ECSE 426
I^{2}C, Accelerometer, Interrupts

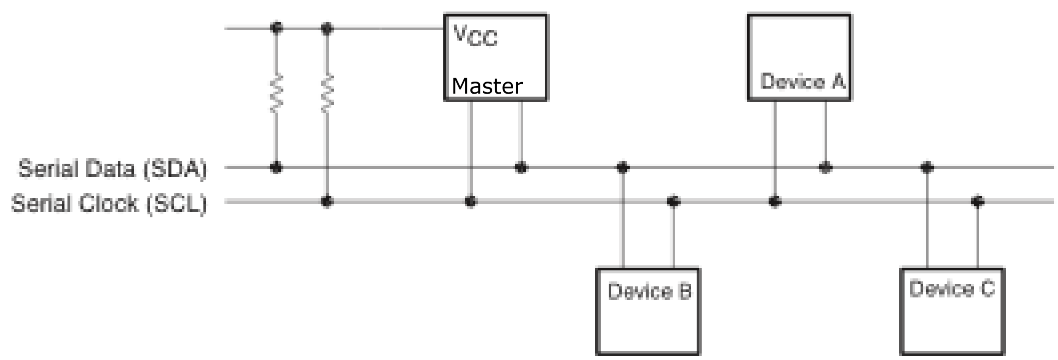
Zeljko Zilic

Room 546
McConnell Building
zeljko@ece.mcgill.ca
www.macs.ece.mcgill.ca/~zeljko

McGill

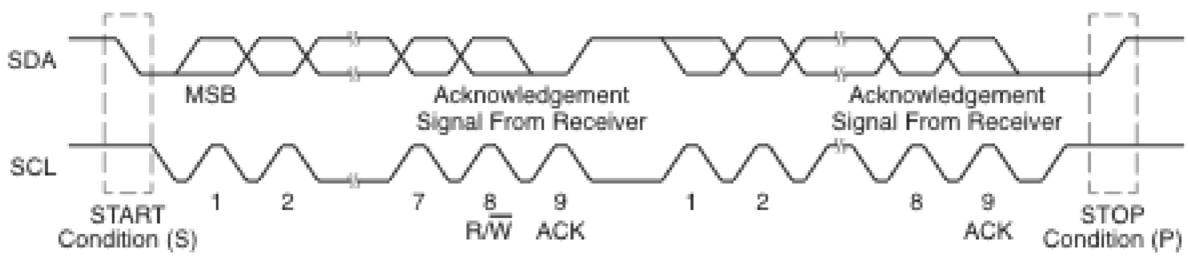
I^2C Standard

- Introduced by Philips
- Only two wires for bidirectional transmission (SDA) and clock (SCL)
- Mutidrop configurations as well
  - Destination encoded in data stream



I^{2}C Data Movement

- Clock toggled for each transferred bit
- Start (S) and stop (P) bits: data changes while SCL is kept high
  - Within sent characters data changes only while SCL low
- ACK sent by a receiver during 9th bit



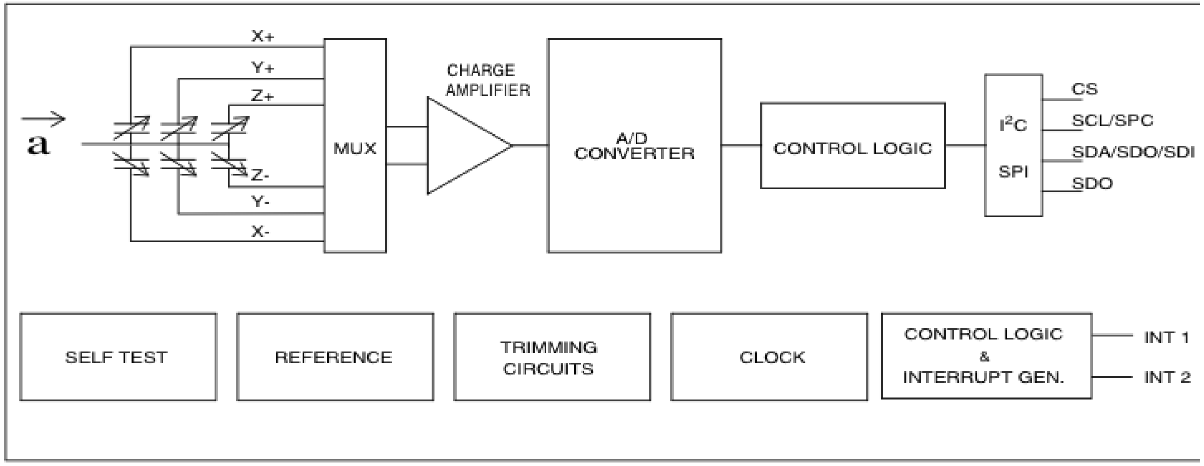
I^{2}C Addressing

- Packet encapsulated between S and P bits
- Address character followed by one or more data characters
  - R/W operation encoded as a bit immediately following address field
- Can send 7 or 10 bit address fields
  - Uses two bytes for latter; three MSB bits preceded by “11110” in the first byte (unique decoding)
- Master can “glue” several transmissions to different slaves by repeating S bits (repeated Start)

I^{2}C Arbitration

- Multiple masters can attempt transmission
- Simple arbitration that is data-dependent
- Wired-OR (more correctly, wired-AND) signal
  - If sent “1” and “0”, the line becomes zero
- Sender noticing mismatch on the line backs off
  - Sent “1”, but “0” observed on line
- Arbitration favours transmitters sending lower binary value
  - Priority by address field
- Complication: repeated Start during arbitration

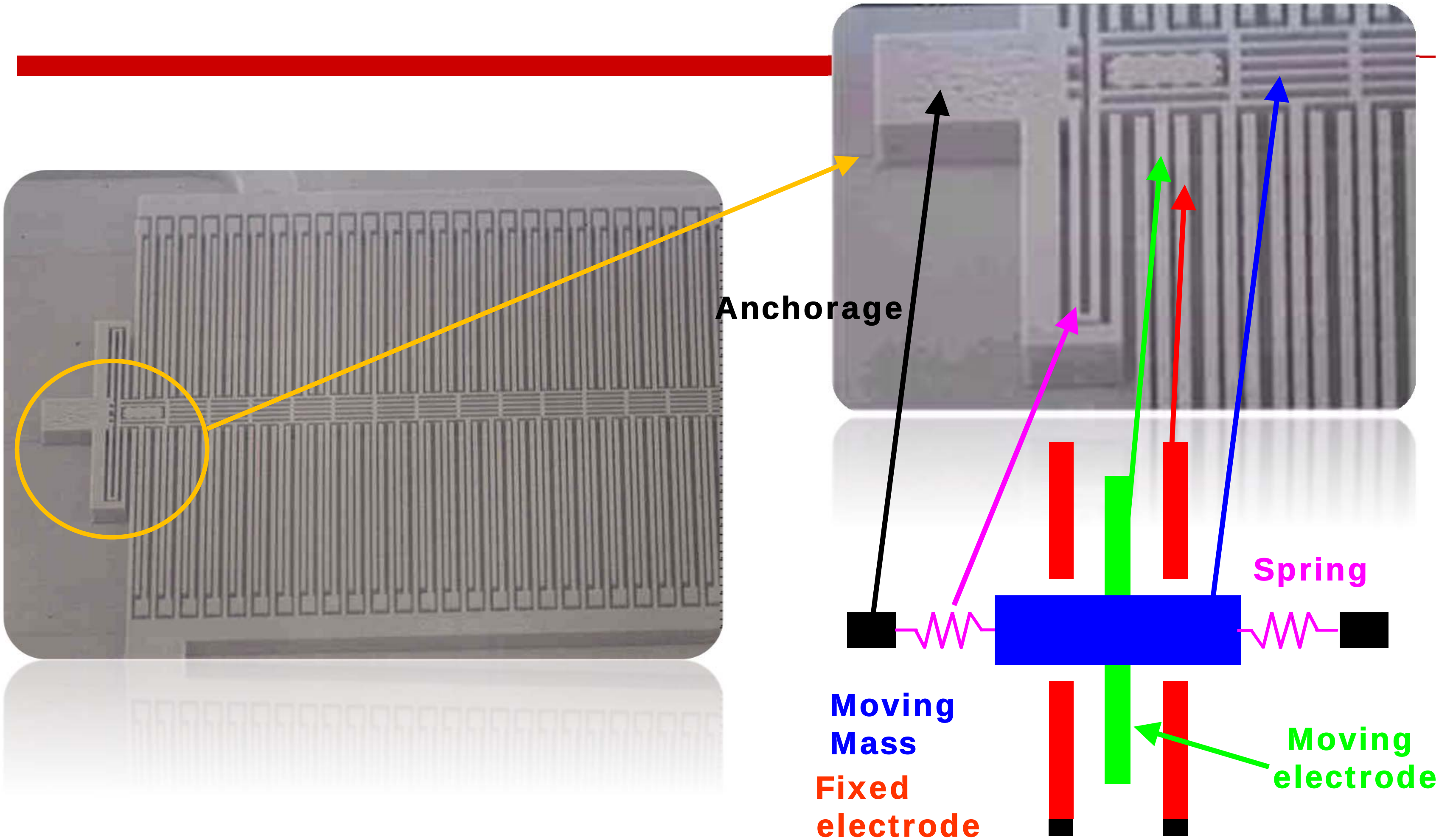
Accelerometer Operation



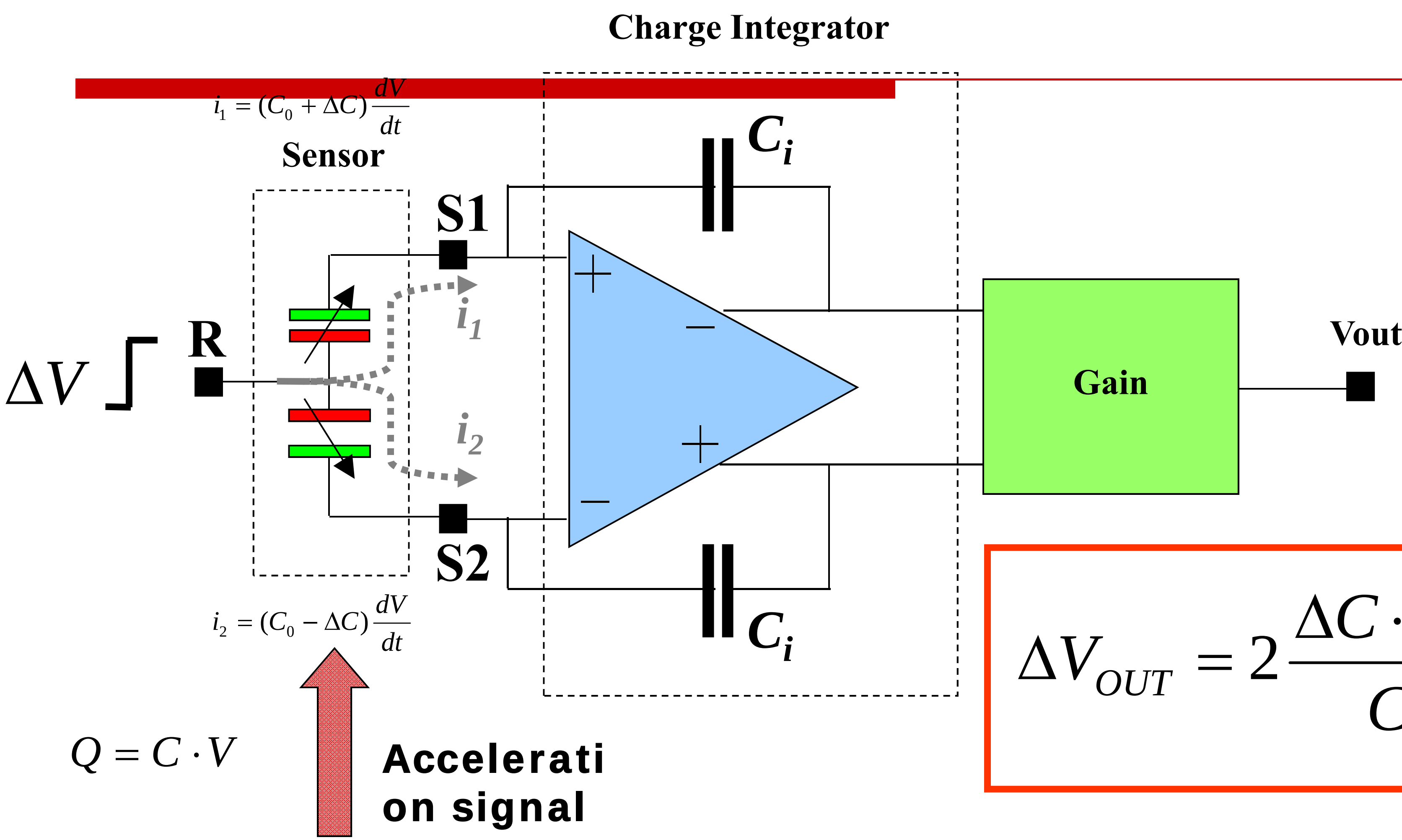
- 3 sensing parts
- SPI or I^2C interface

- Readout and control: user registers
  - Readout regs for accelerometer along X, Y, Z axes
  - Registers for specialized recognition: freefall, clicks/taps, FSMs for complex movement sequences
- Setting via control registers
  - 50 Hz sampling, axes enable, powerdown, filter, interrupts
  - Status regs: data available, overrun bits
- Can filter in SW (e.g., moving averages)

Linear 1-axis Accelerometer



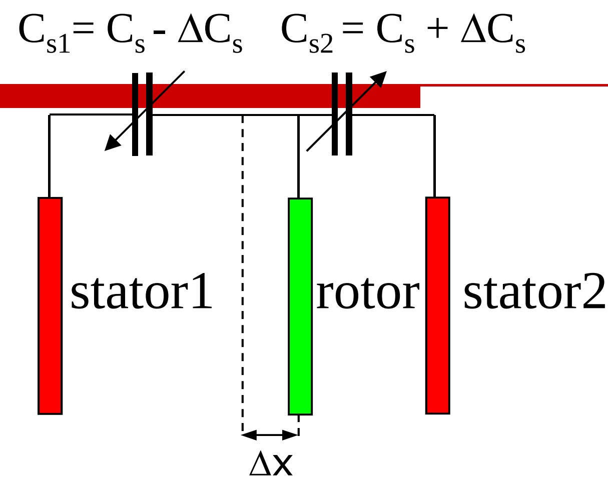
Measurement Chain



$$Q = C \cdot V$$

$$\Delta V_{OUT} = 2\frac{\Delta C \cdot \Delta V}{C_i}$$

Differential structure

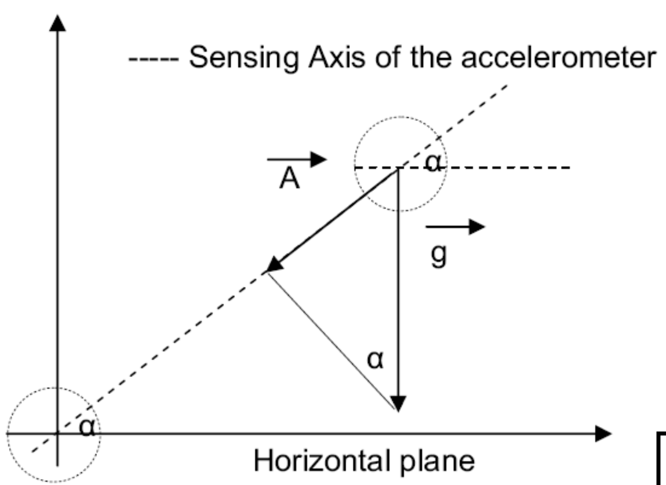


$$C_{s1} = C_s + \Delta C_s \qquad C_{s2} = C_s + \Delta C_s \qquad \boxed{\Delta C_{s12} = C_{s1} - C_{s2} = 0}$$

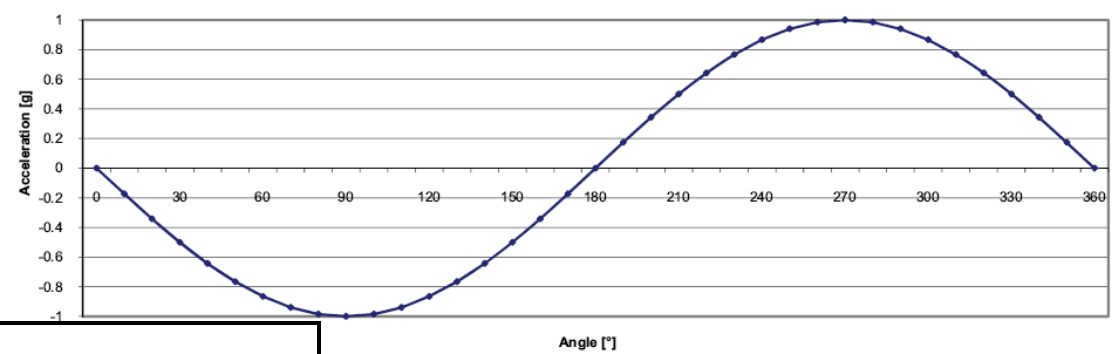
$$C_{s1} = C_s - \Delta C_s \qquad C_{s2} = C_s + \Delta C_s \qquad \boxed{\Delta C_{s12} = C_{s1} - C_{s2} = -2\Delta C_s}$$

Application: Tilt Calculation

- Accelerometer measures the projection of the gravity vector on the sensitive axis
- Amplitude of the sensed acceleration is the sine of the angle α between the sensitive axis and the horizontal plane



Single axis sensing along 360° rotation



$$\boxed{\alpha = \arcsin\left(\frac{a}{g}\right)}$$

McGill

Artificial Horizon on DSC - SLR

Artificial Horizon on DSC – SLR guarantees perfect horizontality while taking pictures, and requires a high level of accuracy (±2°)

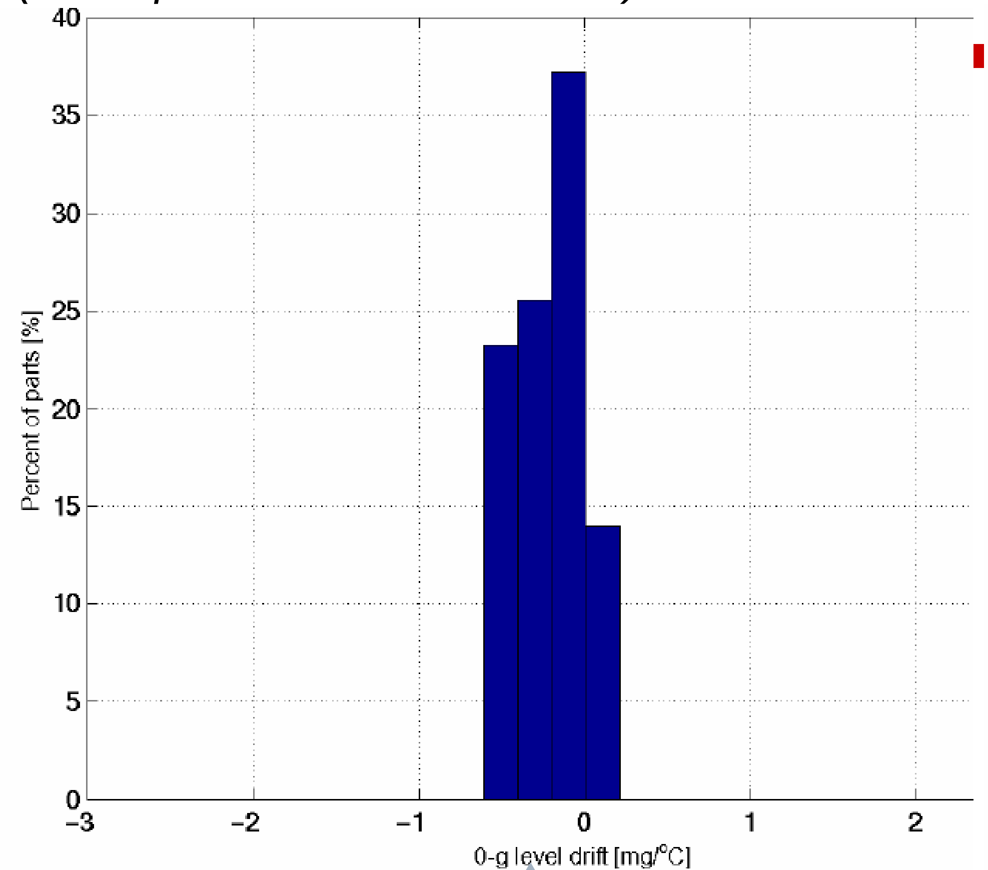
Accurate Tilt Calculation

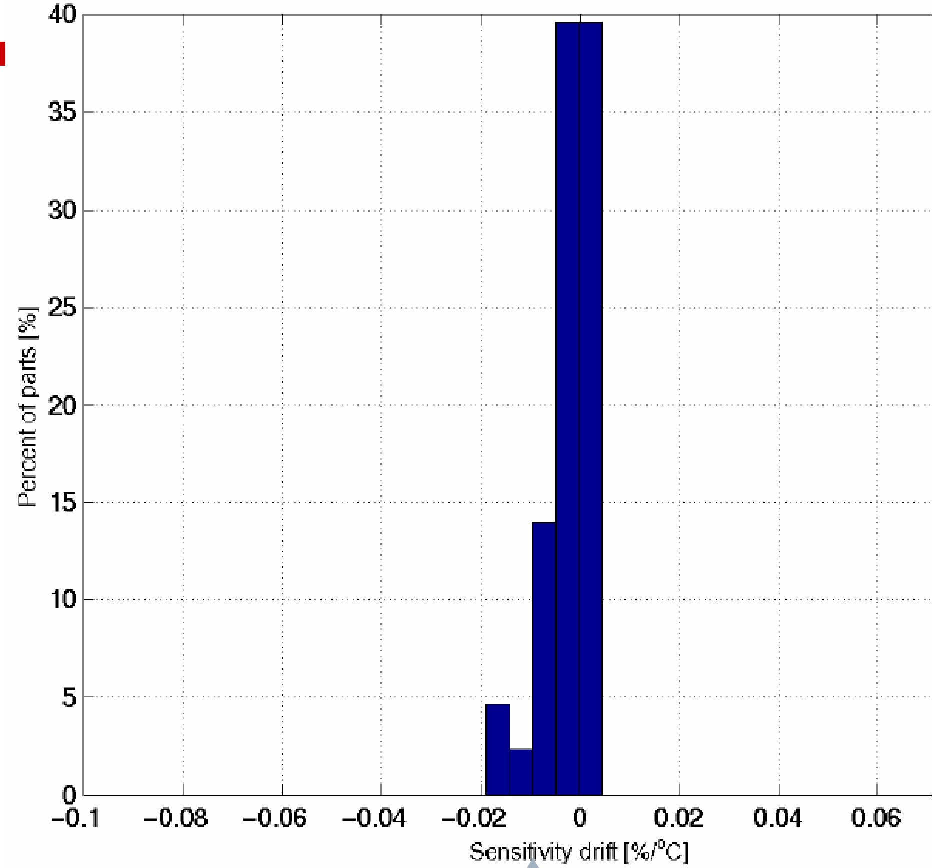
- To get accurate tilt calculation (±2°) using MEMS accelerometers, some non-ideal conditions have to be taken into account and, if needed, compensated
  - Zero-g level offset and Sensitivity accuracy
  - Non-linearity
  - Cross-axis sensitivity
  - Offset and Sensitivity change over temperature

McGill

Zero-g offset and Sensitivity vs. temp.

(Example with 8 bit device…)





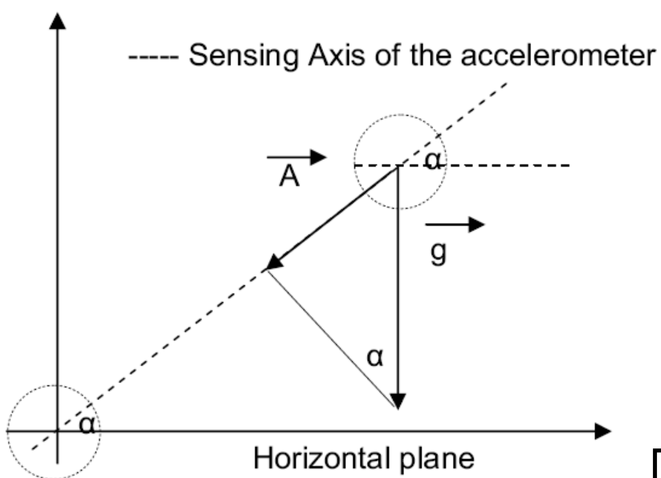
0.1mg/° C * (65° C – 25° C) = 4mg ≈ 0.22°

0.01%/° C * (65° C – 25° C) = 0.4% ≈ 0.004°

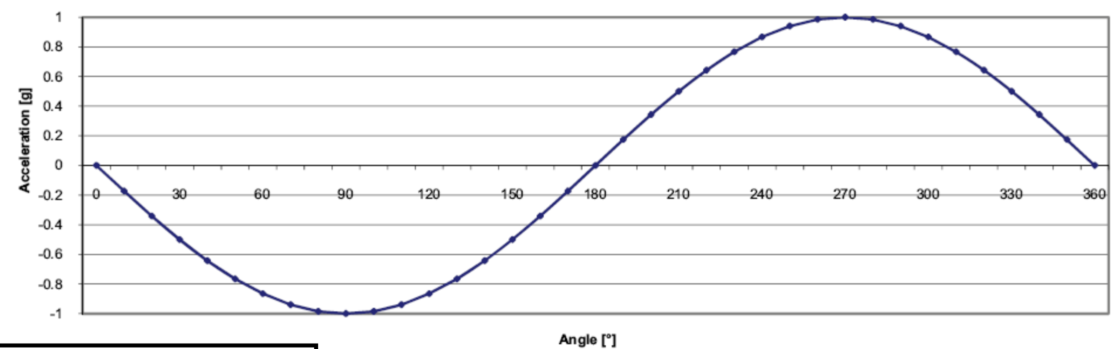
McGill

Recall the Basic Concept…

- Accelerometer measures the projection of the gravity vector on the sensitive axis
- Sensed acceleration proportional to sine of the angle α between the sensitive axis and the horizontal plane



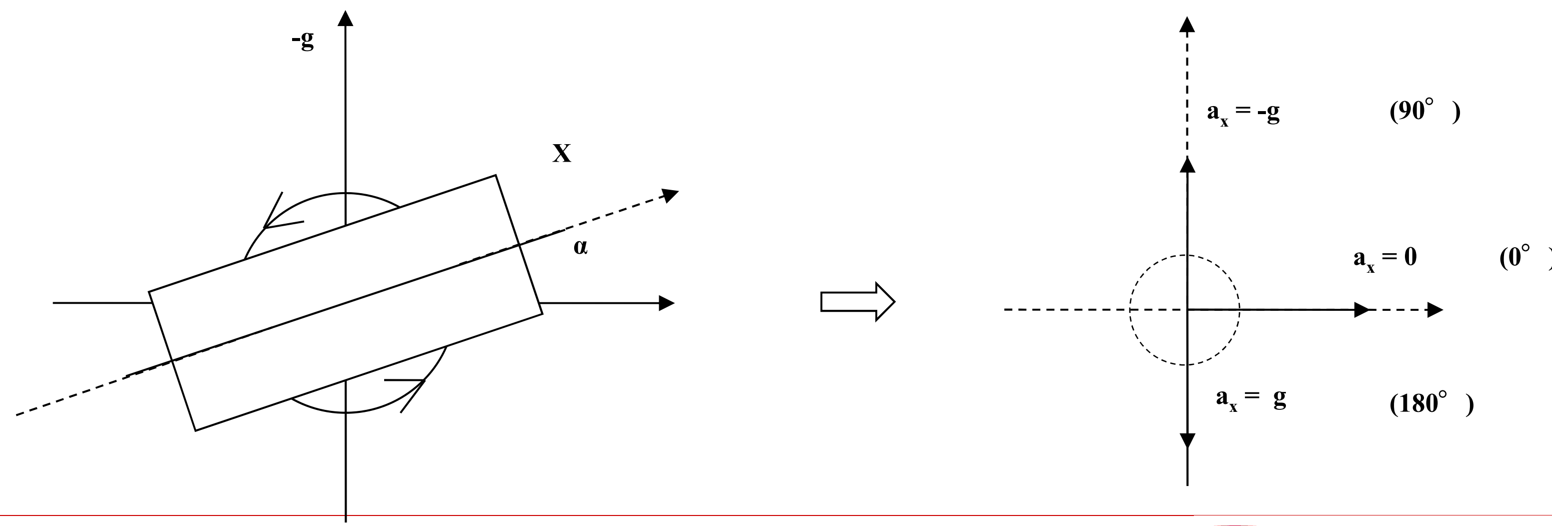
Single axis sensing along 360° rotation



$$\alpha = \arcsin\left(\frac{a}{g}\right)$$

Sensitivity Variation with Angle

- When the sensitive axis is perpendicular to the force of gravity the sensitivity is approximately:
  17.45mg/° = [sin (1°) – sin (0°)]
- Due to the derivate of *sin* function, sensor is less responsive to tilt angle changes when the sensing axis is close to ±1g
  0.15mg/° = [sin (90°) – sin (89°)]

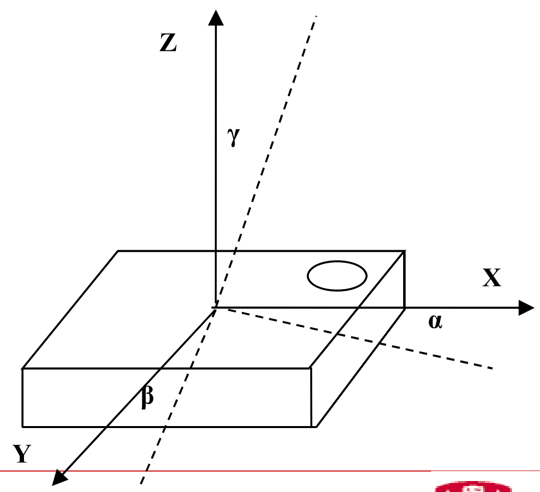


3D Tilt Calculation

- To measure the tilting independently of 3D space required to use 3-axis linear accelerometer, need to sense the vector of gravity along X,Y,Z axes
- Trigonometric means to express the angles α & β as function of a_X, a_Y, a_Z:

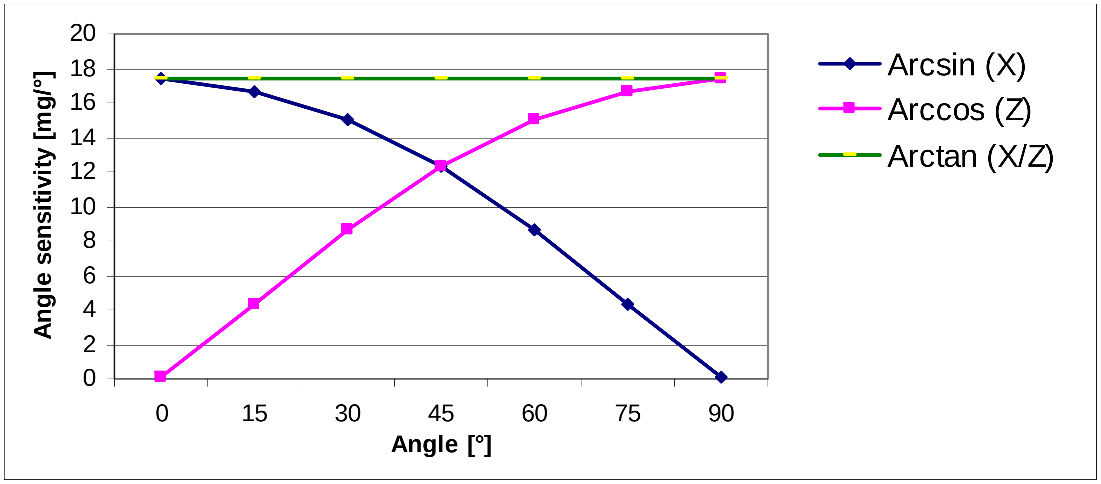
$$\alpha = \arctan\left(\frac{a_X}{\sqrt{(a_Y)^2 + (a_Z)^2}}\right)$$

$$\beta = \arctan\left(\frac{a_Y}{\sqrt{(a_X)^2 + (a_Z)^2}}\right)$$



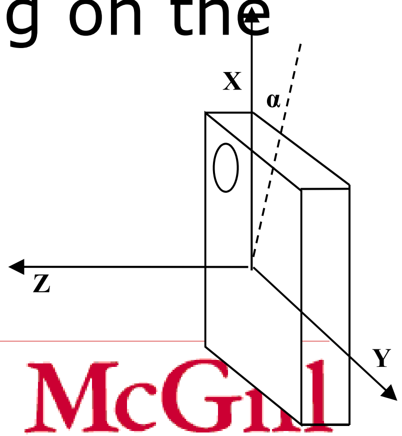
Constant Sensitivity

- Thanks to this approach, sensitivity can be kept constant along a 360° rotation



- The same concept applies to 2-axis tilting on the vertical plane considering $a_Z = 0$.

$$\alpha = \arctan\left(\frac{a_X}{\sqrt{(a_Y)^2 + (a_Z)^2}}\right) \longrightarrow \alpha = \arctan\left(\frac{a_X}{\sqrt{(a_Y)^2}}\right)$$

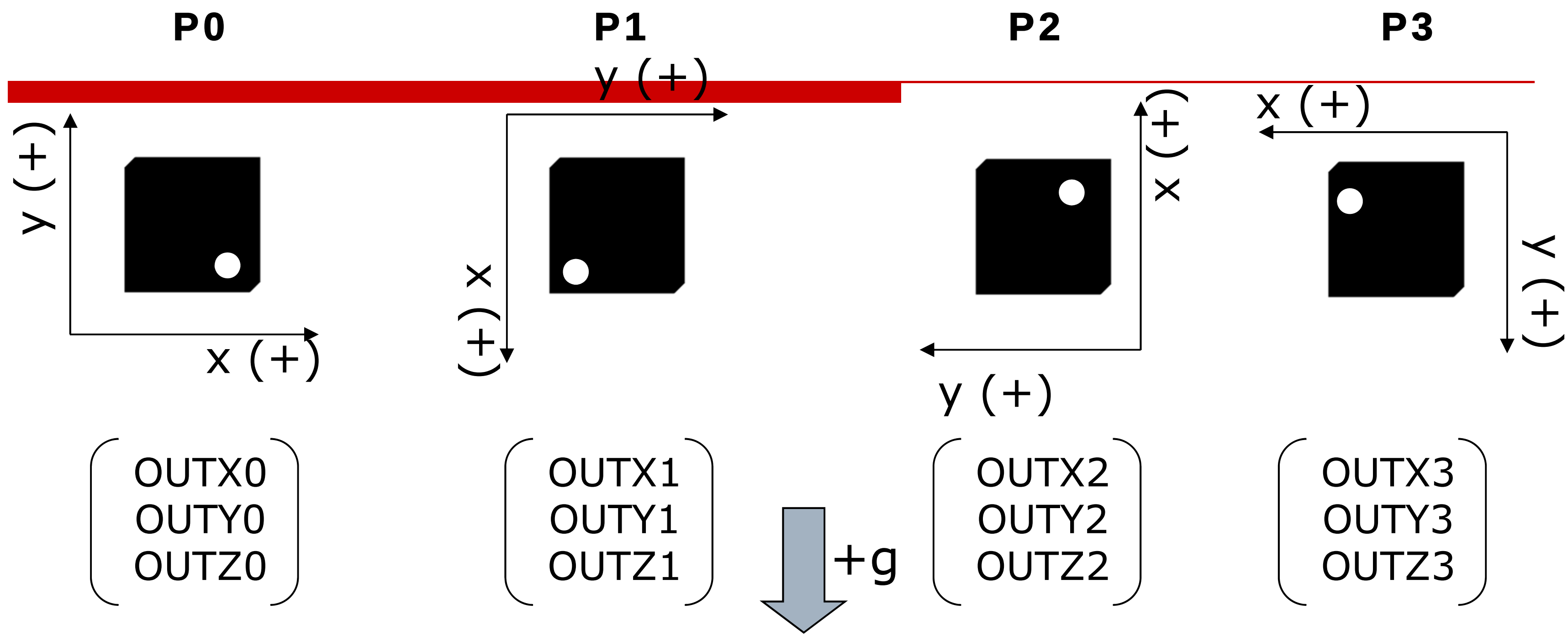


McGill

Highly Accurate Tilt Calculation

- In order to get accurate tilt calculation (±2°) using MEMS accelerometers, few non idealities have to be considered and in case compensated
  - Zero-g level offset and Sensitivity accuracy
  - Non linearity
  - Cross axis
  - Offset and Sensitivity drift vs temperature
- Method to calibrate accelerometers to get accurate 2-axis tilt calculation

Position Required For Full Calibration



- Consider the 4 positions of above and the sensor outputs recorded on each position

Calibration Formulas (1/2)

- Each sensor axis can be calibrated using the following parameters:
  - Offset (OFFX, OFFY) [LSB]
  - Sensitivity (SENSX, SENSY) [LSB/mg]
  - CrossAxis (CXY, CYX) []
- Such parameters can be estimated using the 4 positions in the previous slide using the following approach:

OUTX = OFFX + SENSX * (ACCX + ACCY * CXY)/1000 [LSB]

OUTY = OFFY + SENSY * (ACCX * CYX + ACCY)/1000 [LSB]

where ACCX and ACCY are the known accelerations along X and Y (expressed in ‘g’).

McGill

Calibration Formulas (2/2)

- The parameters will be estimated as follow:

  OFFX = (OUTX0 + OUTX1 + OUTX2 + OUTX3)/4
  SENSX = (OUTX2 – OUTX1)/2
  CXY = (OUTX0 – OUTX3)/(2 * SENSX)
  OFFY = (OUTY0 + OUTY1 + OUTY2 + OUTY3)/4
  SENSY = (OUTY0 – OUTY3)/2
  CYX = (OUTY2 – OUTY1)/(2 * SENSY)
  Where OUTX<u>K</u> is the Output in the <u>K</u> position (See slide 1)

- Then solve iteratively the following equations to obtain estimated acceleration value (ACCX’, ACCY’):

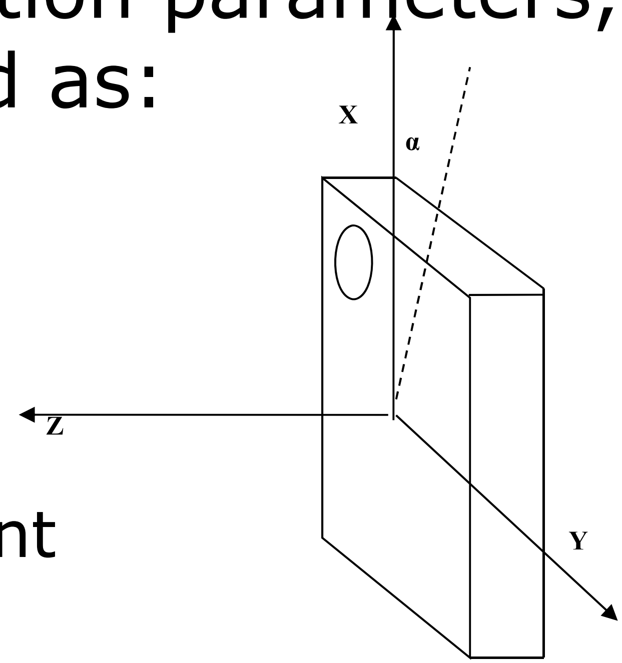
  ACCX’ = (OUTX – OFFX) / SENSX – ACCY’ * CXY
  ACCY’ = (OUTY – OFFY) / SENSY – ACCX’ * CYX

McGill

From calibrated data to tilt value

- Once accelerometers data have been processed through calibration parameters, tilt angle can be estimated as:

$$\alpha = \arctan\left(\frac{\mathrm{ACCX'}}{\sqrt{(\mathrm{ACCY'})^2}}\right)$$

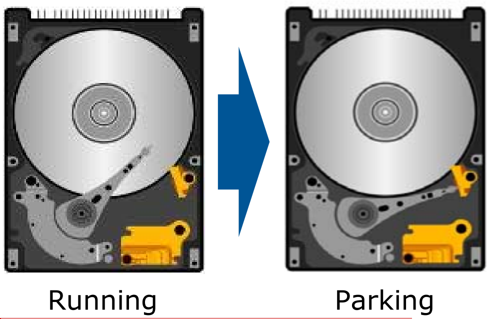


Where ACCX’ and ACCY’ represent acceleration data after applying calibration formulas

Free-fall Detection

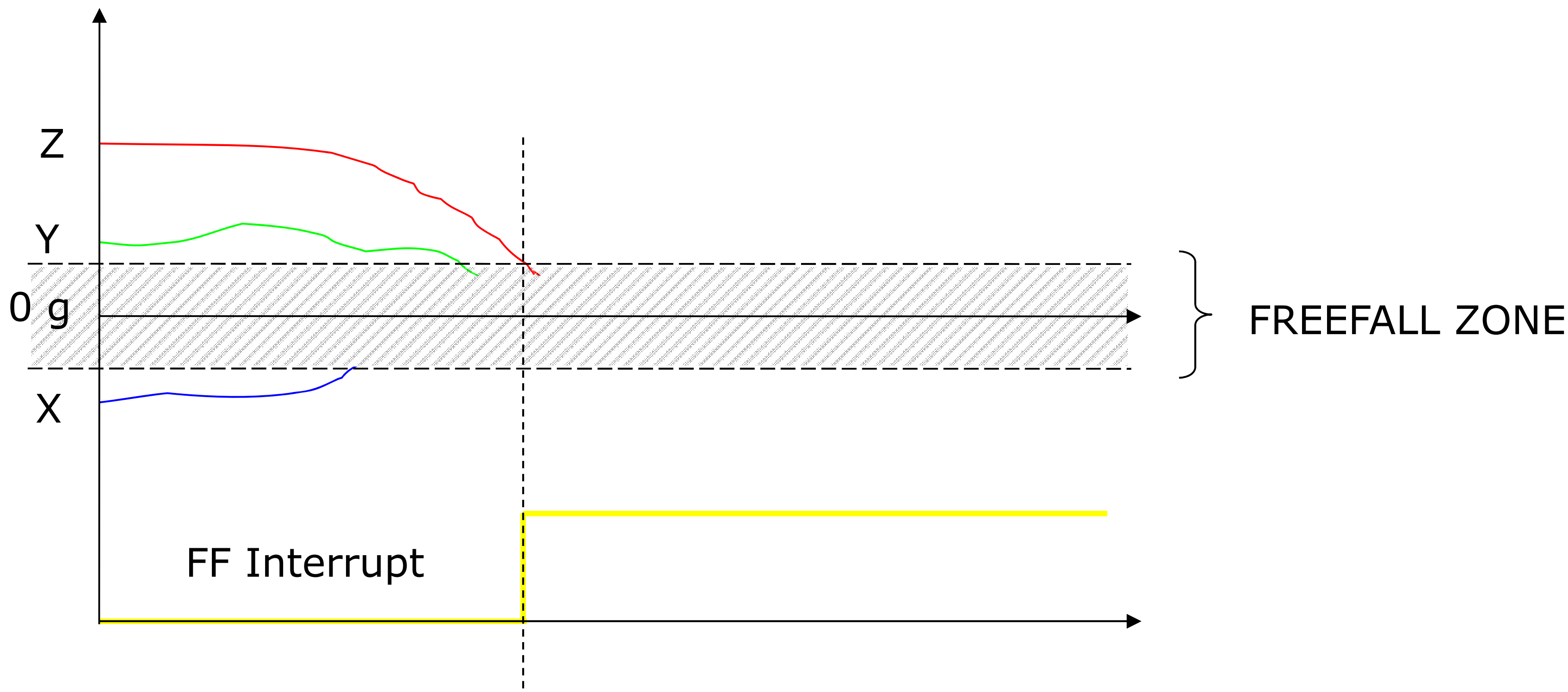
- **F-F: all three axis accelerations equal to zero**
  - While falling, the displacement between the accelerometer moving mass and the substrate (and board) is zero
  - In reality, acceleration values will never be exactly zero even in free-fall conditions because of inherent environment and device non-ideal conditions
  - A certain threshold has to be defined in order to detect the free-fall event

- Free-fall protection implemented on hard-disk drives for portable devices and laptops moves the disk head into a safe position as soon as the free-fall event is detected



Freefall Detection

- A "free-fall zone" defined around the zero-*g* level ->small accelerations generate the interrupt



Freefall Detection Algorithm

- To implement an accurate free-fall detection algorithm, three main points have to be taken into account:
  - Zero-g offset accuracy and Zero-g offset change over temperature
  - During the freefall the X,Y,Z accelerations may vary around the threshold
  - Device rotating while falling

Sensor Offset and Temperature Variation

- The main accelerometer parameters that have to be considered when implementing free fall algorithm are:
  - Zero-g offset accuracy
  - Zero-g offset temperature dependency
- Actually, when the device is falling, the non-zero acceleration is due to the presence of a certain offset and its variation with temperature
- Knowing the values of these parameters help set the suitable threshold to detect the free fall event

McGill

Setting the Suitable Threshold

$$a_f = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

- The formula above represents how “free fall acceleration condition” is calculated
- In ideal case, during free-fall event $a_f = 0$
- In reality, non-ideal conditions to be considered:
  - Some values as a reference:
    - X axis offset: 30mg
    - Y axis offset: 15mg
    - Z axis offset: 20mg
    - No temperature impact is considered

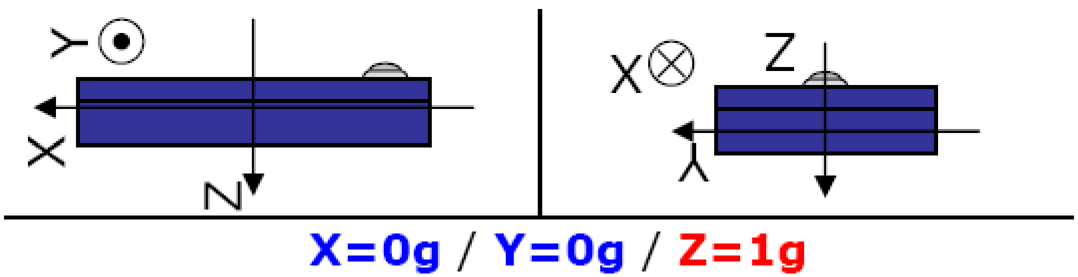
$$40mg = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Setting the Suitable Threshold

- Based on the previous example, even during freefall event, acceleration module will not go below 40mg
- As a consequence, a safe threshold has to be selected in order to recognize the free-fall condition every time the event occurs
- To do so in the right way, offset accuracy and temperature impact have to be considered based on a statistical distribution

Calibrating Sensor to Improve Accuracy

- An easy calibration procedure can be applied to tighten the distribution of parameters that affect accuracy in detecting free fall event
  - Device is placed on a stable horizontal position
  - In this position the three axes will respectively sense:
    - Xacc = 0g + Xoff + Xoff_temp
    - Yacc = 0g + Yoff + Yoff_temp
    - Zacc = 1g + Zoff + Zoff_temp + Zso_accur + Zso_temp + NL



Calibrating sensor to improve accuracy

- Assuming room temperature while calibrating:
  - Xacc = 0g + Xoff
  - Yacc = 0g + Yoff
  - Zacc = 1g + Zoff + Zso_accur + NL
- Reading of Xacc and Yacc provide directly Zero-g offset of X and Y axes
- On Z-axis we can not distinguish between error contribution of Offset and Sensitivity
- Assuming total error on Z-axis is due to Offset, using typical value of sensitivity allows to retrieve Zoff
  - Observed that this assumption leads to reasonable error in the final a_f parameter calculation

Calibration Outcome

- With one single position calibration and the assumptions made, it has been possible to retrieve the following parameters:
  - Xoff, Yoff, Zoff
- From now on, the acceleration data will be compensated in the following way:
  - $Xacc_c = Xacc - Xoff$
  - $Yacc_c = Yacc - Yoff$
  - $Zacc_c = Zacc - Zoff$

Error Estimation

- After applying calibration (at room temperature) the remaining error for each axis is:
  - $Xacc_c$ = Xoff_temp
  - $Yacc_c$ = Yoff_temp
  - $Zacc_c$ = Zoff_temp + (Zso_accur + NL)
- The *"Zso_accur + NL" error arises from the fact that typical sensitivity has been used to estimate offset on Z axis: (8 bit device, ±2g FS)*
  - *Considering a 5% error on Sensitivity and 0.5% NL, the error due to sensitivity is 55 mg*

Setting the Final Threshold

- Total error (@ 1g position) combining the free axis contribution after calibration is (@ 60° C, assuming a typical value of 0.5mg/° C on all the three axis):
  - $Xacc_c$ = Xoff_temp = 30mg
  - $Yacc_c$ = Yoff_temp = 30mg
  - $Zacc_c$ = Zoff_temp + Zso_accur + NL
    = 30mg + 55mg = 85mg

$$a_f = 100mg = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

- A safe value for free-fall TSH can be 300mg

X,Y,Z Accelerations Moving Around the Threshold

- Example: while using notebooks during travels or generally not on a stable desk, it could happen that combination of the three axis accelerations value vary around the free-fall threshold set
- Nowadays algorithms have been improved to consider these borderline situations
- The threshold is continuously adjusted depending on the environmental conditions, avoiding false free-fall detections
- A certain time-duration limit is configured to verify that a_f parameter stays below the TSH at least for a certain period of time

McGill

Device Rotating While Falling

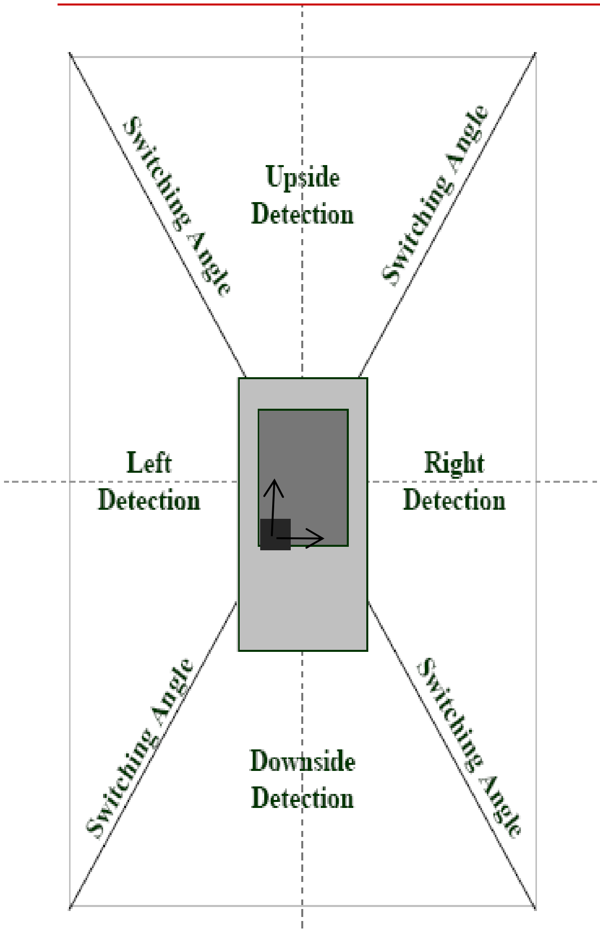
- Consider a common notebook place on a desk
  - When the notebook is pulled down by the cable, a certain rotation is induced while the notebook is falling
- This rotation creates a centripetal force that affect the resulting acceleration sensed by the sensor:
  - It could happen that a_f does not go below the defined threshold
  - A common practice suggestion is to place the sensor on the board as close as possible to the device "static balance point" in order to reduce 'r' to zero

$$CF = r \cdot (2 \cdot \pi \cdot (rotation : rot / \sec))^2$$

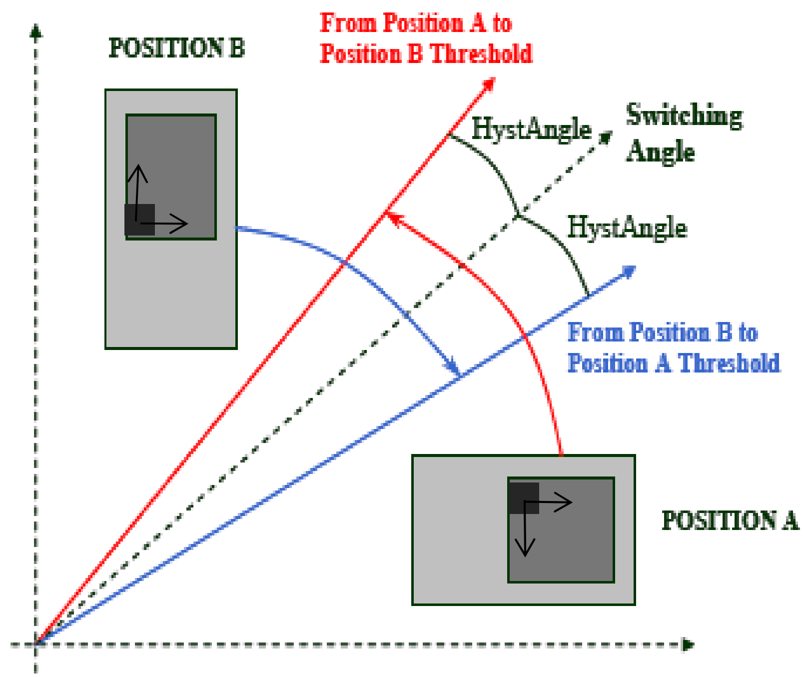
- For aircraft: use magnetometer to maintain yaw/heading

Portrait & Landscape Detection

Four Detectable positions



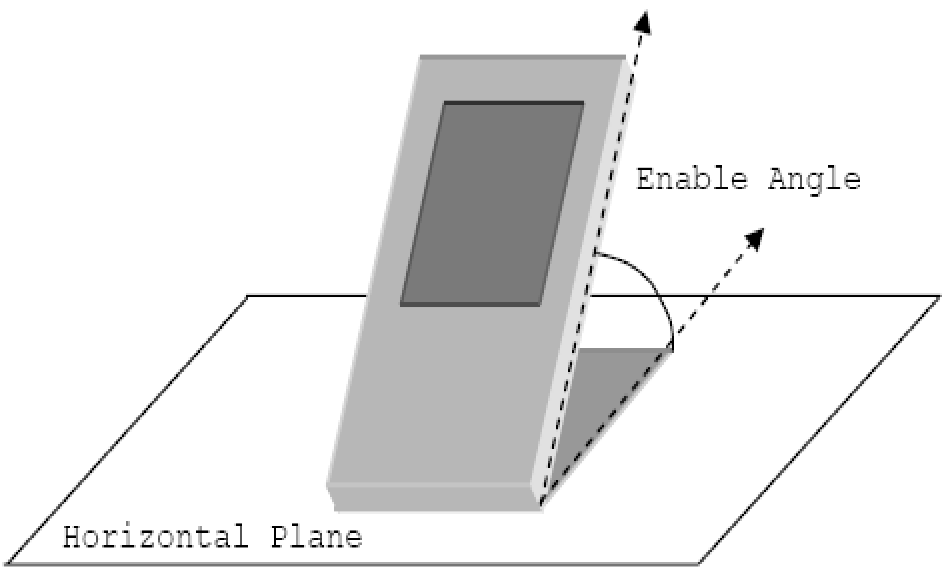
Hysteresis angles



Hysteresis angles help avoiding spurious switching from one position to another
Hysteresis angle sets the real switch threshold.
If Hysteresis angle is set to zero, switch THS is 45° on both directions.

Portrait Landscape Application

Enable angle



Enable angle is the maximum tilt
angle from vertical position that allows P/L algorithm to work

Tilting smaller than enable angle could lead to wrong position detection

30° is the typical value suggested

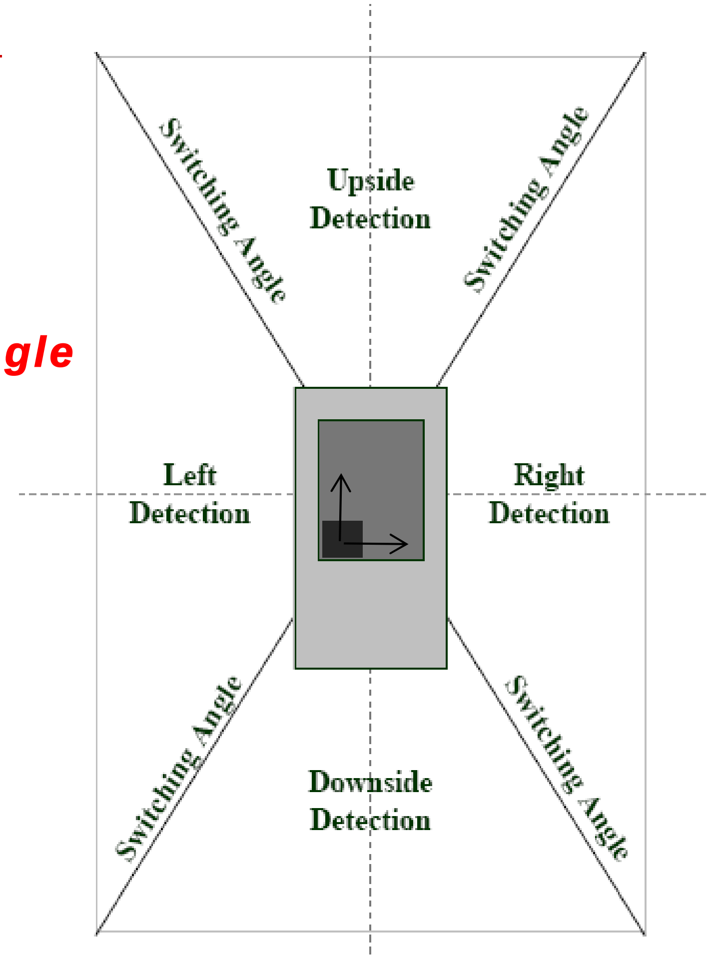
P/L Algorithm inputs/outputs variable

Input variables

TERM_POSIT_THS: hysteresis angle

TERM_POSIT_ENABLE_THS: enable angle

Final switching angle = 45° ± hysteresis angle

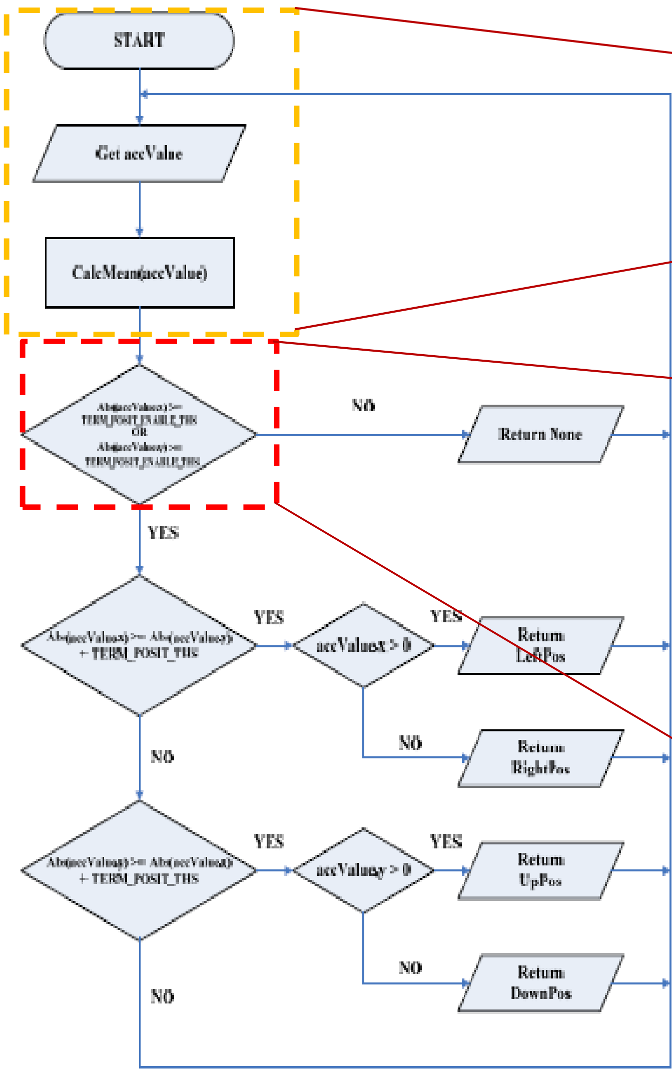


Outputs variables

None: No position recognized

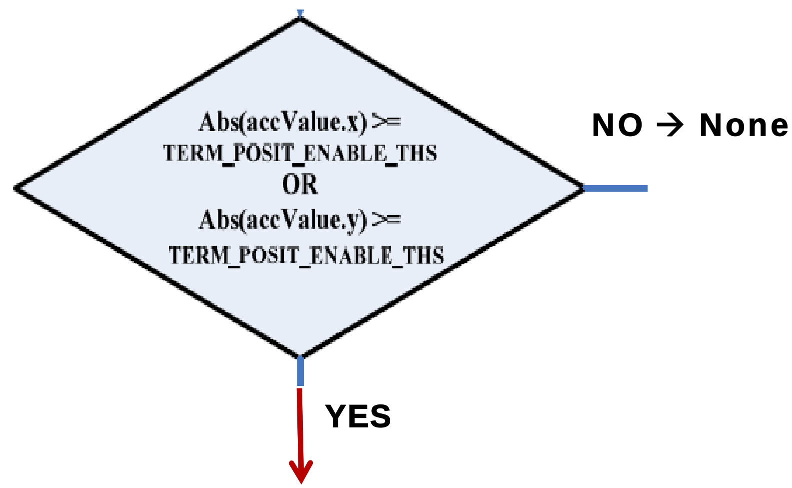
RightPos: Right Position detected
UpPos: Up Position detected
LeftPos: Left Position detected
DownPos: Down Position detected

Portrait landscape application: flow diagram

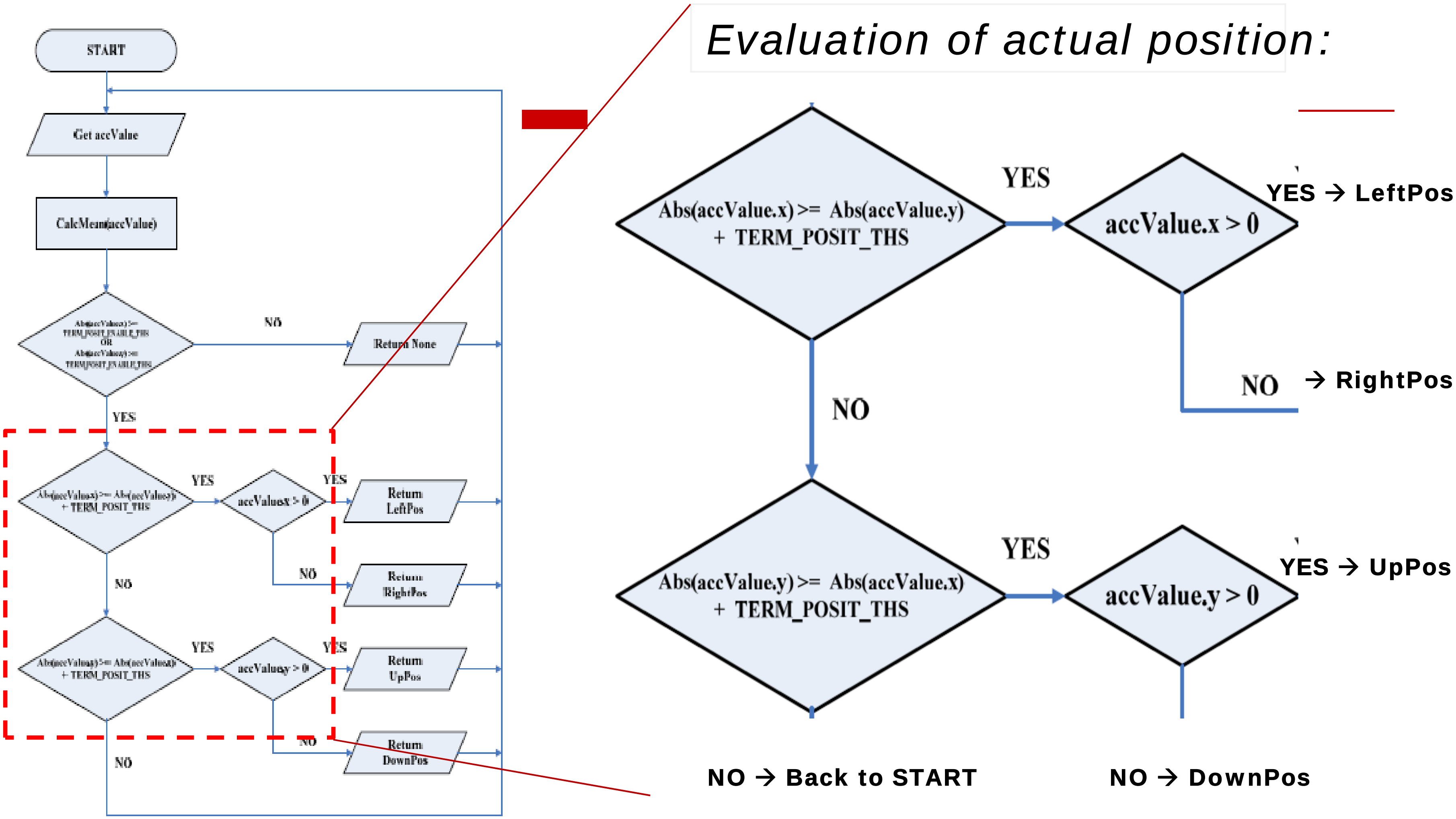


- *Reading acceleration data from the accelerometer*
- *Executing average on few samples acquired*

Control of "Enable angle" condition:



Portrait landscape application: flow diagram



McGill

Main program example code

```
void main()
{
  char data;
  static char timer = 0;
  Application_Init();
  while(1)
  {
    if(DATA_READY)
    {
      acquire(&accel);

      old_status = new_status;
      new_status =
            ST_GetTerminalPosition(accel);
      if(new_status!=old_status)
      {
        Rotate (new_status);
      }
    }
  }
}
```

```
void acquire(ST_Accel_Data_t *acc)
{
  acc->x=Read_Data(OUTX_H) & OUTX_L;
  acc->y=Read_Data(OUTY_H) & OUTY_L;
  acc->z=Read_Data(OUTZ_H) & OUTZ_L;
}
```

Return the actual position starting from the acceleration data read from the sensor

Based on the new_status value a dedicated action is taken by the application

Portrait-landscape Application: Examples

Portrait & Landscape application enables correct pictures orientation inside DSC photo gallery

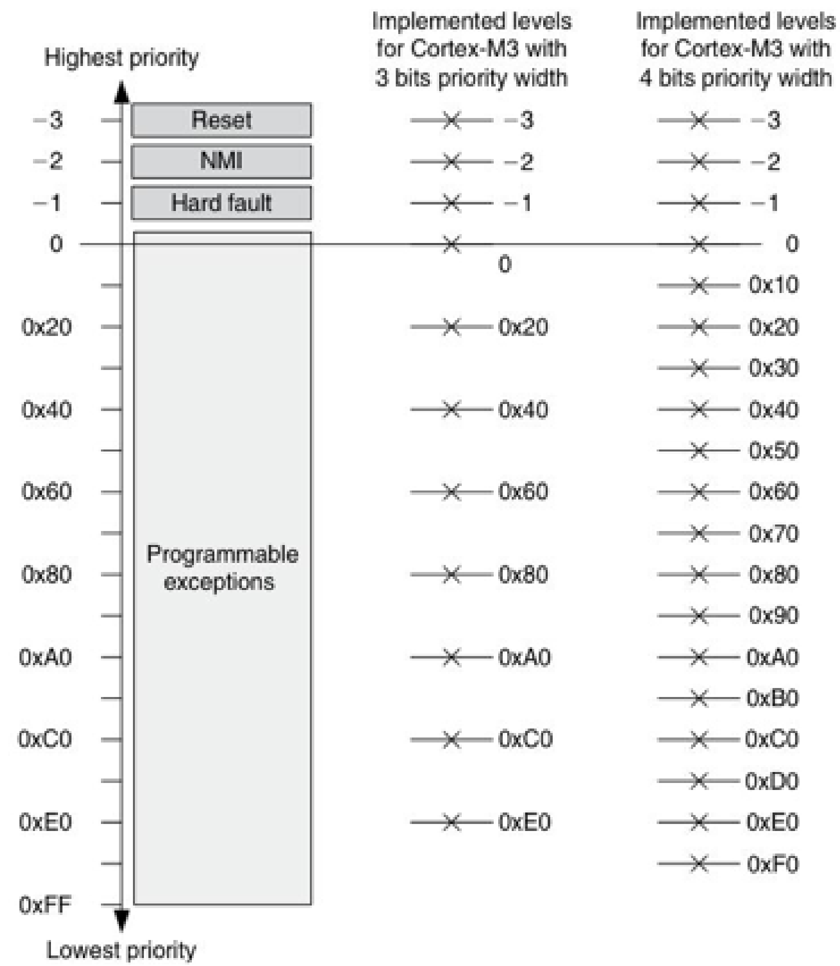
Portrait & Landscape function adjust automatically the picture orientation following the photo-frame position

Portrait & Landscape function rotates the screen on tablet pc

McGill

Exceptions

- Asynchronous processing of various types:
  - Reset
  - Non-maskable interrupt
  - Various faults (mem, bus, usage)
  - Supervisor call
  - Debug and monitoring
  - External interrupts
- Each type can have multiple subtypes/priorities



Cortex-M Exception Types

No.	Exception Type	Priority	Type of Priority	Descriptions
1	Reset	-3 (Highest)	fixed	Reset
2	NMI	-2	fixed	Non-Maskable Interrupt
3	Hard Fault	-1	fixed	Default fault if other hander not implemented
4	MemManage Fault	0	settable	MPU violation or access to illegal locations
5	Bus Fault	1	settable	Fault if AHB interface receives error
6	Usage Fault	2	settable	Exceptions due to program errors
7-10	Reserved	N.A.	N.A.	
11	SVCall	3	settable	System Service call
12	Debug Monitor	4	settable	Break points, watch points, external debug
13	Reserved	N.A.	N.A.	
14	PendSV	5	settable	Pendable request for System Device
15	SYSTICK	6	settable	System Tick Timer
16	Interrupt #0	7	settable	External Interrupt #0
……	…………………..	…………………..	settable	…………………..
256	Interrupt#240	247	settable	External Interrupt #240

McGill

Vector Table

Vector Table starts at address 0

- In the code section of the memory map

Contains addresses of exception handlers

- Not Branch instructions like older ARM processors

Main stack pointer initial value in location 0

- Set up by hardware during Reset

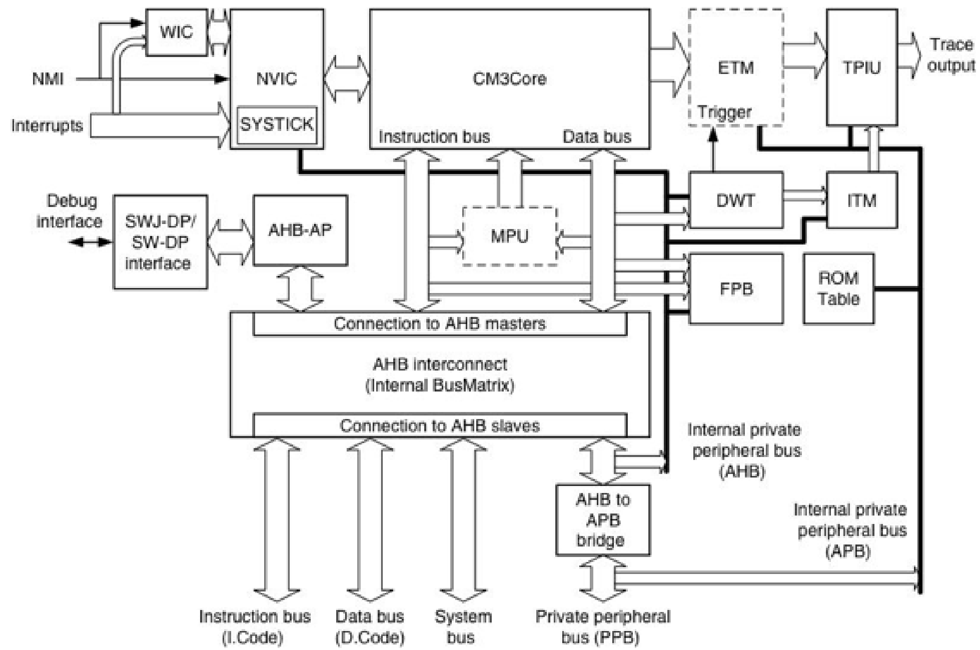
Vector Table can be relocated

Address	Vector
0x00	Initial Main SP
0x04	Reset
0x08	NMI
0x0C	Hard Fault
0x10	Memory Manage
0x14	Bus Fault
0x18	Usage Fault
0x1C-0x28	Reserved
0x2C	SVCall
0x30	Debug Monitor
0x34	Reserved
0x38	PendSV
0x3C	Systick
40	IRQ0
...	More IRQs

Nested Vectored Interrupt Control

NVIC - Interrupts handled in hardware

- Allows handlers written purely in C, no overhead
- Load/Store Multiple instruction (LDM/STM) is interruptible



- STM32 implements 84 exception / interrupt sources
  - 68 maskable peripheral interrupt sources
  - 16 interrupt sources from the Cortex-M3 core
    - 16 programmable priority levels in STM32

Cortex Interrupts

Entry

Processor state automatically saved to stack over data bus (SYSTEM)

{R0-R3, R12, LR, PC, xPSR}

In parallel, ISR prefetched on the instruction bus(ICODE)

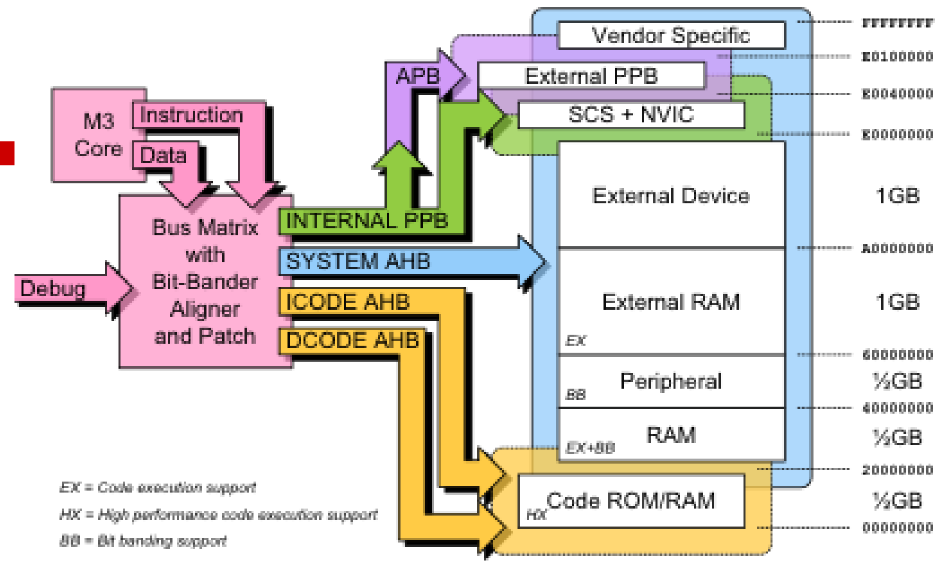
-ISR ready to start executing as soon as stack PUSH complete

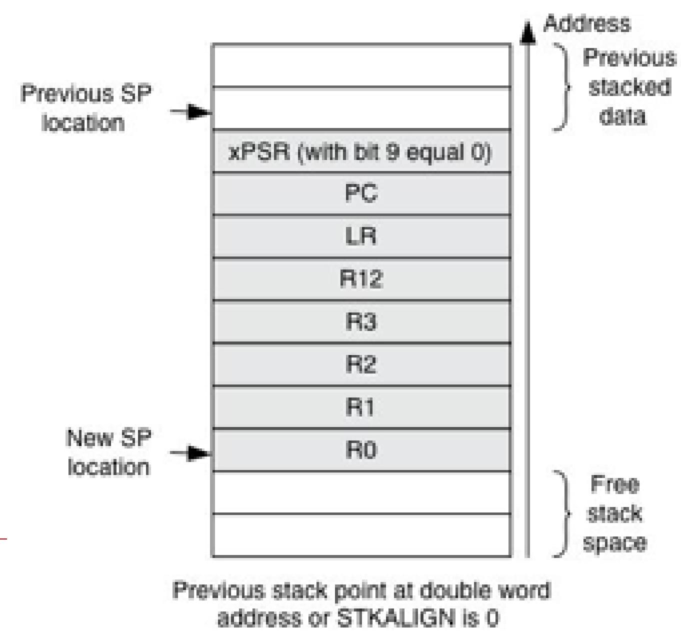
Exit

Processor state is automatically restored from the stack

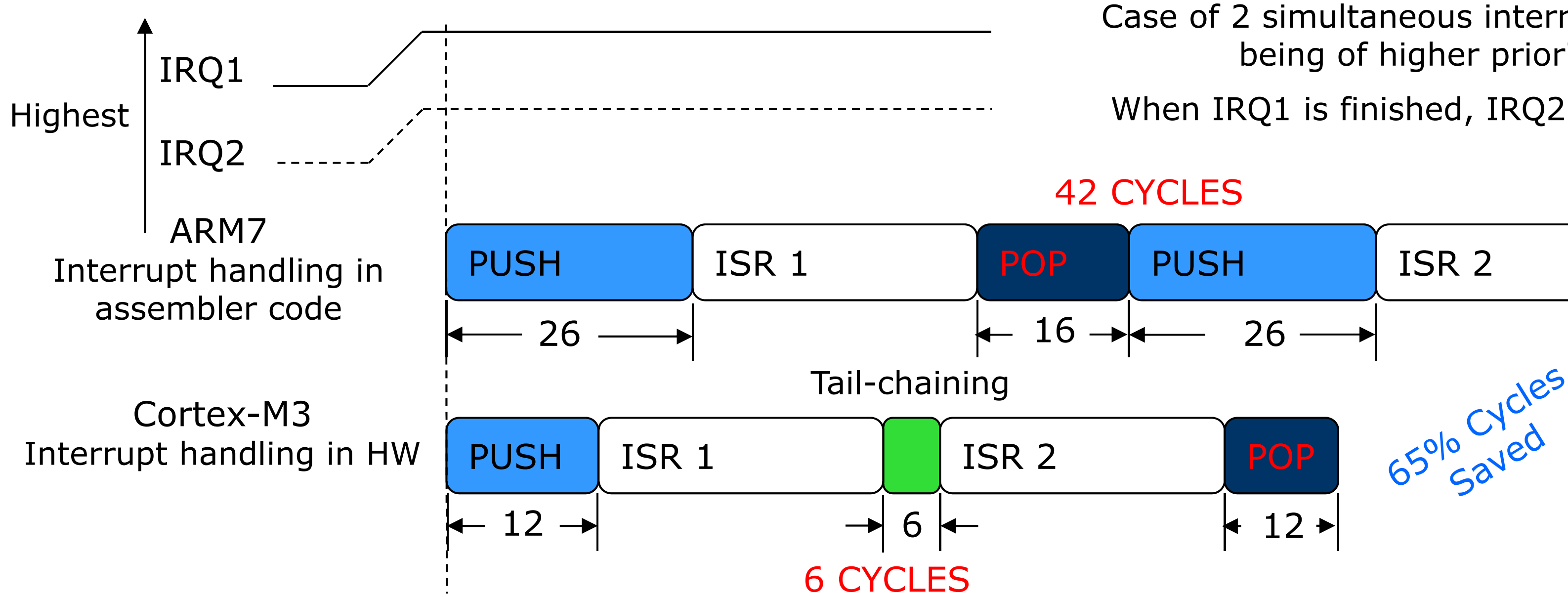
In parallel, interrupted instruction is prefetched for execution upon POP

- Stack POP can be interrupted -> new ISR immediately executed





Fast Interrupts- Tail Chaining



Case of 2 simultaneous interrupts, IRQ1 being of higher priority

When IRQ1 is finished, IRQ2 is serviced

In Cortex-M, ISR2 has only a 6-cycle delay. ISR2 has been '*tail-chained*'

ICODE (vector) and SYSTEM (stack) used in parallel

Interrupt Prioritization

- Each interrupt source has an 8-bit interrupt priority value
- Bits divided into pre-empting and non-pre-empting "sub-priority" levels
  - Sub-priority levels only have an effect if the pre-empting same priority levels
  - Software programmable PRIGROUP register field of the NVIC chooses how many of the 8-bits are used for "group-priority" and how many are used for "sub-priority"
  - Group priority is the pre-empting priority
- Lower numbers are higher priority
- Hardware interrupt number is lowest level of prioritization
  - IRQ3 is higher priority than IRQ4 if the priority registers are programmed the same
- In STM32F10x 16 levels (4-bit) of priority implemented:

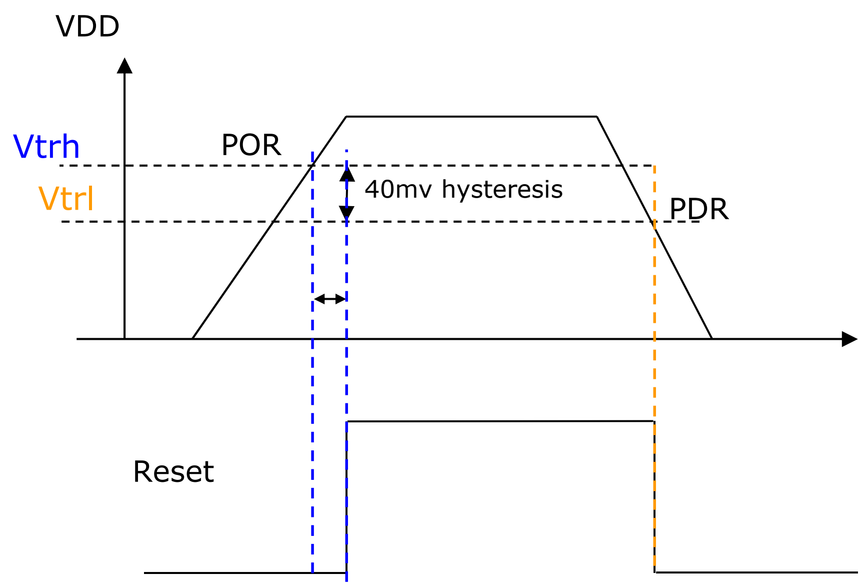
PRIGROUP (3 Bits)	Binary Point (group.sub)		Preempting Priority (Group Priority)		Sub-Priority	
			Bits	Levels	Bits	Levels
011	4.0	gggg	4	16	0	0
100	3.1	gggs	3	8	1	2
101	2.2	ggss	2	4	2	4
110	1.3	gsss	1	2	3	8
111	0.4	ssss	0	0	4	16

Internal Power On Reset

No Need for External Reset Circuit

Integrated POR / PDR guarantees reset when Vdd is out of spec

POR and PDR have a typical hysteresis of 40mV



McGill

Internal Programmable Voltage Detector (PVD)

Provides early warning of power failure

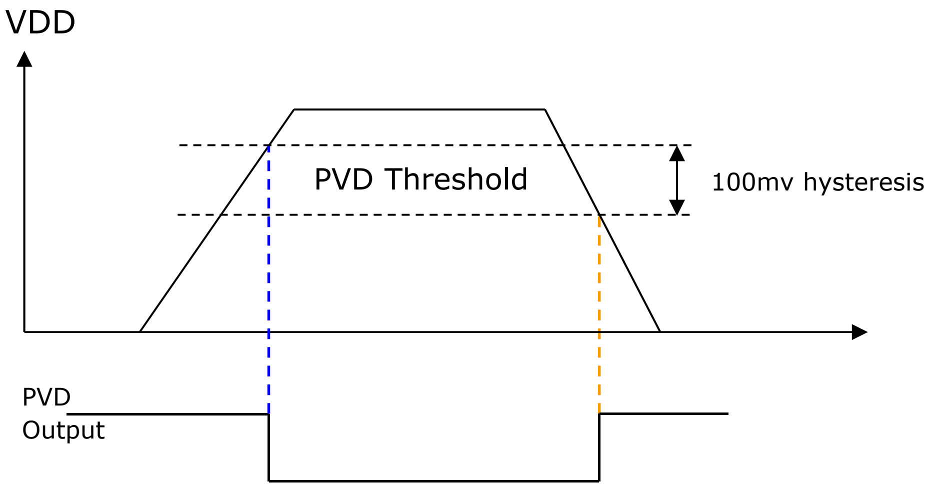
Enabled by software

Compares V_{DD} to a configurable threshold

2.2V to 2.9V, 100mV steps

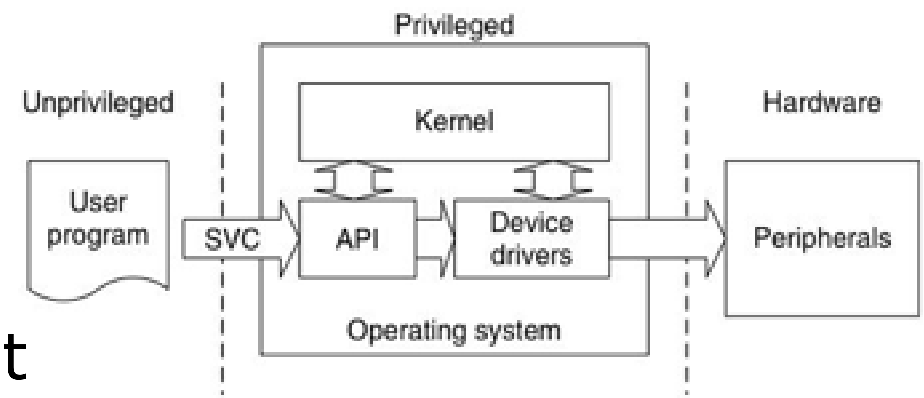
Generates an interrupt

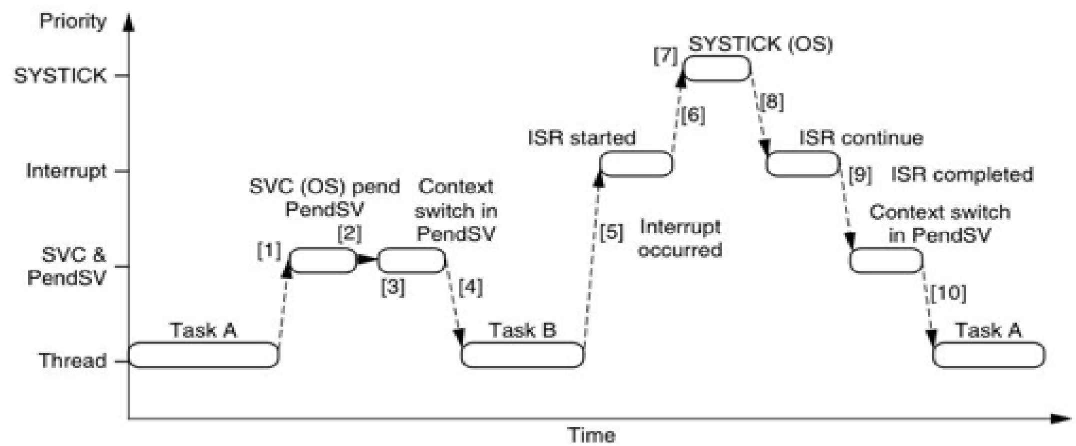
Use: puts the MCU into a safe state



Supervisor Calls: SVC & PendSV

- Supervisor Call (SVC): for system function calls
  - Portable; HW abstraction
  - Can't nest! (no SVC in SVC)
- SVC Instruction
  - Keil/ARM: __svc
- Pendable SV: can wait/nest
- SysTick: OS clock
  - Good for RTOS
  - 24-bit down counter
  - 2 clock sources
  - Only priviledged mode





SVC Handling

- Quickly Identify the User/Supervisor stack based on low-order address bits
  - MRS: Move to register from special register
- Process right type of SVC
- Note assembly inline (__asm)

```
#define SVC_00 0x00
#define SVC_01 0x01
void __svc(SVC_00) svc_zero(const char *string);
void __svc(SVC_01) svc_one(const char *string);
int call_system_func(void)
{
    svc_zero("String to pass to SVC handler zero");
    svc_one("String to pass to a different OS function");
}
```

```
__asm void SVCHandler(void)
{
    IMPORT SVCHandler_main
    TST lr, #4
    MRSEQ r0, MSP
    MRSNE r0, PSP
    B SVCHandler_main
}
void SVCHandler_main(unsigned int * svc_args)
{
    unsigned int svc_number;
    /*
    * Stack contains:
    * r0, r1, r2, r3, r12, r14, the return address and xPSR
    * First argument (r0) is svc_args[0]
    */
    svc_number = ((char *)svc_args[6])[-2];
    switch(svc_number)
    {
        case SVC_00:
            /* Handle SVC 00 */
            break;
        case SVC_01:
            /* Handle SVC 01 */
            break;
        default:
            /* Unknown SVC */
            break;
    }
}
```

Source : Application Note 179 – Cortex™-M3 Embedded Software Development

Reset: Window Watchdog (WWDG)

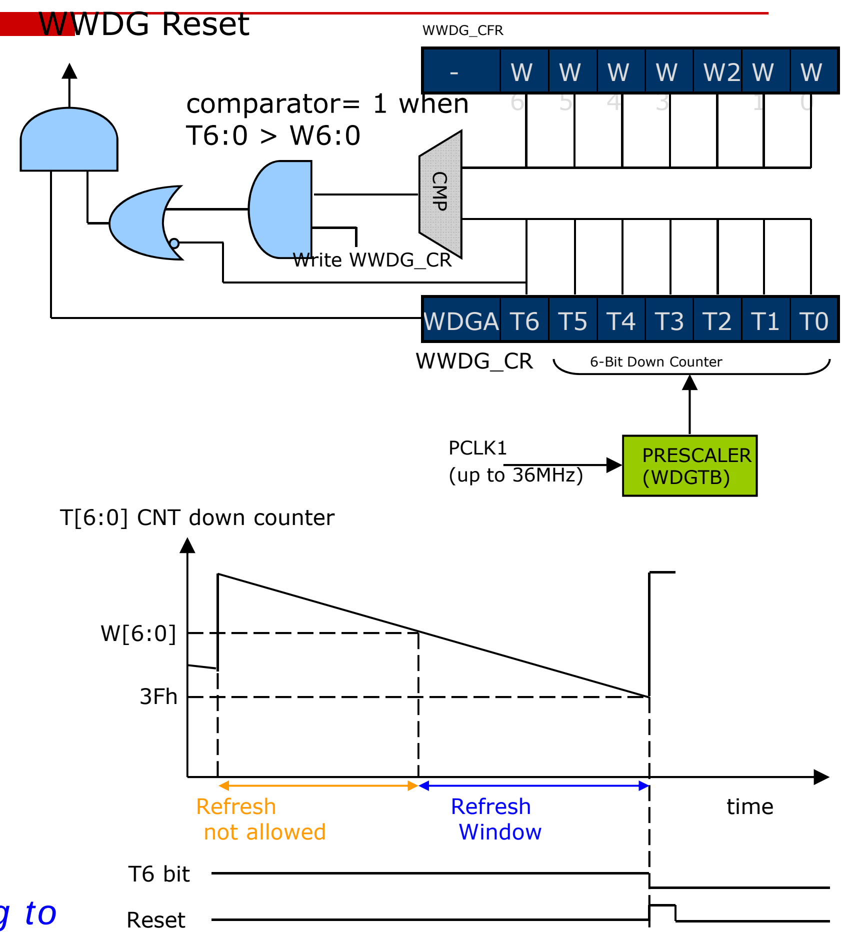
Configurable time-window, can be programmed to detect abnormally late or early application behavior

Conditional reset

- Reset when down-counter value becomes less than 0x40
- Reset if down counter is reloaded outside the time-window

Reset flag to tell when WWDG reset occurs

Min-max timeout value @36MHz: 113µs / 58.25ms



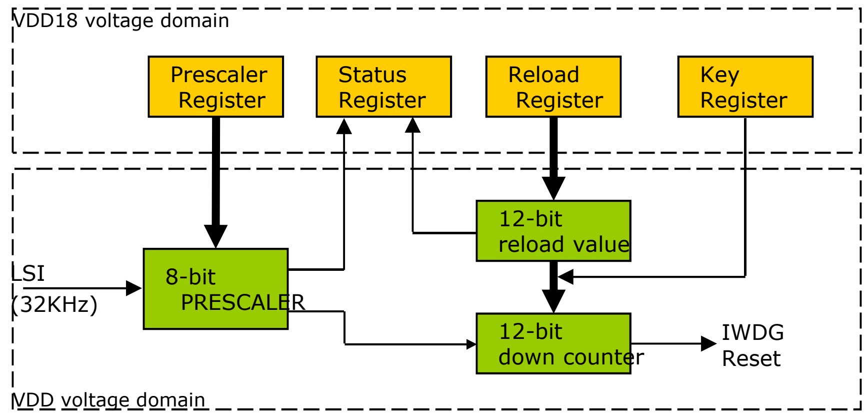
Best suited to applications which require the watchdog to react within an accurate timing window

McGill

Independent Watchdog (IWDG)

Not Clocked by Main Processor Clock!

- Selectable HW/SW start through option byte
- Advanced security features:
  - IWDG clocked by internal low-speed RC clock
  - Stays active even if the main clock fails
  - Once enabled the IWDG can't be disabled
  - Safe Reload Sequence (key)
  - IWDG function remains functional in STOP and STANDBY modes



- To prevent IWDG reset: write 0xAAAA key value at regular intervals before the counter reaches 0

Best suited to applications which require the watchdog to run as a totally independent process outside the main application

- Reset flag to inform when a IWDG reset occurs
- Min-max timeout value @32KHz: 125µs / 32.8s