

STM32C0316-DK Demo Brisk application subset

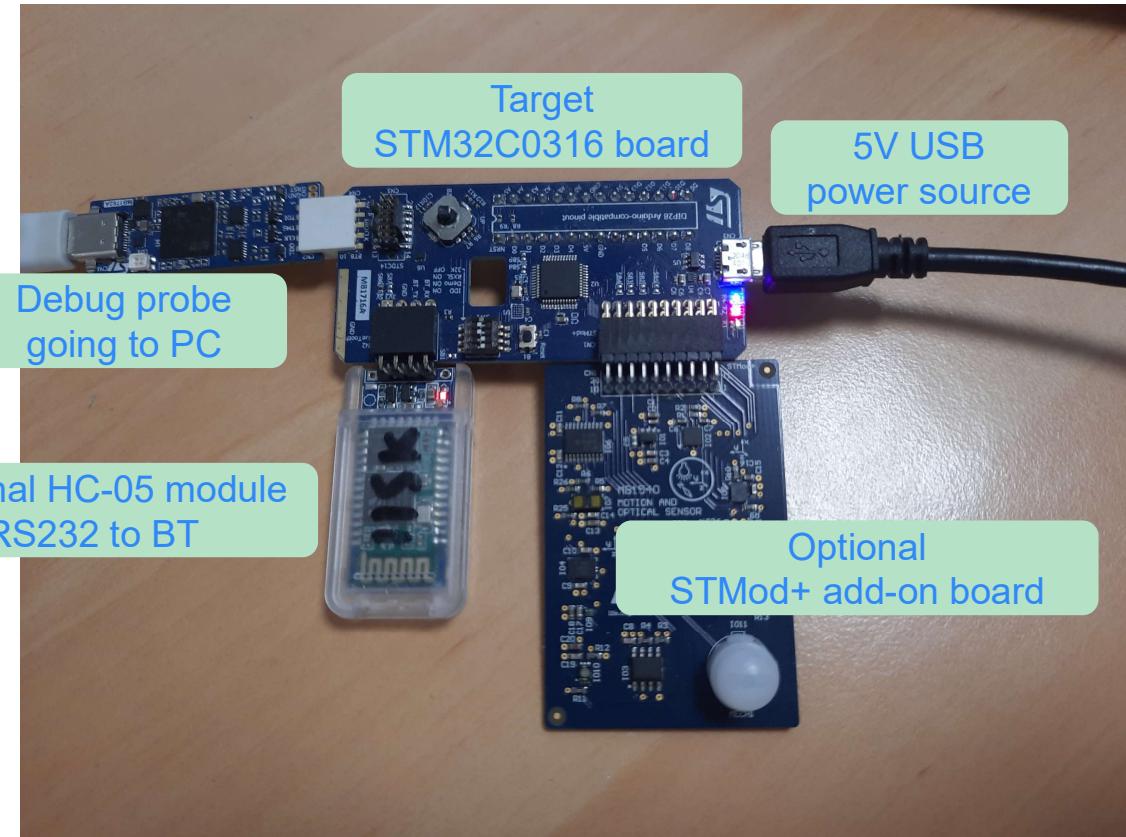
S.Ma

Overview

- Brisk was an internal demo set originally built on STM32L496/L4R
 - It is built from an assembly of various application blocks
 - This demo was reused and stripped down to fit on STM32C0316-DK, first using IAR, and CubeIDE 1.10.0 in 2022
 - Source code was developed aiming to be intuitive/reusable/resource-light/handy for board bring-up
 - It could further be polished up and use some calibration tricks to be core frequency and compile option independent
 - It uses physical units (the human world's) mV, msec, uS, degC_x10
 - It tries to choose the right balance between code efficiency, ease of taking ownership, and modular
- One goal is to be able to go back to the code a year later, and quickly be able to get back to coding

How to play with Brisk demo ?

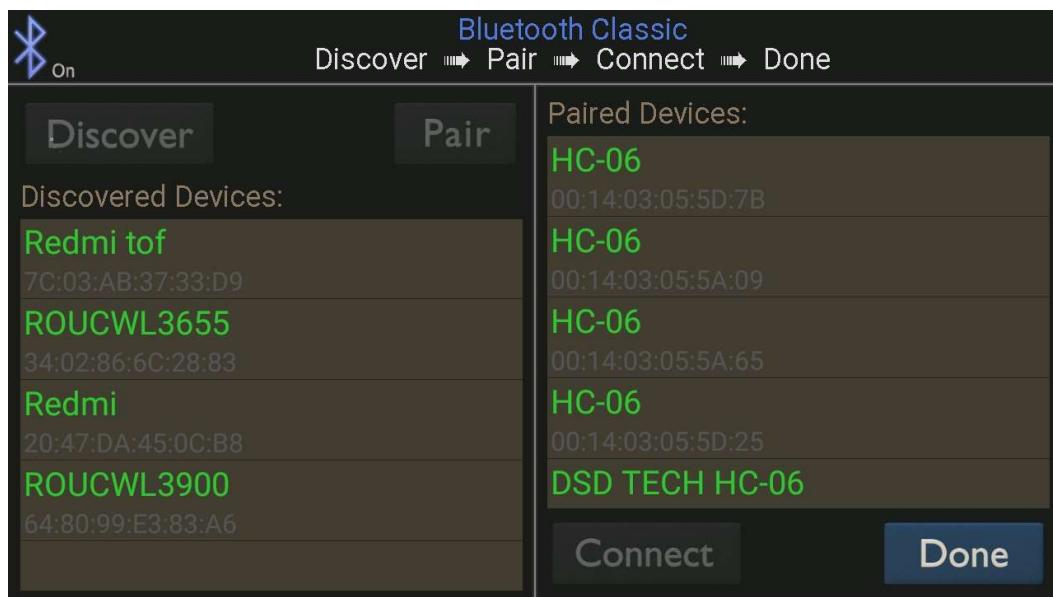
1/2



- Install CubeIDE for STM32C0
- Install and build the **Brisk** Demo
- Provide +5V power by microUSB
- Plug debug probe and connect to PC
- No need to plug an STMod+ add-on
- Optional: Plug HC-05 dongle
 - Dongle setting: 115200bps,8,n,1
 - Its red LED should blink
 - Set the slide switch as in the picture
- Go to Debug mode and run the **Brisk** app
 - Green LED should pulse beat slowly

How to play with Brisk demo ?

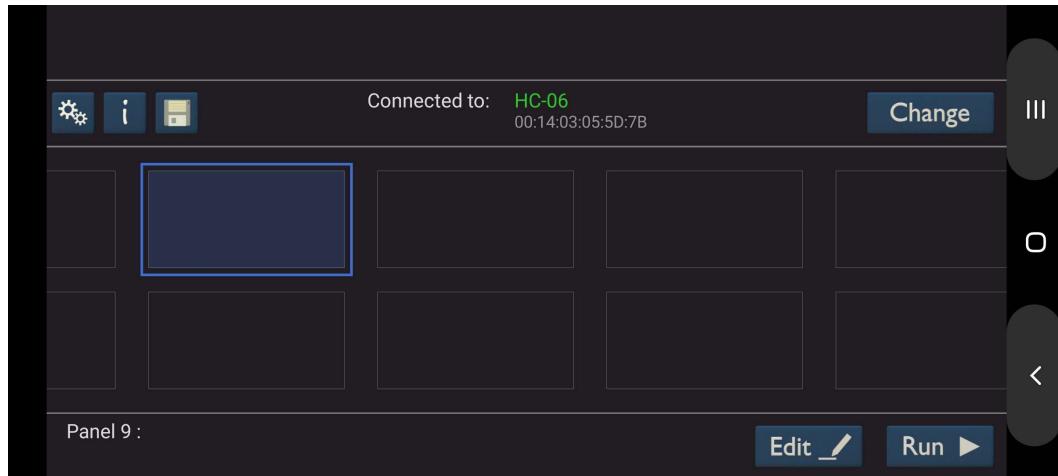
2/2



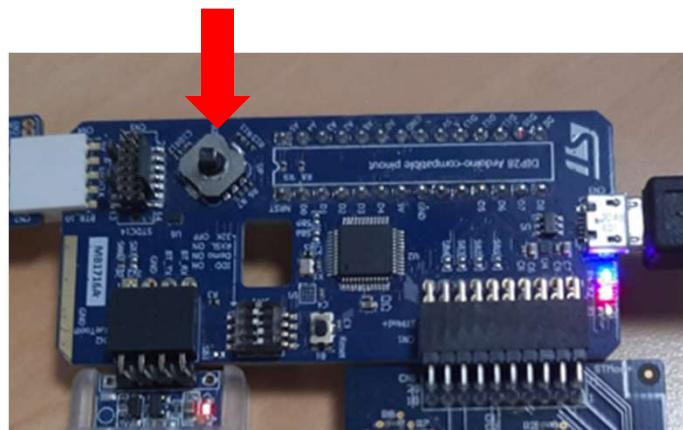
- Run the Android BTEL app
- Select “Connect” while HC-05 is on
- Discover the dongle
 - Its address maybe 00:14:03:05:xx:xx
 - Its name may vary
 - Select and connect
 - HC-05 Red led stops blinking and stays ON
- Click Done

How to play with Brisk demo ?

2/2

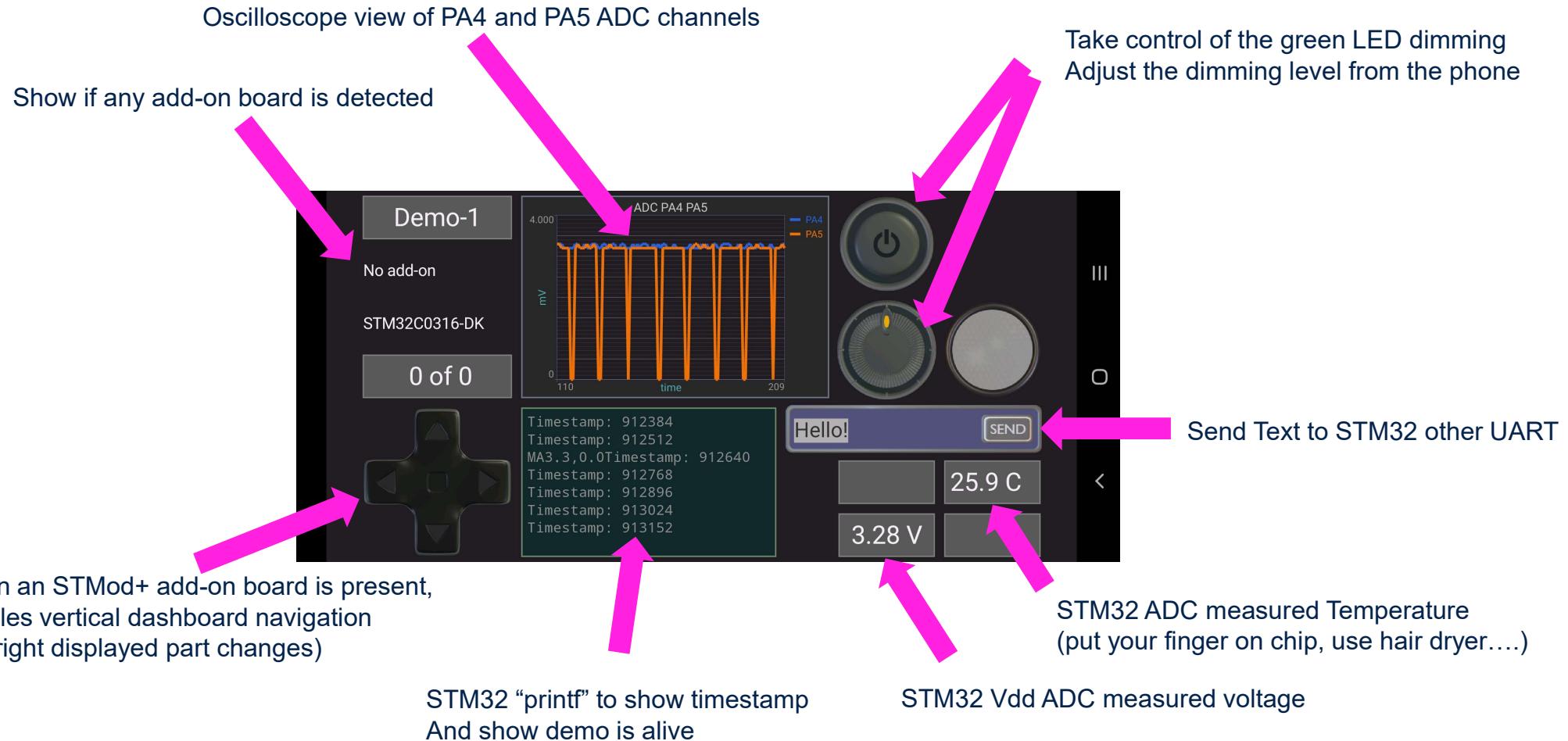


- Select one empty dashboard
- Click “Run”
- Keep pushing down the **key** (5~6 sec)
- A live dashboard will pop up



How to play with Brisk demo ?

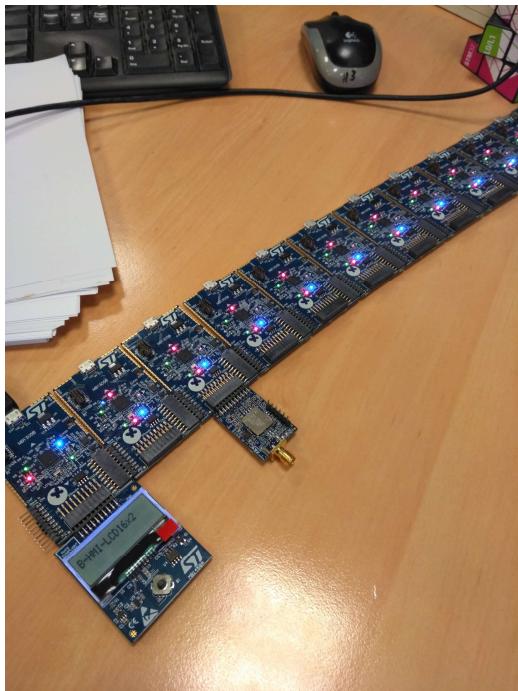
2/2



Brisk Quick Gallery History

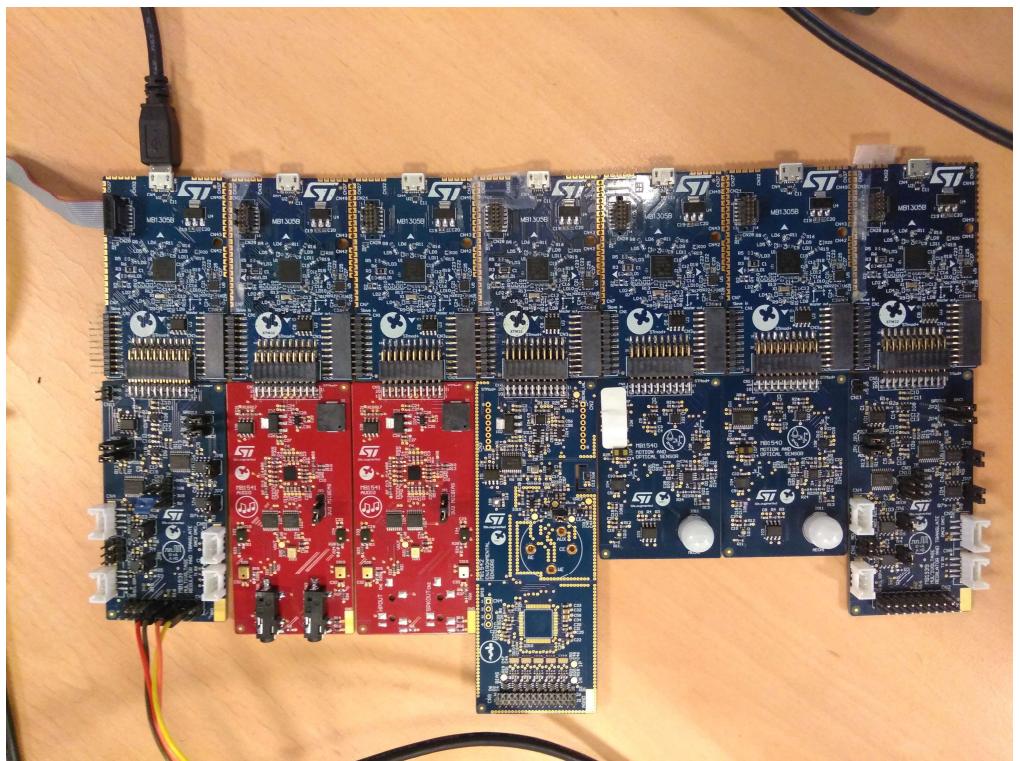
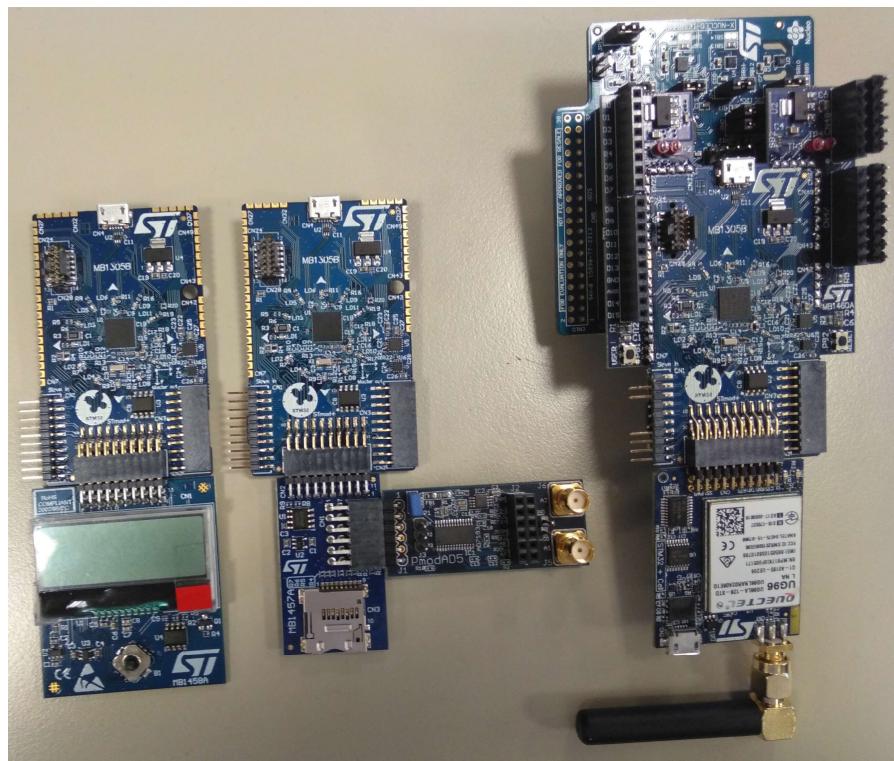
Brisk

Gen 1: STM32L496



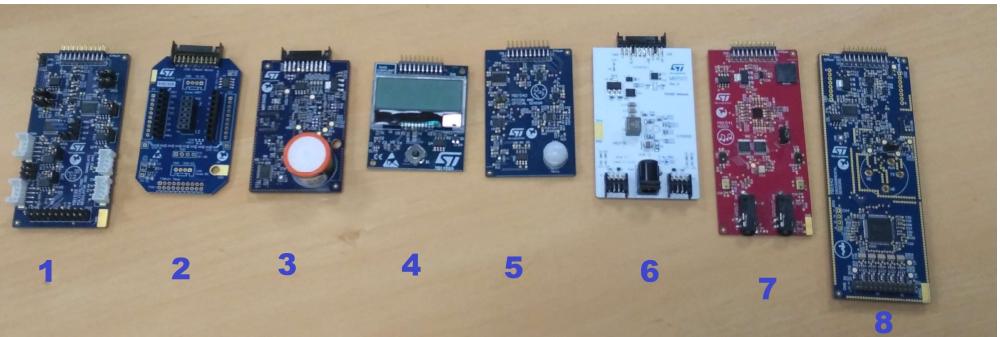
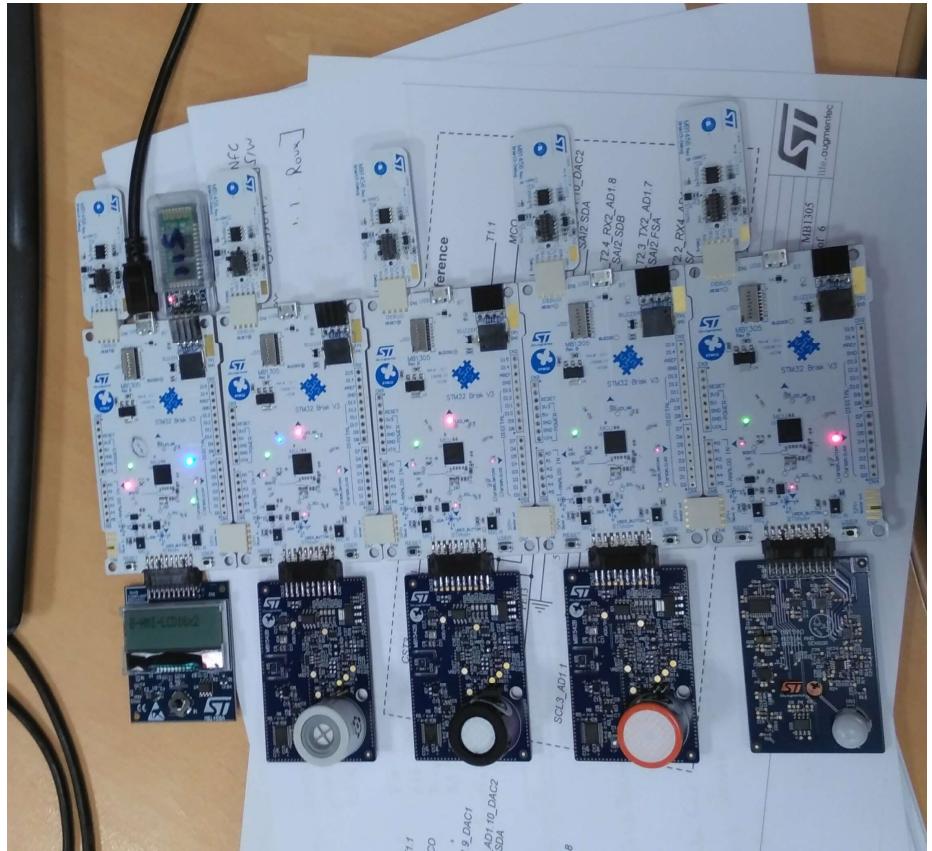
Brisk

Gen 2: STM32L4R



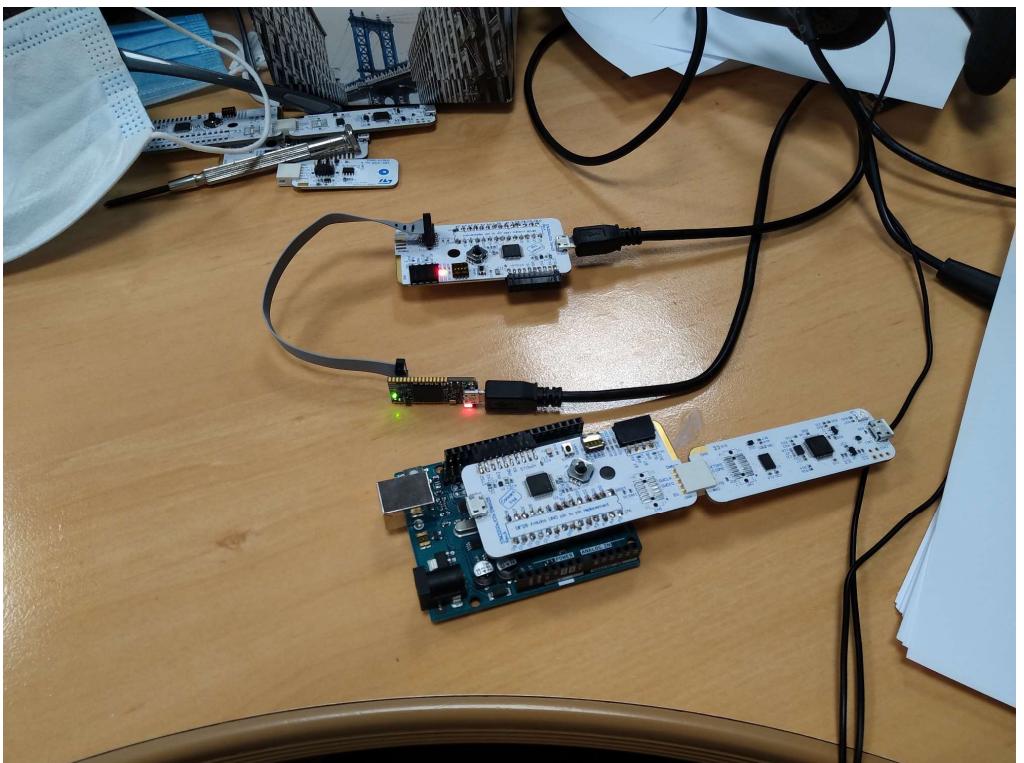
Brisk

Gen 3: STM32L4R



Brisk Subset

Ported over STM32C0316-DK

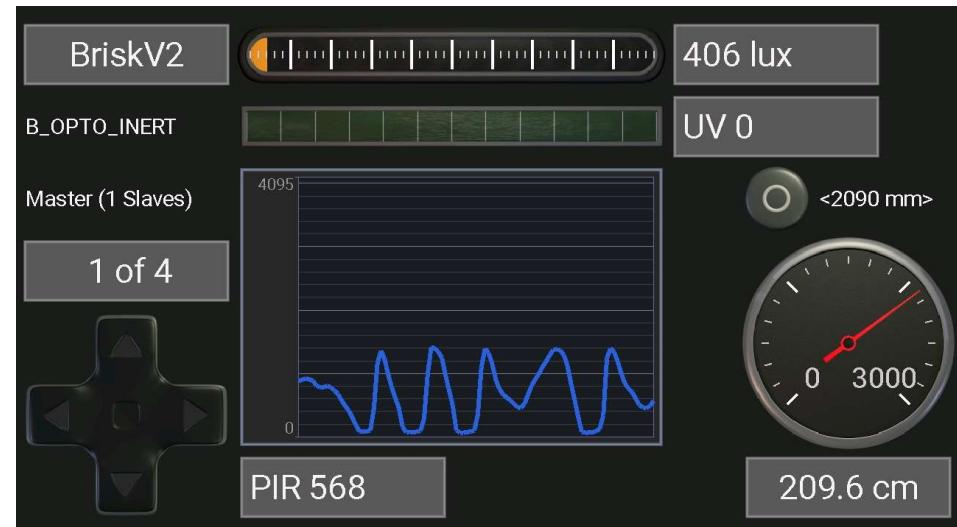


Bluetooth Electronics

Android App



- Cheap RS232 → BT dongle
- Compact RS232 protocol (115200,8,n,1)
- Enable remote dashboard display
- Dashboard layout pushed from MCU to App



Demo Project quick details

Environment Snapshot

CubeIDE 1.10.0

workspace_1.10.0 - STM32C0316_DK_SebDemo/Core/Applications/brisk.c - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer X

- STM32C0316_DK_SebDemo
 - Binaries
 - Includes
 - Core
 - Appli
 - B_ANALOG_ENV
 - B_FAN_OUT2
 - B_HMI_LCD16x2
 - B_OPTO_INERT
 - B_RS485_DCDC
 - Inc
 - IO_DRIVER
 - Rugged_exti_demo
 - Src
 - Startup
 - 20210705_110021_C0_IRQ_OVR.jpg
 - 20210726_115332.jpg
 - 20210809_113119.jpg
 - 20221021_131109.jpg
 - Drivers
 - Debug
 - Import
 - README.MD
 - STM32C0316_DK_SebDemo.ioc
 - STM32C0316_DK_SebDemo.launch
 - STM32C0316_DK_SebDemo.FLASH.ld

main.c S_startup_stm... stm32c0xx_h... B_FAN_OUT2.c io_driver.h IO_pins.h IO_pins.c HTS221.c brisk.c ISM330DLC.c

```

37 #if 0
38 Motion3D_t Motion3D;
39 Magneto3D_t Magneto3D;
40#endif
41
42#ifndef 0
43 const IO_Pin_t UserKeyPin = { GPIOA, { GPIO_PIN_4, GPIO_MODE_INPUT, GPIO_NOPULL, GPIO_SPEED_FREQ_LOW, 0, } }; // as digital push-in button, can
44 const IO_Pin_t PA11_Default = { GPIOA, { GPIO_PIN_11, GPIO_MODE_INPUT, GPIO_PULLUP, GPIO_SPEED_FREQ_LOW, 0, } };
45#endif
46 const IO_Pad_t UserKeyPin = { PA_4, { .Mode = IO_INPUT, .Pull = IO_NOPULL, .Speed = IO_SPEED_01 } }; // as digital push-in button, can also be used as analog ke
47 const IO_Pad_t PA11_Default = { PA_11, { .Mode = IO_INPUT, .Pull = IO_PULLUP, .Speed = IO_SPEED_01 } };
48
49
50 UserKey_t UserKey = {
51     .pPin = (IO_Pad_t*)&UserKeyPin,
52     .Pressed = 0,
53     .PressedCount_50ms = 0,
54 };
55
56 // 1 msec SysTick interrupt handler for Brisk application purpose
57 uint32_t Brisk_1ms_Counter;
58 uint8_t Brisk_5ms_Flag, Brisk_10ms_Flag, Brisk_50ms_Flag, Brisk_100ms_Flag, Brisk_1sec_Flag;
59 uint32_t Brisk_1sec_Counter;
60
61
62void void Brisk_1ms_ISR(void) {
63
64     Brisk_1ms_Counter++;
65     if(Brisk_1ms_Counter % 8/*5*/)==0) { // optimisation for speed and size avoiding division on CortexM0
66         Brisk_5ms_Flag = 1;
67         if(Brisk_1ms_Counter % 16 /*10*/==0) {
68             Brisk_10ms_Flag = 1;
69             if(Brisk_1ms_Counter % 64 /*50*/==0) {
70                 Brisk_50ms_Flag = 1;
71                 if(Brisk_1ms_Counter % 128 /*100*/==0) {
72                     Brisk_100ms_Flag = 1;
73                     if(Brisk_1ms_Counter % 1024 /*1000*/==0) {

```

Problems Tasks Console Properties

STM32C0316_DK_SebDemo

```

..Core/Applications/brisk_reports.c: At top level:
..Core/Applications/brisk_reports.c:112:1: warning: multi-line comment [-Wcomment]
112 | //Add_text(1,5,medium,L,t,245,240,245,9)\n|
| ^
arm-none-eabi-gcc -o "STM32C0316_DK_SebDemo.elf" @objects.list -mcpu=cortex-m0plus -T"C:\Users\marsann\STM32CubeIDE\workspace_1.."
Finished building target: STM32C0316_DK_SebDemo.elf

arm-none-eabi-size STM32C0316_DK_SebDemo.elf
arm-none-eabi-objdump -h -S STM32C0316_DK_SebDemo.elf > "STM32C0316_DK_SebDemo.list"
text    data    bss    dec   hex filename
28340    544    9864   38748   975c STM32C0316_DK_SebDemo.elf
Finished building: default.size.stdout

Finished building: STM32C0316_DK_SebDemo.list

```

12:29:11 Build Finished. 0 errors, 115 warnings. (took 11s.777ms)

Build Analyzer X Static Stack Analyzer Search

STM32C0316_DK_SebDemo.elf - /STM32C0316_DK_SebDemo/Debug - Nov 14, 2022, 12:29:10 PM

Region	Start address	End address	Size	Free	Used
RAM	0x20000000	0x20003000	12 KB	1.84 KB	10.16 KB
FLASH	0x08000000	0x08008000	32 KB	3.79 KB	28.21 KB

Environment Snapshot

CubeIDE 1.10.0

workspace_1.10.1 - STM32C0316-DK-DEMO-BRISK/Brisk/I2C_MasterIO.c - STM32CubeIDE

File Edit Source Refactor Navigate Project Run Window Help

Debug Project Explorer main.c brisk.h add_on_board.h SIF.c startup_stm32c031c6tx.s I2C_MasterIO.c

374 uint8_t IsDevicePresent(I2C_SlaveDevice_t* pD) {
375 return IsSlavePresent(pD->M, pD->SlaveAddr);
376 }
377 */
378 void Brisk_I2C_MasterIO_Init(void) {
379 I2C_MasterIO_Init(&gI2C_STMod);
380 I2C_MasterIO_ConfigTiming(&gI2C_STMod, 100000, 400000, 0);
381 I2C_MasterIO_ConfigM(&gI2C_STMod, PB_11, PB_10);
382 I2C_MasterIO_Enable(&gI2C_STMod);
383
384 ErrorRecovery(&gI2C_STMod);
385 // Test on Nucleo_board
386 I2C_MasterIO_Init(&gI2C_Arduino);
387 I2C_MasterIO_ConfigTiming(&gI2C_Arduino, 100000, 400000, 0);
388 I2C_MasterIO_ConfigM(&gI2C_Arduino, PB_9, PB_8);
389 I2C_MasterIO_Enable(&gI2C_Arduino);
390
391 ErrorRecovery(&gI2C_Arduino);
392 }
393 /*
394 NewI2C_MasterIO_SDA_SCL(&gI2C_STMod, &MIO_SDA_STMod, &MIO_SCL_STMod);
395 I2C_MasterIO_ConfigTiming(&gI2C_STMod, 100000, 400000);
396 I2C_MasterIO_ConfigM(&gI2C_STMod);
397 I2C_MasterIO_Enable(&gI2C_STMod);
398
399 ErrorRecovery(&gI2C_STMod);
400 */
401 }
402 }
403 // remove to save RAM if linker does not do it automatically when not used by the project.
404 uint8_t gSlaveSweep; // use for test only
405
406 void Brisk_I2C_MasterIO_Test(void) { // this function sweeps all slave addresses on one bus.
407 uint16_t StartAddr;
408 // we sweep the slaves
409 StartAddr = 0;
410 while(StartAddr = I2C_MasterIO_FindNextSlaveByAddress(&gI2C_STMod, StartAddr)) {
411
412 gSlaveSweep = StartAddr; // breakpoint here to stop as soon as one slave is detected
413 NOP(); // put breakpoint here
414 } // when StartAddr == 0, it means the sweep is over
415 }

Console Problems Executables Debugger Console Memory STM32C0316-DK-DEMO-BRISK [STM32 C/C++ Application] [pid: 63]

Verifying ...

Download verified successfully

Variables Break... Express... Registers Live E... SFRs

Expression	Type	Value
pRisk	Brisk_t*	{...}
gINT_Stats	INT_Stats_Global_t	{...}
u8fifo_to_SPIP	u8fifo_t [2]	0x20002148 <u8fifo_to_SPIP>
Bytes_to_SPIP	uint8_t [2][1500]	0x2000151c <Bytes_to_SPIP>
StartAddr	I2C_MasterIO_t	{...}
gI2C_STMod	I2C_MasterIO_t	{...}
SDA	PadName_t	PB_11
SCL	PadName_t	PB_10
bitrate_kHz	int32_t	100
ClockStretchTimeout	int16_t	0
AckFail	uint8_t	0 \0'
RawPad_SDA	RawPortPads_t	{...}
pPort	GPIO_TypeDef *	0x50000400
bPin	uint16_t	2048
RawPad_SCL	RawPortPads_t	{...}
fnWaitMethod	int32_t (*)(uint32_t)	0x8001645 <NopsWait>
ctWaitMethod	uint32_t	1
WaitParam	uint32_t	2
gI2C_Arduino	I2C_MasterIO_t	{...}
SDA	PadName_t	PB_9
SCL	PadName_t	PB_8
bitrate_kHz	int32_t	100
ClockStretchTimeout	int16_t	0
AckFail	uint8_t	0 \0'
RawPad_SDA	RawPortPads_t	{...}
RawPad_SCL	RawPortPads_t	{...}
fnWaitMethod	int32_t (*)(uint32_t)	0x8001645 <NopsWait>
ctWaitMethod	uint32_t	n

Environment Snapshot

CubelDE 1.10.0

CubeIDE 1.10.0

The screenshot shows the STM32CubeIDE interface with the following details:

- Project Explorer:** Shows the workspace structure. The main project is "STM32C0316-DK-DEMO-BRISK". Other components include "Binaries", "Includes", "B_ANALOG_ENV", "B_FAN_OUT", "B_HMI_LCD16x2", "B_OPTO_INERT", "B_RS485_DCDC", "Brisk", "add_on_board.c", "add_on_board.h", "analog.c", "analog.h", "brisk_reports.c", "brisk_reports.h", "brisk.c", "brisk.h", "BTLE_drivers.c", "BTLE_drivers.h", "commons.c", "commons.h", "I2C_MasterIO.c", "I2C_MasterIO.h", "IO_pins.c", "IO_pins.h", "IRQ_Self_Settle.c", "IRQ_Self_Settle.h", "LEDs.c", "LEDs.h", "M242560.c", "M242560.h", "SIF.c", "SIF.h", "SPLMasterIO.c", "SPLMasterIO.h", "ubfifo.c", "ubfifo.h", "Core", "Inc", "Src", "main.c", "stm32c0xx_hal_msp.c", "stm32c0xx_it.c", "syscalls.c", "system.c", "system_stm32c0xx.c", "Startup", "Drivers", "IO_Drivers", "Debug", "DOC".
- Code Editor:** Displays the main.c file with the following code snippet:

```
277 // monster hack which destroys the user LED code entirely: // not scalable, not portable, not efficient
278 TIM1->CCR1 = duty; // for a single hardcoded LED...
279
280
281 #endif
282
283 #endif
284
285 #endif
286 }
287
288 //=====
289 // Timed LED
290 TimedLED_t BriskTimedLEDs[LED_COUNT] = {
291     { &BriskLEDs[0], 0, 0, 0, 0, 0, },
292     { &BriskLEDs[1], 0, 0, 0, 0, 0, },
293 };
294
295
296 void BriskTimedLED_10msecTick(void) { // should be called from main loop. not timing critical.
297
298     uint8_t index;
299     TimedLED_t *pTimedLED;
300
301     // coming here every 10msec
302     for(index=1;index<LED_COUNT;index++) {
303
304         pTimedLED = &BriskTimedLEDs[index];
305         if(pTimedLED->Blink_Countdown_x10ms==0) continue; // if countdown is zero, do nothing
306         // countdown is non zero, decrement it.
307         pTimedLED->Blink_Countdown_x10ms--;
308         if(pTimedLED->Blink_Countdown_x10ms==0) { // cycle completed
309             if(pTimedLED->Blink_Countdown==0) continue; // no more blink pulse needed
310             // blink pulse needed
311             if(pTimedLED->Blink_Countdown!=BRISK_LED_BLINK_FOREVER) pTimedLED->Blink_Countdown--; // if not infinite blinking, go to next
312             if(pTimedLED->Blink_Countdown) pTimedLED->Blink_Countdown_x10ms = pTimedLED->Blink_Period_x10ms;
313         };
314
315         // update the LED on/off dimming (refresh every 10 msec in case of dynamic/ISR change or ESD glitch)
316         if(pTimedLED->Blink_Countdown_x10ms<=pTimedLED->Blink_Offtime_x10ms) { // Set LED OFF
317             BriskDimLED(index, 0);
318         } else { // Set LED ON
319             BriskDimLED(index, pTimedLED->Dim_perc);
320         };
321     }
322 }
```

- Variable Viewer:** Shows the expression tree for "BriskTimedLEDs". The tree includes nodes for TimedLED_t, LED1_t, IO_PadName, PadConfig, Mode, Pull, Drive, Speed, Odr, Signal, TIM, TIM_CH, Dim_perc, Blink_Period_x10r, Blink_Offset_x1l, Blink_Countdown, and Dim_perc.



Functional blocks

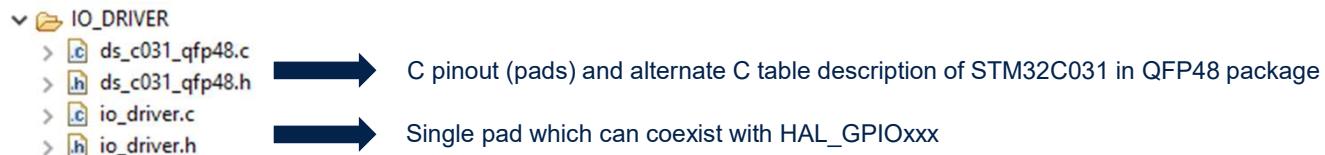
Brief list

- IO_Driver is an alternate way to control IO Pins more intuitively
- I2C_MasterIO provides I2C Master using IO Bit banging to access Slave Devices (sensors, eeprom)
- SPI_MasterIO provides SPI Master using IO Bit banging to drive a 2x16 characters-based LCD
- LEDs enable list of LEDs with digital blinking and/or dimming (PWM as %)
- U8FIFO is a SW Fifo to plumb data between asynchronous data provider and consumer
- SIF and BTEL_Drivers are serial data using Bluetooth Electronics protocol to use remote Phone display
- The STMod+ Add-on boards are plug and play and contain various sensors and other things
 - Add_on_board.c perform the plug and play by regularly searching the presence of the add-on board EEPROM ID
 - Each add-on board is named “B_xxxxx” and its source files and documents are placed in a sub folder
 - Due to small STM32C03 memory, only 1 or 2 boards can be supported with limited sensor support (no float) → add_on_board.h
- Brisk is the top application level called from main()
- Analog perform one shot multiple channels including measuring STM32 Vdda and Temperature

GPIOs can be controlled by either HAL and LL.

These APIs require a pin, defined by a **set** (GPIO port, BitMask).

IO_Driver focuses on intuitiveness and easy to use APIs using a **single** pin name.
IO_Driver uses “**Pads**” to differentiate and coexist with HAL and LL.



Browse the application code to discover its use, which was previously using HAL.
The IO_Driver check pins validity, is interrupt proofed, deals with EXTI and makes pin configuration friendlier with an overall efficient generated code. For bit-bang speedy I/Os, RawPads are introduced to make I2C_Master or SPI_Master emulation.

Datasheet package pinout and alternate functions signals have been C encoded too.

SW Elements

IO_Driver

Search for these API in the application code to find out how it is used... Check the MAP file for code density, check I2C_Master_IO for I2C Bitbang on any pin
If you'd like to migrate from HAL to IO_Driver step by step, use IO_Pins.c/h which first create Pad APIs using HAL, then rename pin into pad to finish the conversion...

```
int32_t IO_PadInit(IO_Pad_t *pIO_Pad); // intuitive
int32_t IO_PadDeInit(PadName_t PadName); // intuitive
int32_t IO_PadGet(PadName_t PadName); // thread safe, speed
int32_t IO_PadSetHigh(PadName_t PadName); // thread safe, speedy
int32_t IO_PadSetLow(PadName_t PadName); // thread safe, speedy
int32_t IO_PadSet(PadName_t PadName, GPIO_PinState PinState); // thread safe, speedy
int32_t IO_PadToggle(PadName_t PadName); // thread safe, speedy
HAL_StatusTypeDef IO_PadLock(PadName_t PadName);
//void HAL_IO_EXTI_IRQHandler(PadName_t PadName); // moved to EXTI driver
//void HAL_IO_EXTI_Rising_Callback(PadName_t PadName);
//void HAL_IO_EXTI_Falling_Callback(PadName_t PadName);

void IO_RawPadSetHigh(RawPortPads_t* RawPad);
void IO_RawPadSetLow(RawPortPads_t* RawPad);
uint32_t IO_RawPadGet(RawPortPads_t* RawPad);

int32_t IO_EXTI_Config(PadName_t PadName, EXTI_Config_t EXTI_Config);
int32_t IO_EXTI_DeInit(PadName_t PadName);

// IO_Pin_t MIO_SDA_STMod = { GPIOB, { GPIO_PIN_11, GPIO_MODE_OUTPUT_OD, GPIO_PULLUP, GPIO_SPEED_FREQ_LOW, 0, } };
IO_Pad_t SDA = { PB_11, { .Mode = IO_OUTPUT, .Pull = IO_PULLUP, .Drive = IO_OPENDRAIN, .Speed = IO_SPEED_01, .Odr = IO_ODR_HIGH } };

const IO_Pad_t LED1_AsOutput = { PA_5, { .Mode = IO_OUTPUT, .Drive = IO_PUSH_PULL, .Pull = IO_NOPULL, .Speed = IO_SPEED_01, .Odr = IO_ODR_HIGH } };
// LED1 = Digital Output, the undefined configuration fields will remain unchanged. .Odr field enables set the output level before turning the pin as output.

const IO_Pad_t LED1_AsPWM = { PA_5, { .Mode = IO_SIGNAL, .Drive = IO_PUSH_PULL, .Pull = IO_NOPULL, .Speed = IO_SPEED_01, .Odr = IO_ODR_HIGH, .Signal = TIM1_CH1/*5*/ } };
// LED1 = Digital PWM T1.1 AF5 IO_SIGNAL means use alternate function signal. No number needed (portability), only valid signal name: TIM1_CH1
```

SW Elements

I2C_MasterIO overview

I2C_MasterIO is an I2C Master implementation with just GPIO and SW. Useful for board bringup and flexible pin assignment, more core bandwidth intensive than using a HW I2C.

>  I2C_MasterIO.c
>  I2C_MasterIO.h



Defines an I2C Master Bitbang and Slave Configuration

The driver defines the I2C bus properties, pins, speed, and slave configuration. By default, this code does not support clock stretching and drive true slaves only.

`int32_t I2C_MasterIO_AccessSlave(I2C_SlaveDevice_t* pD);`

The only API needed to drive all the I2C Slave supported in the Application (browse it) APIs are provided to sweep of all I2C Slave addresses, handy for board bring up.

The module has a local test function as example.

Slave code example are for example M24256D (EEPROM), STTS751, BME280, LPS22HH... .

Note: Sensirion sensor is not a true I2C Slave and has its own bit bang driver.

SW Elements

I2C_MasterIO snapshot

M24256D API extract: 2 bytes sub address for 32 kbyte data with pages of 64 bytes. Here is the API to write data in ID OTP Page and verify.

```
uint8_t EEP_WriteID (EEP_t* pEEP) {  
  
    //uint8_t RetryCount = MAX_RETRIALS_ON_ERROR;  
    uint8_t i,buf[EEPPAGESIZE_BYTE];  
    uint8_t result = 1;  
    I2C_SlaveDevice_t* pDevice = pEEP->pDeviceID;  
  
    pDevice->SubAdrBytes[0] = 0x00;// msb  
    pDevice->SubAdrBytes[1] = 0x00;// lsb;  
    pDevice->pWriteByte = pEEP->ID_Page;  
    pDevice->WriteByteCount = EEPPAGESIZE_BYTE; // no data write  
    pDevice->pReadByte = 0;  
    pDevice->ReadByteCount = 0;  
    result = I2C_MasterIO_AccessSlave(pDevice);  
    //  
    EEP_WaitWriteCompletion(pEEP); // to mature  
    for(i=0;i<EEPPAGESIZE_BYTE;i++) // backup original  
        buf[i] = pEEP->ID_Page[i];  
  
    // verify  
    EEP_ReadID (pEEP);  
    for(i=0;i<EEPPAGESIZE_BYTE;i++) {  
        if(buf[i] != pEEP->ID_Page[i])  
            break;  
        pEEP->ID_Page[i] = buf[i];  
    };  
    if(i != EEPPAGESIZE_BYTE)  
        TrapError(); // warning for hot plug (not supposed to happen for ID Page)  
  
    return result; // if 0 : writing failed
```

...

- To be continued based on feedback

Project Source License Inventory

Project author's source code

file header

```
*****  
* @file      add_on_board.c (STMod+ plug and play add-on board manager)  
* @author    S.Ma  
* @brief     (...)  
*  
*****  
* @attention  
*  
* Copyright (c) 2018-2022 STMicroelectronics.  
* All rights reserved.  
*  
* This software is licensed under terms that can be found in the LICENSE file  
* in the root directory of this software component.  
* If no LICENSE file comes with this software, it is provided AS-IS.  
*  
*****
```

Apache License

Version 2.0, January 2004

<http://www.apache.org/licenses/>

<http://www.apache.org/licenses/LICENSE-2.0>

Copyright 2022 STMicroelectronics. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ST Sensor Driver License

Motion Mems and Environmental

```
/* ****
* @file  ism330dlc_reg.h
* @author Sensors Software Solution Team
* @brief This file contains all the functions prototypes for the
*        ism330dlc_reg.c driver.
*****
* @attention
* <h2><center>&copy; COPYRIGHT(c) 2019 STMicroelectronics</center></h2>
* Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
* 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* 3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

ST Sensor Driver License

Optical Time Of Flight

* Copyright (c) 2017, STMicroelectronics - All Rights Reserved

- This file : part of VL53L1 Core and : dual licensed, either 'STMicroelectronics Proprietary license' or 'BSD 3-clause "New" or "Revised" License' , at your option.

* 'STMicroelectronics Proprietary license'

* License terms: STMicroelectronics Proprietary in accordance with licensing terms at www.st.com/sla0081

* STMicroelectronics confidential Reproduction and Communication of this document : strictly prohibited unless specifically authorized in writing by STMicroelectronics.

* Alternatively, VL53L1 Core may be distributed under the terms of 'BSD 3-clause "New" or "Revised" License', in which case the following provisions apply instead of the ones mentioned above :

* License terms: BSD 3-clause "New" or "Revised" License.

* Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Bosch Sensor

Header

```
/** \mainpage *****/
```

* Copyright (C) 2010 - 2015 Bosch Sensortec GmbH

* File : bmg160.h

* Date : 2015/04/29

* Revision : 2.0.4 \$

* Usage: Sensor Driver for BMG160 sensor

```
*****
```

* \section License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the copyright holder nor the names of the contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

The information provided is believed to be accurate and reliable. The copyright holder assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of the copyright holder.

```
*****/
```



/*

* Copyright (c) 2018, Sensirion AG

* All rights reserved.

* Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* * Neither the name of Sensirion AG nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

*/

Our technology starts with You



Find out more at www.st.com

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to www.st.com/trademarks.

All other product or service names are the property of their respective owners.



- The project has a “Rugged EXTI demo” (IRQ_Self_Settle source files)
- Simple demo of how to prevent EXTI IO interrupt “DDOS” by hacker injecting MHz square wave onto an IO Interrupt enabled pin
- The demo uses the Fan Out board (picture provided)
 - The MCO pin is enabled to output 8 MHz from STM32
 - One of the STMod+ pin is configured as EXTI interrupt
 - Using a jumper wire to short MCO to EXTI will flood the code with interrupts and hang it
 - The green beat pulsing LED will freeze
 - Using the user rock switch, the countermeasure can be enabled so the code become immune
 - A Timer delay is used to delay the EXTI interrupt reactivation limiting the CPU interrupt load

For further support in creating a PowerPoint presentation, including graphic assets, formatting tools and additional information on the ST brand

you can visit the ST Brand Portal

<https://brandportal.st.com>

