Alex Baret
CS5780 Post-Lab 5

Lab 5 GitHub project: https://github.com/alex-baret/CS5780/tree/master/Labs/LAB5/CS5780_LAB5_Alex_Baret

To find *main.c* within the GitHub repo, click the commit link -> click ‘Browse files -> navigate to: Labs/LAB5/CS5780_LAB5_Alex_Baret/Core/Src/main.c

‘Correctly reading WHO_AM_I register. 5.1 complete’ contains the state of main.c for the first lab 5 checkoff (5.1)

‘Changed gyroscope init to follow figure 5.7 exactly. Part 2 working’ contains the state of main.c for the second lab 5 checkoff (5.2)

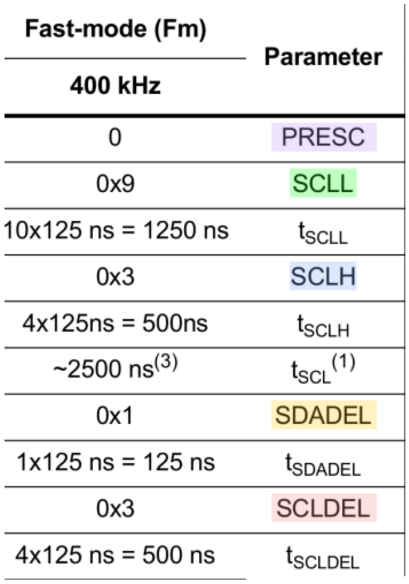
Post-Lab Questions

1. What does the AUTOEND bit in the CR2 register do? Why don’t you want to use it when you’ll be needing a restart condition?

    - When the AUTOEND bit is set, the peripheral will automatically generate a stop condition at the end of a transaction. This setting is undesirable when performing chained writes and reads (where you will need a restart condition) as it stops the I2C bus.

2. This lab used standard-mode 100 kHz I2C speed. What values would you write in the TIMINGR if we were using 400 kHz fast-mode?

    - If we were using 400kHz fast-mode we’d write the following values into the following bit fields of the TIMINGR register (first column are the values and the second column of highlighted ‘Parameter’s are the TIMINGR bits fields in which the parameters are written into):
        - PRESC = 0, SCLL = 0x9, SCLH = 0x3, SDADEL = 0x1, and SCLDEL = 0x3



Fast-mode (Fm)	Parameter
400 kHz	
0	PRESC
0x9	SCLL
10x125 ns = 1250 ns	t_{SCLL}
0x3	SCLH
4x125ns = 500ns	t_{SCLH}
~2500 ns[3]	t_{SCL}[1]
0x1	SDADEL
1x125 ns = 125 ns	t_{SDADEL}
0x3	SCLDEL
4x125 ns = 500 ns	t_{SCLDEL}

3. This lab used blocking code. To implement it completely as non-blocking you would replace all of the wait loops with interrupts. Most flags in the I2C peripheral can trigger an interrupt if the proper enable bit is set. Find the interrupt enable bits that match the following flags: TC, NACKF, TXIS (transmit interrupt), ARLO.

   - TC: **Bit 6 TCIE** of I2C control register 1 (I2C_CR1)
   - NACKF: **Bit 4 NACKIE** of I2C control register 1 (I2C_CR1)
   - TXIS (transmit interrupt): **Bit 1 TXIE** of I2C control register 1 (I2C_CR1)
   - ARLO: **Bit 7 ERRIE** of I2C control register 1 (I2C_CR1)

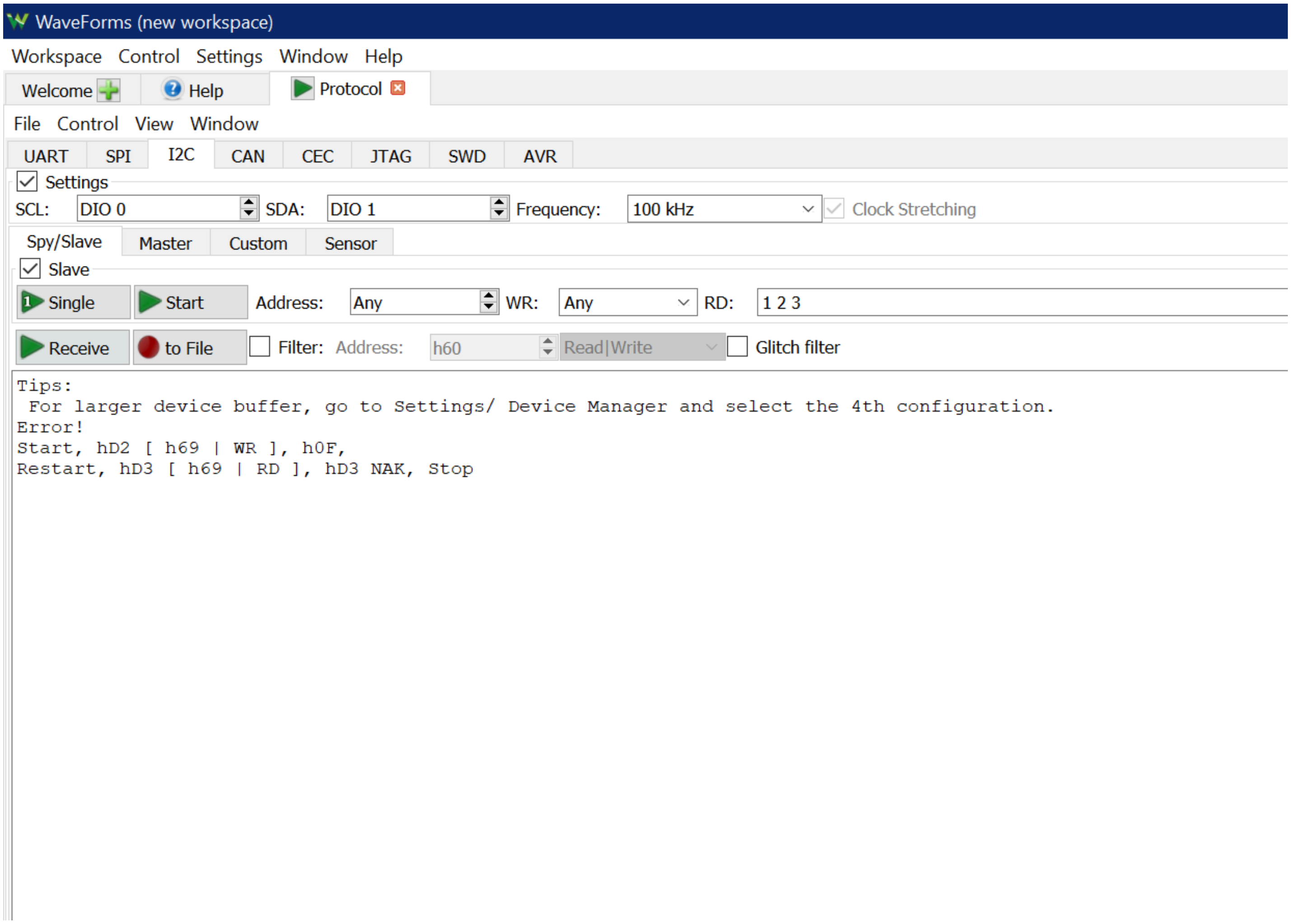
4. The gyro can operate in three full-scale/measurement ranges, measured in degrees-per-second (dps). What are these three ranges?

   - 245/500/2000 dps

5. What is the I2C address of the gyro when the SDO pin is low? The lab has the pin set high, read the I2C section of the gyro datasheet.

   - When the SDO pin is connected to ground, the LSb value is ‘0’ and so the I2C address of the gyro is 1101000b

I2C Transaction Screenshot from section 5.1:


WaveForms (new workspace)
Workspace Control Settings Window Help
Welcome
Help
Protocol
File Control View Window
UART SPI I2C CAN CEC JTAG SWD AVR
Spy/Slave Master Custom Sensor
Single
Start
Receive
to File

☑ Settings

SCL: DIO 0 SDA: DIO 1 Frequency: 100 kHz ☑ Clock Stretching

☑ Slave

Address: Any WR: Any RD: 1 2 3

☐ Filter: Address: h60 Read|Write ☐ Glitch filter

```
Tips:
 For larger device buffer, go to Settings/ Device Manager and select the 4th configuration.
Error!
Start, hD2 [ h69 | WR ], h0F,
Restart, hD3 [ h69 | RD ], hD3 NAK, Stop
```