snickerdoodle

SPI/I^{2}C PROTOCOL

AUGUST 23, 2016

How to Read this Document

This document makes extensive use of links, references and notices in the page margins to detail additional information that can be useful while following the guide.

WARNING A warning notice indicates a potential hazard. If care is not taken to adhere to the safety precautions, damage may be done to snickerdoodle.

Warnings and cautions will be clearly visible in either the body of the text or in the margin and must be paid close attention while following the guided steps.

CAUTION A caution indication denotes a process that requires special attention. If the caution is not exercised and the process not adhered to, failure may result and/or potential damage to snickerdoodle.

Warning, caution and informational notices, such as this one, may also be found in the margin.

Keywords

Keywords and important terms are shown in *italicized* type. Additional important information can be found in the margins of text with superscript notation[1].

Navigation of menus and directories are shown using ***bold italicized*** type. Any hierarchical navigation is shown using an arrow to denote a ***Parent*** → ***child*** relationship.

`Teletype` text is used to highlight inputs, variables and system files within the host environment.

[1] Margin notes, such as this one, reference the body content and highlight technical details or references for further information.

SPI Configuration

Data Size	8 bit
Clock Polarity	Low
Clock Phase	Rising Edge
Byte Order	Most Significant Bit First

Table 1: SPI Peripheral Configuration Parameters

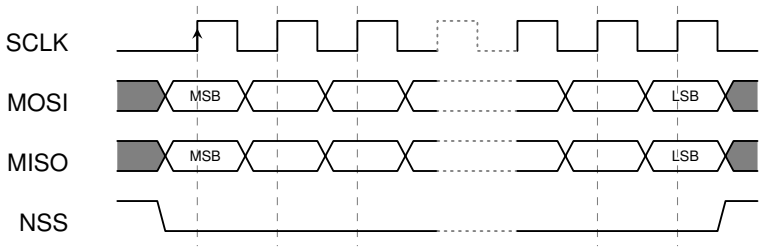

Figure 1: SPI Transmission Timing (Phase, Polarity and Byte Order)

In the following commands and messages, the least significant bit (LSB) is denoted with a 0 index and the most significant bit (MSB) of a single byte transmission is denoted with an index of 7.

I²C Bridge

SPI Commands

Channel Selection Command (0x80)

All transactions begin with the channel selection command. The command requires the three LSBs contain the channel selection, representing one of the eight I²C switch channels, exclusively. Figure 2 shows organization of the 3-bit channel selection within the *channel selection* command.

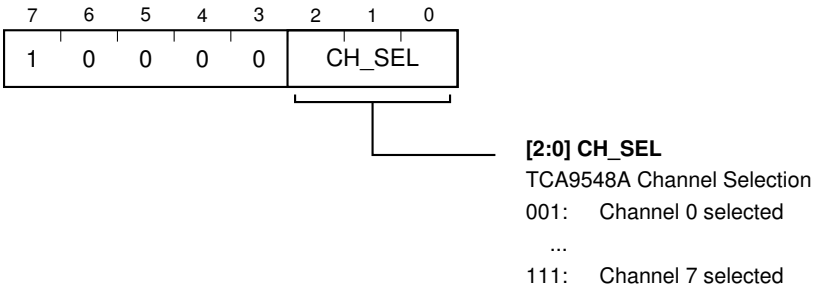

Figure 2: TCA9548A Channel Selection During Begin Read/Write Transaction

Read/Write Command (0xA0)

After initiating the transaction and setting the channel, the device settings and address can be specified for communication. Figure 3 shows the organization of the device settings in the two bytes following the channel selection. The *read/write* command includes specifications for the transaction type, frequency selection, address length and either a 7 or 10 bit device address.

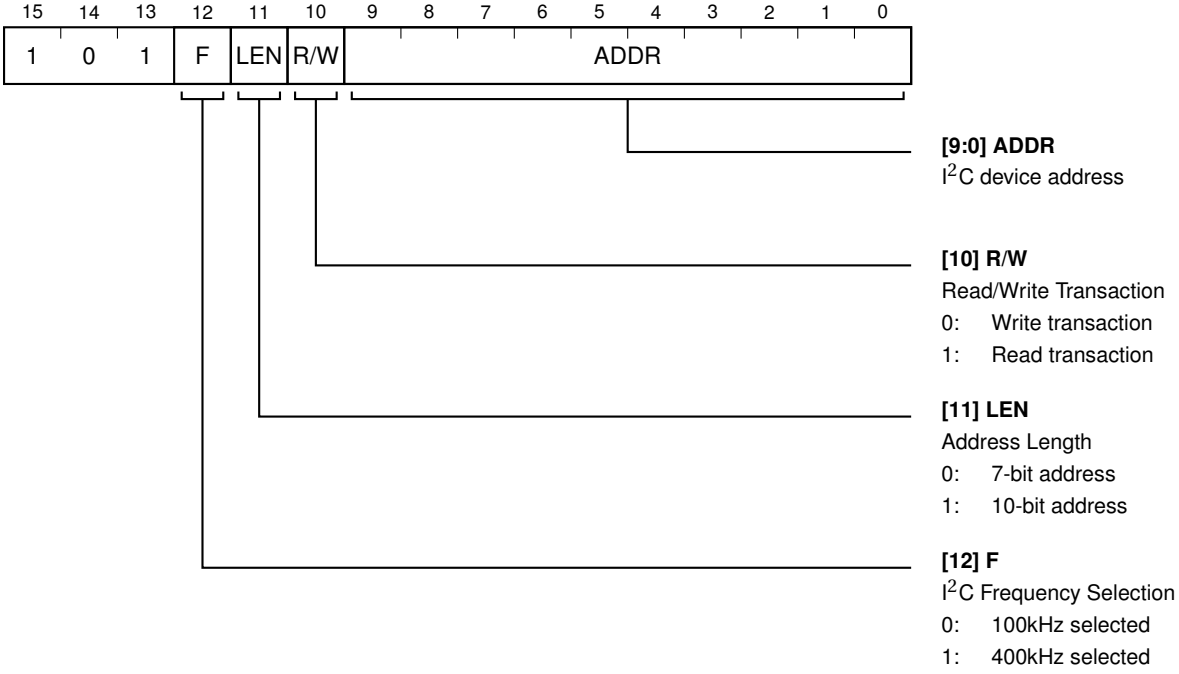

Figure 3: Device Frequency and Address During Begin Read/Write Transaction

> *After specifying a read or write operation, the next byte following the read/write command is expected to represent the length of bytes to transmit or receive. This limits the number of bytes for any read/write command to 255 bytes*

Get Status Command (0xCB)

Following any read or write commands, the status of the operation should be checked with the *get status* command. The response to such a command will be formatted as shown in Figure 4 . The *get status* command does not include any bitwise parameters.

Finish Read Command (0x93)

To finish reading from a device, the finish read command should be used. This command should be used after checking the status of a previous read com-

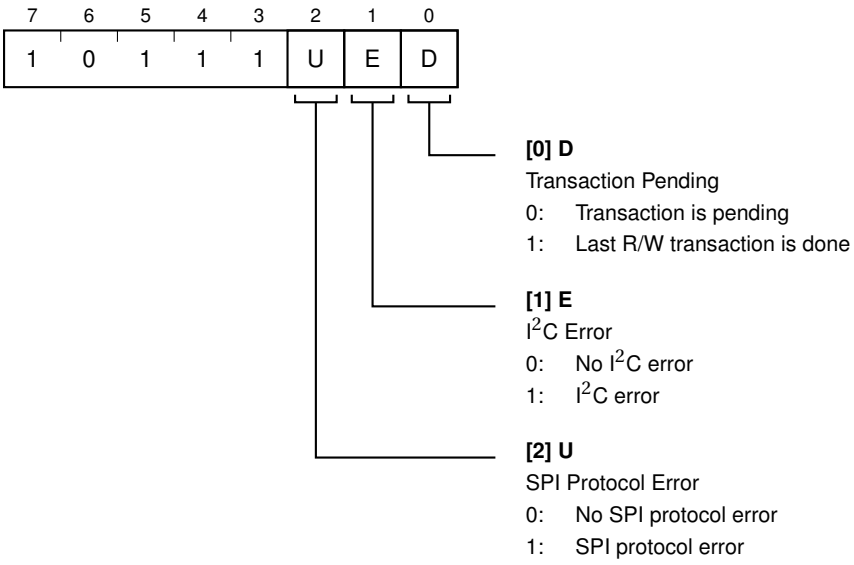

Figure 4: Get Status Response

mand by using the *get status* command. Figure 9 shows an example read interaction with a device of which is finished using the *finish read* command. The *finish read* command does not contain any bitwise parameters. The response from the device will include the number of bytes specified by the previous *read* command.

Examples

Write Operation

In the following example, the first byte sent (0x80) begins the transaction, targeting I^2C channel 0. The second byte specifies a read/write command with parameters specified in Figure 3 . The third byte (0x20) is the device address. The next byte transmitted (0x02) sets the message length of 2 bytes, followed immediately by the data bytes (both 0x00, in this case).

Figure 5: SPI Write Command Example

Figure 6 shows the clocking of the corresponding I^2C transmission to the SPI write operation specified in Figure 5 . The first byte transmitted after the start condition represents a combination of the 7-bit device address (0x20) and the read/write bit. The following two bytes are the data bytes specified in the write command.

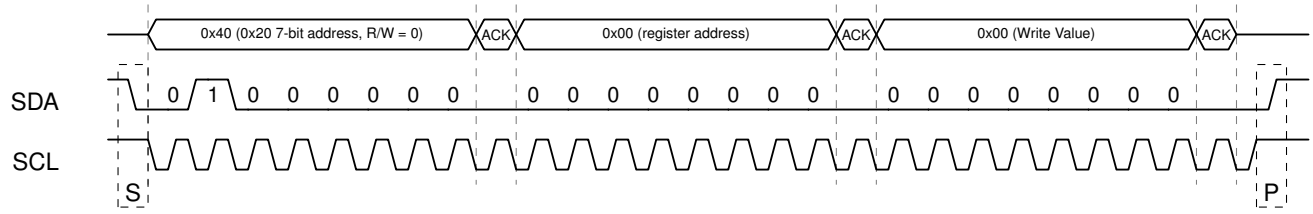

Figure 6: Corresponding I^{2}C Write Operation

Get Status Operation

The status of any previously specified operation on a channel can be checked with the *get status* command. The *get status* command does not trigger any device interaction (*i.e.,* read/write operations).

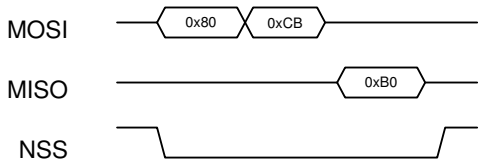

Figure 7: SPI Get Status Command Example

Read Operation

In the example in Figure 9 , a read command is initiated followed by a check of the status before the read transaction is finished. To initiate read requests on an I^{2}C device, any data required to initialize the device needs to be written to the device before sending a read command. For example, if you wish to read a device register, the desired register address should be written to the device to specify the register for subsequent reads/writes. In the case of a write, the register contents can be immediately transmitted to the device. In the case of a write, a separate transaction must be initiated. When controlling an I^{2}C bus directly, this is often done using a re-start before transmitting the device address with the R/W bit set high. This protocol wraps the read portion of the transaction in separate messages, specifying an expected number of bytes to be read from the device.

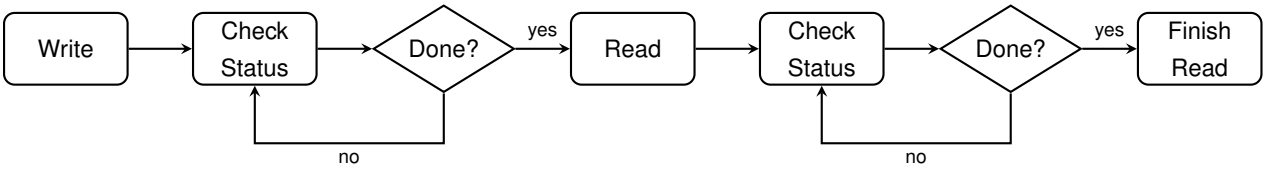

Figure 8: Read Operation Flow

1. Transmit the device address and any initializing data with a write operation
2. Check the status of the transmission using the status command

(a) Re-check for transmission completion (if necessary) or until the process returns an error/timeout

3. Initiate a read operation, specifying the number of bytes to be read from the device

4. Check the status of the read operation using the status command

   (a) Re-check for transmission completion (if necessary) or until the process returns an error/timeout

5. Complete the read operation with the finish read command to get the specified number of bytes from the device

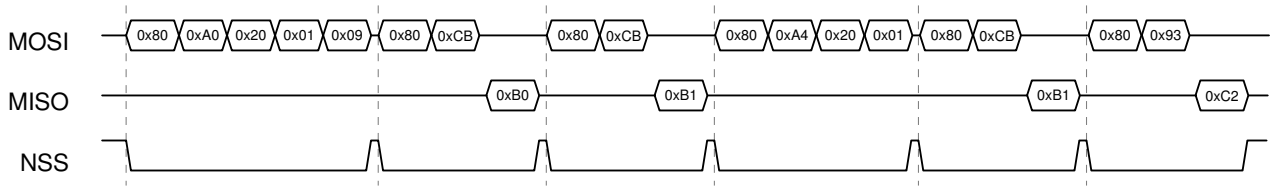

Figure 9: SPI Read Command Example

Figure 9 shows six separate interactions with the device, each separated by activation of the NSS signal. The first interaction writes the value 0x09 to the device at address 0x20. This prepares the device for reading of the register at address 0x09. The next two interactions are queries of the transmission status. The first of these interactions returns an "transaction pending" status, while the second returns "done". After receiving a completed status for the write operation, a read operation is initiated by sending the device 0xA4 followed by 0x01 to specify that one byte should be read. Again, a check of the status returns a value of 0xB1 to indicate the read transaction has been completed. The final interaction completes the read operation with the finish read transaction command (0x93) and reading of the register value from the device (0xC2). Figure 10 shows the I^2C operation corresponding to the SPI write command sequence.

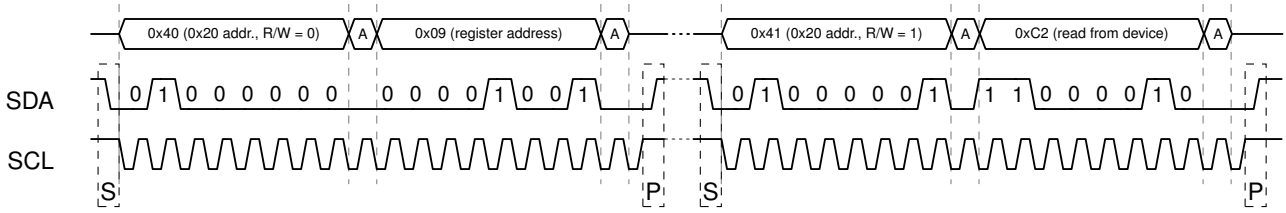

Figure 10: Corresponding I^2C Read Operation

The corresponding I^2C communications demonstrates the read operation division into two sequences. The first sequence writes the address to the device that should be read, while the second operation completes the read with 0xC2 returned from the device.