How to communicate with the Maskless Photolithography Stepper Stage Controller with the I²C Bus

A. Introduction

The I²C bus is a synchronous, multi-host/multi-peripheral, packet switched serial communication standard invented by Phillips in 1982. The standard has seen increased use in the 21st century in the embedded market for its simple two-wire interface and high communication speed compared to traditional asynchronous serial standards.

The ATmega 2560 microcontroller used in the maskless photolithography stepper stage controller supports the I²C protocol. A specific packet structure is defined for controlling the stage controller to achieve various functions like precision stepping, calibration and reset. This article will introduce the packet structure of the stepper controller and how to interface with it using a device supporting the I²C standard like the Raspberry Pi platform.

B. Theory of Operation

1. Serial Vs. Parallel communication

Serial communications, like the name suggests, sends data packets one bit at a time in a series. The data packets usually have a fixed size of 8 bits (1 byte), and it takes 8 bus cycles to transmit one packet. Parallel communication transmits all the bits in one packet at once. If there are 8 bits in one packet, the physical data path is usually 8-bits wide. Theoretically, parallel communication is much faster than serial communication at the same bus speed due to the increased data path width. All the major bandwidth-dependent bus architectures (PCI-E, Backside Bus, internal data paths of the processor) are parallel communication standard. However, the increase wire count of the parallel structure increases demands strong electromagnetic interference shielding to reduce error and large amount of PC board real estate which may not be present on an embedded device.

2. I²C bus basics

The I²C bus utilizes a controller, known as the master or host, to communicate with multiple connected peripheral or “slave” devices. A peripheral device may not transmit data unless it has been addressed by the host. Each device on the I²C bus has a specific device address attached to differentiate between other devices connected to the same bus. To support many peripheral devices, the peripheral’s address must be configured upon startup to avoid bus contention.

The physical I²C bus interface consists of the serial clock (SCL) and serial data (SDA) lines. Both SDA and SCL lines must be pulled up externally via a pull-up resistor connected to Vcc. The size of the pull-up resistors are determined by the amount of parasitic capacitance on the I²C bus itself, and usually ranges from 1k Ohms to 100k Ohms. Data transfer may be initiated only when the bus is idle. The bus is considered idle if both SDA and SCL lines are high (at Vcc) after a STOP condition has been received.

The General procedure for a bus host to access a peripheral device is the following:

1. Suppose a host wants to send data to a peripheral device:
   - Host-transmitter sends a START condition and addresses the peripheral receiver
   - Host-transmitter sends data to the addressed peripheral receiver
   - Host-transmitter terminates the transfer with a STOP condition

2. If a host wants to receive/read data from a peripheral device:
   - Host-receiver sends a START condition and addresses the peripheral transmitter
   - Host-receiver sends the requested register to read to peripheral transmitter
   - Host-receiver receives data from the peripheral transmitter
   - Host-receiver terminates the transfer with a STOP condition

The START and STOP conditions are defined as follows: A high-to-low transition on the SDA line while SCL is high defines a START condition. A low-to-high transition on the SDA line while the SCL line is high defines a STOP condition. An example is given in Figure 1. These signals are integrated into the I^2C hardware layer and are transparent to the user.

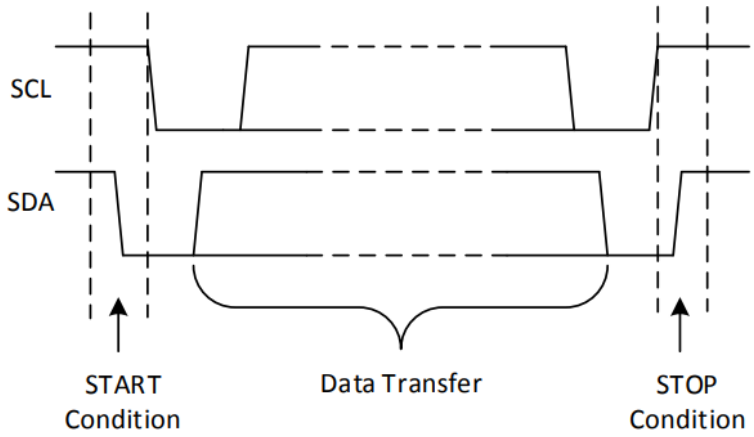

Figure 1: Example of START and STOP condition

3. Datagram structure

A standard datagram for the stage controller consists of 8 bytes for a total of 64 bits of data. The structure is given in Table 1.

64-bit Datagram Structure							
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
MAGIC	COMMAND	DATA				STATUS	CRC

Table 1: I^2C 64-bit datagram structure

The most significant (first) byte in the datagram is the magic number. This is a specially defined number (currently 0x5A) to signal the start of the datagram. The second byte is the command byte. This byte defines what register (or function) the following data packet is intended for. The third to sixth byte

in the datagram is the 32-bit frame intended for transmitted data. The seventh byte is the status packet. This byte will be sent back to the host from the stage controller on every transaction to update the host on the controller's internal status. The last byte is the cyclic redundancy check (CRC), a checksum to tell if any bit error occurred during transmission of the datagram.

CommandValue	Name	Purpose
0x00	HALT	Halt the stage controller
0x01	getWidth	acquire maximum x-coordinate of the stage
0x02	getHeight	acquire maximum y-coordinate of the stage
0x03	getX	acquire current x-coordinate of the stage
0x04	getY	acquire current y-coordinate of the stage
0x05	setX	set target x-coordinate of the stage
0x06	setY	set target y-coordinate of the stage
0x07	calib	enter or exit calibration mode

Table 2: list of available commands

Status Value	Purpose
0b00	stage NOT READY
0b01	stage MOVING
0b10	stage IN POSITION

Table 3: list of STATUS packet values

C. List of equipment required

Our standard photolithography stepper stage and its controllers and an I^2C compatible host computer are required to complete the bus. Our default host computer is a Raspberry Pi 4b single board computer, although a different model would work if they had hardware i^2C built in.

D. Instructions

The stage controller is automatically configured to be a peripheral device on the i^2C bus with address 7 on startup. Please follow the following step-by-step instructions to communicate with the stage controller:

a. Perform a HALT instruction to the controller:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x00. The data frame is discarded upon receiving and can be left as 0 or any other number. However, make sure the CRC matches up with the data selected.

b. Acquire maximum X coordinate of the stage:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x01. The data frame is discarded upon receiving.
    2. Transmit a read request to address 7 for an 8-byte package. The controller will reply with the magic number 0x5A, the command 0x01, the maximum X coordinate in unsigned 32-bit integer format, its internal status, and the CRC.
    3. The controller will internally latch the command, so subsequent read request sent to the controller will get the maximum X coordinate until a new command is received by the controller.

c. Acquire maximum Y coordinate of the stage:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x02. The data frame is discarded upon receiving.
    2. Transmit a read request to address 7 for an 8-byte package. The controller will reply with the magic number 0x5A, the command 0x02, the maximum Y coordinate in unsigned 32-bit integer format, its internal status, and the CRC.
    3. The controller will internally latch the command, so subsequent read request sent to the controller will get the maximum Y coordinate until a new command is received by the controller.

d. Acquire current X coordinate of the stage:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x03. The data frame is discarded upon receiving.
    2. Transmit a read request to address 7 for an 8-byte package. The controller will reply with the magic number 0x5A, the command 0x03, the current X coordinate in unsigned 32-bit integer format, its internal status, and the CRC.
    3. The controller will internally latch the command, so subsequent read request sent to the controller will get the current X coordinate until a new command is received by the controller.

e. Acquire Current Y coordinate of the stage:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x04. The data frame is discarded upon receiving.
    2. Transmit a read request to address 7 for an 8-byte package. The controller will reply with the magic number 0x5A, the command 0x04, the current Y coordinate in unsigned 32-bit integer format, its internal status, and the CRC.
    3. The controller will internally latch the command, so subsequent read request sent to the controller will get the current Y coordinate until a new command is received by the controller.

f. Set target X coordinate of the stage:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x05. The data frame needs to contain the target X coordinate in unsigned 32-bit integer format.
    2. The stage will begin moving towards the target X coordinate.
    3. To acquire the stage state and when the stage has reached the target coordinate, the “acquire current Y” or “acquire current X” command can be sent and followed up with read requests. The controller will reply with its current position along with its internal states.

g. Set target Y coordinate of the stage:
    1. Transmit a datagram to address 7 with the magic number 0x5A and the command 0x06. The data frame needs to contain the target Y coordinate in unsigned 32-bit integer format.
    2. The stage will begin moving towards the target Y coordinate.
    3. To acquire the stage state and when the stage has reached the target coordinate, the "acquire current Y" or "acquire current X" command can be sent and followed up with read requests. The controller will reply with its current position along with its internal states.

h. Calibration process:
    Due to the inherent drift properties of the stepper motors, the stage controller's internal plane map may not be representative of its real world location. To remedy this problem, the "calibration" command packet is implemented. Unlike other commands, the "start calibration" command has some additional steps.

    1. Start calibration:
        The stage needs to be stationary and ready to receive new coordinates (status packet 0b10). This information can be obtained by sending the controller the "acquire X" or the "acquire Y" command then followed up with a read request.

        Once the stage is ready, transmit a datagram to address 7 with the magic number 0x5A and the command 0x07. The data frame in this case acts as a begin/end packet. The data frame needs to be 0xFFFFFFFF (all 1s) to start calibration.

    2. Calibration process:
        Following the instruction outlined in section D.e and D.f to send specific target coordinates to the stage. The current internal coordinates of the controller will be latched and the controller will move in the direction it is commanded to.

    3. Finish calibration
        Once the stage is ready, transmit a datagram to address 7 with the magic number 0x5A and the command 0x07. The data frame needs to be 0x00 to indicate a finish calibration condition. At this point, the controller will set the current location as the latched coordinate. Subsequent "Get X" and "Get Y" commands can be sent to verify the calibration.

E. Troubleshooting Procedures

The stage controller's command interpreter comes with a built-in debugging function to automatically output the received packets on the bus to Serial console 0. To enable this feature, make sure to set the public variable "bool debugMode" to true in the instructionhandler.h file.

If no text is available on the serial console after turning on the debug mode and transmitting data to the stage controller, verify using an oscilloscope that the SCK line is demonstrating a 400KHz clock signal and bus activity is present on the SDA line. If not, change the clock configuration on the host controller to 400KHz and retry communication.

If The SDA and SCK lines are constantly low, make sure to include a pull-up resistor on the SDA and SCL line – the I^2C standard requires them.

F. Warning

The stepper controller is certified to comply with FCC class B limits, Part 15 of FCC rules. See instructions available on Interference Resolution | Federal Communications Commission (fcc.gov) if interference to radio reception is suspected.

This product complies with DHHS Rules. 21 CFR, Subchapter J, applicable at date of manufacture.

To prevent electrical shock, do not remove cover to the stepper motor controller module. No user-serviceable parts inside. Refer servicing to qualified service personnel.

Glossary

ATmega 2560: High-performance, low-power 8-bit AVR RISC-based microcontroller.

Datagram: an independent, self-contained message sent over the network or a bus.

FCC: The Federal Communication Commission

Oscilloscope: A type of electronic test instrument that graphically displays varying electrical voltages as a two-dimensional plot of one or more signals as a function of time.

Photolithography: A patterning process in which a photosensitive polymer is selectively exposed to light through a mask, leaving a latent image in the polymer that can then be selectively dissolved to provide patterned access to an underlying substrate.

Synchronous bus: A bus used to interconnect devices that comprise a computer system where the timing of transaction between devices is under the control of a synchronizing clock signal.

Summary:

1. What to change if a Pitt student has done the procedure before but in a different way:

   Omit the operation theory section but keeping the datagram structure. This student should be familiar with how the I2C bus works so no detailed explanation is required.

2. What to change if a Pitt student has done the procedure before but in a different situation:

   Same as part 1. The I2C bus is a standard that can't be changed. The only special part of this instruction is the datagram structure and the I/O steps.

3. What to change if a different product comes into play

   Keep the introduction and operation theory section. The entire instruction section needs to be redone based on the datagram structure of the other product.

Your name: Damien Hu

Audience Profile for <u>How to communicate with the Maskless Photolithography Stepper Stage Controller with the I2C Bus</u>

Question to ask yourself	**Primary readers**	**Secondary readers**
What do they want/need to accomplish?	Learn how to communicate with the stage controller	Learn how to communicate with the stage controller
What do they already know about the topic?	What the product is and how it works	How the I2C bus works and how the stage controller works
What do they need to know about the topic?	The exact working of the device, including the bus architecture and API.	The datagram structure
How will they use this document?	During the Sr. Design Expo	In their office, sitting in front of a computer
Under what conditions?	In a room full of people or in front of their computer	Potentially sleep deprived and needing more coffee.
How do they feel about the topic you are writing about?	Interested	Passionate
How do they feel about you, the writer and the organization you represent?	Interesting group of people doing magic tech stuff	Team member
Could anything hinder their understanding of what you are trying to say?	Understanding of the computer engineering topic	Nothing
Are they from a particular demographic? That is, are they of a certain age? a certain gender? Do they hold certain political viewpoints? Live in a certain area or under certain conditions?	College students, most likely in the engineering school	College students
Are there any cultural differences affect their expectations and interpretations? If so, how?	Not applicable	Not applicable.

SUMMARY of audience analysis:

This instruction is targeted towards Pitt students looking to interface with the stepper stage controller, either as a host computer or add-on accessory. The secondary reader is one of our team members, who will take care of the host computer algorithm and needs to communicate with the stepper stage controller.