Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Real-Time Systems

Arduino+VsCode+PlatformIO+freeRTOS+Matlab environment configuration

Antonio Camacho Santiago
antonio.camacho.santiago@upc.edu

VsCode

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Download VsCode (zipped version is enough)

VsCodium is an open source alternative (not tested yet) without telemmetry

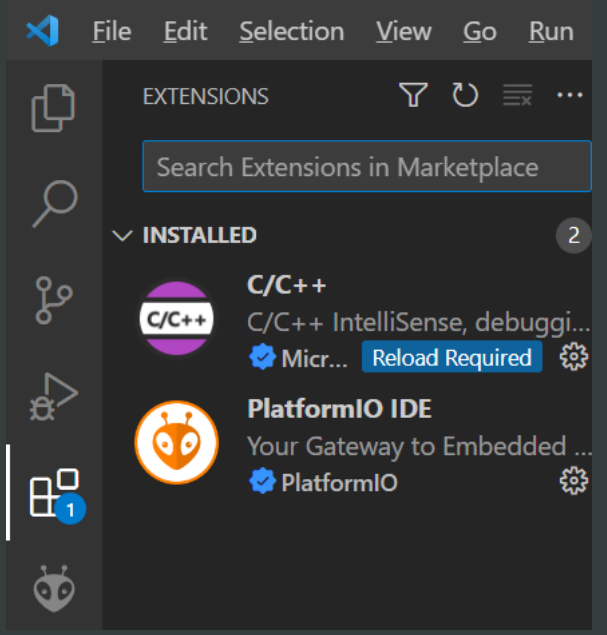
Go to menu Extensions icon on the left and install PlatformIO

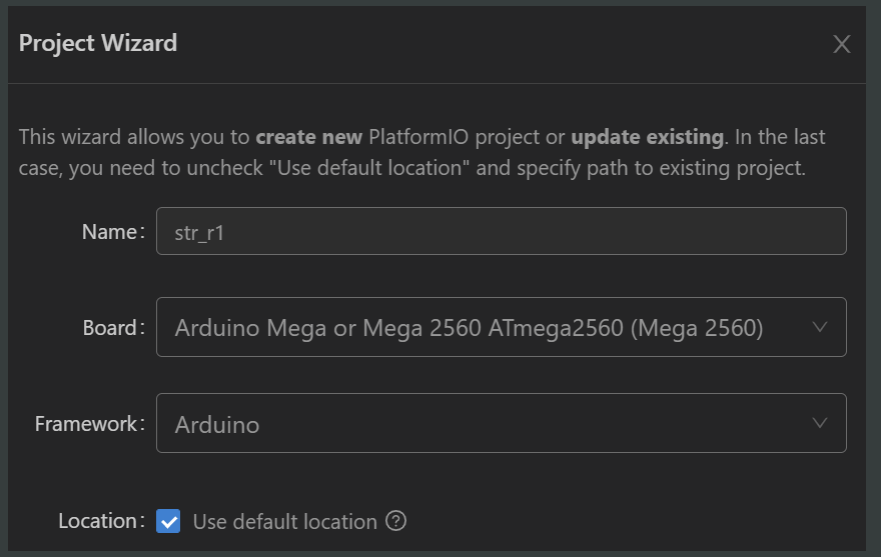
You can set an environmental variable as:

PLATFORMIO_CORE_DIR = c:\your_path_to_platformio if you want

Click on the platformio icon on the left

Click Open →New Project





Check platformio.ini file, here you can adjust your settings, add folders to compile path, etc.
For example, let's add the freeRTOS library, first unzip the Arduino_FreeRTOS_Library-master.zip into the project_folder\lib
Then, in the platformio.ini file, the following is required

```
build_flags =  -Ilib/Arduino_FreeRTOS_Library-master/src
```

Add `#include <Arduino.h>` on top of the main.cpp file
Create a Task to blink the `LED_BUILTIN` led
Create a Task to print the `xTaskGetTickCount()` over Serial

Compile (F1→PlatformIO: Build)
Download code (F1→PlatforomIO: Upload)

See the results in a serial terminal by adding `monitor_speed = 115200` to platformio.ini

Modify freeRTOS to trigger interrupts each 10ms based on Timer1 instead of Watchdog

In FreeRTOSVariant.h:

```
//STR
// #ifndef portUSE_WDTO
//     #define portUSE_WDTO        WDTO_15MS    // portUSE_WDTO to use the Watchdog Timer for xTaskIncrementTick
// #endif
#ifndef portUSE_TIMER1
    #define portUSE_TIMER1        TIMER1_10ms    // portUSE_WDTO to use the Watchdog Timer for xTaskIncrementTick
#endif

#if defined( portUSE_WDTO )
    #define configTICK_RATE_HZ  ( (TickType_t)( (uint32_t)128000 >> (portUSE_WDTO + 11) ) )  // 2^11 = 2048 WDT scaler for 128kHz
Timer
    #define portTICK_PERIOD_MS  ( (TickType_t) _BV( portUSE_WDTO + 4 ) )
#else
#if defined(portUSE_TIMER1)
    #warning "STR has defined TIMER1 as kernel tick @ 10ms, `configTICK_RATE_HZ` and `portTICK_PERIOD_MS` are defined here"
    #define configTICK_RATE_HZ  100
    #define portTICK_PERIOD_MS  ( (TickType_t) 1000 / configTICK_RATE_HZ )
#else
    #error "Variant configuration must define `configTICK_RATE_HZ` and `portTICK_PERIOD_MS` as either a macro or a constant"
    #define configTICK_RATE_HZ  1
    #define portTICK_PERIOD_MS  ( (TickType_t) 1000 / configTICK_RATE_HZ )
#endif
#endif
//STR
```

Modify freeRTOS to trigger interrupts each 10ms based on Timer1 instead of Watchdog

In port.c:

```
#if defined( portUSE_TIMER1 )

void prvSetupTimerInterrupt( void )
    {
        //From http://www.8bit-era.cz/arduino-timer-interrupts-calculator.html
        // TIMER 1 for interrupt frequency 100 Hz:
        cli(); // stop interrupts
        TCCR1A = 0; // set entire TCCR1A register to 0
        TCCR1B = 0; // same for TCCR1B
        TCNT1  = 0; // initialize counter value to 0
        // set compare match register for 100 Hz increments
        OCR1A = 19999; // = 16000000 / (8 * 100) - 1 (must be <65536)
        // turn on CTC mode
        TCCR1B |= (1 << WGM12);
        // Set CS12, CS11 and CS10 bits for 8 prescaler
        TCCR1B |= (0 << CS12) | (1 << CS11) | (0 << CS10);
        // enable timer compare interrupt
        TIMSK1 |= (1 << OCIE1A);
        sei(); // allow interrupts
    }

#else
    #warning "The user is responsible to provide function `prvSetupTimerInterrupt()`"
    extern void prvSetupTimerInterrupt( void );
#endif
```

Modify freeRTOS to trigger interrupts each 10ms based on Timer1 instead of Watchdog

In port.c:

```
void vPortEndScheduler( void )
{
    /* It is unlikely that the ATmega port will get stopped.  If required simply
     * disable the tick interrupt here. */

    //STR
    //wdt_disable();      /* disable Watchdog Timer */
    //disable timer1
    cli();
    TCCR1B = 0;
    //TIMSK1 |= (0 << OCIE1A); // deactivate timer's interrupt.
    TCCR1B &= ~(1<< CS12);  // turn off the clock altogether
    TCCR1B &= ~(1<< CS11);
    TCCR1B &= ~(1<< CS10);
    TIMSK1 &= ~(1 << OCIE1A); // turn off the timer interrupt
    //STR
}
```

Modify freeRTOS to trigger interrupts each 10ms based on Timer1 instead of Watchdog

In port.c:

```
//STR
#if defined( portUSE_WDTO ) || defined( portUSE_TIMER1 )
//STR
void vPortDelay( const uint32_t ms ) __attribute__ ( ( hot, flatten ) );
```

Add trace functionality:

In ArduinoFreeRTOS.h:

```
//STR
extern void str_trace(void);
#ifndef traceTASK_SWITCHED_IN
/* Called after a task has been selected to run.  pxCurrentTCB holds a pointer
 * to the task control block of the selected task. */
    //#define traceTASK_SWITCHED_IN()
    #define traceTASK_SWITCHED_IN() str_trace()
#endif
//STR
```

To remove undefined reference to `eTaskGetState‘ change define

In Arduino_FreeRTOS.h:

```
#ifndef INCLUDE_eTaskGetState
    //STR
    //#define INCLUDE_eTaskGetState 0
    #define INCLUDE_eTaskGetState 1
    //STR
#endif
```

In FreeRTOSConfig.h: (sometimes tracing fails when applying round robin for tasks with the same priority, avoid it by increasing the number of priorities)

```
#define configMAX_PRIORITIES 10 //STR 4
```

Arduino code

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Add trace functionality:

In main main.cpp, a handler for each task is required:

```
xTaskCreate(
    Task1
    ,  (const portCHAR *)"Task1"   // A name just for humans
    ,  128  // This stack size can be checked & adjusted by reading the Stack Highwater
    ,  NULL
    ,  3  // Priority, with 3 (configMAX_PRIORITIES - 1) being the highest, and 0 being the lowest.
    ,  &Task1Handle );

xTaskCreate(
    Task2
    ,  (const portCHAR *) "Task2"
    ,  128
    ,  NULL
    ,  2
    ,  &Task2Handle );
```

Arduino code

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Add `getTime()` to get custom time

In main.cpp:

```
//return time expired in milliseconds
float getTime(void)
{
//  float t=(float)0.5e-3*((float)OCR1A*xTaskGetTickCount()+TCNT1));//Sent time in milliseconds!!!
  float t=10.0*(float)xTaskGetTickCount())+0.0008*(float)(TCNT1*);//Sent time in milliseconds!!!
return t;
}
```

Arduino code

Add `str_compute()` to waste some time:

In main.cpp:

```
//compute is only used to waste time without using delays
void str_compute(unsigned int milliseconds)
{
  unsigned int i = 0;
  unsigned int imax = 0;
  imax = milliseconds * 92;
  volatile float dummy = 1;
  for (i = 0; i < imax; i++)
  {
    dummy = dummy * dummy;
  }
}
```

Arduino code

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Add handlers to tasks in order to be able to account for their state in `str_trace()`.

Handlers should be global to make them available for trace

In main.cpp:

```
//tasks handlers, required in last parameter of xTaskCreate when
accessing task info
TaskHandle_t Task1Handle;
TaskHandle_t Task2Handle;
TaskHandle_t Task3Handle;
TaskHandle_t Task4Handle;

// define two tasks for Blink & AnalogRead
void Task1( void *pvParameters );
void Task2( void *pvParameters );
void Task3( void *pvParameters );
void Task4( void *pvParameters );
```

Arduino code

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Create circular buffers

In main code.cpp:

```
//circular buffer for debugging
#define BUFF_SIZE 500
float t[BUFF_SIZE] = {};
byte circ_buffer1[BUFF_SIZE] = {};
byte circ_buffer2[BUFF_SIZE] = {};
byte circ_buffer3[BUFF_SIZE] = {};
byte circ_buffer4[BUFF_SIZE] = {};
float debug_data1[BUFF_SIZE] = {};
unsigned int circ_buffer_counter = 0;

void trace(void)
{
  circ_buffer_counter++;
  if (circ_buffer_counter >= BUFF_SIZE)
  {
    circ_buffer_counter = 0;
  }

  t[circ_buffer_counter] = getTime();//sent time in milliseconds
  circ_buffer1[circ_buffer_counter] = eTaskGetState(Task1Handle);
  circ_buffer2[circ_buffer_counter] = eTaskGetState(Task2Handle);
  circ_buffer3[circ_buffer_counter] = eTaskGetState(Task3Handle);
  circ_buffer4[circ_buffer_counter] = eTaskGetState(Task4Handle);
  debug_data1[circ_buffer_counter]=2.7;
}
```

Arduino code

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Create a timer to stop the kernel and send data to Matlab

In main code.cpp:

```
#include "src/timers.h"
```

```
//timer handlers
TimerHandle_t xOneShotTimer;
BaseType_t xOneShotStarted;
```

```
 xOneShotTimer = xTimerCreate("OneShotTimer",
pdMS_TO_TICKS( 1000 ) , pdFALSE, 0,
OneShotTimerCallback );
  xOneShotStarted = xTimerStart( xOneShotTimer, 0 );
```

```
void OneShotTimerCallback( TimerHandle_t xTimer )
{
  TickType_t xTimeNow;
  xTimeNow = xTaskGetTickCount();
  //oneshottimer_count++;
  //stop the kernel...
  vTaskSuspend(Task1Handle);
  vTaskSuspend(Task2Handle);
  vTaskSuspend(Task3Handle);
  vTaskSuspend(Task4Handle);
  vTaskSuspendAll();

  //...and sent data to the host PC
  unsigned int i;
  for (i = 0; i < BUFF_SIZE; i++)
  {
    Serial.print((float)t[i]);
    Serial.write((uint8_t)circ_buffer1[i]);
    Serial.write((uint8_t)circ_buffer2[i]);
    Serial.write((uint8_t)circ_buffer3[i]);
    Serial.write((uint8_t)circ_buffer4[i]);
    Serial.print((float)debug_data1[i]);
    Serial.println();
  }
  delay(50);
}
```

Matlab

Arduino+VsCode+PlatformIO+freeRTOS+Matlab

Create a Matlab script to get data from arduino

Launch the Matlab script rs232_r1.m