AN12456

Shell Solution on NXP LPC5500 Series

Rev. 0 — June, 2019 Application Note

Contents

1 Introduction

The LPC5500 is an Arm® Cortex®-M33-based micro-controller for embedded applications. These devices include:

- up to 320 KB of on-chip SRAM
- up to 640 KB on-chip flash
- high-speed and full-speed USB host
- device interface with crystal-less full-speed operations
- five general-purpose timers
- one SCTimer/PWM
- one RTC/alarm timer
- one 24-bit Multi-Rate Timer (MRT)
- one Windowed WatchDog Timer (WWDT)
- eight flexible serial communication peripherals (each can be an USART, SPI, I^2C, or I^2S interface)
- one 16-bit 1.0 Msps ADC
- one temperature sensor

The Arm Cortex- M33 provides a security foundation, offering isolation to protect valuable IP and data with TrustZone® technology.

In the embedded system application, a shell function is helpful to output log information and easily debug some standalone function API. Natural Tiny Shell (NT-Shell, written by Shinichiro Nakamura, is a C library for embedded systems. It 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 commands from the terminal. We have added how to control the led toggle status command based on the basic NT-Shell demonstration.

The sample software is tested on LPCXpresso55S69 EVK evaluation board. The software is available for three IDE's/toolchains:

- MCUXpresso
- Keil µVision
- IAR EWARM

2 NT-Shell overview

2.1 Features

NT-Shell contains the following features:

- Compatible with VT100
- Really simple
- Highly portable
  - Compatible with C89
  - No dependencies (even libc!)
  - No dynamic memory allocation (no need for an operation system!)
- Small code foot print
  - ROM: 10 KB
  - RAM: 1 KB

2.2 License claim

The license of NT-Shell is MIT. For details, refer to http://opensource.org/licenses/mit-license.php.

`vtparse` and `vtparse_table` are in the public domain.

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

You can also select TOPPERS license. For details, refer to https://www.cubeatsystems.com/ntshell/license.html.

2.3 Source code download

Users can download NT-Shell source codes from https://www.cubeatsystems.com/ntshell/download.html.

2.4 Architecture

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

The **core** branch includes four parts, as shown in Figure 1. on page 3.

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

The **Utility** branch only contains `ntopt.c/.h` and `ntstdio.c/.h`.

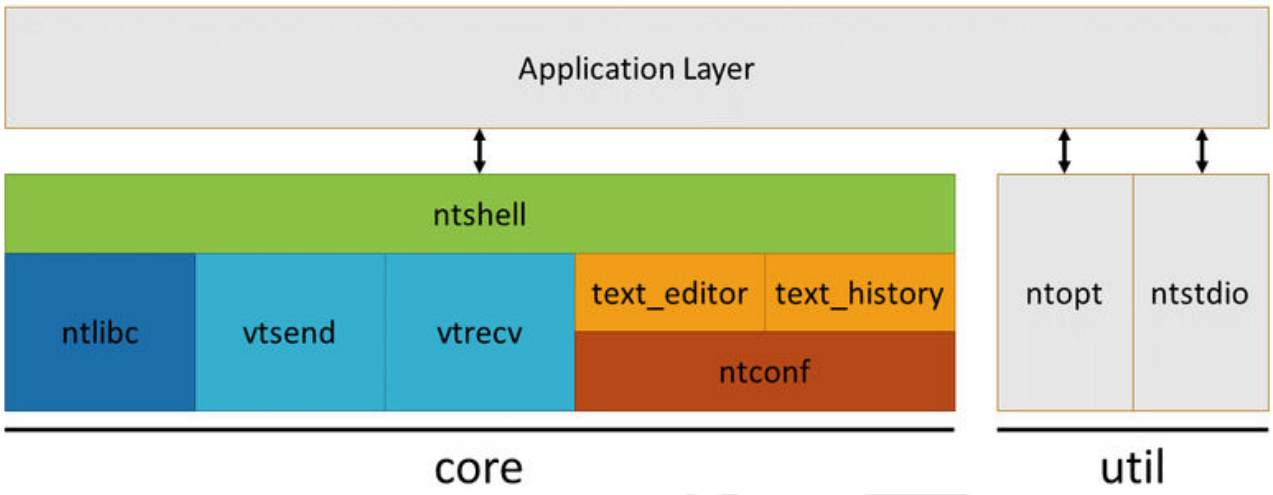

Figure 1. NT-Shell architecture

Figure 2. on page 4 shows the NT-Shell functions call graph. NT-Shell function APIs are quite simple.

To enable the NT-Shell function in a real application, users only need to call the following function APIs in thr main or RTOS thread.

- func_read()
- func_write()
- func_callback()
- func_init()
- func_set_prompt()
- func_execute()

When porting the NT-Shell to a new MCU platform, check the following codes carefully:

- uart_getc()
- uart_putc()

For the porting activities, refer to Porting NE-Shell to a new platform on page 8.

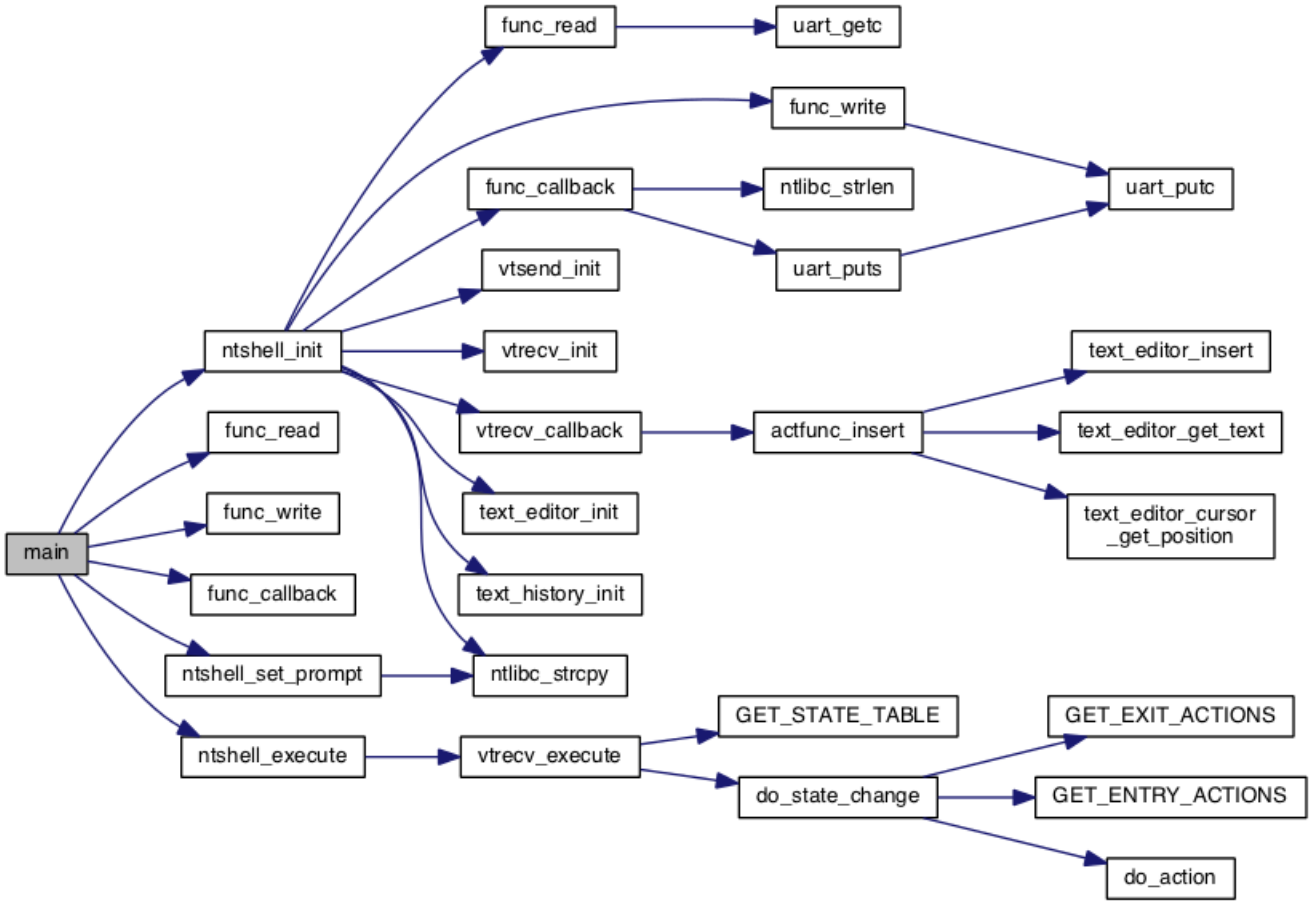

Figure 2. NT-Shell function call graph

3 NT-Shell on LPC5500 demo

3.1 LPC55S69Xpresso board

The LPC55S69Xpresso board supports a VCOM serial port connection via **P6**. To observe debug messages from the board, set the terminal program to the appropriate COM port and use the setting of **115200-8-N-1-none**. To make the debug messages easier to read, set the new line receive to **auto**.

3.2 Board setup

The LPCXpress55S69 development board is used for customer evaluation. Figure 3. on page 5 shows the board functions and setup.

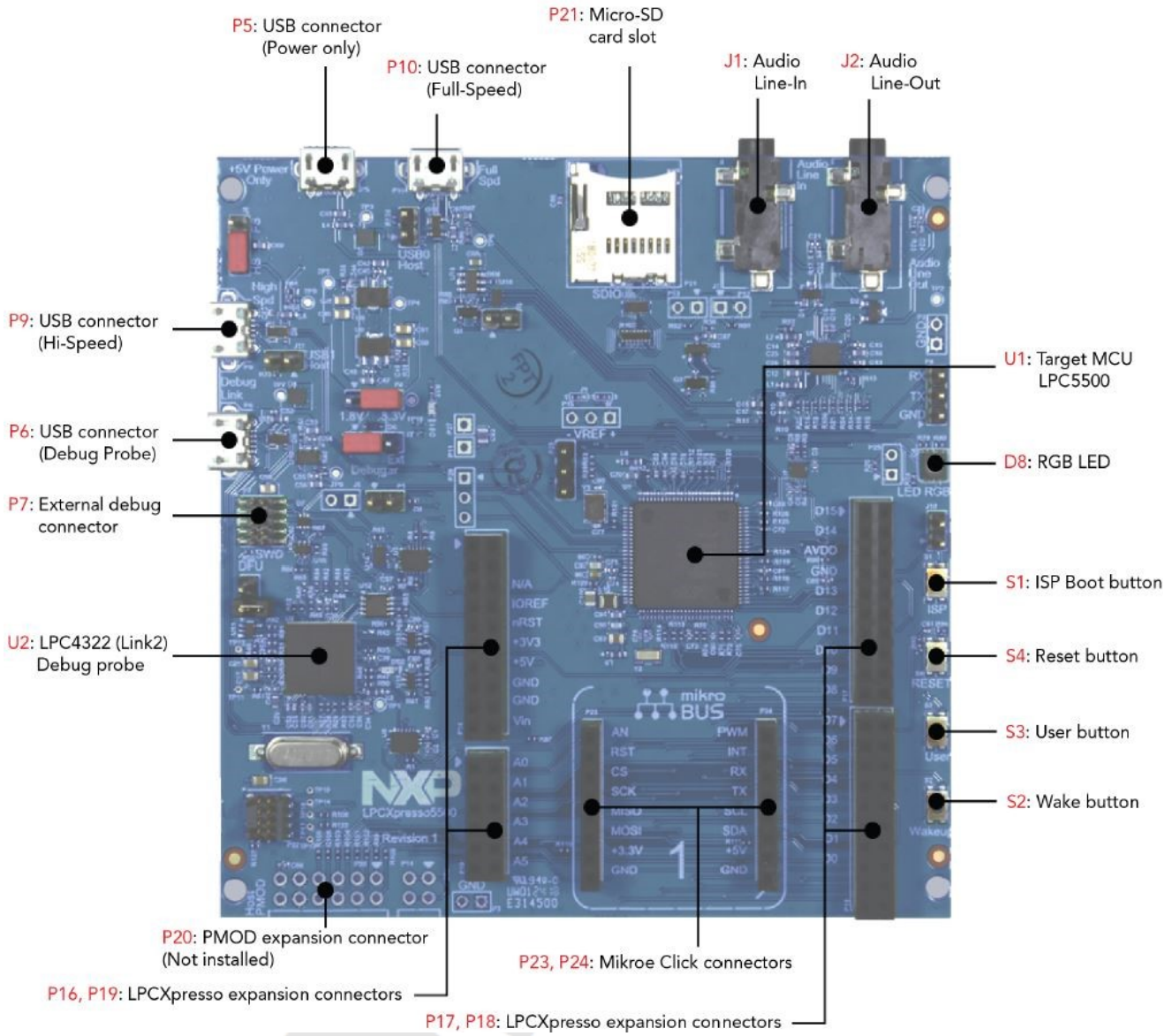

Figure 3. LPCpresso55S69 EVK

The board ships with CMSIS-DAP debug firmware programmed. For more information on CMSIS_DAP debug firmware, visit the following FAQ:

https://www.nxp.com/downloads/en/software/lpc_driver_setup.exe

To debug and terminate debug messages, connect a USB cable to P6 USB connector, as shown in Figure 4. on page 6. Board schematics are available on https://www.nxp.com.

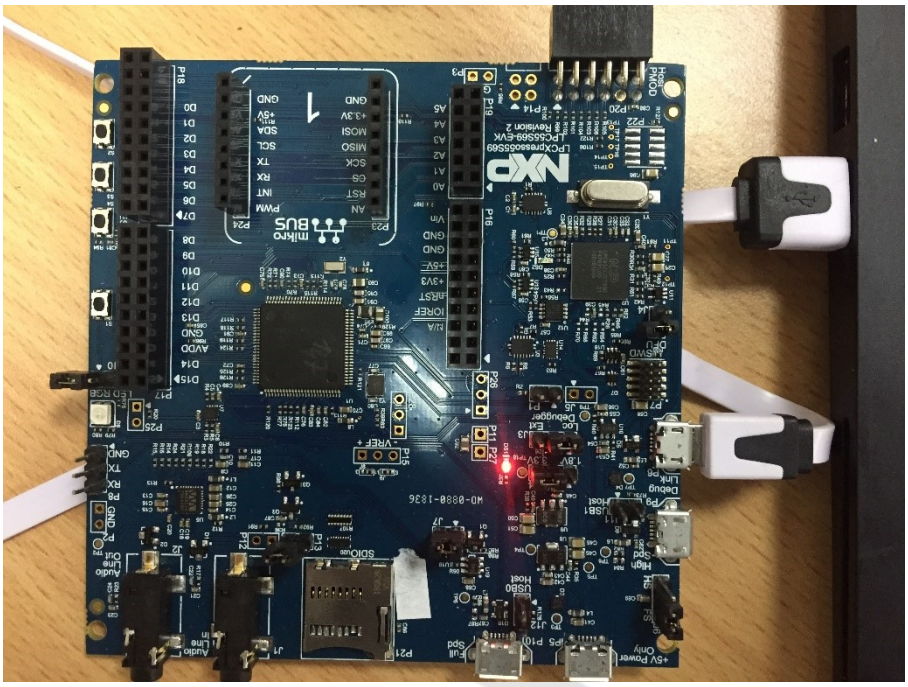

Figure 4. Connecting LPCpresso55S69 with PC

3.3 Software setup

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

- KEIL MDK
- IAR Embedded Workbench v8.32.1
- MCUXpresso IDE v10.3.1 (it can be downloaded from https://www.nxp.com)

Terminal software:

Tera Term or other terminal support uart serial port is suggested (they can be downloaded from https://ttssh2.osdn.jp/index.html.en).

3.4 Program verification

When downloading the NT-Shell project and pressing the **RESET(S4)** button to run the code, user can follow the information from USB Virtual COM (VCOM) port and input the command you want to test. Once the **RESET** button is pressed, there are prompt messages on the terminal, as shown in Figure 5. on page 7.

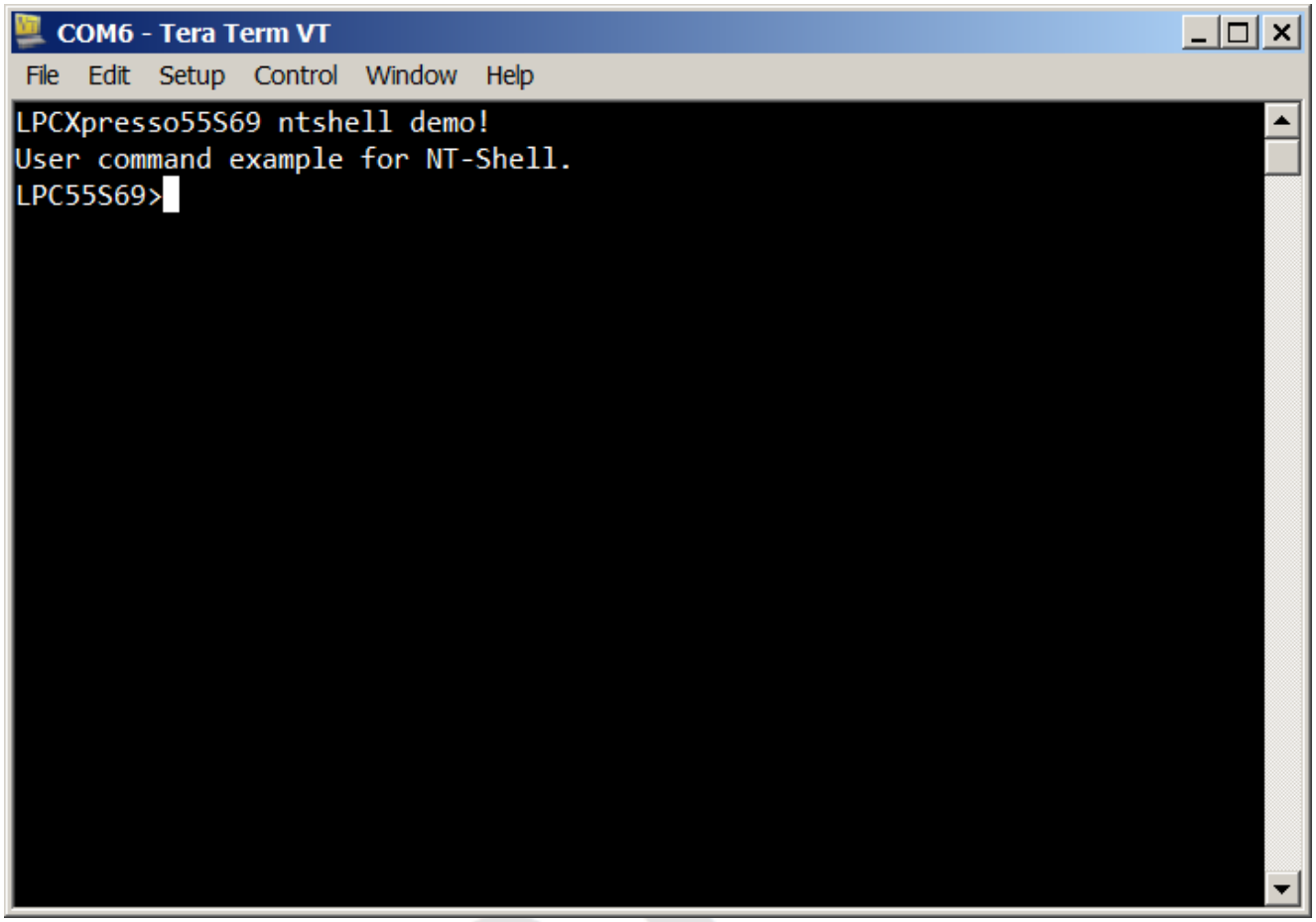

Figure 5. NT-Shell prompt message

3.5 Demo function instruction

Once programmed NT-Shell demo code on LPCXpresso55S69 board and the demo prompt message shows on terminal. Users can use the following commands to print system information or control led status. Users can use the **Tab** key to complete the commands.

This demo provides multiple commands, such as, help, system information get, and control led. Table 1. NE-Shell demo support command on page 7 lists all the support commands.

Table 1. NE-Shell demo support command

Action	Commands by key input
To show help message	help
The system information help message	info
To get system information message	info sys
To get system version message	Info ver

Table continues on the next page...

Table 1. NE-Shell demo support command (continued)

The command help message for LED status	color
To toggle RED led	color red
To toggle Green led	color green
To toggle Blue led	color blue

3.6 NT-Shell-supported edit controls

NT-Shell supports multiple edit control hotkeys, such as, search history command, move the cursor to a special position, and so on. Table 2. NT-Shell-supported edit control hotkeys on page 8 describes the hotkeys and related information.

Table 2. NT-Shell-supported edit control hotkeys

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 Porting and using NT-Shell

4.1 Porting NE-Shell to a new platform

After the NT-Shell source code package is unzipped, the structure of NT-Shell official sample code file tree is as shown in Figure 6. on page 9.

To port NT-Shell to a new MCU platform, copy the source code files under the `lib` folder to the new project and add the `.c/.h` files under the `lib` folder to the new project compile list.

You can add a new command into the `usrcmd.c` by copying the `usrcmd.c` and `usrcmd.h` to the new project.

NOTE

Make sure that the STACK size of the new project is enough. Otherwise, the code will generate hard-fault when running.

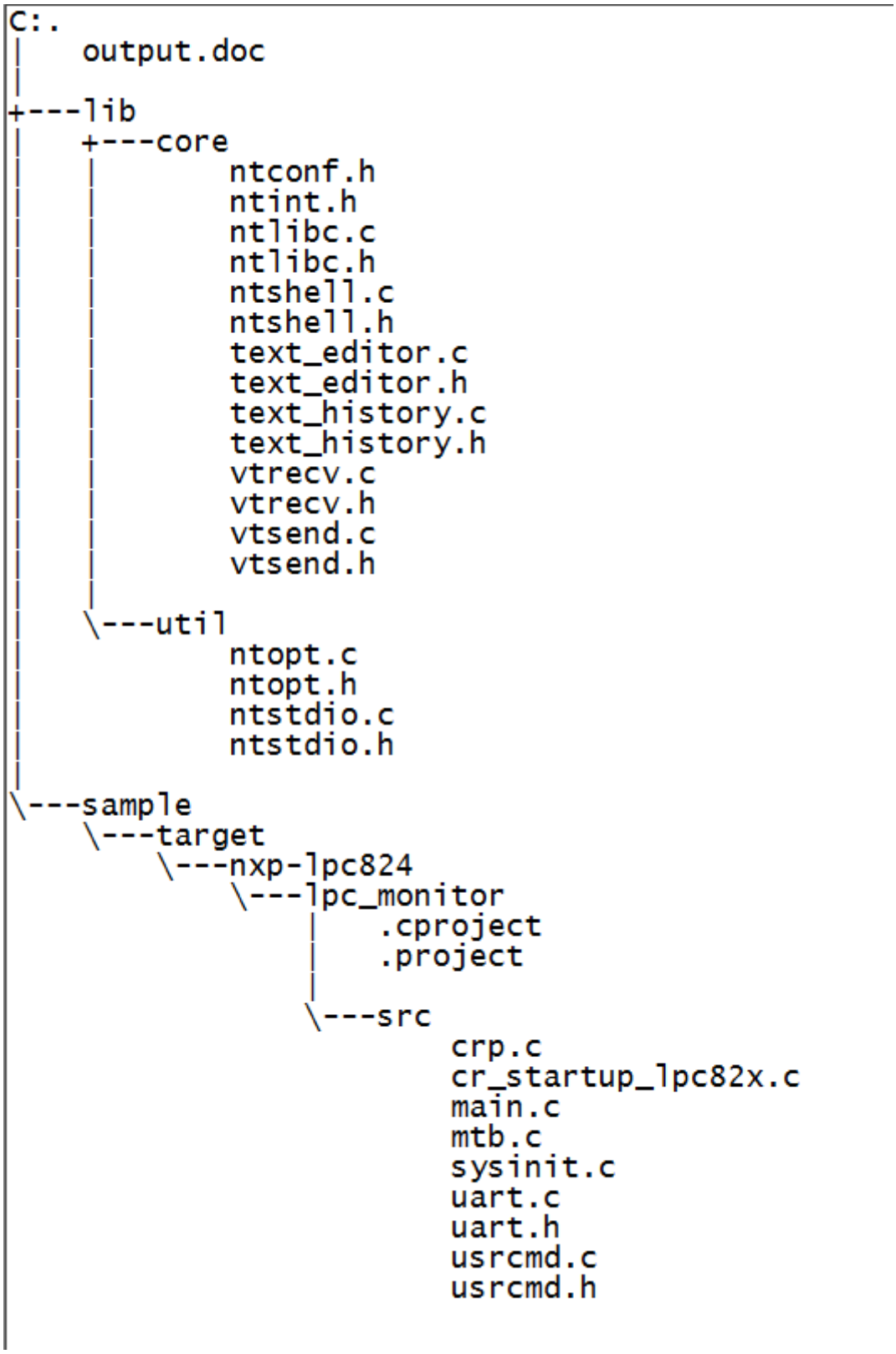

Figure 6. NT-Shell prompt message

After adding the required NT-Shell files into the new project, you can complete the `uart_getc()`, `uart_putc()`, and `uart_puts()` functions with SDK UART API. Figure 7. on page 10 shows the example to implement the three API in the `app_printf.c` file.

```
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 operation API

Also, you can add `serial_read()`, `serial_write()`, and `user_callback()` functions into the NT-Shell initialization file. Figure 8. on page 11 shows the example to add the three functions into the `main.c`.

```
/*******************************************************************************
 * 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 Using NT-shell

You can find the functions and examples of NT-Shell key APIs on https://www.cubeatsystems.com/ntshell/api.html.

The following API execution examples are as shown in Figure 9. on page 12.

- To initialize the NT-Shell, you need to initialize the UART port first and then call `ntshell_init()` API.
- With the `ntshell_set_prompt()` API, you can set the prompt name.
- `ntshell_execute()` is the NT-Shell task function. You can call it in a while loop or in a RTOS task.

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

/* log info */
PRINTF("LPCXpresso55S69 ntshell demo!\r\n");
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, "LPC55S69>");

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

Figure 9. NT-Shell execution examples

Figure 10. on page 13 shows how to add a new command.

A new command can be added in the `usrcmd.c` file. In this AN example, an LED toggle is added with the name of **color**.

After a **color** command is created in the `cmdlist[]`, the `usrcmd_color()` function is achieved.

```
 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     { "color", "Color command for color LED.",                      usrcmd_color },
 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_color(int argc, char **argv)
111 {
112     if (argc != 2) {
113         uart_puts("color red\r\n");
114         uart_puts("color blue\r\n");
115         uart_puts("color green\r\n");
116         return 0;
117     }
118     if (ntlibc_strcmp(argv[1], "red") == 0) {
119         led_toggle(LEDR_NUM);
120         return 0;
121     }
122     if (ntlibc_strcmp(argv[1], "blue") == 0) {
123         led_toggle(LEDB_NUM);
124         return 0;
125     }
126     if (ntlibc_strcmp(argv[1], "green") == 0) {
127         led_toggle(LEDG_NUM);
128         return 0;
129     }
130     uart_puts("Unknown sub command found\r\n");
131     return -1;
132 }
```

Figure 10. Adding a new command

5 Conclusion

This application note describes a shell solution which makes debug and get log information when developing the LPC55S69 with its SDK. The NT-Shell is easy for porting and it supports VT100.

Great thanks to Shinichiro Nakamura for creating such a beautiful shell code.

6 Reference

- LPC55S6x User Manual, UM11126 (Rev. 1.2), NXP Semiconductors, 3 May 2019
- 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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: June, 2019
Document identifier: AN12456

arm