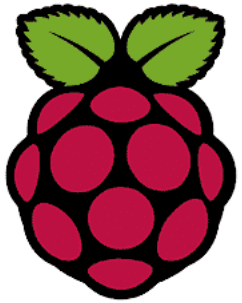


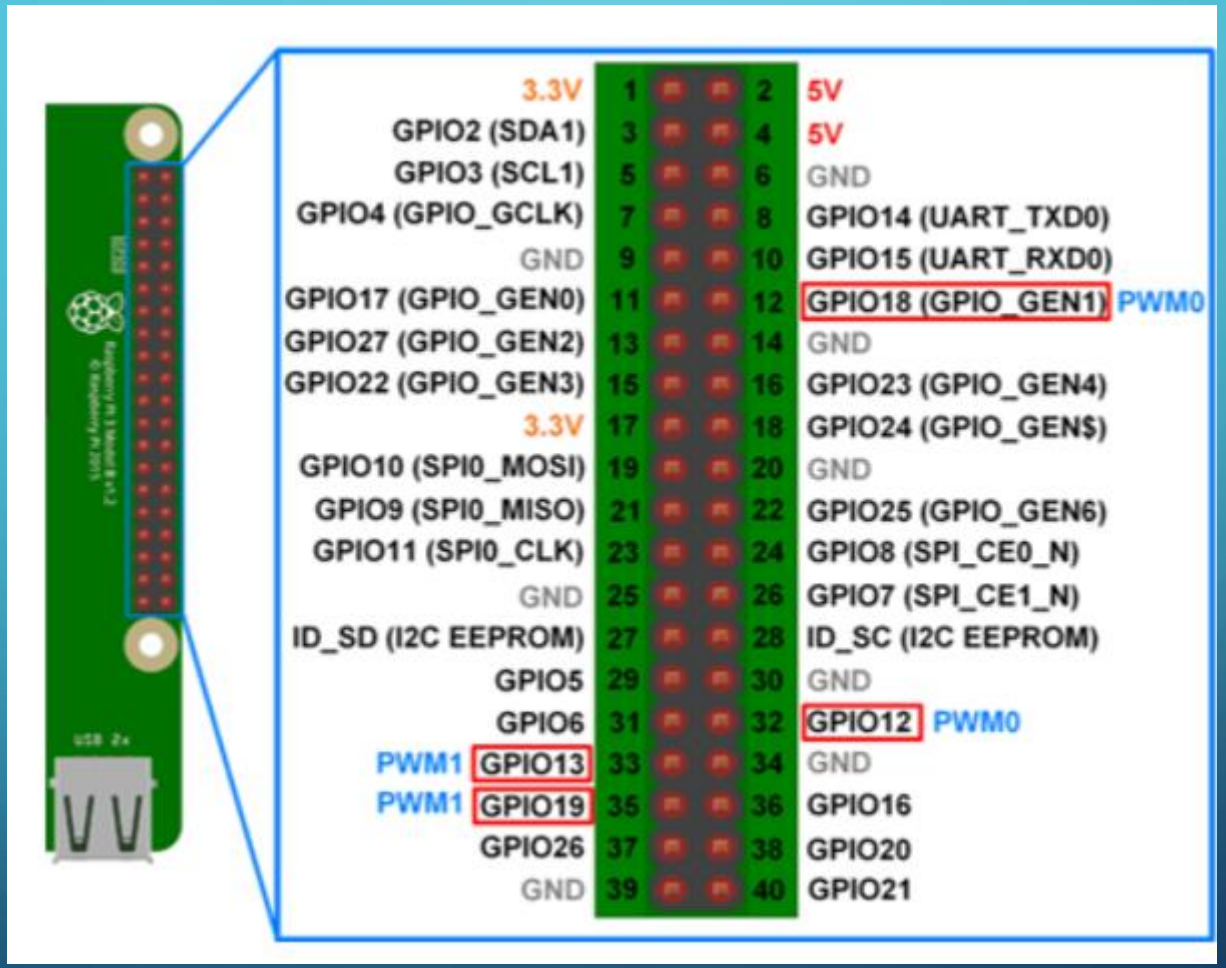
RaspberryPi

PWM, ADC(SPI)

PMOD 김효진

목차

1) Hardware PWM



1) Hardware PWM

void pinMode(int pin , int mode)
pin : wiringPi 핀번호(BCM번호 아님)
mode : PWM_OUTPUT pwm 출력 지원
- 하드웨어 pwm핀 번호는 채널2개 핀 4개
BCM기준 gpio18(pwm0),gpio12(pwm0),gpio13(pwm1),gpio19(pwm1)
- main 프로그램에 관계없이 계속 출력함

pwmSetMode(int mode);
mode : PWM_MODE_MS(mark:space mode), PWM_MODE_BAL(balanced mode)
- 위의 함수로 모드 변경 가능 WiringPi는 defalut값이 Balanced 모드임
- mark space 사이클 시작부터 듀티비율 부분까지 high 후 off 됨(서보모터 사용시 일반적인 pwm 파형모드)
- balanced mode 전체 주기에 걸쳐 high된 시간이 분배됨(그림으로 비교하면 이해하기 편함)

pwmSetClock (int divisor);

divisor : 제수(나누는 수)
전체주기 = divisor/19.2Mhz
- bcm2835PWMClockDivider에서 PWM CLOCK 19.2MHz

pwmSetRange(unsigned int range);

range : 분해능 defalut=1024
(divisor/19.2Mhz)*(1/range) : on되는 시간

void pwmWrite(int pin,int value);
pin : 출력할 핀번호
value : 0~1024
- value/1024의 듀티비를 갖는 PWM 파형을 PWM레지스터에 기록함

1) Software PWM

- 낮은 cpu 사용량을 유지하기 위해 최소 펄스 폭은 100us이다.
- 펄스 폭 변경시 100us 미만의 지연 필요
- 프로그램 실행중 PWM신호 비활성화 못함
- PWM출력 유지하려면 프로그램 계속 실행해야함.
- 원하는 핀에 PWM신호 출력가능
- 여러 개 사용시 CPU에 부하걸림
- duty ratio,frequency 조절 어려움

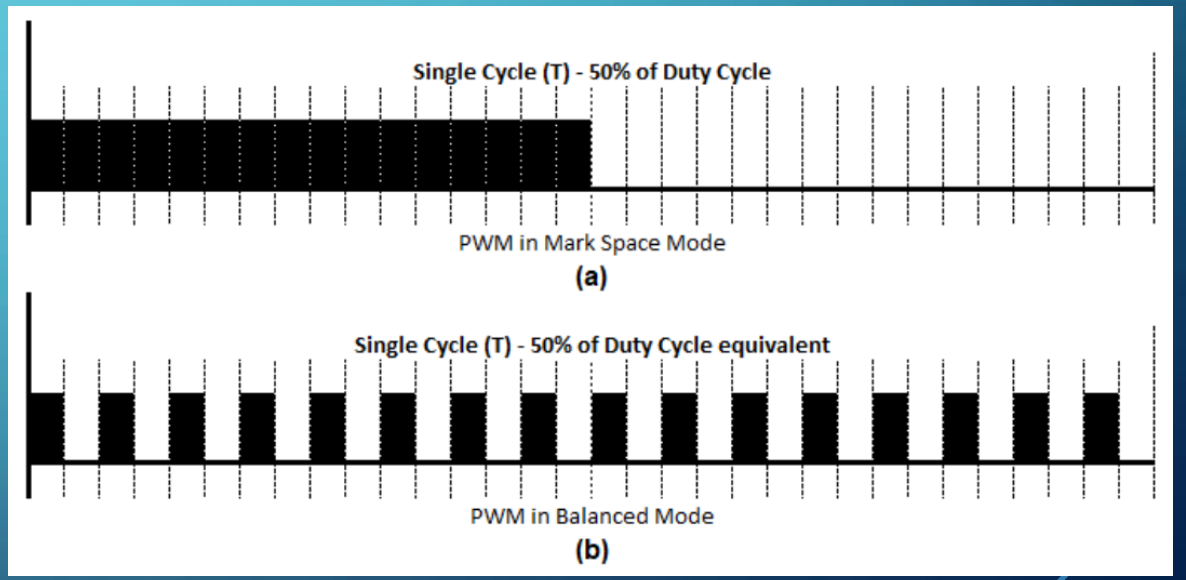
2) Software PWM

- #include <softPwm.h>
- pinMode 동일
- 제약
    1. To maintain a low CPU usage,the minimum pulse width is 100µS.
    2. That combined with the default suggested range of 100 gives a PWM frequency of 100Hz.
    3. If you change the pulse-width in the driver code,
       then be aware that at delays of less than 100µS wiringPi does it in a software loop,
       which means that CPU usage will rise dramatically,
       and controlling more than one pin will be almost impossible.
    4. Also note that while the routines run themselves at a higher and real-time priority,
       Linux can still affect the accuracy of the generated signal.
    5. There is currently no way to disable softPWM on a pin while the program in running.
    6. You need to keep your program running to maintain the PWM output!
- gcc -o myprog myprog.c -lwiringPi -lpthread (pthread library를 포함 시켜야만함!)

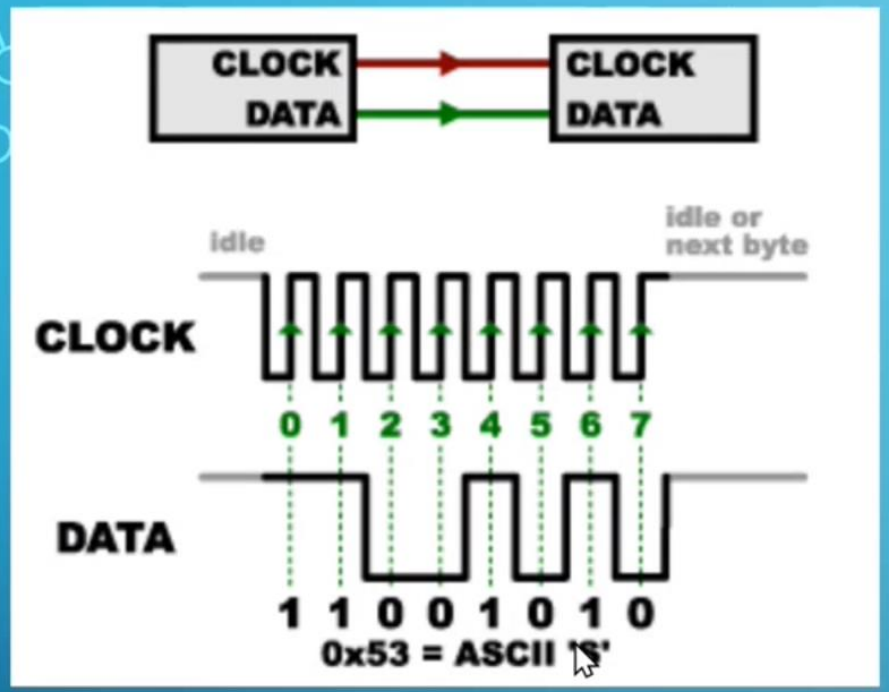
int softPwmCreate(int pin, int initialValue, int pwmRange);
    pin : pwm신호를 출력할 핀 번호
    initialValue : default duty ratio
    pwmRange : pwmRange*0.1ms 주기를 가지는 pwmRange단계의 pwm 신호를 만듬
    return : 0 for success

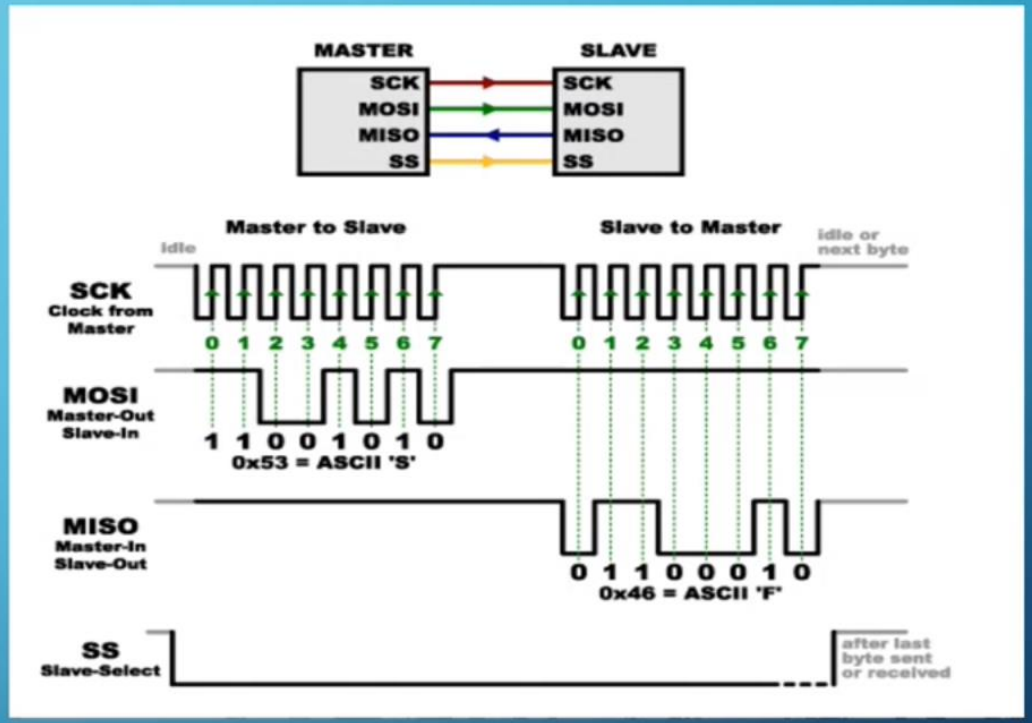
void softPwmWrite(int pin , int value);
    pin : pwm신호를 출력할 핀 번호
    value : value/pwmRange=duty ratio
        value<=pwmRange

3) Mark space, Balanced Mode



4) SPI





- Serial Peripheral Interface
- 동기식 전이중통신
- 하나의 Master, 여러 개의 Slave
- 1:N 통신
- Mutislave 모드

장점	단점
• 프로토콜 유연성(8bit제한 x) • 단순한 IC 하드웨어 처리 • CLOCK 속도 제한 X	• 하드웨어 슬레이브 인식 X • 오류 검사 프로토콜 X • 노이즈 스파이크에 많이 영향 • 짧은 거리에서 동작 • 하나의 마스터 장치만 지원

4) SPI

SPI Mode	CPOL	CPHA	Clock Polarity in Idle State	Clock Phase Used to Sample and/or Shift the Data
0	0	0	Logic low	Data sampled on rising edge and shifted out on the falling edge
1	0	1	Logic low	Data sampled on the falling edge and shifted out on the rising edge
2	1	1	Logic high	Data sampled on the falling edge and shifted out on the rising edge
3	1	0	Logic high	Data sampled on the rising edge and shifted out on the falling edge

4) SPI

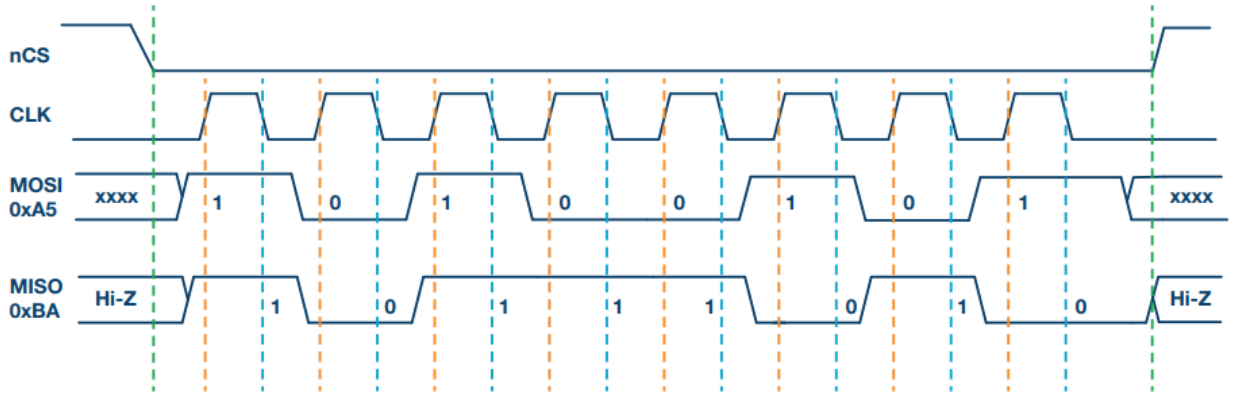


Figure 2. SPI Mode 0, CPOL = 0, CPHA = 0: CLK idle state = low, data sampled on rising edge and shifted on falling edge.

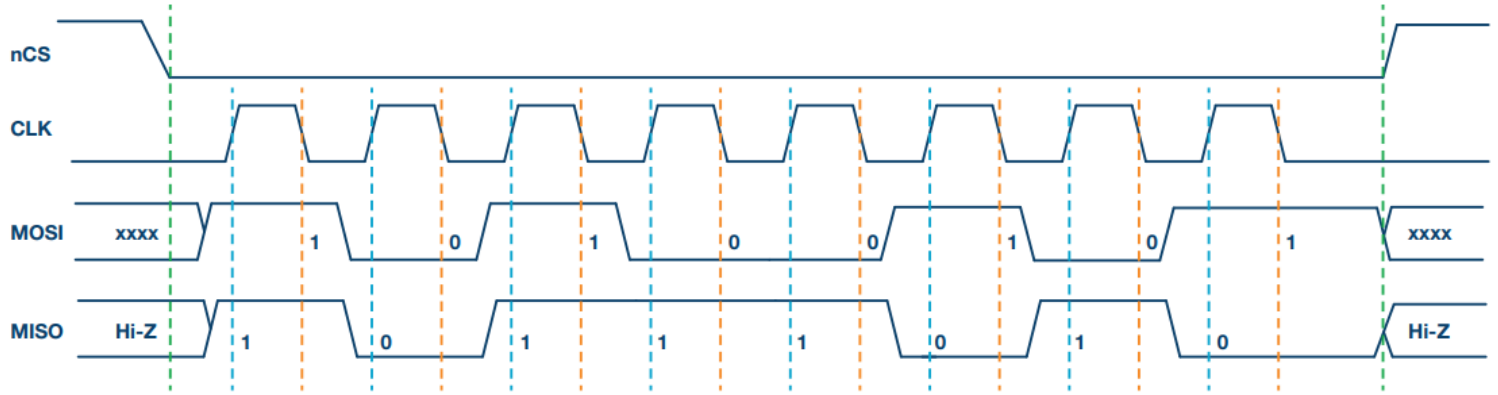


Figure 3. SPI Mode 1, CPOL = 0, CPHA = 1: CLK idle state = low, data sampled on the falling edge and shifted on the rising edge.

4) SPI

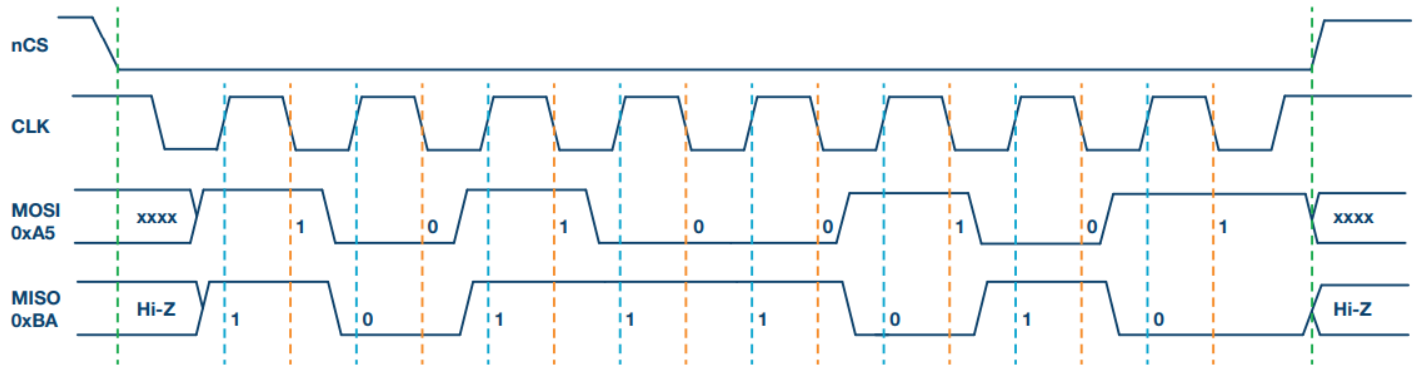


Figure 4. SPI Mode 2, CPOL = 1, CPHA = 1: CLK idle state = high, data sampled on the falling edge and shifted on the rising edge.

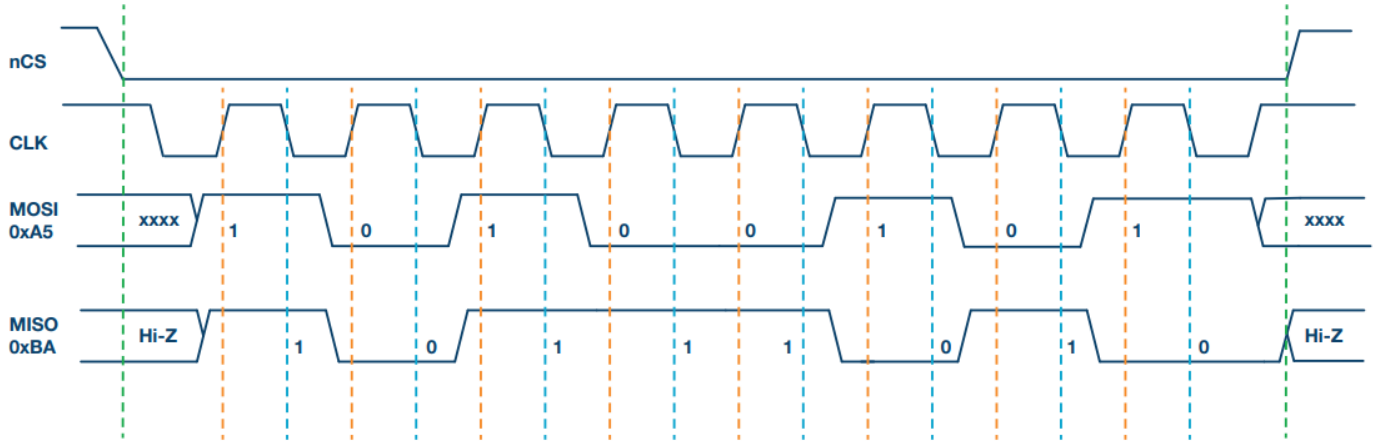
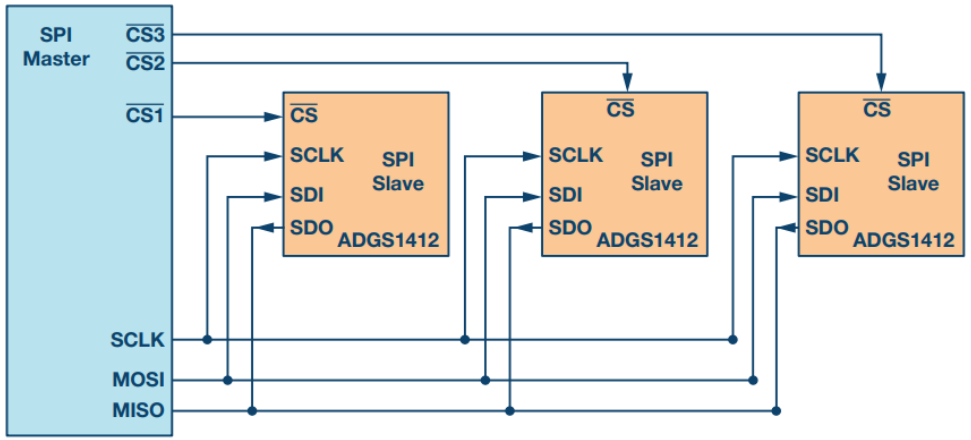


Figure 5. SPI Mode 3, CPOL = 1, CPHA = 0: CLK idle state = high, data sampled on the rising edge and shifted on the falling edge.

4) SPI



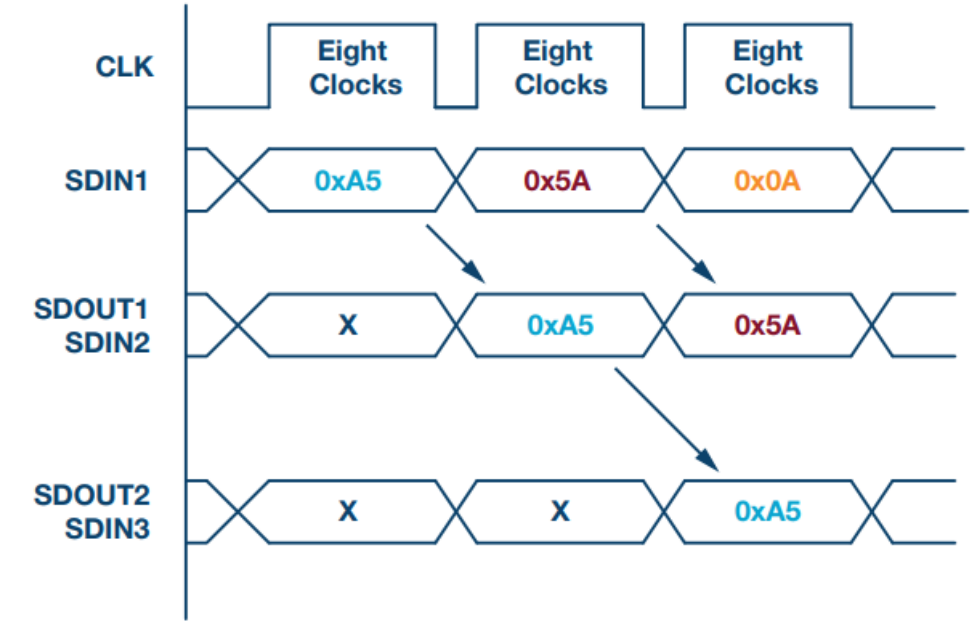
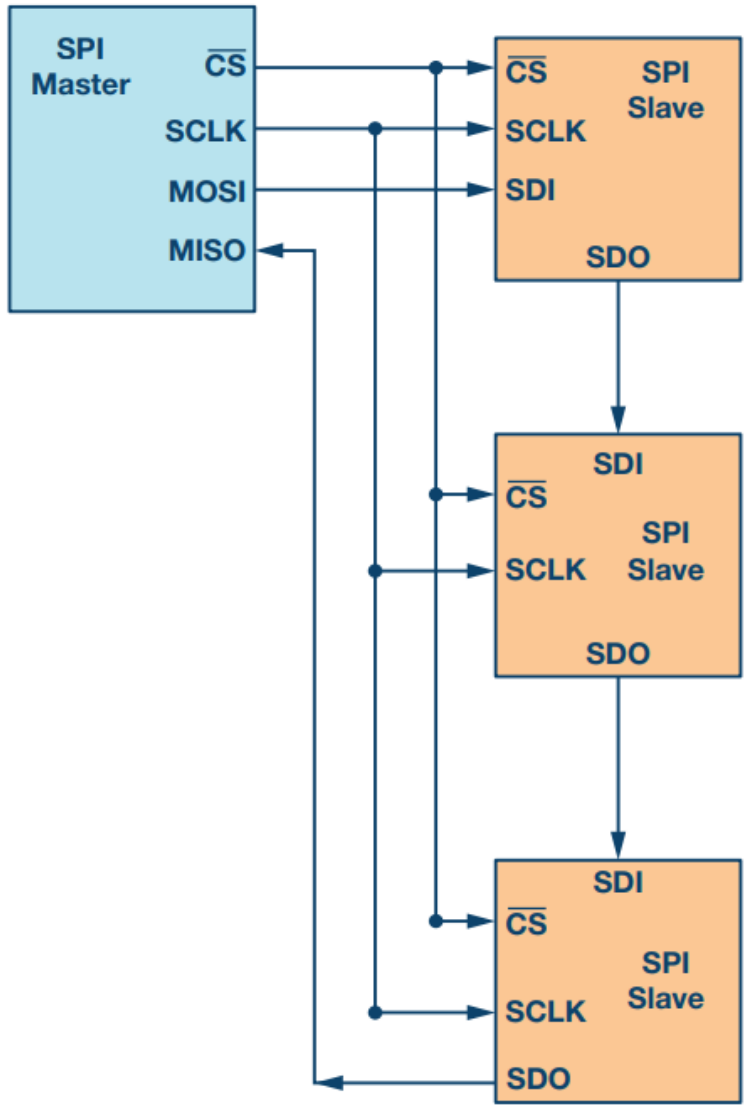


Figure 8. Daisy-chain configuration: data propagation.

Daisy-Chain Method:



4) SPI

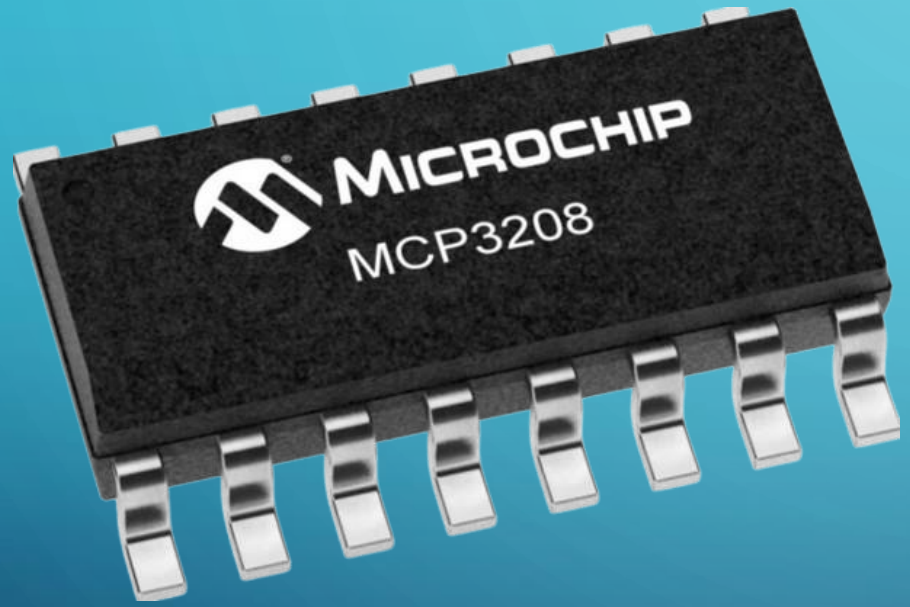
#include <wiringPiSPI.H>

- int wiringPiSPISetup(int channel,int speed);
  channel : 채널 0,1을 선택하여 초기화
  speed : 500khz~32000khz범위 내의 정수형 clock speed 설정
  return value : 오류시 -1 or Linux file_descriptorfor device
  ,standard errno global variable를 사용하여 원인 파악 가능

- int wiringPiSPIDataRW(int channel. unsigned char *dat, int len);
  channel : 초기화된 채널 선택
  *data : 3개의 8bit buffer에 있는 데이터가 자동 전송 수신된 되이터는 buffer에 덮어쓰기
  len : buffer size
  선택된 spi bus에서 동시에 read/write, buffer에 있던 data는 새로운 data에 의해 덮어짐
  A/D, D/A converter는 동시에 쓰기/읽기 수행을 해야하함

- int wiringPiSPIGetFd(int channel);
  채널에 대한 fd를 반환

5) MCP3208-BI/P



- 8채널,12-bit A/D Converter
- 라즈베리파이는 내장 ADC X
- SAR ADC(축차 비교형)
- SPI환경

5) MCP3208-BI/P

Features

- 12-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL (MCP3204/3208-B)
- ± 2 LSB max INL (MCP3204/3208-C)
- 4 (MCP3204) or 8 (MCP3208) input channels
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 100ksps max. sampling rate at VDD = 5V
- 50ksps max. sampling rate at VDD = 2.7V
- Low power CMOS technology: - 500nA typical standby current, 2µA max. - 400µA max. active current at 5V
- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

5) MCP3208-BI/P

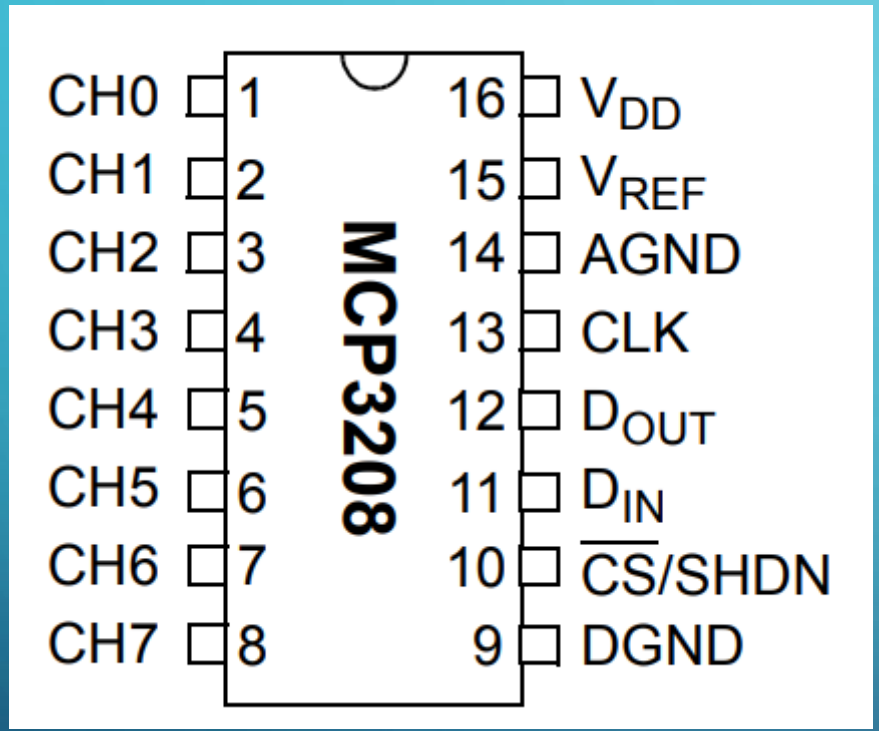
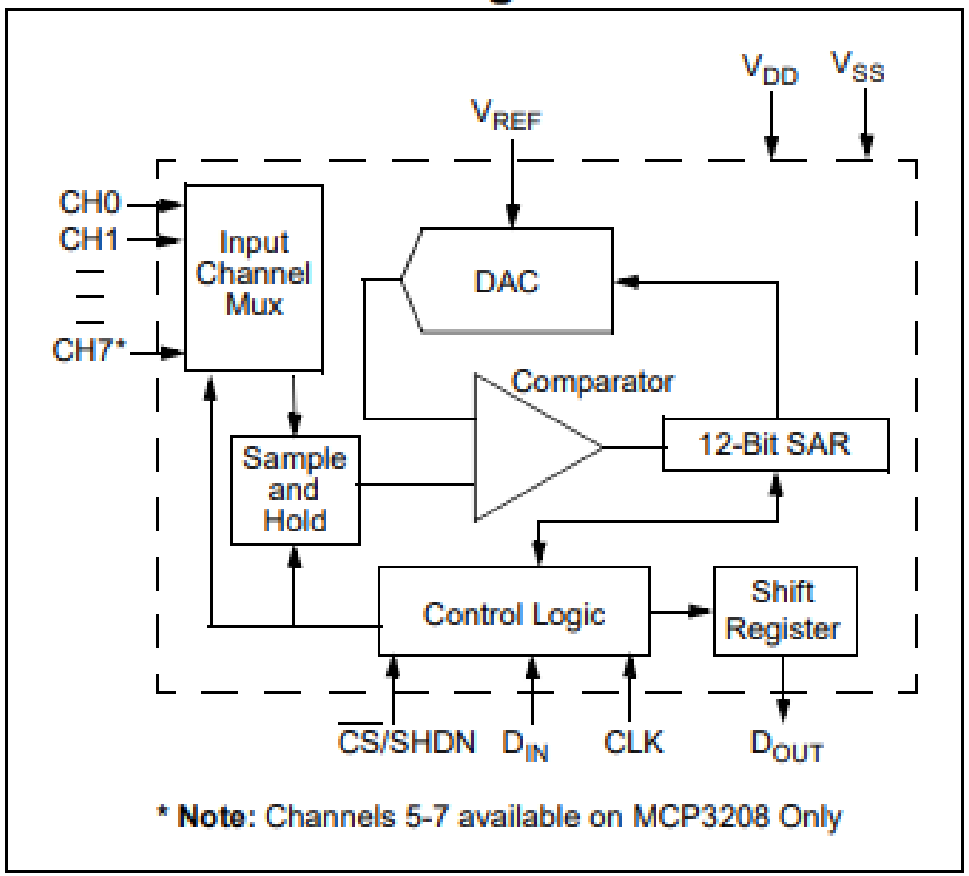


TABLE 3-1: PIN FUNCTION TABLE

Name	Function
V_{DD}	+2.7V to 5.5V Power Supply
DGND	Digital Ground
AGND	Analog Ground
CH0-CH7	Analog Inputs
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
$\overline{CS}$/SHDN	Chip Select/Shutdown Input
V_{REF}	Reference Voltage Input

5) MCP3208-BI/P

Functional Block Diagram



* **Note:** Channels 5-7 available on MCP3208 Only

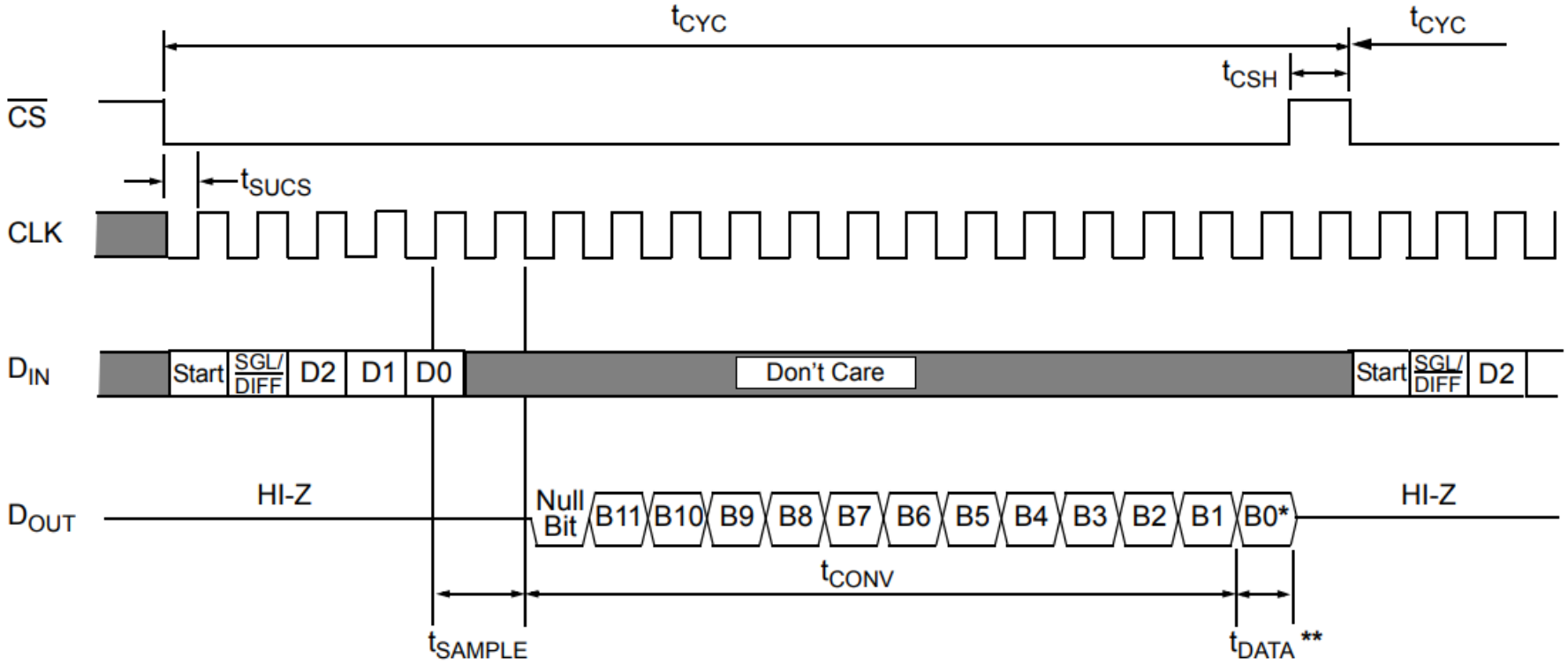
EQUATION

$$Digital\ Output\ Code = \frac{4096 \times V_{IN}}{V_{REF}}$$

V_{IN} = *analog input voltage*

V_{REF} = *reference voltage*

5) MCP3208-BI/P

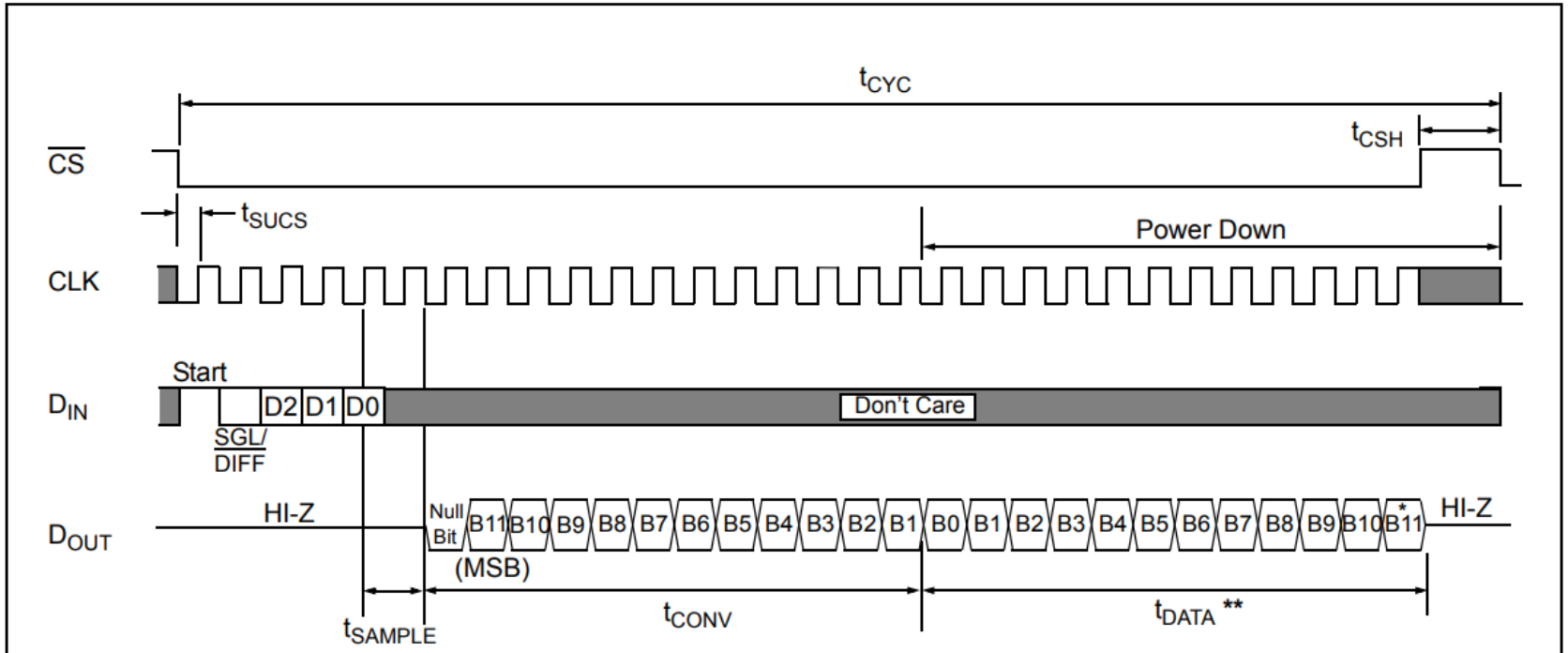


* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the A/D converter will output LSB first data, followed by zeros indefinitely (see Figure 5-2 below).

** t_{DATA}: during this time, the bias current and the comparator power down while the reference input becomes a high impedance node, leaving the CLK running to clock out the LSB-first data or zeros.

FIGURE 5-1: *Communication with the MCP3204 or MCP3208.*

5) MCP3208-BI/P



* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the A/D converter will output zeros indefinitely.

** t_{DATA}: During this time, the bias circuit and the comparator power down while the reference input becomes a high impedance node, leaving the CLK running to clock out LSB first data or zeroes.

FIGURE 5-2: *Communication with MCP3204 or MCP3208 in LSB First Format.*

5) MCP3208-BI/P

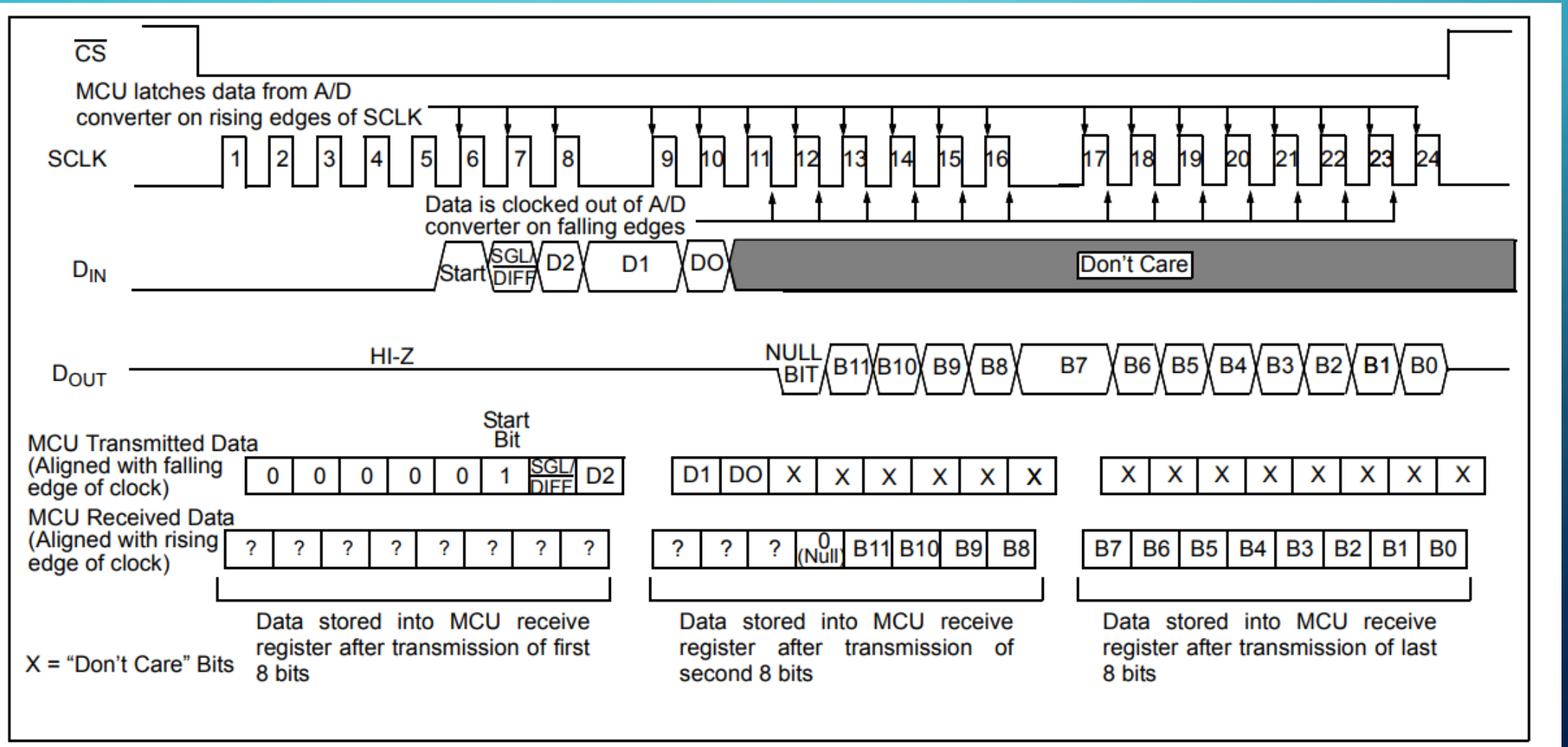


FIGURE 6-1: *SPI Communication using 8-bit segments (Mode 0,0: SCLK idles low).*

5) MCP3208-BI/P

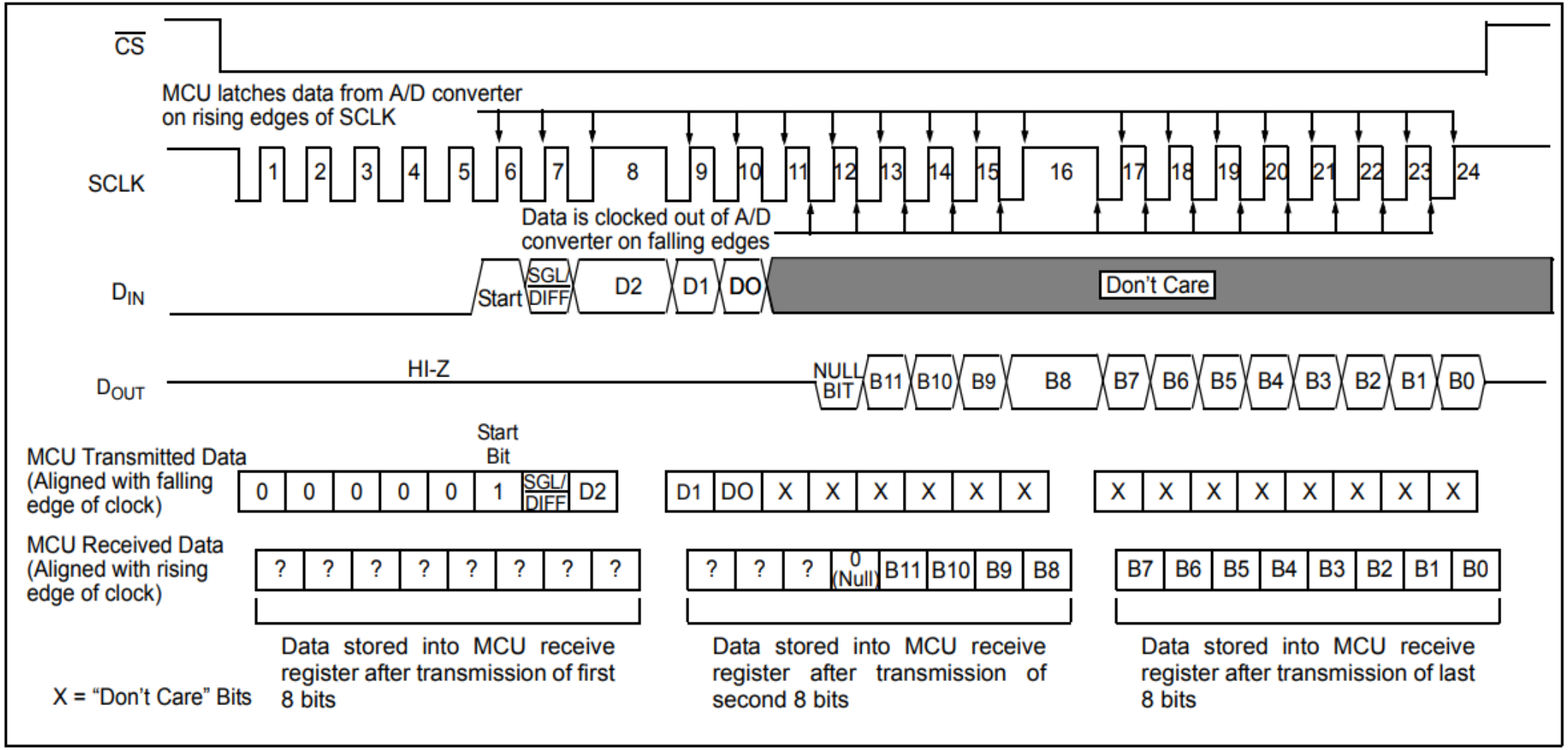


FIGURE 6-2: *SPI Communication using 8-bit segments (Mode 1,1: SCLK idles high).*

5) MCP3208-BI/P

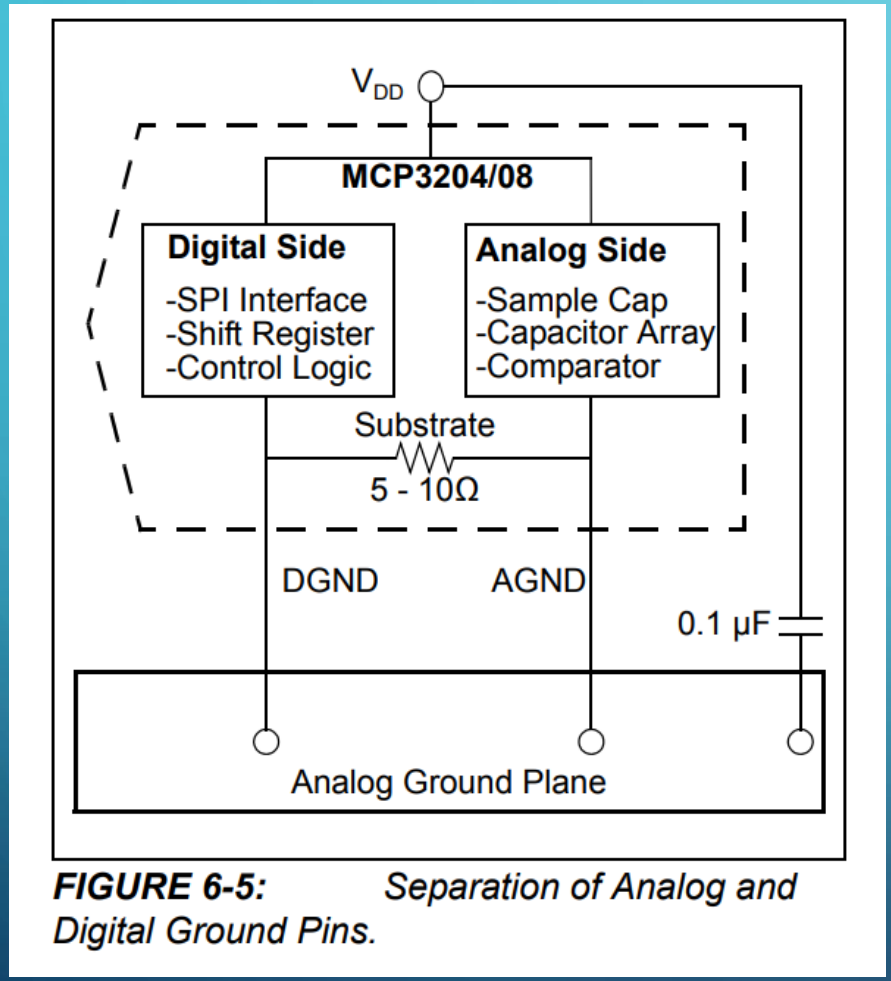
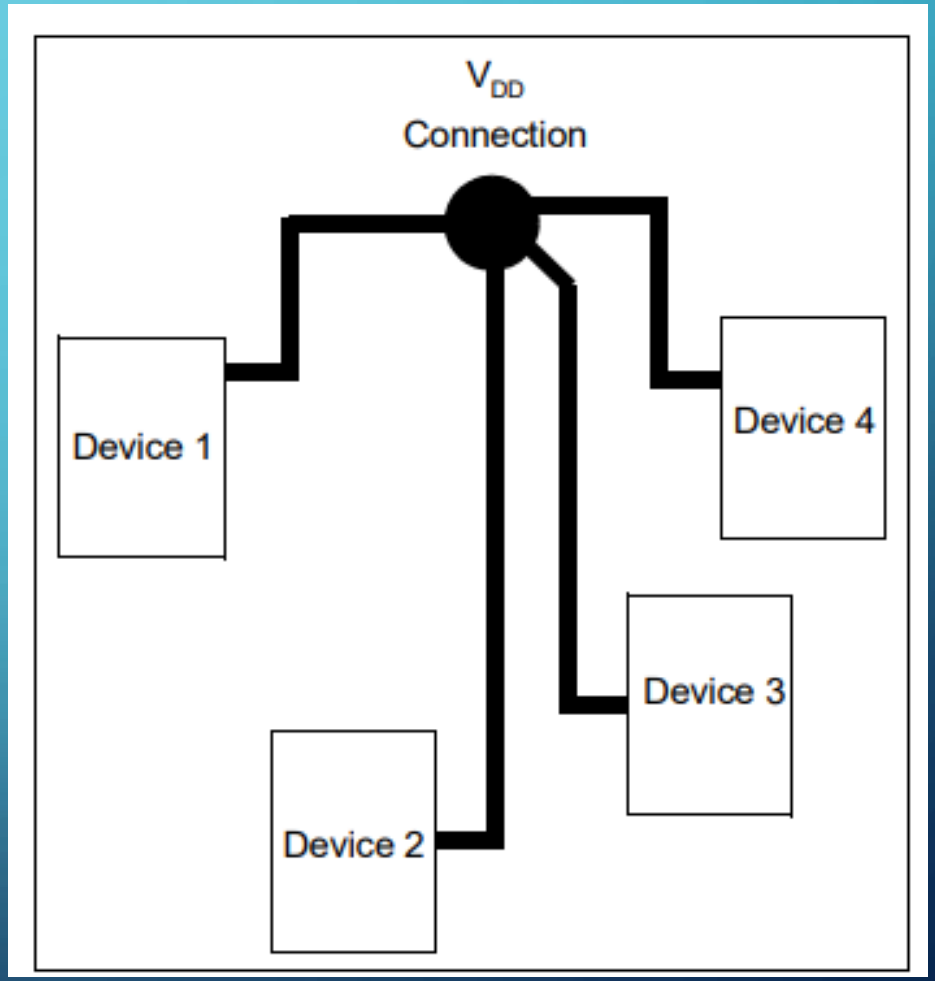
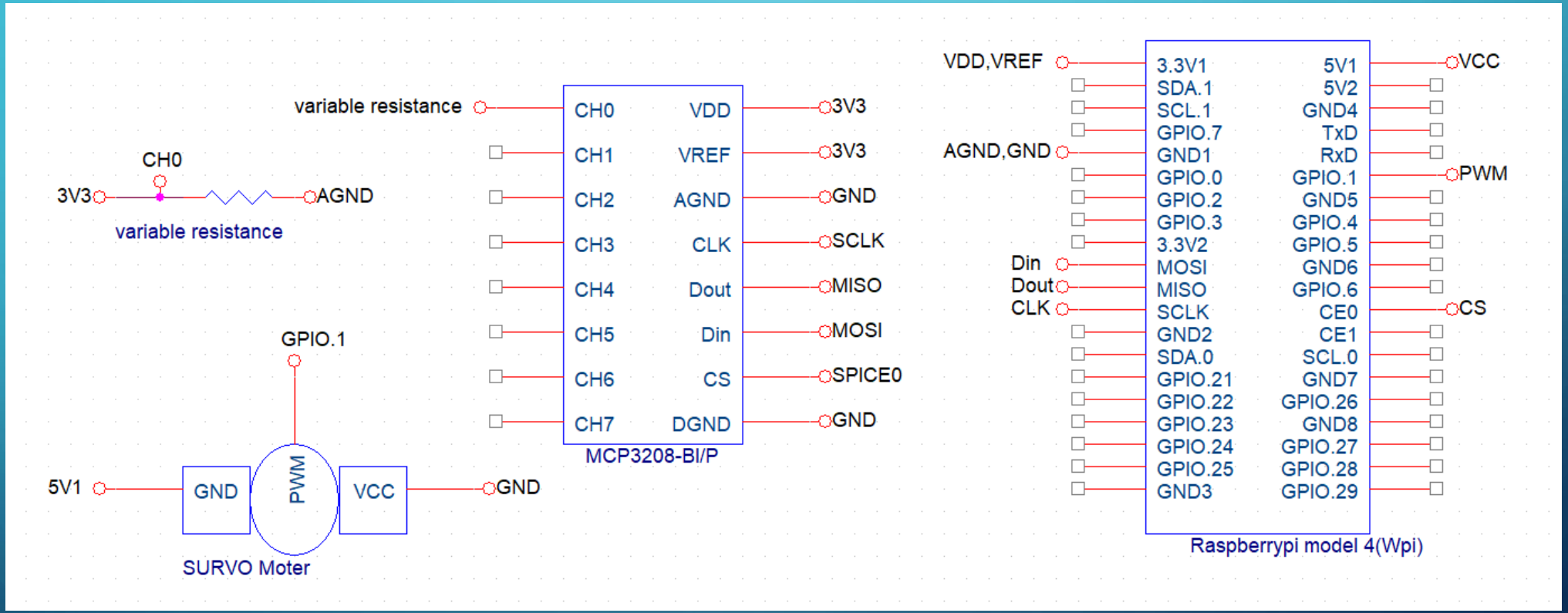


FIGURE 6-5: *Separation of Analog and Digital Ground Pins.*



6) 회로도


variable resistance
CH0
VDD
3V3
CH1
VREF
3V3
CH2
AGND
GND
CH3
CLK
SCLK
CH4
Dout
MISO
CH5
Din
MOSI
CH6
CS
SPICE0
CH7
DGND
GND
MCP3208-BI/P
CH0
3V3
AGND
variable resistance
GPIO.1
5V1
GND
PWM
VCC
GND
SURVO Moter
VDD,VREF
AGND,GND
Din
Dout
CLK
3.3V1
SDA.1
SCL.1
GPIO.7
GND1
GPIO.0
GPIO.2
GPIO.3
3.3V2
MOSI
MISO
SCLK
GND2
SDA.0
GPIO.21
GPIO.22
GPIO.23
GPIO.24
GPIO.25
GND3
5V1
5V2
GND4
TxD
RxD
GPIO.1
GND5
GPIO.4
GPIO.5
GND6
GPIO.6
CE0
CE1
SCL.0
GND7
GPIO.26
GND8
GPIO.27
GPIO.28
GPIO.29
VCC
PWM
CS
Raspberrypi model 4(Wpi)

7) 구동영상