CmpE242

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \quad \ldots (1)$$

March 10 (Wed)

$$G(x,y) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad \ldots (2)$$

Note $\mu_x = \mu_y = 0$, $\sigma_x = \sigma_y = \sigma$

Fig 1.

From Eqn (4), Plot from the github 2018S-15-Lec6-V3 ...

Let $y=0$, to have one indep. variable x.

$$\nabla^2 G(x,y)\Big|_{y=0} = \nabla^2 G(x,0) \quad \ldots (3)$$

$$-\frac{1}{\sqrt{2\pi}\,\sigma^3} e^{-\frac{x^2}{2\sigma^2}} + \frac{x^2}{\sqrt{2\pi}\,\sigma^5} e^{-\frac{x^2}{2\sigma^2}}$$

LoG(x), or $\nabla^2 G(x,0)$

Example: (1) Use LoG(x) to Build a Convolutional Kernel (2) to Compute Derivatives of the Error

22

Note: LoG(x) is NOT Exactly the Computation for derivatives, But we use it, for its Low pass feature, and 2nd order derivative.

Sol.

(1) "Mapping" to a Kernel

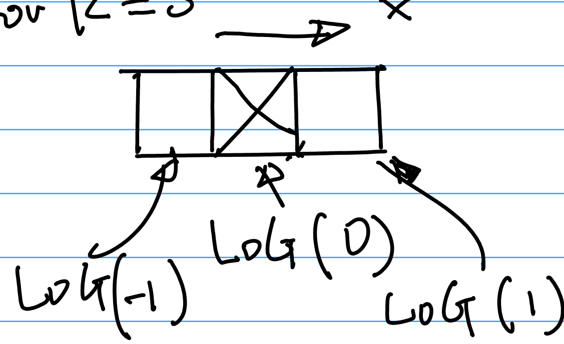
Build a kernel with "odd" Number of grids, elements

$K \times 1$

No. of elements

One Row

for $K = 3$



(a) Identify the Center Reference

(b) from LoG(x) (or $\nabla^2 G(x,0)$), map it to the Kernel

Solve for y...

Driver Implementation $\begin{cases} f_{PWM} \\ D.C. \end{cases}$

2018S-10-0 ~
PWM Driver

Add Duty Cycle Function to Device Driver.

$\begin{cases} \text{Theoretical Aspect} \\ \text{Implementation C Code.} \end{cases}$

SPRs $\begin{cases} \text{TCON} \\ \text{TCNT} \\ \text{TCMP} \end{cases}$

CFG0

CFG1

Root

Control Reg.

Counter Reg.

Compare Reg.

Prefix "Timer"

PWM Output Square Waveform

T

$\begin{cases} \text{TCNTB} \\ \text{TCmpB} \end{cases}$

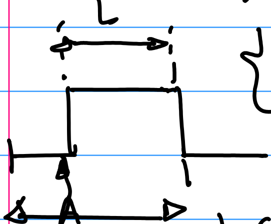


Fig. 1

$T_{PWM} = 1/f_{PWM}$

$$f_{PWM} = \frac{CLK_P}{(\text{Prescaler}+1)(\text{divider})} \quad \dots (3)$$

PP 1118, CPU Datasheet

Prescaler: 8 bit, [0, 255]

Divider: 1, 2, 4, 8, 16

Note CFG / CON are responsible for Setting Prescaler / Divider value.

$$f_{PWM} = \frac{50\times10^6}{(Pres+1)\cdot Div.} \quad \dots (*)$$

If we need $f_{PWM} = 2\times10^3$

Find SPR, Set SPR to Realize this frequency.

From CPU Datasheet CFG0 CON.

Sol

$$2\times10^3 = \frac{50\times10^6}{(Pres+1)Div}$$

$$(Pres+1)Div = \frac{50\times10^6}{2\times10^3}$$

$$(Pres+1)Div. = 25\times10^3$$

Let Div = 16,

Solve for Pres.

$$Pres+1 = \frac{25\times10^3}{16}$$

$$Pres = \frac{25\times10^3-16}{16} \simeq \; > 255$$

Iteration,

Change PCLK to 10MHz,

then, we have

CMPE/EE242

$$Pres = \frac{10\times10^6 - 16\times2\times10^3}{16\times2\times10^3}$$ if it is still too big

therefore, then Low the CLK_p

try $CLK_p \simeq 2\times10^6$. please verify it!

Note: SPR Responsible for f_{PWM}

T CNT B — Timer, Cnt, Buffer

$f_{PWM} = 1\times10^3$ given.

f ~~master~~ Clock peripherial

N Counts for CNT SPR.

$f = f_{PWM} \cdot N$ ...(4)

Given, Target, Unknown to be Calculated

Note: T Cmp B — Comparison Register for Duty Cycle

2nd Counts Value for "Cmp" Derived from Duty Cycle.

March 17 (Wed) f_{PWM} By Setting SPR's value; Duty Cycle

Arm11 Datasheet

T CNT B ← "N" Counts ↕ f_{PWM}

T Cmp B

2018S-10-

$f_{PCLK}/f_{PWM} = N$ ...(1)

Define the period; CNT

Duty Cycle → % → Counts → Cmp

Percentage

GPFCON PP.522

GPFCON[29:28] = 10 → PWM

GPFCON[31:30] = 10 → PWM

#define S3C64xx_GPFCON 0x7... *.h

&= ~(0x3U<<28)

"AND" Neg. "11" "00"

|= (0x2U<<28)

"OR" "10" unsigned

Set 2 Bits GPFCON[29:28] = 10

March 22nd (Mon)
Review.

1° 3± Questions.

a Basic Concepts CPU Architecture, Block Diagram,
Memory Map, Peripheral Controllers

GPP (GPIO) } SPRs: CON, DAT, ...
PWM

PWM: TCNTB, TCMPB, CFG0, 1

Architecture → Mem → SPR

Code

- User Space Programming
- Kernel Space Device Driver

Kconf Script ←

Define Compilation + Build Process.

Programming Requirements, No Program Code Writing

However! Debug/Change the existing Code is Needed;

b Design Related Question(s)

SCH. Design, CKT for PWM

Pin(s), f_{PWM} — GPIO

Motor Drive

Pin(s), Label(s) Stepper motor I/F

GPP I/O Testing ("Hello, the World")

- Input Testing CKT
- Output Testing CKT

Resistance Value Calculation

c Theoretical Aspects

C1. PID Controller Design
Basic Concepts
Block Diagram.

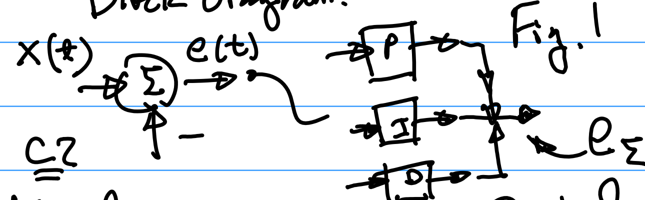


Fig. 1

C2 Kernels
F.D. 2x1 → Central D.
B.D. 2x1
$\frac{1}{2}(F.D + B.D)$

With Noise Reduction 3x1

Low Pass Filter: G(x) Gaussian.

+ 2nd Order Derivation as in Computer Vision

∇^2: Laplacian $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \rightarrow$ 1D $\frac{\partial^2}{\partial x^2}$

LoG(x)

Note: One page formula sheet is Allowed, However No verbal Description And/OR Examples Allowed.

Note: Calculator Is Allowed.

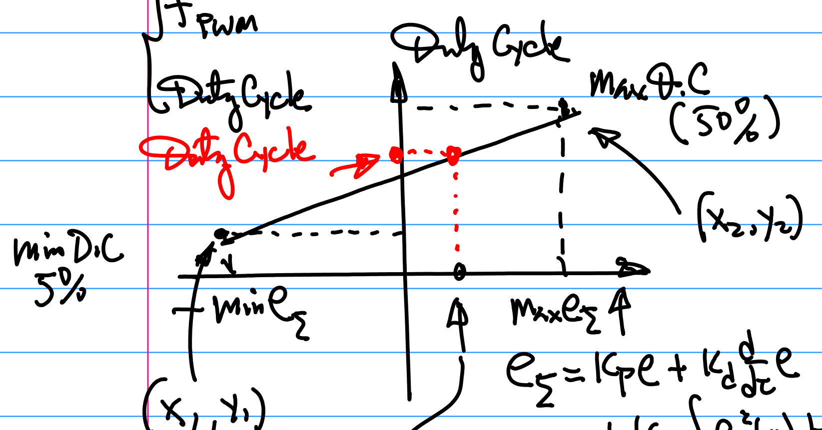
Close Book, Close Notes
Datasheets if needed will be provided;

Convolution with Kernel(s)
Table of E(t), find $\frac{d}{dt}E(t)$ Convolution

$$\int K_I e^2(\tau)d\tau \approx \sum_{i=0}^{I} K_I e^2(\tau)$$

Mapping to Control function PWM

$\begin{cases} f_{PWM} \\ \text{Duty Cycle} \end{cases}$



$e_\Sigma = K_p e + K_d \frac{d}{dt} e + K_I \int_I e^2(\tau)d\tau$

$e_\Sigma = 0.57$

find Control Action, → D.C = ?

Implementation: Reference Platform ARM11

Architecture → Memory map.
CPU Block Diagram

2^{32} (4 GB)

8 Equal BANK

↓ 3 Addr Bits

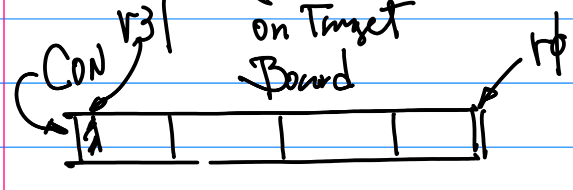
$a_{31} a_{30} a_{29}$

* SPRs. ←
(Peripheral Controllers)

$\begin{cases} \text{PWM} \\ \text{GPP} \end{cases}$

GPx CON
GPx DAT
T CNT B
T Cmp B

Design Spec. → SPRs → CPU Datasheet
(Pins)
on Target Board



To perform init & Config:
1° Binary Pattern for SPR.

Read/modify user Application Programs | Kernel Space Device Driver Program.

C Code for this purpose.

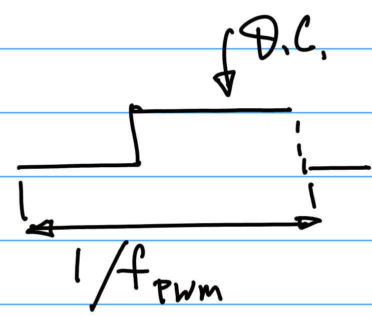
Note: PWM Waveform, e.g. Duty Cycle Calculation.

N. for CNT
N' for Cmp
T B

$f_{PCLK} / f_{PWM} = N$

$$f_{PWM} = \frac{PCLK}{(Prest+1) * DVR}$$

$\%(D.C) * N = N' \rightarrow Cmp$



Pre-requisit
$\begin{cases} 1° \text{ O.S Source Distribution} \\ 2° \text{ Tool Chain Distro. "Cross Comp"} \\ 3° \text{ Datasheet.} \end{cases}$

↓

Tool Chain Installed Running

make menuconfig

Continued → Kconf (at \drivers ~\char)

↓

Script. Add your Device Driver

↓

make menuconfig

↓

invoke your change, Compile & Build (Module Only for Simplicity Purpose)

↓

Object "KO".

↓

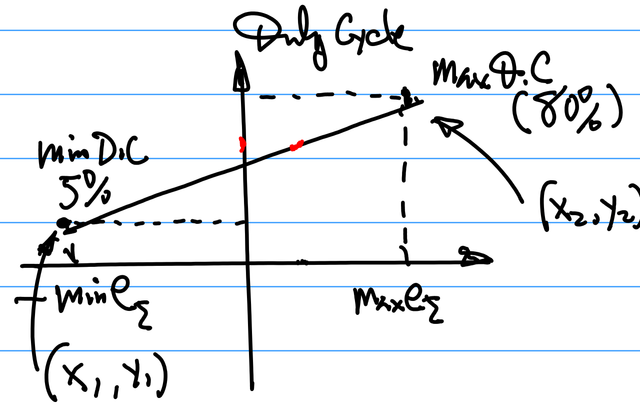
Copy "USB", upLoad by "CP" Copy Command to your target

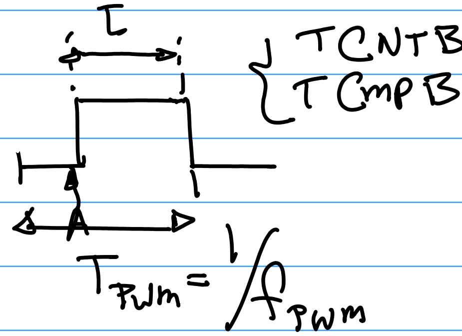
↓

"insmod" mytest.KO (To make it as a part of Kernel Image)

↓

run your user application program (By calling the module)





https://github.com/hualili/CMPE242-Embedded-Systems-/blob/master/2018S-16-AngularSensing-i2c-L8

CMPE/EE242

April 5th (Monday) 32

1. midterm Graded, the key was posted online, github, search under folder 2019S, "key"
2. 2nd half of this course. IIoT (Industrial IoT)

Sensors I/F { Digital Sensors — I2C I/F.
Analog Sensors — ADC

FFT to find/characterize Analog Sensor Data (Nyquist Theorem)
Validation of Sensor Data.
OpAmps to Build Processing CKT.
"SPICE" Simulation.

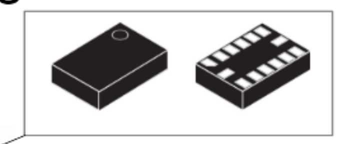
Example: LSM303

Note: Next Project use LSM303.

CMPE242-Embedded-Systems- / 2018S-16-AngularSensing-i2c-LSM303- final HL 2017-3-13.pdf

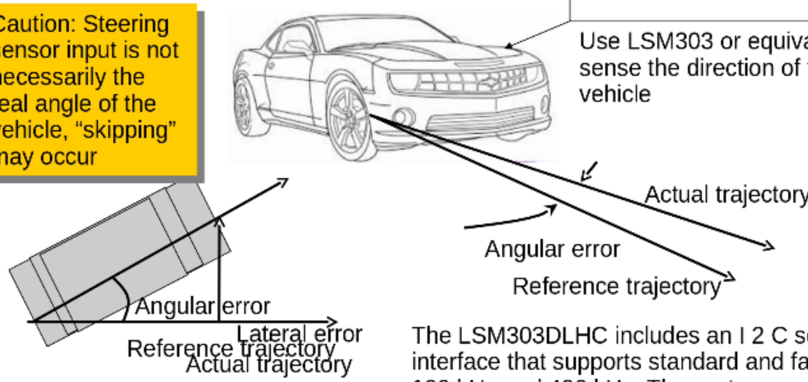
Sensors for Driving Direction and Turning Angle

eCompass module:
3D accelerometer and 3D magnetometer



Use LSM303 or equivalent to sense the direction of the vehicle

Caution: Steering sensor input is not necessarily the real angle of the vehicle, "skipping" may occur



The LSM303DLHC includes an I 2 C serial bus interface that supports standard and fast mode 100 kHz and 400 kHz. The system can be configured to generate interrupt signals by inertial wake-up/free-fall events as well as by the position of the device itself.

larry Li, Ph.D. April 2015

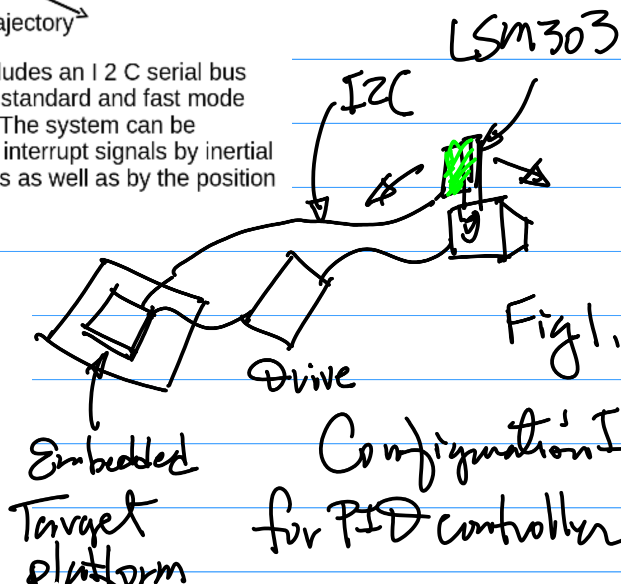
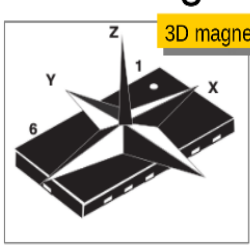


Fig 1. Configuration I for PID controller

3D Accelerometer and 3D Magnetomete LMS303

3D magnetometer



3D accelerometer

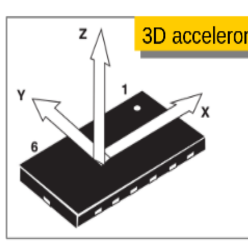


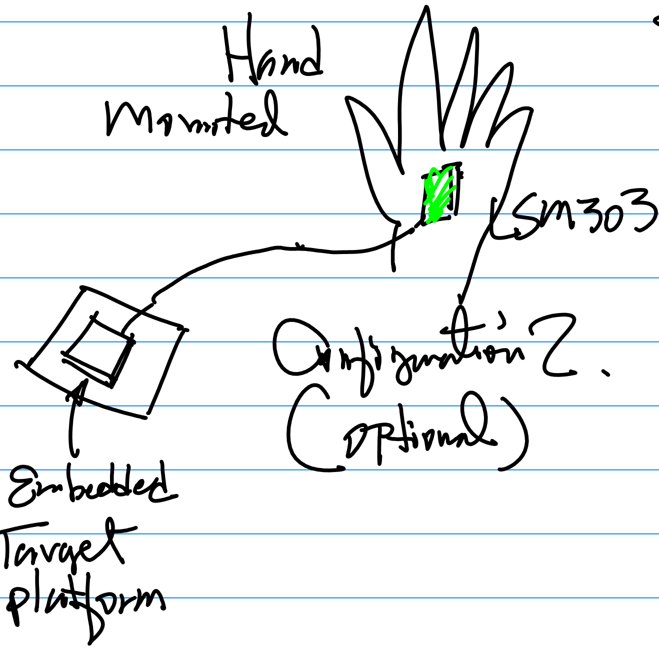
Table 9

Pin name	Pin description
SCL	I^2C serial clock (SCL)
SDA	I^2C serial data (SDA)

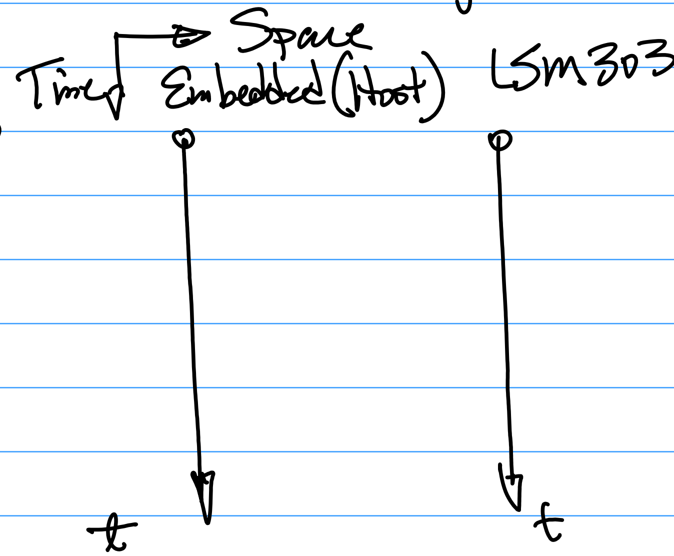
I2C Interface

(1) The transaction started through a START (ST) signal, defined as a high-to-low on the data line while the SCL line is held high.
(2) After ST, the next byte contains the slave address (the first 7 bit), bit 8 for if the master is receiving or transmitting data.

Homework: Implementation I2C LSM303 Sensor I/F. Due April 16 (Fri)



3. a. Space-Time Diagram



b. To Describe "Hand-Shaking"

Three Small Steps.

Step 1. → Step 2 → Step 3

Host Command for the Target via Address for Init & Config

Slave "Ack"

Data Transmission will start

* Be sure read Datasheet to map the Steps of the I/F to Space-Time Diagram.

3. Datasheet Table 9 & 11. pp. 20.

Notation:

1. ST → SP

"Start" "Stop"

A Frame

Submission On Canvas

Objective: ① To be Able to Read Sensor Input,

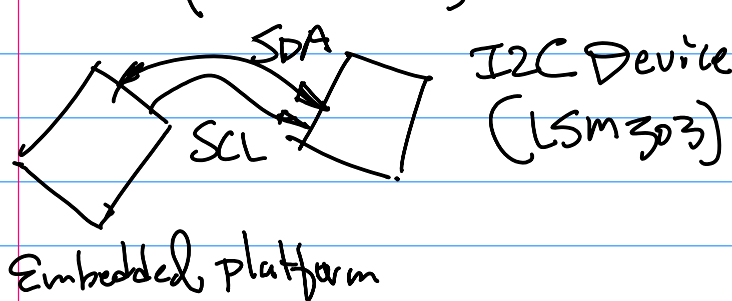
② To be Able Config the Sensor.

1. Note: LSM303 for ST-micro

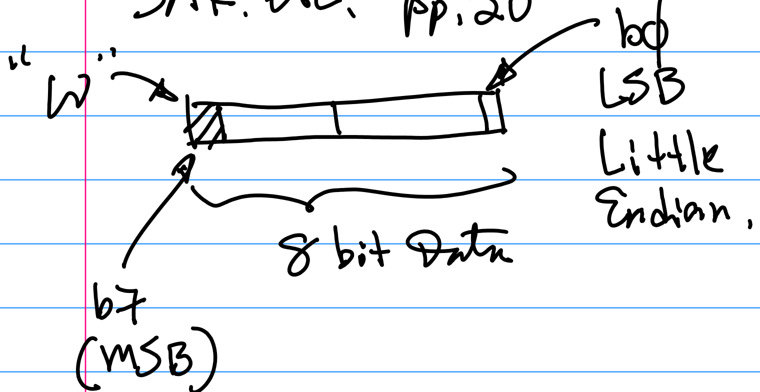
Sensor Supports
- Acceleration X-Y-Z axis
- Magnetometer
- Temperature

2. I2C

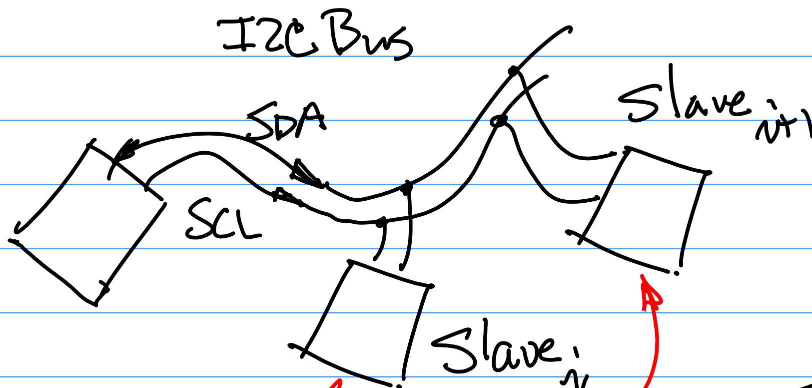
- SDA (Serial Data) Bidirectional Data;
- SCL (Serial Clock)

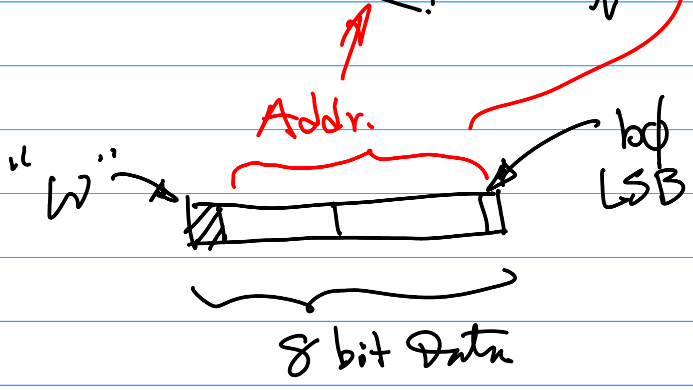


2. The Notations in Table 11
SAD, SAD+W, SUB, DATA,
SAK. etc. pp.20



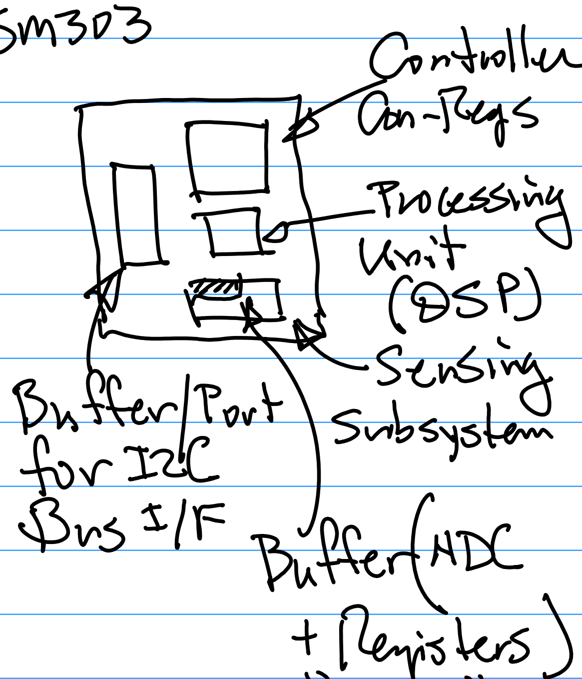
3. from pp.20 (Datasheet)





4. 1st Address (7 bits) lower, 2nd Address SUB.
All from the host (Target platform)

Consider A slave device LSM303



2nd Address "SUB" is for Identifying the target inside the Slave Device.

5. 127 Devices Possible (Theoretically) on I2C Bus, In Reality this has to be Checked By "FAN-IN" "FAN-OUT"

128 Internal Addresses → Special Purpose Registers.