AN13034

NT-Shell Solution on NXP LPC55(S)0x Series

Rev. 0 — 10/2020 Application Note

Contents

1 Introduction

The LPC55(S)0x is an Arm® Cortex®-M33 based micro-controller for embedded applications. These devices include:

- CASPER Crypto engine
- Up to 96 KB of on-chip SRAM
- Up to 256 KB on-chip flash
- PRINCE module for on-the-fly flash encryption/decryption
- One CAN-FD controller
- 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 of which can be a USART, SPI, I^2C, or I^2S interface
- One 16-bit 2.0 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 much 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 will use the USART0 to print information and get command from terminal. We've also added control the led toggle status command based on the basic NT-Shell demonstration.

The sample software is tested on LPC55S06-EVK evaluation board. The 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: 10 KB
  - RAM: 1 KB

2.2 License claim

NT-Shell's license is MIT.

The license is MIT.

`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.

The reference link is https://www.cubeatsystems.com/ntshell/license.html.

2.3 Downloading source code

NT-Shell source code can be downloaded from NT-Shell Download.

2.4 Architecture

NT-Shell have two part: **core** and **util**. Figure 1 shows the file structure.

1. Core branch includes four parts.
   - 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`)
2. **Utility** branch only contains `ntopt.c/.h` and `ntstdio.c/.h`.

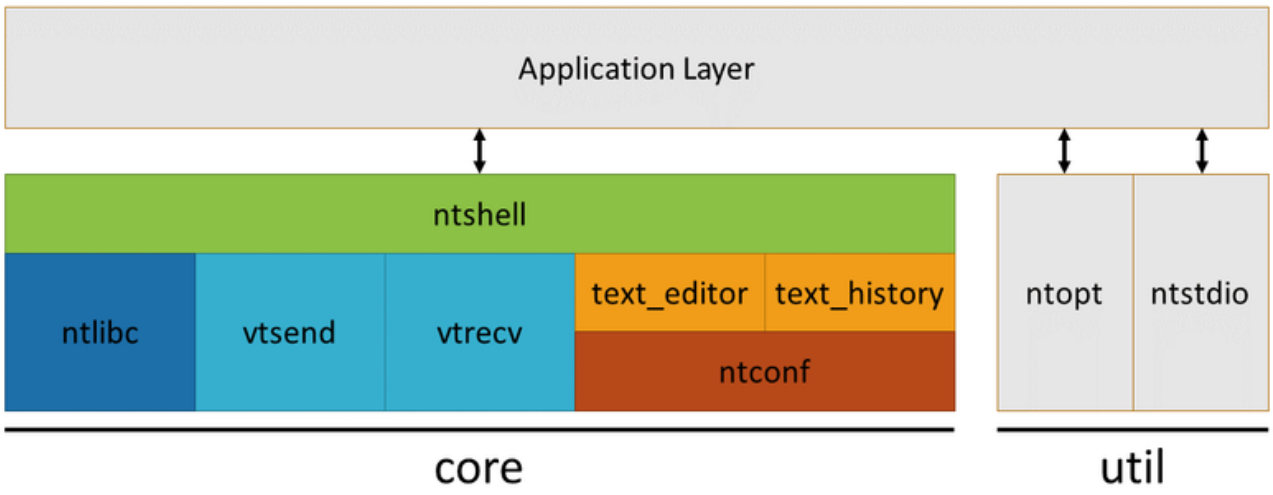

Figure 1. NT-Shell architecture

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

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

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

When porting NT-Shell to a new MCU platform, take care the coder.

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

The porting activities are introduced in How to port NT-Shell to a new platform.

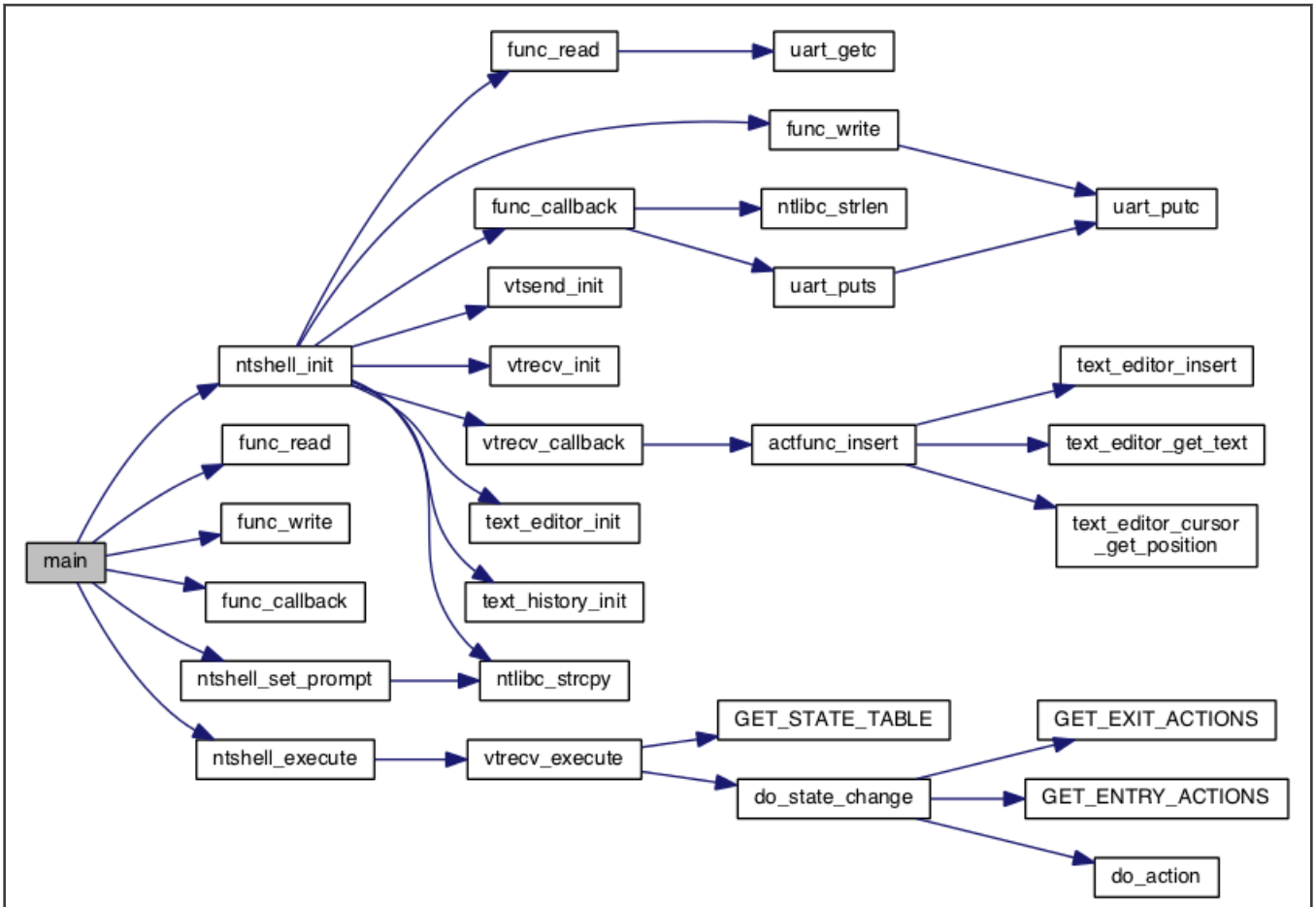

Figure 2. NT-Shell function call graph

3 NT-Shell on LPC55S06-EVK demo

3.1 LPC55S06-EVK board

The LPC55S06-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 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

NOTE

Please shunt **JP12** to enable LPC-Link2 VCOM TXD port to MCU.

The LPC55S06-EVK development board is used for customer evaluation. Figure 3 shows the function and board pictures.

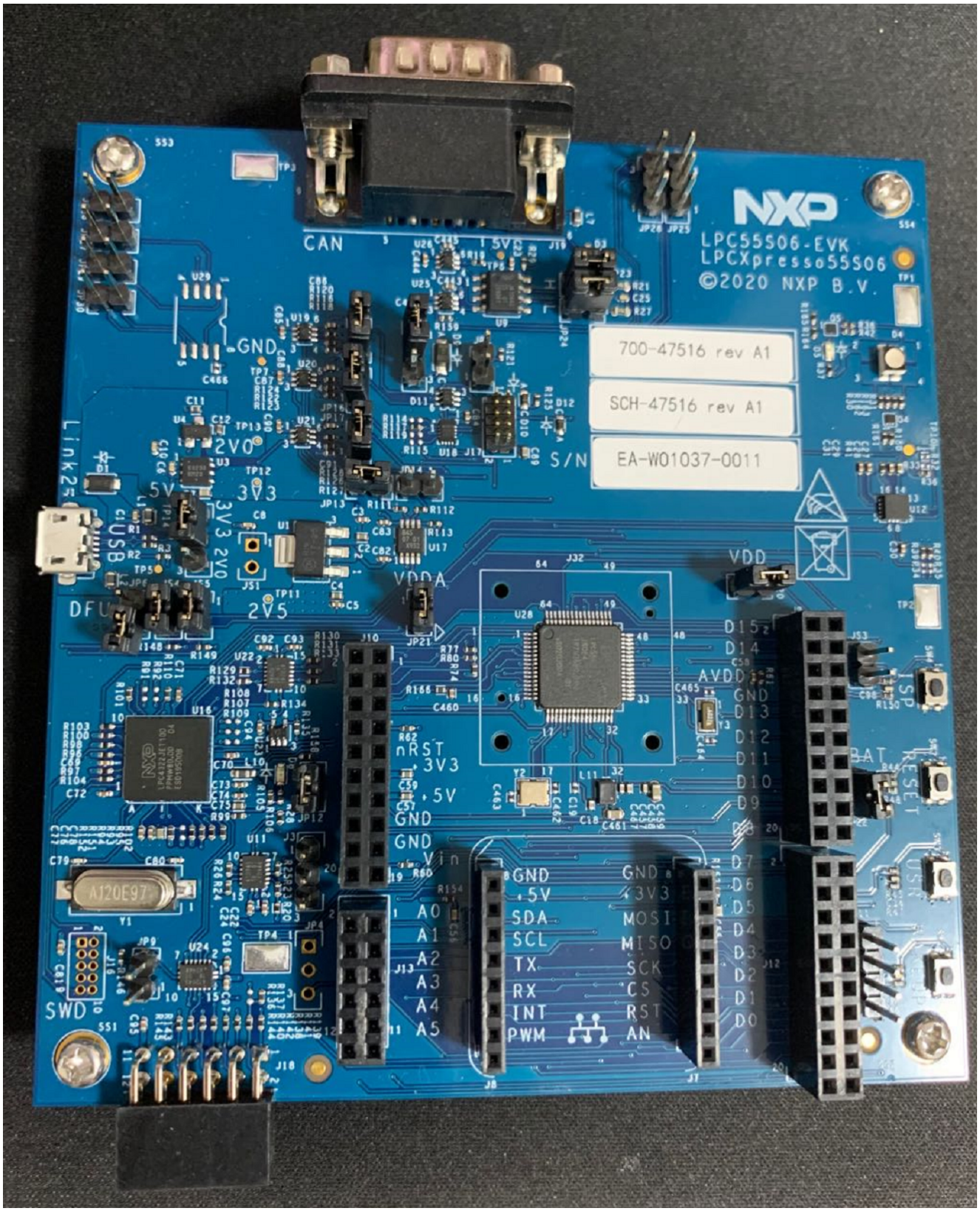

Figure 3. LPC55S06-EVK

The board ships with CMSIS-DAP debug firmware programmed. Visit FAQ for more information on `CMSIS_DAP` debug firmware.

Download the driver from LPC Driver Setup.

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

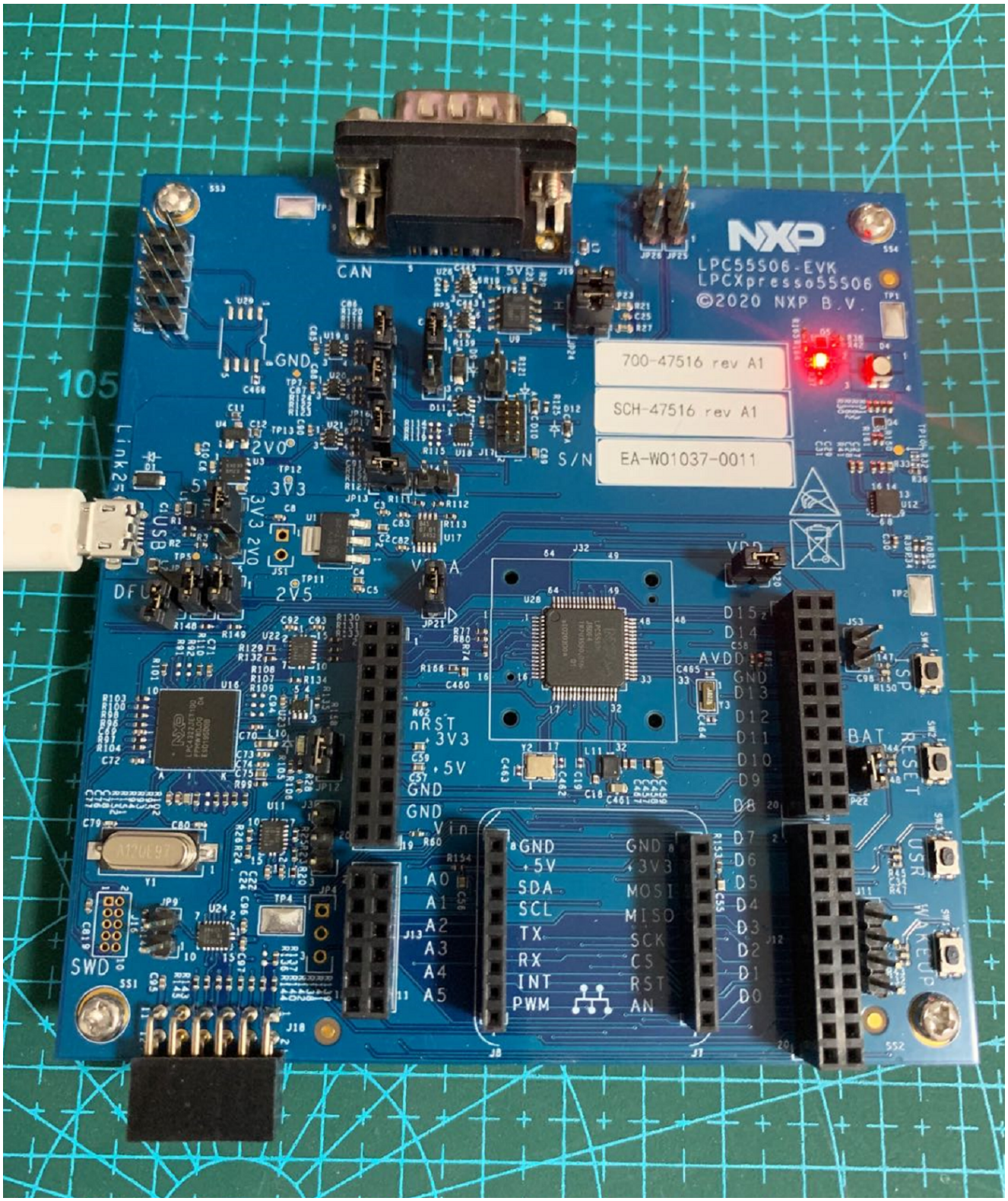

Figure 4. Connecting LPC55S06-EVK with PC

3.3 Software setup

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

- KEIL MDK
- IAR Embedded Workbench v8.50.6
- MCUXpresso IDE v11.2.1, which can be downloaded from MCUXpresso-IDE.

Terminal software:

Suggest to use Tera Term or other terminal support uart serial port, which can be downloded from Tera Term Home Page.

3.4 Program verification

When downloading NT-Shell project and pressing the **RESET (SW2)** button to run the code, users can follow the information from USB Virtual COM (VCOM) port and input the command as they want. Once the **RESET** button is pressed, there are prompt messages in the terminal, as shown in Figure 5.

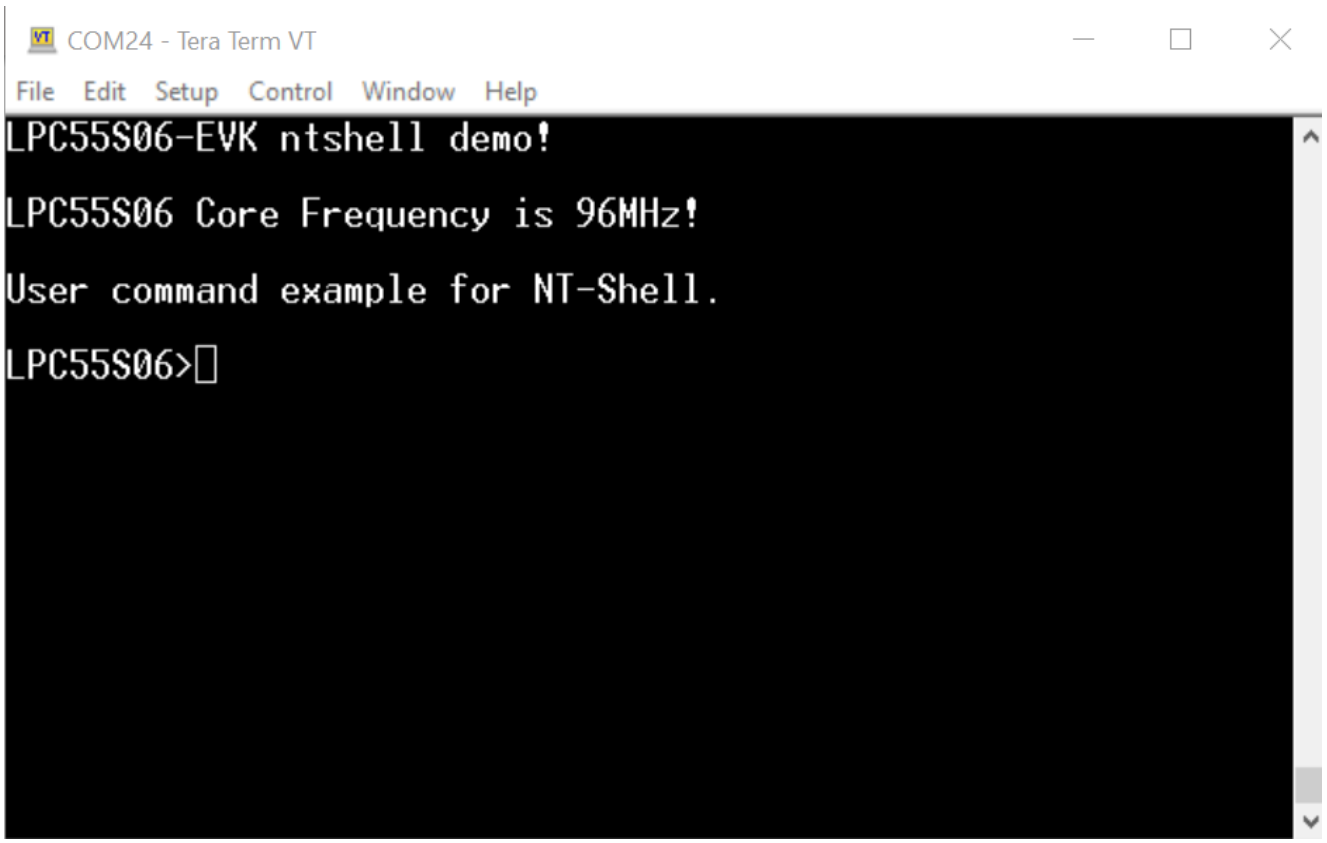

Figure 5. NT-Shell prompt message information

3.5 Demo function instruction

Once programmed NT-Shell demo code on LPC55S06-EVK board and the demo prompt message show on the terminal, users can use commands in Table 1 to print system information or control led status. Users can also use **Tab** on the keyboard to complete the command.

This demo provides kinds of commands, such as, 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 supports kinds of edit control hotkeys, such as, search history command, move the cursor to a special position, etc. Table 2 lists the detailed information about the hotkeys.

Table 2. NT-Shell 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 How to port and use NT-Shell

4.1 How to port NT-Shell to a new platform

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

To port NT-Shell to a new MCU platform, perform the steps as follows:

1. Copy the source code files under the **lib** folder to the new project.

2. Add the `.c/.h` files under the **lib** folder to the new project compile list.
3. Copy the `usrcmd.c` and `usrcmd.h` to the new project. New commands can be added in `usrcmd.c`.

NOTE

Please make sure the new project's STACK size is enough. Otherwise, the code will generate hard-fault when running.

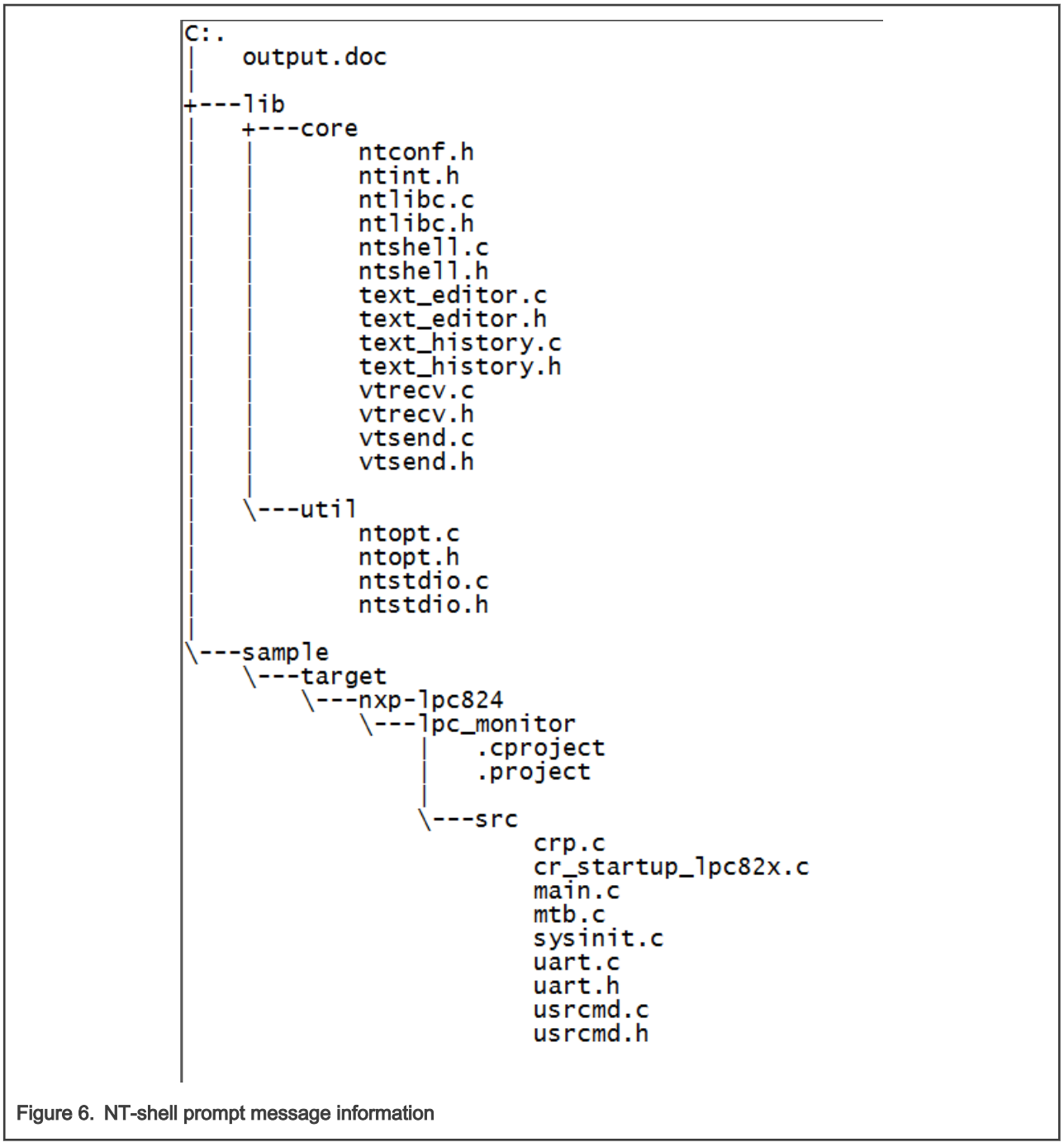

Figure 6. NT-shell prompt message information

4. After adding NT-Shell necessary files into the new project, complete the `uart_getc()`, `uart_putc()`, and `uart_puts()` functions with SDK UART API. For the example codes in this document, we implement the three API in `app_printf.c` file, as shown in Figure 7.

```
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 users should add `serial_read()`, `serial_write()`, and `user_callback()` functions into the NT-Shell initialization file. We added those three function in `main.c` of this example code, as shown in Figure 8.

```
/*******************************************************************************
 * 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

For keeping API functions and examples of NT-shell, see API.

When initializing the NT-shell, initialize the UART port first, and then the NT-Shell by calling `ntshell_init()` API.

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

After initializing and setting the prompt to NT-shell, `ntshell_execute()` is the NT-Shell task function. Users can call it in a while loop or in a RTOS task.

Figure 9 shows the APIs execution examples.

```
 84
 85      /* Init debug uart port with 115200, 8n1 */
 86      debug_init(DEBUG_UART_BAUDRATE);
 87
 88      /* log info */
 89      PRINTF("LPC55S06-EVK ntshell demo!\r\n");
 90      PRINTF("LPC55S06 Core Frequency is %dMHz!\r\n", SystemCoreClock/1000000);
 91      uart_puts("User command example for NT-Shell.\r\n");
 92      /* Init ntshell */
 93      ntshell_init(&nts, serial_read, serial_write, user_callback, extobj);
 94      /* Set ntshell prompt name */
 95      ntshell_set_prompt(&nts, "LPC55S06>");
 96
 97      while (1)
 98      {
 99          /*ntshell tasks */
100          ntshell_execute(&nts);
101      }
102  }
103
```

Figure 9. NT-Shell execution example

Users can add new command functions in the `usrcmd.c` file. For the example codes in this document, we added an LED toggle command named as **color**.

We created a **color** command in `cmdlist[]` and achieved the `usrcmd_color()` function, as shown in Figure 10.

```
 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 gets log information when develop LPC55S0x with the SDK. The NT-Shell is easy to port and it supports VT100.

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

6 References

1. *LPC55S0x/LPC550x User manual* (document UM11424)
2. NT-Shell official site

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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

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

Date of release: 10/2020

Document identifier: AN13034

arm