StructArray

src : wokwi

Purpose : the purpose of this demo is so we dont need to define a lot of variable with the type struct that we need for certain function.

Reason for os : the reason for os is so we dont need to define and create our own scheduler, thread model, mutex, etc.

Example source code :

```
/*
 * Example of a basic FreeRTOS queue
 * https://www.freertos.org/Embedded-RTOS-Queues.html
 * src : https://wokwi.com/projects/new/arduino-uno
 */

// Include Arduino FreeRTOS library
#include <Arduino_FreeRTOS.h>

// Include queue support
#include <queue.h>

// Define a Structure Array
struct Arduino{
  int pin[2];
  int ReadValue[2];
};

//Function Declaration
void Blink(void *pvParameters);
void POT(void *pvParameters);
void TaskSerial(void *pvParameters);

/*
 * Declaring a global variable of type QueueHandle_t
 *
 */
QueueHandle_t structArrayQueue;

void setup() {

  /**
   * Create a queue.
   * https://www.freertos.org/a00116.html
   */
structArrayQueue=xQueueCreate(10, //Queue length
                              sizeof(struct Arduino)); //Queue item size

if(structArrayQueue!=NULL){


  xTaskCreate(TaskBlink,  // Task function
```

```
              "Blink",// Task name
              128,// Stack size
              NULL,
              0,// Priority
              NULL);

 // Create other task that publish data in the queue if it was created.
  xTaskCreate(POT,// Task function
              "AnalogRead",// Task name
              128,  // Stack size
              NULL,
              2,// Priority
              NULL);

   // Create task that consumes the queue if it was created.
   xTaskCreate(TaskSerial,// Task function
              "PrintSerial",// A name just for humans
              128,// This stack size can be checked & adjusted by reading the Stack Highwater
              NULL,
              1, // Priority, with 3 (configMAX_PRIORITIES - 1) being the highest, and 0 being the lo
              NULL);
}
}

void loop() {}

/*
 * Blink task.
 * See Blink_AnalogRead example.
 */
void TaskBlink(void *pvParameters){
  (void) pvParameters;

  pinMode(13,OUTPUT);

  digitalWrite(13,LOW);

  for(;;)
  {
    digitalWrite(13,HIGH);
    vTaskDelay(250/portTICK_PERIOD_MS);
    digitalWrite(13,LOW);
    vTaskDelay(250/portTICK_PERIOD_MS);
  }
}

/**
 * Analog read task for Pin A0 and A1
 * Reads an analog input on pin 0 and pin 1
 * Send the readed value through the queue.
 * See Blink_AnalogRead example.
 */
```

```
void POT(void *pvParameters){
  (void) pvParameters;
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  for (;;){
  // Read the input on analog pin 0:
  struct Arduino currentVariable;
  currentVariable.pin[0]=0;
  currentVariable.pin[1]=1;
  currentVariable.ReadValue[0]=analogRead(A0);
  currentVariable.ReadValue[1]=analogRead(A1);

  /**
    * Post an item on a queue.
    * https://www.freertos.org/a00117.html
    */
  xQueueSend(structArrayQueue,&currentVariable,portMAX_DELAY);

  // One tick delay (15ms) in between reads for stability
  vTaskDelay(1);
  }
}

/**
 * Serial task.
 * Prints the received items from the queue to the serial monitor.
 */
void TaskSerial(void *pvParameters){
  (void) pvParameters;

  // Init Arduino serial
  Serial.begin(9600);

  // Wait for serial port to connect. Needed for native USB, on LEONARDO, MICRO, YUN, and other 32u4
  while (!Serial) {
    vTaskDelay(1)
  }

  for (;;){

    struct Arduino currentVariable;

   /**
     * Read an item from a queue.
     * https://www.freertos.org/a00118.html
     */
    if(xQueueReceive(structArrayQueue,&currentVariable,portMAX_DELAY) == pdPASS ){
      for(int i=0;i<2;i++){
      Serial.print("PIN:");
      Serial.println(currentVariable.pin[i]);
      Serial.print("value:");
      Serial.println(currentVariable.ReadValue[i]);
```

```
      }
    }
    vTaskDelay(500/portTICK_PERIOD_MS);
  }
}
```

- The new data structure is defined at this line with the member `pin` and `ReadValue` as an array of integer with the size of 2.

```
...

struct Arduino{
  int pin[2];
  int ReadValue[2];
};

...
```

- This line will create the handler for the queue

```
...

QueueHandle_t structArrayQueue;

...
```

- Queue will be created using `xQueueCreate` function that accept the len of the queue and the size of the stuff/data that will be put inside the queue.

```
...

structArrayQueue=xQueueCreate(10, sizeof(struct Arduino));

...
```

- The first function that will be launched with the thread is `TaskBlink` that accept pointer of void as the args.

```
...

void TaskBlink(void *pvParameters){
  (void) pvParameters;

  pinMode(13,OUTPUT);

  digitalWrite(13,LOW);
```

```
  for(;;)
  {
    digitalWrite(13,HIGH);
    vTaskDelay(250/portTICK_PERIOD_MS);
    digitalWrite(13,LOW);
    vTaskDelay(250/portTICK_PERIOD_MS);
  }
}
```

...

- `pinMode(13, OUTPUT)` make the 13th pin to be output
- `digitalWrite(13, LOW)` will give small electricity to pin 13
- `digitalWrite(13, LOW)` will give high electricity to pin 13

- The second function that will be launched with the second task is `POT`.

...

```
void POT(void *pvParameters){
  (void) pvParameters;
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
  for (;;){
  // Read the input on analog pin 0:
  struct Arduino currentVariable;
  currentVariable.pin[0]=0;
  currentVariable.pin[1]=1;
  currentVariable.ReadValue[0]=analogRead(A0);
  currentVariable.ReadValue[1]=analogRead(A1);

  /**
    * Post an item on a queue.
    * https://www.freertos.org/a00117.html
    */
  xQueueSend(structArrayQueue,&currentVariable,portMAX_DELAY);

  // One tick delay (15ms) in between reads for stability
  vTaskDelay(1);
  }
}
```

...

- `xQueueSend()` will accept the handler then the data and lastly the delay.
- The delay have a maximum amount of time the task should block waiting for space to become available on the queue, should it already be full. The call will return immediately if the queue is full and xTicksToWait is set to 0. The time is defined in tick periods so the constant portTICK_PERIOD_MS should be used to convert to real time if this is required.

- The last task that will be spawned is `TaskSerial`.

...

```
void TaskSerial(void *pvParameters){
  (void) pvParameters;

  // Init Arduino serial
  Serial.begin(9600);

  // Wait for serial port to connect. Needed for native USB, on LEONARDO, MICRO, YUN, and other 32u4
  while (!Serial) {
    vTaskDelay(1);
  }

  for (;;){

    struct Arduino currentVariable;

   /**
     * Read an item from a queue.
     * https://www.freertos.org/a00118.html
     */
    if(xQueueReceive(structArrayQueue,&currentVariable,portMAX_DELAY) == pdPASS ){
      for(int i=0;i<2;i++){
      Serial.print("PIN:");
      Serial.println(currentVariable.pin[i]);
      Serial.print("value:");
      Serial.println(currentVariable.ReadValue[i]);
      }
    }
    vTaskDelay(500/portTICK_PERIOD_MS);
  }
}
```

...

- `Serial.begin()` will init the lowest serial freq
- `xQueueReceive()` accept the handler then data and lastly the delay.
- the function will return `pdPASS` or `pdFAIL`.

- The other things that can be changed is how big the array will be. In the example it have the len of 2

...

```
#define n 10;
struct Arduino{
  int pin[n];
  int ReadValue[n];
};

...
```

- with that code the len of the array will be 10.

- About `xTaskCreate(function, readable function name, stack size, args to func, priority, handler for the task)`
  - From that example we can say that if the `xTaskCreate` priority args set to be bigger then it will have the most priority and get to run first
  - priority is `uint8_t` with the alias of `UBaseType_t`
  - so any unsigned integer 8 bits are possible to be an args for the `xTaskCreate` function.
  - when the priority is `0` it will be an idle task that will be only run when idle (no other work to do).
- About `vTaskDelay(how long the task will be delayed in tick)`
  - `vTaskDelay()` specifies a time at which the task wishes to unblock relative to the time at which `vTaskDelay()` is called. For example, specifying a block period of 100 ticks will cause the task to unblock 100 ticks after `vTaskDelay()` is called.