Adding a new NT-Shell command

Overview

embARC is an open software platform designed to help accelerate the development and production of embedded systems based on DesignWare® ARC® processors.

Natural Tiny Shell (NT-Shell) is a tiny shell software component for embedded systems included in embARC. It is useful for debugging and system management using a simple command line interface.

This article provides instructions and an example on how to add a new command to NT-Shell.

Development Environment

The development environment used in this article is the following:

Development host operating system:

- **Windows 7**

Development Toolchain for Target Platform:

- **GNU Toolchain for DWC ARC Processors, version 2015.06**

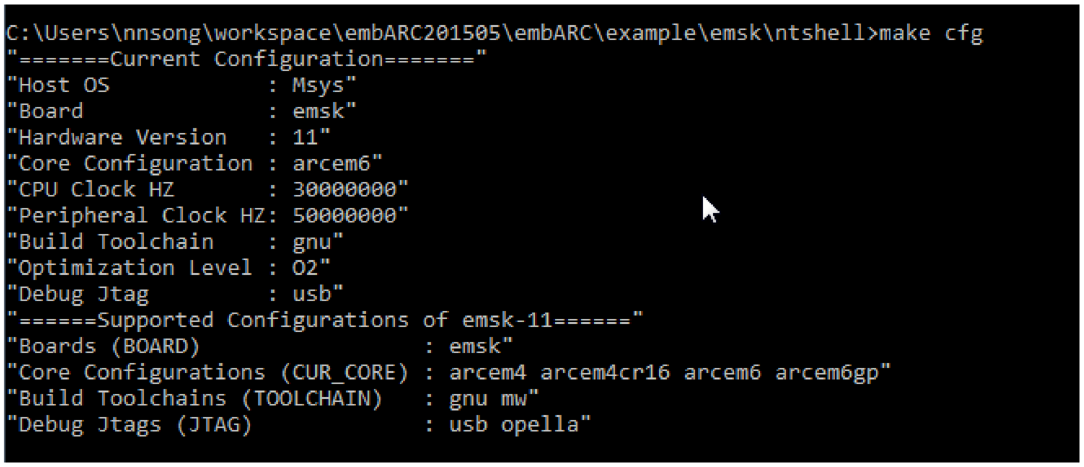
Target platform:

- **ARC EM Starter Kit (EMSK), version 1.1.**

NOTE: This article assumes the reader is already familiar with embARC. If this is your first project with embARC, please start by first reading our "Quick start" article to ensure your development environment is properly setup before you begin.

Adding a new NT-Shell command

1) Use “make cfg” to check the EMSK configuration.



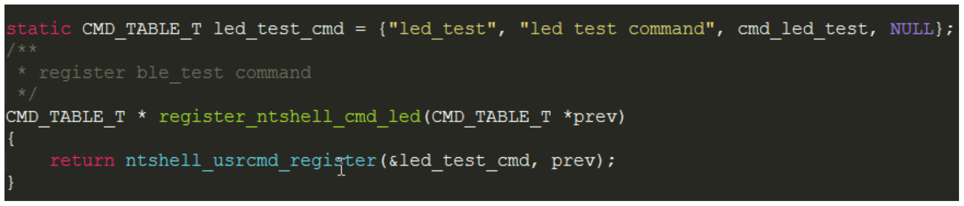
```
C:\Users\nnsong\workspace\embARC201505\embARC\example\emsk\ntshell>make cfg
"=======Current Configuration======="
"Host OS             : Msys"
"Board               : emsk"
"Hardware Version    : 11"
"Core Configuration  : arcem6"
"CPU Clock HZ        : 30000000"
"Peripheral Clock HZ: 50000000"
"Build Toolchain     : gnu"
"Optimization Level  : O2"
"Debug Jtag          : usb"
"======Supported Configurations of emsk-11======"
"Boards (BOARD)                : emsk"
"Core Configurations (CUR_CORE) : arcem4 arcem4cr16 arcem6 arcem6gp"
"Build Toolchains (TOOLCHAIN)   : gnu mw"
"Debug Jtags (JTAG)             : usb opella"
```

2) Add a new.c file for the new command to the NT-Shell project. In this article we show how to create a new command to toggle board LEDs on and off.

   Create a new file named “**cmd_led_test.c**” from sources in Appendix A.

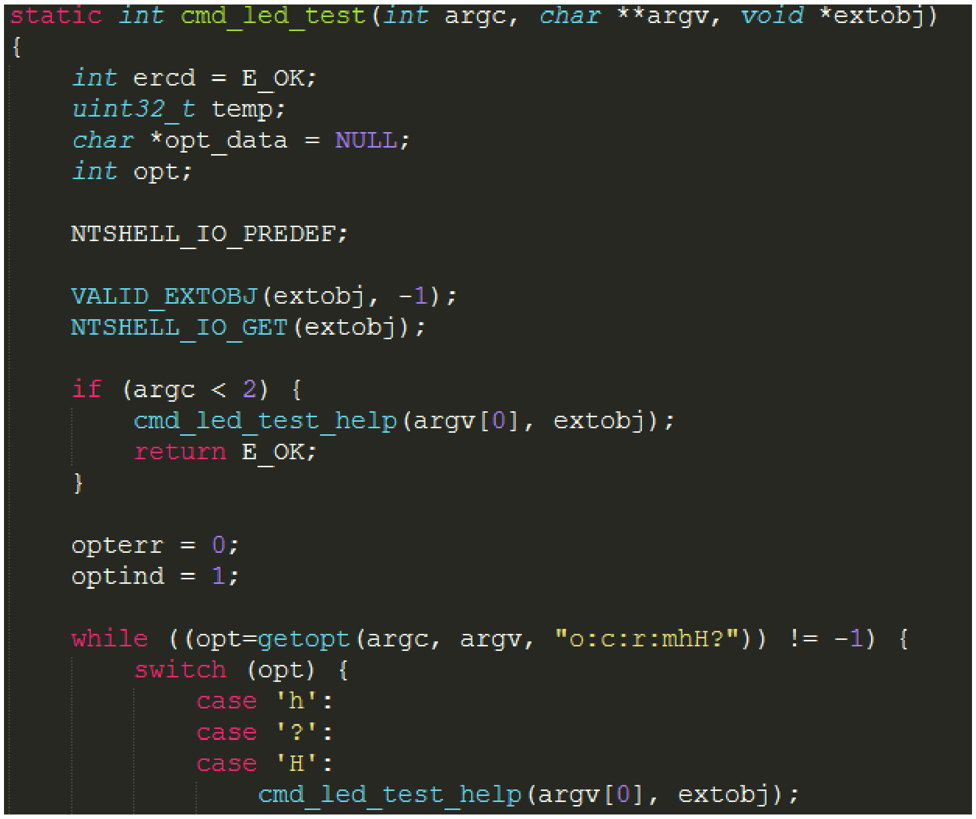
   Let’s take a closer look at the content of this file:

   a. The function “*register_ntshell_cmd_led*” is the register function of NT-Shell. “*CMD_TABLE_T*” is the structure of NT-Shell command. String “*led_test*”, String “*led_test_command*” and function “*cmd_led_test*” are command name, command simple description and command callback function, respectively. The last term in “*led_test_cmd*” is used to configure the command linked list. It should be set to “*NULL*” for command initialization.



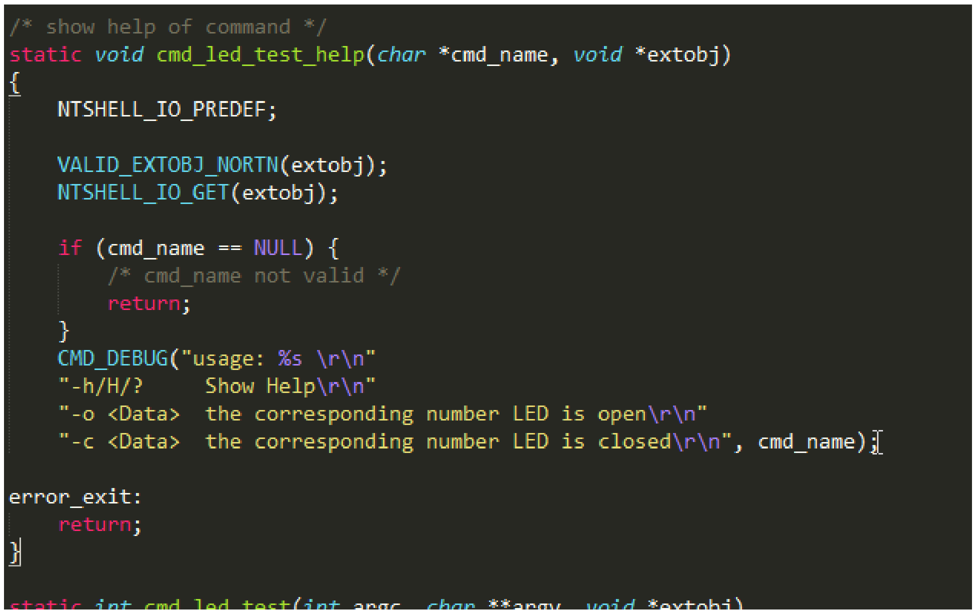
```
static CMD_TABLE_T led_test_cmd = {"led_test", "led test command", cmd_led_test, NULL};
/**
 * register ble_test command
 */
CMD_TABLE_T * register_ntshell_cmd_led(CMD_TABLE_T *prev)
{
    return ntshell_usrcmd_register(&led_test_cmd, prev);
}
```

b. The callback function “*cmd_led_test*” is a standard function for NT-Shell command.



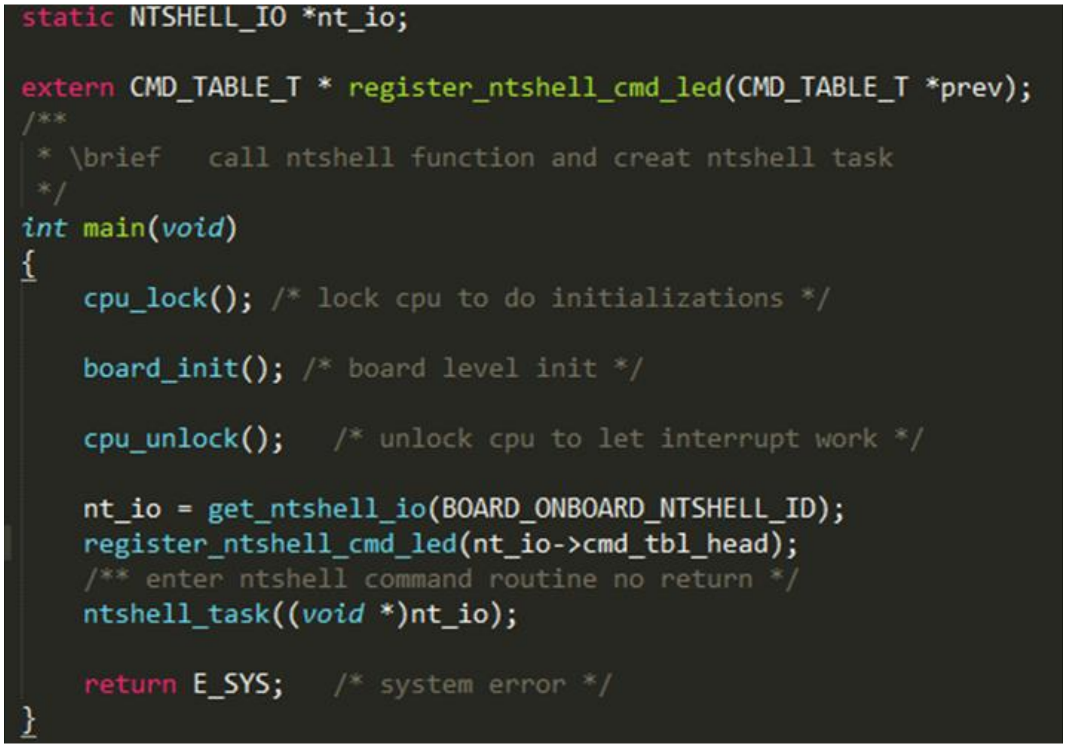
```
static int cmd_led_test(int argc, char **argv, void *extobj)
{
    int ercd = E_OK;
    uint32_t temp;
    char *opt_data = NULL;
    int opt;

    NTSHELL_IO_PREDEF;

    VALID_EXTOBJ(extobj, -1);
    NTSHELL_IO_GET(extobj);

    if (argc < 2) {
        cmd_led_test_help(argv[0], extobj);
        return E_OK;
    }

    opterr = 0;
    optind = 1;

    while ((opt=getopt(argc, argv, "o:c:r:mhH?")) != -1) {
        switch (opt) {
            case 'h':
            case '?':
            case 'H':
                cmd_led_test_help(argv[0], extobj);
```

c. The command help function “*cmd_led_test_help*” is used to display the help description of the “led_test” command.



```
/* show help of command */
static void cmd_led_test_help(char *cmd_name, void *extobj)
{
    NTSHELL_IO_PREDEF;

    VALID_EXTOBJ_NORTN(extobj);
    NTSHELL_IO_GET(extobj);

    if (cmd_name == NULL) {
        /* cmd_name not valid */
        return;
    }
    CMD_DEBUG("usage: %s \r\n"
    "-h/H/?     Show Help\r\n"
    "-o <Data>  the corresponding number LED is open\r\n"
    "-c <Data>  the corresponding number LED is closed\r\n", cmd_name);

error_exit:
    return;
}

static int cmd_led_test(int argc, char **argv, void *extobj)
```

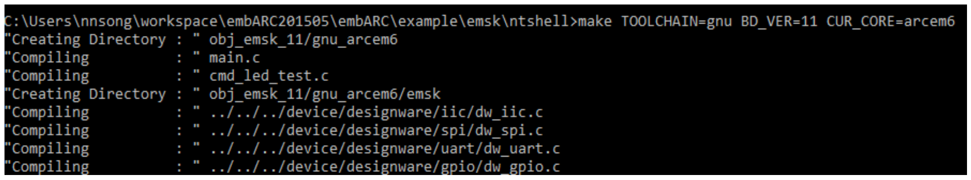
3) Edit the “*main*” function in main.c to register “*led_test*” command.



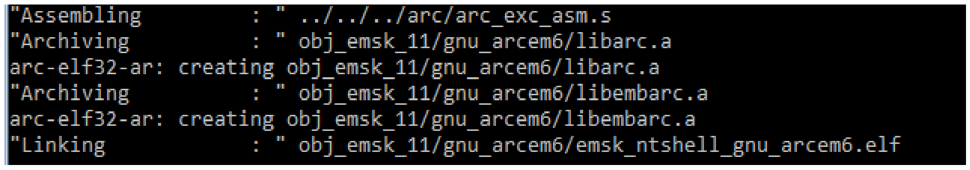
```
static NTSHELL_IO *nt_io;

extern CMD_TABLE_T * register_ntshell_cmd_led(CMD_TABLE_T *prev);
/**
 * \brief   call ntshell function and creat ntshell task
 */
int main(void)
{
    cpu_lock(); /* lock cpu to do initializations */

    board_init(); /* board level init */

    cpu_unlock();   /* unlock cpu to let interrupt work */

    nt_io = get_ntshell_io(BOARD_ONBOARD_NTSHELL_ID);
    register_ntshell_cmd_led(nt_io->cmd_tbl_head);
    /** enter ntshell command routine no return */
    ntshell_task((void *)nt_io);

    return E_SYS;   /* system error */
}
```

4) Compile and debug:

   a. Enter “***make TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6***” in command line to generate “**emsk_ntshell_gnu_arcem6.elf**”. *TOOLCHAIN*, BD_*VER* and *CUR_CORE* are the compiler parameter “*tool chain*”, “*board version*” and “*EMSK core*”.

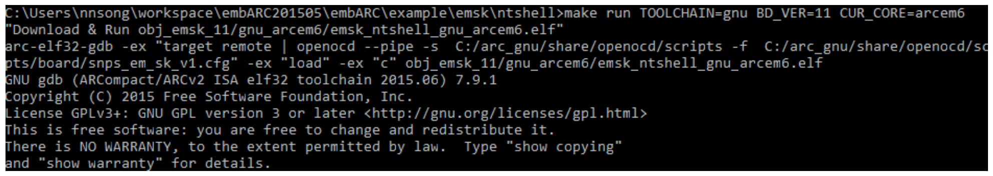


```
C:\Users\nnsong\workspace\embARC201505\embARC\example\emsk\ntshell>make TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6
"Creating Directory : " obj_emsk_11/gnu_arcem6
"Compiling          : " main.c
"Compiling          : " cmd_led_test.c
"Creating Directory : " obj_emsk_11/gnu_arcem6/emsk
"Compiling          : " ../../../device/designware/iic/dw_iic.c
"Compiling          : " ../../../device/designware/spi/dw_spi.c
"Compiling          : " ../../../device/designware/uart/dw_uart.c
"Compiling          : " ../../../device/designware/gpio/dw_gpio.c
```

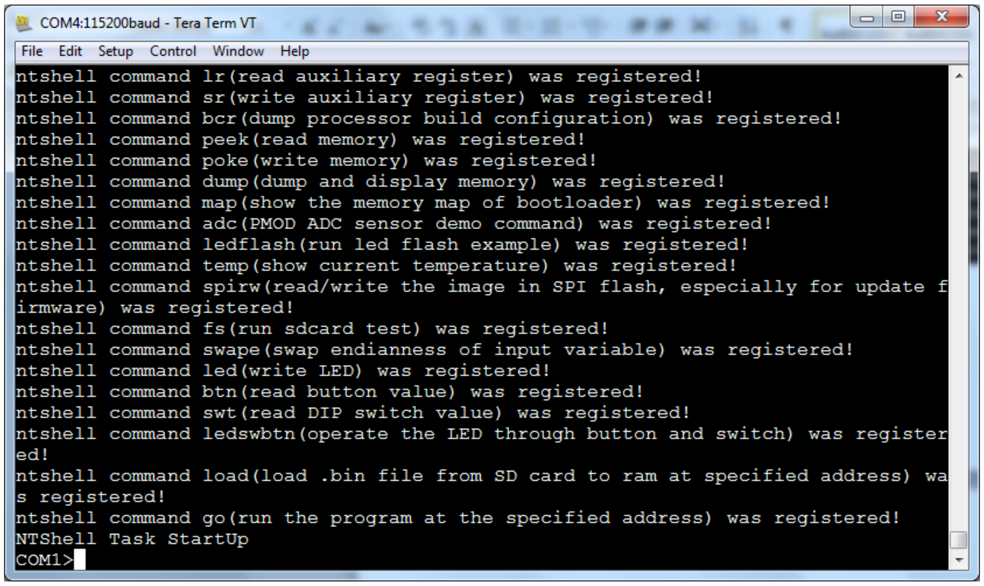


```
"Assembling         : " ../../../arc/arc_exc_asm.s
"Archiving          : " obj_emsk_11/gnu_arcem6/libarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libarc.a
"Archiving          : " obj_emsk_11/gnu_arcem6/libembarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libembarc.a
"Linking            : " obj_emsk_11/gnu_arcem6/emsk_ntshell_gnu_arcem6.elf
```

b. Enter “***make run TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6***” to run the NT-Shell application. The NT-Shell command line is shown in Tera Term.

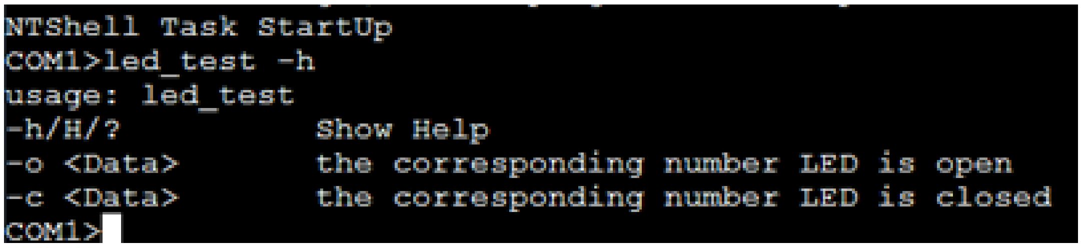


```
C:\Users\nnsong\workspace\embARC201505\embARC\example\emsk\ntshell>make run TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6
"Download & Run obj_emsk_11/gnu_arcem6/emsk_ntshell_gnu_arcem6.elf"
arc-elf32-gdb -ex "target remote | openocd --pipe -s  C:/arc_gnu/share/openocd/scripts -f  C:/arc_gnu/share/openocd/sc
pts/board/snps_em_sk_v1.cfg" -ex "load" -ex "c" obj_emsk_11/gnu_arcem6/emsk_ntshell_gnu_arcem6.elf
GNU gdb (ARCompact/ARCv2 ISA elf32 toolchain 2015.06) 7.9.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
```


COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

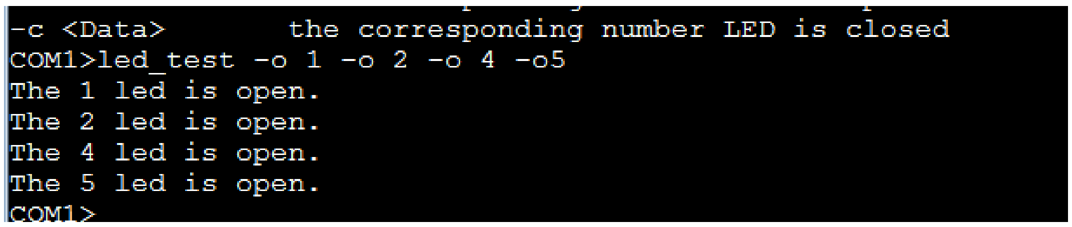
```
ntshell command lr(read auxiliary register) was registered!
ntshell command sr(write auxiliary register) was registered!
ntshell command bcr(dump processor build configuration) was registered!
ntshell command peek(read memory) was registered!
ntshell command poke(write memory) was registered!
ntshell command dump(dump and display memory) was registered!
ntshell command map(show the memory map of bootloader) was registered!
ntshell command adc(PMOD ADC sensor demo command) was registered!
ntshell command ledflash(run led flash example) was registered!
ntshell command temp(show current temperature) was registered!
ntshell command spirw(read/write the image in SPI flash, especially for update f
irmware) was registered!
ntshell command fs(run sdcard test) was registered!
ntshell command swape(swap endianness of input variable) was registered!
ntshell command led(write LED) was registered!
ntshell command btn(read button value) was registered!
ntshell command swt(read DIP switch value) was registered!
ntshell command ledswbtn(operate the LED through button and switch) was register
ed!
ntshell command load(load .bin file from SD card to ram at specified address) wa
s registered!
ntshell command go(run the program at the specified address) was registered!
NTShell Task StartUp
COM1>
```

5) Enter “***led_test -h***” in Tera Term.   The command help information from function “*cmd_led_test_help*” is displayed.

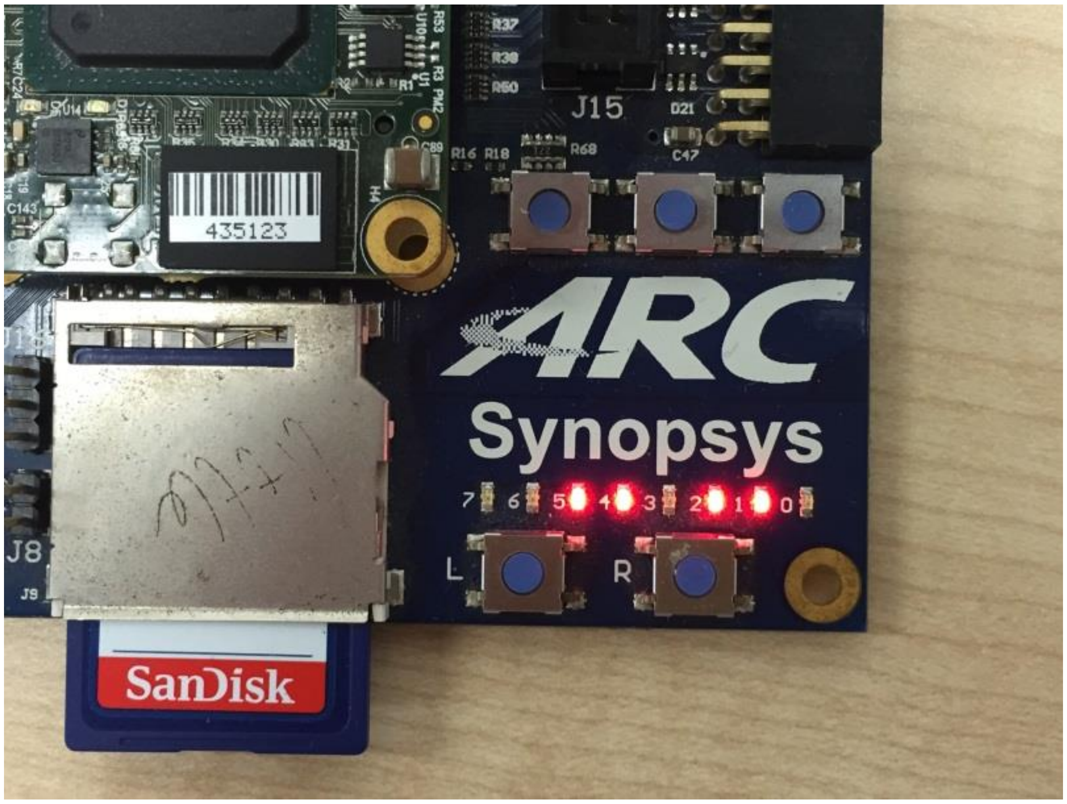


```
NTShell Task StartUp
COM1>led_test -h
usage: led_test
-h/H/?          Show Help
-o <Data>       the corresponding number LED is open
-c <Data>       the corresponding number LED is closed
COM1>
```

6) Enter “***led_test –o 1 –o 2 –o 4 –o 5***”. LEDs 1, 2, 4, 5 on the EMSK board will light up (on).



```
-c <Data>       the corresponding number LED is closed
COM1>led_test -o 1 -o 2 -o 4 -o5
The 1 led is open.
The 2 led is open.
The 4 led is open.
The 5 led is open.
COM1>
```



Similarly, entering command “***led_test –c 1 –c 2 –c 4 –c 5***” will turn off the LEDs.

Appendix A – cmd_led_test.c sources

"cmd_led_test.c"

```
/* ---------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------- */
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include "dev_uart.h"
#include "embARC.h"
#include "embARC_debug.h"

#include "board.h"

#ifndef USE_NTSHELL_EXTOBJ /* don't use ntshell extobj */
#define CMD_DEBUG(fmt, ...)            DBG(fmt, ##__VA_ARGS__)
#endif

/* show help of command */
static void cmd_led_test_help(char *cmd_name, void *extobj)
{
  NTSHELL_IO_PREDEF;

  VALID_EXTOBJ_NORTN(extobj);
  NTSHELL_IO_GET(extobj);

  if (cmd_name == NULL) {
    /* cmd_name not valid */
    return;
  }
  CMD_DEBUG("usage: %s \r\n"
        "-h/H/?         Show Help\r\n"
        "-o <Data>      the corresponding number LED is open\r\n"
        "-c <Data>      the corresponding number LED is closed\r\n", cmd_name);

error_exit:
  return;
}
```

```
static int cmd_led_test(int argc, char **argv, void *extobj)
{
	int ercd = E_OK;
	uint32_t temp;
	char *opt_data = NULL;
	int opt;

	NTSHELL_IO_PREDEF;

	VALID_EXTOBJ(extobj, -1);
	NTSHELL_IO_GET(extobj);

	if (argc < 2) {
		cmd_led_test_help(argv[0], extobj);
		return E_OK;
	}

	opterr = 0;
	optind = 1;

	while ((opt=getopt(argc, argv, "o:c:r:mhH?")) != -1) {
		switch (opt) {
			case 'h':
			case '?':
			case 'H':
				cmd_led_test_help(argv[0], extobj);
				goto error_exit;
				break;
			case 'o':
				temp=atoi(optarg);
				CMD_DEBUG("The %d led is open.\r\n", temp);
				temp = 0x01 << temp;
				led_write(temp, temp);
				break;
			case 'c':
				temp=atoi(optarg);
				CMD_DEBUG("The %d led is closed.\r\n", temp);
				temp = 0x01 << temp;
				led_write(0x00, temp);
				break;
			default:
				CMD_DEBUG("unrecognized option:%c\r\n", opt);
```

```
                                        break;
                        }
            }
            return E_OK;

error_exit:
            return ercd;
}

static CMD_TABLE_T led_test_cmd = {"led_test", "led test command", cmd_led_test, NULL};
/**
 * register ble_test command
 */
CMD_TABLE_T * register_ntshell_cmd_led(CMD_TABLE_T *prev)
{
            return ntshell_usrcmd_register(&led_test_cmd, prev);
}
```