电子科学与技术专业课

嵌入式系统

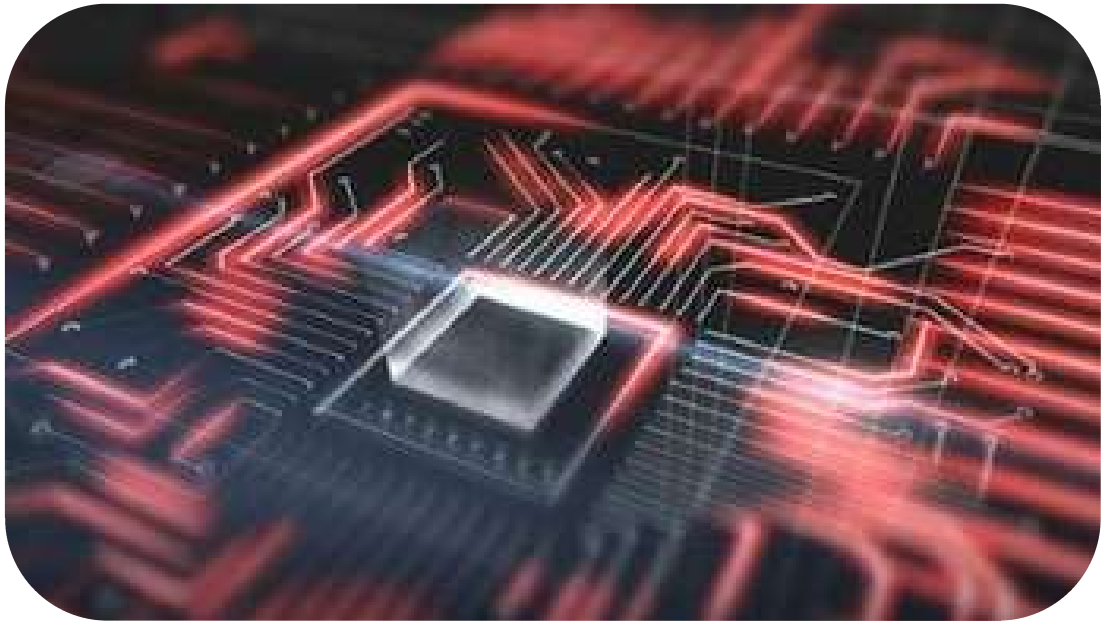
Embedded System

信息学院 光电子系

第5章 通信接口与总线

- 5.1 通信概述
- 5.2 异步串行通信UART
- 5.3 串行外设接口SPI
- **5.4 集成电路总线I2C**

嵌入式系统(EMBEDDED SYSTEM)



第5章 通信接口与总线

❀ **5.4 集成电路总线I2C**

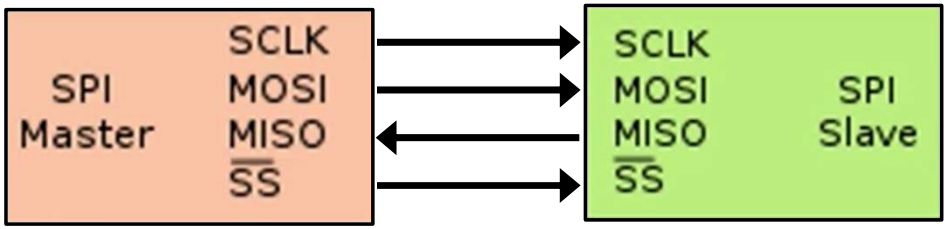
回顾 - SPI通信

- SPI – Serial Peripheral Interface

  -- 同步串行外设接口　　全双工数据通信

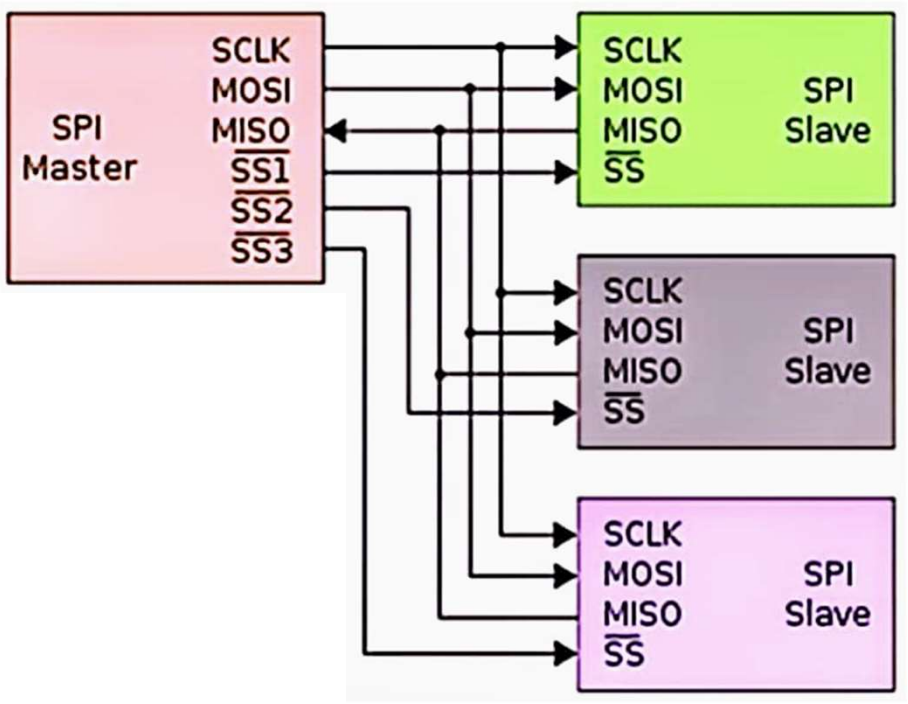
- 方便连接各种外设/芯片

  -- ADC / RTC / LCD / ROM / DAC / Sensor ……



四线制连接

从设备，由SS选择



多机连接

5.4 集成电路总线I2C

- 本节内容安排
  - I2C通信简介
  - I2C通信的协议
  - STM32的I2C接口
  - I2C常用库函数
  - I2C编程实例 – 加速度传感器
  - 编程练习E5 ： I2C接口加速度传感器编程

I2C通信简介

- I2C – Inter-IC Bus （Inter-Integrated Circuit）

  -- I2C/I²C/IIC

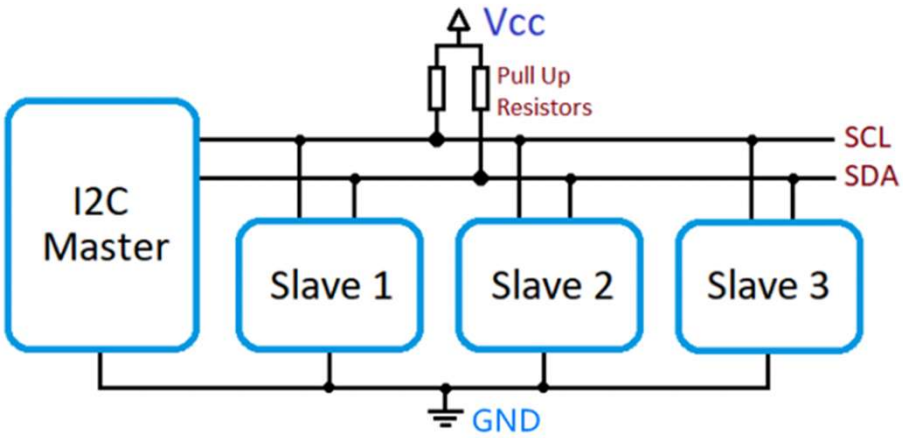
  -- 一种双向2线制同步串行通信接口

- 简单、方便地连接各种外设/芯片

  -- 一种双向2线制串行总线

  -- 支持多从机设备

2线制：时钟线、数据线

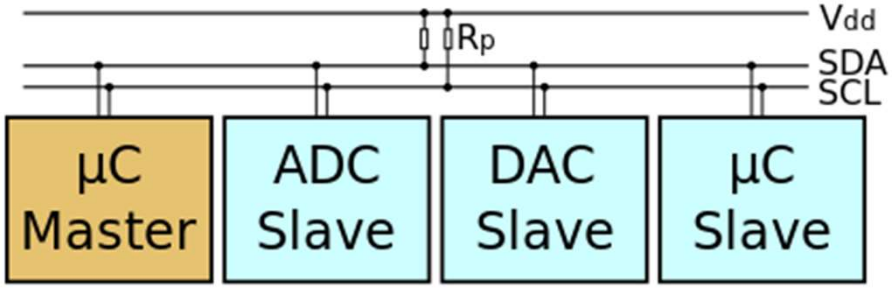


I2C – Inter-IC Bus 的历史

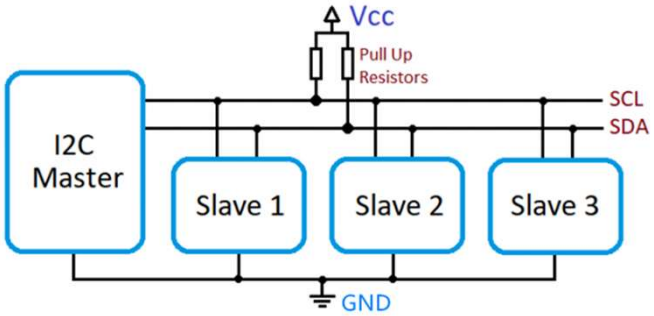
- In 1982, the original 100KHz I2C system was created, by Philips.
- In 1992, Version 1 added 400KHz Fast-mode(Fm) and a 10-bit addressing mode to increase capacity to 1008 nodes.
- In 1998, Version 2 added 3.4MHz High-speed mode(Hs).
- In 2007, Version 3 added 1MHz Fast mode(Fm+), and a device ID mechanism.
- In 2012, Version 4 added 5MHz Ultra Fast-speed mode(UFm) for new USDA and USCL lines using push-pull logic without pull-up resistors, and added assigned manufacturer ID table.
- In 2012, Version 5 corrected mistakes.
- In 2014, Version 6 corrected two graphs. This is the most recent standard.

I2C – Inter-IC Bus 基本特征

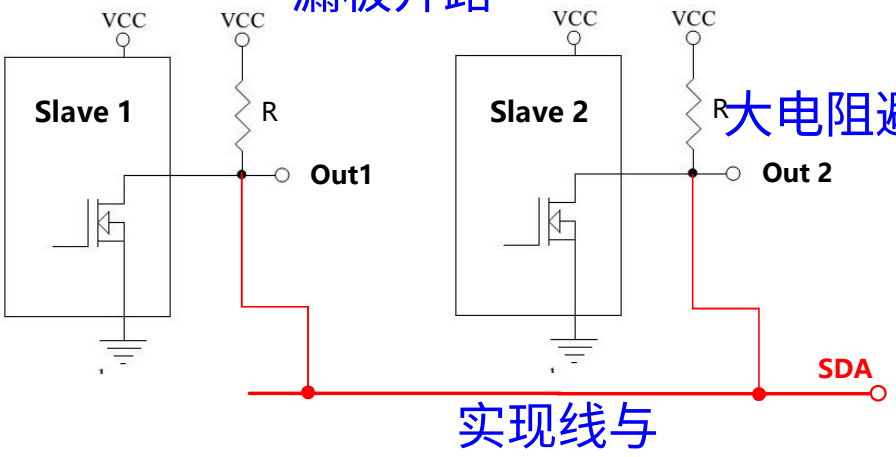
- 基本特征
  - 串行 – 2根线
  - 同步 – 时钟信号
  - 单端 – 数据传输　非双端差分
  - 双向（但非全双工）　一条数据线
  - 主从（Master/Slave）
  - 总线（Bus）为了应对但数据线与多设备之间的通信
  - 协议（Protocol）



I2C – Inter-IC Bus 电气特征



漏极开路



大电阻避免线与时烧毁电路

实现线与

- **两根信号线（以及共地）**

  -- SDA 串行数据线

  -- SCL 串行时钟线

- **SDA和SCL引脚特殊** {与推挽结构区别→ 

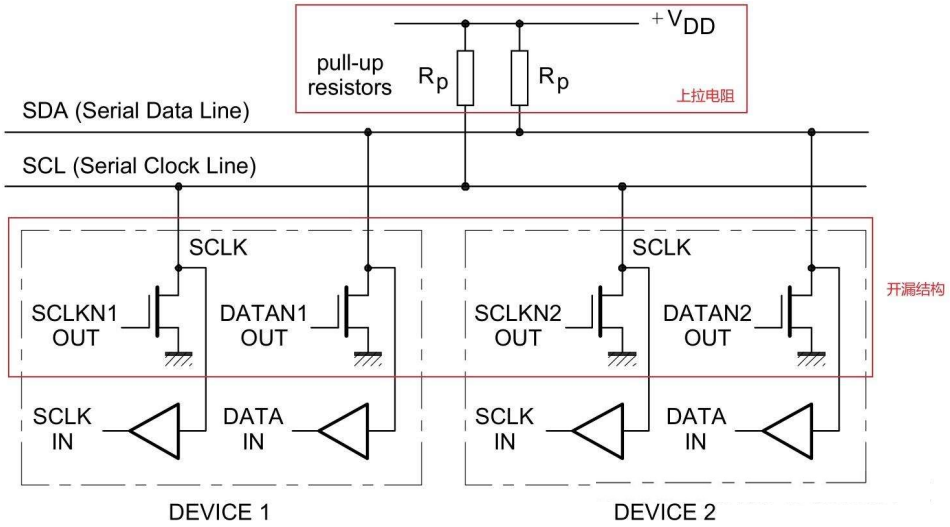
  -- 漏极开路（Open Drain, OD）

  -- 集电极开路（Open Collector, OC）

  -- “线与”并联，有0则0

Out1	Out2	SDA
0	X	0
X	0	0
1	1	1

I2C – Inter-IC Bus 电气特征



- 两根信号线

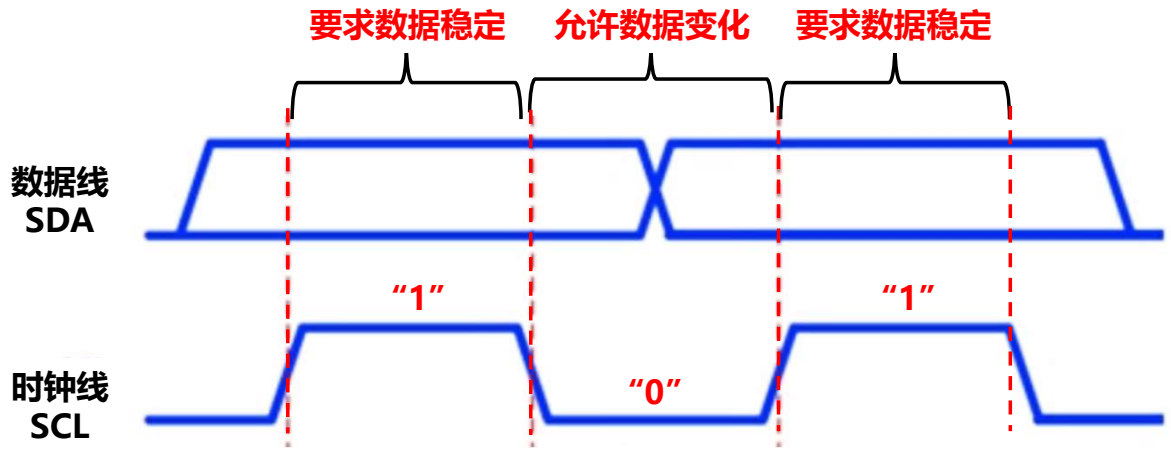
  -- SDA 串行数据线

  -- SCL 串行时钟线
- SDA和SCL都接了上拉电阻（Pull-up）
- 所有的SDA引脚、所有的SCL引脚，是逻辑与的关系（线与）
- 总线空闲时，都是高电平（不耗电流）
- 兼容性好，5V和3.3V都可以

I2C – Inter-IC Bus 优点

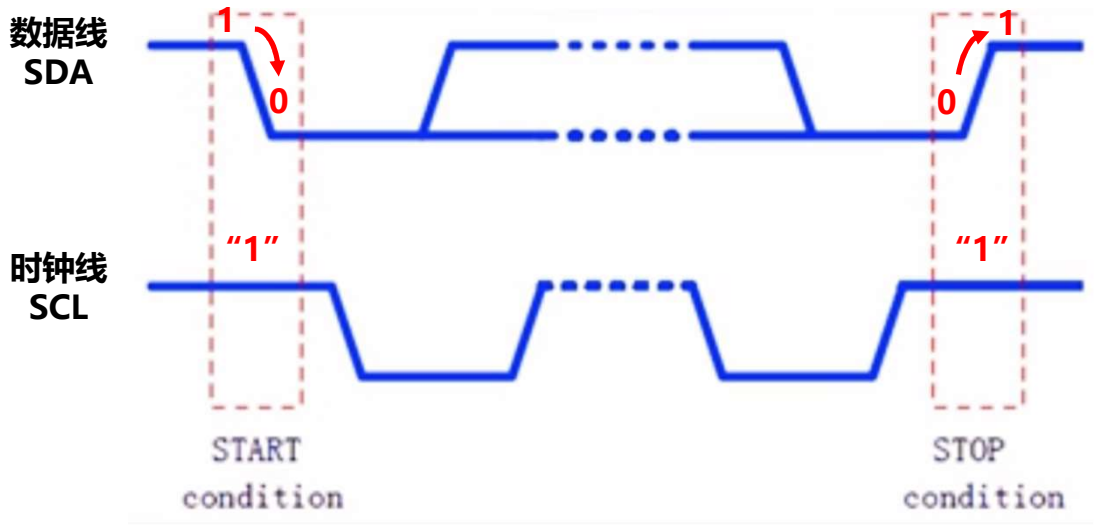
- 只需要2根信号线（节约面积、引脚、成本）
- 协议简单（？）
- 协议容易实现（硬件模块、开漏引脚、GPIO）
- 支持的器件多、功能丰富（NXP、TI、ST、Maxim。。。）
- 总线可以同时挂载多个器件
- 总线电气兼容性好（5V、3.3V）
- 速率较高（100kbps ~ 400kbps ~3.4Mbps）
- 距离较远（几米，降低速率~十几米）

I2C – Inter-IC Bus 的电平逻辑



- I2C是电平有效的，与SPI不同（边沿有效）
- 信号电平规范
  - -- SDA的电平，在SCL为高电平时，保持稳定不变；
  - -- SDA的电平，在SCL为低电平时，允许发生变化；
  - -- 仅在特殊情况下（通信开始或结束时），SDA在SCL为高时变化。

I2C – Inter-IC Bus 的起始和停止



- **信号定义**

-- 起始位：当SCL处于高电平时，SDA从高电平向低电平负跳变，产生“起始”位，简记为S。
总线在起始产生后便处于忙(Busy)状态。

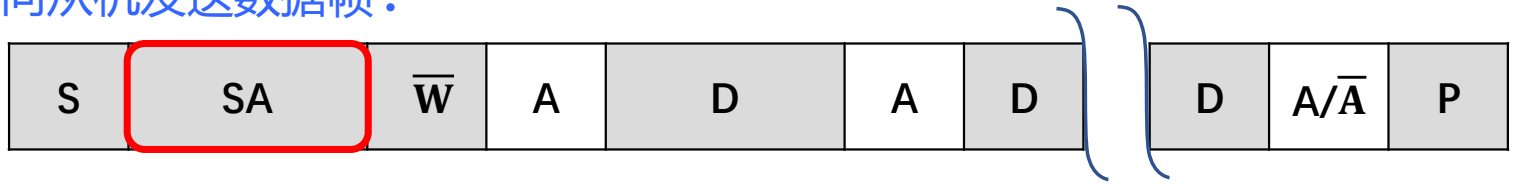
-- 停止位：当SCL处于高电平时，SDA从低电平向高电平正跳变，产生“停止”位，简记为P。
总线在停止产生后便处于空闲(Idle)状态。

5.4 集成电路总线I2C

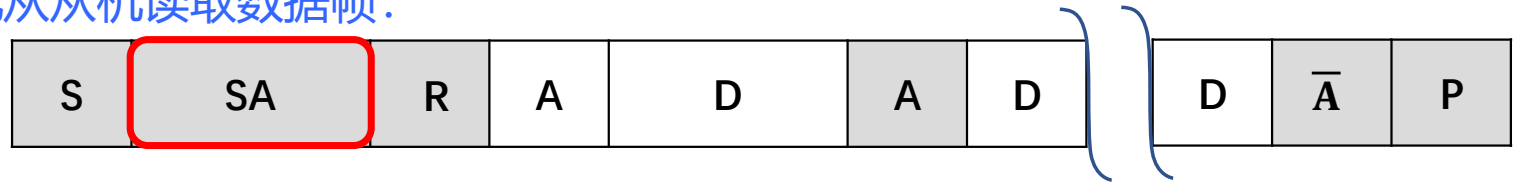
- Next .....
  - I2C通信简介
  - I2C通信的协议　　带有地址信息的通信协议
  - STM32的I2C接口
  - I2C常用库函数
  - I2C编程实例 – 加速度传感器
  - 编程练习E5 ： I2C接口加速度传感器编程

I2C – Inter-IC Bus 的通信数据帧

主机向从机发送数据帧：

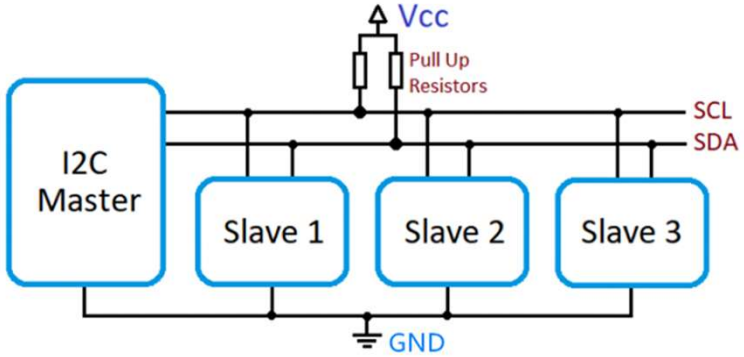


主机从从机读取数据帧：



灰格: 主机行为

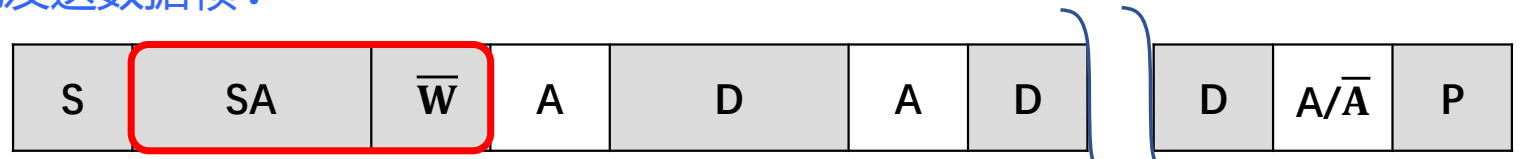
白格: 从机行为



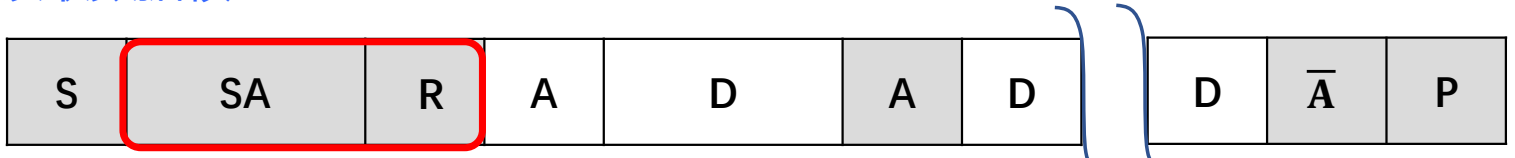
- SA- 从机地址 Slave Address
  - I2C可以连接多个从机设备，而无需片选信号引脚；
  - 多个器件设备，使用地址进行区分；
  - 主机不需要地址，每个从机必须有个地址；
  - 从机地址不能重复，地址为7 bits。

I2C – Inter-IC Bus 的通信数据帧

主机发送数据帧：



主机读取数据帧：



灰格: 主机行为

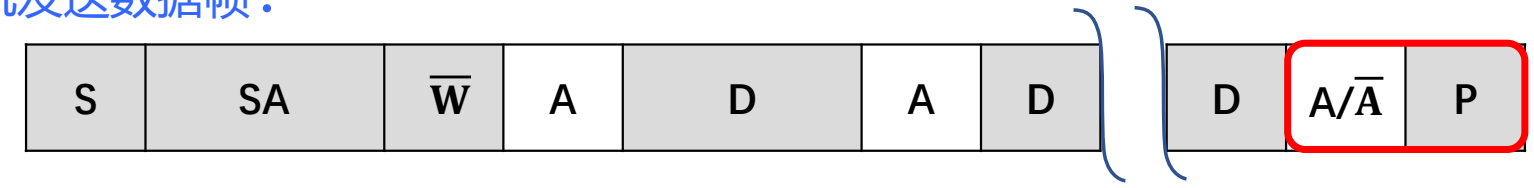
白格: 从机行为

符号	定义	位宽
S	起始位 (Start)	1
SA	从机地址 (Slave Address)	7
$\overline{W}$	写操作控位 (=0)	1
R	读操作控位 (=1)	1
A	应答 (Ack, =0)	1
$\overline{A}$	无应答 (Not Ack, =1)	1
D	数据 (Data)	8
P	停止位 (Stop)	1

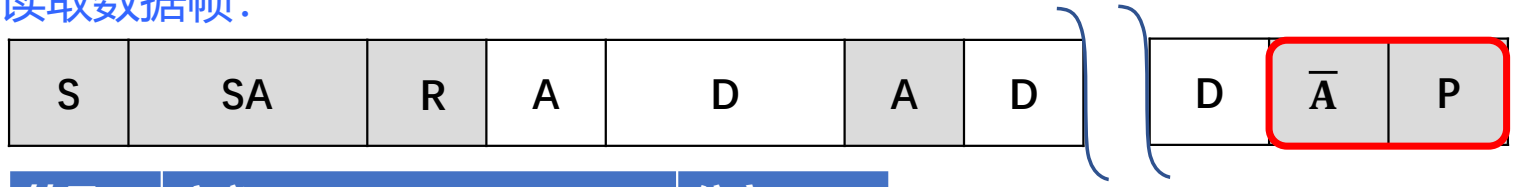
- 通信由主机发起和结束
  -- 从机地址后跟1bit的读/写操作控制位
  -- 主机、从机通过释放/拉动数据线“互动”
- A/$\overline{A}$ - 应答与非应答
  -- 应答表示对收到信息的回应(OK)，Ack = 0 Why?
  -- 无应答表示不再进行后续通信
- 数据D 8bits，MSB在先，LSB在后

I2C – Inter-IC Bus 的通信数据帧

主机发送数据帧：



主机读取数据帧：



灰格: 主机行为

白格: 从机行为

符号	定义	位宽
S	起始位 (Start)	1
SA	从机地址 (Slave Address)	7
$\overline{W}$	写操作控位 (=0)	1
R	读操作控位 (=1)	1
A	应答 (Ack, =0)	1
$\overline{A}$	非应答 (Not Ack, =1)	1
D	数据 (Data)	8
P	停止位 (Stop)	1

- **通信截止的两种情况**

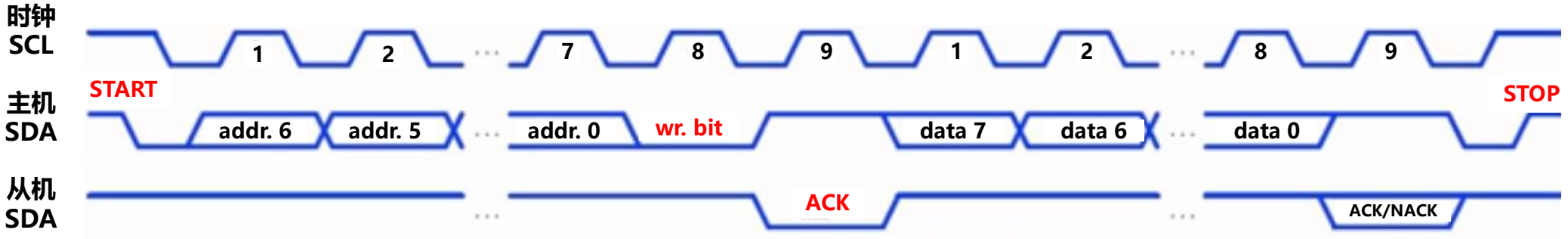
-- 正常应答情况下(A)，主机主动结束(P)；

-- 接收方无应答情况下($\overline{A}$)，主机结束(P)。

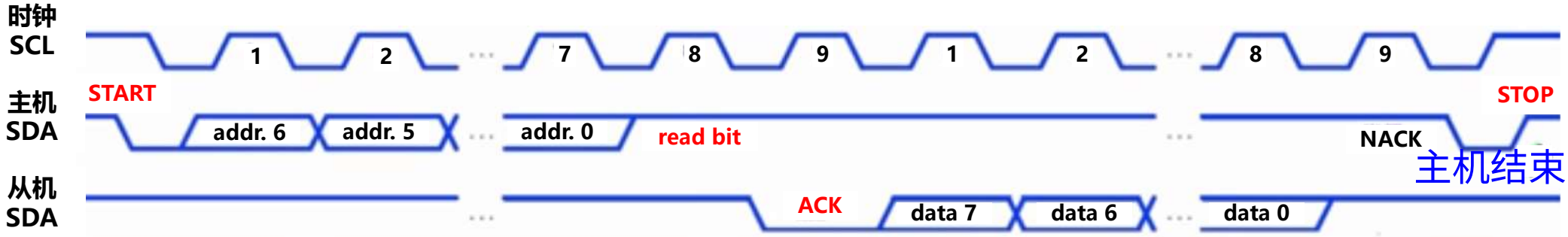
-- 即A-P 和 $\overline{A}$ - P

I2C – Inter-IC Bus 的通信数据帧

主机发送1个字节数据：



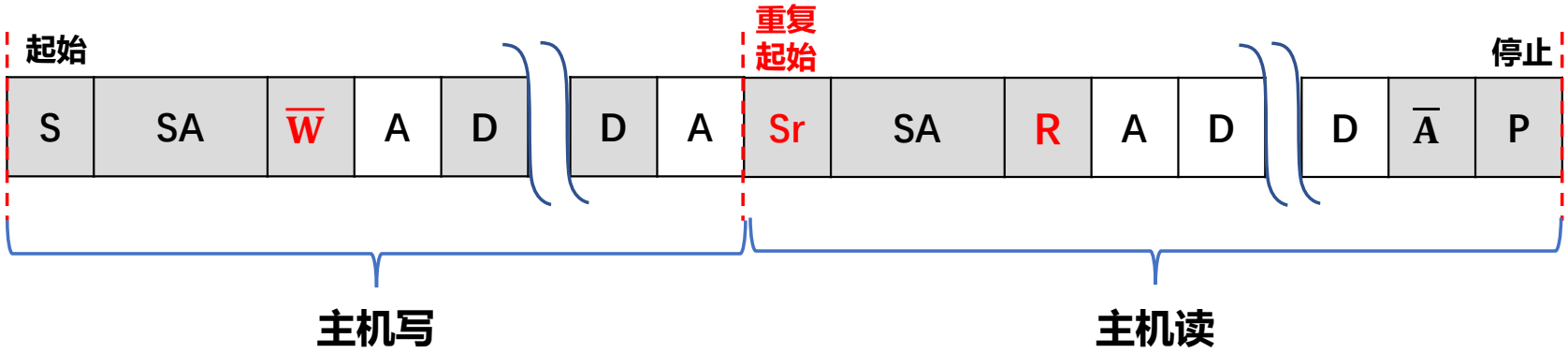
主机读取1个字节数据：



I2C高阶 – 重复起始和子地址

◆ Sr 重复起始（Repeated START）

- I2C通信中，有时需要切换数据收发的方向；
- 例如I2C设备是个EEPROM存储器，先写入若干数据（主->从），再读出若干数据（从->主）；
- 此时无需给出停止位，然后再给起始位；
- 而是直接再产生一次开始位，称为“重复起始”位，记为Sr。

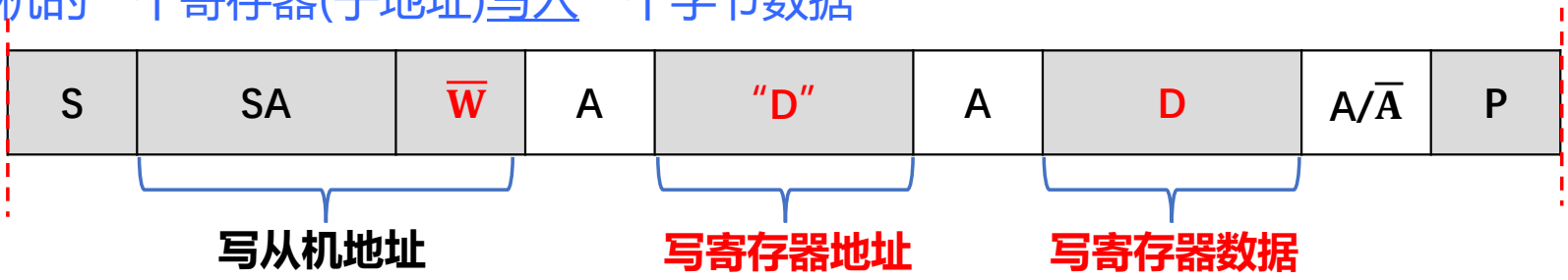


灰：主机
白：从机

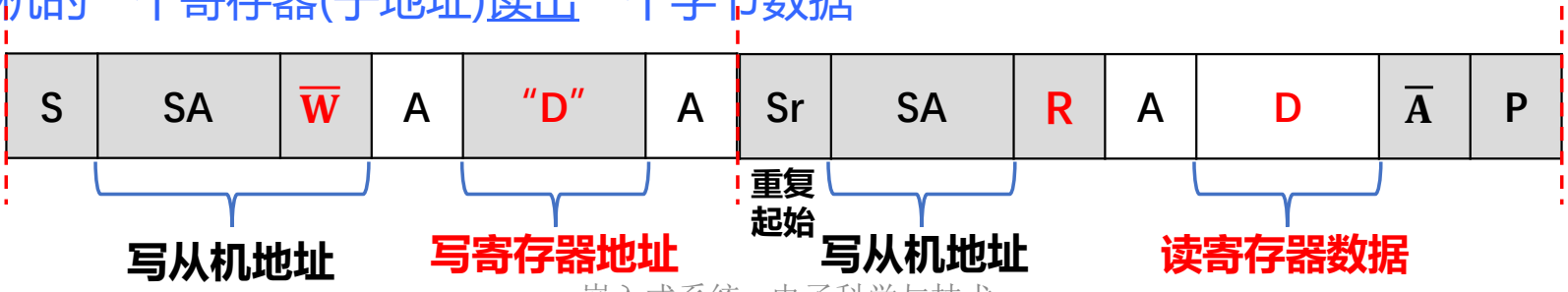
I2C高阶 – 重复起始和子地址

- **子地址**（Sub-address）
  - 有些I2C器件，除了自身的地址SA外，其内部还有若干个单元可被访问，相应具有子地址（寄存器）
  - 典型的如EEPROM存储器、本课实践用到的加速度传感器；
  - I2C器件子地址可以是1字节~N字节，通信时作为“数据”发送。

主机向从机的一个寄存器(子地址)写入一个字节数据



主机从从机的一个寄存器(子地址)读出一个字节数据

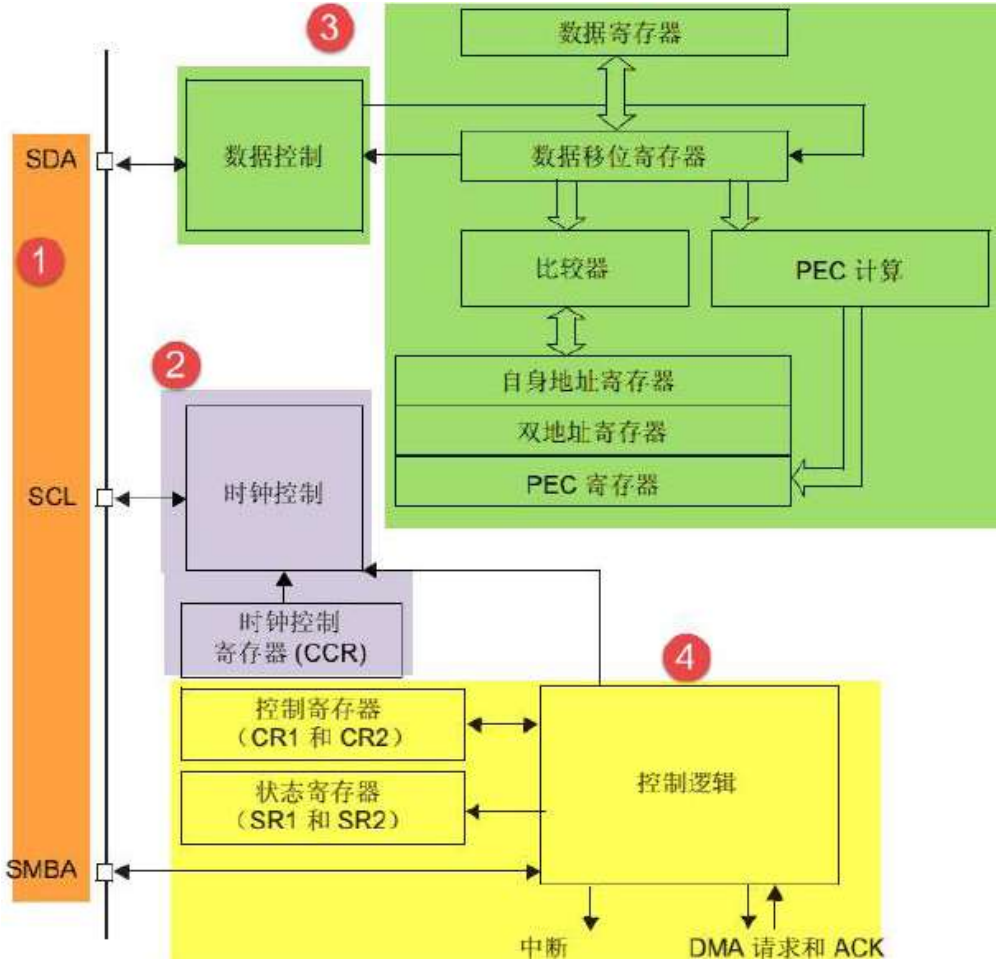


5.4 集成电路总线I2C

- Next .....
  - I2C通信简介
  - I2C通信的协议
  - STM32的I2C接口
  - I2C常用库函数
  - I2C编程实例 – 加速度传感器
  - 编程练习E5 ： I2C接口加速度传感器编程

I2C 基本结构

- **① 通信引脚**
  - -- 数据SDA、时钟SCL
- **② 时钟控制逻辑**
  - -- 传输模式 → 传输速率
  - -- 时钟占空比 → 采样时间(高电平)
- **③ 数据控制逻辑**
  - -- 移位寄存器 ↔ 数据寄存器
  - -- 数据校验计算
  - -- 从机地址匹配
- **④ 整体控制逻辑**
  - -- 控制寄存器CR1/CR2
  - -- 状态寄存器SR1/SR2



I2C 引脚资源

- **F407IG芯片**
  --数据手册 Chapter 3：Pinouts...description – Table 9. Alternate function mapping

引脚	APB1 总线 (max. 42MHz)		
	I2C1	I2C2	I2C3
SCL	PB6/PB8	PB10/PF0/PH4	PA8/PH7
SDA	PB7/PB9	PB11/PF1/PH5	PC9/PH8

传输速率：

-- 标准模式(Standard Mode)，max. 100Kbps

-- 快速模式(Fast Mode) max. 400Kbps

5.4 集成电路总线I2C

- Next .....
  - I2C通信简介
  - I2C通信的协议
  - STM32的I2C接口
  - I2C常用库函数
  - I2C编程实例 – 加速度传感器
  - 编程练习E5 ： I2C接口加速度传感器编程

常用库函数

- I2C_Init (I2C_TypeDef* I2Cx, I2C_InitTypeDef* I2C_InitStruct)

  功能：将串口 I2Cx 按照结构体 I2C_InitStruct 的参数进行初始化，

  结构体格式如下

```
typedef  struct
{
  uint32_t      I2C_Mode;                /* 工作模式：I2C或SMBUS模式 */
  uint16_t      I2C_DutyCycle;           /* SCL时钟占空比：Tlow/Thigh = 2:1或16:9 */
  uint32_t      I2C_ClockSpeed;          /* 传输速率：高速模式下≤400Kbps */
  uint16_t      I2C_Ack;                 /* 应答控制：使能或失能 */
  uint16_t      I2C_OwnAddress1;         /* (MCU）自身地址，7bits或10bits */
} I2C_InitTypeDef;                       (具体选择参见stm32f4xx_i2c.h第95行起)
  //定义初始化结构体变量
  I2C_InitTypeDef   I2C_InitStructure;
```

常用库函数

- I2C_SendData(I2C_TypeDef* I2Cx, uint8_t Data)

  功能：将数据Data通过串口I2Cx口发送出去

  -- Data[7:0]，8位数据，MSB在先

- uint8_t I2C_ReceiveData(I2C_TypeDef* I2Cx)

  功能：从I2Cx接收一个字节数据并将其返回

- I2C_Cmd(I2C_TypeDef* I2Cx, FunctionalState NewState)

  功能：使能/关闭I2Cx，NewStat = ENABLE或DISABLE

常用库函数

- I2C_AcknowledgeConfig (I2C_TypeDef* I2Cx, FunctionalState NewState)

  功能：打开/关闭I2Cx的应答功能，NewState：ENABLE-开启应答(A)；DISABLE-关闭应答($\overline{A}$)

- I2C_GeneratesSTART (I2C_TypeDef* I2Cx, FunctionalState NewState)

  功能：I2Cx的起始位(S)生成控制，NewState：ENABLE-生成，DISABLE-不生成

- I2C_GeneratesSTOP (I2C_TypeDef* I2Cx, FunctionalState NewState)

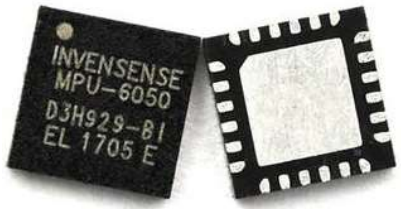
  功能：I2Cx的停止位(S)生成控制，NewState：ENABLE-生成，DISABLE-不生成

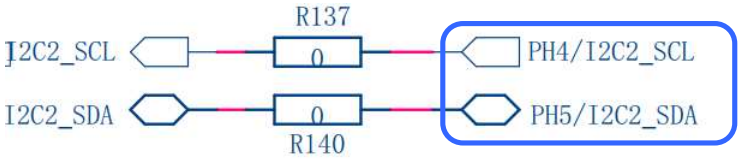
5.4 集成电路总线I2C

- Next .....
  - I2C通信简介
  - I2C通信的协议
  - STM32的I2C接口
  - I2C常用库函数
  - I2C编程实例 – 加速度传感器
  - 编程练习E5： I2C接口加速度传感器编程

I2C → 加速度传感器 Accelerometer

MPU6050-三轴加速度传感器和陀螺仪

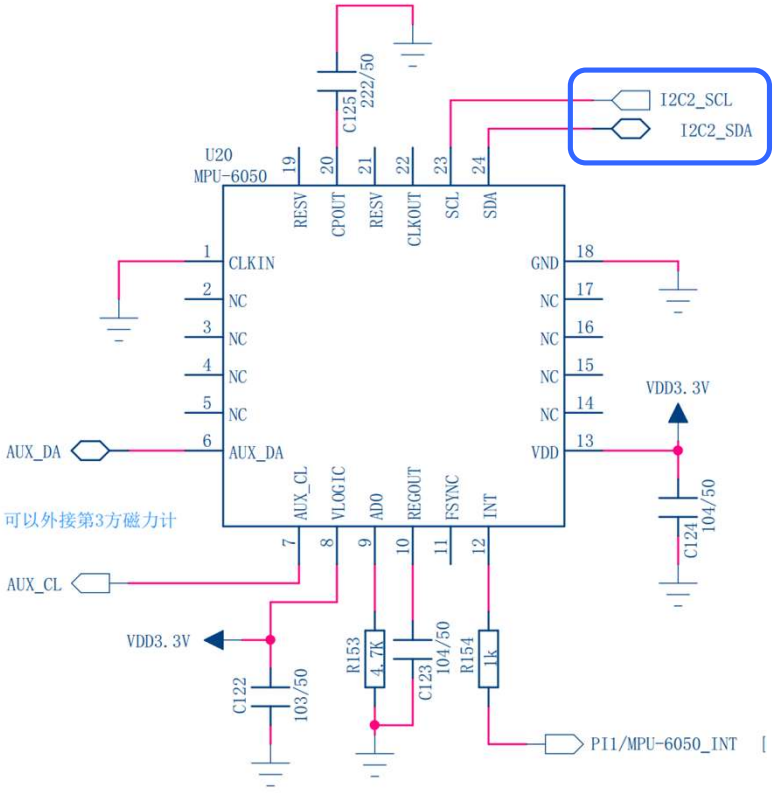




使用的I2C2引脚是

-- PH4：SCL 时钟

-- PH5：SDA 数据



I2C编程

- **1：I2C初始化**
  - 打开GPIO时钟、打开I2C时钟
  - 开启GPIO引脚复用功能
  - 配置GPIO具体参数、配置I2C具体参数
- **2：I2C高一级函数**
  - 标准函数之上的更高一级封装
  - 以字节为单位的设备读写函数
- **3：I2C高级函数应用**
  - 设备控制寄存器写入
  - 设备数据寄存器读出

I2C编程1 – 初始化

```
void  MPU6050_I2C2_Init(void)
{   GPIO_InitTypeDef       GPIO_InitStructure;
    I2C_InitTypeDef        I2C_InitStructure;
    //打开设备时钟
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOH, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C2,  ENABLE);
    //开启PH4-SCL, PH5-SDA复用为I2C通信的功能
    GPIO_PinAFConfig(GPIOH, GPIO_PinSource4,  GPIO_AF_I2C2);
    GPIO_PinAFConfig(GPIOH, GPIO_PinSource5,  GPIO_AF_I2C2);
    //初始化GPIO PH4-SCL, PH5-SDA 两个复用端
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;       /*引脚复用工作模式*/
    GPIO_InitStructure.GPIO_OType = GPIO_OType_OD;    /* OpenDrain- 开漏输出 */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
    GPIO_Init(GPIOH, &GPIO_InitStructure);
```

I2C编程1 – 初始化

```
//配置I2C参数
I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;          /* 工作模式 */
I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;  /* 占空比设置  Tlow : Thigh = 2 */
I2C_InitStructure.I2C_ClockSpeed = 100*10^3;        /*  时钟速度100kHz，最高400k */
I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;         /*  允许应答 */
I2C_Init(I2C2, &I2C_InitStructure);  /* 完成配置 */
I2C_Cmd (I2C2, ENABLE);              /* 使能I2C2 */
}
```

附：MCU6050数据手册：Chapter 6.7-I2C Timing Characterization(时序描述)

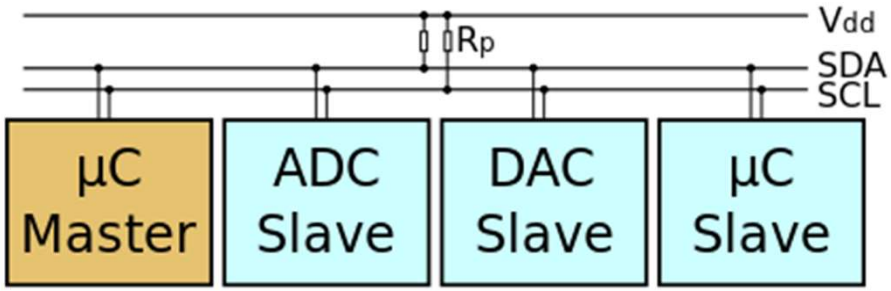
6.7 I^2C Timing Characterization

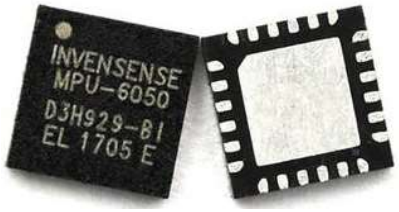
Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

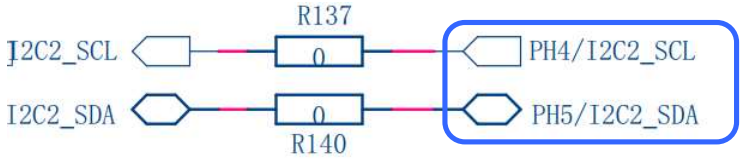
Parameters	Conditions	Min	Typical	Max	Units	Notes
I^2C TIMING	**I^2C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	

回顾 I2C – Inter-IC Bus

- 基本特征
  - 串行 – 2根线
  - 同步 – 时钟信号
  - 双向（非全双工）
  - 主从（Master/Slave）
  - 总线（Bus）
  - 协议（Protocol）

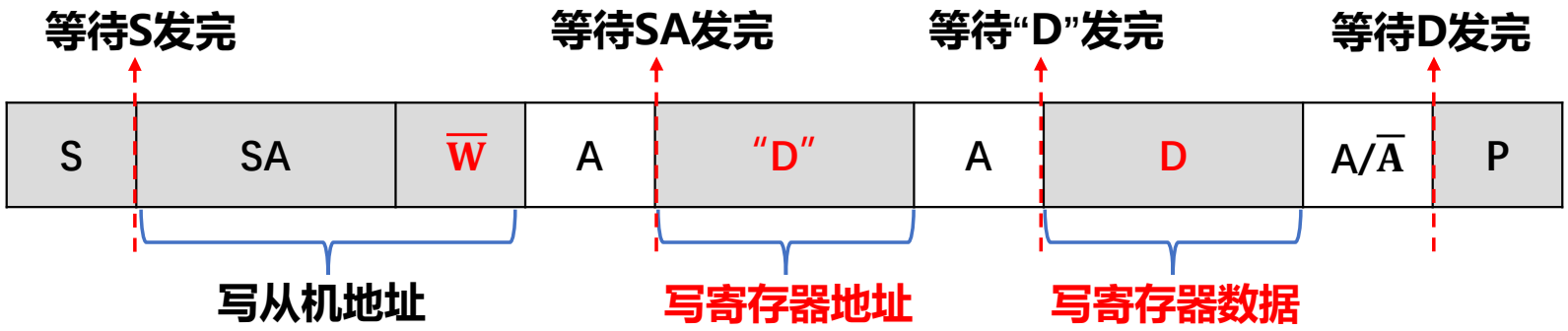




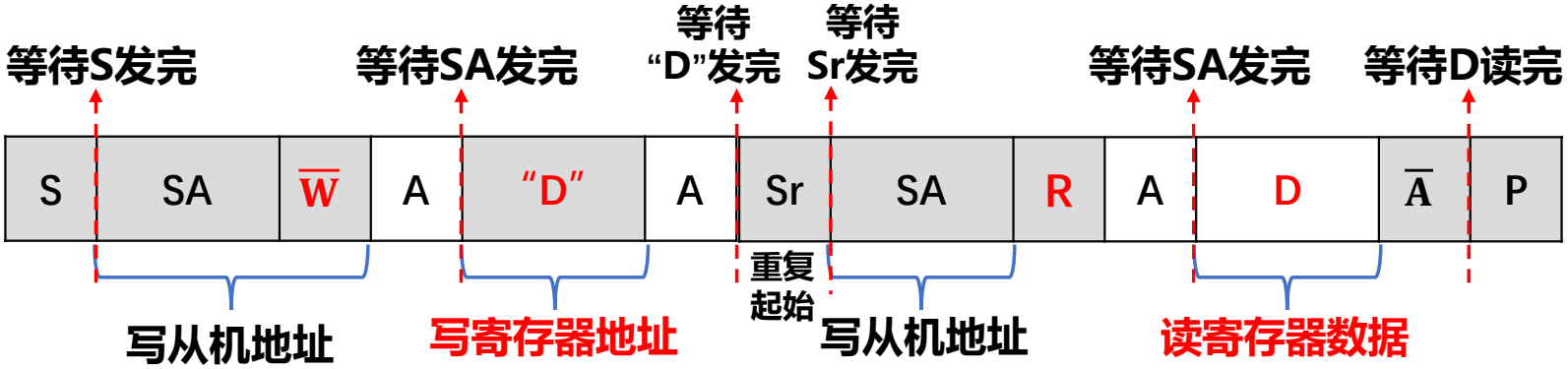


I2C – 设备寄存器读写数据帧

主机向设备的寄存器写入一个字节



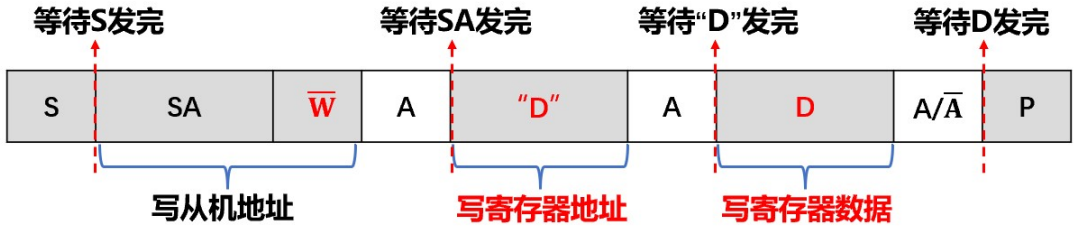
主机从设备的寄存器读出一个字节



I2C编程2 – 寄存器读写函数

- 设备寄存器写入函数（伪代码）

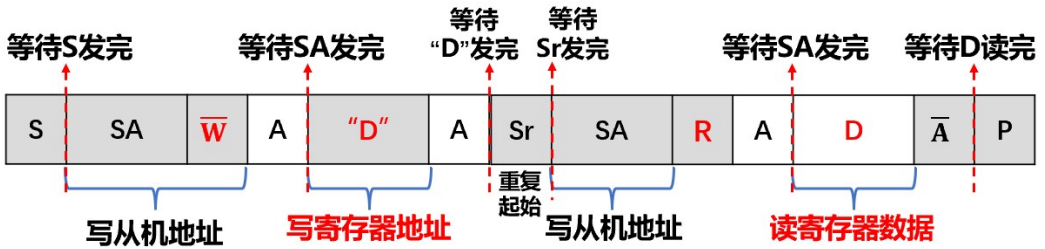
```
void  MPU6050_WriteReg(uint8_t  regaddr, int8_t  data)
{
  //I2C Start：调用库函数I2C_GeneratesSTART
  //等待起始信号发送完毕

  //发送一字节从机地址，标记“写”：调用库函数I2C_SendData
  //等待从机地址发送完毕

  //发送一字节寄存器地址：调用库函数I2C_SendData
  //等待子地址发送完毕

  //发送一字节寄存器数据：调用库函数I2C_SendData
  //等待数据发送完毕

  //I2C Stop：调用I2C_GeneratesSTOP
}
```



I2C编程2 – 寄存器读写函数

- 设备寄存器读出函数（伪代码）

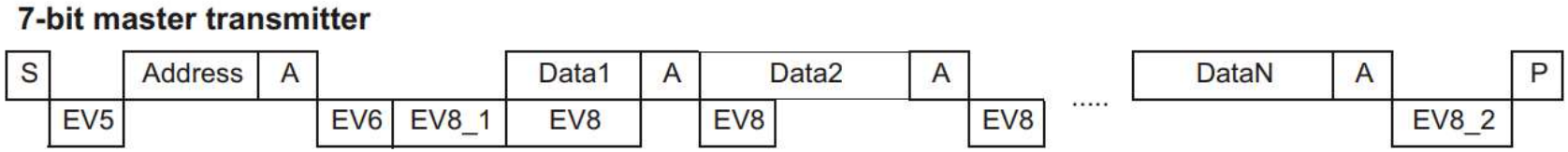
```
uint8_t  MPU6050_ReadReg(uint8_t  regaddr)
{
  //I2C Start
  //等待起始信号发送完毕

  //发送一字节从机地址，标记“写”
  //等待从机地址发送完毕

  //发送一字节寄存器地址
  //等待子地址发送完毕

  //I2C Start
  //等待重启信号Sr发送完毕

  //发送一字节从机地址，标记“读”
  //等待从机地址发送完毕

  //I2C关闭应答，即非应答NACK
    (只收一个字节情况)

  //等待数据接收完毕
  //读取一个字节寄存器数据

  //I2C Stop
  //I2C开启应答(之后通信)
}
```



I2C – 通信进程控制

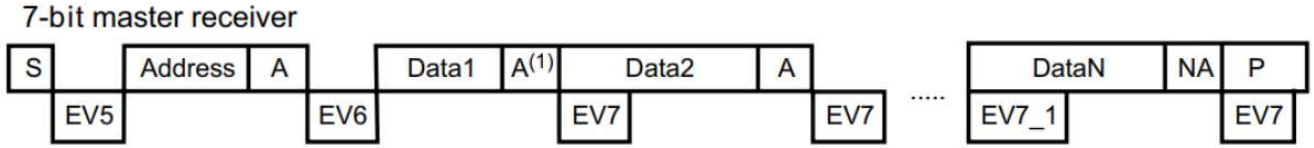
- 主机发送模式下的进程事件EVx (F407寄存器手册27.3.3 I2C master mode - Transmitter)

Figure 243. Transfer sequence diagram for master transmitter



- 主机接收模式下的进程事件EVx (F407寄存器手册27.3.3 I2C master mode - Receiver)

Figure 244. Transfer sequence diagram for master receiver



-- 到达某一进程时，CPU设置I2C状态寄存器中的相应标志位，从而产生相应事件(EVENT)

例如：事件EV8时，设置TxE位 = 1，代表数据发送完成　　通过设置标志位，状态寄存器

-- 用户通过查询标位为来判断事件的发生，来控制I2C通信的进程

I2C – 进程事件宏定义

/***************I2C通信进程事件说明*****************************/

*EV5 - Event 5，事件5发生代表I2C起始位发送完成

*EV6 - Event 6，事件6发生代表I2C“从机写”地址发送完成

*EV6 - Event 6，事件6发生也代表I2C“从机读”地址发送完成

*EV5 - Event 7，事件7发生代表I2C接收一字节数据完成

*EV5 - Event 8，事件8发生代表I2C发送一字节数据完成

/***/

Figure 243. Transfer sequence diagram for master transmitter

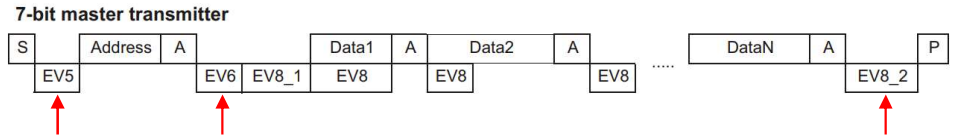
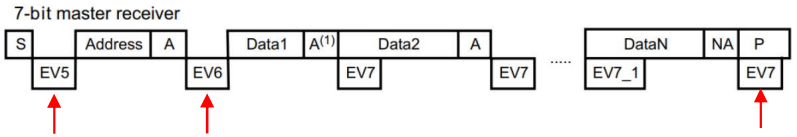


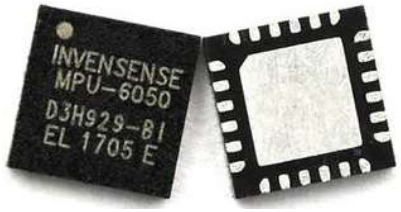
Figure 244. Transfer sequence diagram for master receiver



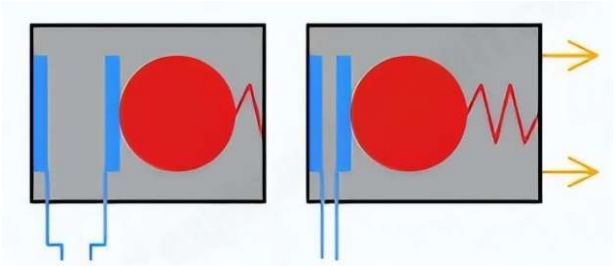

```
#define  I2C2_START_Wait()          while(! I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_MODE_SELECT) )                    //EV5
#define  I2C2_SendWrAddr_Wait()     while(! I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED) )  //EV6
#define  I2C2_SendRdAddr_Wait()     while(! I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED) )      //EV6
#define  I2C2_ReceiveData_Wait()    while(! I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_BYTE_RECEIVED) )               //EV7
#define  I2C2_SendData_Wait()       while(! I2C_CheckEvent(I2C2, I2C_EVENT_MASTER_BYTE_TRANSMITTED) )            //EV8
```

I2C → 加速度传感器 芯片手册

MPU6050-三轴加速度传感器和陀螺仪



电容加速度传感器测量原理



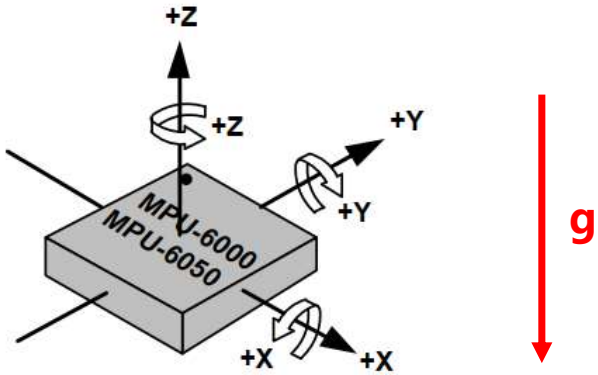
电容板间距在质量/力挤压下发生改变

容值*C*→ 受力*F*→ 加速度: $a = F/m$

测量输出

沿XYZ三个轴正向的加速度

绕XYZ三个轴旋转的角速度

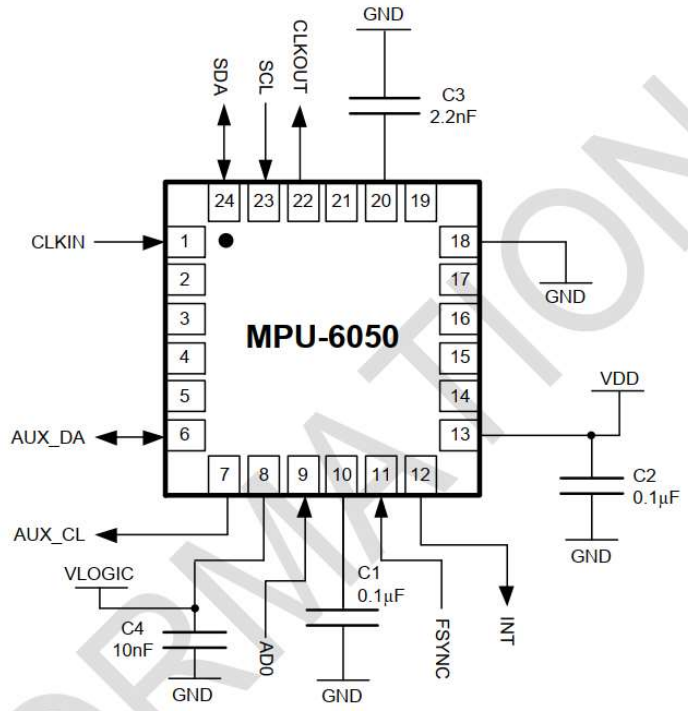


静态下：存在沿一个轴的向下的重力加速度g

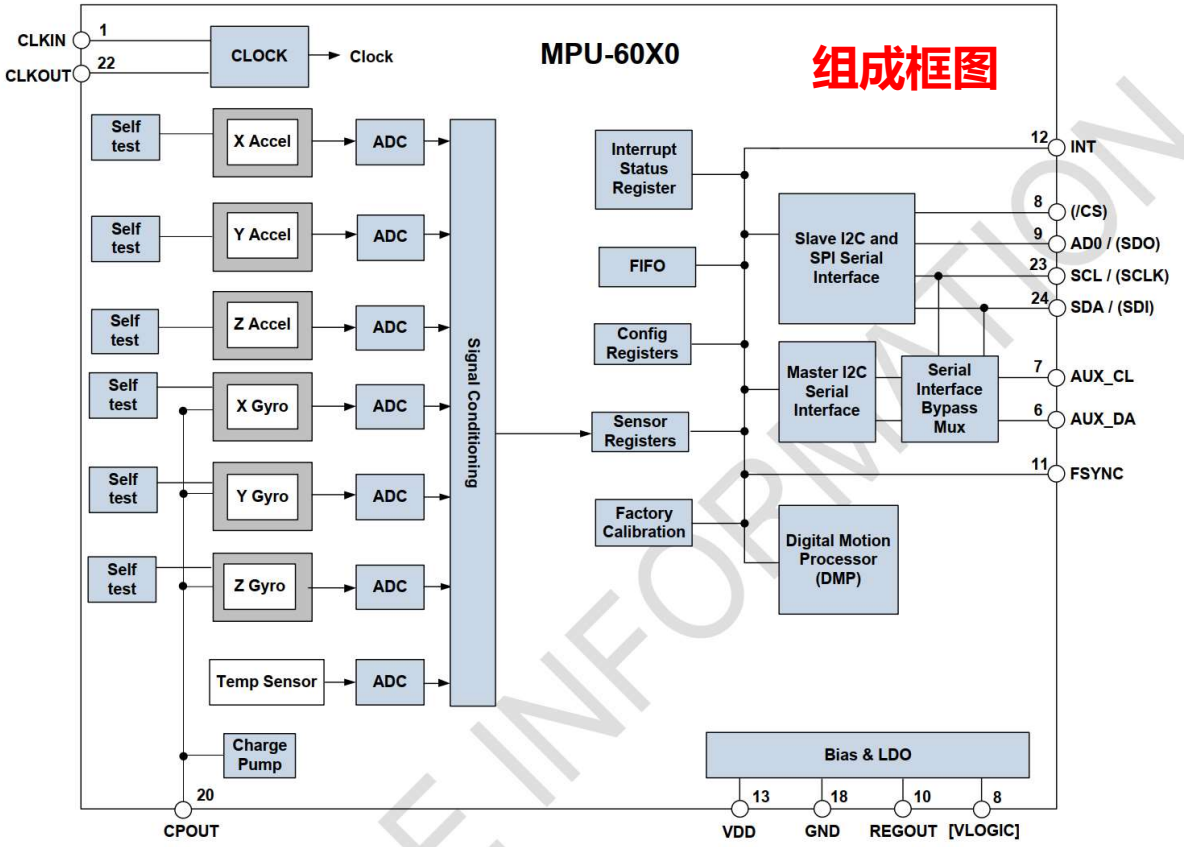
（但输出值为+g）

I2C → 加速度传感器 芯片手册

In Page 22, 24 of MCU60x0 Data Sheet



典型电路连接



I2C → 加速度传感器 芯片手册

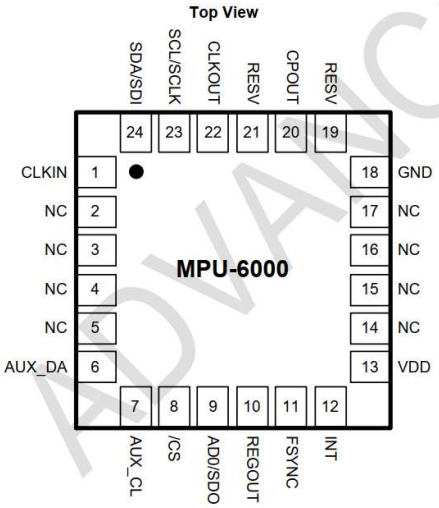
In Page 21 of MCU60x0 Data Sheet

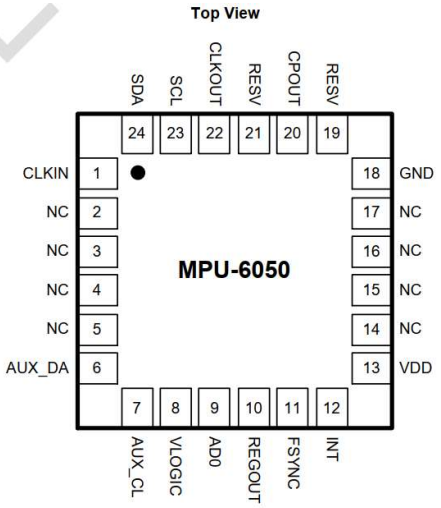
7.1 Pin Out and Signal Description

Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock (SCL)
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data (SDA)

姊妹型号

- 功能相同
- 6000型有SPI接口





I2C → 加速度传感器 芯片手册

In Page 34, 36 of MCU60x0 **Data Sheet**

9.2 I^2C Interface

The slave address of the MPU-60X0 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two MPU-60X0s to be connected to the same I^2C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin AD0 is logic low) and the address of the other should be b1101001 (pin AD0 is logic high).

MPU-60x0有7bit的地址：0b110100X, X由AD0引脚的逻辑决定

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

I2C时序图：单/多字节的读/写时序

I2C → 加速度传感器 寄存器手册

In Page 6 of MCU60x0 **Register Map and Descriptions**

3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
01	1	AUX_VDDIO	R/W	AUX _VDDIO	-	-	-	-	-	-	-
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		ACCEL_HPF[2:0]		

配置寄存器：

外部同步及低通滤波配置、陀螺仪配置、加速度计配置

I2C → 加速度传感器 寄存器手册

In Page 13 of MCU60x0 Register Map and Descriptions

Register 28 – Accelerometer Configuration　加速度测量配置寄存器
ACCEL_CONFIG

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
1C	28	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]		ACCEL_HPF[2:0]		

寄存器地址1C　**XYZ轴自检位**　**量程选择**　**高通滤波设置**

自检控制
0 – 不自检
1- 自检

AFS_SEL	Full Scale Range
0	± 2*g*
1	± 4*g*
2	± 8*g*
3	± 16*g*

ACCEL_HPF	Filter Mode	Cut-off Frequency
0	Reset	None
1	On	5Hz

```
//加速度传感器初始化函数
void MPU6050Init(void)
{ ……
  //不自检，量程±2g，高通5Hz
  MPU6050_WriteReg(0x1C, 0x01); /* 0x1c代表寄存器地址，0x01是写入的配置值 */
  …… }
```

I2C → 加速度传感器 寄存器手册

In Page 31 of MCU60x0 Register Map and Descriptions

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							

16bits加速度数据-X/Y/Z轴的H、L字节

```
//加速度数据寄存器读出函数
void MPU6050ReadData (.....)
{
    uint8_t  i,  readbuf[6];
    for (i = 0; i < 6; i++)
        readbuf[i] = MPU6050_ReadReg (0x3B + i);  /* 从X轴高字节地址开始，连续读出6个字节数据 */
..... }
```

测量范围	分辨率
±2g	1g/16384
± 4g	1g/8192
± 8g	1g/4096

分辨率 = 加速度范围/数字量范围

= (+2g ~ -2g)/(+32767 ~ -32768)

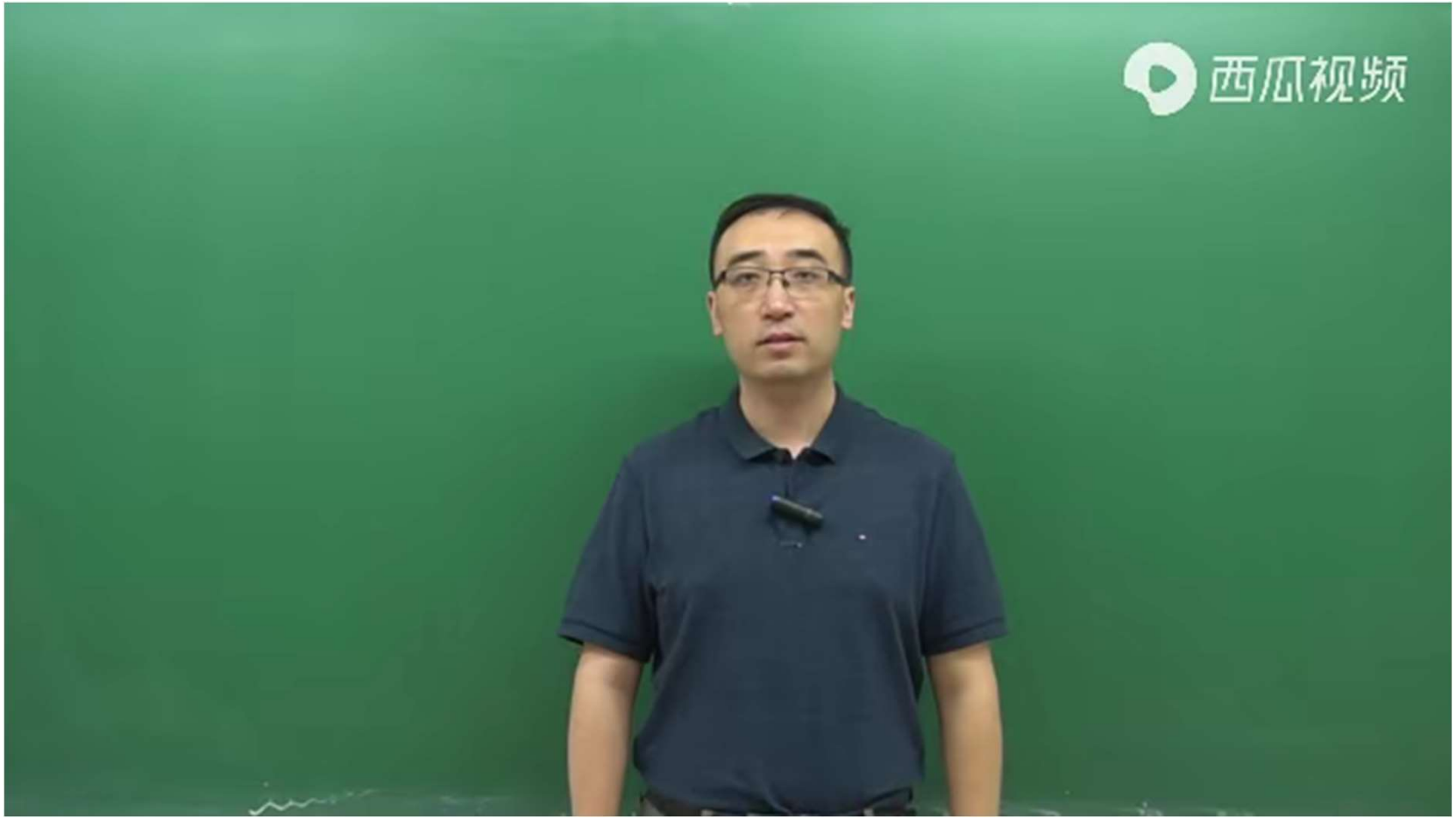
= 1g/16384　　(即1g ⇔ 16384)

4g/2**16

范围越大，分辨率越小

李永乐讲加速度与缓冲

老哥的一句“我先走了”，成生命中最后一句话！高空跳水有多危险？李永乐老师对你说：电影里都是骗人的！



Do a test

- **坠地瞬间加速度实验**
  - 高度 40cm（地面放一绒布垫）
  - 理论计算

    -- 落地前速度 $V = \sqrt{2gh} \approx 2.83m/s$

    -- 落地时加速度 a = (0-2.83)/0.1s = 2.83g
  - 实测结果

    -- 数据以2g~3g范围为最多

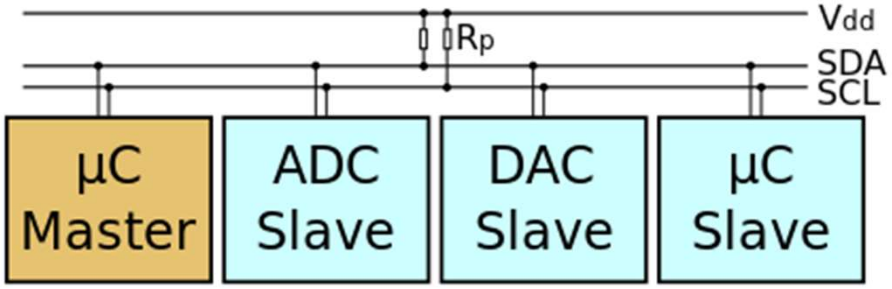
    -- 取三次平均（2.23+2.54+2.45)/3 ≈ 2.41g

Have a try

使用I2C，

让加速传感器用起来，

让实验板玩起来~



5.4 集成电路总线I2C

- Next .....
  - I2C通信简介
  - I2C通信的协议
  - STM32的I2C接口
  - I2C常用库函数
  - I2C编程实例 – 加速度传感器
  - 编程练习E5 ： I2C接口加速度传感器编程

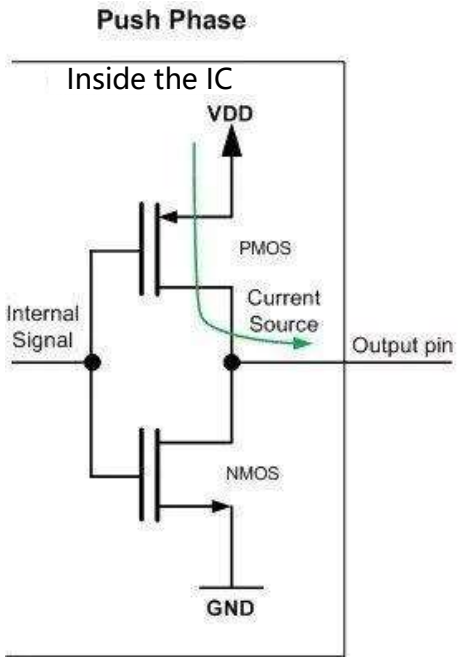
✓ 编程练习E5.1&E5.2
I2C接口-加速度传感器
(配套视频E5.1)

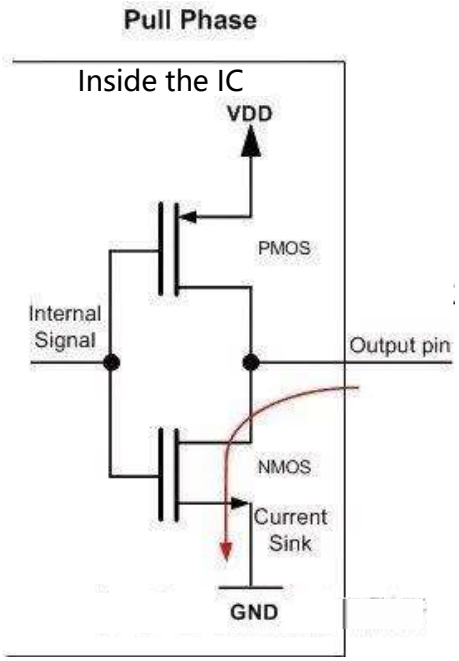
思考题 – I2C总线

- I2C通信方式为什么是“半双工”
- I2C两根信号线的名称及用途
- 推挽输出和OD输出结构上的区别？OD门线与连接原理(为何通过一根导线就可实现与逻辑)
- I2C的信号电平规范（三点）
- 描述I2C主机写从机寄存器的数据帧格式（课件20页）
- 描述I2C主机读从机寄存器的数据帧格式（课件20页）
- MPU6050的为什么可以有两个7bit地址？分别是什么？
- 若MPU6050初始化配置为：不自检，量程±4g，高通滤波5Hz，则语句“MPU6050_WriteReg(0x1C, 0x01);”该如何修改？（课件44页）
- 如果想读取MPU6050三个轴的角速度数据，则语句“for (i = 0; i < 6; i++) {
  readbuf[i] = MPU6050_ReadReg (0x3B + i);}”该如何修改？（课件45页结合MPU6050寄存器手册）

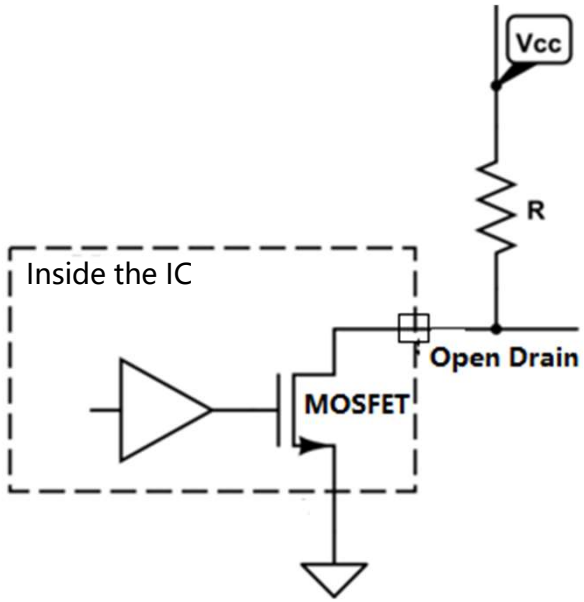
推挽输出与开路输出

- 推挽输出结构 (Push pull)
  -- 2只MOS管
  -- 自身能输出1和0

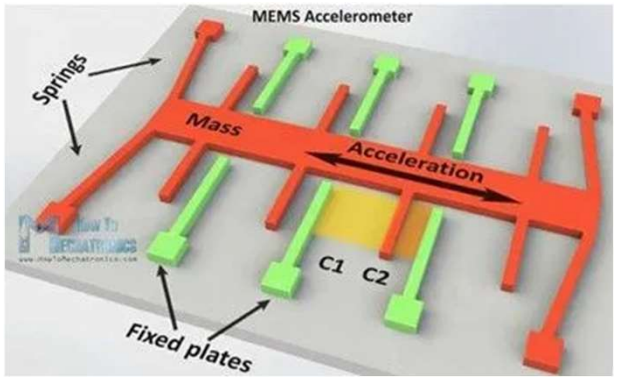


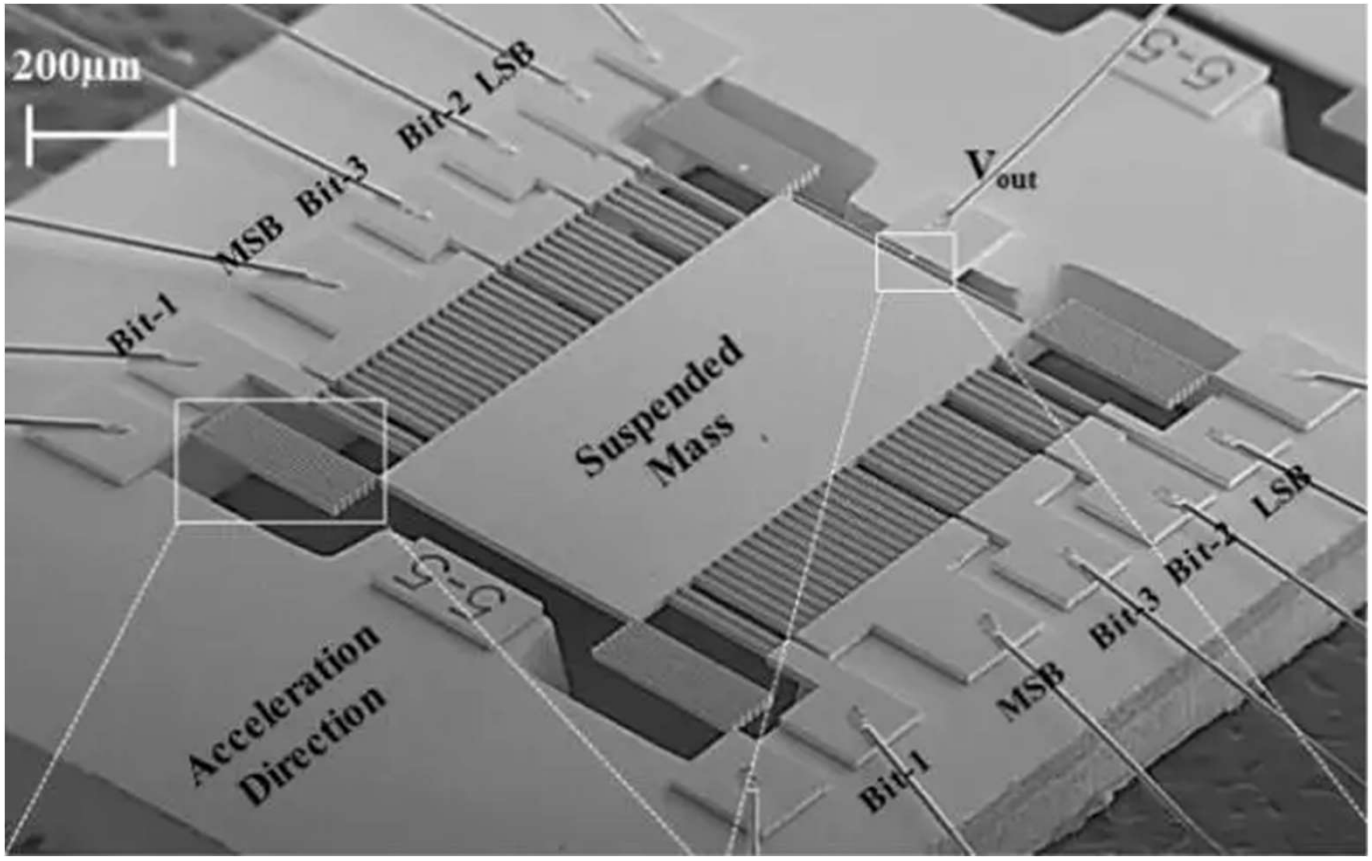


- 漏极开路输出（Open Drain）
  -- 1只MOS管
  -- 截止时，依靠外部上拉输出1
  -- 导通时，依靠接地下拉输出0



MEMS工艺的运动传感器





科技带来足球革命

- **2022卡塔尔世界杯**

  逐梦之旅Al Rihla内置球“芯”

包含一个惯性测量单元（IMU）传感器，能记录足球的初始速度、运行速度、运动角度等数据。