PINS

- PB2 -> AIN0 (analog comparator)
- PB3 -> ANI1 (analog comparator), OC0 (Timer0)
- PB4 -> SS (SPI)
- PB5 -> MOSI (SPI)
- PB6 -> MISO (SPI)
- PB7 -> SCK (SPI)
- PD2 -> INT0
- PD3 -> INT1
- PD4 -> OCIB (Timer1)
- PD5-> OC1A (Timer1)
- PD 6 -> ICPI pin (input capture)

Input Capture

- ICPI pin -> PD 6
- The input capture function is used in many applications such as:
  - Pulse width measurement
  - Period measurement
  - Capturing the time of an event
- `ICF1 ->` flag in the TIFR register to see if the edge has arrived
- TCNT1 value is loaded into the ICR1 register automatically by the controller.
- ICNC1 -> Noise canceler
- ICES1 -> Edge Detector
  - **0 =** Capture on the falling edge
  - **1 =** Capture on rising edge
- ACIC
  - is a bit of the ACSR reg
  - flag to select the event source (ICPI pin or output of the analog comparator)

Analog Comparator

- it provides the trigger signal for the Input capture unit.
- two pins for analog voltage compare AIN0 (PB2) and AIN1 (PB3). AIN0 is the positive terminal whereas AIN1 is the negative terminal.
- If the voltage of AIN1 is higher than AIN0, the comparator's output is 1.

DC Motor

- DC motor moves continuously ,but stepper motor moves in steps of 1 to 15 degrees.
- In stepper motor, if we know the starting position ,we can know easily count the number of the steps the motor has moved and calculate the final position. This is not possible in DC motor .
- The max speed of a DC motor is indicated in rpm (it has 2 rpms : no-load , loaded)
- L293D motor driver will be used to reverse the direction of the current thus the direction of movement.
- The L293D motor driver uses H-Bridge circuit configuration which outputs required current to Motor.
- في الكود :
  - اخد بالي ان SW -> (PORTA & (1<<7))
  - لو بيتكلم عن move with clockwise or counterclockwise -> بحدد 3 pins as output
  - لو بتكلم عن duty cycle pulse بستخدم 1 pin as output وبشغله ب delay بنفس نسبه duty cycle وبرجع اطفيه ب delay قيمته (100- duty cycle%)

PWM

- The speed of motor depends on 3 factors : load , voltage , currant.
- The wider the pulse ,the higher the speed
- $F_{generated} = F_{oscillator} / (256*N)$ (N-> prescaler)
- $F_{time\ clock} = F_{oscillator} / N$ (N-> prescaler)
- Duty cycle in non-inverted mode = ((OCR0+1)/256) *100
- Duty cycle in inverted mode = ((256-OCR0)/256) *100

SPI

- It is full duplex.
- Extremely easy to interface (It took me much less time to setup and transmit data through SPI as compared to I2C and UART)
- Less power consumption as compared to I2C
- Higher hit rates
- Wireless transmissions through Zigbee, Bluetooth et
- In a SPI link there could as many Slaves as required
- When we connect more than one Slave devices, then we choose them using the SS' signal
- The ability to connect several devices to the same SPI-bus is based on **only one master and only one slave is active at the same time**
- Unidirectional SPI devices require just the clock line and one of the data lines
- SPIF -> is set in 2 situations :
  - When serial transfer is completed
  - When SS pin (PB4) is an input and is driven low by an external device
- SPIF is cleared by hardware when executing the corresponding interrupt handler.
- WCOL -> is set when you write on SPDR during data transfer
- Both WCOL bit and SPIF bit is cleared when you read the SPI status register and then access the SPI Data Register (SPDR)
- SPI2X ->set when SPI in master mode
- SPI control register :
  - SPIE -> enable the SPI interrupt
  - SPE -> enable SPI
  - DORD -> if set to 1 , the LSB is transmitted then MSB otherwise the MSB is transmitted then LSB
  - MSTR -> if set to 1 , you work in master mode otherwise you work in slave mode
  - CPOL -> set the base value of clock when it is idle
  - CPHA -> =0? Means sample on the leading (first) clock edge
    -> =1? Means sample on the trailing (second) clock edge
  - SPR1 , SPR0 -> control the SCK rate of the device in master mode
- There is no requirement of a pull-up resistor in the case of the SPI.

- SPI does not verify that data is received correctly or not.
- SPI -> you have to select the slave using the slave select pin for the communication

I2C

- I2C is a half-duplex communication protocol.
- Master -> device that generate the clock system, initiates and terminates a transmission.
- Slave -> node that receive the clock and the addressed by the master
- There are 4 modes of operation:
  - Master transmitter
  - Master receiver
  - Slave transmitter
  - Slave receiver
- Node can have more than one mode of operation at different times, but it has only one mode at a given time .
- I2C can be multi-master and multi-slave
- I2C is slower than SPI.
- I2C draws more power than SPI.
- I2C is cheaper to implement than the SPI communication protocol
- I2C work on wire and logic and it has a pull-up resistor.
- In I2C communication we get the acknowledgment bit after each byte.
- I2C ensures that the data sent is received by the slave device.
- I2C is the address base bus protocol, you have to send the address of the slave for the communication.
- I2C has start and stop bits.

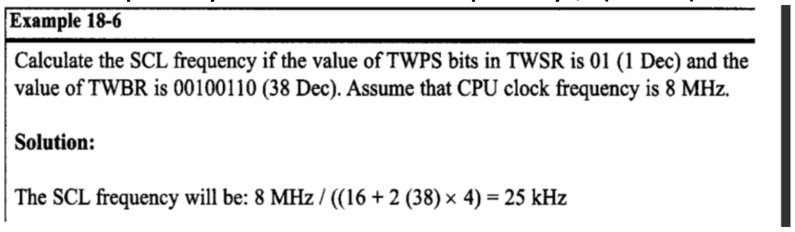
I2C	SPI
half-duplex communication protocol	Full duplex communication protocol
there can be more than one master and slave attached to the I2C bus.	there can be only one master attached to the SPI bus
I2C supports multiple devices on the same bus without any additional select lines (work on the basis of device address).	SPI requires additional signal (slave select lines) lines to manage multiple devices on the same bus.
I2C is better for long-distance.	SPI is better for a short distance
I2C has start and stop bits	SPI doesn’t have start and stop bits

I2C is the address base bus protocol, you have to send the address of the slave for the communication.	you have to select the slave using the slave select pin for the communication
I2C is cheaper to implement than the SPI	Costly as compared to I2C
I2C ensures that the data sent is received by the slave device	SPI does not verify that data is received correctly or not.
we get the acknowledgment bit after each byte.	Acknowledgment bit is not supported
I2C work on wire and logic and it has a pull-up resistor	There is no requirement of a pull-up resistor in the case of the SPI.

- The start and stop conditions are generated by master by keeping the level of SCL line high then changing the level of the SDA line.
- The start condition is generated by a high-to-low change in SDA line when SCL is high.
- The stop condition is generated by a low-to- high change in SDA line when SCL is low.
- START condition releases the bus, and other master may seize the bus.
- REPEATED START condition -> دا لو مش عايزه اسيب bus علشان محتاجاه في transmit تاني لازم يتم بعد اللي شغال فلو عملت stop ممكن ع ما ارجع اعمل start الاقي slave تاني اخد مني bus فدي بتخليني ضامنه ان bus بتاعي لحد ما اخلص
- EEPROM -> بحرق عليه Program ، I2C مربوط بيه عند address (0x34, 0x35)
- Conditions to generate NACK:
    - The receiver is unable to receiver transmit as it is performing some real-time function and isn’t ready to start communication with the master.
    - During the transfer, the receiver gets data or commands that it doesn’t understand.
    - During the transfer, the receiver cannot receive any more data bytes
    - A master-receiver is done reading data and indicates this to the slave through a NACK.
- Address packet -> 9 bits:
    - 7 address bits
    - 1 bit -> READ/WRITE control bit
    - 1 bit -> Acknowledge
- Address bits are used to address specific slave device on the bus
- 119 devices can share an I2C bus
- The address 0000000 is reserved for general call (1)

- All addresses of the format 1111xxx are reserved (8 addresses)
- In I2C bus the MSB of address is transmitted first
- Write to one register in a slave:
    - Start bit
    - Slave address -> 7 bits
    - R/W bit -> 0 (if w) / 1 (if R)
    - Ack bit (slave)
    - Register address -> 8 bits
    - Ack bit (slave)
    - Data byte to register -> 8 bits
    - Ack bit(slave)
    - Stop bit
- Multiple burst write - > we provide the address of first location then the data for that location
- Clock stretching :
    - it kind of flow control
    - if an addressed slave device is not ready to process more data, it will stretch the clock by holding the clock line (SCL) low after receiving or sending a bit of data
    - master will wait until the slave releases the SCL line to show it is ready to transfer the next bit
- Arbitration -> معناها مين هيفوز ب bus دا بيحصل لما 2 master دخلوا في نفس اللحظه وعملوا start
- طب بيختار ع أساس ايه ؟
    - الكبير هينحسب ويتحول ل slave mode ويسيب bus
- طب هو بيعرفهم امتى ؟
    - لو عندي مثلا 001111 و 000111 كل واحد فيهم هفضل ينزل Bit بتاعته يعني 0 ثم 0 ثم اول واحد هينزل 1 والتاني 0 فهلاقي قيمتين فاللي هينزل 1 هينسحب (اللي هو الكبير)
- Reading from slave:
    - Start bit
    - Slave address -> 7 bits
    - R/W bit -> 0 (w)
    - Ack bit (slave)
    - Register address -> 8 bits
    - Ack bit (slave)
    - Repeated start bit
    - Slave address -> 7 bits
    - R/W bit -> 1 (R)

  - Ack bit (slave)
  - Data byte to register -> 8 bits
  - NACK bit (master)
  - Stop bit (master)
- **(47 , 46) reading from slave , multiple burst read ممكن تطلب رسم**
- TWSR:
  - TWPS -> control the bit rate prescaler (2 bits)
  - TWS -> show the status of TWI control and bus (5 bits)
- SCL Frequency = CPU clock frequency / (16 + (2*TWBR*4^{TWPS}))



Example 18-6

Calculate the SCL frequency if the value of TWPS bits in TWSR is 01 (1 Dec) and the value of TWBR is 00100110 (38 Dec). Assume that CPU clock frequency is 8 MHz.

Solution:

The SCL frequency will be: 8 MHz / ((16 + 2 (38) × 4) = 25 kHz

- TWCR:
  - TWINT -> set by hardware when TWI module has finished its current job
  - TWEA -> enable generation of ACK
  - TWSTA -> generate start condition if bus is free
  - TWSTO -> generate stop condition (**cleared by hardware**)
  - TWWC -> access The TWI data register when TWINT is low (**cleared by writing to TWDR register when TWINT is high**)
  - TWIE -> enable TWI interrupt if general interrupt is enabled

Servo

- Servo Motor is a DC Motor equipped with error sensing negative feedback to control the exact angular position of the shaft.
- Unlike **DC Motors** it will not **rotate continuously**. **Servo** is used to make **angular rotations** such as 0-90° , 0-180°
- Stepper Motors can also be used for making precise angular rotations. But Servo Motors are preferred in angular motion applications
- It needs no extra drivers like stepper motor and only angular motion is possible
- the angular position is determined by the width of the pulse at the control input

- Angular range and control pulse width are different for different servo motors
- The torque rating of the servo is the maximum amount of force the servo can exert.

Stepper Motor

- It is a Digital motor because it operates on pulses.
- It has excellent position control and hence can be used for precise control application.
- it has high life time than normal DC or servo motor
- stepper motor **rotates in steps**
- When a coil gets energized it acts as a magnet and the rotor pole gets aligned to it. When the rotor rotates to adjust itself to align with the stator it is called as one step.
- Modes of operation in Stepper Motor:
    - Full Step Mode: achieve a full 360° rotation with minimum number of steps
        - One phase-on stepping: only one terminal (phase) of the motor will be energized at any given time. This has less number of steps
        - Two Phase-on stepping: since two coils are energized at a time it can provide better torque and speed compared to the previous method. However it consumes more power.
    - Half Step Mode: combining both the methods we will have to perform 8-step
- The speed of rotation can be changed by changing the rate at which the control signals are applied.