AN12746

Shell Solution on NXP LPC55S1x Series

Rev. 0 — 20 February 2020

Application Note

Contents

1 Introduction

The LPC55S1x/LPC551x is an Arm Cortex®-M33-based microcontroller for embedded applications. These devices include up to 96 KB of on-chip SRAM, up to 256 KB on-chip flash, high-speed, and full-speed USB host and device interface with crystal-less operation for full-speed, one CAN-FD controller, five general-purpose timers, one SCTimer/PWM, one RTC/alarm timer, one 24-bit Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), eight flexible serial communication peripherals (each of which can be a USART, SPI, I2C, or I2S interface), one 16-bit 1.0 Msps ADC, temperature sensor. The Arm Cortex®- M33 provides a security foundation, offering isolation to protect valuable IP and data with TrustZone® technology.

In embedded systems application, a shell function is helpful to output log information and easy debug some standalone function API. Natural Tiny Shell(NT-Shell) is written by Shinichiro Nakamura, it is a C library for embedded systems, provides VT100 compatible terminal control feature, and needs only serial read/write functions for the porting.

This application note describes how to integrate NT-Shell files on the NXP LPC5500 with SDK and how to use the shell function. NT-Shell uses the USART0 to print information and get command from terminal. We have also added control the led toggle status command based on the basic NT-Shell demonstration.

The sample software is tested on LPC55S16-EVK evaluation board. Software is available for three IDE's/toolchains:

• MCUXpresso

• Keil µVision

• IAR EWARM

2 NT-Shell overview

2.1 Features

- Compatible with VT100
- Really simple
- Highly portable
  - Compatible with C89
  - No dependencies. (even libc !)
  - No dynamic memory allocation. (no need an operating system!)
- Small code foot print

— ROM: 10KB

— RAM: 1KB

2.2 License claim

NT-Shell's license is MIT.

The license is MIT.

Please see also http://opensource.org/licenses/mit-license.php.

vtparse, vtparse_table are in the public domain.

ntshell, ntopt, ntlibc, text_editor, text_history are in the MIT license.

You can also select TOPPERS license.

Refer links : https://www.cubeatsystems.com/ntshell/license.html

2.3 Source code download

User can download NT-Shell source code from below link:

https://www.cubeatsystems.com/ntshell/download.html

2.4 Architecture

NT-Shell have two part: **core** and **util**.

Core branch include 4 parts, Figure 1 shows the file structure:

- Top interface module(ntshell.c/.h)
- VT100 sequence controller (vtsend.c/.h, vtrecv.c/.h, vtparse_table.c/.h)
- Text controller (text_editor.c/.h, text_history.c/.h)
- C Runtime Library(ntlibc.c/.h)

Utility branch only contains ntopt.c/.h and ntstdio.c/.h

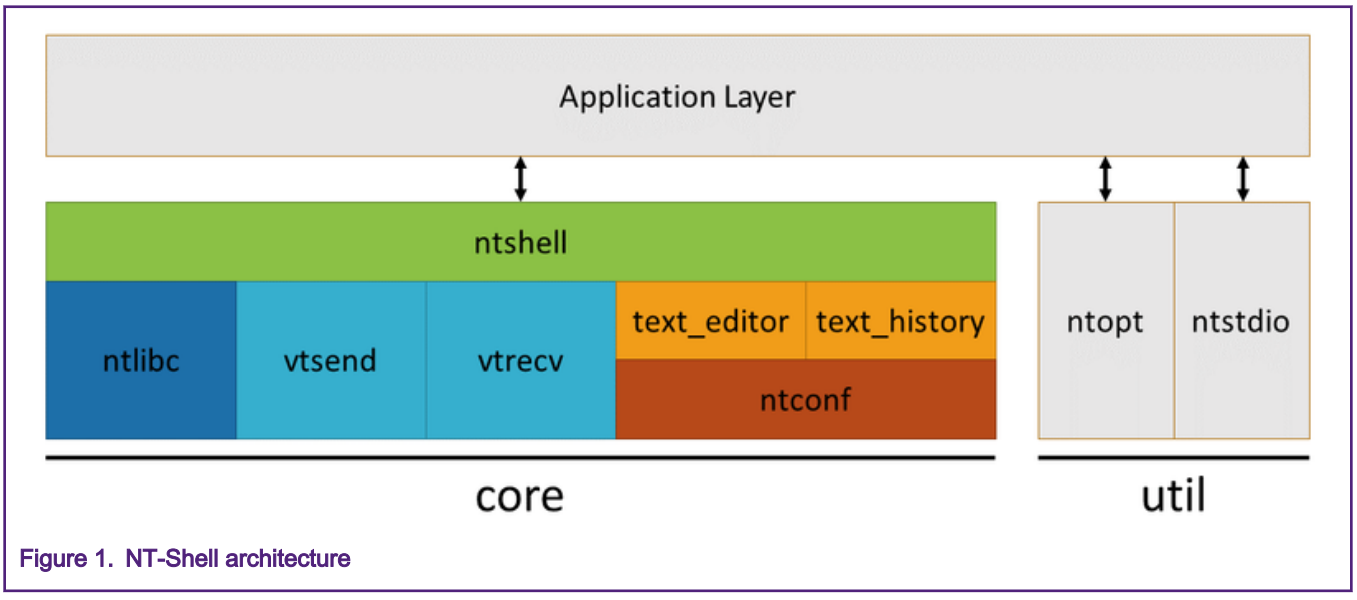

Figure 1. NT-Shell architecture

Figure 2 exposed the NT-Shell functions call graph. NT-Shell function APIs are quite simple.

To enable NT-Shell function in real application, user only need call below function APIs in main or RTOS thread.

```
func_read()
func_write()
func_callback()
ntshell_init()
ntshell_set_prompt()
ntshell_execute()
```

When ported NT-Shell to a new MCU platform, coder need care about

```
uart_getc()
uart_putc()
```

Chapter X.X will introduce the porting activities.

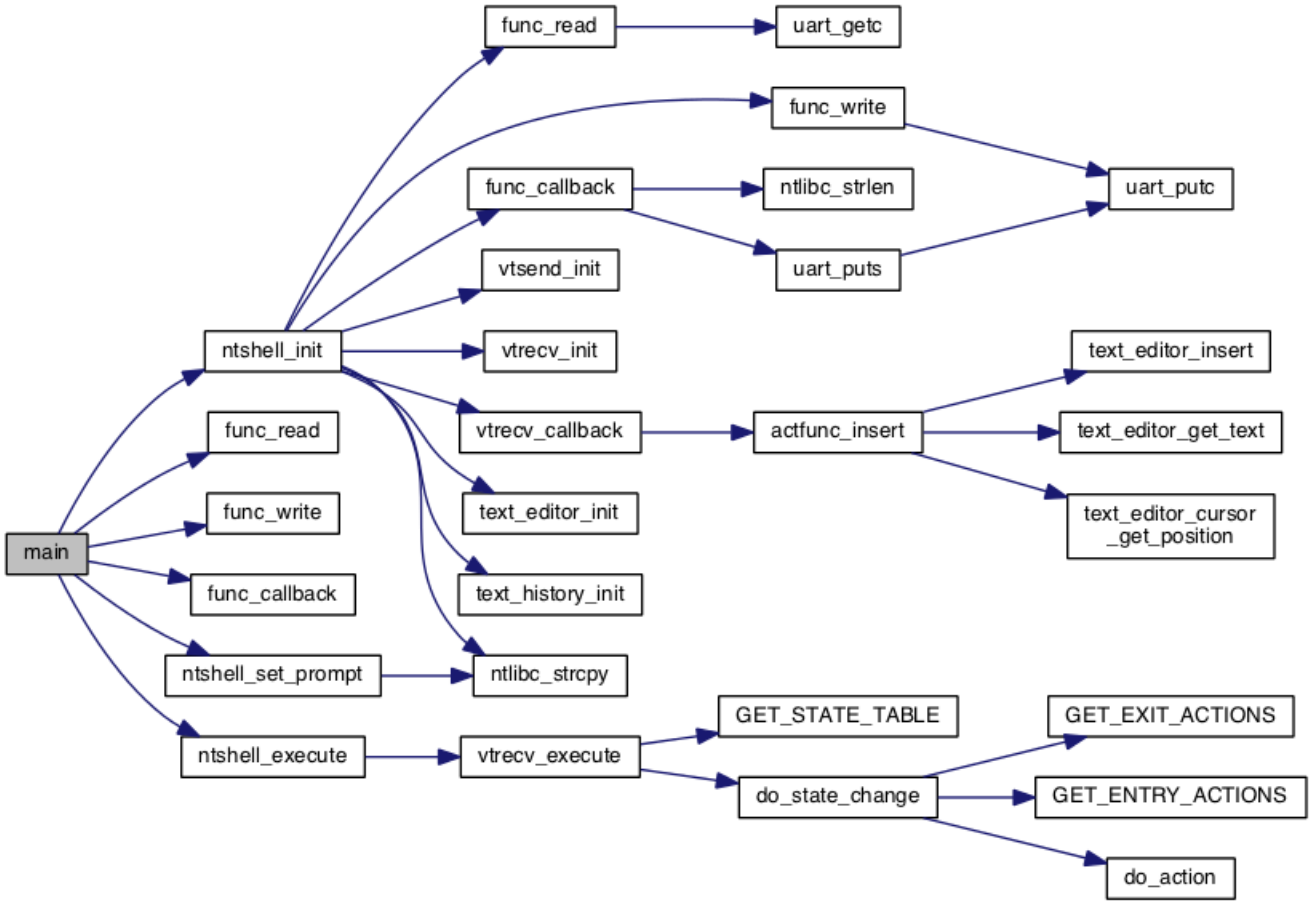

Figure 2. NT-Shell function call graph

3 NT-Shell on LPC55S16-EVK Demo

3.1 LPC55S16-EVK board

The LPC55S16-EVK board supports a VCOM serial port connection via **J1**. To observe debug messages from the board set the terminal program to the appropriate COM port and use the setting '115200-8-N-1-none'. To make the debug messages easier to read, the new line receive setting should be set to auto.

3.2 Board Setup

The LPC55S16-EVK development board is used for customer evaluation, the function and board pictures as Figure 3 shows:

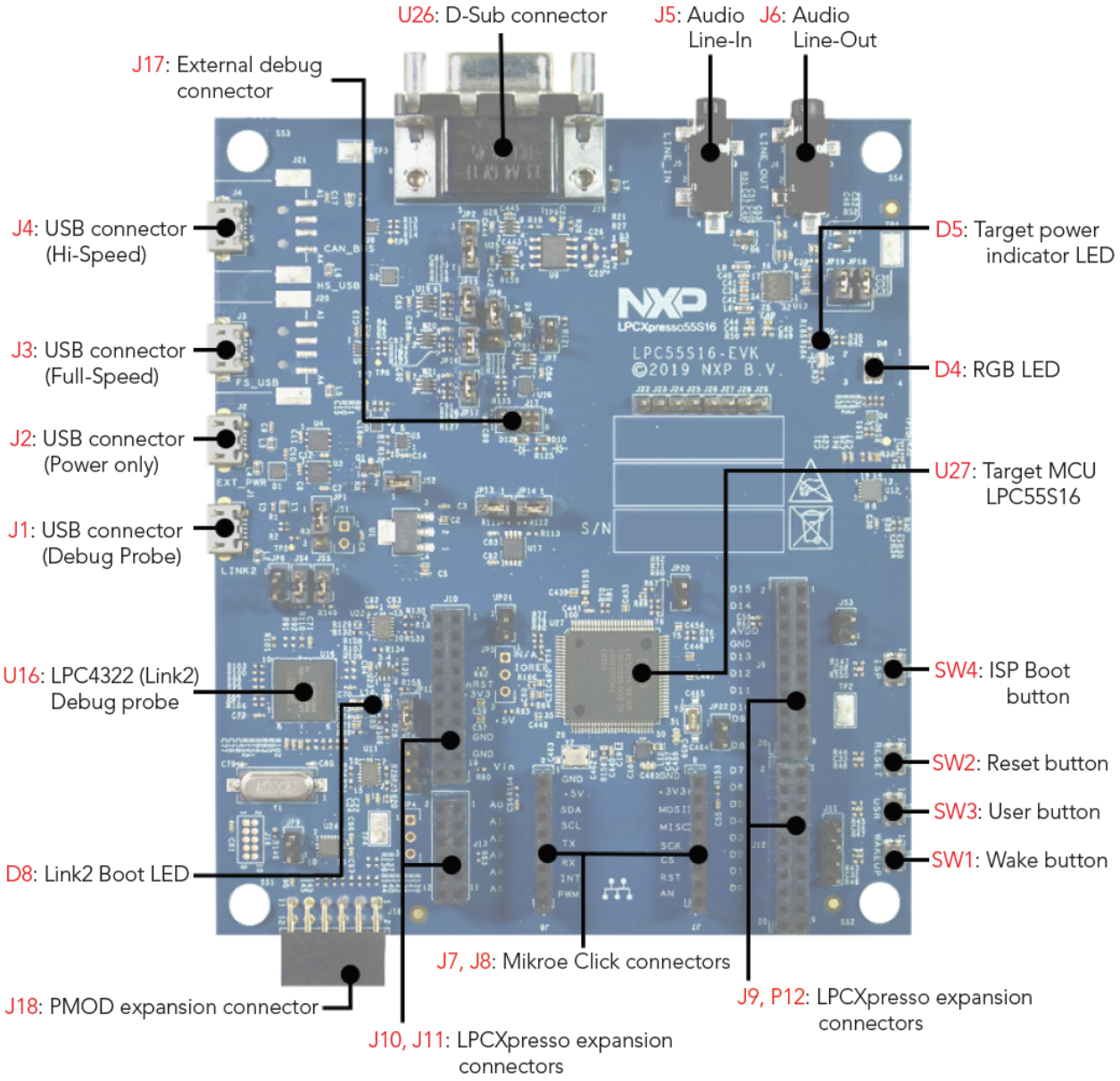

Figure 3. LPC55S16-EVK

The board ships with CMSIS-DAP debug firmware programmed. Visit the following FAQ for more information on CMSIS_DAP debug firmware: https://www.nxp.com/downloads/en/software/lpc_driver_setup.exe

For debugging and terminal debug messages, connect a USB cable to **J1** USB connector. Board schematics are available on www.nxp.com.

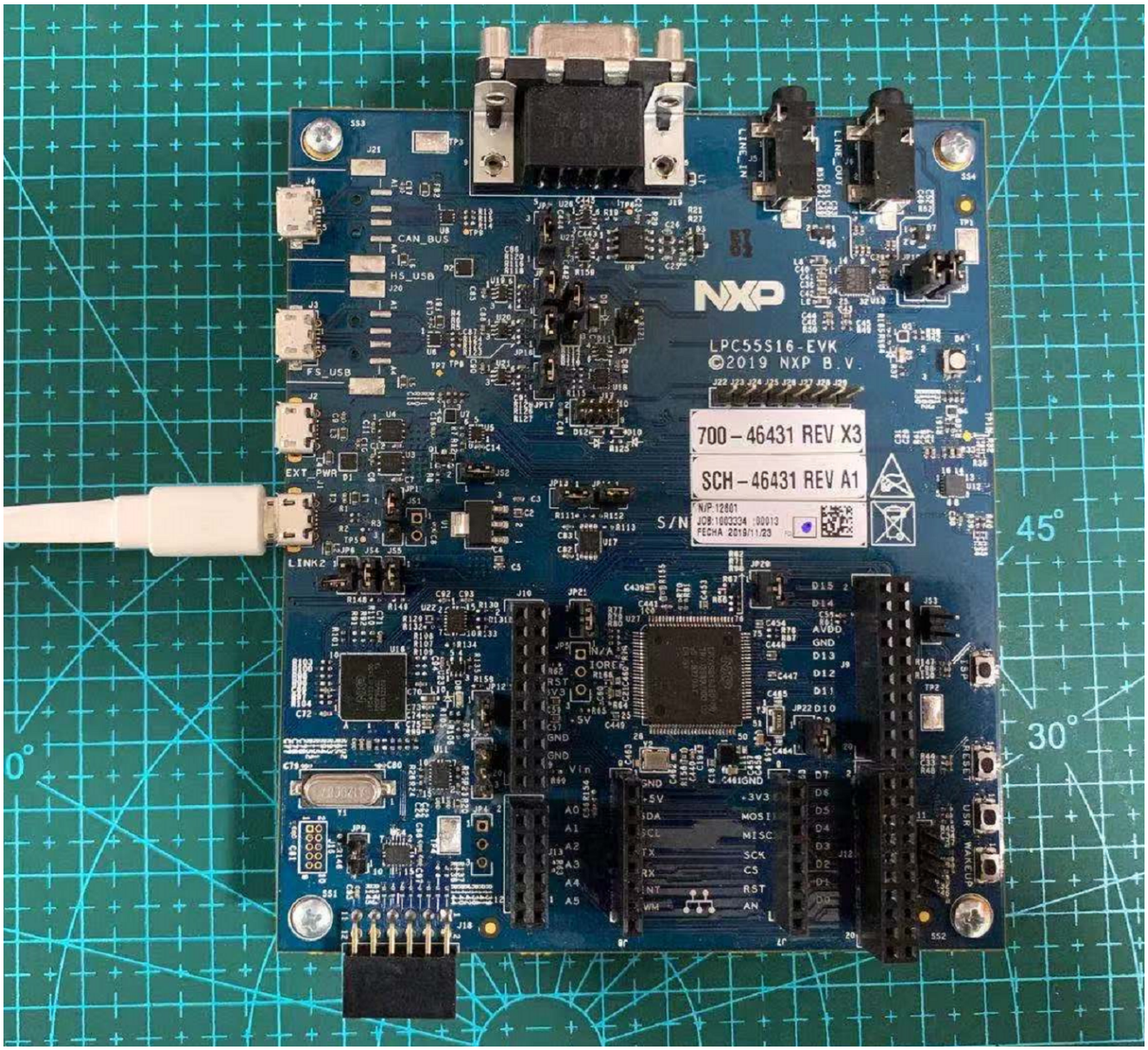

Figure 4. Connect LPC55S16-EVK with PC

3.3 Software Set-up

Three IDEs were used to verify the NT-Shell example projects:

- KEIL MDK
- IAR Embedded Workbench v8.40.2
- MCUXpresso IDE v11.1.0 (can download from www.nxp.com)

Terminal software:

Suggest use Tera Term or other terminal support uart serial port.

(can download from https://ttssh2.osdn.jp/index.html.en)

3.4 Program verification

When user download NT-Shell project and press 'RESET(SW2)' button to run the code, user can follow the information from USB Virtual COM (VCOM) port and input the command you want to test. As the Figure 5 shows, once pressed the "RESET" button, there are prompt messages in termina.

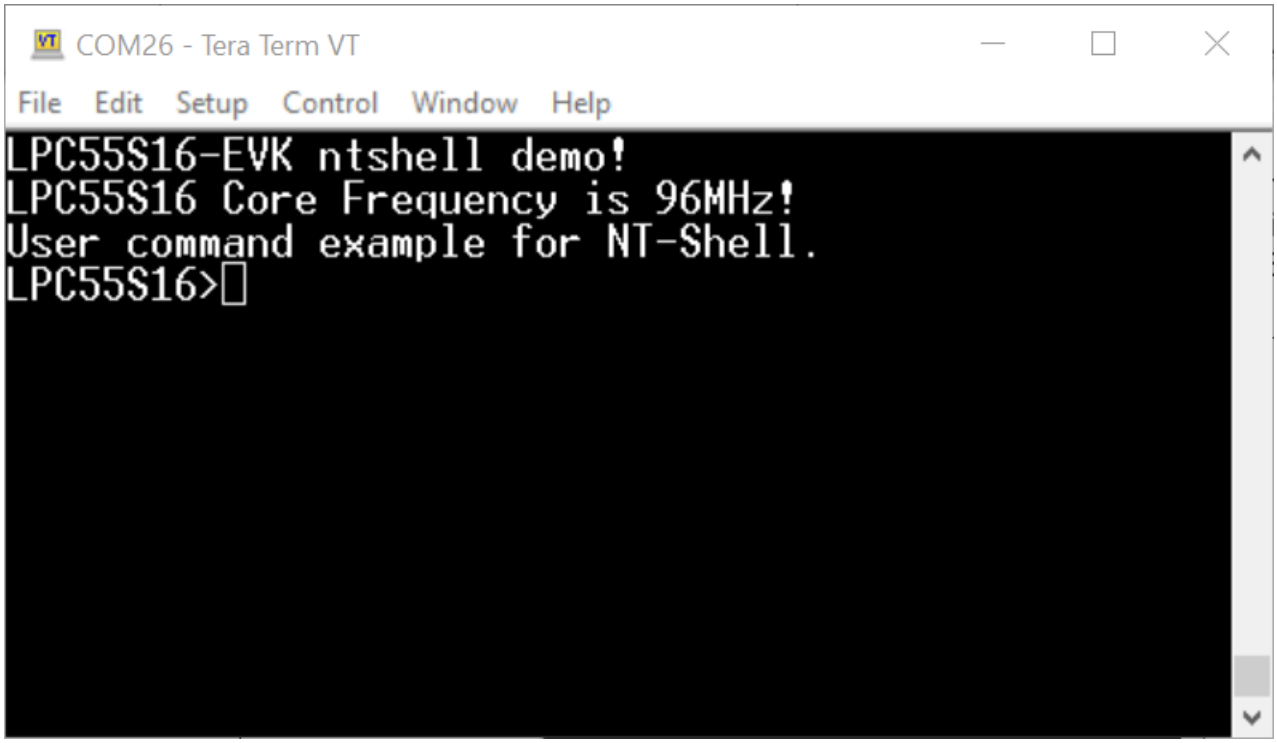

Figure 5. NT-Shell prompt message information

3.5 Demo function instruction

Once programmed NT-Shell demo code on LPC55S16-EVK board and the demo prompt message shows on terminal. User can use below commands to print system information or control led status. User can use keyboard's "Tab" to complete the command.

This demo provides kinds of commands, like help, system information get, control led. Table 1 list all the support commands.

Table 1. NT-Shell demo support command

Action	Commands by key input
Show help message	help
System information help message	info
Get system information message	info sys
Get system version message	Info ver
Command help message for LED status	led
Turn on LED	led on
Turn off LED	led off

3.6 NT-Shell support edit controls

NT-Shell support kinds of edit control hotkeys, like search history command, move the cursor to a special position, etc. See Table 2 to get the hotkeys detail information.

Table 2. NT-Shell edit control Hotkey

Action	Key Input
Move to the start of line	CTRL+A or Home
Move to the end of line	CTRL+E or End
Move forward one character	CTRL+F or Right arrow
Move back one character	CTRL+B or Left arrow
Delete previous character	Backspace
Delete current character	CTRL+D or Delete
Cancel current input line	CTRL+C
History search (backward)	CTRL+P
History search (forward)	CTRL+N
Input suggestion from history record	TAB

4 How to port and use NT-Shell

4.1 Ported NT-Shell to a new platform

After unzip the NT-Shell source code package, the NT-Shell official sample code file tree structure as Figure 6 shows.

If user needs ported NT-Shell to a new MCU platform, copy the source code files under “lib” folder to the new project and added the .c/.h files under “lib” folder to the new project compile list.

Then copied the usrcmd.c and usrcmd.h to the new project too, user can add new command in usrcmd.c.

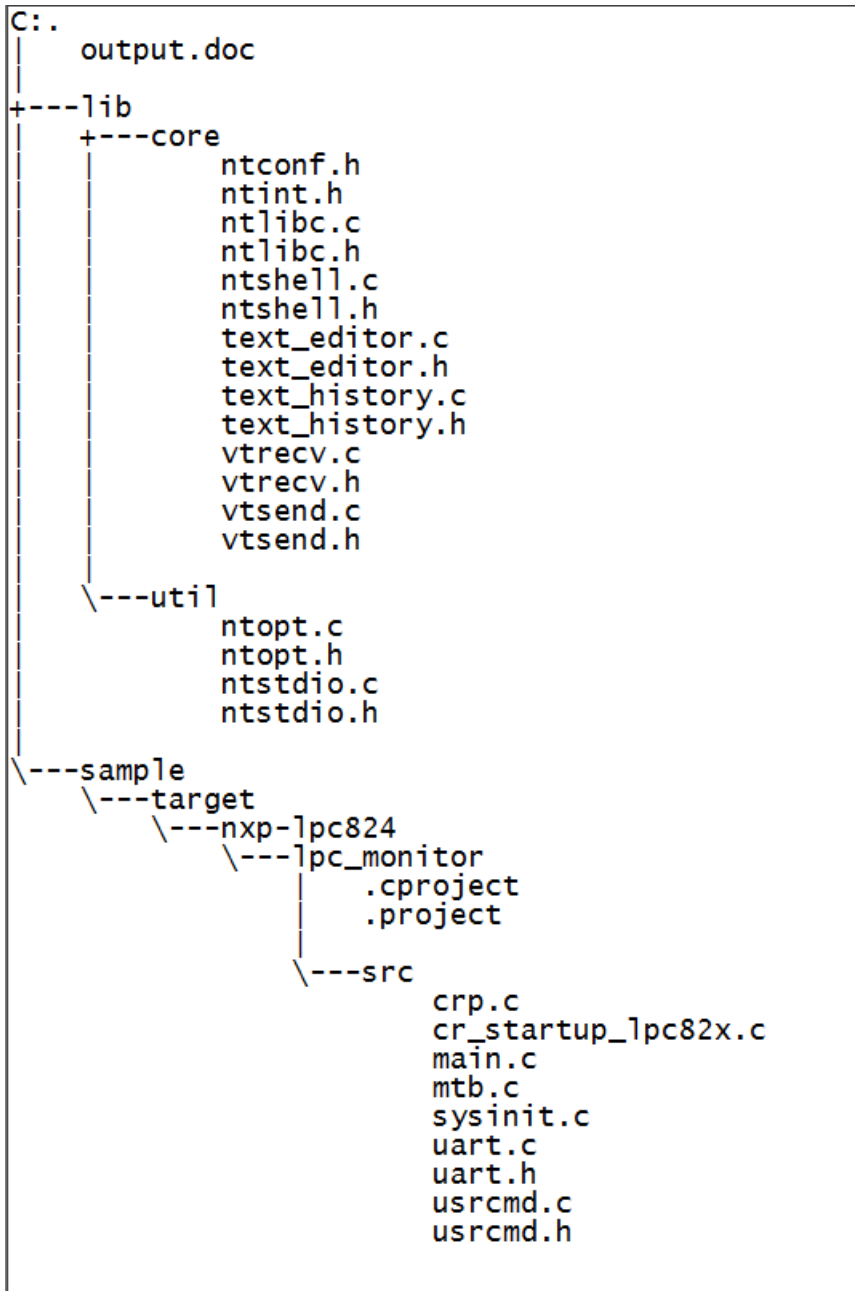

Figure 6. NT-Shell prompt message information

After added NT-Shell necessary files into the new project, user should complete the uart_getc(), uart_putc() and uart_puts() functions with SDK UART API. In this AN example code, we implement those 3 API in app_printf.c file as Figure 7 shows.

```
107 uint8_t uart_getc(void)
108 {
109     uint8_t c = 0;
110     while (1) {
111         int bytes = RingBuf_Read1Byte(&g_DebugRBuffer, &c);
112         if (bytes > 0) {
113             return c;
114         }
115     }
116 }
117
118 void uart_putc(uint8_t c)
119 {
120     USART_WriteBlocking(DEBUG_UART, &c, 1);
121 }
122
123
124 void uart_puts(char *str)
125 {
126     while (*str) {
127         uart_putc(*str++);
128     }
129 }
130
```

Figure 7. UART Operations API

Then user should added serial_read(), serial_write() and user_callback() functions into the NT-Shell initialize file. We added those three function in main.c of this example code, as Figure 8 show.

```
/*******************************************************************************
 * Code
 ******************************************************************************/
static int serial_read(char *buf, int cnt, void *extobj)
{
    for (int i = 0; i < cnt; i++) {
        buf[i] = uart_getc();
    }
    return cnt;
}

static int serial_write(const char *buf, int cnt, void *extobj)
{
    for (int i = 0; i < cnt; i++) {
        uart_putc(buf[i]);
    }
    return cnt;
}

static int user_callback(const char *text, void *extobj)
{
#if 0
    /*
     * This is a really simple example codes for the callback function.
     */
    uart_puts("USERINPUT[");
    uart_puts(text);
    uart_puts("]\r\n");
#else
    /*
     * This is a complete example for a real embedded application.
     */
    usrcmd_execute(text);
#endif
    return 0;
}
```

Figure 8. NT-Shell serial call API

4.2 How to use NT-Shell

NT-Shell introduced the key api functions and examples in below link:

https://www.cubeatsystems.com/ntshell/api.html

When initializing the NT-Shell, user should initialize the uart port first. Then we can initialize the NT-Shell by call ntshell_init() API.

Then user can set the prompt name by API ntshell_set_prompt() .

After initialize and set the prompt to NT-Shell, ntshell_execute() is the NT-Shell task function, user can call it in a while loop or in an RTOS task.

Those API's execution example as Figure 9 shows

```
/* Init debug uart port with 115200, 8n1 */
debug_init(115200);

/* log info */
PRINTF("LPC55S16-EVK ntshell demo!\r\n");
PRINTF("LPC55S16 Core Frequency is %dMHz!\r\n", SystemCoreClock/1000000);
uart_puts("User command example for NT-Shell.\r\n");
/* Init ntshell */
ntshell_init(&nts, serial_read, serial_write, user_callback, extobj);
/* Set ntshell prompt name */
ntshell_set_prompt(&nts, "LPC55S16>");

while (1)
{
    /*ntshell tasks */
    ntshell_execute(&nts);
}
```

Figure 9. NT-Shell execution example

User can add new command functions in usrcmd.c file. In this AN example, we added an LED toggle command named it as “color”.

We created a “color” command in cmdlist[], then achieve the usrcmd_color() function as Figure 10 show:

```
51  static const cmd_table_t cmdlist[] = {
52      { "help", "This is a description text string for help command.", usrcmd_help  },
53      { "info", "This is a description text string for info command.", usrcmd_info  },
54      { "led",  "LED control command for turn on/off the blue LED.",   usrcmd_led   },
55  };
56
57  int usrcmd_execute(const char *text)
58  {
62  static int usrcmd_ntopt_callback(int argc, char **argv, void *extobj)
63  {
78  static int usrcmd_help(int argc, char **argv)
79  {
91  static int usrcmd_info(int argc, char **argv)
92  {
110 static int usrcmd_led(int argc, char **argv)
111 {
112     if (argc != 2) {
113         uart_puts("led on\r\n");
114         uart_puts("led off\r\n");
115         return 0;
116     }
117     if (ntlibc_strcmp(argv[1], "on") == 0) {
118         led_set(LEDB_NUM, 1);
119         return 0;
120     }
121     if (ntlibc_strcmp(argv[1], "off") == 0) {
122         led_set(LEDB_NUM, 0);
123         return 0;
124     }
125     uart_puts("Unknown sub command found\r\n");
126     return -1;
127 }
```

Figure 10. How to add new command

5 Conclusion

This application note describes a shell solution which makes debug and get log information when develop LPC55S1x with the SDK. The NT-Shell is easy porting and support VT100.

Great thanks to Shinichiro Nakamura created such a beautiful shell code.

6 Reference

[1] LPC55S1x User Manual UM11295 (Rev. 0.7), NXP Semiconductors, 13 Dec 2019

[2] The NT-Shell official site, https://www.cubeatsystems.com/ntshell/

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express 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 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: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

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

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

Date of release: 20 February 2020

Document identifier: AN12746

arm