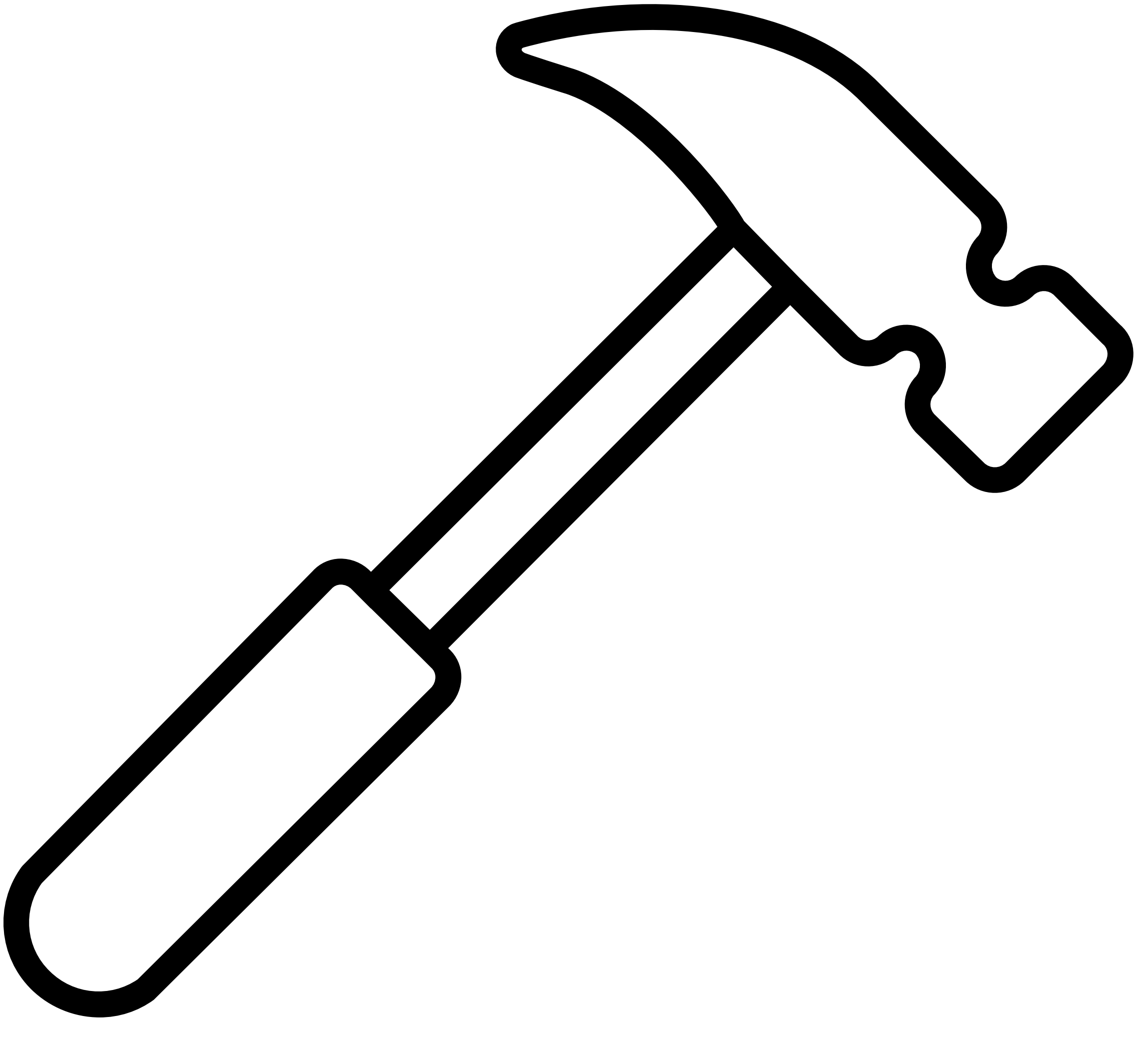
Maintainer Document

Richard Ojo



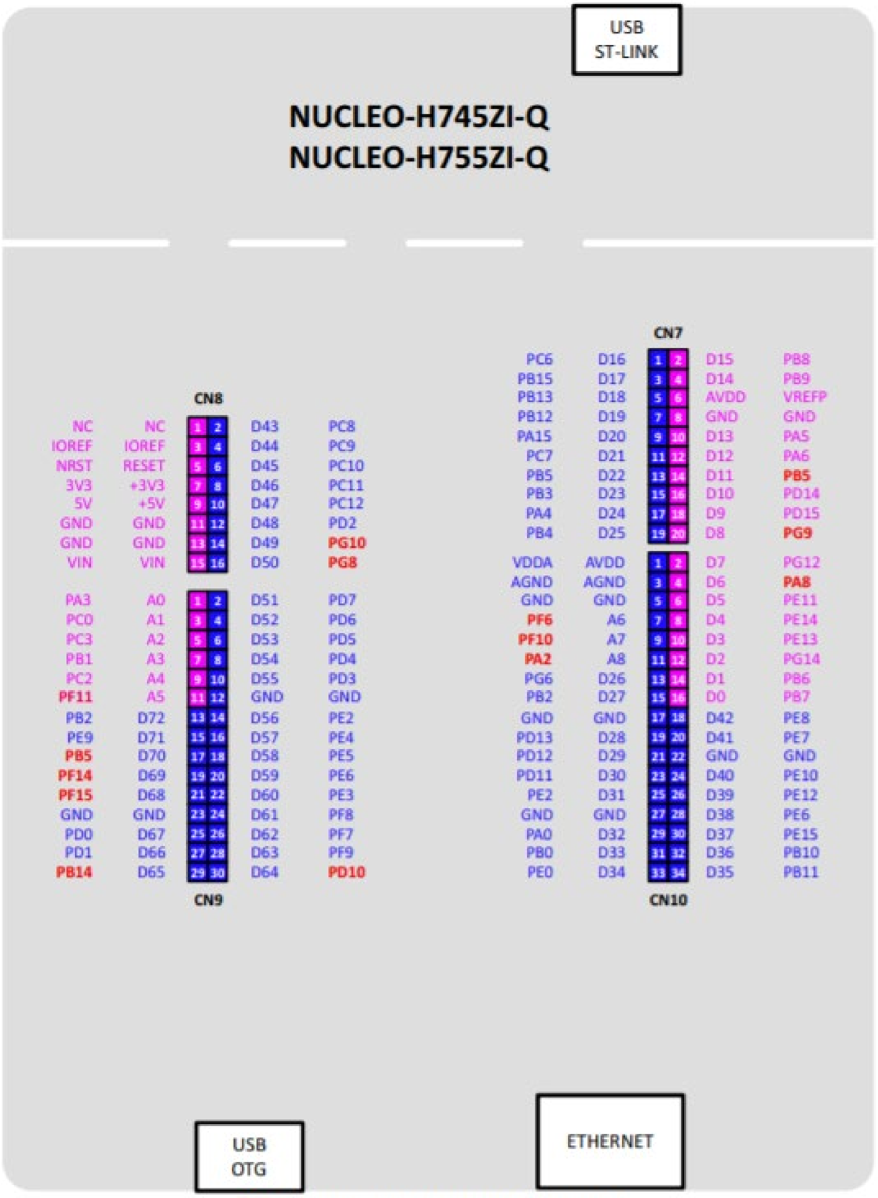
Abstract

This maintainers document is for maintaining the testy project and for those who want to modify or service the device. The project, 05c_testy_ws, is designed, programmed, and made for use with the STM32H745ZI-Q Nucleo-144 board. By sending commands to the H745 board through either USART or testterm (along with an accompanying script), different devices on the board can be initialized, configured, and/or modified for different purposes. These devices include TIM, SPI, GPIO, etc. Each device's code is contained within its corresponding file(s). These files contained within the project begin with "MoTdevice" and contains the associated device's name.

.

Table of Contents

STM32H745ZI-Q Pin Layout



CN8

NC	NC	1	2	D43	PC8
IOREF	IOREF	3	4	D44	PC9
NRST	RESET	5	6	D45	PC10
3V3	+3V3	7	8	D46	PC11
5V	+5V	9	10	D47	PC12
GND	GND	11	12	D48	PD2
GND	GND	13	14	D49	**PG10**
VIN	VIN	15	16	D50	**PG8**

PA3	A0	1	2	D51	PD7
PC0	A1	3	4	D52	PD6
PC3	A2	5	6	D53	PD5
PB1	A3	7	8	D54	PD4
PC2	A4	9	10	D55	PD3
PF11	A5	11	12	GND	GND
PB2	D72	13	14	D56	PE2
PE9	D71	15	16	D57	PE4
PB5	D70	17	18	D58	PE5
PF14	D69	19	20	D59	PE6
PF15	D68	21	22	D60	PE3
GND	GND	23	24	D61	PF8
PD0	D67	25	26	D62	PF7
PD1	D66	27	28	D63	PF9
PB14	D65	29	30	D64	**PD10**

CN9

CN7

PC6	D16	1	2	D15	PB8
PB15	D17	3	4	D14	PB9
PB13	D18	5	6	AVDD	VREFP
PB12	D19	7	8	GND	GND
PA15	D20	9	10	D13	PA5
PC7	D21	11	12	D12	PA6
PB5	D22	13	14	D11	**PB5**
PB3	D23	15	16	D10	PD14
PA4	D24	17	18	D9	PD15
PB4	D25	19	20	D8	**PG9**

VDDA	AVDD	1	2	D7	PG12
AGND	AGND	3	4	D6	**PA8**
GND	GND	5	6	D5	PE11
PF6	A6	7	8	D4	PE14
PF10	A7	9	10	D3	PE13
PA2	A8	11	12	D2	PG14
PG6	D26	13	14	D1	PB6
PB2	D27	15	16	D0	PB7
GND	GND	17	18	D42	PE8
PD13	D28	19	20	D41	PE7
PD12	D29	21	22	GND	GND
PD11	D30	23	24	D40	PE10
PE2	D31	25	26	D39	PE12
GND	GND	27	28	D38	PE6
PA0	D32	29	30	D37	PE15
PB0	D33	31	32	D36	PB10
PE0	D34	33	34	D35	PB11

CN10

PXX : differences compared with NUCLEO-H743ZI

🟪 Arduino subset of Zio = A0 to A5 and D0 to D15

🟦 Zio extension = A6 to A8 and D16 to D72

General Use – Useful Material

RM0399 Reference manual

UM2408 User manual

STM32H745xI/G Datasheet

General Use – MoT structure

Control Structure

XEQCMD holds address of the device's command handler

XEQTASK holds address of the device's task handler

NEXTTASK holds address of the device's successor device in linked-list task design

XEQC holds execution address of device task functions written in C (void-void functions to be dispatched by the XEQCMD)

SLEEPSAVE preserves XEQTASK address when the device's XEQTASK is replaced by the 'sleep()' task's execution address

TASKRESUME will hold return address of the task which called the sleep() task

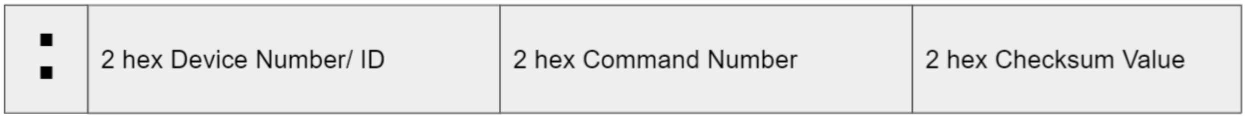
MSECWAKETIMELO will hold lsword of sleep()'s wakeup time (abs msecs)

MSECWAKETIMEHI will hold msword of sleep()'s task wakeup time (abs msecs)

General Use - Commands

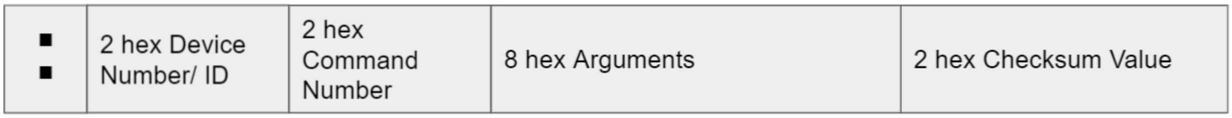
Commands sent to the testy device can be different sizes, but they all follow certain rules; or in other words they have a similar pattern.

Form 1: No Reload/ Count/ Arguments



:	2 hex Device Number/ ID	2 hex Command Number	2 hex Checksum Value

Form 2: With Reload/ Count/ Arguments



:	2 hex Device Number/ ID	2 hex Command Number	8 hex Arguments	2 hex Checksum Value

Every command begins with a colon /:.

Device Number: 0xYY, where YY is a number in hexadecimal. Specifies that device you want to interact with.

Command Number: 0xYY, where YY is a number in hexadecimal. Specifies the command that you want to run for that device.

<u>Optional</u>*

Arguments: 0xYYYYYYYY, where Y is a number in hexadecimal. Commands may or may not require arguments.

Checksum Value: Checksum = 0x100 – (Device # - Command # - Arguments)

General Use – Create Device

***For this example, we will create a device called FakeDev**

1. Create a corresponding MoTdevice class and lower-level section file. Example, "MoTdevice_ FakeDev_LL.S" and "MoTdevice_FakeDev.c".

2. In the "MoTdevice_ FakeDev_LL.S" file, Create 3 MoTdevice structures. Example, "MoTdevice FakeDev, FakeDev_cmdHandler, MoT_skipTask structures"

3. Add FakeDev device to "CM7_main12.c" file. Example, "extern deviceCTL_t TIM;"

4. Add FakeDev device to CM7_devicelist array.

5. Add commands to the FakeDev device. ***Refer to General Use – Modifications**

General Use – Modifications in C file

Most modifications for any device will occur in the corresponding MoTdevice class and lower-level section file. Example, "MoTdevice_TIM_LL.S" and "MoTdevice_TIM.c".

For example, to edit the TIM PWM's duty cycle, simply edit the CCR3 and ARR register for TIM3 in the TIM3_PWM function in "MoTdevice_TIM_LL.S."

General Use - Adding Commands in C file

To add a command to the TIM device, use the following steps:

1. Create the function you want to run in “MoTdevice_TIM_LL.S.”
2. Add the function declaration to “MoTdevice_TIM.c.” Example,
   void fakeFunc();
3. Add a task function declaration. This function should call the function. Example,
   void task_fakeFunc();
4. Add a start task function declaration. This should call the task function. Example,
   void start_fakeFunc();
5. Add the start function to the switch statement contained in the TIM_cmdHandler function. Example,
   case Y: start_fakeFunc (Cmdtail+1); break;
6. Write the fakeFunc(), task_fakeFunc(), and start_fakeFunc() functions.
7. Add a skiptask command to end that command

Sample function and corresponding start function:

```
void start_TIM3_PFM_Decreasetask(void *Cmdtail1)
{
        TIM3_reload = *(int32_t *)Cmdtail1;
        TIM3_count = *(int32_t *)Cmdtail1;
        MoT_linkTask( MoT_doCtask, TIM3_PFM_Decreasetask);
}

void TIM3_PFM_Decreasetask()
{
        TIM3_PFM_Decrease();
        MoT_postMsg(&Decrease_msg_pfm, &USART3_msglist);
}
```

General Use – MoT_Sleep

Function:

startSleep - called from within .S or .c tasks. sleeps the task for Nmsecs < 2**32.

*startSleep can be used inside MoTdevices to create scheduled functions/tasks/commands

General Use – MoT_linkTask

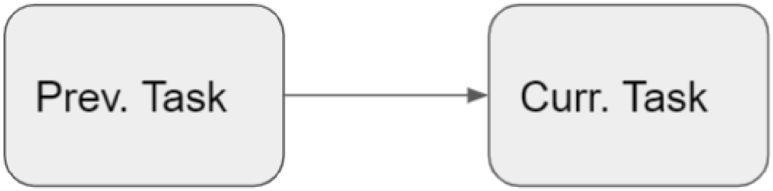
MoT_linkTask can be used to execute a C function/task after the current task.

Definition: void MoT_linkTask(void(*)(), void(*)());

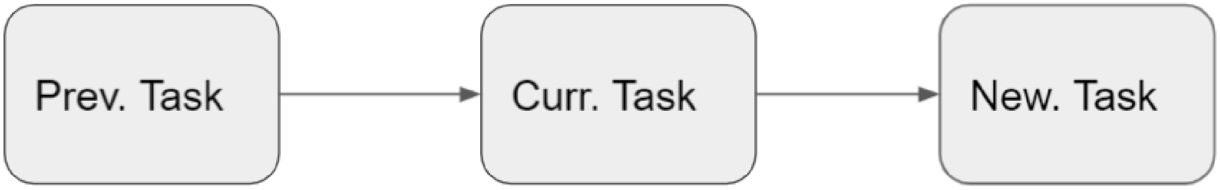
Example: MoT_linkTask(MoT_doCtask, CtaskDummy);

Where CtaskDummy is a valid MoT C task.

Before:



After:



General Use – Modifications in Section file

Most modifications for devices contained only in a section file will occur in the corresponding section file. Example, “MoTdevice_greenLED.S”

General Use - Adding Commands in Section file

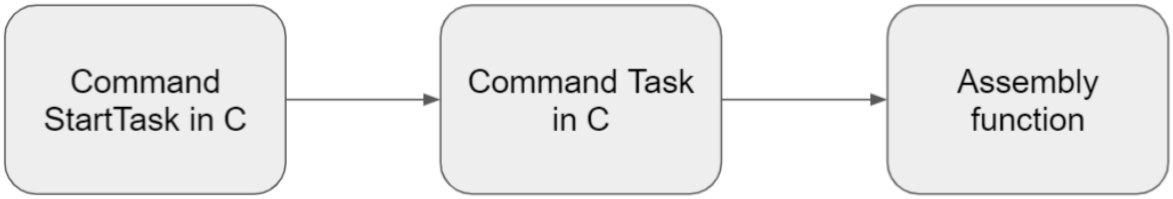
To add a command to the greenLED device, use the following steps:

1. Create the function you want to run in “MoTdevice_greenLED.S.”
2. Create a task that will call the function. Example,
   bl greenLED_ON

3. Create a start task that will call the task. Example,
   MOV_imm32 r0,greenLED_ONtask
   mov r1,#NULL
   bl MoT_linkTask
4. Add start task to fns_greenLED table.
   Example, .byte (fakeStartTask- fns_greenLED)/2

General Use - Task Flow

Tasks corresponding to a particular command for a particular device typically follow this flow/structure:



Zero Device and Nth Device

Device 0/ Nth

The Zero Device is always the device linked list’s head.

The Nth Device is always the device linked list’s tail.

'Nth' device (end task) will 'tail-recurse' back to calle

It is recommended to not alter either of these structures.

Green LED

Device 1

The greenLED device utilizes the stm32h745zi-q's greenLED. Corresponding structures, commands, and variables is contained in the "MoTdevice_greenLED.S" section file.

The greenLED device has 4 commands: initGPIOBbit0, start_ONtask, start_OFFtask, reportGPIOBbit0.

The stm32h745zi-q's greenLED is connected to GPIO PB0.

Commands

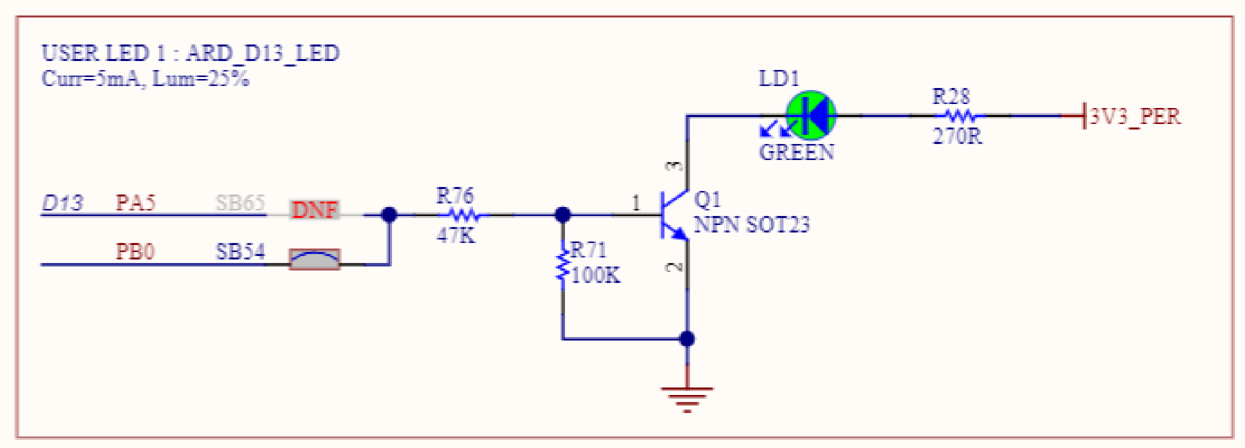
:0100FF initGPIOBbit0 - Configures greenLED (GPIO Output)

:0101FE start_ONtask – Turns greenLED On

:0102FD start_OFFtask – Turns greenLED Off

:0103FC reportGPIOBbit0 – Report whether greenLED is On or Off

:010400000000FB start_Disabletask - Turn off and Disable greenLED



Modifications

*Refer to **General Use – Modifications in Section file**

Adding commands

*Refer to **General Use - Adding Commands in Section file**

Red LED

Device 2

The redLED device utilizes the stm32h745zi-q's redLED. Corresponding structures, commands, and variables is contained in the "MoTdevice_redLED.c" C file and "MoTdevice_redLED.S" section file.

The redLED device has 3 commands: init_redLED, start_ONtask, start_OFFtask.

The stm32h745zi-q's redLED is connected to GPIO PB14.

Commands

:0200FE init_redLED - Configures redLED (GPIO Output)

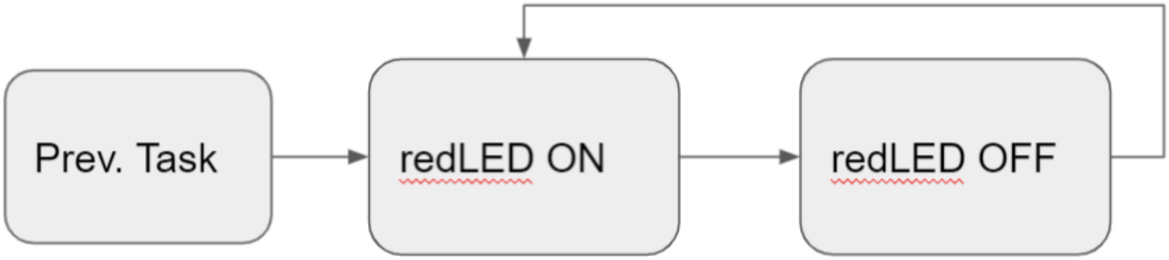
:0201XXXXXXXX YY start_ONtask – Repeatedly Turns redLED On then Off

:0202XXXXXXXX YY start_OFFtask – Repeatedly Turns redLED Off then On

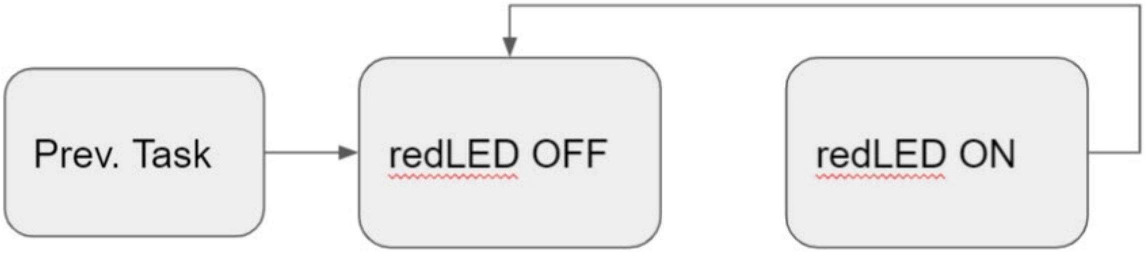
XXXXXXXX – Count value

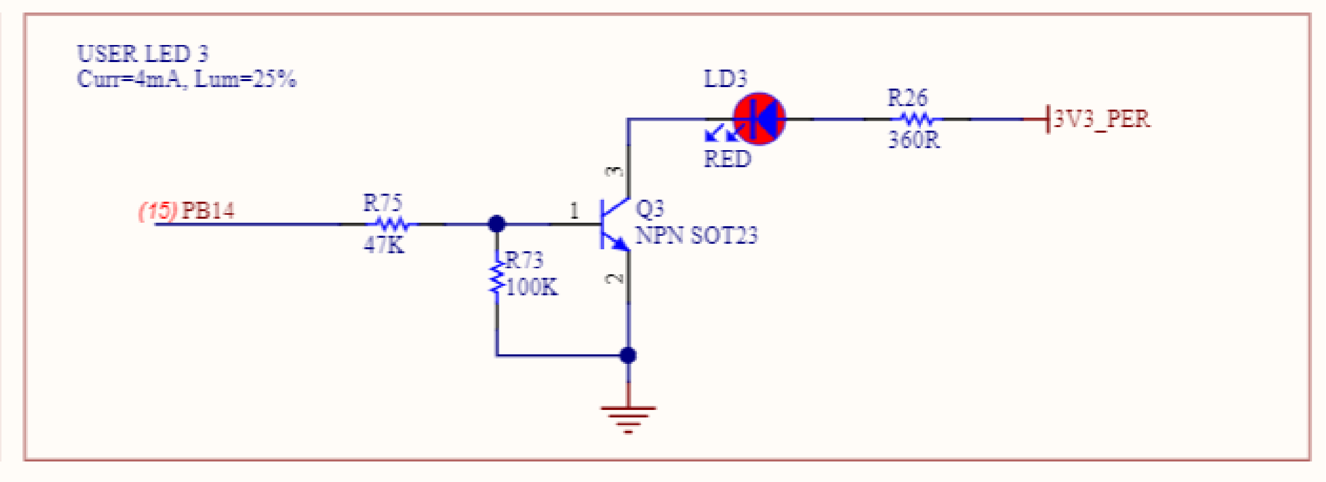
*Refer to **user's manual** for generating appropriate command based on count value.

When Count == 0, if ONtask then execute redLED_OFFtask task



When Count == 0, if OFFtask then execute redLED_ONtask task





Modifications

*Refer to **General Use – Modifications in C file**

Adding commands

*Refer to **General Use - Adding Commands in C file**

Blue Button

Device 3

The Blue Button device utilizes the stm32h745zi-q’s Blue/User Button. Corresponding structures, commands, and variables is contained in the “MoTdevice_blueBUTTON.S” section file.

The blueButton device has 4 commands: initGPIOCpin13, start_ONtask, start_OFFtask, reportGPIOCpin13.

The stm32h745zi-q’s greenLED is connected to GPIO PC13.

Commands

:0300FF initGPIOCpin13 - Initialize hardware and install blueBUTTON blink task on task list

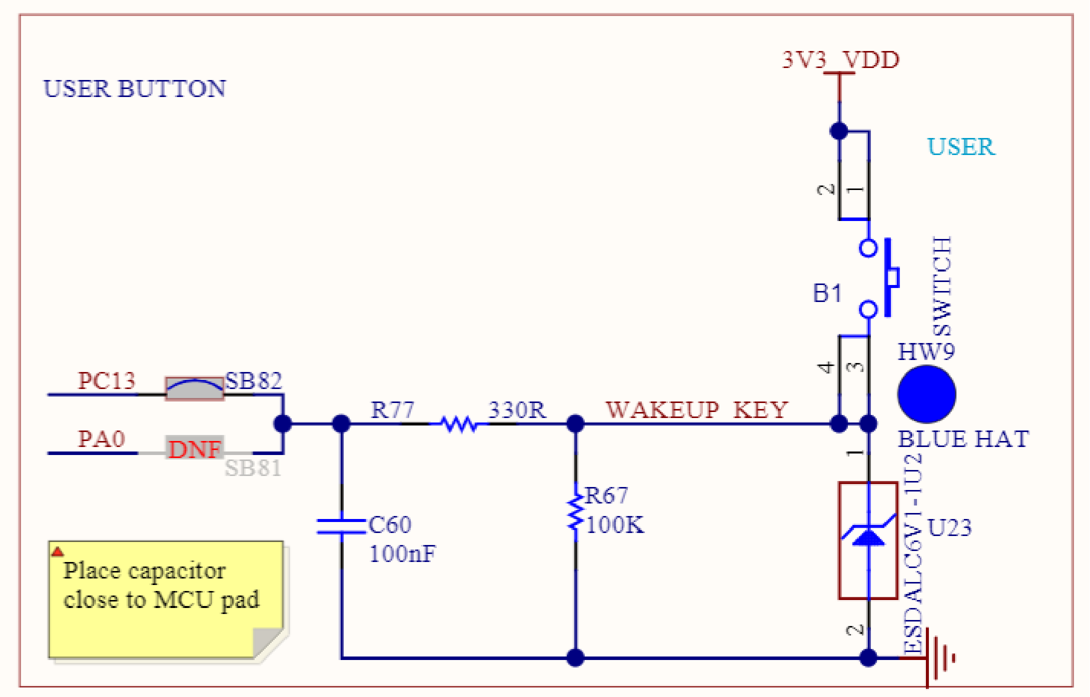
:0301XXXXXXXXYY start_ONtask – Set blink rate and start device_blueBUTTON_ONtask

:0302XXXXXXXXYY start_OFFtask – Set blink rate and start device_blueBUTTON_OFFtask

:0303FC reportGPIOCpin13– Report blueBUTTON state ('ON' or 'OFF')

YY - Checksum

XXXXXXXX – Count value or delay in milliseconds



Modifications

*Refer to **General Use – Modifications in Section file**

Adding commands

*Refer to **General Use - Adding Commands in Section file**

TIM

Device 4

The TIM device utilizes the STM32H745ZI-Q Nucleo-144 board's TIM3 and associated peripherals.

The TIM device's commands, functions, and variables are contained in the files "MoTdevice_TIM_LL.S" and "MoTdevice_TIM.c".

Commands

Pulse width modulation

:0400FC	TIM3_PWM - Initialize TIM3 to 50% duty cycle, ms resolution, 1s period
:0401FB	TIM3_PWM_Increase - increase duty cycle by 10%
:0402FA	TIM3_PWM_Decrease - decrease duty cycle by 10%

Pulse frequency modulation

:0403F9	TIM3_PFM - Initialize TIM3 to 50% duty cycle, ms resolution, 1s period
:0404F8	TIM3_PFM_Increase - increase freq by 10%
:0405F7	TIM3_PFM_Decrease - decrease freq by 10%

greenLED

:0406F6	TIM3_greenLED_PWM_PFM - TIM3 PWM/PFM is input to greenLED
:0407F5	start_TIM3_Disabletask – Disable TIM device and greenLED

*Must initialize TIM3_PWM or TIM3_PFM before using the corresponding increase and decrease tasks.

*Must initialize TIM3_PWM or TIM3_PFM before using the corresponding greenLED tasks.

Modifications

*Refer to **General Use – Modifications in C file**

Adding commands

*Refer to **General Use - Adding Commands in C file**

GPIO

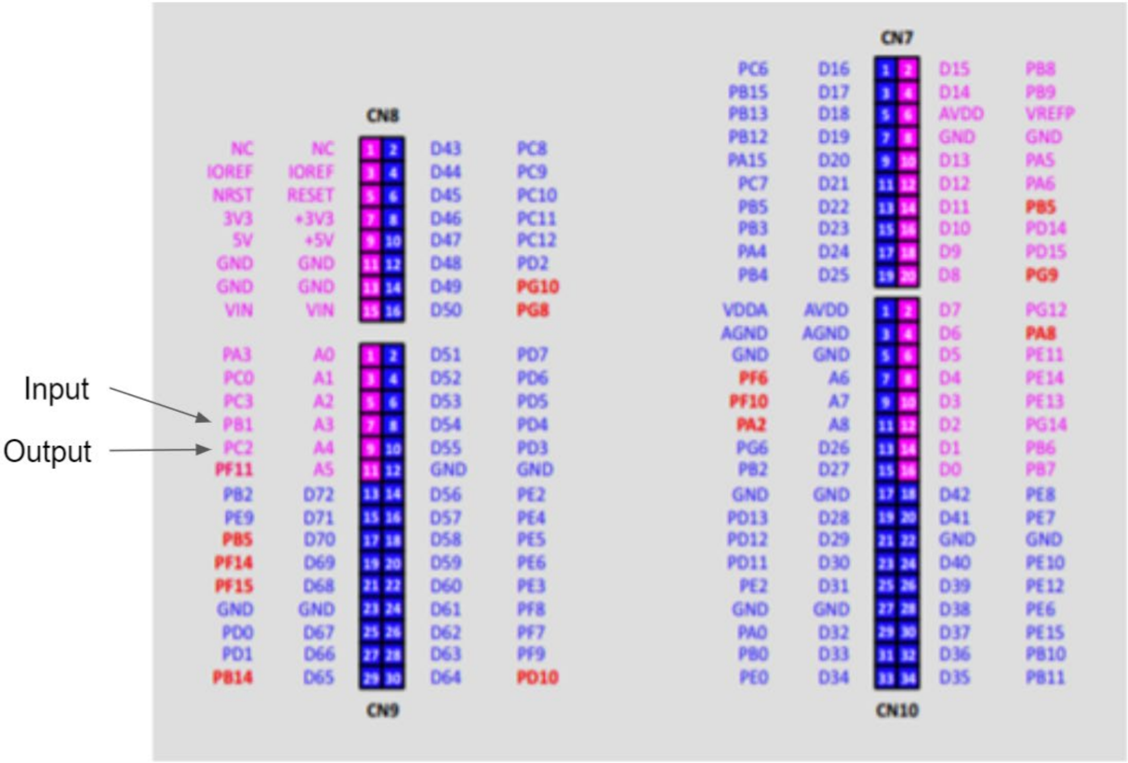
Device 5

The GPIO device utilizes the STM32H745ZI-Q Nucleo-144 board's GPIOs and associated peripherals.

The SPI device's commands, functions, and variables are contained in the files "MoTdevice_GPIO_LL.S" and "MoTdevice_GPIO.c".

Input GPIO: PB1

Output GPIO: PC2



Commands

:0500FB	init_GPIO_Input - Initialize PB1 as GPIO input
:0501FA	init_GPIO_Output - Initialize PC2 as GPIO output
:0502F9	GPIO_output_set - Set PC2 to 1

:0503F8	GPIO_output_reset - Set PC2 to 0
:0504F7	GPIO_input_read - Read value from PB1 IDR
:0505XXXXXXXXYY	GPIO_Input_repetitive - Read value from PB1 IDR continuously
:0506XXXXXXXXYY	GPIO_Output_repetitive - Output 1 and 0 on PC2 continuously
:0507XXXXXXXXYY	GPIO_Input_scheduled - Read Input PB1 after x milliseconds
:0508XXXXXXXXYY	GPIO_Output_scheduled - Output 1 on output PC2 after x milliseconds
:0509F2	GPIO_Off_task - Disable both GPIO input and GPIO output

YY - Checksum

XXXXXXXX – Count value or delay in milliseconds

*Refer to **user's manual** for generating appropriate command based on arguments.

Polling loop

Establishing a connection between GPIO Input (PB1) and GPIO Output (PC2) can be done with a jumper/Dupont wire to test GPIO functionality.

1. Read from GPIO input
2. Set GPIO output
3. Read from GPIO input
4. Reset GPIO output
5. Repeat

Using different GPIOs

To utilize a different port as input, change the code for init_GPIO_Input, which is contained in the lower-level file.

To utilize a different port as output, change the code for init_GPIO_Output, which is contained in the lower-level file.

Modifications

*Refer to **General Use – Modifications in C file**

Adding commands

*Refer to **General Use - Adding Commands in C file**

SPI

Device 6

The SPI device utilizes the STM32H745ZI-Q Nucleo-144 board's SPI1 (Master), SPI3(slave) and associated peripherals.

The SPI device's commands, functions, and variables are contained in the files "MoTdevice_SPI_LL.S", "MoTdevice_SPI_M_LL.S", "MoTdevice_SPI_S_LL.S", and "MoTdevice_SPI.c".

MoTdevice_SPI_M_LL.S

Contains functions and information for initializing, sending data, and receiving data for the SPI master.

MoTdevice_SPI_S_LL.S

Contains functions and information for initializing, sending data, and receiving data for the SPI slave.

<u>SPI ports/pins</u>

SPI Master - SPI1

SPI1_NSS - PA4

SPI1_SCK - PA5

SPI1_MISO - PA6

SPI1_MOSI - PB5

SPI Slave – SPI3

SPI3_NSS - PA15

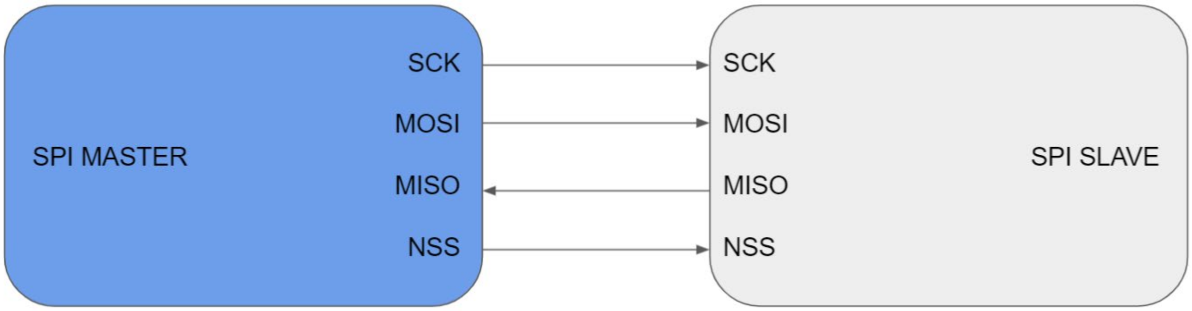
SPI3_SCK - PC10

SPI3_MISO - PC11

SPI3_MOSI - PC12

*Reference **Pin layout** for locating GPIO pins

SPI is initialized in FULL DUPLEX MODE, with 32bit data, and single data packet transfers.



Commands

:0600FA	init_SPItask	initializes SPI1 as master, SPI3 as slave, Full Duplex 32bit transfers
:0601XXXXXXXXYY	MasterTxtask	Send word from Master MOSI to Slave MOSI
:0602F8	MasterRxtask	Receive word on MISO
:0603XXXXXXXXYY	SlaveTxtask	Send word from Slave MISO to Master MISO
:0604F6	SlaveRxtask	Receive word on MOSI
:0605XXXXXXXXYY	MasterTxtask_polled	Continuous Send word from Master MOSI to Slave MOSI
:0606F4	MasterRxtask_polled	Continuous Receive word on MISO
:0607XXXXXXXXYY	SlaveTxtask_polled	Continuous Send word from Slave MISO to Master MISO
:0608F2	SlaveRxtask_polled	Continuous Receive word on MOSI
:0609F1	Disable_SPI	Disable SPI Master and Slave

YY - Checksum

XXXXXXXX – 32-bit Data value

To initialize the SPI master, four GPIOs must be configured appropriately for that SPI. The GPIOs should be in the appropriate alternate function mode.

To initialize the SPI slave, four GPIOs must be configured appropriately for that SPI. The GPIOs should be in the appropriate alternate function mode.

*To use the SPI device, it must be initialized first (Command 0 or :0600XX).

*Altering the data size can be done in MoTdevice_SPI_M_LL.S Section file in the init function.

*Altering the clock can be done in MoTdevice_SPI_M_LL.S Section file in the init function.

*Altering the fifo threshold level can be done in MoTdevice_SPI_M_LL.S Section file in the init function.

*To send data from slave to master using MISO, the master and slave TX fifos must be populated with data first. Then a clock signal is initiated by a transfer from the master.

*The SPI can be configured with one master and multiple slaves.

Modifications

*Refer to **General Use – Modifications in C file**

Adding commands

*Refer to **General Use - Adding Commands in C file**