PSOC6 SumoBot

0.0.1

Generated by Doxygen 1.11.0

Chapter 1

Coding Guidelines

This documentation describes some of the coding guidelines I follow in this project. Some guidelines have good reasons while others are just based on my personal preference. Some are also enforced by the formatter (clang-formatter). The goals with them are to reduce common errors, get a more consistent code-base, and to avoid discussions about petty details.

1.1 Naming

- snake_case for everything including filenames except defines
  - E.g. i2c.c, uart.c
    ```
     C
    void this_is_a_function(void)
    {
        uint8_t this_is_a_variable;
    }
    ```
- UPPER_CASE for defines
  ```
   C
  #define THIS_IS_A_CONSTANT (0)
  ```
- One code module per header and implementation file
  - A module is something that makes sense as a single idea
  - e.g. uart.c/uart.h, i2c.c/i2c.h, tb6612fng.c/tb6612fng.h,line_sensor.c/line_sensor.h
- Prefix function names with module name
  ```
   C
  void uart_init(void);
  void uart_write(void);
  void i2c_init(void);
  uint8_t i2c_read(void);
  void led_set(led_e led, led_state state);
  ```

1.2 Indentation

- Four spaces
- No tabs

1.3 Comments

- Comment what's not obvious from just reading the code
- Focus on why over how and what
- Don't overdo it
- Prefer good variable and function names over comments
- Use // for single line comments and /∗ ∗/ for multi-line comments.

1.4 Asserts

Asserts are mainly for catching obvious mistakes during development. In some sense, they also serve as documentation.

- Use asserts for detecting invalid function arguments, return values, and invalid variable content.
- Don't use asserts to check values received from sensors and other devices

1.5 Defines

- Define all constant numbers and comment if it's unclear where they come from
- Always use parenthesis (even for single numbers) to avoid unexpected macro expansion
  ```
   C
  // This number ...
  #define CONSTANT_NUMBER (100)
  ```
- Use do { } while(0) for multi-statement macros to have them act more like functions, i.e. can be followed by a semicolon ; without confusion.
  ```
   C
  #define ASSERT(expression)                              \
      do {                                                \
          if (!(expression)) {                            \
              assert_handler();                           \
          }                                               \
      } while (0)
  ```
- Avoid define macros when possible since they are error-prone and less readable

1.6 Header files

- Use #include guards in every header file to avoid duplicated and recursive inclusions
- A brief comment describing the module at top of every header file
  ```
   C
  #ifndef UART_H
  // A UART driver for setting up and operating the UART peripheral. DMA...
  #endif // UART_H
  ```
- Include header files in order from local to global
  ```
   C
  #include "drivers/uart.h"
  #include "common/defines.h"
  #include "msp430.h"
  ```
- Limit includes in headers

1.7 Switch statements

- If switching on an enum value, handle all cases and skip the default case
- Without the default case, the compiler warning will force you to update the switch statements when a new enum value is added

```
 C
switch (io)
{
case IO_TEST_LED:
    // ...
    break;
case IO_UART_RX:
    // ...
    break;
case IO_UART_TX:
    // ...
    break;
}
```

1.8 if/else statements

- Always use brackets, even for single statement

1.9 Functions

- Functions that are only used internally inside a single translation unit (.c file) should be static
- Use void as parameter in function declarations without parameters, because in C, functions with empty paranthesis can be called with any number of parameters.

```
 C
// io.h
void io_init(void);
```

```
 C
// io.c
static void io_helper_function(void)
{
    // ...
}

void io_init(void)
{
    io_helper_function();
    // ...
}
```

- Pass structs by pointer where applicable to reduce stack usage

```
 C
struct
{
    io_dir_e dir;
    io_out_e out;
} io_config;

void io_configure(io_e io, struct io_config *config)
{
    // ...
}
```

1.10 Data types

1.10.1 Typedef

- Typedef function pointers to give them simpler names where appropriate
- Always typedef enums to avoid using the keyword enum before every enum value
- Always suffix typedefed enums with _e
- Enum values should be UPPER_CASE and prefixed with enum name
 C
```
typedef enum
{
    IO_TEST_LED,
    IO_UART_RX,
    IO_UART_TX,
} io_e;
```
- Don't typedef structs
- Don't suffix with _t (it's reserved by POSIX)

1.10.2 Fixed-width integers (stdint.h)

- Use fixed-width integers (uint8_t, uint16_t, int16_t, int32_t, etc.)
- It makes memory usage obvious and porting easier.

1.10.3 Const correctness

- All variables that shouldn't change should be const
- It protects against accidentally modifying a variable that shouldn't be modified.
- It makes the code more explicit and readable
 C
```
void io_configure(io_e io, const struct io_config *config)
{
    const struct io_config current_config = get_config(io);
    // ...
}
```

Chapter 2

System Details

This project aims to Sumobot project from Embedded System Project Series using PSoC™ 6 MCU.

View this README on GitHub.

2.1 Requirements

- Ubunut 20.04
- ModusToolboxSetupInstaller_1.0.0.468_linux_x64
- Programming language: C
- ModusToolboxProgtools_1.1.0.1126
- fw-loader-3.7.0.2317-kitprog3-package-2.60.0.1450-linux

2.2 Toolchaine used

- GNU Arm® embedded compiler v10.3-2021.10 (`GCC_ARM`) - Default value of `TOOLCHAIN`

2.3 HW Kit

- PSoC™ 62S2 Wi-Fi Bluetooth® pioneer kit (`CY8CKIT-062S2-4343W`)

2.4 Make

- Builds the PSOC6 Project

2.5 Make flash

- Programs the device with generate hex file

Chapter 3

Topic Index

3.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Topic Documentation

5.1 gpio.c

Functions

- void **gpio_init** ()
- void **LedTask** (void ∗arg)

Variables

- uint32_t **delay** = 0

5.1.1 Detailed Description

The GPIO Modules handles all the initialization of the IO Pins, configures P13.7 USER LED which toggles every second. This Led acts as the Heart Beat of the System

5.2 Sumobot

Macros

- #define **DWT_CTRL** 0xE0001000
- #define **DWT_CYCLE_COUNT** 0xE0001004

Functions

- void **system_clk_init** (void)
- void **vApplicationGetIdleTaskMemory** (StaticTask_t ∗∗ppxIdleTaskTCBBuffer, StackType_t ∗∗ppxIdle←TaskStackBuffer, uint32_t ∗pulIdleTaskStackSize)
- void vApplicationGetTimerTaskMemory (StaticTask_t ∗∗ppxTimerTaskTCBBuffer, StackType_t ∗∗ppxTimer←TaskStackBuffer, uint32_t ∗pulTimerTaskStackSize)

  This is to provide the memory that is used by the RTOS daemon/time task.
- int **main** (void)

Variables

- uint32_t **clkPerifreq**

5.2.1 Detailed Description

- Main file of the project.

5.2.2 Function Documentation

5.2.2.1 vApplicationGetTimerTaskMemory()

```
void vApplicationGetTimerTaskMemory (
            StaticTask_t ** ppxTimerTaskTCBBuffer,
            StackType_t ** ppxTimerTaskStackBuffer,
            uint32_t * pulTimerTaskStackSize)
```

This is to provide the memory that is used by the RTOS daemon/time task.

If configUSE_STATIC_ALLOCATION is set to 1, so the application must provide an implementation of vApplicationGetTimerTaskMemory() to provide the memory that is used by the RTOS daemon/time task.

5.3 uart.c

Functions

- void **uart_init** ()
- void **uart_transmit** ()
- void **UartTask** (void ∗arg)

Variables

- const char ∗ **string** = "HelloWorld : "
- cy_stc_scb_uart_context_t **uartContext**
- const cy_stc_scb_uart_config_t uartConfig
- uint8_t **count** = 0

5.3.1 Detailed Description

The UART Module handles the uart debug messages, which will be printed using SCB5

5.3.2 Variable Documentation

5.3.2.1 uartConfig

```
const cy_stc_scb_uart_config_t uartConfig
```

Initial value:

```
= {
    .uartMode = CY_SCB_UART_STANDARD,
    .enableMutliProcessorMode = false,
    .smartCardRetryOnNack = false,
    .irdaInvertRx = false,
    .irdaEnableLowPowerReceiver = false,
    .oversample = 12UL,
    .enableMsbFirst = false,
    .dataWidth = 8UL,
    .parity = CY_SCB_UART_PARITY_NONE,
    .stopBits = CY_SCB_UART_STOP_BITS_1,
    .enableInputFilter = false,
    .breakWidth = 11UL,
    .dropOnFrameError = false,
    .dropOnParityError = false,
    .receiverAddress = 0UL,
    .receiverAddressMask = 0UL,
    .acceptAddrInFifo = false,
    .enableCts = false,
    .ctsPolarity = CY_SCB_UART_ACTIVE_LOW,
    .rtsRxFifoLevel = 0UL,
    .rtsPolarity = CY_SCB_UART_ACTIVE_LOW,
    .rxFifoTriggerLevel = 0UL,
    .rxFifoIntEnableMask = 0UL,
    .txFifoTriggerLevel = 0UL,
    .txFifoIntEnableMask = 0UL,
}
```

Chapter 6

File Documentation

6.1 FreeRTOSConfig.h

```
/*
 * FreeRTOS Kernel V10.5.0
 * Copyright (C) 2021 Amazon.com, Inc. or its affiliates.  All Rights Reserved.
 * Copyright (C) 2019-2024 Cypress Semiconductor Corporation, or a subsidiary of
 * Cypress Semiconductor Corporation.  All Rights Reserved.
 *
 * Updated configuration to support CM4.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy of
 * this software and associated documentation files (the "Software"), to deal in
 * the Software without restriction, including without limitation the rights to
 * use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
 * the Software, and to permit persons to whom the Software is furnished to do so,
 * subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in all
 * copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
 * FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
 * COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
 * IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
 * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
 * https://www.FreeRTOS.org
 * https://github.com/FreeRTOS
 * http://www.cypress.com
 *
 */

#ifndef FREERTOS_CONFIG_H
#define FREERTOS_CONFIG_H

/*-----------------------------------------------------------
 * Application specific definitions.
 *
 * These definitions should be adjusted for your particular hardware and
 * application requirements.
 *
 * THESE PARAMETERS ARE DESCRIBED WITHIN THE 'CONFIGURATION' SECTION OF THE
 * FreeRTOS API DOCUMENTATION AVAILABLE ON THE FreeRTOS.org WEB SITE.
 *
 * See http://www.freertos.org/a00110.html.
 *----------------------------------------------------------*/

#include "cy_utils.h"

/* Get the low power configuration parameters from
 * the ModusToolbox Device Configurator GeneratedSource:
 * CY_CFG_PWR_SYS_IDLE_MODE     - System Idle Power Mode
 * CY_CFG_PWR_DEEPSLEEP_LATENCY - Deep Sleep Latency (ms)
 */

#define configUSE_PREEMPTION                    1
#define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
extern uint32_t SystemCoreClock;
#define configCPU_CLOCK_HZ                      SystemCoreClock
```

```
#define configTICK_RATE_HZ                       1000u
#define configMAX_PRIORITIES                     7
#define configMINIMAL_STACK_SIZE                 128
#define configMAX_TASK_NAME_LEN                  16
#define configUSE_16_BIT_TICKS                   0
#define configIDLE_SHOULD_YIELD                  1
#define configUSE_TASK_NOTIFICATIONS             1
#define configUSE_MUTEXES                        1
#define configUSE_RECURSIVE_MUTEXES              1
#define configUSE_COUNTING_SEMAPHORES            1
#define configQUEUE_REGISTRY_SIZE                10
#define configUSE_QUEUE_SETS                     0
#define configUSE_TIME_SLICING                   1
#define configENABLE_BACKWARD_COMPATIBILITY      0
#define configNUM_THREAD_LOCAL_STORAGE_POINTERS 5

/* Memory allocation related definitions. */
#define configSUPPORT_STATIC_ALLOCATION  1
#define configSUPPORT_DYNAMIC_ALLOCATION 1
#define configTOTAL_HEAP_SIZE            10240
#define configAPPLICATION_ALLOCATED_HEAP 0

/* Hook function related definitions. */
#define configUSE_IDLE_HOOK                0
#define configUSE_TICK_HOOK                0
#define configCHECK_FOR_STACK_OVERFLOW     2
#define configUSE_MALLOC_FAILED_HOOK       1
#define configUSE_DAEMON_TASK_STARTUP_HOOK 0

/* Run time and task stats gathering related definitions. */
#define configGENERATE_RUN_TIME_STATS        0
#define configUSE_TRACE_FACILITY             1
#define configUSE_STATS_FORMATTING_FUNCTIONS 0

/* Co-routine related definitions. */
#define configUSE_CO_ROUTINES           0
#define configMAX_CO_ROUTINE_PRIORITIES 1

/* Software timer related definitions. */
#define configUSE_TIMERS             1
#define configTIMER_TASK_PRIORITY    3
#define configTIMER_QUEUE_LENGTH     10
#define configTIMER_TASK_STACK_DEPTH (configMINIMAL_STACK_SIZE * 2)

/*
Interrupt nesting behavior configuration.
This is explained here: http://www.freertos.org/a00110.html

Priorities are controlled by two macros:
- configKERNEL_INTERRUPT_PRIORITY determines the priority of the RTOS daemon task
- configMAX_API_CALL_INTERRUPT_PRIORITY dictates the priority of ISRs that make API calls

Notes:
1. Interrupts that do not call API functions should be >= configKERNEL_INTERRUPT_PRIORITY
   and will nest.
2. Interrupts that call API functions must have priority between KERNEL_INTERRUPT_PRIORITY
   and MAX_API_CALL_INTERRUPT_PRIORITY (inclusive).
3. Interrupts running above MAX_API_CALL_INTERRUPT_PRIORITY are never delayed by the OS.
*/
/*
PSoC 6 __NVIC_PRIO_BITS = 3

0 (high)
1           MAX_API_CALL_INTERRUPT_PRIORITY 001xxxxx (0x3F)
2
3
4
5
6
7 (low)     KERNEL_INTERRUPT_PRIORITY       111xxxxx (0xFF)


CAT3 XMC devices __NVIC_PRIO_BITS = 6

0 (high)
1           MAX_API_CALL_INTERRUPT_PRIORITY 000001xx (0x07)
..
..
..
..
63 (low)    KERNEL_INTERRUPT_PRIORITY       111111xx (0xFF)

!!!! configMAX_SYSCALL_INTERRUPT_PRIORITY must not be set to zero !!!!
See http://www.FreeRTOS.org/RTOS-Cortex-M3-M4.html

*/

```

```
/* Put KERNEL_INTERRUPT_PRIORITY in top __NVIC_PRIO_BITS bits of CM4 register */
#define configKERNEL_INTERRUPT_PRIORITY 0xFF
/*
Put MAX_SYSCALL_INTERRUPT_PRIORITY in top __NVIC_PRIO_BITS bits of CM4 register
NOTE For IAR compiler make sure that changes of this macro is reflected in
file portable\TOOLCHAIN_IAR\COMPONENT_CM4\portasm.s in PendSV_Handler: routine
*/
#ifdef COMPONENT_CAT3
#define configMAX_SYSCALL_INTERRUPT_PRIORITY 0x07
#else
#define configMAX_SYSCALL_INTERRUPT_PRIORITY 0x3F
#endif
/* configMAX_API_CALL_INTERRUPT_PRIORITY is a new name for configMAX_SYSCALL_INTERRUPT_PRIORITY
 that is used by newer ports only. The two are equivalent. */
#define configMAX_API_CALL_INTERRUPT_PRIORITY configMAX_SYSCALL_INTERRUPT_PRIORITY

/* Set the following definitions to 1 to include the API function, or zero
to exclude the API function. */
#define INCLUDE_vTaskPrioritySet            1
#define INCLUDE_uxTaskPriorityGet           1
#define INCLUDE_vTaskDelete                 1
#define INCLUDE_vTaskSuspend                1
#define INCLUDE_xResumeFromISR              1
#define INCLUDE_vTaskDelayUntil             1
#define INCLUDE_vTaskDelay                  1
#define INCLUDE_xTaskGetSchedulerState      1
#define INCLUDE_xTaskGetCurrentTaskHandle   1
#define INCLUDE_uxTaskGetStackHighWaterMark 0
#define INCLUDE_xTaskGetIdleTaskHandle      1
#define INCLUDE_eTaskGetState               1
#define INCLUDE_xEventGroupSetBitFromISR    1
#define INCLUDE_xTimerPendFunctionCall      1
#define INCLUDE_xTaskAbortDelay             0
#define INCLUDE_xTaskGetHandle              0
#define INCLUDE_xTaskResumeFromISR          1
#define INCLUDE_pxTaskGetStackStart         1

#include "SEGGER_SYSVIEW_FreeRTOS.h"
/* Normal assert() semantics without relying on the provision of an assert.h
header file. */
#if defined(NDEBUG)
#define configASSERT(x) CY_UNUSED_PARAMETER(x)
#else
#define configASSERT(x)                                                                          \
    if ((x) == 0) {                                                                              \
        taskDISABLE_INTERRUPTS();                                                                \
        CY_HALT();                                                                               \
    }
#endif

/* Definitions that map the FreeRTOS port interrupt handlers to their CMSIS
standard names - or at least those used in the unmodified vector table. */
#define vPortSVCHandler     SVC_Handler
#define xPortPendSVHandler  PendSV_Handler
#define xPortSysTickHandler SysTick_Handler

/* Dynamic Memory Allocation Schemes */
#define HEAP_ALLOCATION_TYPE1 (1) /* heap_1.c*/
#define HEAP_ALLOCATION_TYPE2 (2) /* heap_2.c*/
#define HEAP_ALLOCATION_TYPE3 (3) /* heap_3.c*/
#define HEAP_ALLOCATION_TYPE4 (4) /* heap_4.c*/
#define HEAP_ALLOCATION_TYPE5 (5) /* heap_5.c*/
#define NO_HEAP_ALLOCATION    (0)

#define configHEAP_ALLOCATION_SCHEME (HEAP_ALLOCATION_TYPE3)

/* Check if the ModusToolbox Device Configurator Power personality parameter
 * "System Idle Power Mode" is set to either "CPU Sleep" or "System Deep Sleep".
 */
#if defined(CY_CFG_PWR_SYS_IDLE_MODE)                                                            \
    && ((CY_CFG_PWR_SYS_IDLE_MODE == CY_CFG_PWR_MODE_SLEEP)                                      \
        || (CY_CFG_PWR_SYS_IDLE_MODE == CY_CFG_PWR_MODE_DEEPSLEEP))

/* Enable low power tickless functionality. The RTOS abstraction library
 * provides the compatible implementation of the vApplicationSleep hook:
 * https://github.com/infineon/abstraction-rtos#freertos
 * The Low Power Assistant library provides additional portable configuration layer
 * for low-power features supported by the PSoC 6 devices:
 * https://github.com/infineon/lpa
 */
extern void vApplicationSleep(uint32_t xExpectedIdleTime);
#define portSUPPRESS_TICKS_AND_SLEEP(xIdleTime) vApplicationSleep(xIdleTime)
#define configUSE_TICKLESS_IDLE                 2

#else
#define configUSE_TICKLESS_IDLE 0
#endif
```

```

/* Deep Sleep Latency Configuration */
#if (CY_CFG_PWR_DEEPSLEEP_LATENCY > 0)
#define configEXPECTED_IDLE_TIME_BEFORE_SLEEP CY_CFG_PWR_DEEPSLEEP_LATENCY
#endif

/* Allocate newlib reeentrancy structures for each RTOS task.
 * The system behavior is toolchain-specific.
 *
 * GCC toolchain: the application must provide the implementation for the required
 * newlib hook functions: __malloc_lock, __malloc_unlock, __env_lock, __env_unlock.
 * FreeRTOS-compatible implementation is provided by the clib-support library:
 * https://github.com/infineon/clib-support
 *
 * ARM/IAR toolchains: the application must provide the reent.h header to adapt
 * FreeRTOS's configUSE_NEWLIB_REENTRANT to work with the toolchain-specific C library.
 * The compatible implementations are also provided by the clib-support library.
 */
#define configUSE_NEWLIB_REENTRANT 1

#endif /* FREERTOS_CONFIG_H */
```

6.2 gpio.h

```
#ifndef GPIO_H
#define GPIO_H

#include "cy_pdl.h"
#include <string.h>

#if PSOC62 || PSOCWB

#define USER_LED_PORT (GPIO_PRT13)
#define USER_LED_PIN  (P13_7_PIN)

#endif

#if PSOCS3

#define USER_LED_PORT (GPIO_PRT11)
#define USER_LED_PIN  (P11_1_PIN)

#endif

void gpio_init();
void gpio_toggle_user_led();

// FreeRTOS Task
void LedTask(void *arg);

#endif
```

6.3 main.h

```
#ifndef MAIN_H
#define MAIN_H

/*
 *
 *
 * Peripheral Modules
 *
 *
 */
#include "gpio.h"
#include "uart.h"

/*
 *
 * Free RTOS Header Files
 *
 */
#include "FreeRTOS.h"
#include "task.h"

#include "SEGGER_SYSVIEW.h"

#endif
```

6.4 uart.h

```
#ifndef UART_H
#define UART_H

#include "cy_pdl.h"

/* Assign pins for UART on SCB5: P5[0], P5[1] */
#if PSOC62 || PSOCWB
#define UART        SCB5
#define UART_CLK    PCLK_SCB5_CLOCK
#define UART_PORT   P5_0_PORT
#define UART_RX_NUM P5_0_NUM
#define UART_TX_NUM P5_1_NUM
#define SCB_UART_RX P5_0_SCB5_UART_RX
#define SCB_UART_TX P5_1_SCB5_UART_TX
#endif

#if PSOCS3
#define UART        SCB1
#define UART_CLK    PCLK_SCB1_CLOCK
#define UART_PORT   P10_0_PORT
#define UART_RX_NUM P10_0_NUM
#define UART_TX_NUM P10_1_NUM
#define SCB_UART_RX P10_0_SCB1_UART_RX
#define SCB_UART_TX P10_1_SCB1_UART_TX
#endif

/* Assign divider type and number for UART */
#define UART_CLK_DIV_TYPE   (CY_SYSCLK_DIV_16_BIT)
#define UART_CLK_DIV_NUMBER (0U)

void uart_init();
void uart_transmit();

void UartTask(void *arg);
#endif
```

6.5 src/gpio.c File Reference

```
#include "gpio.h"
#include "FreeRTOS.h"
#include "task.h"
```

Include dependency graph for gpio.c:

6.6 src/main.c File Reference

Redesign the Sumobot Project using PSOC6.

```
#include "cy_pdl.h"
#include "main.h"
#include "system_psoc6.h"
#include "cy_device.h"
```

Include dependency graph for main.c:

Macros

- #define **DWT_CTRL** 0xE0001000
- #define **DWT_CYCLE_COUNT** 0xE0001004

Functions

- void **system_clk_init** (void)
- void **vApplicationGetIdleTaskMemory** (StaticTask_t ∗∗ppxIdleTaskTCBBuffer, StackType_t ∗∗ppxIdle←↩ TaskStackBuffer, uint32_t ∗pulIdleTaskStackSize)
- void vApplicationGetTimerTaskMemory (StaticTask_t ∗∗ppxTimerTaskTCBBuffer, StackType_t ∗∗ppxTimer←↩ TaskStackBuffer, uint32_t ∗pulTimerTaskStackSize)

  This is to provide the memory that is used by the RTOS daemon/time task.
- int **main** (void)

Variables

- uint32_t **clkPerifreq**

6.6.1 Detailed Description

Redesign the Sumobot Project using PSOC6.

Author

Hari Udayakumar

Date

06-07-2024

6.7 src/uart.c File Reference

```
#include "uart.h"
#include "FreeRTOS.h"
#include "task.h"
```
Include dependency graph for uart.c:

Functions

- void **uart_init** ()
- void **uart_transmit** ()
- void **UartTask** (void ∗arg)

Variables

- const char ∗ **string** = "HelloWorld : "
- cy_stc_scb_uart_context_t **uartContext**
- const cy_stc_scb_uart_config_t uartConfig
- uint8_t **count** = 0

6.7.1 Detailed Description

Author

Hari Udayakumar

Date

06-07-2024

Index