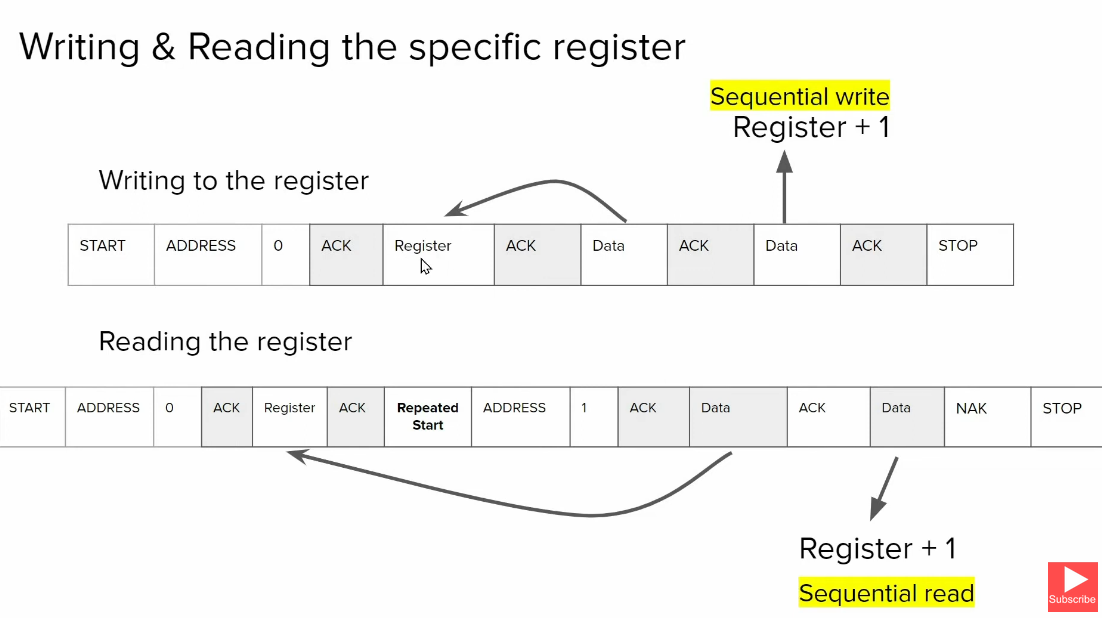
ODHS I2C with MPU6050 using STM32L432KCU6

# Problem statement

*Read from the who am I register of the gyroscope(mpu6050) using I2C protocol using STM32L432KC*

# theory of i2c read and write

in i2c generally sequential write and read is supported (as it is in the case of mpu6050).



## To write to a specific register:-

* start bit
* address of slave
* R/W bit (=0 to write)
* Get Ack from slave
* Send 1st register address to be written to
* Send data (the data is written to the said register)
* If any following data is sent, it is sequentially written onto register+i th register
* Stop bit

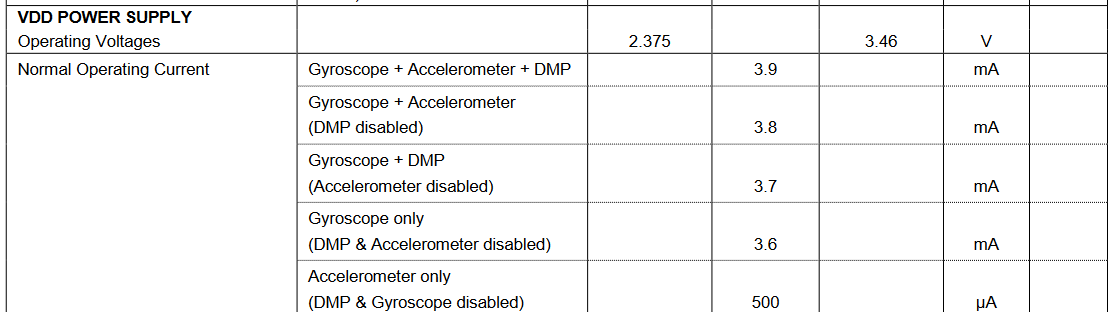
## To read data from a register:-

* Start bit
* Address of slave
* R/W bit =0
* Get ack from slave
* Send 1st register address to be read
* Ack from slav
* Then resend start bit
* Address of the slave followed by R/W bit =1 (to read mode)
* Slave send ack and sends the data on that register
* If master sends back ACK then the slave sends the data in register +1 . this keeps happening till when master keeps sending ack
* To stop the conversation, master send a NAK followed by stop bit

# MPU6050

The MPU-60X0 is an integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) and a dedicated I2C sensor bus.

Electrical specifications:-

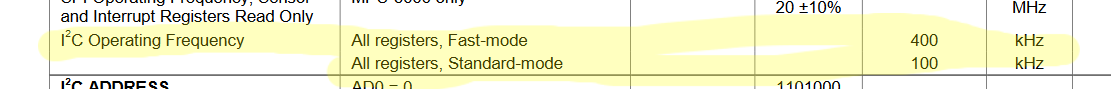


Part 6.3 of MPU6050 Datasheet

* VDD can vary from 2.375V to 3.46V and is Thus tied to the 3V output of the STM32F4.. board

## I2C specifications:-

### Operating Frequency

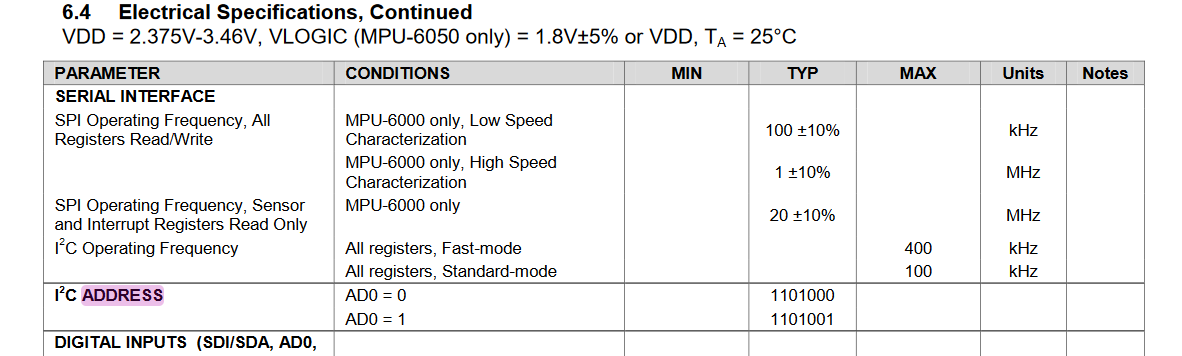


In fast mode the MPU can do upto 400kHz and in standard mode it can do upto 100kHz

* For this project standard mode is used at a frequency of 100000 Hz i.e. 100kHz

### Address of MPU6050

The address of MPU6050 is :-



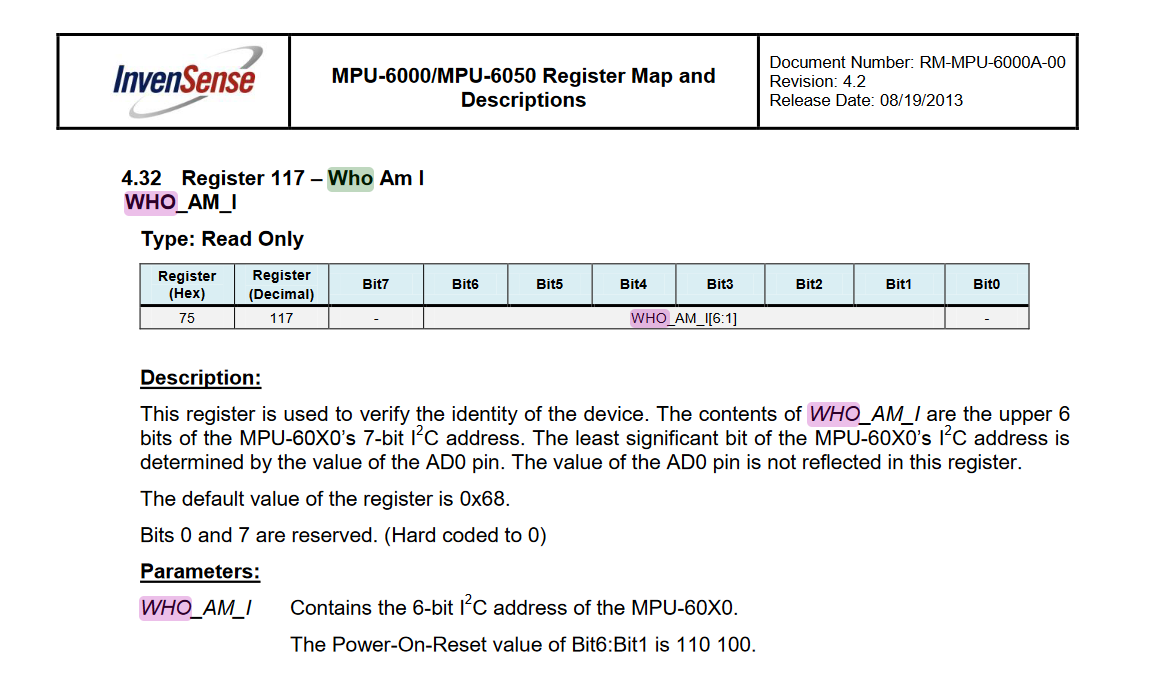
When AD0=0 address= 1101000 (using this)

When AD0=1 address= 1101001

* The AD0 pin is connected to GND to make it 0 and thus the Address of the MPU6050 = (0b110100<<1 +0)

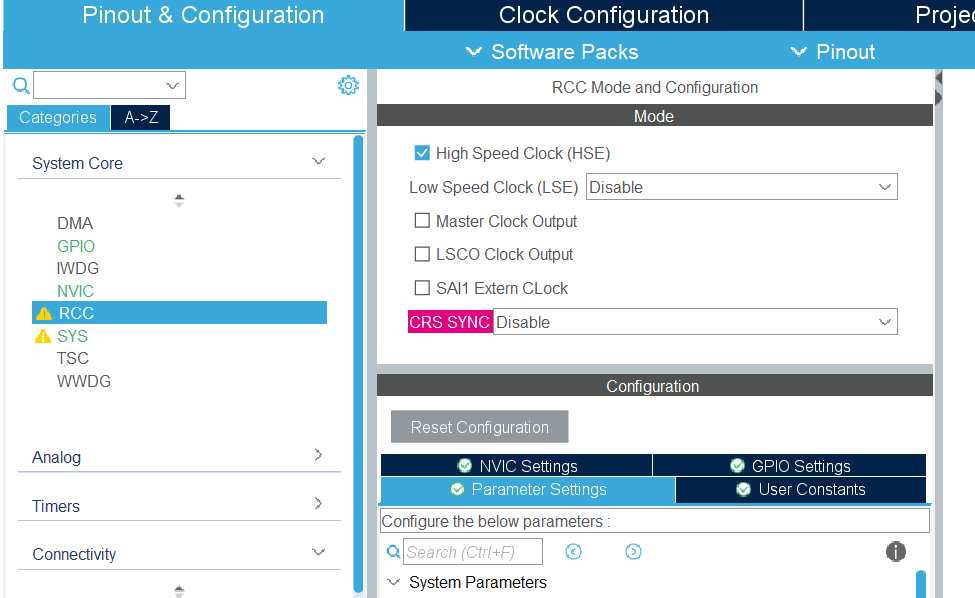
### Register to be accessed

Register 117- Who AM I is to be accessed

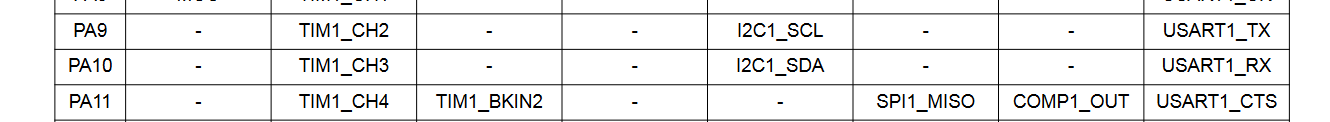


# Setting up the project

* Open a new STM32 project on STM32cubeIDE and select the STM32L432KCU6 processor.
* Connect the RCC Clock



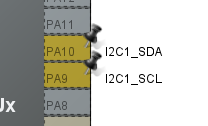
* According to the datasheet of STM32L4



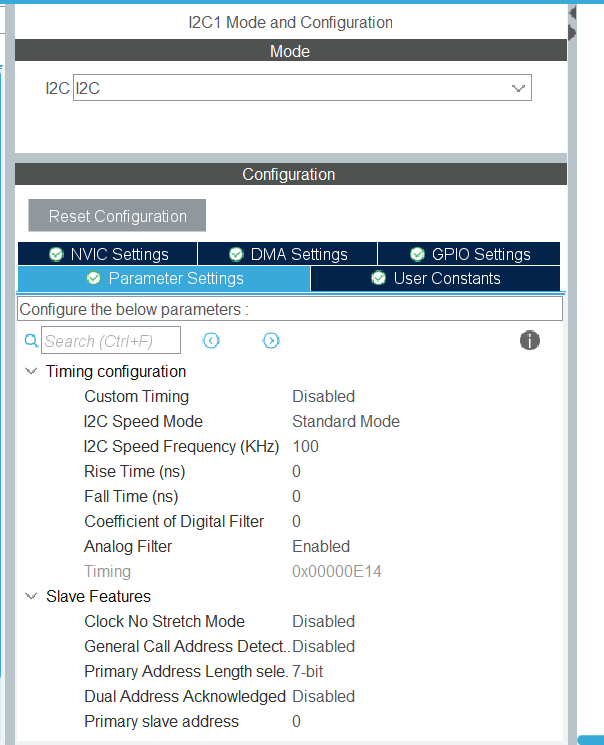
PA9= I2C1 SCL

PA10 =I2C1 SDA

Thus the pins are configured as such



* Under connectivity>I2C1 > mode is turned from disabled to i2c



All the parameters are left as default

i.e. i2c speed mode = standard mode

i2c speed frequency = 100kHz

* In the file /Drivers/STM32L4xx\_HAL\_Driver/Inc/stm32l4xx\_hal\_i2c.h , there are 3 different types of IO operation functions available for I2C

1. Blocking mode: Polling - In this mode, the CPU waits for the communication and cant be used for anyother task (hence blocking)
2. Non blocking mode : Interrupt mode – works on interrupt basis
3. Non blocking : DMA mode- DMA is a hardware process that can handle data transfers without processor intervention. Thus, events can happen behind the scene without needing to interrupt the processor

* In this case,polling mode is used.

/\* IO operation functions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\* Blocking mode: Polling \*/

**HAL\_StatusTypeDef** **HAL\_I2C\_Master\_Transmit**(**I2C\_HandleTypeDef** \*hi2c, **uint16\_t** DevAddress, **uint8\_t** \*pData,

**uint16\_t** Size, **uint32\_t** Timeout);

**HAL\_StatusTypeDef** **HAL\_I2C\_Master\_Receive**(**I2C\_HandleTypeDef** \*hi2c, **uint16\_t** DevAddress, **uint8\_t** \*pData,

**uint16\_t** Size, **uint32\_t** Timeout);

**HAL\_StatusTypeDef** **HAL\_I2C\_Slave\_Transmit**(**I2C\_HandleTypeDef** \*hi2c, **uint8\_t** \*pData, **uint16\_t** Size,

**uint32\_t** Timeout);

**HAL\_StatusTypeDef** **HAL\_I2C\_Slave\_Receive**(**I2C\_HandleTypeDef** \*hi2c, **uint8\_t** \*pData, **uint16\_t** Size,

**uint32\_t** Timeout);

**HAL\_StatusTypeDef** **HAL\_I2C\_Mem\_Write**(**I2C\_HandleTypeDef** \*hi2c, **uint16\_t** DevAddress, **uint16\_t** MemAddress,

**uint16\_t** MemAddSize, **uint8\_t** \*pData, **uint16\_t** Size, **uint32\_t** Timeout);

**HAL\_StatusTypeDef** **HAL\_I2C\_Mem\_Read**(**I2C\_HandleTypeDef** \*hi2c, **uint16\_t** DevAddress, **uint16\_t** MemAddress,

**uint16\_t** MemAddSize, **uint8\_t** \*pData, **uint16\_t** Size, **uint32\_t** Timeout);

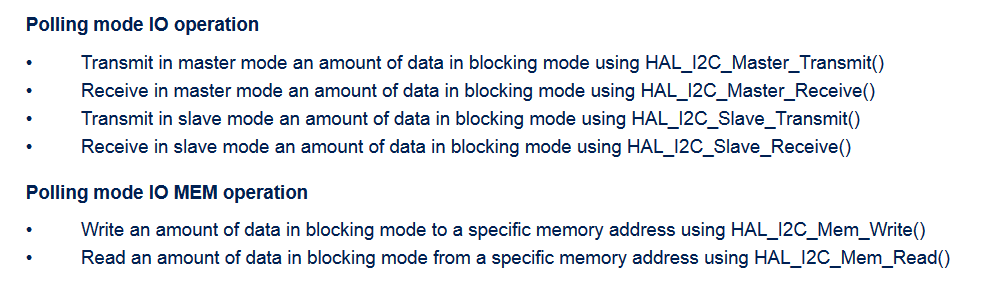
**HAL\_StatusTypeDef** **HAL\_I2C\_IsDeviceReady**(**I2C\_HandleTypeDef** \*hi2c, **uint16\_t** DevAddress, **uint32\_t** Trials,

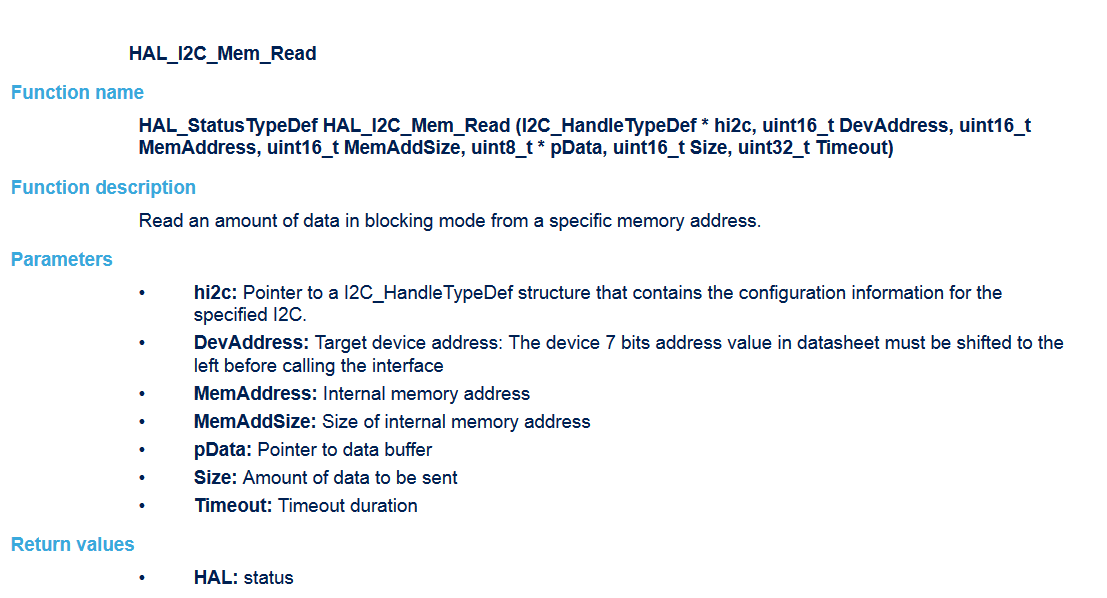
**uint32\_t** Timeout);

here for normal i2c communications first four functions (i.e HAL\_I2C\_Master\_Transmit to HAL\_I2C\_Slave\_Receive are used)

where as to read registers/memory HAL\_I2C\_Mem\_Write and HAL\_I2C\_Mem\_Read is used

HAL\_I2C\_IsDeviceReady is used to check whether the device is ready or not.



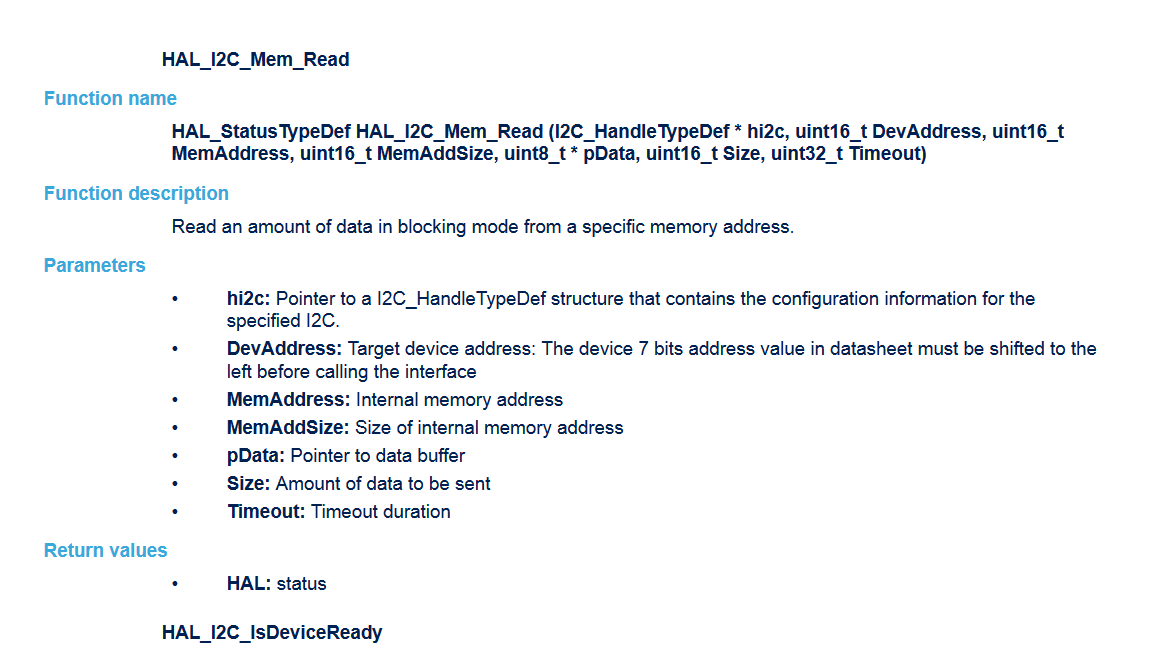


# Code

In the main.c file

* Define the device address and register address
* /\* USER CODE BEGIN PD \*/
* **#define** **MPU6050\_address** (0x68<<1)+0 //device address with AD0 to GND
* **#define** **who\_am\_i\_reg\_add** 0x75//register address
* /\* USER CODE END PD \*/
* Main function
* **int** **main**(**void**)
* {
* /\* USER CODE BEGIN 1 \*/
* **HAL\_StatusTypeDef** ret; //to store the return of the function
* **uint8\_t** rData; // data buffer to store the data read
* /\* USER CODE END 1 \*/
* /\* MCU Configuration--------------------------------------------------------\*/
* /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/
* **HAL\_Init**();
* /\* USER CODE BEGIN Init \*/
* /\* USER CODE END Init \*/
* /\* Configure the system clock \*/
* **SystemClock\_Config**();
* /\* USER CODE BEGIN SysInit \*/
* /\* USER CODE END SysInit \*/
* /\* Initialize all configured peripherals \*/
* **MX\_GPIO\_Init**();
* **MX\_I2C1\_Init**();
* /\* USER CODE BEGIN 2 \*/
* ret=**HAL\_I2C\_Mem\_Read**(&hi2c1,MPU6050\_address,who\_am\_i\_reg\_add, 1,&rData, 1, 1000);
* /\* USER CODE END 2 \*/
* /\* Infinite loop \*/
* /\* USER CODE BEGIN WHILE \*/
* **while** (1)
* {
* /\* USER CODE END WHILE \*/
* /\* USER CODE BEGIN 3 \*/
* }
* /\* USER CODE END 3 \*/
* }

Function used:-

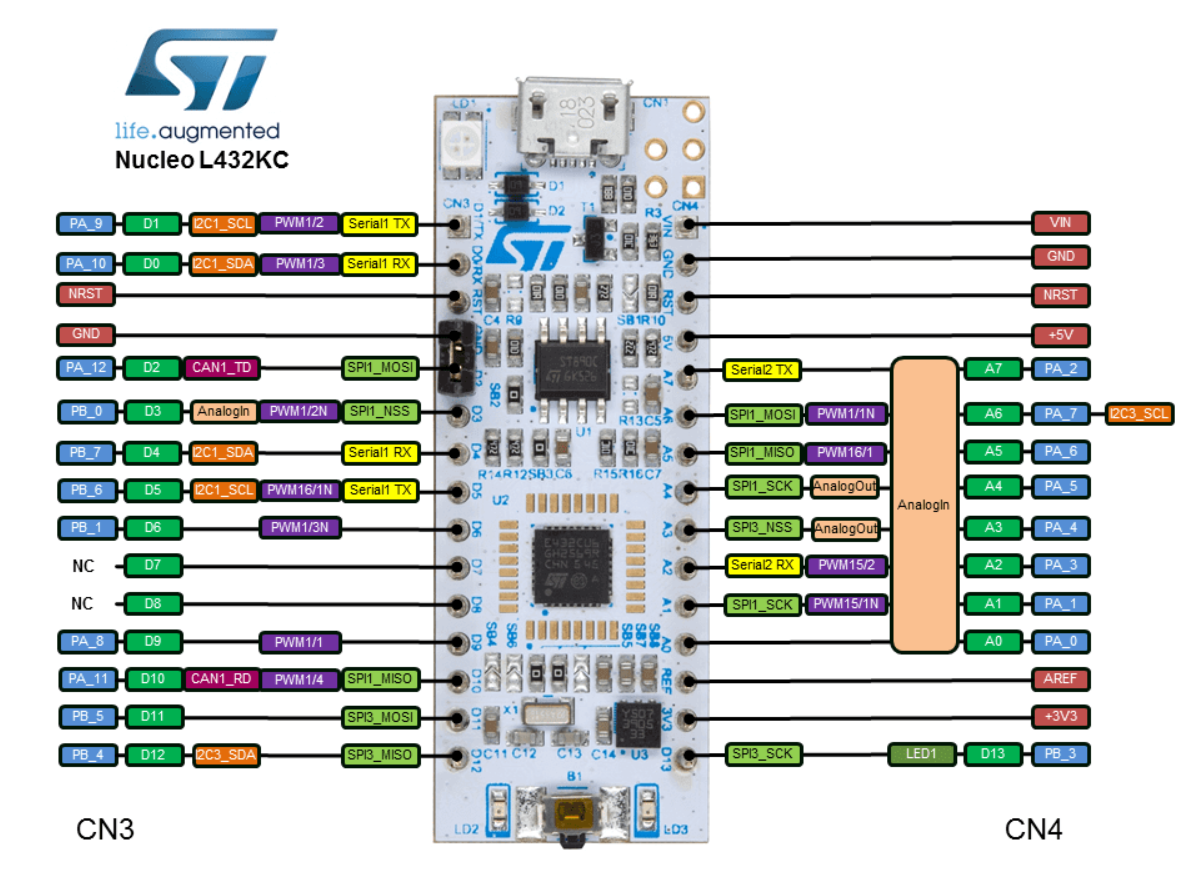


* Upon building the project, console:-
* 20:01:25 \*\*\*\* Incremental Build of configuration Debug for project i2c\_STM32L4\_MPU6050 \*\*\*\*
* make -j8 all
* arm-none-eabi-gcc "../Core/Src/main.c" -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE\_HAL\_DRIVER -DSTM32L432xx -c -I../Core/Inc -I../Drivers/STM32L4xx\_HAL\_Driver/Inc -I../Drivers/STM32L4xx\_HAL\_Driver/Inc/Legacy -I../Drivers/CMSIS/Device/ST/STM32L4xx/Include -I../Drivers/CMSIS/Include -O0 -ffunction-sections -fdata-sections -Wall -fstack-usage -fcyclomatic-complexity -MMD -MP -MF"Core/Src/main.d" -MT"Core/Src/main.o" --specs=nano.specs -mfpu=fpv4-sp-d16 -mfloat-abi=hard -mthumb -o "Core/Src/main.o"
* ../Core/Src/main.c: In function 'main':
* ../Core/Src/main.c:70:27: warning: variable 'ret' set but not used [-Wunused-but-set-variable]
* 70 | HAL\_StatusTypeDef ret; //to store the return of the function
* | ^~~
* arm-none-eabi-gcc -o "i2c\_STM32L4\_MPU6050.elf" @"objects.list" -mcpu=cortex-m4 -T"D:\parikshit\ODHS\STM32CubeIDE\STM32\i2c\_STM32L4\_MPU6050\STM32L432KCUX\_FLASH.ld" --specs=nosys.specs -Wl,-Map="i2c\_STM32L4\_MPU6050.map" -Wl,--gc-sections -static --specs=nano.specs -mfpu=fpv4-sp-d16 -mfloat-abi=hard -mthumb -Wl,--start-group -lc -lm -Wl,--end-group
* Finished building target: i2c\_STM32L4\_MPU6050.elf
* arm-none-eabi-size i2c\_STM32L4\_MPU6050.elf
* arm-none-eabi-objdump -h -S i2c\_STM32L4\_MPU6050.elf > "i2c\_STM32L4\_MPU6050.list"
* text data bss dec hex filename
* 10052 20 1652 11724 2dcc i2c\_STM32L4\_MPU6050.elf
* Finished building: default.size.stdout
* Finished building: i2c\_STM32L4\_MPU6050.list
* 20:01:26 Build Finished. 0 errors, 1 warnings. (took 1s.248ms)

# Debugging and output

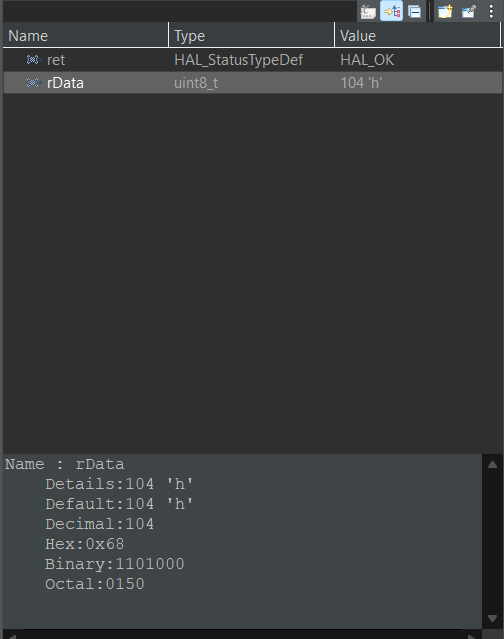
* MPU6050 is connected as follows :-

|  |  |  |
| --- | --- | --- |
| PIN name (on STM32) | STM32 Pin | Connected to (pin on MPU6050) |
| D1 | PA9 | SCL |
| D0 | PA10 | SDA |
| 3V | - | Vcc |
| GND | -- | GND |
| GND | - | AD0 |



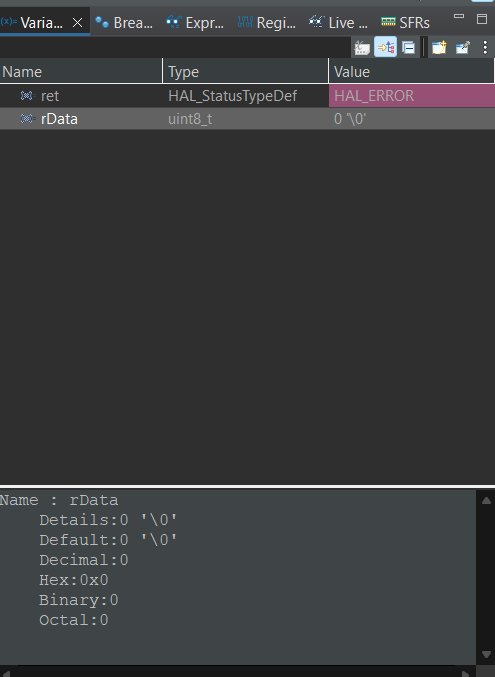
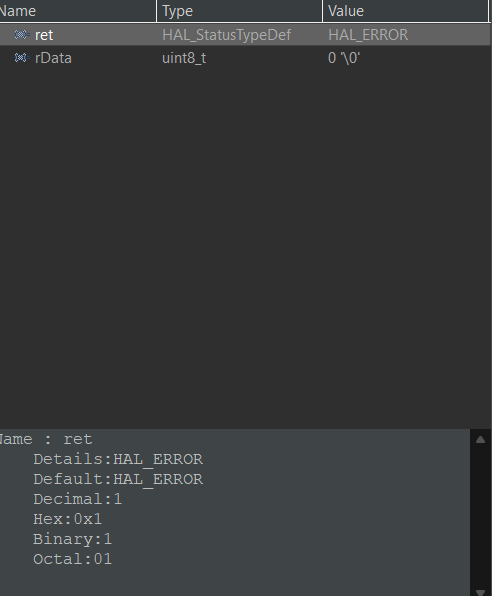
* STM32cubeIDE is put into debugging mode, the code is debugged

## OUTPUT:-



# Errors:-

When using RCC LSE clock , the HAL\_I2C\_Mem\_Read function returns the following.

Solution:- Thus, HSE clock is used

# References:-

<https://www.youtube.com/playlist?list=PLmXXQ1iFwiyKKURU4wAeGT_d1HsIe1YCY>

MPU6050 datasheet:-

<https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>

MPU6050 register map:-

<https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf>

STM32L462KCU6 Datasheet:-

<https://www.st.com/en/microcontrollers-microprocessors/stm32l432kc.html#documentation>

HAL Documentation:-

<https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf>