**When we use SetPosition twice in a row, the second SetPosition is ignored.**

#pragma config(Sensor, S1, M\_CONTROL, sensorI2CCustom)

#include "saas\_common.h"

//saas\_common is a brief version of all of Xanders array management tools and write i2c tools

void SetPosition(long Input)

{

//initializes the arrays

tByteArray I2Crequest;

//sets the number of bytes to send:

I2Crequest[0] = 8;

//sends the adress as the first byte

I2Crequest[1] = 0x02;

//sets the starting position to start sending data at

I2Crequest[2] = 0x40;

I2Crequest[3] = (byte)((Input >> 24) & 0x000000ff);

I2Crequest[4] = (byte)((Input >> 16) & 0x000000ff);

I2Crequest[5] = (byte)((Input >> 8) & 0x000000ff);

I2Crequest[6] = (byte)(Input & 0x000000ff);

//sets the motor mode

I2Crequest[7] = 0b00000010;

//sets the motor power

I2Crequest[8] = 0b00010000;

writeI2C(S1, I2Crequest);

}

Task main()

{

SetPosition(0);

SetPosition(1440); //ignores this instruction, only goes to encoder position 0  
}

**But if there is a wait greater than or equal to 9 milliseconds, it will attempt to go to for the first position as long as it waits then go to the second.**

Task main()

{

SetPosition(0);

Sleep(9);

SetPosition(1440); //ignores this instruction, only goes to encoder position 0  
}

**When both SetPosition and getEncoderValue are used in the same loop, the motor moves accurately to the position but does not display the getEncoderValue return value.**

#pragma config(Sensor, S1, M\_CONTROL, sensorI2CCustom)

#include "saas\_common.h"

//saas\_common is a brief version of all of Xanders array management tools and write i2c tools

void SetPosition(long Input)

{

//initializes the arrays

tByteArray I2Crequest;

//sets the number of bytes to send:

I2Crequest[0] = 8;

//sends the adress as the first byte

I2Crequest[1] = 0x02;

//sets the starting position to start sending data at

I2Crequest[2] = 0x40;

I2Crequest[3] = (byte)((Input >> 24) & 0x000000ff);

I2Crequest[4] = (byte)((Input >> 16) & 0x000000ff);

I2Crequest[5] = (byte)((Input >> 8) & 0x000000ff);

I2Crequest[6] = (byte)(Input & 0x000000ff);

//sets the motor mode

I2Crequest[7] = 0b00000010;

//sets the motor power

I2Crequest[8] = 0b00010000;

writeI2C(S1, I2Crequest);

}

long getEncoderValue()

{

//initializes the arrays

tByteArray I2Crequest;

tByteArray I2Cresponse;

//sets the number of bytes to send:

I2Crequest[0] = 2;

//sends the adress as the first byte

I2Crequest[1] = 0x02;

//sets the starting position to start sending data at

I2Crequest[2] = 0x4C;

//writes the data, and gets the response

writeI2C(S1, I2Crequest, I2Cresponse, 4);

//creates a long out of the bytes

long EncoderValue = (I2Cresponse[0] << 24) + (I2Cresponse[1] << 16) + (I2Cresponse[2] << 8) + (I2Cresponse[3] << 0);

//returns the long

return EncoderValue;

}

Task main()

{

While(true)

{

SetPosition();

eraseDisplay();

nxtDisplayString(0, “%i”, getEncoderPosition);

}

}

**When getEncoderValue is used in a loop with any sleep command, getEncoderValue displays the last value successfully retrieved but does not update the value.**

Task main()

{  
 while(true)

{

getEncoderPosition(); //displays the same value, even when the motor is moves.

//this value appears to be the last value gotten by getEncoderPosition without the sleep command in the same loop

sleep(1);

}

}