# PICO 2

# A computer chip with different colored labels AI-generated content may be incorrect.

# MCP4912

The MCP4912 is a Dual 10-Bit Voltage Output DAC (Digital to Analog Converter) that operates from a 2.7 V to a 5.5V supply and are SPI compatible.

A white rectangular object with black text

AI-generated content may be incorrect.

**Fig 1**: MCP4912 pinouts.

**Pin Function Table**:

A computer program with text

AI-generated content may be incorrect.

**Pin Descriptions**:

1. **CS**: Tells the peripheral that data is ready to be sent and allows the user to select the peripheral it want to talk to. The Chip select pin is HIGH by default and becomes LOW when data is ready to be sent thereby activating the peripheral. After data has been sent it returns to HIGH.
2. **LDAC**: Transfer input latch registers to their corresponding DAC registers (VOUT).
   1. When LOW both Vouta and Voutb are updated simultaneously with their input register contents..
   2. When tied to VSS, Vout is updated at the rising edge of the CS pin.
   3. Or can be driven by an external control device (MCU I/O pin)
3. **SHDN:** When LOW, both DAC channels are shut down. No DAC output available during the shutdown.
4. **Vouta, Voutb:** output pins each with its own amplifier. The output amplifier of each can drive the output pin with a range of VSS to VDD.
5. **Vrefa, Vrefb:** Determines the reference voltage that will be used by the DAC to determine the output voltage. If the input value is 50%, the output voltage will be 50% of Vref.

**GENERAL OVERVIEW**

**Note**: The MCP4912 takes values strictly in binary form.

A white paper with black text and symbols

AI-generated content may be incorrect.

**Fig 2**: Equation for analog output voltage.

**Output range**:

MCP4912 takes an input of 10 bit. Considering a constant gain of 1 for simplicity.

**n=10, G=1, Vref = 3.3v**

**Range** = volts

**Serial Interface:**

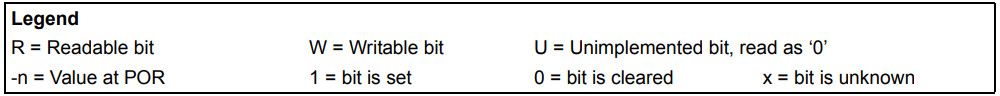
Data is sent to the device via the SDI pin and is clocked in on the rising edge of the SCK (unidirectional communication). CS is LOW during the duration of a write operation. The write command is initiated by driving the CS pin low, followed by clocking the 4 configuration bits and the 12 data bits into the SDI pin on the rising edge of SCK. CS the goes HIGH causing data to be latched into the selected DAC’s input registers.

All writes are 16-bit words and anything past 16 bits is ignored.

* 4 MSBs 🡪 Configuration bits
* 12 remaining 🡪 Data bits.

A white rectangular box with black text

AI-generated content may be incorrect.



A white background with black text

AI-generated content may be incorrect.

**Note**:

MOSI (TX)🡪 Master Out Slave Input: transmits data from the master to the slave.

MISO (RX)🡪 Master In Slave Output: transmits data from the slave to the master.

# 23K256 SRAM.

A diagram of a circuit board

AI-generated content may be incorrect.A table with text on it

AI-generated content may be incorrect.The 23K256 chip is a memory chip with 256 Kbits accessed via SPI. Requires a SCK (clock) input, data in (SI) and data out (SO). Chip select determines the access mode.

**Operation:**

* Chip contains an 8-bit instruction register accessed via SI.
* SI must be LOW and HOLD must be HIGH while accessing this register
* Instructions are transferred MSB first then LSB last.

**A close-up of a list

AI-generated content may be incorrect.**

* Addresses are 16 bits

Read sequence

* CS to LOW
* Send 8-bit instruction
* Send 16-bit address (8-bits at a time)
* Read outputs from SO pin

Write sequence:

* CS to LOW
* Send write instruction
* Send 16-bit address
* Write data
* Bring CS to HIGH

Number of addresses =

Address bits = 16 bit in total where the MSB is ignored since only 15 bits are needed to address all 32 000 spots.

# MCP 23008 (I2C communication)

I2C 🡪 Integrated Circuit communication

It only uses 2 pins:

* SDA: Serial Data (connected to pull up resistor)
* SCL: Serial Clock (connected to pull up resistor)

How I2C works:

* I2c is a master slave system where 1or more masters can have up to 128 slaves for the 7-bit address version.
* ***IDLE state***: this is when both SDA and SCL are low and no communication is happening.
* ***Start condition***: a master device first pulls SDA low then SCL low too.
* ***Claim bus***: when the node has pulled down SDA and SCL. In this state no other node is allowed to use the interface.
* Master device starts the clock.
* The master then sends the address of the node that it wants to talk to. Each slave has a unique 7-bit address.
* After the address, the master sends the write/read bit. (0: write, 1: read)
* The user then has to read the acknowledge bit to determine if the slave is ready to read/write. (ACK: 0, NACK: 0)
* Next the data byte is sent, this could be done sequentially where 1 byte is first sent, then an ACK bit then another byte.
* To end communication, the clock goes back high and stays high and as SDA transitions the transition happens when SCL is high which indicates a stop condition.

A row of blue and orange rectangular shapes

AI-generated content may be incorrect.

**Note**: SDA only transitions when the clock is low since the opposite indicates the start/stop condition.

A diagram of a diagram

AI-generated content may be incorrect.

MCP23008 pinout:

sA diagram of a circuit board

AI-generated content may be incorrect.

MCP23008 Functionality:

* There are 8 I/O pins that can be accessed via I2C or SPI communication.
* Has 3 hardware address pins A0-A2 allowing multiple devices on the same bus. (8)
* It has multiple 8 bit registers to determine:
  + Whether each I/O pin is an input or output
  + The output data values
  + The polarity (whether HIGH=1 or 0 for inputs)
* An external system eg microcontroller can write to registers to configure and control I/Os
* Input and output data is stored in separate registers
* Input polarity can be inverted with a special register
* The chip ***resets all its internal registers and initializes its internal logic*** when powered.

Register Addresses:

A table with numbers and letters

AI-generated content may be incorrect.

**MCP23008 Operations**:

1. Sequential Operation:

The Sequential Operation bit (SEQOP) in the IOCON register controls the operation of the address pointer. When enabled (default) the address pointer updates/increments automatically after each data transfer.

***Sequential mode***: IOCON.SEQOP = 0 address pointer automatically increments to the next address after each byte is clocked.

***Byte Mode***: IOCON.SEQOP = 1 address pointer stays at that address which allows the user to continually read from that address for polling purposes.

1. Write Operation:

Includes the control byte, register address, 8-bits of data and the ACK bit from the chip. The operation is ended with a STOP or RESTART condition generated by the host.

A diagram of a computer program

AI-generated content may be incorrect.

P: stop

S: start

1. Read Operation:

Includes the control byte, another control byte that contains the start condition and ACK with R/W=1. The chip then transmits data contained in the address register. Operation ends with the STOP or RESTART condition.

A screenshot of a computer

AI-generated content may be incorrect.

1. Sequential R/W operation:

Instead of transmitting the STOP/RESTART condition after data transfer, the host clocks the next byte. Operation ends the STOP/RESTART condition is sent.

**Note**: the address rolls over to zero when the last address is reached.

**Addressing**:

A screenshot of a computer

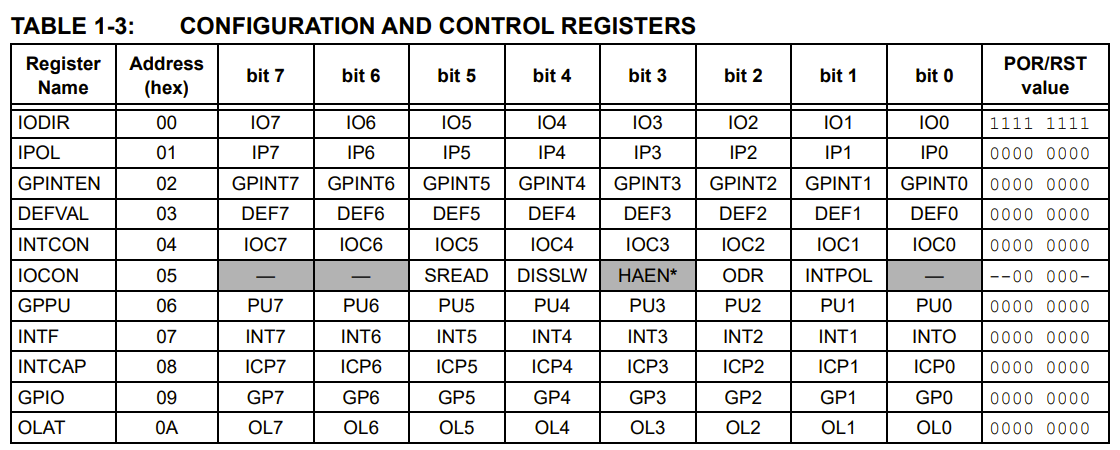
AI-generated content may be incorrect.Allows 7-bit client addressing, and the ACK bit which is the 8-bit of the control byte. There are 4 fixed bits and 3 address bits (A0-A3).

A diagram of a computer program

AI-generated content may be incorrect.

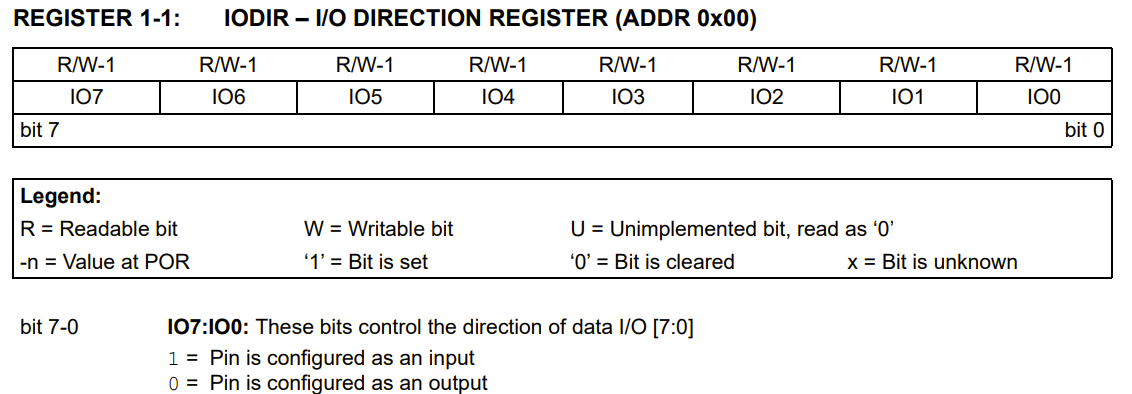
**GPIO port:**

* Reading the GPIO register: Realtime value of the physical pins.
* Reading the OLAT register: gives the last value write to the output latch
* Writing to the GPIO register: Same as writing to the output latch.
* Writing to the OLAT register: sets the output level the pin should drive.

****

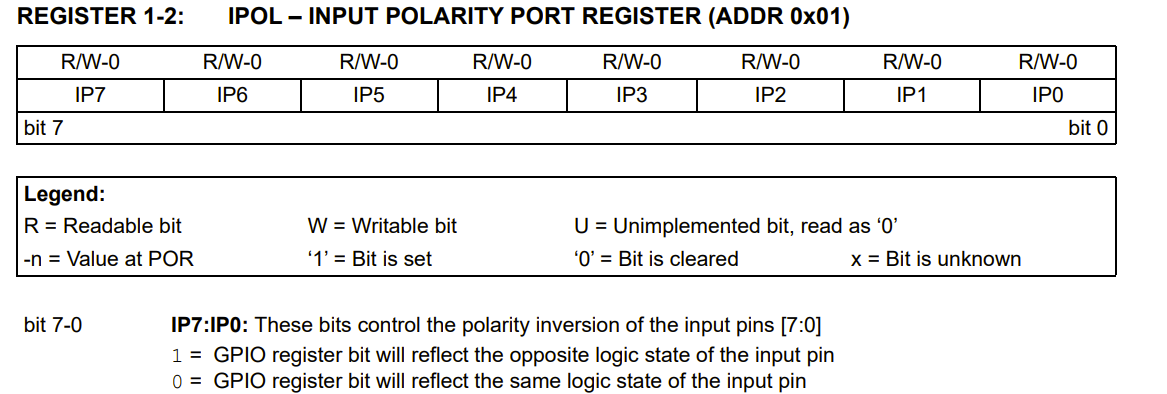
**Registers:**

1. **I/O direction register (IODIR):**

****

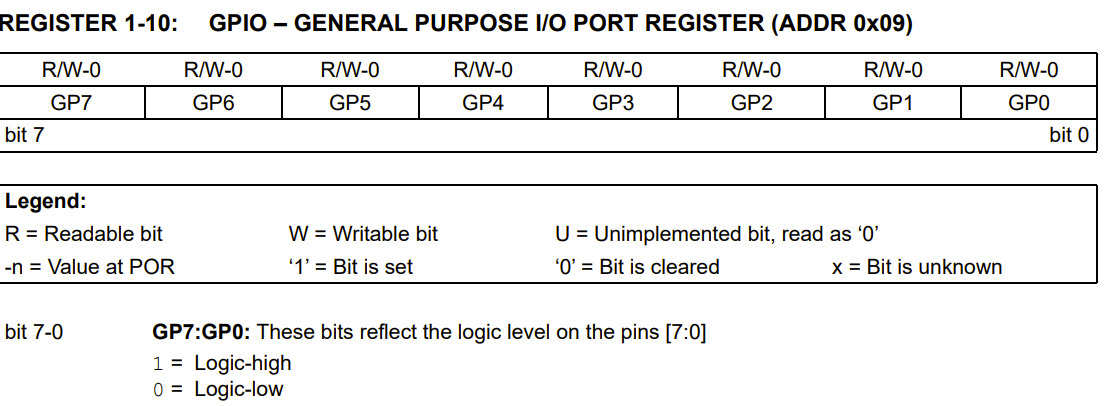
1. **Input Polarity Register (IPOL)**

Determines the polarity of the input pins.



1. **Port Register (GPIO)**

Reading from the GPIO gives the value of the port, and writing to this register modifies the output latch. (OLAT)



1. **Output Latch Register (OLAT)**

Reading from this returns the value of the output latch not the port itself. A write to this register modifies the Output latch and modifies the output pins.

