '1' to initialize messages and AF Look-up Table.

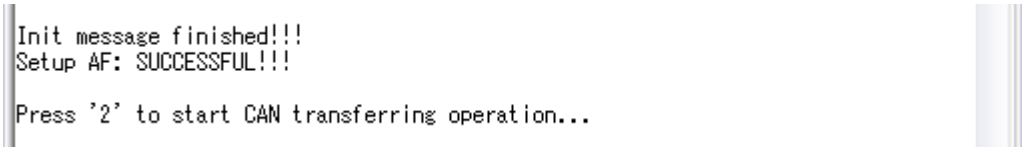

Fig 19. CAN messages and AFLUT initialized after press '1'

Press '2', the sending messages will start.

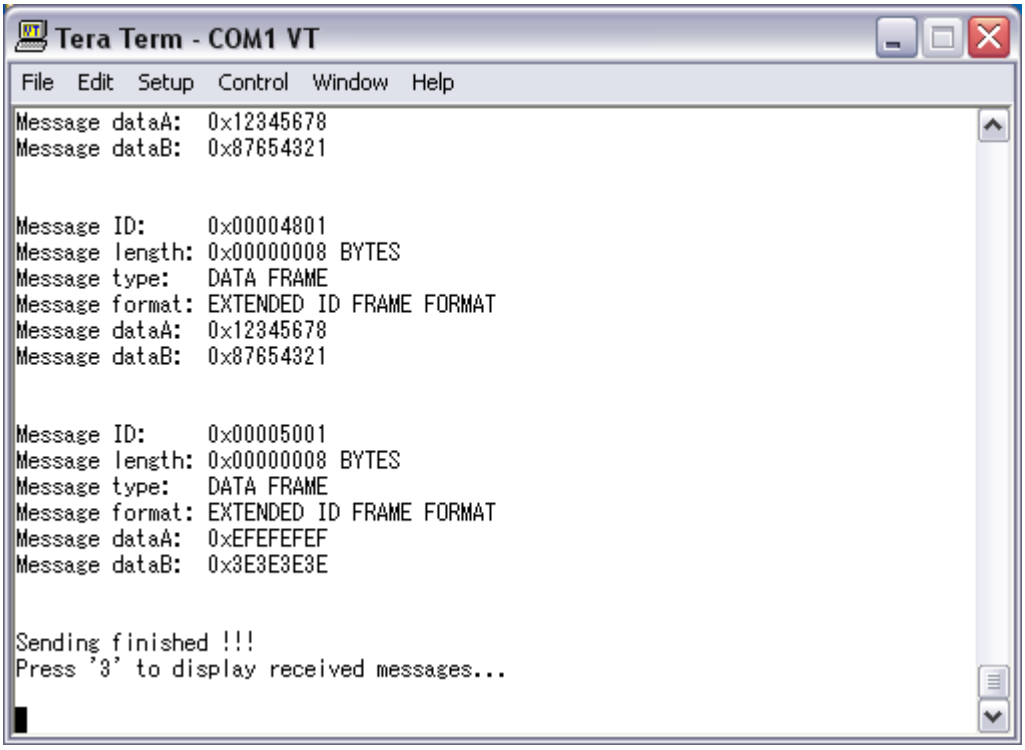

Fig 20. CAN messages were sent after press '2'

Press '3', received messages will be displayed like this:

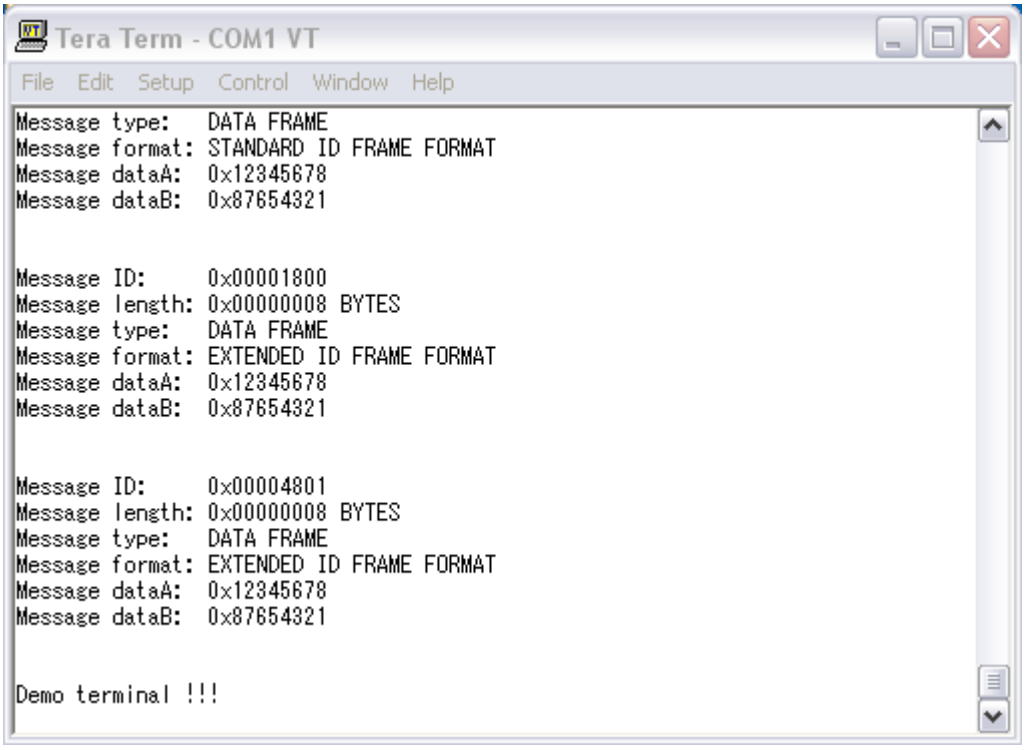

Fig 21. CAN messages were received after transferring finished

15. CAN test setup AFLUT dynamically

15.1 Purpose

This example used to check functions that add/remove AFLUT entry dynamically

15.2 Hardware configuration

Port 2 of CAN1 connects to Port 2 of CAN2

Port 7 of CAN1 connects to Port 7 of CAN2

15.3 Sofware configuration

Required files

can_AFLUT_dynamic.c

Running mode

Default

15.4 Procedure

After reset, the welcome screen appears like this:

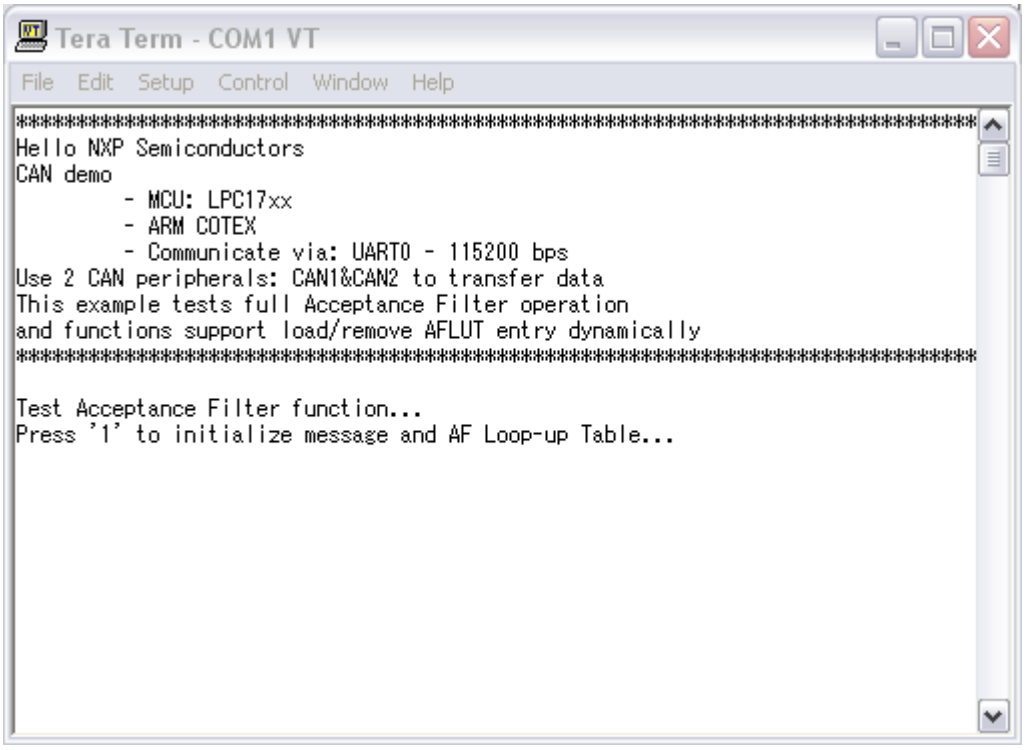

Fig 22. Welcome screen

Press “1”…

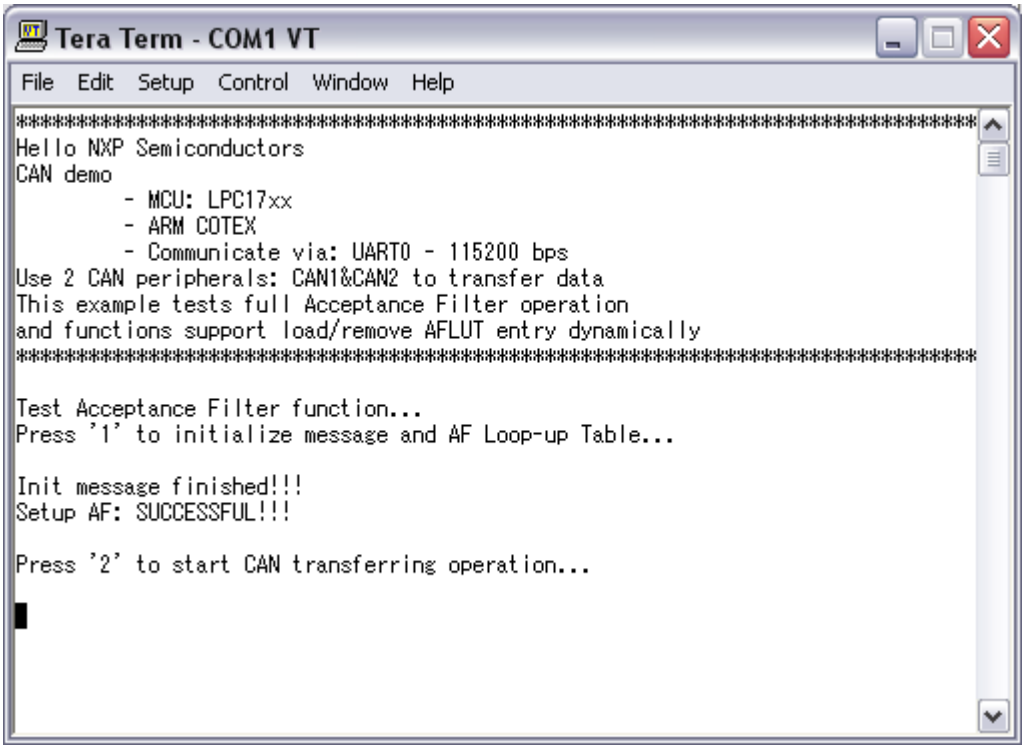

Fig 23. AFLUT was setup and messages were initialized

Press ‘2’… CAN will send messages

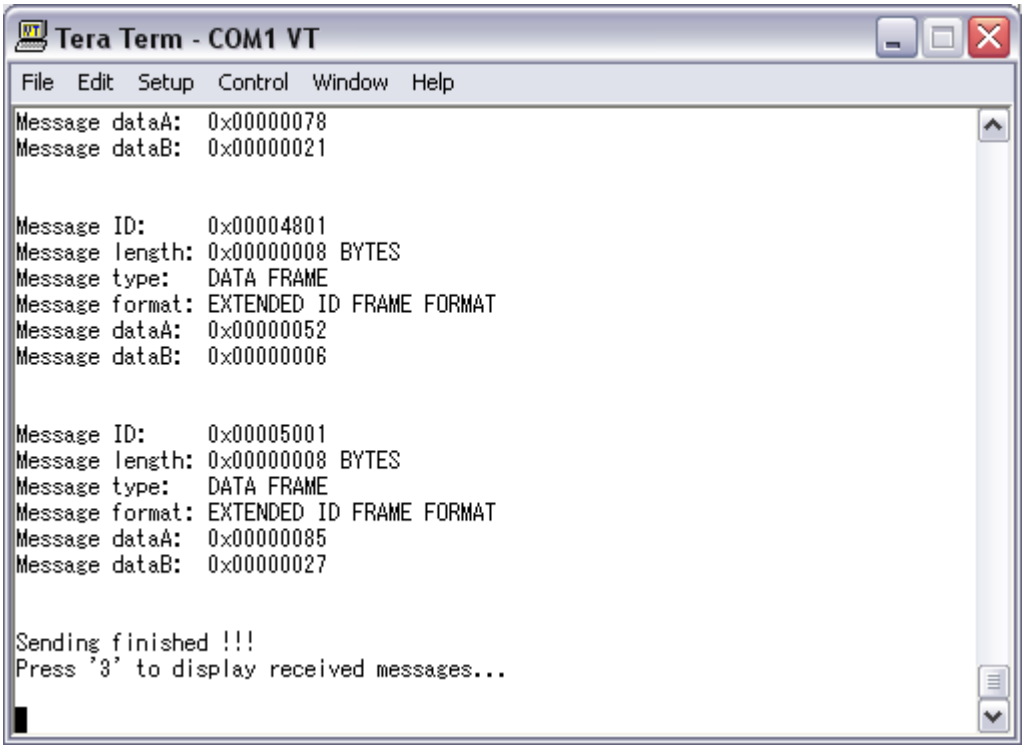

Fig 24. CAN sent messages

Sending finised. Press ‘3’ to display received message. We received message 0,2,4,6,8 like this:

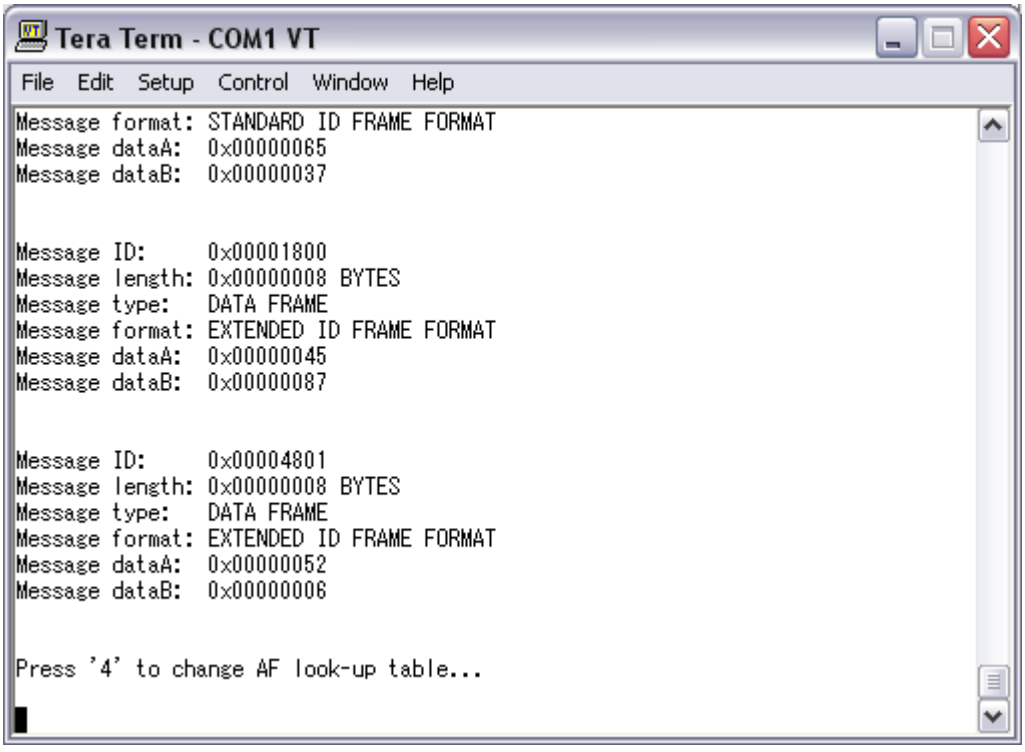

Fig 25. CAN sent messages

Press '4' to change AFLUT dynamically, 5 entries will be add and 5 other entries will be removed out of AF look-up table.

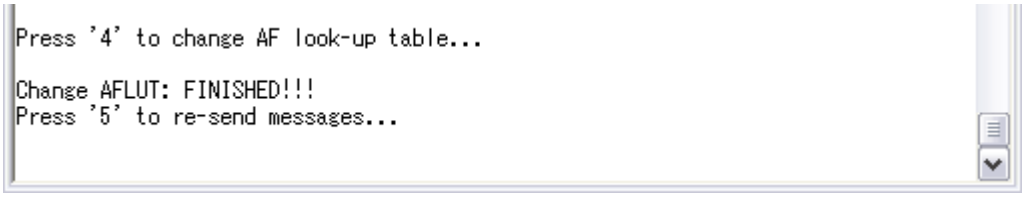

Fig 26. CAN change AFLUT

Press '5' to re-send 10 message above…

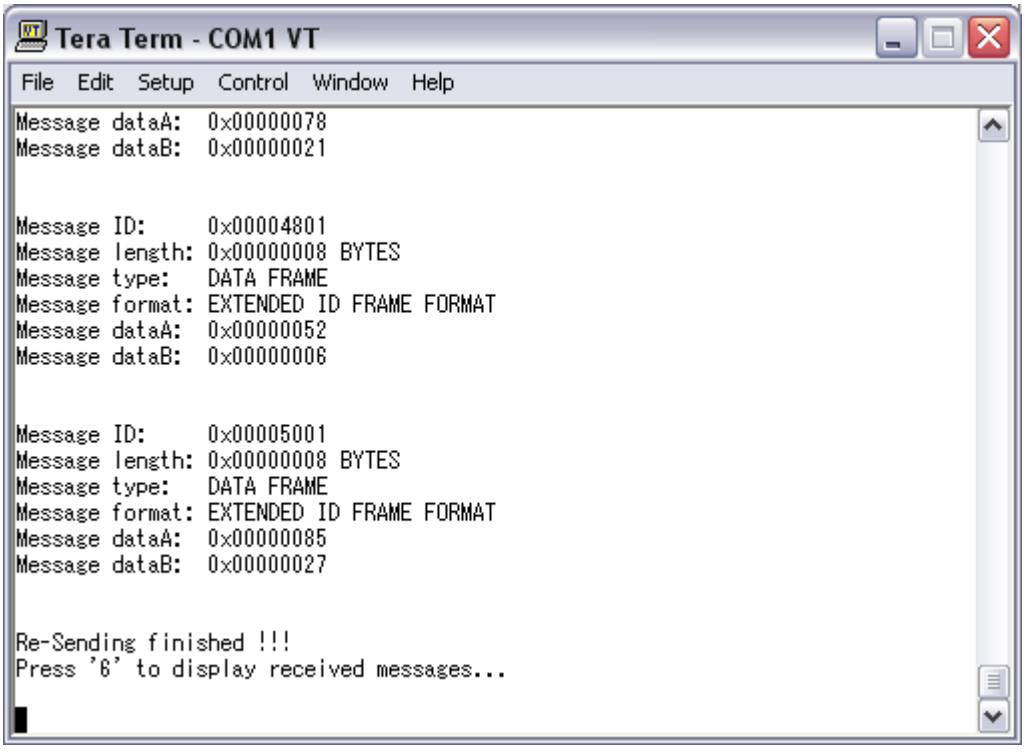

Fig 27. CAN re-sent messages

After change AFLUT, we received 5 messages 1,3,5,7,9 instead of 5 messages 0,2,4,6,8.

Press ‘6’ to display received message…

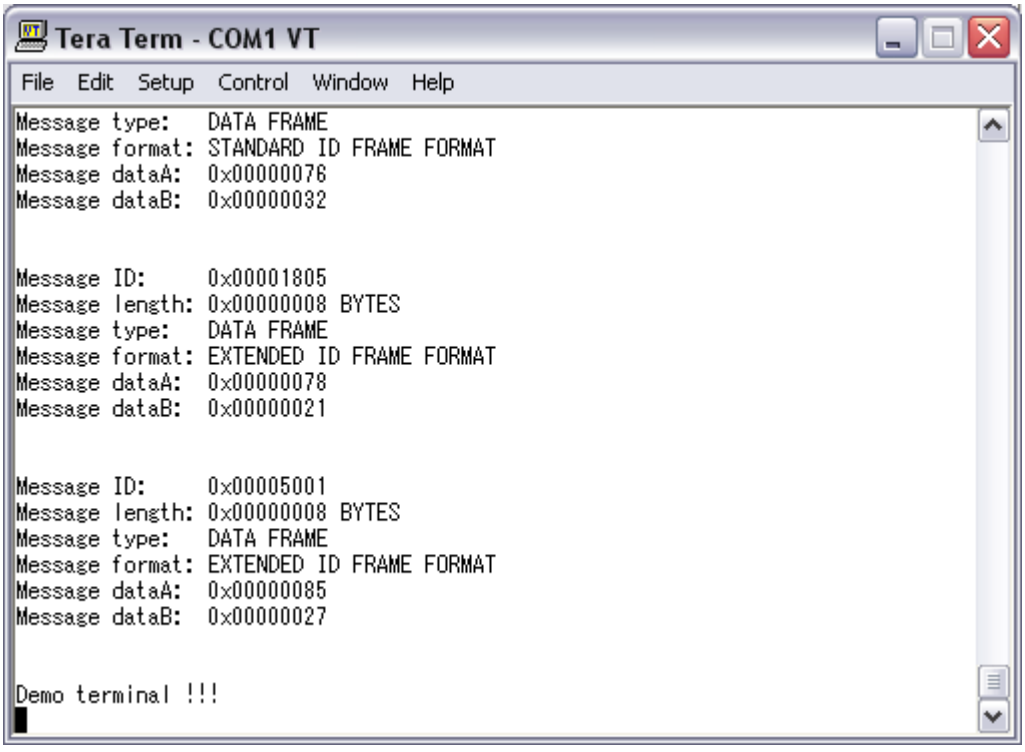

Fig 28. CAN received messages

16. CAN test in two board example

16.1 Purpose

Use 2 CAN1 channels in separate boards to transfer data with each other

16.2 Hardware configuration

CAN1-Port2 of two boards connect with each other.

CAN1-Port7 of two boards connect with each other.

16.3 Software configuration

Required files

can_test_two_kit.c

Running mode

The first board is used to transmit messages run in FLASH mode.

The second board is used to receive messages run in RAM mode.

16.4 Procedure

Setting macro “CAN_TRANSMIT” = 1 before building this program in FLASH mode and port into first board.

```
#define CAN_TRANSMIT        0
#define CAN_RECEIVE         !CAN_TRANSMIT
#define TX_BUFFER_SIZE      4
#define RX_BUFFER_SIZE      2
```

Fig 29. Setting macro

After reset, CAN sends messages immediately…

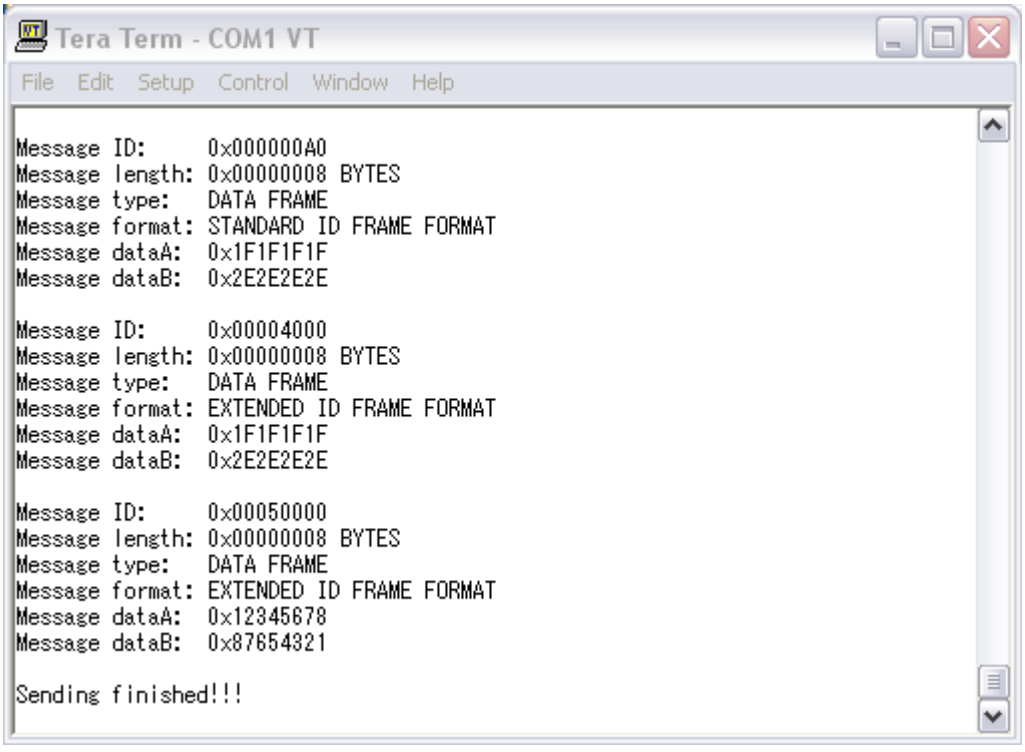

Fig 30. CAN SLAVE sends messages

Set macro to “CAN_TRANSMIT” = 0 before building this program in RAM mode.

At first, the welcome screen appears like this:

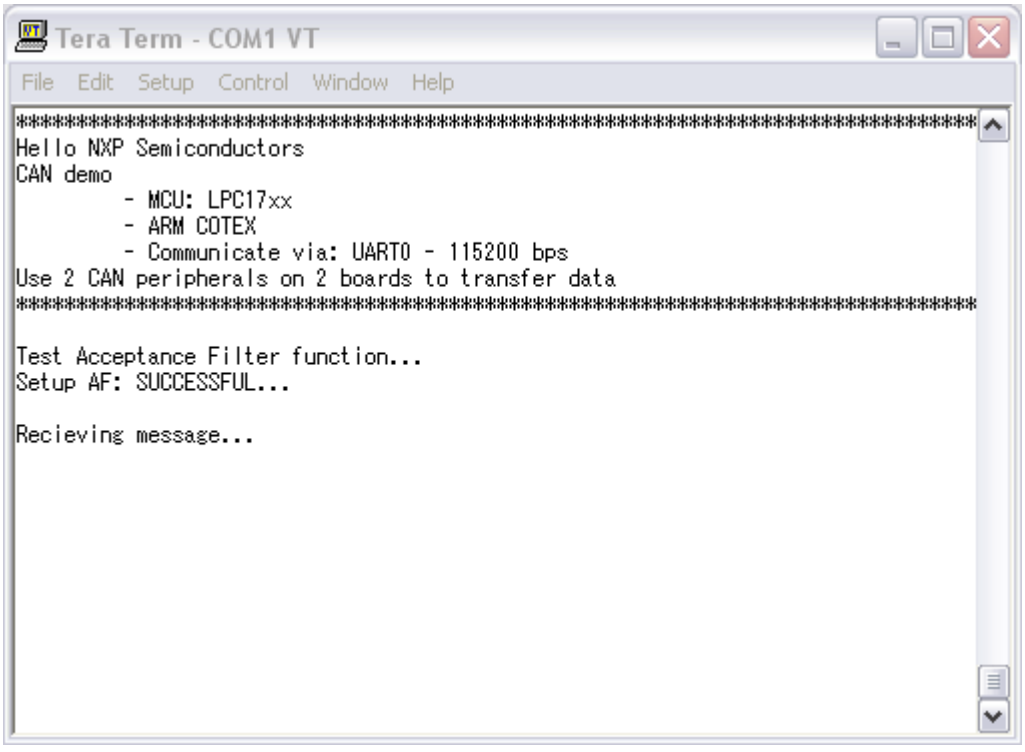

Fig 31. CAN MASTER welcome screen

Press “RESET” button in the first board to start sending messages to second board.

Received message will be displayed like this:

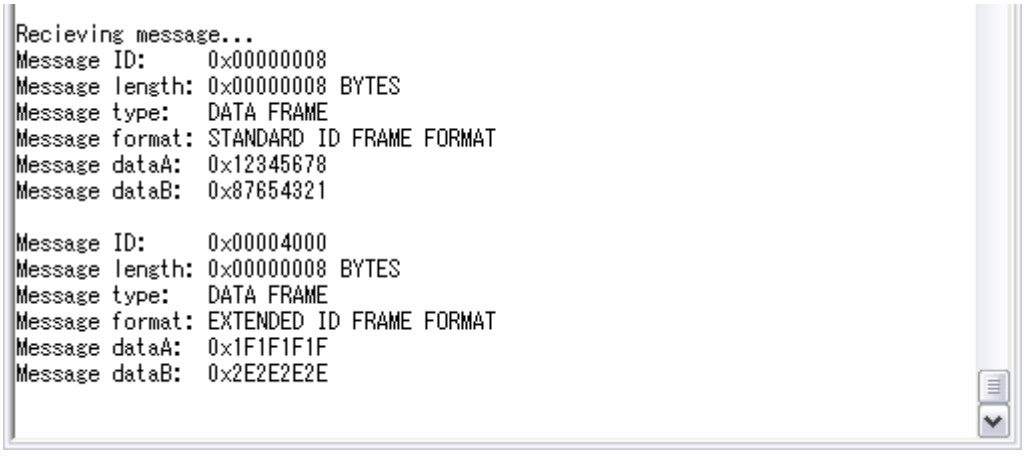

Fig 32. CAN messages were received after transferring finished

17. DAC test example

17.1 Purpose

This is a D/A conversion example: Write the new DAC value that increased by the time and output to speaker.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this transfer.

17.2 Hardware configuration

Please see abstract.txt file for more details.

17.3 Software configuration

Required files

dac_test.c

Running mode

Please see abstract.txt file for more details.

17.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

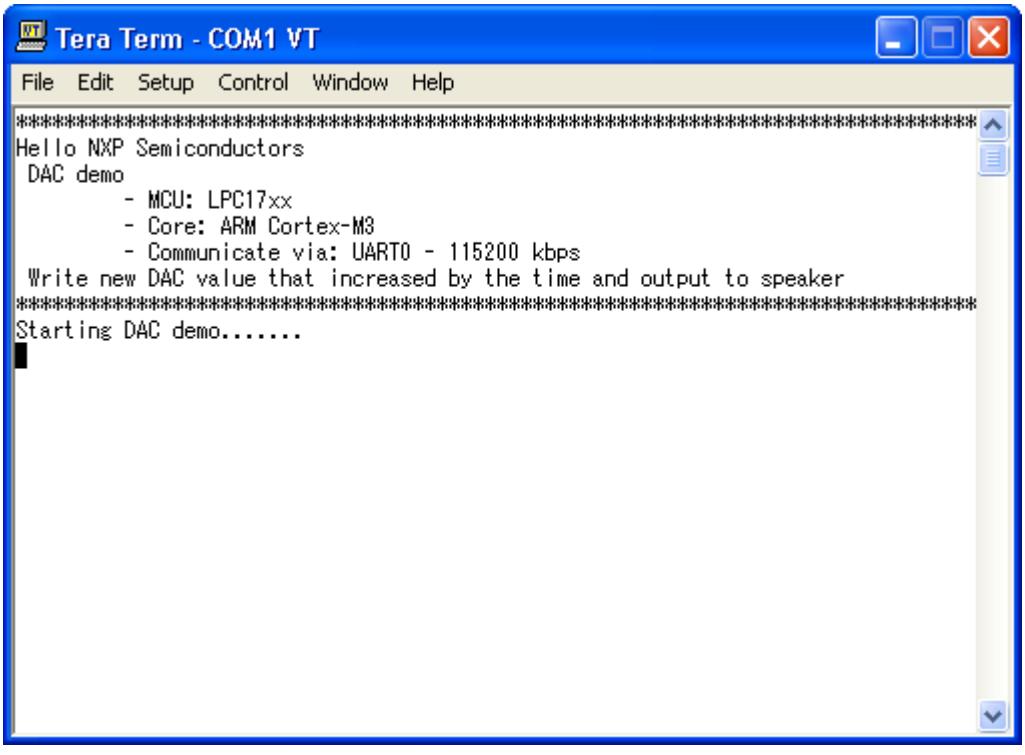

Fig 33. DAC test demo

18. DAC DMA example

18.1 Purpose

This is a DMA example to apply for transfer memory to DAC peripheral

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this transfer.

18.2 Hardware configuration

Please see abstract.txt file for more details.

18.3 Software configuration

Required files

dac_dma.c

Running mode

Please see abstract.txt file for more details.

18.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

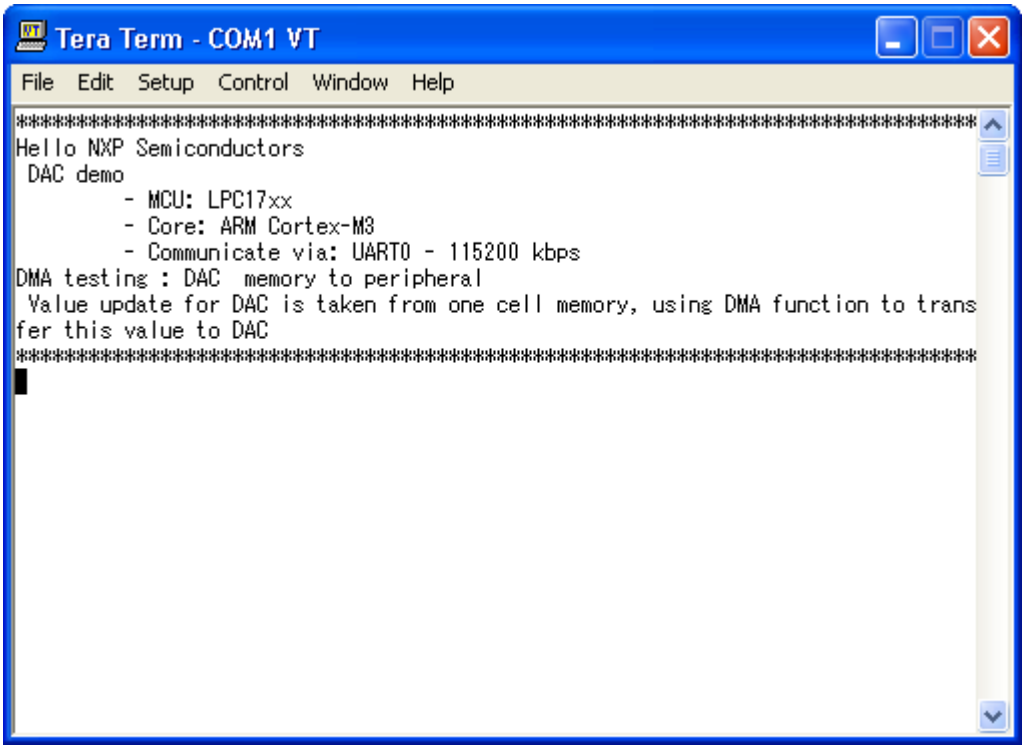

Fig 34. DAC DMA test demo

19. EMAC – Raw example

19.1 Purpose

This example is used to test an EMAC driver with raw packet frame format that is not related with any upper-layer (i.e. TCP/IP...).

See more in abstract.txt file.

19.2 Hardware configuration

Please see abstract.txt file for more details.

19.3 Software configuration

Required files

emac_test.c contain main application.

Running mode

Please see abstract.txt file for more details.

19.4 Procedure

This example can be built into two modes of operation:

- One for 'TX_ONLY' side.

- The other for 'BOUNCE_RX' side.

+ Burn image code into two MCB1700 boards.

+ Hit reset button on two boards.

+ Wait for EMAC initilization completes on two board.

+ If ENABLE_WOL is enabled on board 'BOUNCE_RX' side, after initializing EMAC, it will enter sleep mode to be waked-up on LAN (WoL).

+ On 'TX_ONLY' side, hit INT0 button to send a frame.

+ After receiving frame, 'BOUNCE_RX' side will be waked-up and operates properly.

Please see abstract.txt file for more details.

The screen will be like this:

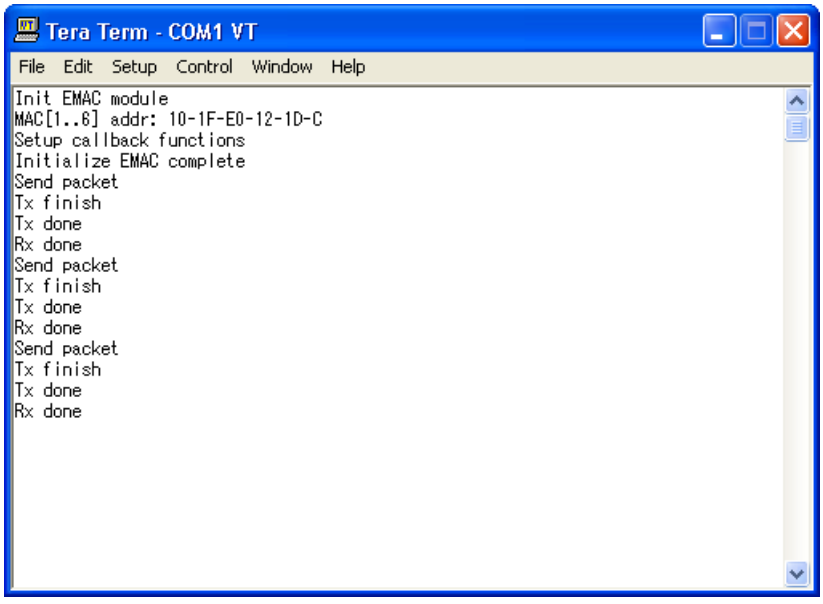

Fig 35. Status on 'TX_ONLY' side

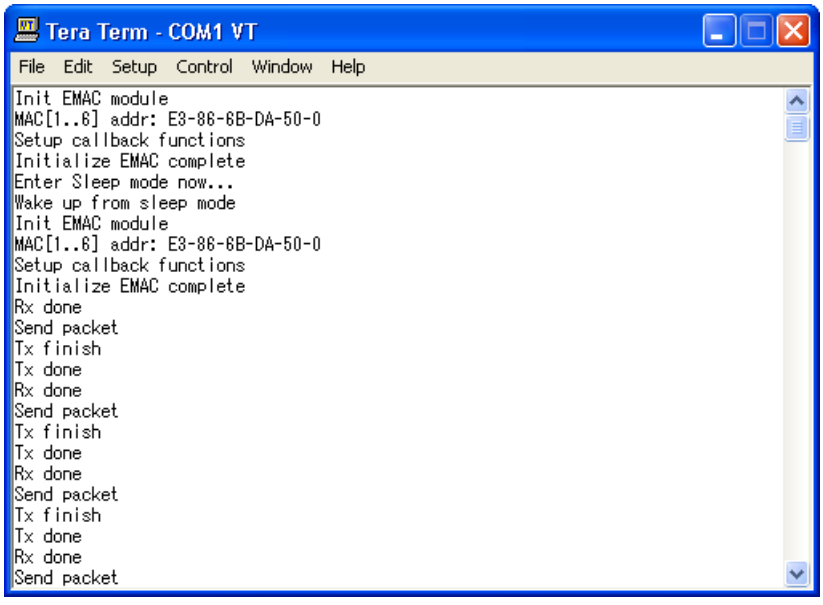

Fig 36. Status on 'BOUNCE_RX' side

20. EMAC – EasyWeb example

20.1 Purpose

An example demo using EasyWeb application to test EMAC driver on LPC1768.

Use UART0 – 115200bps – No parity – No FlowControl to display the status information.

Please see abstract.txt file for more details.

20.2 Hardware configuration

Please see abstract.txt file for more details.

20.3 Software configuration

Required files

easyweb.c contain mainfunction.

Running mode

Please see abstract.txt file for more details.

20.4 Procedure

- Use CrossOver cable to connect from your PC to MCB1700 board.
- Set IP and subnet mask on your PC, i.e. 192.168.0.200 and 255.255.255.0 are used in this case.
- Hit reset button, monitor the status via UART0 until EMAC initialized.
- Open command prompt window, execute ‘ping 192.168.0.100’ command.
- Open web browser, access to address ‘http://192.168.0.100’ to display the content of webserver.

Please see abstract.txt file for more detail.

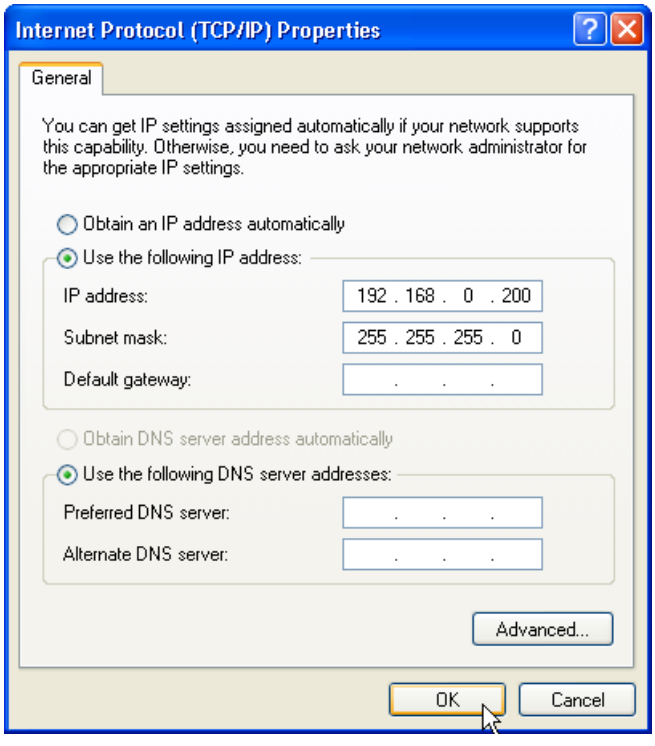

Fig 37. Configure IP address on your PC

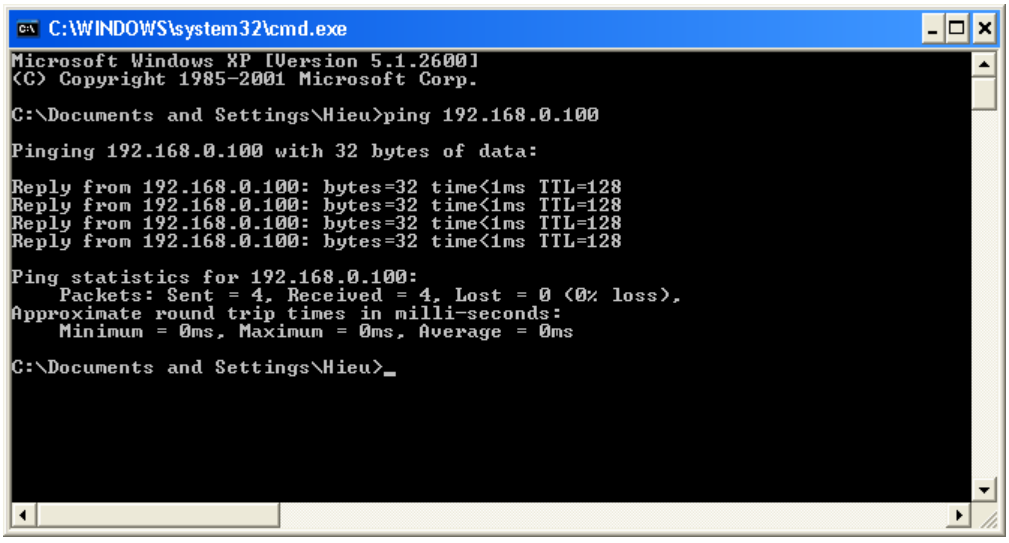

Fig 38. 'ping' command

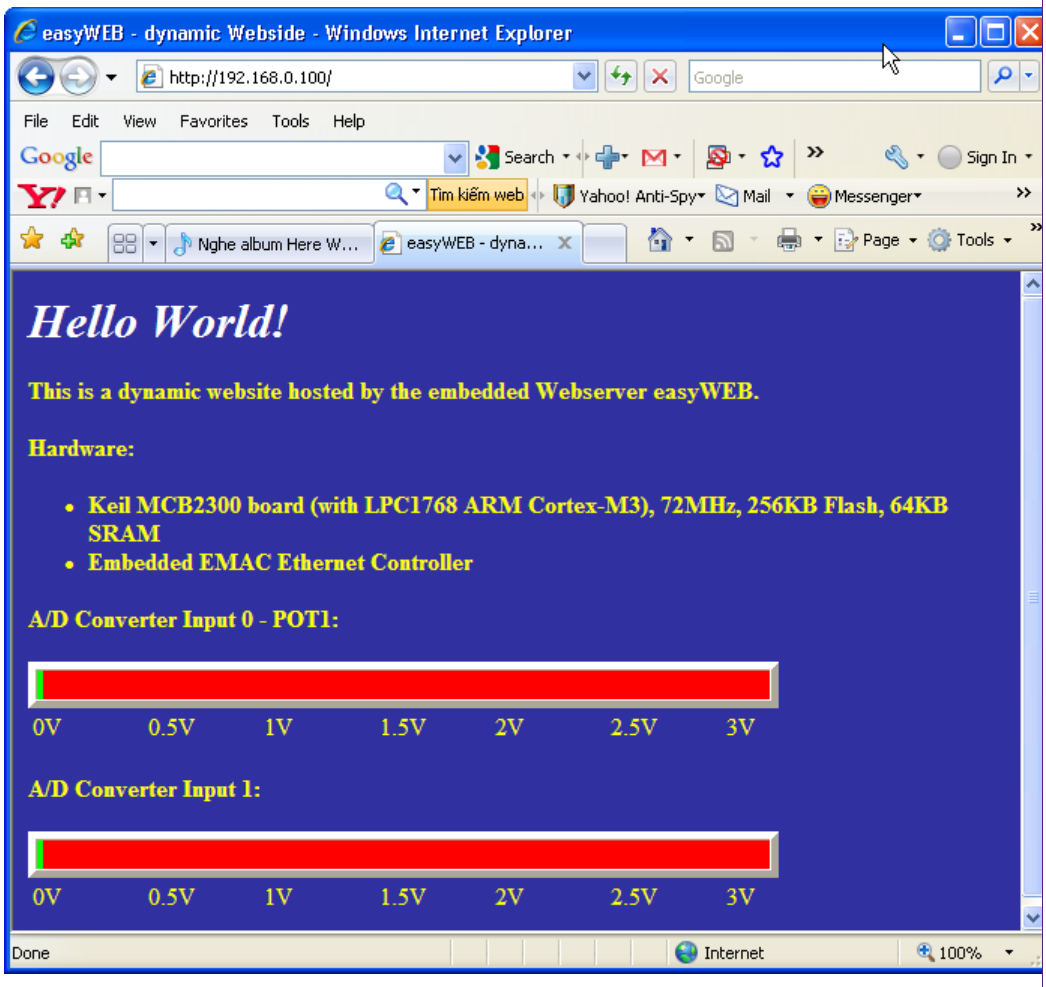

Fig 39. Work with web browser

21. EMAC – uIP example

21.1 Purpose

An example demo using uIP stack with HTTP server to test an EMAC driver on LPC1768.

Use UART0 – 115200bps – No parity – No FlowControl to display the status information.

Please see abstract.txt file for more details.

21.2 Hardware configuration

Please see abstract.txt file for more details.

21.3 Software configuration

Required files

main.c under .\lpc17xx_port' contain main application.

Running mode

Please see abstract.txt file for more details.

21.4 Procedure

Procedure in this example is similar to EMAC-EasyWeb example.

Please see abstract.txt file for more details.

The screen will be like this:

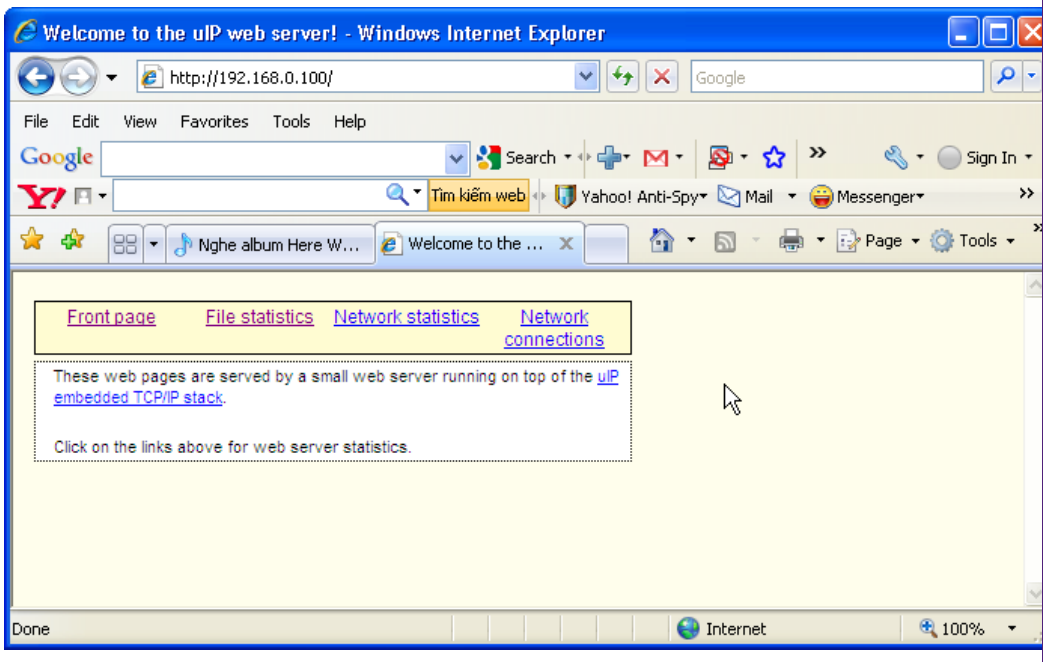

Fig 40. Work with Web browser

22. GPDMA example

22.1 Purpose

This example will transfer 2 blocks of data from memory boundary (AHBRAM1_BASE - USB RAM) to the other memory boundary on RAM using GPDMA module with interrupt.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display information.

22.2 Hardware configuration

Please see abstract.txt file for more details.

22.3 Software configuration

Required files:

gpdma_m2m_test.c

Running mode

Please see abstract.txt file for more details.

22.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

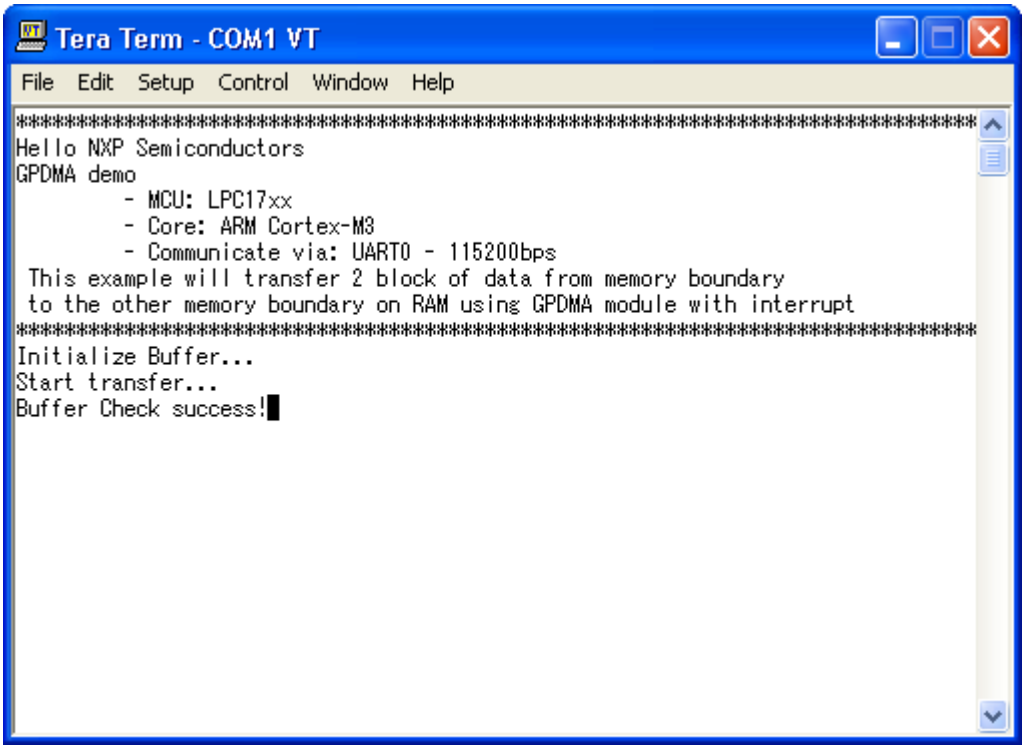

Fig 41. GPDMA demo

23. GPIO External Power down example

23.1 Purpose

An example using external interrupt on INT0 as wake up source in each power mode:

- Sleep
- Deep sleep
- Power down

23.2 Hardware configuration

Please see abstract.txt file for more details.

23.3 Software configuration

Required files

eint_powerdown_test.c

Running mode

This example can run on FLASH mode

23.4 Procedure

After reset , LED will blink a few times, the system will enter to target power down mode.

Hit INT0 button to wake it up

24. GPIO interrupt example

24.1 Purpose

A simple program to test external interrupt on INT0 and GPIO interrupt on P0.25

24.2 Hardware configuration

Please see abstract.txt file for more details.

24.3 Software configuration

Required files

gpio_int.c

Running mode

Please see abstract.txt file for more details.

24.4 Procedure

Please see abstract.txt file for more details.

25. GPIO Blinky

25.1 Purpose

This is a simple program to test GPIO interrupt functionality to drive an LED.

25.2 Hardware configuration

Please see abstract.txt file for more details.

25.3 Software configuration

Required files

LedBlinky.c

Running mode

Please see abstract.txt file for more details.

25.4 Procedure

Please see abstract.txt file for more details.

26. GPIO – Port LCD

26.1 Purpose

This is a simple program to test GPIO interrupt functionality to drive an LCD on MCB1700.

26.2 Hardware configuration

Please see abstract.txt file for more details.

26.3 Software configuration

Required files

lcdtest.c

Running mode

Please see abstract.txt file for more details.

26.4 Procedure

Please see abstract.txt file for more details.

27. I2C master example

27.1 Purpose

This example uses I2C as a master device to transfer data from/to an I2C slave device.

- First, the master transmits to slave a number of data bytes
- Then, the master receives a number of data bytes from slave.
- Finally, the master sends two bytes to slave, sends a repeat start immediately and receives from slave a number of data bytes.
- Using in polling mode.

27.2 Hardware configuration

Please see abstract.txt file for more details.

27.3 Software configuration

Required files

master.c

Running mode

Please see abstract.txt file for more details.

27.4 Procedure

These steps should be done in sequence as follows:

- The slave must start first to be ready to receive data from master.
- Press ‘1’ to transmit data from master to slave.
- Press ‘2’ to receive.
- Press ‘3’ to transmit, then repeat start and receive.

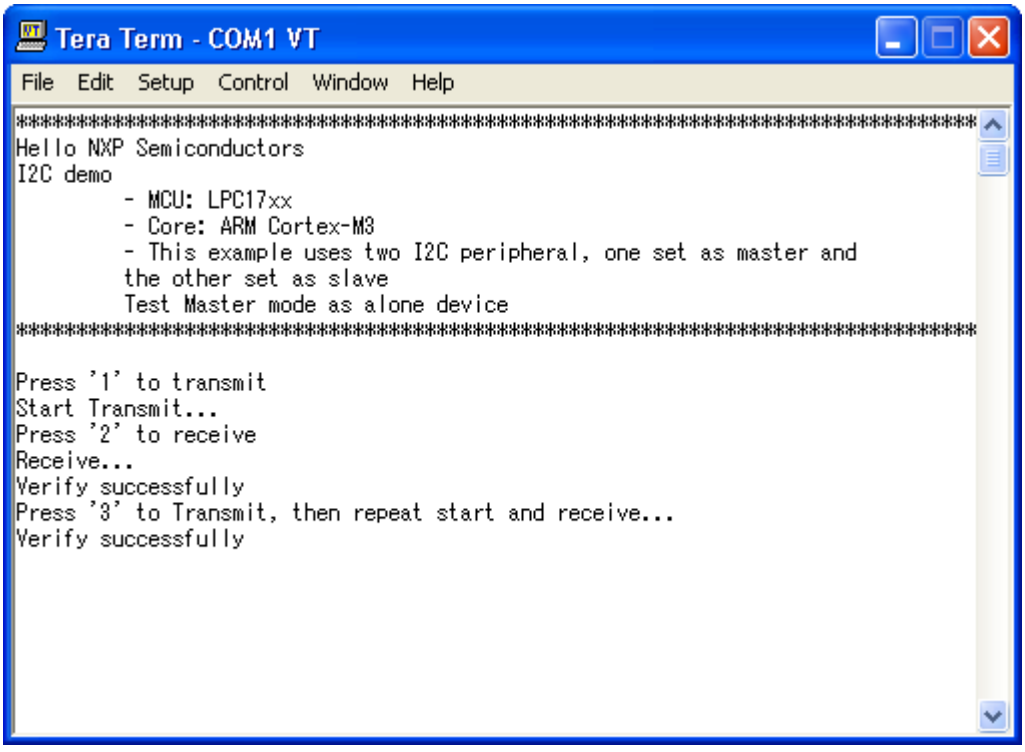

Fig 42. I2C Master demo

28. I2C slave example

28.1 Purpose

This example uses I2C as a slave device to transfer data from/to I2C master device.

- First, the master transmits to slave a number of data bytes
- Then, the master receives a number of data bytes from slave.
- Finally, the master sends two bytes to slave, sends a repeat start immediately and receives from slave a number of data bytes.
- Using in polling mode.

28.2 Hardware configuration

Please see abstract.txt file for more details.

28.3 Software configuration

Required files

slave.c

Running mode

Please see abstract.txt file for more details.

28.4 Procedure

Press '1' to start communication with master.

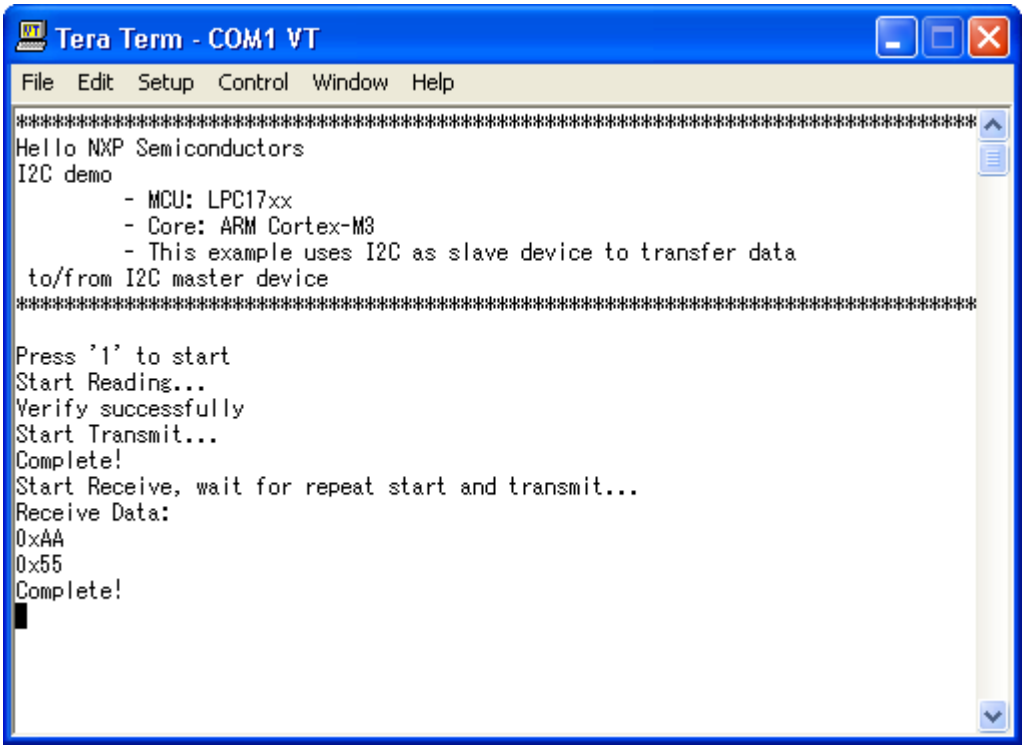

Fig 43. I2C slave demo

29. I2C Master-Slave interrupt example

29.1 Purpose

This example uses two I2C peripherals on the same chip LPC1768, one set as master and the other set as slave.

- First, the master transmits to slave a number of data bytes
- Then, the master receives a number of data bytes from slave.
- Finally, the master sends two bytes to slave, sends a repeat start immediately and receives from slave a number of data bytes.
- Both of them are used in interrupt mode.

29.2 Hardware configuration

Please see abstract.txt file for more details.

29.3 Software configuration

Required files

i2c_master_slave_int_test.c

Running mode

Please see abstract.txt file for more details.

29.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

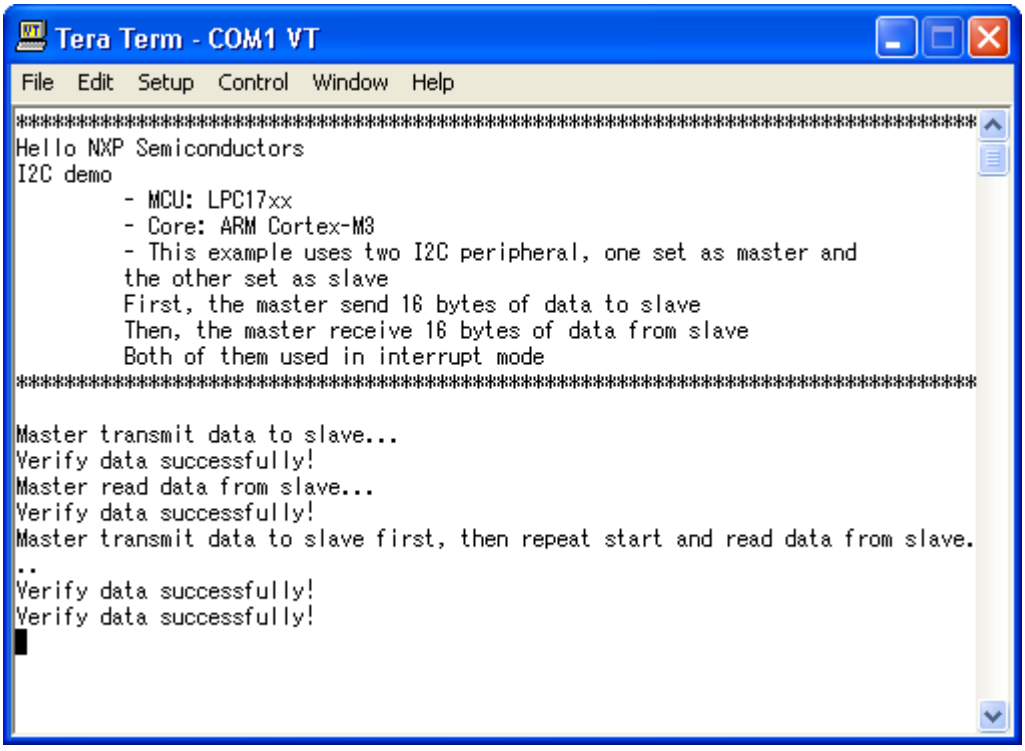

Fig 44. I2C Master-Slave interrupt demo

30. I2C - PCA8581 polling example

30.1 Purpose

This is an example of I2C using polling mode to test the I2C driver.

Using EEPROM PCA8581 to transfer a number of data byte.

30.2 Hardware configuration

Please see abstract.txt file for more details.

30.3 Software configuration

Required files

pca8581_test.c

Running mode

Please see abstract.txt file for more details.

30.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

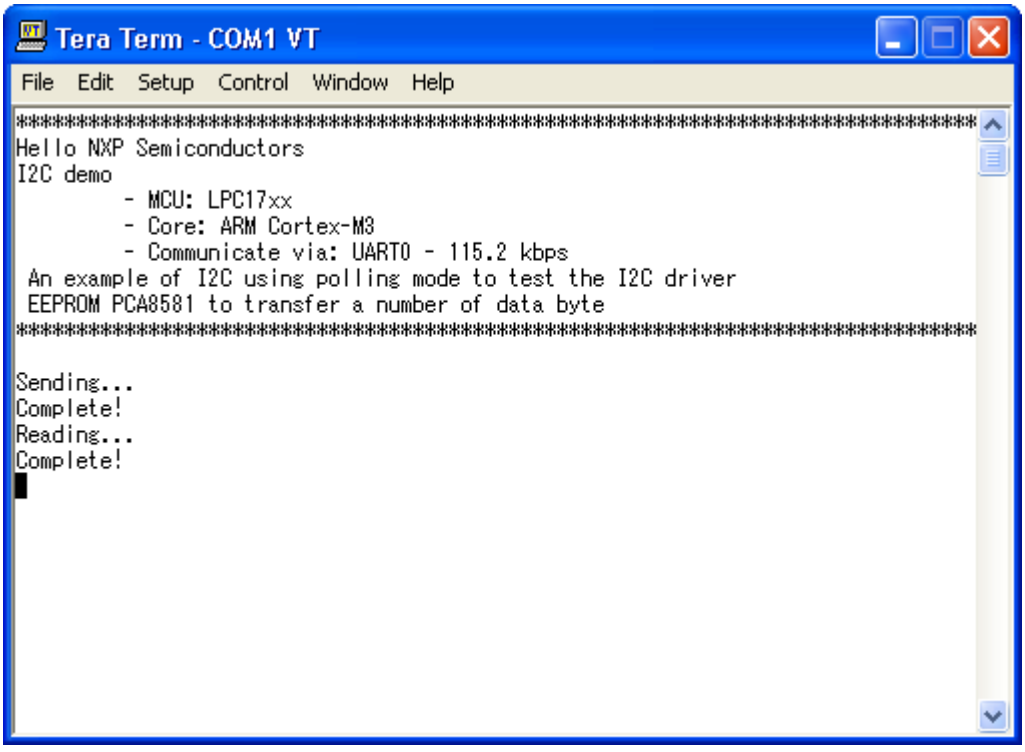

Fig 45. I2C pca8581 polling test demo

31. I2C - SC16IS750 polling test example

31.1 Purpose

This is an example of I2C using polling mode to test the I2C driver.

Using I2C at mode I2C master/8bit on LPC1768 to communicate with SC16IS750/760 Demo Board.

31.2 Hardware configuration

Please see abstract.txt file for more details.

31.3 Software configuration

Required files

i2c_polling_test.c

Running mode

Please see abstract.txt file for more details.

31.4 Procedure

- Press ‘1’ to turn ON LEDs, ‘2’ to turn OFF LEDs.

Please see abstract.txt file for more details.

The screen will be like this:

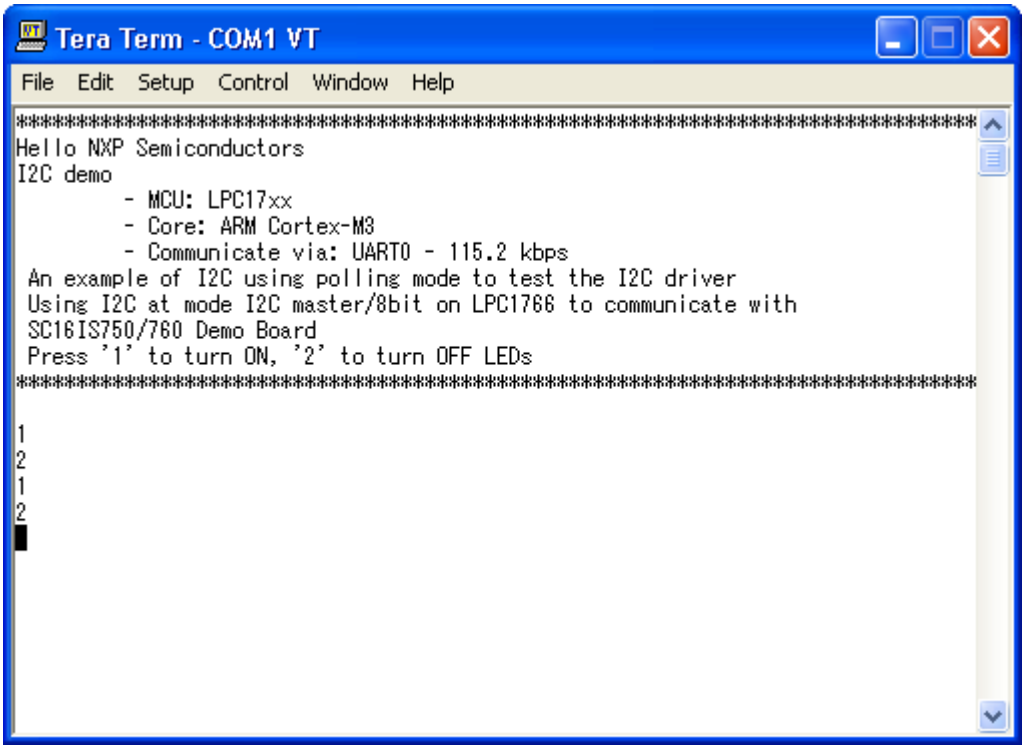

Fig 46. I2C SC16IS750 polling demo

32. I2C SC16IS750 interrupt example

32.1 Purpose

This is an example of I2C using interrupt mode to test the I2C driver.

Using I2C at mode I2C master/8bit on LPC1766 to communicate with SC16IS750/760 Demo Board

32.2 Hardware configuration

Please see abstract.txt file for more details.

32.3 Software configuration

Required files

i2c_interrupt_test.c

Running mode

Please see abstract.txt file for more details.

32.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

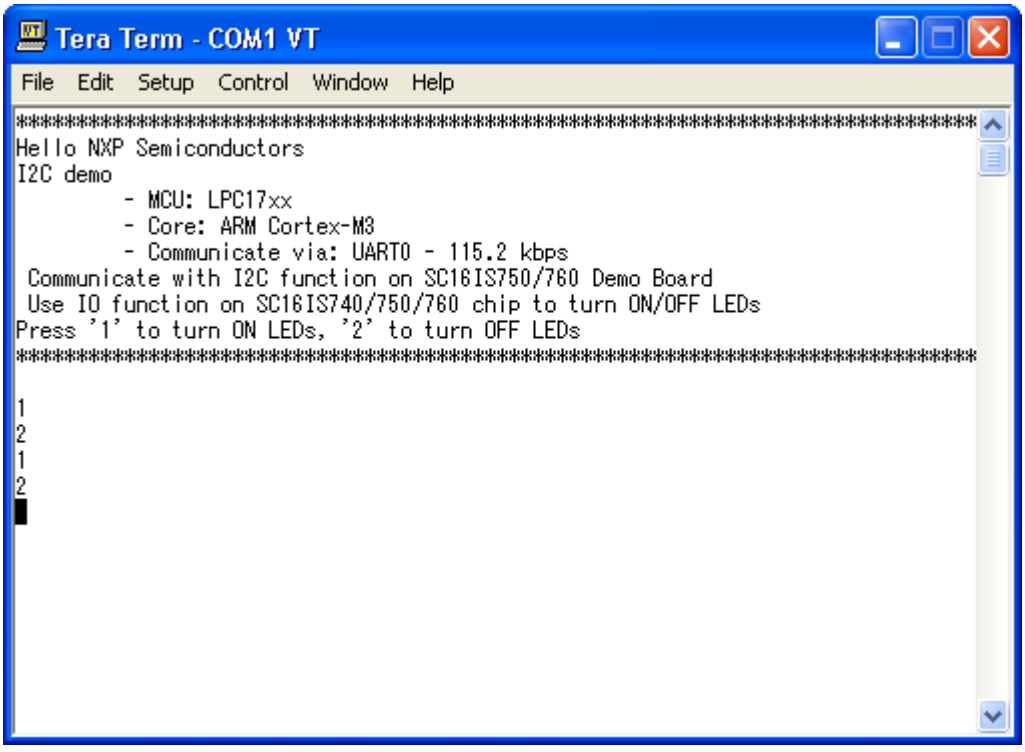

Fig 47. I2C SC16IS750 interrupt demo

33. I2S self-test in polling mode example

33.1 Purpose

Use two I2S channels in the same board to send/receive data in polling mode.

33.2 Hardware configuration

Pin P0.4 connects to Pin P0.7.

Pin P0.5 connects to Pin P0.8.

Pin P0.6 connects to Pin P0.9.

33.3 Software configuration

Required files

i2s_polling.c

Running mode

Default

33.4 Procedure

After restart, the welcome screen appears like this:

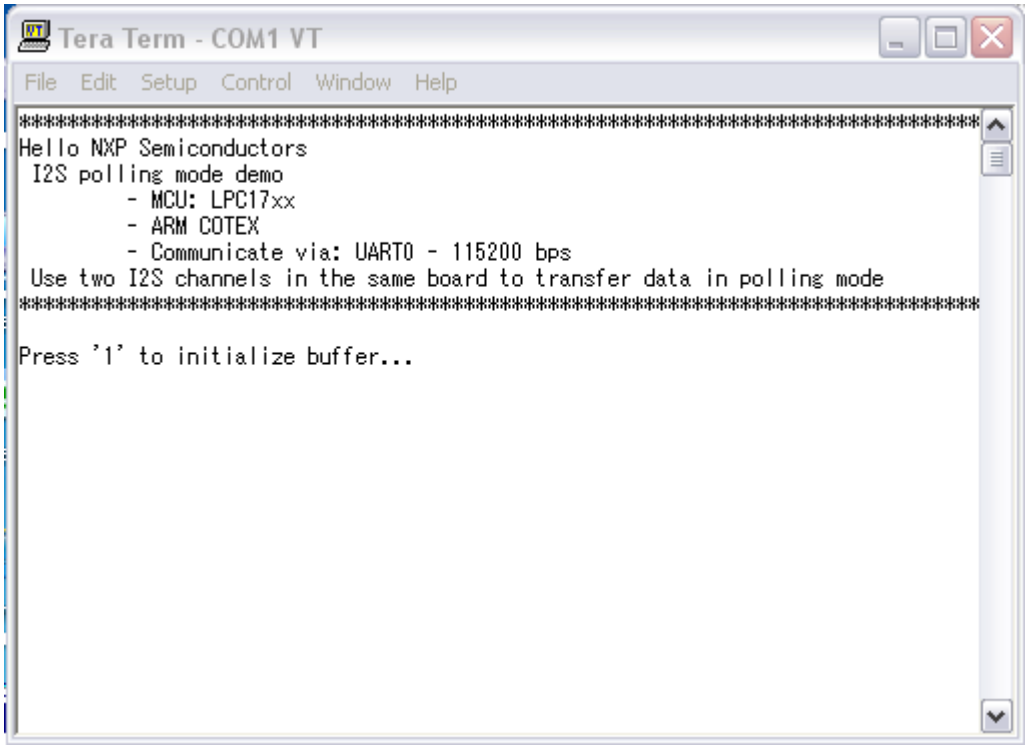

Fig 48. I2S Welcome Screen in polling mode

Press '1' to initialize data…

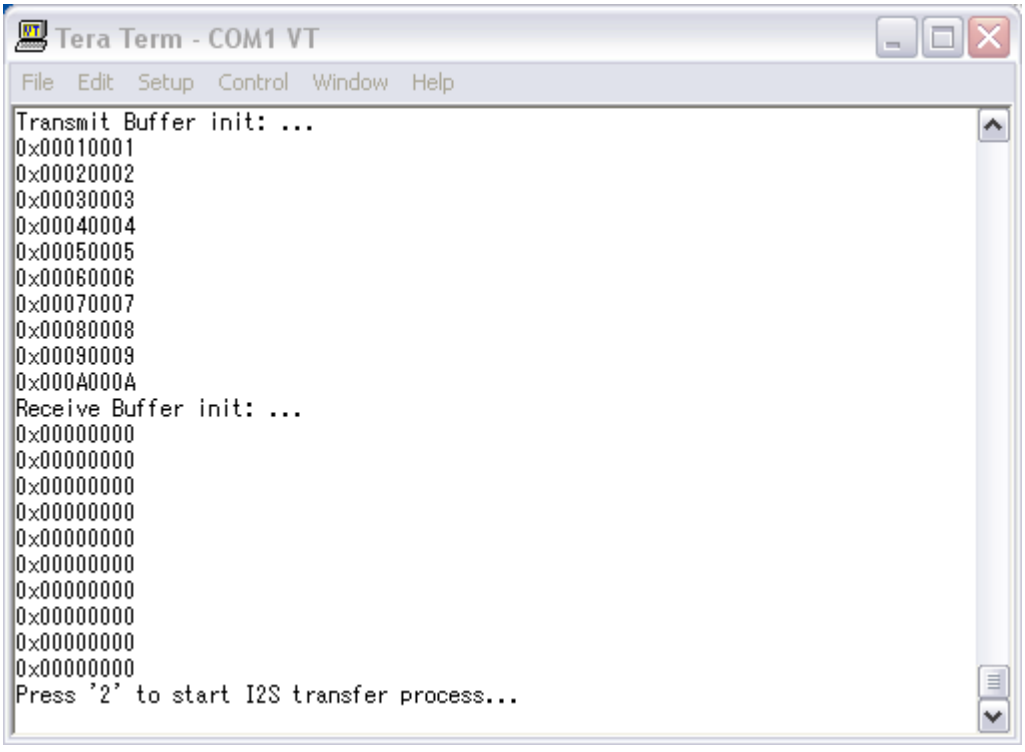

Fig 49. I2S Transmit and Receive Buffer data initialized after press '1'

Press '2' to start I2S transfer process….After I2S process finished, Receive Buffer data will be displayed like that:

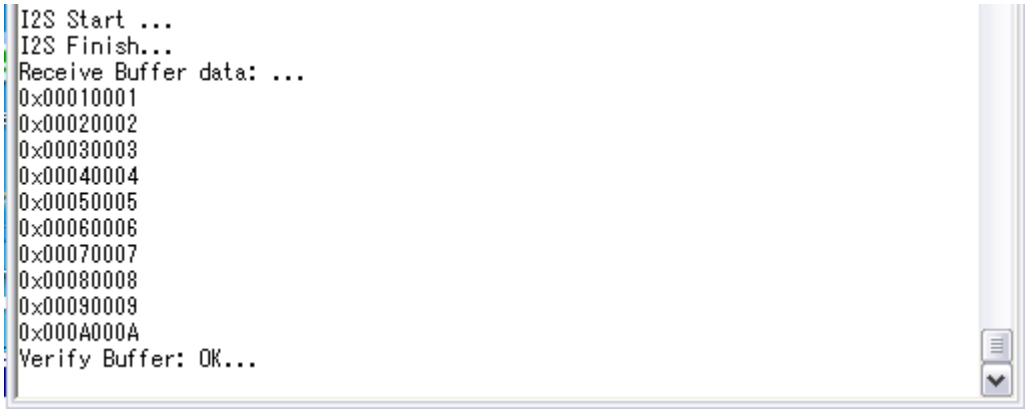

Fig 50. I2S Receive Buffer after transferring completed

34. I2S self-test in interrupt example

34.1 Purpose

Use two channels (I2S Transmit and Receive) in the same board to send/receive data in interrupt mode.

34.2 Hardware configuration

Pin P0.4 connects to Pin P0.7.

Pin P0.5 connects to Pin P0.8.

Pin P0.6 connects to Pin P0.9.

34.3 Software configuration

Required files

i2s_irq_test.c

Running mode

Default

34.4 Procedure

After restart, the welcome screen appears like this:

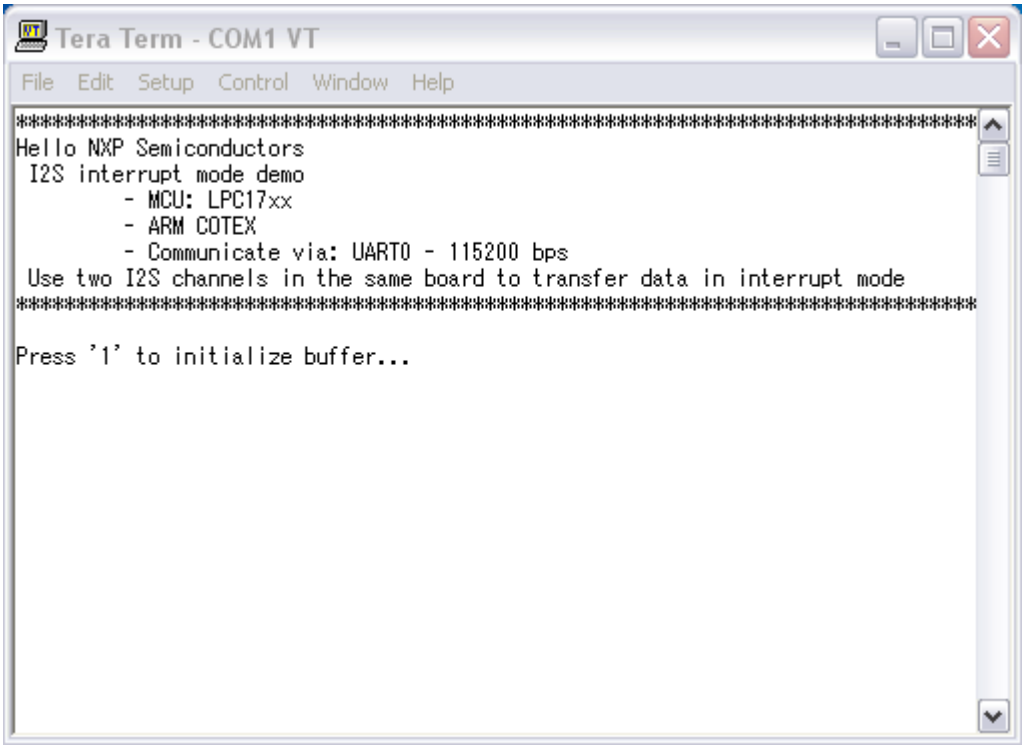

Fig 51. I2S Welcome Screen in interrupt mode

Press ‘1’ to initialize buffer.

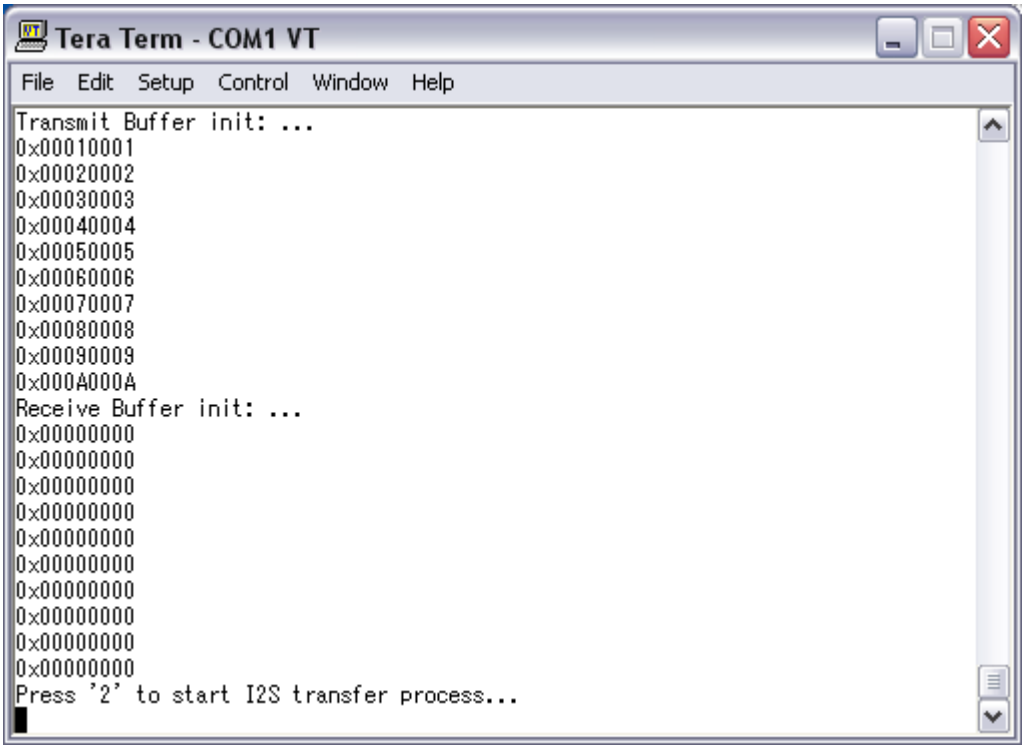

Fig 52. I2S Transmit and Receive Buffer after press ‘1’

Press ‘2’ to start I2S transfer process. After the I2S process is finished, the Receive Buffer will be displayed as shown:

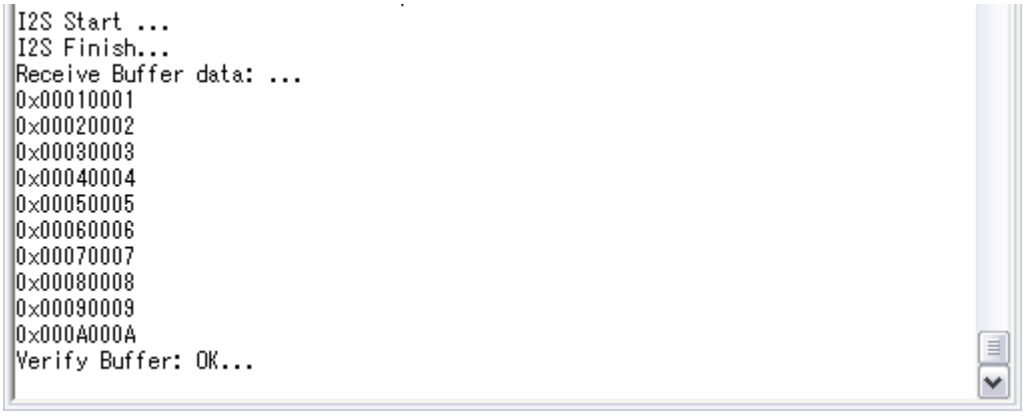

Fig 53. I2S Receive Buffer after transferring completed

35. I2S self-test DMA mode example

35.1 Purpose

Use two channels (I2S Transmit and Receive) in the same board to send/receive data in DMA mode.

35.2 Hardware configuration

Pin P0.4 connects to Pin P0.7.

Pin P0.5 connects to Pin P0.8.

Pin P0.6 connects to Pin P0.9.

35.3 Software configuration

Required files

i2s_dma_test.c

Running mode

Default

35.4 Procedure

After restart, the welcome screen appears like this:

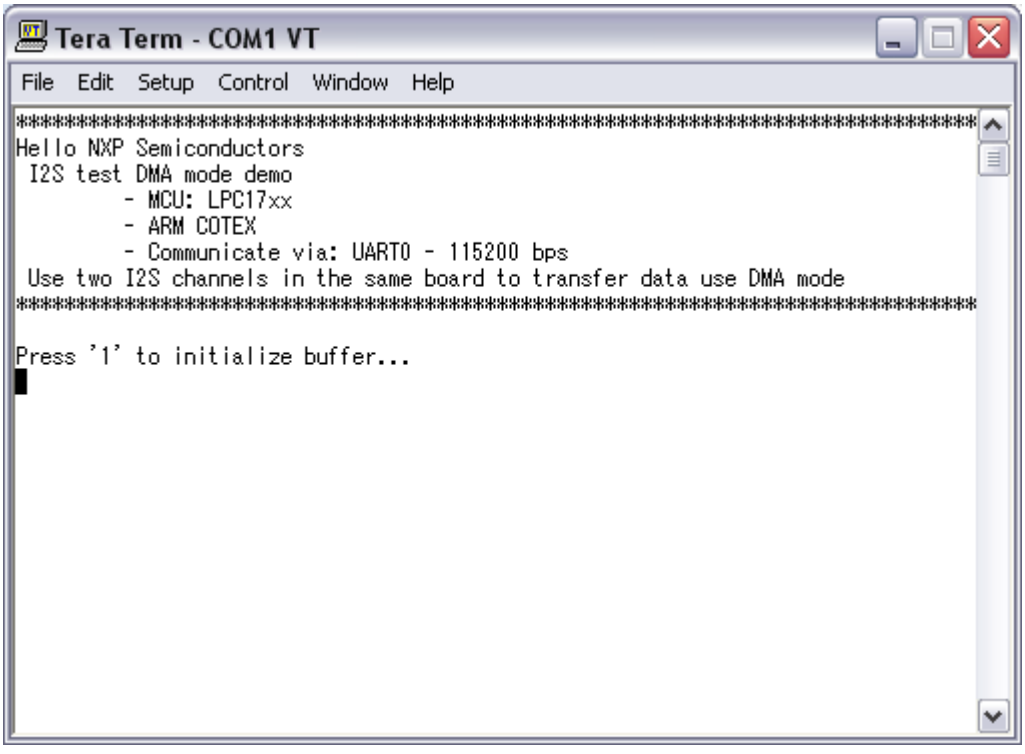

Fig 54. I2S welcome screen in DMA mode

Press ‘1’ to initialize buffer.

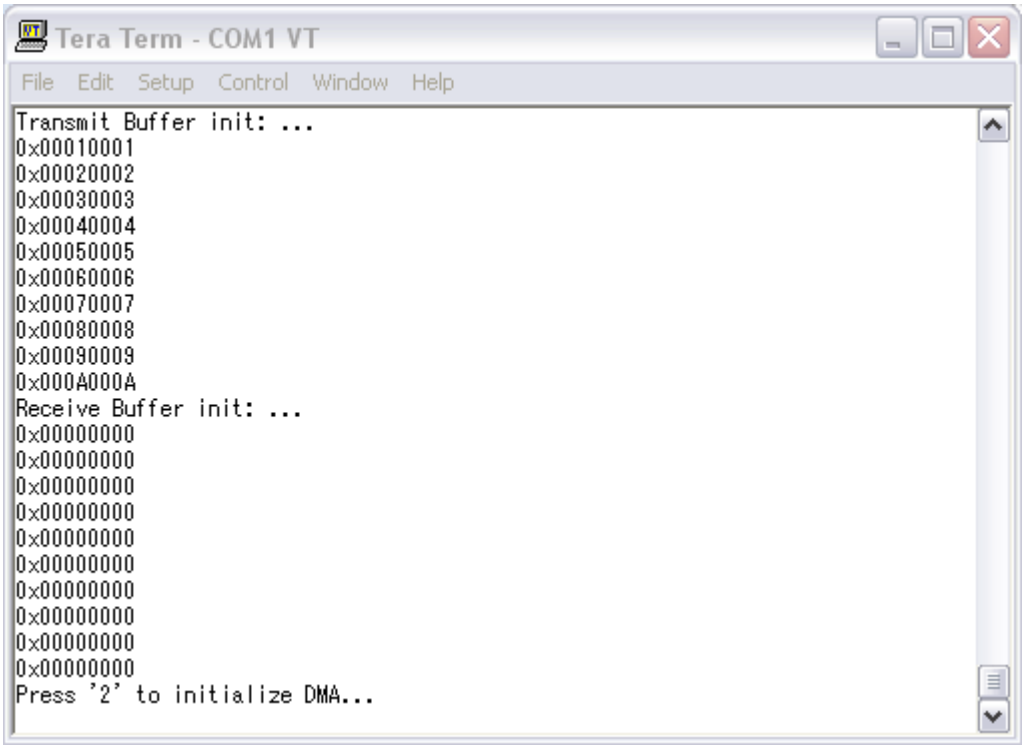

Fig 55. I2S Transmit and Receive Buffer initialized after press ‘1'

Press ‘2’ to initialize DMA.

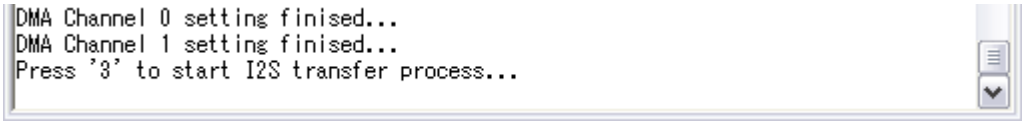

Fig 56. Setting 2 DMA channels

Press ‘3’ to start the I2S transfer process.

After the I2S process is finished, the Receive Buffer data will be displayed as follows:

```
I2S Start...
I2S Finish...
Receive Buffer data: ...
0x00000000 ->Dummy data
0x00010001
0x00020002
0x00030003
0x00040004
0x00050005
0x00060006
0x00070007
0x00080008
0x00090009
0x000A000A
Verify Buffer: OK...
```

Fig 57. I2S Receive Buffer after transferring completed

36. I2S test 4-wire mode

36.1 Purpose

This example used for test 4-wire mode

I2S receiver is set in 4-wire mode, sharing the transmitter bit clock and WS

36.2 Hardware configuration

Pin I2SRX_SDA (P0.4) connect with pin I2STX_SDA(P0.9)

36.3 Software configuration

Required files

i2s_test_4_wire.c

Running mode

Default

36.4 Procedure

After reset, the welcome screen appear like this:

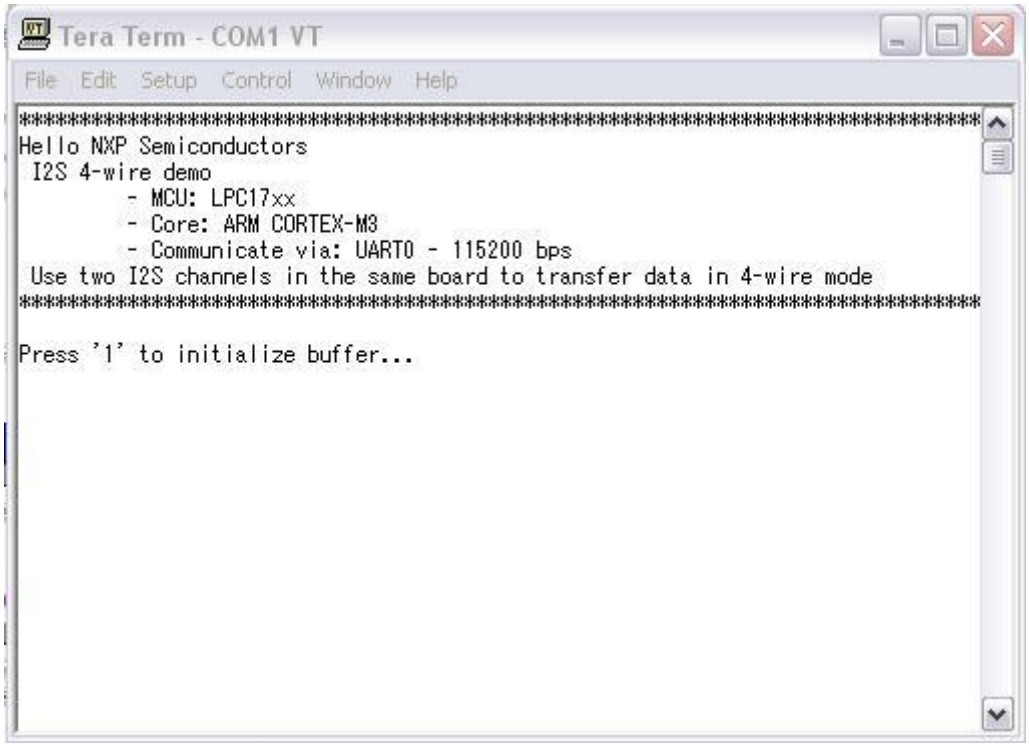

Fig 58. I2S 4-wire welcome screen

Press ‘1’…

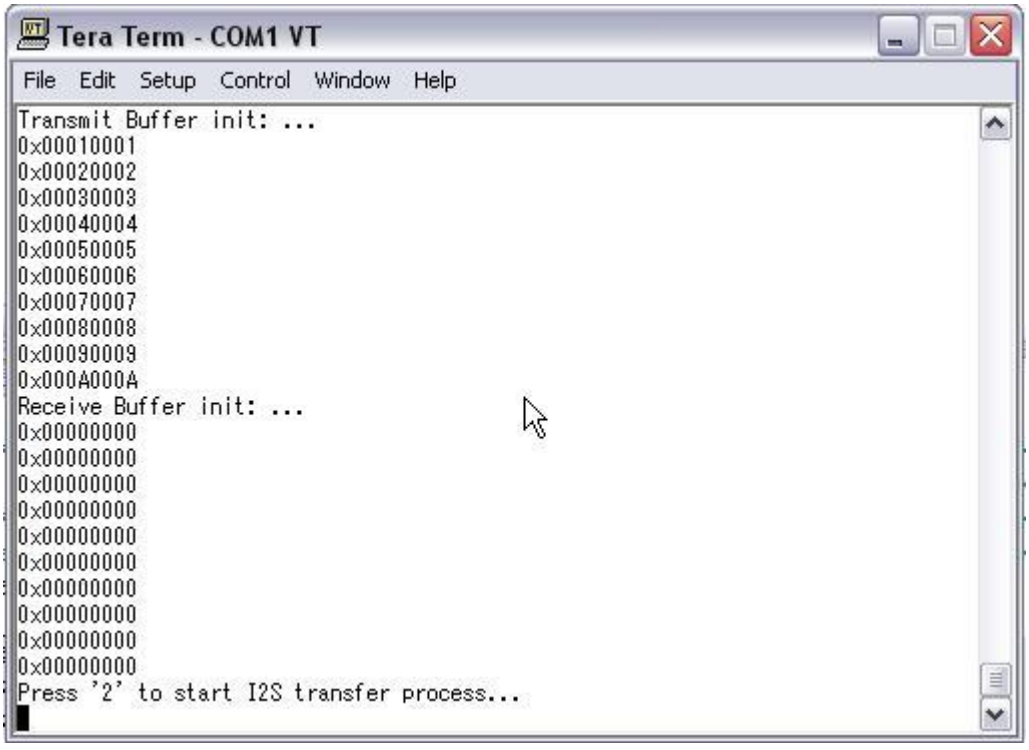

Fig 59. I2S init transmit and receive buffers

Press '2'…

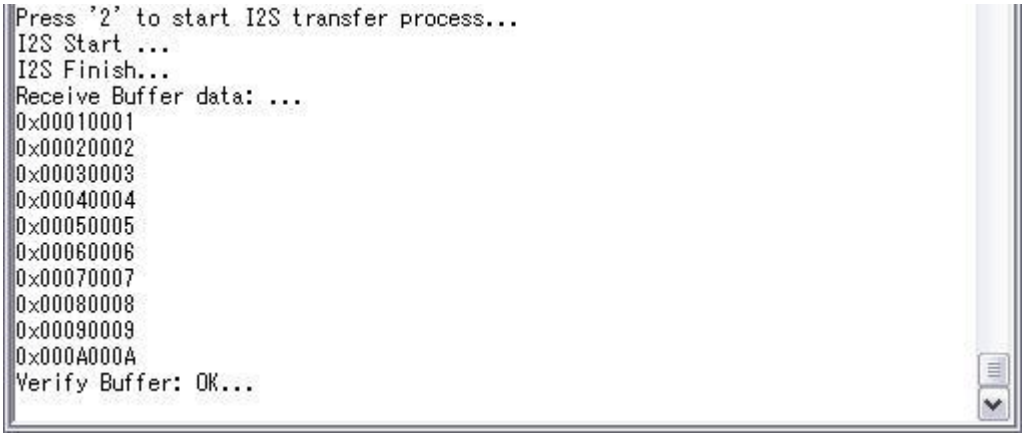

Fig 60. I2S transfer finish

37. I2S Master-Slave mode example

37.1 Purpose

Use two I2S channels in separate boards to transfer data with each other.

37.2 Hardware configuration

Use the first board as a MASTER to receive data.

Use the second board as a SLAVE to transmit data.

Pin P0.4 of board 1 connects to Pin P0.7 of board 2.

Pin P0.5 of board 1 connects to Pin P0.8 of board 2.

Pin P0.6 of board 1 connects to Pin P0.9 of board 2.

37.3 Software configuration

Required files

i2s_two_kit.c

Running mode

Master run in FLASH mode.

Slave run in RAM mode.

37.4 Procedure

Port I2S Master program into the first board.

Setting macro “I2S_TRANSMIT”=1

```
#include "lpc17xx_i2s.h"
#include "lpc17xx_libcfg.h"
#include "lpc17xx_nvic.h"
#include "lpc17xx_pinsel.h"
#include "debug_frmwrk.h"

/************************** PRIVATE MACROS **************************/
#define I2S_TRANSMIT            1
#define I2S_RECEIVE             !I2S_TRANSMIT
```

Fig 61. Setting I2S_TRANSMIT macro

Build and port this program into the first board in FLASH mode.

Then, port I2S Slave program into the second board:

Setting “I2S_TRANSMIT” = 0

Build and port this program into the second board in RAM mode.

After reset, the welcome screen appears like this:

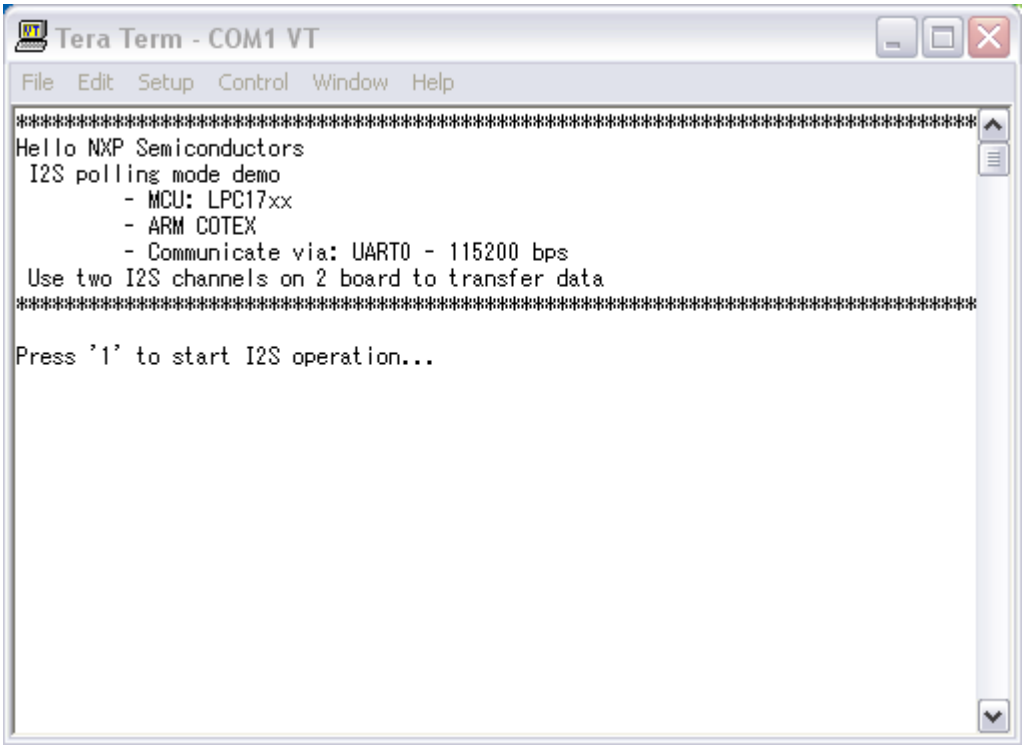

Fig 62. I2S MASTER Welcome Screen

Press '1' to start I2S operation. After that, the received data will be displayed as follows:

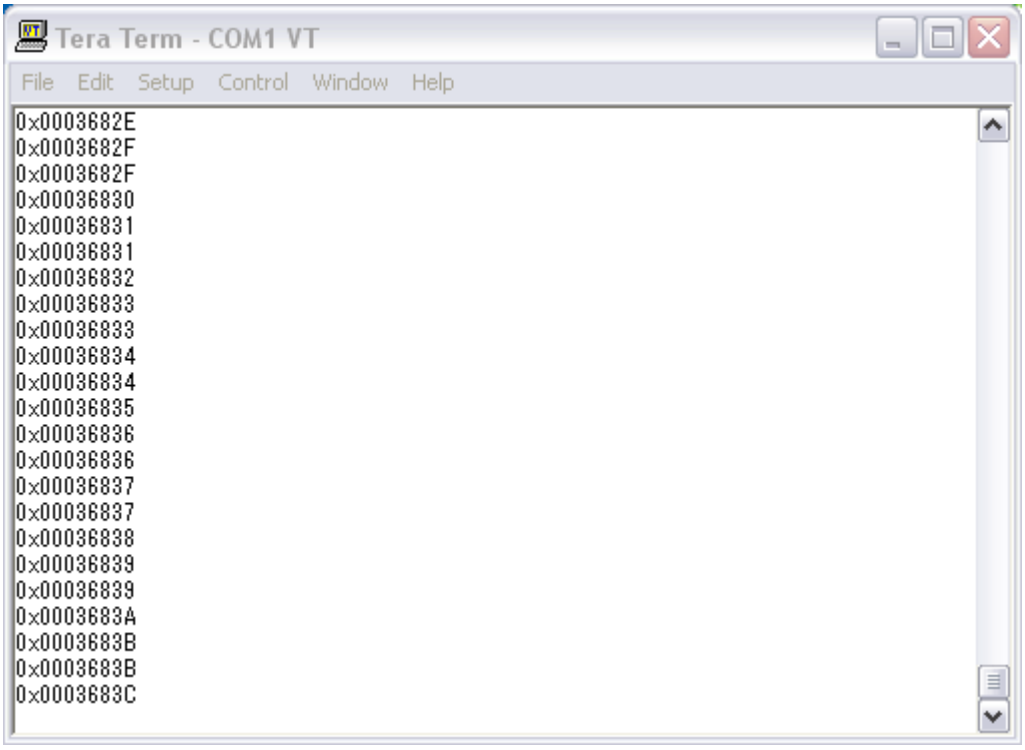

Fig 63. I2S Received data after press '1'

38. MCPWM Simple

38.1 Purpose

This example will test Motor Control PWM module in LPC17xx.

Tested function on MCPWM could be: 3-phase AC mode, 3-phase DC mode, capture on Motor Control Feed back input pin.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display information.

38.2 Hardware configuration

Please see abstract.txt file for more details.

38.3 Software configuration

Required files

mcpwm_simple.c

Running mode

Please see abstract.txt file for more details.

38.4 Procedure

Please see abstract.txt file for more details.

39. PWM – Dual Edge example

39.1 Purpose

This is a simple example about PWM function on LPC17xx.

This program illustrates the PWM signal on 3 Channels in both edge mode and single mode.

Peripheral clock for PWM: PWM_PCLK = CCLK / 4 = 72MHz/4 = 18MHz and there is no prescale for PWM. The PWM timer/counter clock is at 18MHz. The base rate is set to 100.

The base PWM frequency is at 18MHz/100 = 180 KHz.

Each PWM channel will be configured as follows:

- Channel 2: Double Edge
- Channel 4: Double Edge
- Channel 5: Single Edge

The Match register values are as follows:

- MR0 = 100 (PWM rate)
- MR1 = 41, MR2 = 78 (PWM2 output)
- MR3 = 53, MR4 = 27 (PWM4 output)
- MR5 = 65 (PWM5 output)

PWM Duty on each PWM channel:

- Channel 2: Set by match 1, Reset by match 2.
- Channel 4: Set by match 3, Reset by match 4.
- Channel 5: Set by match 0, Reset by match 5.

Use an oscilloscope to observe the PWM signals.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this conversion.

39.2 Hardware configuration

Please see abstract.txt file for more details.

39.3 Software configuration

Required files

pwm_dual_edge.c

Running mode

Please see abstract.txt file for more details.

39.4 Procedure

Please see abstract.txt file for more details.

40. PWM – Match Interrupt

40.1 Purpose

This is a simple example about PWM function on LPC17xx. This program illustrates the PWM signal on 6 Channels in single edge mode.

Peripheral clock for PWM: PWM_PCLK = CCLK / 4 = 72MHz/4 = 18MHz and there is no prescale for PWM. The PWM timer/counter clock is at 18MHz. The base rate is set to 256.

The base PWM frequency is at 18MHz/256 = 70.312 KHz (Period = ~14.22 microsecond)

Each PWM channel (1 to 6) will be configured as follows:

+ PWM1.1 = (10/256) (period = 0.56 microsecond)
+ PWM1.2 = (20/256) (period = 1.11 microsecond)
+ PWM1.3 = (30/256) (period = 1.67 microsecond)
+ PWM1.4 = (40/256) (period = 2.22 microsecond)
+ PWM1.5 = (50/256) (period = 2.78 microsecond)
+ PWM1.6 = (60/256) (period = 3.33 microsecond)

Use an oscilloscope to observe the PWM signals.

Here, PWM1.1 value is not stable; it will increase by the time from 0 to 256 period and restart. Match interrupt for channel 0 is set, when the PWM timer reaches 256 (value of channel 0 match), an interrupt for matching will generate and update the value of PWM1.1, this value will be updated every 4096 match interrupts or:

Period * 4096 = 14.22 * 4096 = 58,245 (microsecond)

And this value will be reset to 0 after:

Period * 4096 * 256 = 14,910,750.72 (microsecond) = ~15 (second)

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this conversion.

40.2 Hardware configuration

Please see abstract.txt file for more details.

40.3 Software configuration

Required files

pwm_match_int.c

Running mode

Please see abstract.txt file for more details.

40.4 Procedure

Please see abstract.txt file for more details.

41. PWM – Single Edge example

41.1 Purpose

This is a simple example about PWM function on LPC17xx.

This program illustrates the PWM signal on 6 Channels in single edge mode

Peripheral clock for PWM: PWM_PCLK = CCLK / 4 = 72MHz/4 = 18MHz and there is no prescale for PWM. The PWM timer/counter clock is at 18MHz. The base rate is set to 256.

The base PWM frequency is at 18MHz/256 = 70.312 KHz (Period = ~14.22 microsecond)

Each PWM channel (1 to 6) will be configured as follows:

+ PWM1.1 = (10/256) (period = 0.56 microsecond)
+ PWM1.2 = (20/256) (period = 1.11 microsecond)
+ PWM1.3 = (30/256) (period = 1.67 microsecond)
+ PWM1.4 = (40/256) (period = 2.22 microsecond)
+ PWM1.5 = (50/256) (period = 2.78 microsecond)
+ PWM1.6 = (60/256) (period = 3.33 microsecond)

Use an oscilloscope to observe the PWM signals

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display this conversion.

41.2 Hardware configuration

Please see abstract.txt file for more details.

41.3 Software configuration

Required files

pwm_single_edge.c

Running mode

Please see abstract.txt file for more details.

41.4 Procedure

Please see abstract.txt file for more details.

42. QEI example

42.1 Purpose

This example will test the Quadrature Encoder Interface module in LPC17xx with velocity calculation (RPM) to display via UART0.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display information.

42.2 Hardware configuration

Please see abstract.txt file for more details.

42.3 Software configuration

Required files

qei_test_velo.c

Running mode

Please see abstract.txt file for more details.

42.4 Procedure

Burning code in the first board, use it to generate signal supplying for QEI peripheral by using timer match interrupt ouput.

After load code in the second board, connect:

- P1.19 (board1) to P0.20 (board 2)
- P1.21 (board1) to P0.23 (board 2)

Screen wil be displayed like this:

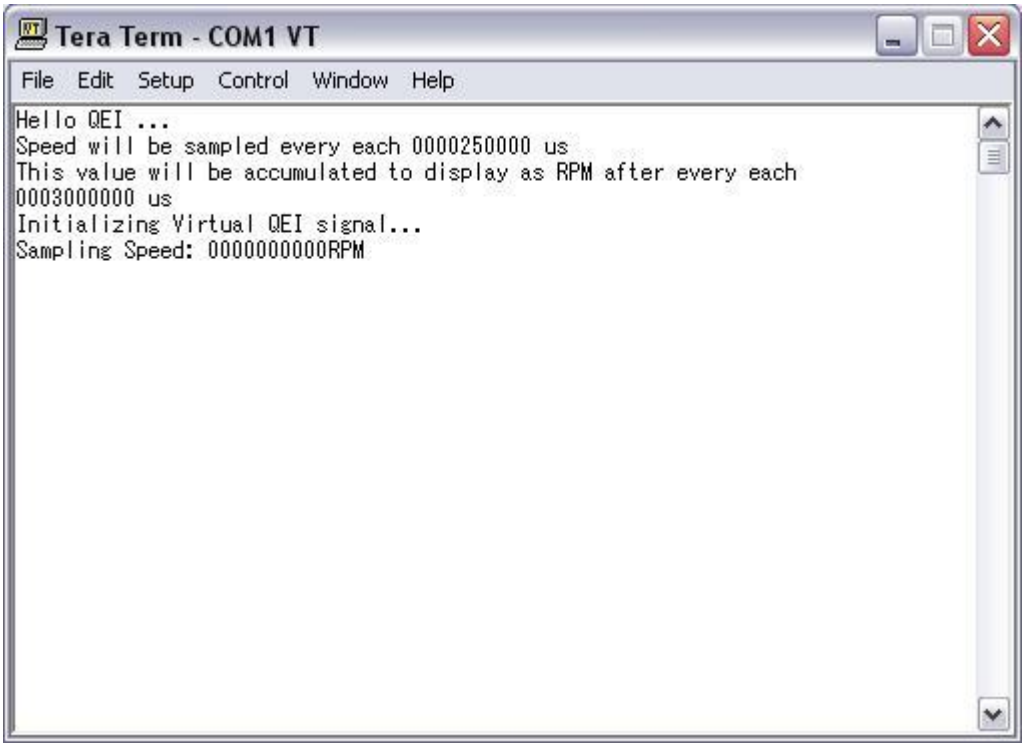

Fig 64. QEI init

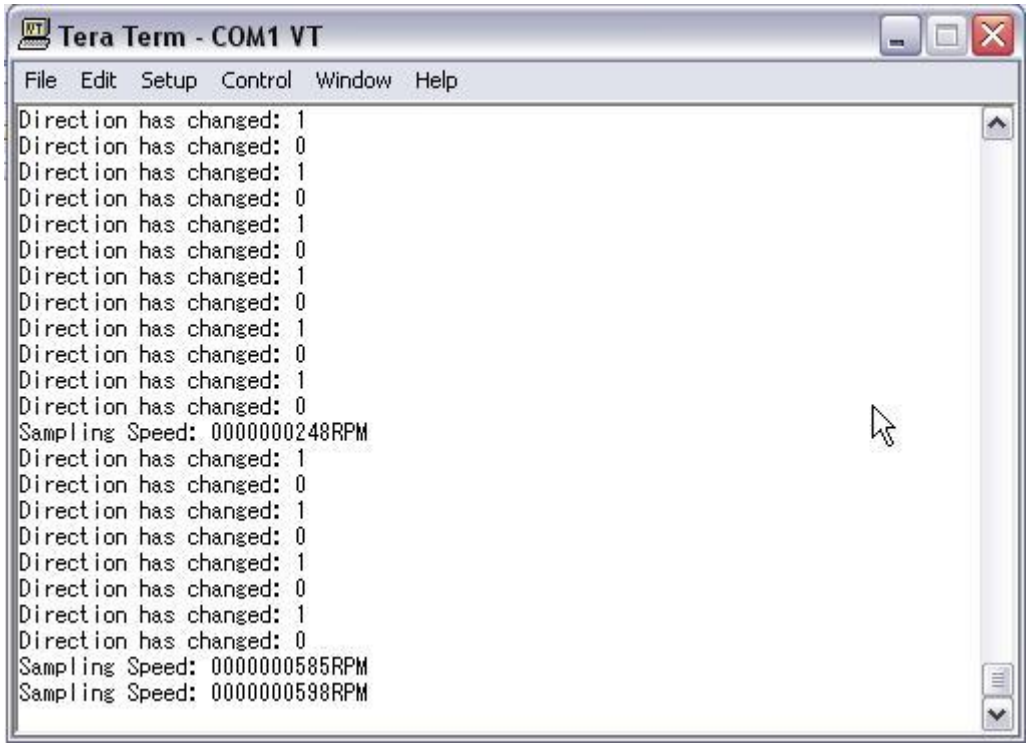

Fig 65. QEI process after connect pin

43. RIT polling example

43.1 Purpose

Use RIT as a timer to control 8 LED display.

43.2 Hardware configuration

Please see abstract.txt file for more details

43.3 Software configuration

Required files

rit_polling.c

Running mode

Default

43.4 Procedure

After restart, 8 LEDs will sequentially blink from LED 0 -> LED 7 and reverse.

Welcome screen and blinking LED processing will be displayed like this:

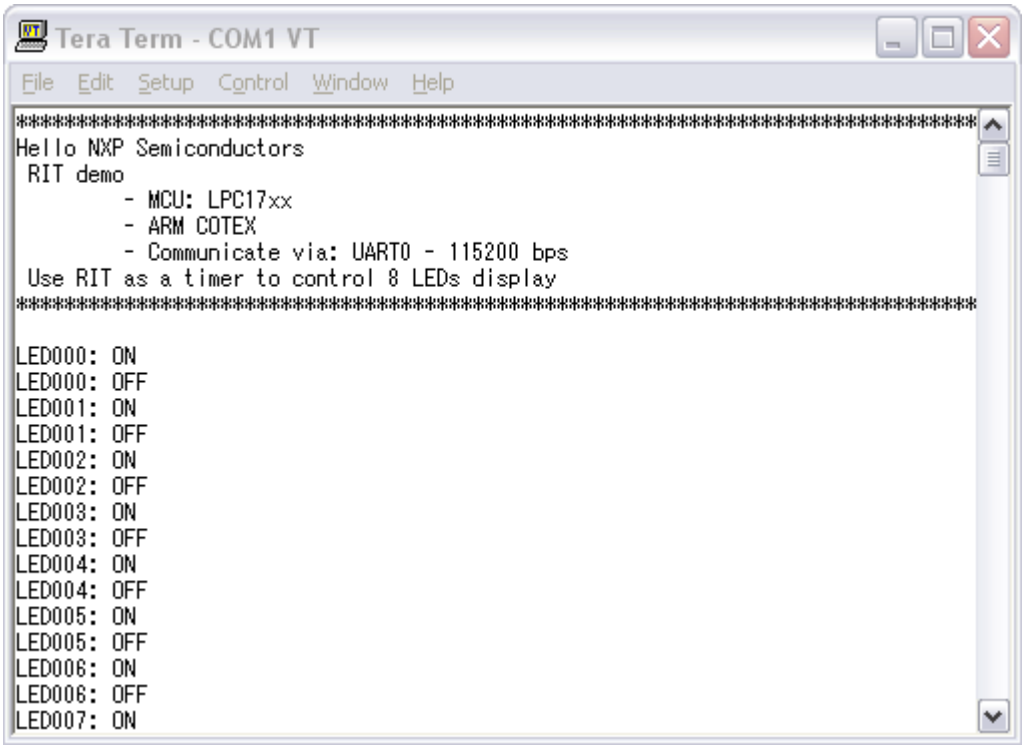

Fig 66. RIT demo

44. RIT interrupt example

44.1 Purpose

This is a simple RIT test: use RIT as a timer to generate interrupt to drive LED

44.2 Hardware configuration

Please see abstract.txt file for more information

44.3 Software configuration

Required files

rit_interrupt.c

Running mode

Please see abstract.txt file for more details.

44.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

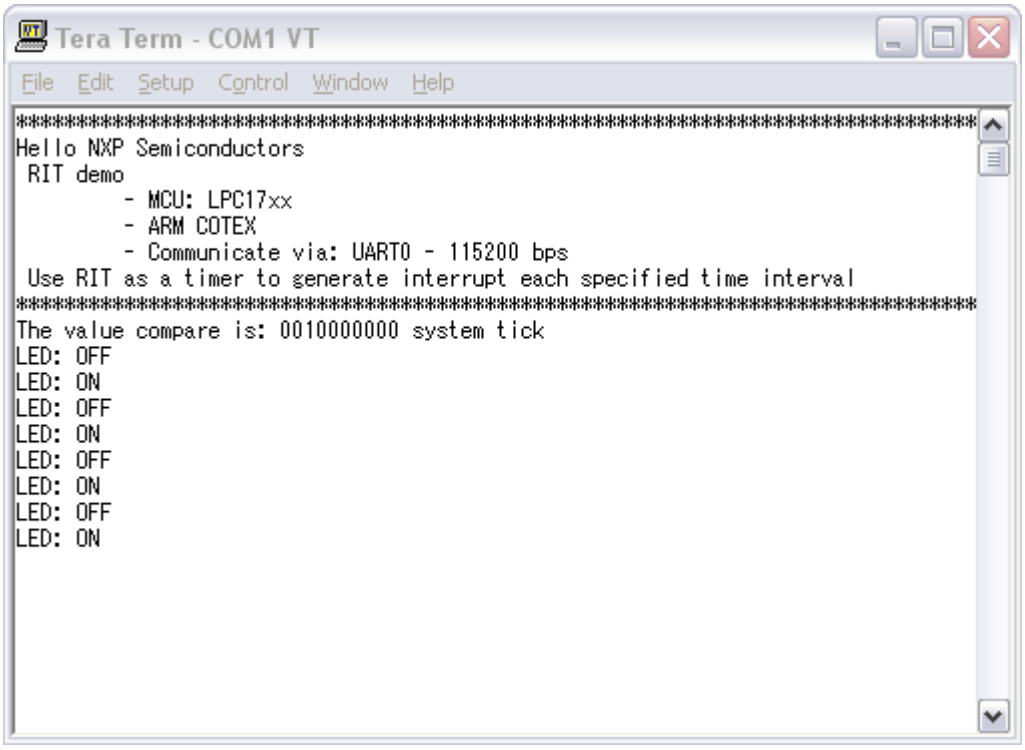

Fig 67. RIT screen

45. RTC example

45.1 Purpose

This is a simple RTC example, using RTC to generate an interrupt in Second Counter Increment Interrupt (1s) and Alarm interrupt at 10s.

Use UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control to display information.

45.2 Hardware configuration

Please see abstract.txt file for more details.

45.3 Software configuration

Required files

rtc_alarm_cntincr_int.c

Running mode

Please see abstract.txt file for more details.

45.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

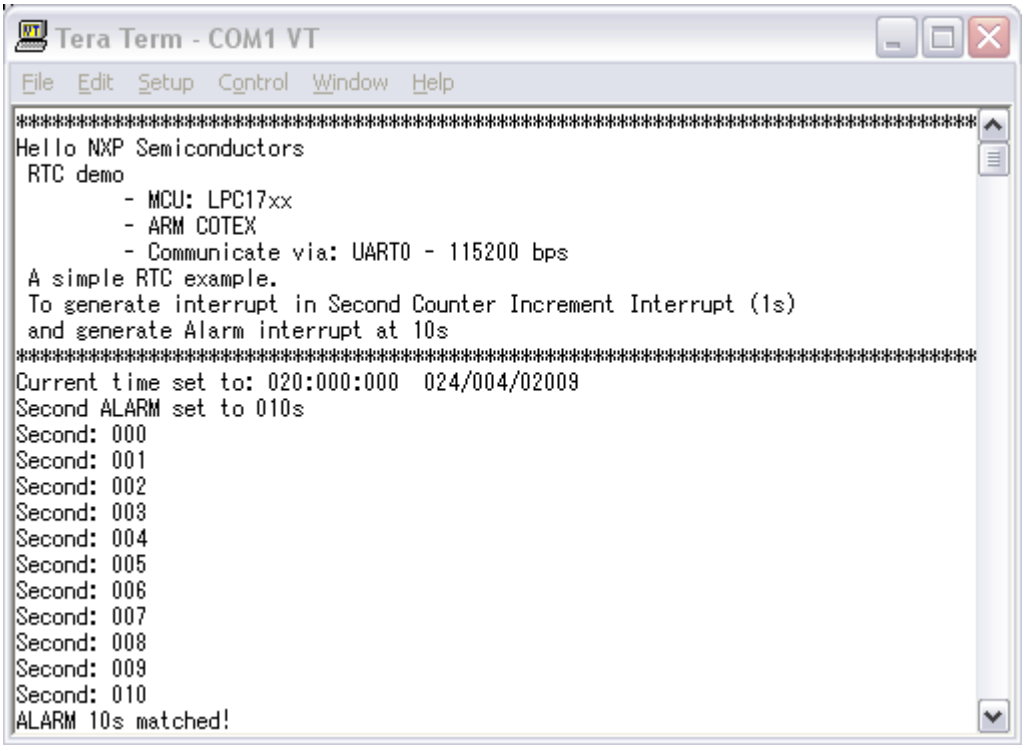

Fig 68. RTC screen

46. SPI Loop back example

46.1 Purpose

This is an example of SPI using interrupt mode to test the SPI driver.

This example uses SPI in loop-back mode to transfer a number of data byte.

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

46.2 Hardware configuration

Please see abstract.txt file for more details.

46.3 Software configuration

Required files

spi_loopback_test.c

Running mode

Please see abstract.txt file for more details.

46.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

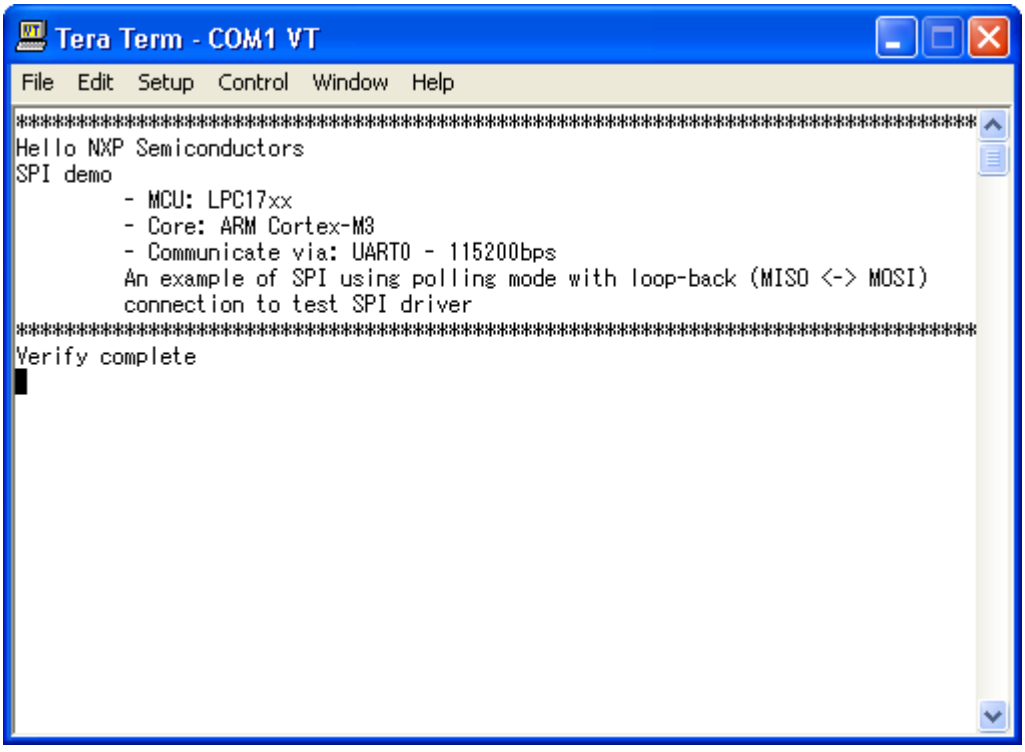

Fig 69. SPI Loopback demo

47. SPI master example

47.1 Purpose

This is an example of SPI using interrupt mode to test the SPI driver.

This example uses SPI in master mode to communicate with SPI slave device.

The master and slave transfer together a number of data byte.

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

47.2 Hardware configuration

Please see abstract.txt file for more details.

47.3 Software configuration

Required files

spi_master.c

Running mode

Please see abstract.txt file for more details.

47.4 Procedure

These steps should be in the following sequence:

- The slave must start first to be ready to receive data from master.
- Press '1' to transmit data from master to slave.

Please see abstract.txt file for more details.

The screen will be like this:

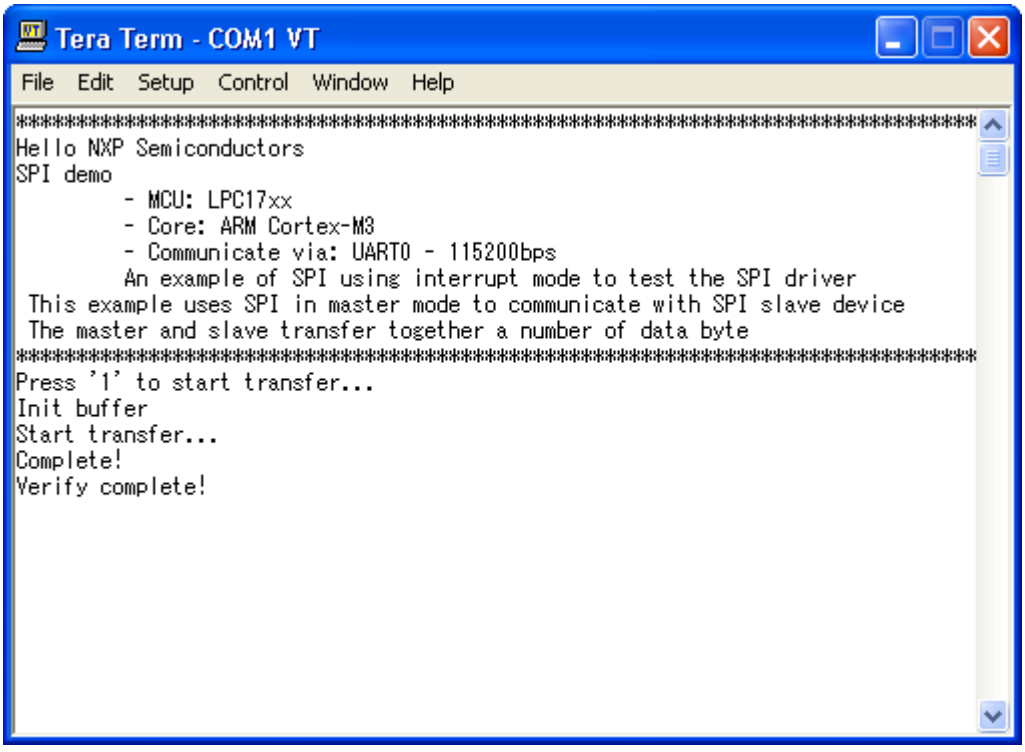

Fig 70. SPI master demo

48. SPI slave example

48.1 Purpose

This is an example of SPI using polling mode to test the SPI driver.

This example uses SPI in slave mode to communicate with an SPI master device.

The master and slave transfer together a number of data byte.

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

48.2 Hardware configuration

Please see abstract.txt file for more details.

48.3 Software configuration

Required files

spi_slave.c

Running mode

Please see abstract.txt file for more details.

48.4 Procedure

After power-up, slave is ready to communicate with master.

Please see abstract.txt file for more details.

The screen will be like this:

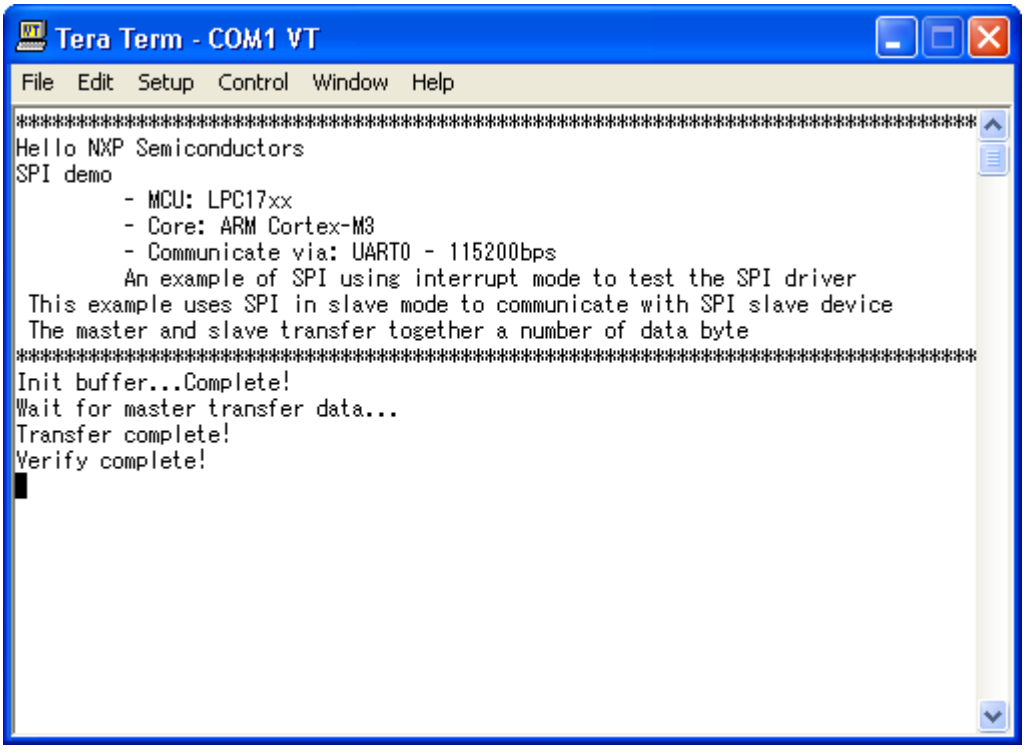

Fig 71. SPI slave demo

49. SPI – SC16IS750 interrupt example

49.1 Purpose

This is an example of SPI using interrupt mode to test the SPI driver.

Using SPI at mode SPI master/8bit on LPC1768 to communicate with SC16IS750/760 Demo Board.

49.2 Hardware configuration

Please see abstract.txt file for more details.

49.3 Software configuration

Required files

spi_interrupt_test.c

Running mode

Please see abstract.txt file for more details.

49.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

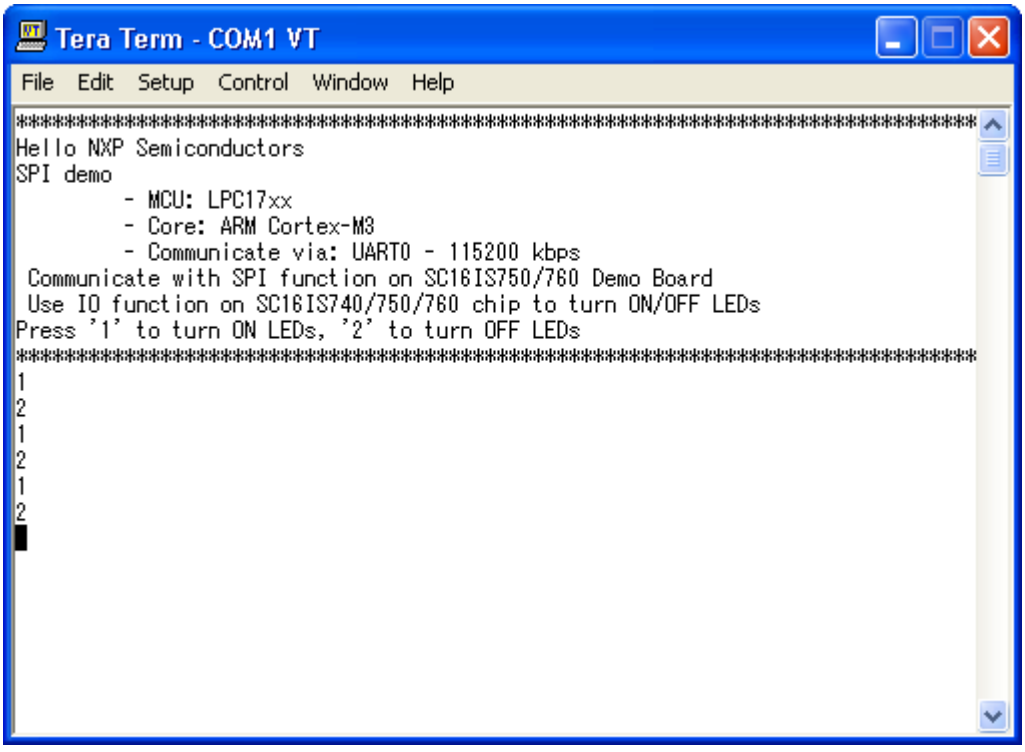

Fig 72. SPI – SC16IS750 interrupt demo

50. SPI – SC16IS750 polling example

50.1 Purpose

This is an example of SPI using polling mode to test the SPI driver.

Using SPI at mode SPI master/8bit on LPC1768 to communicate with SC16IS750/760 Demo Board.

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

50.2 Hardware configuration

Please see abstract.txt file for more details.

50.3 Software configuration

Required files

spi_polling_test.c

Running mode

Please see abstract.txt file for more details.

50.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

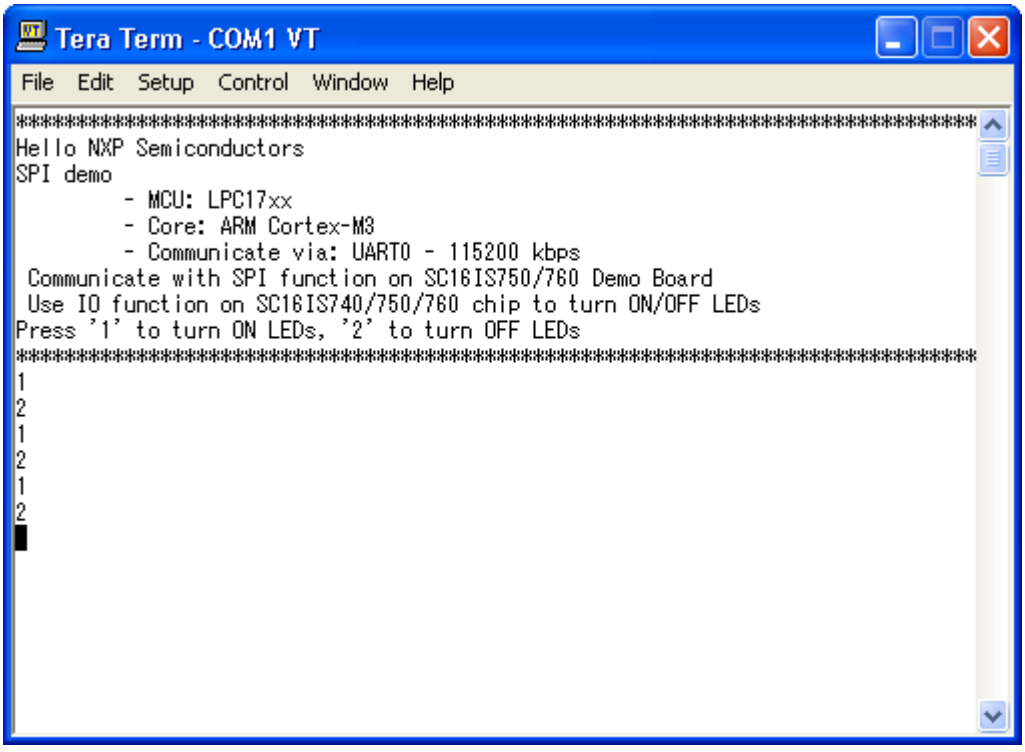

Fig 73. SPI – SC16IS750 polling demo

51. SSP – Master example

51.1 Purpose

This is an example of SSP using Polling mode to test the SSP driver.

This example uses SSP in SPI frame as master to communicate with an SSP slave device.

The master and slave transfer together a number of data byte.

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

51.2 Hardware configuration

Please see abstract.txt file for more details.

51.3 Software configuration

Required files

ssp_master.c

Running mode

Please see abstract.txt file for more details.

51.4 Procedure

These steps must be in the following sequence:

- Slave must start first to be ready to communicate with master.
- On master, press '1' to start transfer data.

Please see abstract.txt file for more details.

The screen will be like this:

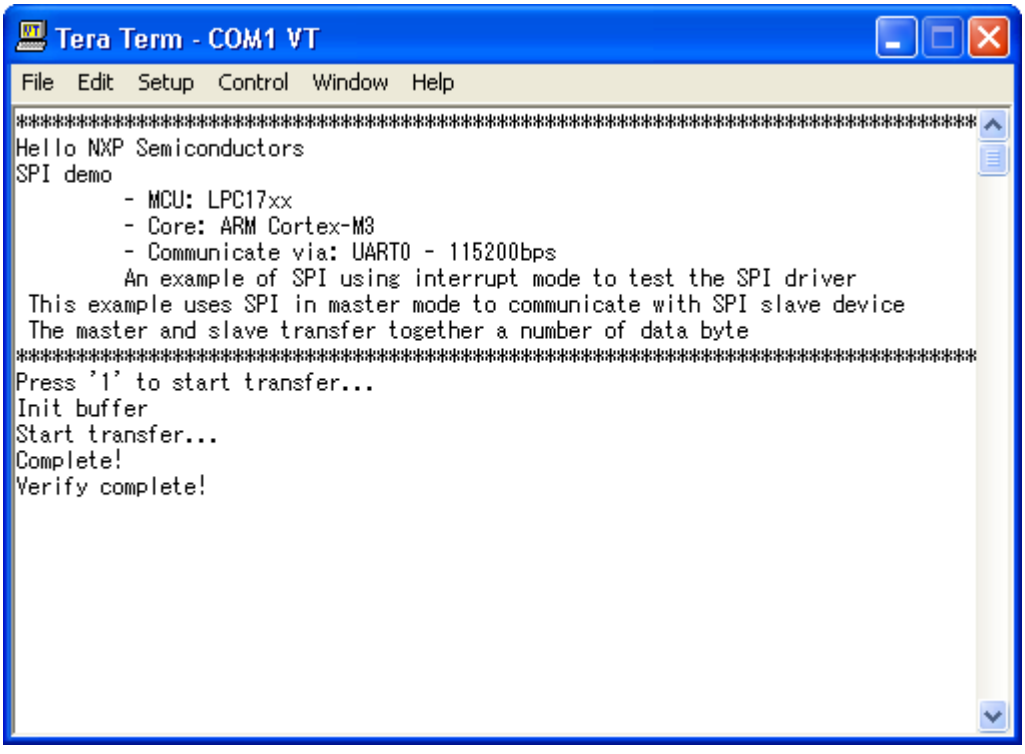

Fig 74. SSP – Master demo

52. SSP – Slave example

52.1 Purpose

This is an example of SSP using interrupt mode to test the SSP driver.

This example uses SSP in SPI frame as slave mode to communicate with SSP master device.

The master and slave transfer together a number of data byte.

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

52.2 Hardware configuration

Please see abstract.txt file for more details.

52.3 Software configuration

Required files

ssp_slave.c

Running mode

Please see abstract.txt file for more details.

52.4 Procedure

After power-up, the slave is ready to communicate with master.

Please see abstract.txt file for more details.

The screen will be like this:

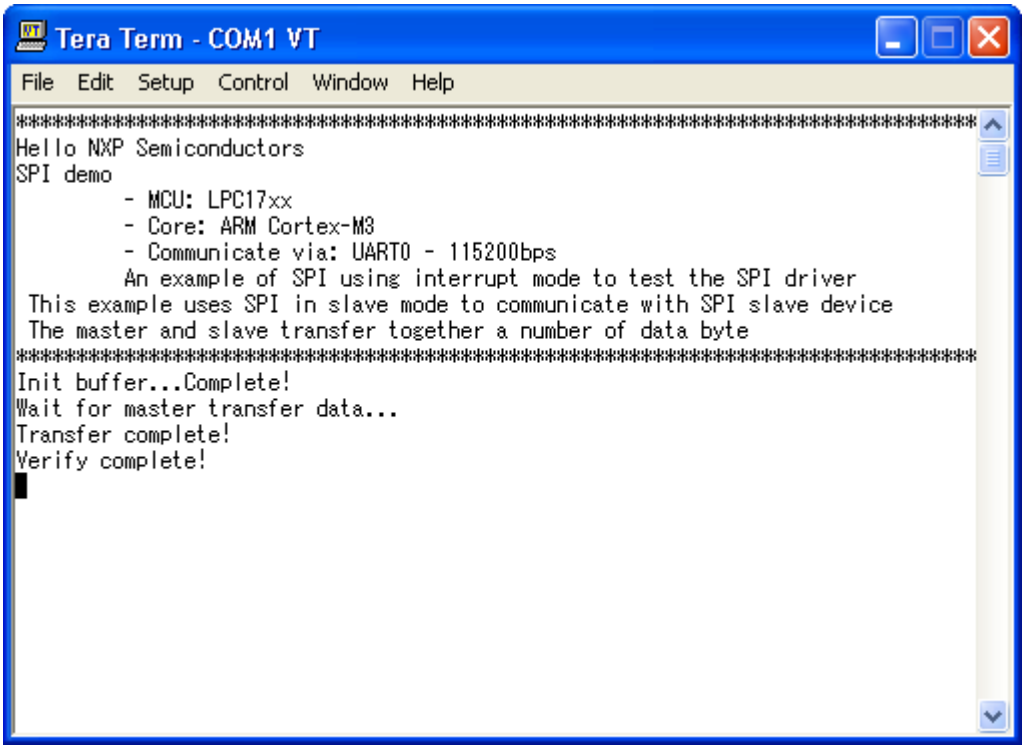

Fig 75. SSP – Slave demo

53. SSP- DMA example

53.1 Purpose

This example uses SSP function in MASTER mode with Loop-back mode (MOSI <-> MISO).

Transfer a number of data byte (in DMA mode for both Tx and Rx channel)

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

53.2 Hardware configuration

Please see abstract.txt file for more details.

53.3 Software configuration

Required files

ssp_dma.c

Running mode

Please see abstract.txt file for more details.

53.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

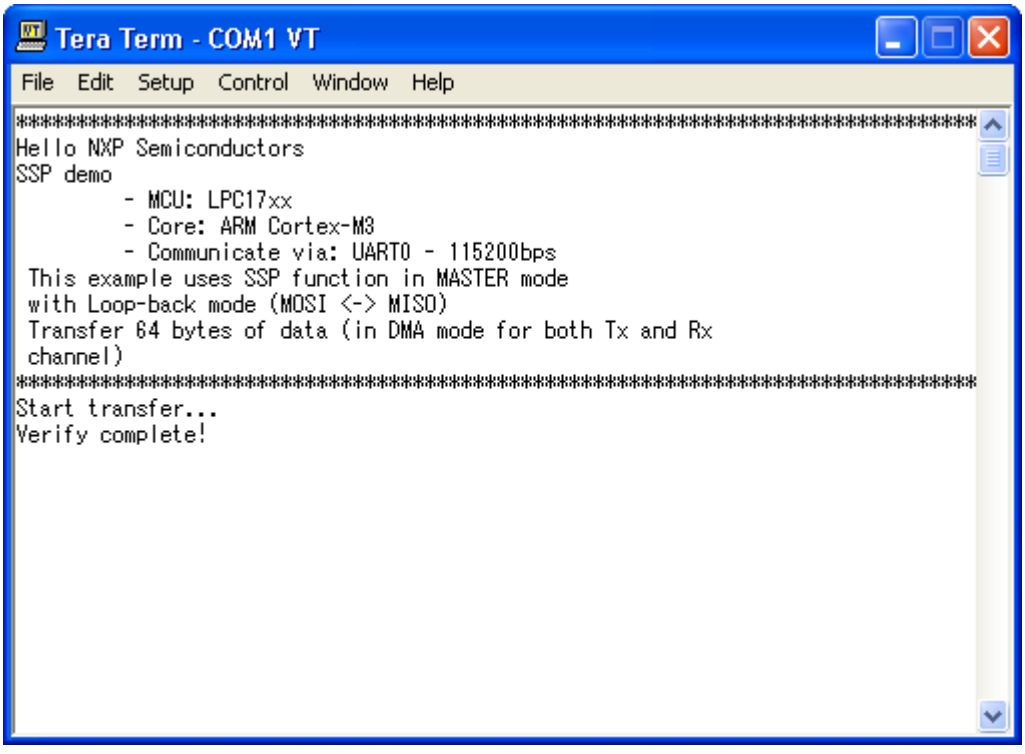

Fig 76. SSP – DMA demo

54. SSP – SC16IS750 polling example

54.1 Purpose

This is an example of SSP using polling mode to test the SSP driver.

Using SSP in SPI frame mode as master/8bit on LPC1768 to communicate with SC16IS750/760 Demo Board

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

54.2 Hardware configuration

Please see abstract.txt file for more details.

54.3 Software configuration

Required files

sc16is750_int.c

Running mode

Please see abstract.txt file for more details.

54.4 Procedure

- Press ‘1’ to turn ON LEDs, ‘2’ to turn OFF LEDs.

Please see abstract.txt file for more details.

55. SSP – SC16IS750 interrupt example

55.1 Purpose

This is an example of SSP using interrupt mode to test the SSP driver.

Using SSP in SPI frame mode as master/8bit on LPC1768 to communicate with SC16IS750/760 Demo Board

UART0 – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control.

55.2 Hardware configuration

Please see abstract.txt file for more details.

55.3 Software configuration

Required files

sc16is750_int.c

Running mode

Please see abstract.txt file for more details.

55.4 Procedure

- Press ‘1’ to turn ON LEDs, ‘2’ to turn OFF LEDs.

Please see abstract.txt file for more details.

56. TIMER – Delay example

56.1 Purpose

This is a simple example for Timer in polling mode: Wait 1 second to turn ON/OFF LED in sequence.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control

56.2 Hardware configuration

Please see abstract.txt file for more details.

56.3 Software configuration

Required files

timer_delay_test.c

Running mode

Please see abstract.txt file for more details.

56.4 Procedure

Please see abstract.txt file for more details.

57. TIMER – Interrupt example

57.1 Purpose

This is a simple example for Timer interrupt mode: turn on/off P2.0 and toggle MAT0.0 (pinP0.28) at frequency 10Hz.

Use UART0 with this configuration – 115200bps – 8 data bit – No parity – 1 stop bit – No flow control

57.2 Hardware configuration

Please see abstract.txt file for more details.

57.3 Software configuration

Required files

timer_interrupt_test.c

Running mode

Please see abstract.txt file for more details.

57.4 Procedure

Please see abstract.txt file for more details.

58. TIMER – Capture example

58.1 Purpose

This example used to test Capture Timer function.

We use Timer 0 to take a snapshot of the timer value when an input on CAP0.0(Pin 0.26) transistions.

58.2 Hardware configuration

Please see abstract.txt file for more details

58.3 Software configuration

Required files

timer_capture.c

Running mode

Please see abstract.txt for more details.

58.4 Procedure

After reset, the welcome screen appears like this:

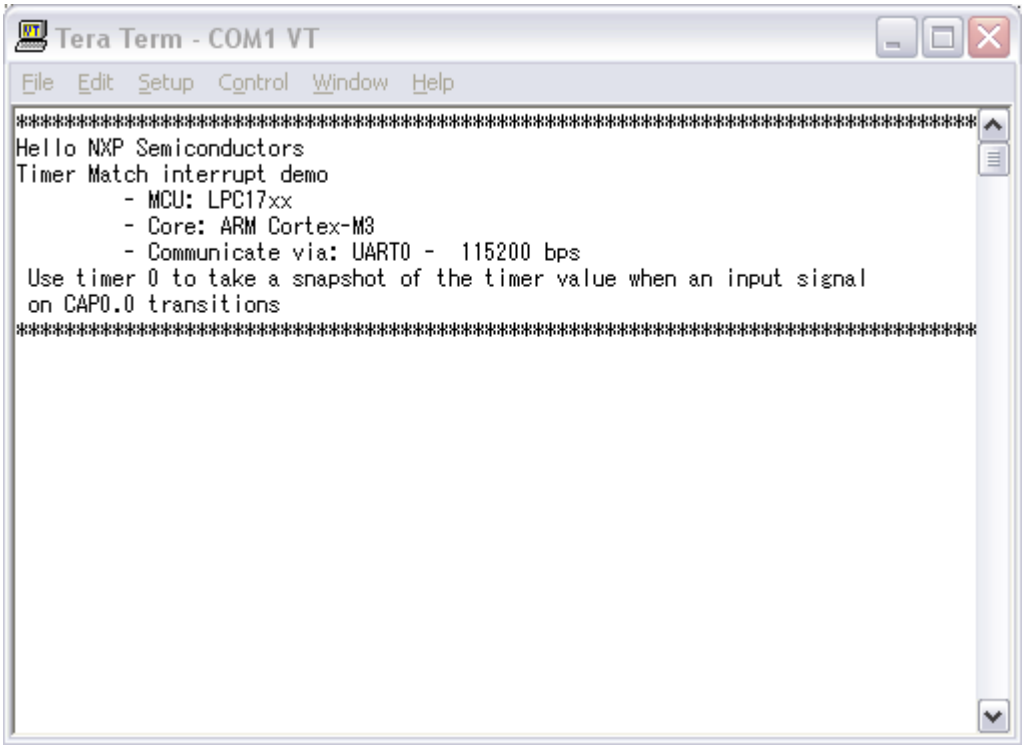

Fig 77. SSP – Welcome screen

Change connecting CAP0.0 with GND and VCC continuously, the time (s) will be captured like this:

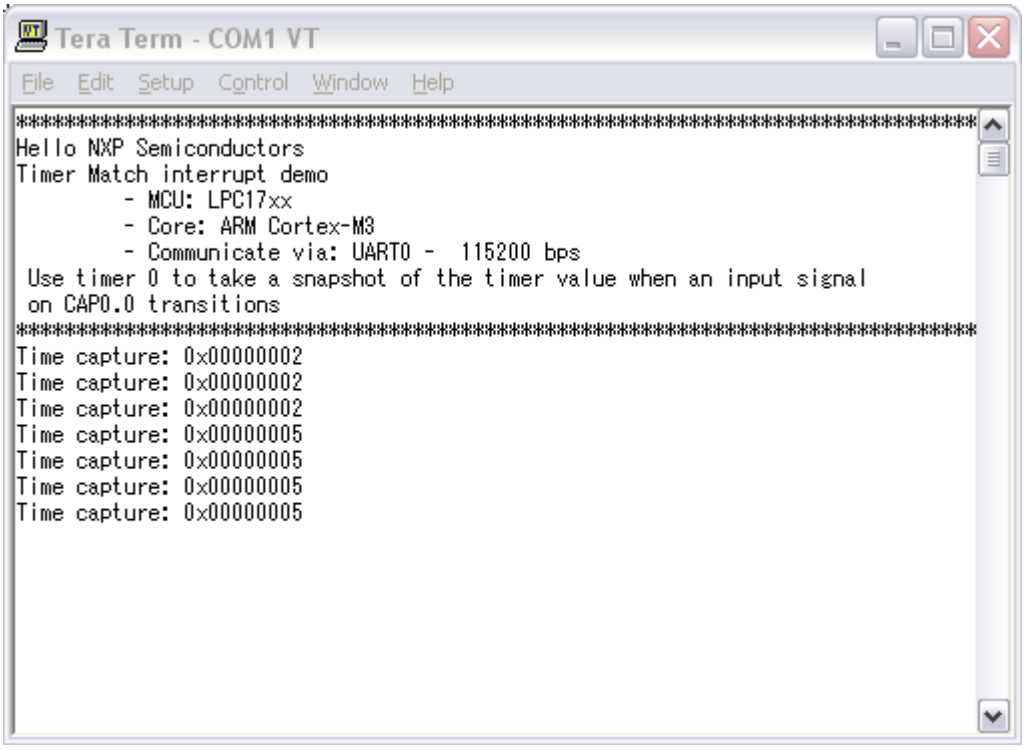

Fig 78. SSP – Time captured

59. USB device – HID example

59.1 Purpose

The HID project is a demo program for the LPC1768 using Keil MCB1700 board.

Note: This example comes from the LPC17xx code bundle with a few modifications to be compiled with GNU toolchain.

59.2 Hardware configuration

Please see abstract.txt file for more details.

59.3 Software configuration

Required files

demo.c

Running mode

Please see abstract.txt file for more details.

59.4 Procedure

Please see abstract.txt file for more details.

60. USB device – Mass storage example

60.1 Purpose

The Memory project is a demo program for the LPC1768 using Keil MCB1700 board.

Note: This example comes from the LPC17xx code bundle with a few modifications to be compiled with GNU toolchain.

60.2 Hardware configuration

Please see abstract.txt file for more details.

60.3 Software configuration

Required files

memory.c

Running mode

Please see abstract.txt file for more details.

60.4 Procedure

Please see abstract.txt file for more details.

61. WDT - Interrupt example

61.1 Purpose

This is a simple example for Watchdog timer application in interrupt mode

61.2 Hardware configuration

Please see abstract.txt file for more details.

61.3 Software configuration

Required files

wdt_interrupt_test.c

Running mode

Please see abstract.txt file for more details.

61.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

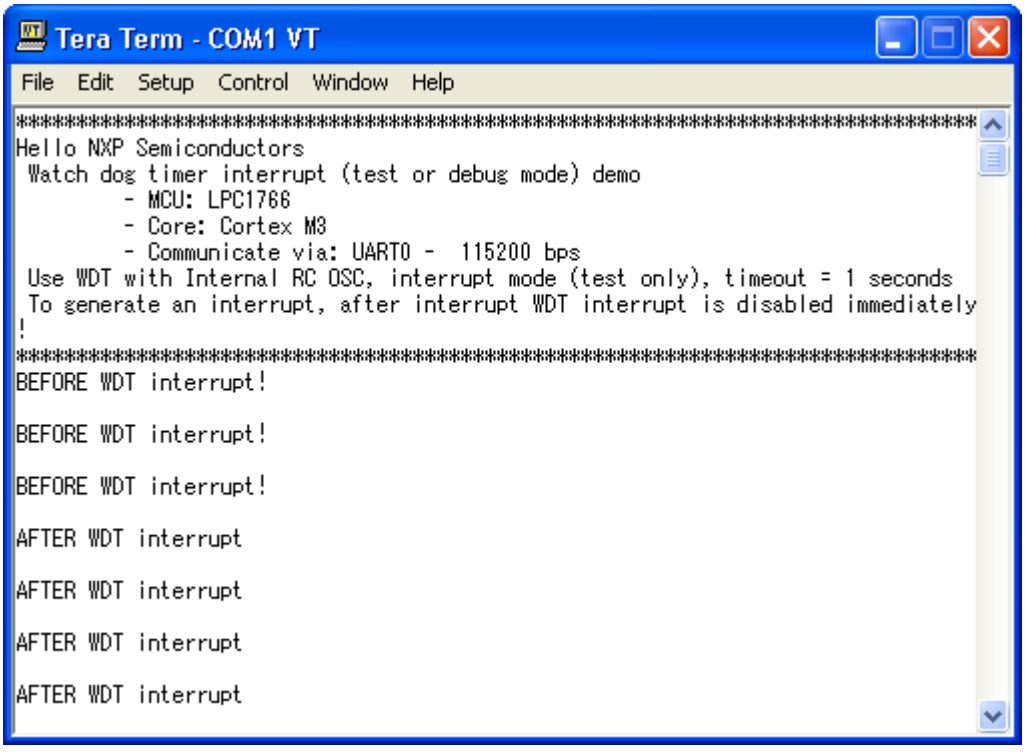

Fig 79. WDT interrupt demo

62. WDT - Reset example

62.1 Purpose

This is a simple example for Watchdog timer application in reset mode

62.2 Hardware configuration

Please see abstract.txt file for more details.

62.3 Software configuration

Required files

wdt_reset_test.c

Running mode

Please see abstract.txt file for more details.

62.4 Procedure

Please see abstract.txt file for more details.

The screen will be like this:

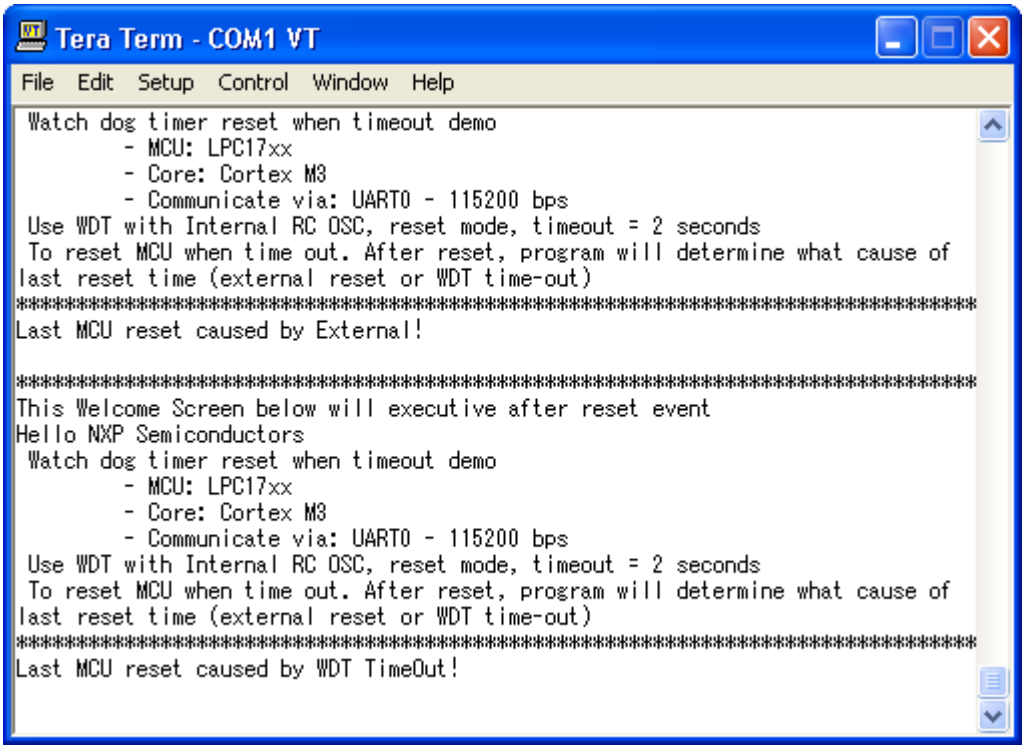

Fig 80. WDT reset demo

63. Legal information

63.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

63.2 Disclaimers

General — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is for the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

63.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

64. Contents

1. Introduction 3
2. Additional condition 3
2.1 Serial display 3
2.2 Additional files requirement 6
2.3 Running mode 7
2.3.1 RAM mode 7
2.3.2 ROM (FLASH) mode: 7
3. UART polling example 7
3.1 Purpose 7
3.2 Hardware configuration 7
3.3 Software configuration 7
3.4 Procedure 7
4. UART auto baud rate example 8
4.1 Purpose 8
4.2 Hardware configuration 8
4.3 Software configuration 8
4.4 Procedure 8
5. UART DMA Example 9
5.1 Purpose 9
5.2 Hardware configuration 9
5.3 Software configuration 9
5.4 Procedure 9
6. UART Interrupt example 10
6.1 Purpose 10
6.2 Hardware configuration 10
6.3 Software configuration 10
6.4 Procedure 10
7. UART Full modem example 11
7.1 Purpose 11
7.2 Hardware configuration 11
7.3 Software configuration 11
7.4 Procedure 11
8. UART RS485 master example 12
8.1 Purpose 12
8.2 Hardware configuration 12
8.3 Software configuration 12
8.4 Procedure 12
9. UART RS485 slave example 13
9.1 Purpose 13
9.2 Hardware configuration 13
9.3 Software configuration 13
9.4 Procedure 13
10. ADC polling mode example 14
10.1 Purpose 14
10.2 Hardware configuration 14
10.3 Software configuration 14
10.4 Procedure 14
11. ADC Interrupt example 15
11.1 Purpose 15
11.2 Hardware configuration 15
11.3 Software configuration 15
11.4 Procedure 15
12. ADC DMA example 16
12.1 Purpose 16
12.2 Hardware configuration 16
12.3 Software configuration 16
12.4 Procedure 16
13. CAN test Bypass mode example 17
13.1 Purpose 17
13.2 Hardware configuration 17
13.3 Software configuration 17
13.4 Procedure 17
14. CAN test Acceptance Filter mode example 19
14.1 Purpose 19
14.2 Hardware configuration 19
14.3 Software configuration 19
14.4 Procedure 19
15. CAN test setup AFLUT dynamically 22
15.1 Purpose 22
15.2 Hardware configuration 22
15.3 Sofware configuration 22
15.4 Procedure 22
16. CAN test in two board example 28
16.1 Purpose 28
16.2 Hardware configuration 28
16.3 Software configuration 28
16.4 Procedure 28
17. DAC test example 31
17.1 Purpose 31
17.2 Hardware configuration 31
17.3 Software configuration 31
17.4 Procedure 31
18. DAC DMA example 32
18.1 Purpose 32
18.2 Hardware configuration 32
18.3 Software configuration 32
18.4 Procedure 32
19. EMAC – Raw example 33

continued >>

19.1 Purpose....33
19.2 Hardware configuration....33
19.3 Software configuration....33
19.4 Procedure....33
20. EMAC – EasyWeb example....34
20.1 Purpose....34
20.2 Hardware configuration....35
20.3 Software configuration....35
20.4 Procedure....35
21. EMAC – uIP example....37
21.1 Purpose....37
21.2 Hardware configuration....37
21.3 Software configuration....37
21.4 Procedure....38
22. GPDMA example....38
22.1 Purpose....38
22.2 Hardware configuration....38
22.3 Software configuration....38
22.4 Procedure....39
23. GPIO External Power down example....39
23.1 Purpose....39
23.2 Hardware configuration....39
23.3 Software configuration....39
23.4 Procedure....40
24. GPIO interrupt example....40
24.1 Purpose....40
24.2 Hardware configuration....40
24.3 Software configuration....40
24.4 Procedure....40
25. GPIO Blinky....40
25.1 Purpose....40
25.2 Hardware configuration....40
25.3 Software configuration....40
25.4 Procedure....40
26. GPIO – Port LCD....40
26.1 Purpose....40
26.2 Hardware configuration....41
26.3 Software configuration....41
26.4 Procedure....41
27. I2C master example....41
27.1 Purpose....41
27.2 Hardware configuration....41
27.3 Software configuration....41
27.4 Procedure....41
28. I2C slave example....42
28.1 Purpose....42
28.2 Hardware configuration....42
28.3 Software configuration....42
28.4 Procedure....43
29. I2C Master-Slave interrupt example....43
29.1 Purpose....43
29.2 Hardware configuration....43
29.3 Software configuration....43
29.4 Procedure....44
30. I2C - PCA8581 polling example....44
30.1 Purpose....44
30.2 Hardware configuration....44
30.3 Software configuration....44
30.4 Procedure....45
31. I2C - SC16IS750 polling test example....45
31.1 Purpose....45
31.2 Hardware configuration....45
31.3 Software configuration....45
31.4 Procedure....46
32. I2C SC16IS750 interrupt example....46
32.1 Purpose....46
32.2 Hardware configuration....46
32.3 Software configuration....46
32.4 Procedure....47
33. I2S self-test in polling mode example....47
33.1 Purpose....47
33.2 Hardware configuration....47
33.3 Software configuration....47
33.4 Procedure....48
34. I2S self-test in interrupt example....50
34.1 Purpose....50
34.2 Hardware configuration....50
34.3 Software configuration....50
34.4 Procedure....50
35. I2S self-test DMA mode example....52
35.1 Purpose....52
35.2 Hardware configuration....52
35.3 Software configuration....52
35.4 Procedure....52
36. I2S test 4-wire mode....54
36.1 Purpose....54
36.2 Hardware configuration....54
36.3 Software configuration....54
36.4 Procedure....54
37. I2S Master-Slave mode example....56
37.1 Purpose....56
37.2 Hardware configuration....57
37.3 Software configuration....57
37.4 Procedure....57
38. MCPWM Simple....59
38.1 Purpose....59

continued >>

38.2 Hardware configuration 59
38.3 Software configuration 59
38.4 Procedure....59
39. PWM – Dual Edge example....60
39.1 Purpose....60
39.2 Hardware configuration 60
39.3 Software configuration 60
39.4 Procedure....60
40. PWM – Match Interrupt 61
40.1 Purpose....61
40.2 Hardware configuration 61
40.3 Software configuration 61
40.4 Procedure....61
41. PWM – Single Edge example....62
41.1 Purpose....62
41.2 Hardware configuration 62
41.3 Software configuration 62
41.4 Procedure....62
42. QEI example....62
42.1 Purpose....62
42.2 Hardware configuration 62
42.3 Software configuration 63
42.4 Procedure....63
43. RIT polling example 64
43.1 Purpose....64
43.2 Hardware configuration 64
43.3 Software configuration 64
43.4 Procedure....64
44. RIT interrupt example 65
44.1 Purpose....65
44.2 Hardware configuration 65
44.3 Software configuration 65
44.4 Procedure....65
45. RTC example....66
45.1 Purpose....66
45.2 Hardware configuration 66
45.3 Software configuration 66
45.4 Procedure....66
46. SPI Loop back example 67
46.1 Purpose....67
46.2 Hardware configuration 67
46.3 Software configuration 67
46.4 Procedure....68
47. SPI master example 68
47.1 Purpose....68
47.2 Hardware configuration 68
47.3 Software configuration 68
47.4 Procedure....69
48. SPI slave example 69
48.1 Purpose....69
48.2 Hardware configuration 69
48.3 Software configuration....70
48.4 Procedure....70
49. SPI – SC16IS750 interrupt example 70
49.1 Purpose....70
49.2 Hardware configuration 70
49.3 Software configuration....71
49.4 Procedure....71
50. SPI – SC16IS750 polling example 71
50.1 Purpose....71
50.2 Hardware configuration 71
50.3 Software configuration....72
50.4 Procedure....72
51. SSP – Master example....72
51.1 Purpose....72
51.2 Hardware configuration 73
51.3 Software configuration....73
51.4 Procedure....73
52. SSP – Slave example....73
52.1 Purpose....73
52.2 Hardware configuration 74
52.3 Software configuration....74
52.4 Procedure....74
53. SSP- DMA example....75
53.1 Purpose....75
53.2 Hardware configuration 75
53.3 Software configuration....75
53.4 Procedure....75
54. SSP – SC16IS750 polling example....76
54.1 Purpose....76
54.2 Hardware configuration 76
54.3 Software configuration....76
54.4 Procedure....76
55. SSP – SC16IS750 interrupt example....76
55.1 Purpose....76
55.2 Hardware configuration 76
55.3 Software configuration....76
55.4 Procedure....76
56. TIMER – Delay example....77
56.1 Purpose....77
56.2 Hardware configuration 77
56.3 Software configuration....77
56.4 Procedure....77
57. TIMER – Interrupt example 77
57.1 Purpose....77
57.2 Hardware configuration 77

continued >>

57.3 Software configuration 77
57.4 Procedure....77
58. TIMER – Capture example 78
58.1 Purpose....78
58.2 Hardware configuration 78
58.3 Software configuration 78
58.4 Procedure....78
59. USB device – HID example 79
59.1 Purpose....79
59.2 Hardware configuration 79
59.3 Software configuration 79
59.4 Procedure....79
60. USB device – Mass storage example 80
60.1 Purpose....80
60.2 Hardware configuration 80
60.3 Software configuration 80
60.4 Procedure....80
61. WDT - Interrupt example 80
61.1 Purpose....80
61.2 Hardware configuration 80
61.3 Software configuration....80
61.4 Procedure....80
62. WDT - Reset example 81
62.1 Purpose....81
62.2 Hardware configuration 81
62.3 Software configuration....81
62.4 Procedure....81
63. Legal information 83
63.1 Definitions....83
63.2 Disclaimers....83
63.3 Trademarks 83
64. Contents 84

continued >>