M480 uIP HTTP Led

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	An example of HTTP server using uIP TCP/IP stack
BSP Version	M480 Series BSP CMSIS V3.03.001
Hardware	NuMaker-PFM-M487 Ver 3.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

1.1 Introduction

NuMicro® M480 series support 10/100M Ethernet controller. M480 BSP includes EMAC sample programs based on TCP/IP open-source projects. Sample program bsp\SampleCode\StdDriver\EMAC_lwiperf shows the example TCP/IP framework using lwIP, a lightweight TCP/IP stack. Sample program bsp\SampleCode\StdDriver\EMAC_uIP_httpd and EMAC_uIP_telnetd are HTTP daemon and Telnet daemon using uIP, a micro TCP/IP stack.

Using uIP TCP/IP stack, the example code EC_M480_uIP_HTTP_Led_V1.00 shows an HTTP server with a very simple web page. HTTP clients, for example IE or Chrome, can connect to M480 NuMaker board running this HTTP server code via hyperlink http://192.168.0.2. Once connected, the web browser should display a simple web page containing a single button. Users can click the button to turn on/off a LED on M480 NuMaker board.

1.2 Principle

This example code shows how to implement an HTTP server on Nuvoton M480 platform. Users can easily extend functionality of web server based on this framework.
This example uses fixed IP setting. According to the physical network environment, users may have to modify the IP address of local host and default router.

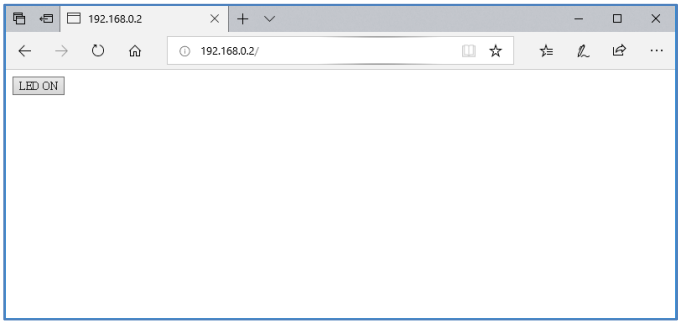
1.3 Demo Result

The following text is the UART message log of M480 NuMaker board running this example code.

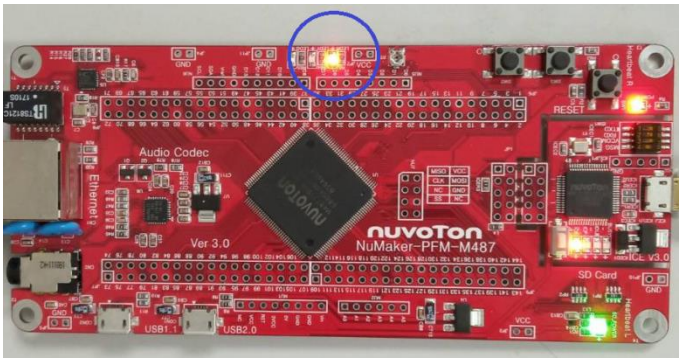
```
M480 uIP_HTTP_Led example code start
http server connected !
uip_appdata = GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134
Accept-Language: zh-TW
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
```

```
Accept-Encoding: gzip, deflate
Host: 192.168.0.2
Connection: Keep-Alive



index.html
HTTP server closed !
HTTP server connected !
uip_appdata = GET /html/open_led.html HTTP/1.1
Referer: http://192.168.0.2/
Cache-Control: max-age=0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134
Accept-Language: zh-TW
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Host: 192.168.0.2
Connection: Keep-Alive



open_led.html
HTTP server closed !
```

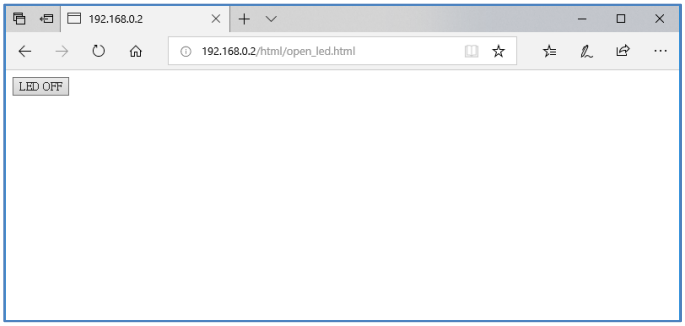
On connecting to M480 HTTP server, the client browser should display the following page:



Once user click the “LED ON” button, M480 HTTP server will turn on the LED.



The web page button text becomes “LED OFF:



Repeatedly, the user can click web page button to turn LED on/off.

2 Code Description

In main.c, the following code initialize EMAC multiple function pins and hardware.

```
    /* Enable EMAC engine clock */
    CLK_EnableModuleClock(EMAC_MODULE);

    // Configure MDC clock rate to HCLK/(127 + 1)=1.5 MHz if system is running at 192 MHz
    CLK_SetModuleClock(EMAC_MODULE, 0, CLK_CLKDIV3_EMAC(127));

    // Configure RMII pins
    SYS->GPA_MFPL |= SYS_GPA_MFPL_PA6MFP_EMAC_RMII_RXERR |
                     SYS_GPA_MFPL_PA7MFP_EMAC_RMII_CRSDV;
    SYS->GPC_MFPL |= SYS_GPC_MFPL_PC6MFP_EMAC_RMII_RXD1 |
                     SYS_GPC_MFPL_PC7MFP_EMAC_RMII_RXD0;
    SYS->GPC_MFPH |= SYS_GPC_MFPH_PC8MFP_EMAC_RMII_REFCLK;
    SYS->GPE_MFPH |= SYS_GPE_MFPH_PE8MFP_EMAC_RMII_MDC |
                     SYS_GPE_MFPH_PE9MFP_EMAC_RMII_MDIO |
                     SYS_GPE_MFPH_PE10MFP_EMAC_RMII_TXD0 |
                     SYS_GPE_MFPH_PE11MFP_EMAC_RMII_TXD1 |
                     SYS_GPE_MFPH_PE12MFP_EMAC_RMII_TXEN;

    // Enable high slew rate on all RMII TX output pins
    PE->SLEWCTL = (GPIO_SLEWCTL_HIGH << GPIO_SLEWCTL_HSREN10_Pos) |
                  (GPIO_SLEWCTL_HIGH << GPIO_SLEWCTL_HSREN11_Pos) |
                  (GPIO_SLEWCTL_HIGH << GPIO_SLEWCTL_HSREN12_Pos);


    // Select RMII interface by default
    EMAC_Open(ethaddr.addr);         // Initialize EMAC controller and auto-negotiate PHY
    NVIC_EnableIRQ(EMAC_TX_IRQn);
    NVIC_EnableIRQ(EMAC_RX_IRQn);
    EMAC_ENABLE_RX();
    EMAC_ENABLE_TX();
```

In main.c, the following code configures network environment and initialize uIP TCP/IP stack.

```
    /* set IP address local host */
    uip_ipaddr(ipaddr, 192, 168, 0, 2);
    uip_sethostaddr(ipaddr);
```

```
    /* set IP address of default router */
    uip_ipaddr(ipaddr, 192, 168, 0, 1);
    uip_setdraddr(ipaddr);

    /* set net mask */
    uip_ipaddr(ipaddr, 255, 255, 255, 1);
    uip_setnetmask(ipaddr);

    /* set MAC address */
    uip_setethaddr(ethaddr);

    /* initalize Http server and LEDs */
    http_led_init();
```

In http_led.c, the following code initialize HTTP server.

```
    /* HTML content of “LED ON” web page */
    static const char *close_led = "<html><body><form name='input'
action='/html/open_led.html' method='get'><input type='submit' value='LED ON'
/></form></body></html>";
    /* HTML content of “LED OFF” web page */
    static const char *open_led = "<html><body><form name='input'
action='/html/close_led.html' method='get'><input type='submit' value='LED OFF'
/></form></body></html>";


    /* Set multi-function pins PH.0 for LED and initially off */
    SYS->GPH_MFPL &= ~(SYS_GPH_MFPL_PH0MFP_Msk);
    GPIO_SetMode(PH, BIT0, GPIO_MODE_OUTPUT);
    PH0 = 1;                    /* turn off red LED         */
    /* set IP address of default router */
    uip_listen(HTONS(80));      /* Listen on HTTP port 80   */
```

In http_led.c, the following code service HTTP connections and requests.

```
void httpd_appcall(void)
{
    struct http_server_state  *hs;

    hs = (struct http_server_state *)(uip_conn->appstate);

    if (uip_connected())
    {
        printf("HTTP server connected !\n");
        hs->connect_flag = 1;
        hs->time_out = 0;
    }
    else if (uip_newdata())
    {
        printf("uip_appdata = %s\n", uip_appdata);

        if ((uip_len >= 5) && (strncmp((const char *)uip_appdata, "GET /", 5) == 0))
        {
            if (strncmp((const char *)uip_appdata, "GET /html/open_led.html",
                strlen("GET /html/open_led.html")) == 0)
            {
                printf("open_led.html \n");
                uip_send((volatile u8_t *)open_led, strlen(open_led));
                PH0 = 0;   // LED on
                hs->connect_flag = 2;
            }
            else if (strncmp((const char *)uip_appdata, "GET /html/close_led.html",
                      strlen("GET /html/close_led.html")) == 0)
            {
                printf("close_led.html \n");
                uip_send((volatile u8_t *)close_led, strlen(close_led));
                PH0 = 1;   // LED off
                hs->connect_flag = 2;
            }
            else if ((strncmp((char const *)uip_appdata, "GET /index.html", 15) == 0)
                    || (strncmp((char const *)uip_appdata, "GET / ", 6) == 0))
            {
                printf("index.html \n");
                uip_send((volatile u8_t *)close_led, strlen(close_led));
                PB10 = 1;  // LED off
```

```
                    hs->connect_flag = 2;
                }
            }
        }
        else if (uip_poll())
        {
            if (hs->connect_flag == 2)
            {
                uip_close();
                hs->connect_flag = 0;
                hs->time_out = 0;
            }
            else if (hs->connect_flag == 1)
            {
                if ((++hs->time_out) > 10)
                {
                    hs->connect_flag = 2;
                }
            }
        }
        else if (uip_closed())
        {
            printf("HTTP server closed !\n");
        }
    }
```

3 Software and Hardware Environment

Software Environment

- BSP version
  - M480 Series BSP CMSIS v3.03.001
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-FPM-M487
  - Ethernet jack (CN1)
- Diagram

M480		Ethernet jack (CN1)
PA.6	←	RMII_RXERR
PA.7	←	RMII_CRSDV
PC.6	←	RMII_RXD1
PC.7	←	RMII_RXD0
PC.8	←	RMII_REFCLK
PE.8	→	RMII_MDC
PE.9	↔	RMII_MDIO
PE.10	→	RMII_TXD0
PE.11	→	RMII_TXD1
PE.12	→	RMII_TXEN

4 Directory Information

- EC_M480_uIP_HTTP_Led_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code
  - ThirdParty
    - uip-0.9 — uIP is a very small implementation of the TCP/IP stack that is written by Adam Dunkels <adam@sics.se>. More information can be obtained from the uIP homepage at http://www.sics.se/~adam/uip/

5 How to Execute Example Code

1. Browsing into sample code folder by 4.Directory Information and double click uIP_HTTP_Led.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
June 17, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.