

Real-time programming



Lab practice

Tasks

Task 1. Browse and look major FreeRTOS source files (port.c, task.c, etc.) @ https://github.com/feilipu/Arduino_FreeRTOS_Library/tree/master/src and in FreeRTOS documentation

Task 2. Explore and try Blink and AnalogRead example provided by FreeRTOS library in Arduino IDE.

Task 3. Modify previous task to printout button names.

Task 4. Explore and try IntegerQueue example provided by FreeRTOS library in Arduino IDE.

Task 5. Modify previous task to change the blink rate using queue. Rate is changed and continue to be changed during as long as the button is pressed. This task shows scenario when using a queue is not a good idea. Solve the task using global variable.

Task 6. Explore and try Task Status example provided by FreeRTOS library in Arduino IDE.

Task 7. Explore scheduler operation. Write program which contains two tasks which print one character each iteration (different characters are printed in different tasks, e.g. numbers and letters):

a) Task can be either in *Running* or *Ready* state

a.1) Control time spent in each iteration between print operation

a.2) Change priority of one task

b) Both task can enter *Block* state (100 ticks)

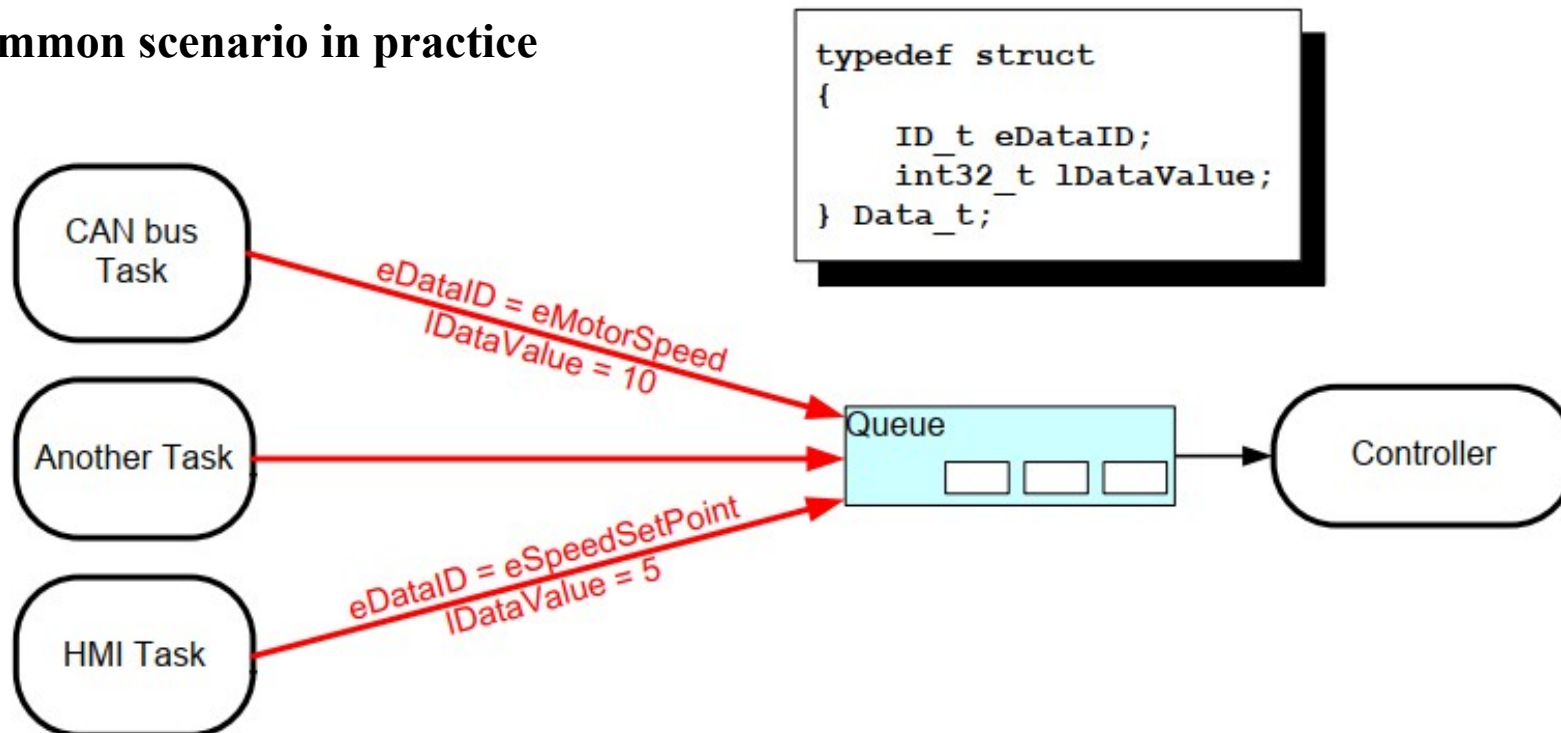
b.1) Implement Idle task hook where a variable is incremented

b.2) Suspend one task when variable reaches certain value (100.000) and resume it later (200.000)

Tasks

Task 8. Write a program which has 3 task executing concurrently. Two tasks are sender tasks. Sender 2 writes into the queue a randomly generated number from 1 to 100 every 200 ms and Sender 1 a number incremented every 500 ms. Third task is receiver task and reads data from queue and forwards it to the serial communication. When Receiver task reads data from queue it needs to know which sender task placed that data in queue and write information on serial port.

Common scenario in practice



Tasks

Task 9. Implement a mailbox. Write a program which contains two tasks, one sender task which writes into mailbox and the second receiver task which read from mail box. Data in the mailbox need to contain a time stamp (use ticks). Sender task updates mailbox every 1s with random number from 0 to 1000. Receiver task checks the mail box every 100 ms and prints out data and time stamp.

Task 10. Write a program that continuously toggles the state of the built-in LED using software timer. Toggling period can be set using Control task which reads data form the serial port.

Task 10a. Modify previous task to enable adjustment of the toggling duty cycle. Solution must use another software timer that operates in one shot mode for duty cycle adjustment

Tasks


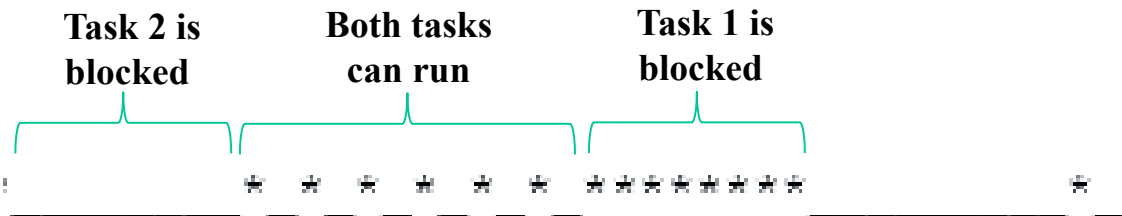


Task 11. Using FreeRTOS write a program that supports pin change interrupt. When interrupt occurs, programs needs to determine if button is pressed or released and button id if pressed (LEFT, RIGHT, UP, DOWN). When interrupt occurs processing should be deferred to Interrupt Handler Task. Once processing is completed information about interrupt cause (button status) should be placed in queue. Another task, Servicing Task, reads queue and displays information via serial port.

Task 12. Implement vSerialPrint() function, a 'thread safe' version of the Serial.print() using mutex. Using FreeRTOS write a program that has two task of equal priority which use serial port. In each iteration, one task is printing continuous string of 20 '_' characters ("_____"), and the other continuous string of 20 '*' characters ("*****"). Printing is done using baud rate of 600 bps in following way

- a) Using only Serial.print() function (no blocking state, nor any delays between function calls)
- b) Using Serial.print() function after which tasks are blocked for 10 ticks (vTaskDelay)
- c) As in b) except 'thread safe' vSerialPrint() is used instead of Serial.print()
- d) As in a) except 'thread safe' vSerialPrint() is used instead of Serial.print()

Notice: Clear serial monitor periodically to see currently printed characters

Task 12. Solution: Printed patterns

- a) Scheduler periodically swaps task (tasks are *Running* or *Ready*)
- 
- b)
- 
- c) Thread safe function ensures complete string is printed before resource is given to the other task
- 
- d) Task 2 is unable to get the resource (it starved out)
- 

LCD Key template

```
key=readKey();
if(key && !pressed)
{
    switch(key)
    {
        case SELECT:
            //...
            break;
        case LEFT:
            //...
            break;
        case UP:
            //...
            break;
        case DOWN:
            //...
            break;
        case RIGHT:
            //...
            break;
    }
    pressed=1;
}
else if(!key && pressed)
{
    pressed=0;
}
```

```
//Keys definitions
#define NONE      0
#define SELECT    1
#define LEFT      2
#define UP        3
#define DOWN      4
#define RIGHT     5

int readKey()
{
    int tmp = analogRead(0); //key pressed is read
    through
    //analog input 0
    if (tmp > 635 && tmp < 645) //SELECT
        return 1;
    if (tmp > 405 && tmp < 415) //LEFT
        return 2;
    if (tmp > 95 && tmp < 105) //UP
        return 3;
    if (tmp > 252 && tmp < 262) //DOWN
        return 4;
    if (tmp < 5) //RIGHT
        return 5;

    return 0; //no key is pressed
}
```