Facultad de Ingeniería

Universidad Nacional Autónoma de México

Ingeniería Mecatrónica

Robótica

Registros Dynamixel

Tobón Sosa Edgar

Ciudad de México

Fecha 24/05/2022

1.Introducción

Este documento se creó con el objetivo de conocer cuales son los registros principales de un Dynamixel Serie AX-12W y así el usuario tenga el conocimiento de que registros son los necesarios para poder manipular el AX-12W, así como conocer cuales son las capacidades y limitantes de los mismos; permitiéndole determinar si el modelo AX-12W es el adecuado para su proyecto.

2.Registro Principales

2.1.Angle Limit

Registros de la EEPROM con los cuales se modifica el ángulo mínimo y máximo.

CW Angle Limit: El mínimo valor (valor por defecto 0°)

CCW Angle Limit: El máximo valor (valor por defecto 300°).

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
6	2	CW Angle Limit	Clockwise Angle Limit	RW	0
8	2	CCW Angle Limit	Counter ClockWise	RW	1023

CW/CC Angle Limit(6,8)

CW Angle Limit: El mínimo valor (0°)

CCW Angle Limit: El máximo valor (300°).

2.2.Temperatura

Registros de la EEPROM con los cuales se modifica el ángulo mínimo y máximo.

CW Angle Limit: El mínimo valor (valor por defecto 0°)

CCW Angle Limit: El máximo valor (valor por defecto 300°).

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
11	1	Temperature Limit	Maximum Temperature Limit	RW	70

(RAM)

Address	Size(byte)	Data Name	Description	Access	Initial value
43	1	Present Temperature	Present Temperature	R	-

2.3.Voltaje

Cada unidad equivale a 0.1 [V] (60 = 6[V]).

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
12	1	Min Voltage Limit	Minimum Voltage Input	RW	60
13	1	Max Voltage Limit	Maximum Voltage Input	RW	140

(RAM)

Address	Size(byte)	Data Name	Description	Access	Initial value
42	1	Present Voltage	Present Voltage	R	-

2.4.Torque

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
14	2	Max Torque	Maximum Torque	RW	1023

Data 1,023 (0x3FF) means that DYNAMIXEL will use 100% of the maximum torque it can produce while Data 512 (0x200) means that DYNAMIXEL will use 50% of the maximum torque.

(RAM)

Address	Size(byte)	Data Name	Description	Access	Initial value
24	1	Torque Enable	Motor Torque ON/OFF	RW	0
34	2	Torque Limit	Torque Limit	RW	Max Torque

(Registro: 24)

0: Torque OFF.

1: Torque ON.

(Registro: 34)

It is the value of the maximum torque limit.
0 ~ 1,023(0x3FF) is available, and the unit is about 0.1%.
For example, if the value is 512, it is about 50%; that means only 50% of the maximum torque will be used.

2.5.Posición Actual

(RAM)

Address	Size(byte)	Data Name	Description	Access	Initial value
36	2	Present Position	Present Position	R	-

It is the present position value of DYNAMIXEL. The range of the value is 0~1023 (0x3FF), and the unit is 0.29 [°].

2.6.Velocidad Actual

(RAM)

Address	Size(byte)	Data Name	Description	Access	Initial value
40	2	Present Speed	Present Speed	R	-

It is the present moving speed. 0~2,047 (0x7FF) can be used. If a value is in the range of 0~1,023, it means that the motor rotates to the CCW direction. If a value is in the range of 1,024~2,047, it means that the motor rotates to the CW direction. That is, the 10th bit becomes the direction bit to control the direction, and 0 and 1,024 are equal. The unit of this value varies depending on operation mode.

- **Joint Mode**
  The unit is about 0.111rpm. For example, if it is set to 300, it means that the motor is moving to the CCW direction at a rate of about 33.3rpm.
- **Wheel Mode**
  The unit is about 0.1%. For example, if it is set to 512, it means that the torque is controlled by 50% of the maximum torque to the CCW direction.

2.7.Movimiento

(RAM)

Address	Size(byte)	Data Name	Description	Access	Initial value
46	1	Moving	Movement Status	R	0

Value	Description
0	Goal position command execution is completed
1	Goal position command execution is in progress

3.Registro Secundarios

3.1.Model Number

Almacena el número de modelo que es nuestro Dynamixel.

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
0	2	Model Number	Model Number	R	300

3.2.Firmware Version

Almacena la versión de firmware de nuestro dynamixel.

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
2	1	Firmware Version	Firmware Version	R	-

3.3.ID

Contiene el ID de cada uno de nuestros dynamixel con el objetivo de identificarlos cuando hay más de un conectado.

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
3	1	ID	Dynamixel ID	RW	1

3.4.Baud Rate

Determina la comunicación Serial entre el controlador y el Dynamixel.

(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
4	1	Baud Rate	Communication Speed	RW	1

Value	Baud Rate	Margin of Error
1(Default)	1M	0.000%
3	500,000	0.000%
4	400,000	0.000%
7	250,000	0.000%
9	200,000	0.000%
16	115200	-2.124%
34	57600	0.794%
103	19200	-0.160%
207	9600	-0.160%

3.5.Return Delay Time

El tiempo en que Dynamixel notifica el estado del paquete(DATA) que recibió.

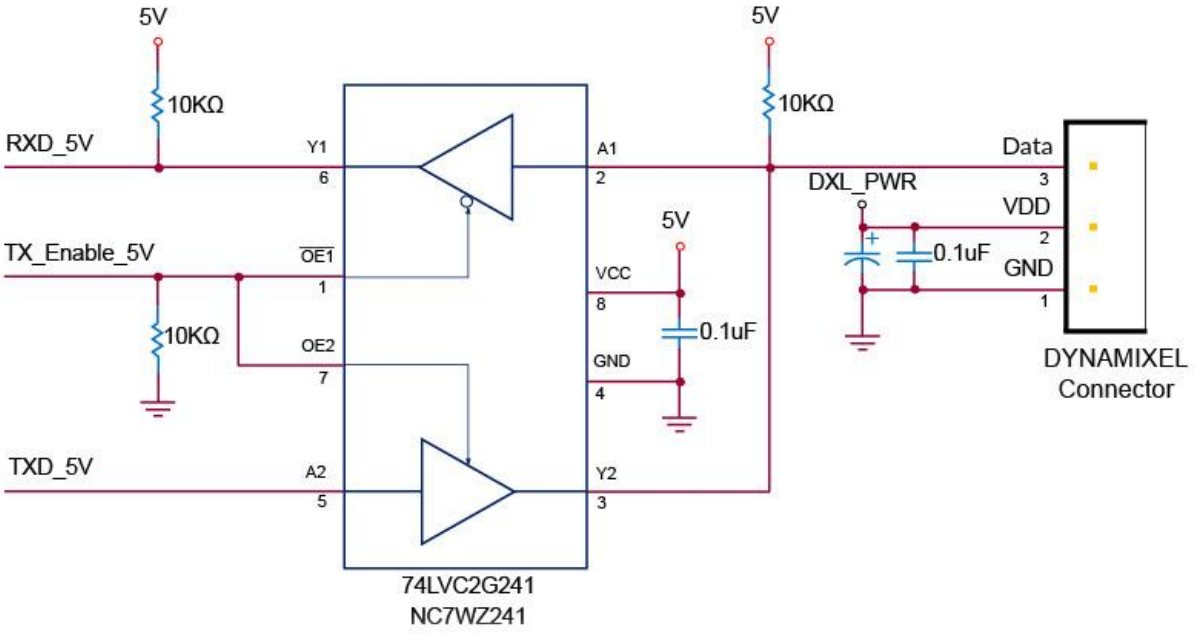
(EEPROM)

Address	Size(byte)	Data Name	Description	Access	Initial value
5	1	Return Delay Time	Response Delay Time	RW	250

4.Circuito de Comunicación

To control the DYNAMIXEL actuators, the main controller needs to convert its UART signals to the half duplex type. Se recomienda diagrama del circuito mostrado abajo.

4.1.Circuito de comunicación TTL



5.Dynamixel Protocolo 1.0

Para que nuestro controlador Dynamixel pueda establecer una comunicación de datos binario con la computadora es necesario un protocolo para este modelo de Dynamixel solo es compatible el protocolo 1.0.

5.1.Instruction Packet

El paquete de instrucciones es el comando de información enviado al dispositivo.

Header1	Header2	Packet ID	Length	Instruction	Param 1	…	Param N	Checksum
0xFF	0xFF	Packet ID	Length	Instruction	Param 1	…	Param N	CHKSUM

5.1.1.Header

Campo que indica donde inicia el paquete.

5.1.2.Packet ID

Indica el ID del dispositivo que debería recibir el ‘Instruction Packet’.

5.1.3.Length

Indica el tamaño en Byte de la instrucción.

5.2.Instruction

Este campo define las instrucciones que podemos hacer al dispositivo Dynamixel.

Value	Instructions	Description
0x01	Ping	Instruction that checks whether the Packet has arrived to a device with the same ID as Packet ID
0x02	Read	Instruction to read data from the Device
0x03	Write	Instruction to write data on the Device
0x04	Reg Write	Instruction that registers the Instruction Packet to a standby status; Packet is later executed through the Action instruction
0x05	Action	Instruction that executes the Packet that was registered beforehand using Reg Write
0x06	Factory Reset	Instruction that resets the Control Table to its initial factory default settings
0x08	Reboot	Instruction that reboots DYNAMIXEL (See supported products in the description)
0x83	Sync Write	For multiple devices, Instruction to write data on the same Address with the same length at once
0x92	Bulk Read	For multiple devices, Instruction to write data on different Addresses with different lengths at once (See supported products in the description)

6.Controlador OpenCM 9.04

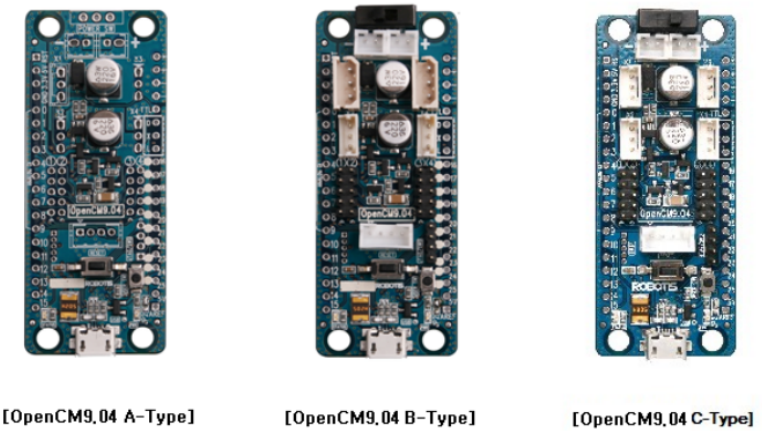


Figura 6.1.Controlador OpenCM 9.04

El primer método para mover los motores de Dynamixel será a través de la placa OpenCM 9.04(Figura 6.1) y también con la placa de expansión(Figura 6.2) en cual conectaremos los Dynamixel.

Figura 6.2.Placa de expansión para el controlador OPENCM 9.04

Utilizaremos el software del IDE de Arduino para programar nuestro controlador OpenCM 9.04 para ello es necesario agregar la placa manualmente al IDE e instalar las bibliotecas de Dynamixel para Arduino. El siguiente link es la documentación completa de como realizar este método, está disponible para sistemas operativos Window, MACOS y Linux: OpenCM 9.04 (robotis.com).

6.1.Model Scan

Si se realizó correctamente las instrucciones de la documentación vamos a poder utilizar los ejemplos del Workbench (Figura 6.3).

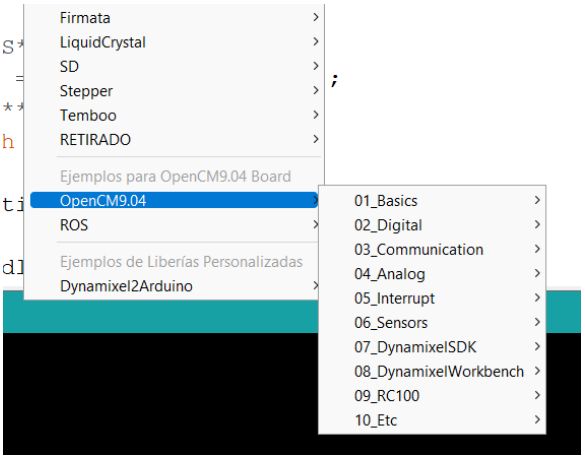


Figura 6.3.Ejemplos Workbench

El primer ejemplo a utilizar es el de Model Scan, este algoritmo nos indica la información mínima necesaria sobre nuestros dynamixel como lo es el modelo y el más importante el ID, Puede omitirse este paso si ya se conoce esta información.

6.2.PING

Con el ejemplo de PING nos permite saber si es que se está detectando correctamente la conexión de los motores DYNAMIXEL y el controlador OpenCM 9.04.

6.3.Position

Con este algoritmo podremos cambiar el registro de posición de nuestros motores Dynamixel al igual que hace un Read a la posición en la cual se encuentra el Dynamixel; además de modificar el registro para pasar nuestro Dynamixel a modo Junta.

6.4.Sync_Write

Con este código podremos mover dos Dynamixel conectados, este movimiento es simultáneo.

7.Practica 1

El objetivo de esta práctica es mover dos dynamixel conectados en serie a través de un algoritmo que permita al usuario, a través de comunicación serial, indicar el Dynamixel que deseamos mover y la posición a la cual lo deseamos mover.

Para llevar a cabo esta práctica nos basaremos en los ejemplos del Workbench vistos anteriormente. También realizaremos un código en el cual al recibir por serial el mensaje “ID:N,Pos:Nxx”, en el cual para podremos indicar el ID del motor Dynamixel que deseamos mover y la posición a la cual deseamos moverlo.

```
//****LECTURA LINEA TXT*****
while(Serial.available()>0){
  delay(20);
  buffer +=(char)Serial.read();

  //******BUFFER TXT ANALISIS**********
  if(buffer.startsWith("ID:",0)&&buffer.startsWith("Pos:",5)){
    Id=String(buffer[3]).toInt();
    //Serial.println(ID);
    while(i<buffer.length()-1){
      substring+=buffer[i];
       i++;
    }
```

Figura 7.1.Fragmento de código para la lectura de mensaje serial

Como en este caso vamos a encargarnos de mover dos Dynamixel vamos a poner las configuraciones mínimas necesarias para poder trabajar con ellos, estas configuraciones son el ID las cuales almacenaremos en un array y los baudios con los que trabajan los motores.

```
#include <DynamixelWorkbench.h>

#if defined(__OPENCM904__)
  #define DEVICE_NAME "3" //Dynamixel on Serial3(USART3)  <-OpenCM 485EXP
#elif defined(__OPENCR__)
  #define DEVICE_NAME ""
#endif

#define BAUDRATE  1000000
#define DXL_ID_1  1
#define DXL_ID_2  3

//********GLOBALES**********
uint8_t dxl_id[2] = {DXL_ID_1, DXL_ID_2};
//*************************
DynamixelWorkbench dxl_wb;
```

Figura 7.2.Configuración Inicial.

Como se observa hay un objeto ‘dxl_wb’ de la clase “DynamixelWorkbench” este objeto es muy importante ya que con él usaremos los métodos de ping,position y otros métodos importantes para el control del Dynamixel.

Como ya pudimos segmentar los datos de ID y POS, los alamcenaremos en dos variables una nombrada Id y otra Pos, con nuestro algoritmos ya solo hay que convertir estas

variables a tipos ‘unit8_t’ ya que es el tipo de variable requerido para los métodos del objeto antes mencionado.

```
 //**** ID  Y POS SEGMENTADO**********
  if(substring.length()==((buffer.length()-1)-9)){
        Serial.println(" ");
        Serial.println("INSTRUCCIÓN PARA: ");
        Serial.print("Dynamixel ID: ");
        Serial.print((uint8_t)Id);
        delay(3000);

//*****JOINT MODE CONFIGURACION********
        result = dxl_wb.jointMode((uint8_t)Id, 0, 0, &log);
        if (result == false)
         {
           Serial.println(log);
           Serial.println("Error al cambiar a modo Junta");
         }
       else
       {
           Serial.println("Correctamente cambiado a modo Junta");


       }
          ,
//*** INSTRUCCION MOVER DYNAMIXEL*****
         dxl_wb.goalPosition((uint8_t)Id, (int32_t)Pos);
          Serial.println("**GIRO COMPLETADO**");
          Serial.print("Dynamixel ID: ");
          Serial.print((uint8_t)Id);
          Serial.print("         ");
          Serial.print("Posicion Actual: ");
          Serial.println((uint32_t)Pos);
          delay(3000);
   }
```