Get Body Temperature

Aydın Güler

1. PURPOSE

This project has been created for EEM 449 project. The subject is to implement a program that will measure body temperature. In this project, Code Composer Studio is used as working environment. GY-906-MLX90614ESF thermometer sensor is connected to EK-TM4C1294-XL launchpad.

2. THE ALGORİTHM

Multiple tasks are used in this project. In addition to these, semaphores and mailboxes were also used in this project to ensure synchronization. Some of the tasks are explained below.

`Void` **`getNTPTimeTask`**`(UArg arg0, UArg arg1):`

Semaphore0 is posted by `Timer_ISR` task to wake `getNTPTimeTask` up. At the beginning of this task, semaphore0 pended.

```
Semaphore_pend(semaphore0, BIOS_WAIT_FOREVER);
GPIO_write(Board_LED0, 1);
timeNTP(TIMEIP,OUTGOING_PORT,calculatedTime,strlen(calculatedTime));
GPIO_write(Board_LED0, 0);
```

`Void` **`Timer_ISR`**`(UArg arg1):`

Two different semaphores(`semaphore0`, `semaphore2`) are posted here. `Semaphore0` is pended by `updateDateTimeEverySecondTask` and `serverSocketTask`. `Semaphore2` is pended by `getNTPTimeTask`.

```
Void Timer_ISR(UArg arg1)
{
   Semaphore_post(semaphore2);
   Semaphore_post(semaphore0);
}
```

`Void` **`updateDateTimeEverySecondTask`**`(UArg arg1):`

In this task, there is a while loop. At the beginning of the loop, it waits a semaphore(`semaphore2`) that `Timer_ISR` will post in every 1 second. Then, raw data format is converted to a readable fashion. After that, the calculated time is increased by 1 in every second to keep it updated. Then, the updated time is posted with a mailbox(`mailbox1`). Later, this mailbox(`mailbox1`) is pended in several conditions by `serverSocketTask`.

```
Void updateDateTimeEverySecondTask(UArg arg1)
{
   while(1){
      Semaphore_pend(semaphore2, BIOS_WAIT_FOREVER);
      calculatedTime  = raw1900Time[0]*16777216 +
                        raw1900Time[1]*65536 +
                        raw1900Time[2]*256 +
                        raw1900Time[3];
      updatedNTPTime = calculatedTime + 10800 + ctr++;
      Mailbox_post(mailbox1, &updatedNTPTime, BIOS_NO_WAIT);
      System_printf("Date: %s", ctime(&updatedNTPTime));
      System_flush();
   }
}
```

`Void` **`getTempTask`**`(UArg arg0, UArg arg1):`

I2C sensor reading is done within this task. Device ID is defined here as a slave address. It reads the temperature data and posts this data with a mailbox(`mailbox0`) to the `serverSocketTask`. To activate this task, `Timer_ISR` must post the semaphore.

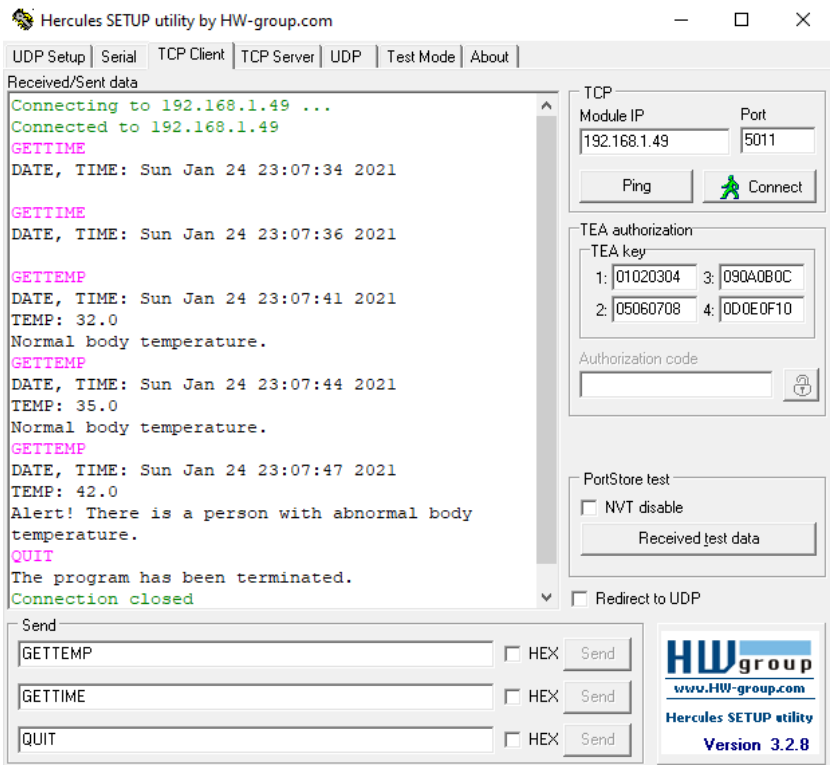
```
...
Semaphore_pend(semaphore1, BIOS_WAIT_FOREVER);
...
Mailbox_post(mailbox0, &temperature, BIOS_NO_WAIT);
...
```

```
Void serverSocketTask(UArg arg0, UArg arg1):
```

It contains lots of if conditions. In this conditions mailboxes and semaphores are used.

```
else if(!strcmp(buffer, "GETTEMP")) {
        uint8_t tempVal;
        uint8_t tempAverage = 0;
        uint16_t totalTemp=0;
        int updatedTime;
        int i;
        for (i = 0; i < 100; i++) {
          Semaphore_post(semaphore1);
          Mailbox_pend(mailbox0,&tempVal,BIOS_WAIT_FOREVER);
          totalTemp += tempVal;
        }
        tempAverage = totalTemp /100;
        sendThisTemp = (float)(tempAverage);
        Semaphore_post(semaphore2);
        Mailbox_pend(mailbox1,&updatedTime,
        BIOS_WAIT_FOREVER);
        Mailbox_pend(mailbox1,&updatedTime,BIOS_WAIT_FOREVE
        R);
        sprintf(outstr, " NOW: %s", ctime(&updatedTime));
        send(new_socket , outstr , strlen(outstr) , 0);
        sprintf(outstr, " TEMP: %2.1f\n", sendThisTemp);
        send(new_socket , outstr , strlen(outstr) , 0);
    }
```

Test:




```
ss in flash
Starting the HTTP GET example
System provider is set to SysMin. Halt the target to view any SysMin contents in ROV.
Service Status: DHCPC    : Enabled  :          : 000
Service Status: DHCPC    : Enabled  : Running  : 000
Network Added: If-1:192.168.1.47
Service Status: DHCPC    : Enabled  : Running  : 017
Date: Mon Jan  1 03:00:00 1900
32 bit raw data is obtained from timeNTP
Date: Sun Jan 24 15:26:41 2021
Date: Sun Jan 24 15:26:42 2021
Accepted connection
Date: Sun Jan 24 15:26:43 2021
message received: GETTIME
Date: Sun Jan 24 15:26:44 2021
Date: Sun Jan 24 15:26:45 2021
message received: GETTIME
Date: Sun Jan 24 15:26:46 2021
Date: Sun Jan 24 15:26:47 2021
Date: Sun Jan 24 15:26:48 2021
message received: GETTIME
Date: Sun Jan 24 15:26:49 2021
Date: Sun Jan 24 15:26:50 2021
Date: Sun Jan 24 15:26:51 2021
message received: GETTEMP
Date: Sun Jan 24 15:26:52 2021
Date: Sun Jan 24 15:26:53 2021
Date: Sun Jan 24 15:26:54 2021
message received: GETTEMP
Date: Sun Jan 24 15:26:55 2021
Date: Sun Jan 24 15:26:56 2021
message received: GETTEMP
Date: Sun Jan 24 15:26:57 2021
Date: Sun Jan 24 15:26:58 2021
message received: GETTEMP
Date: Sun Jan 24 15:26:59 2021
Date: Sun Jan 24 15:27:00 2021
message received: GETTEMP
Date: Sun Jan 24 15:27:01 2021
Date: Sun Jan 24 15:27:02 2021
Date: Sun Jan 24 15:27:03 2021
Date: Sun Jan 24 15:27:04 2021
message received: QUIT
```