Application Note

I2C: What is the Auto Increment Feature?

Duy Nguyen

ABSTRACT

This application note includes the different types of auto increment, how the increment is performed, and the benefits of using the auto increment feature.

Table of Contents

List of Figures

Trademarks

1 What is Auto Increment?

In I2C auto increment refers to the register address automatically increasing to the next byte (such as from register 1 to register 2) when a consecutive byte is read or written without performing a stop condition.

Traditionally in I2C, to write data to a register the user needs to send the I2C target address followed by the register number then the data required to be written into the register. Figure 1-1 shows a generic example of an I2C single write transaction. If the user needed to write to additional registers in the I2C device, additional write transactions are required. This process looks similar to Figure 1-1 with multiple transactions instead of just one.

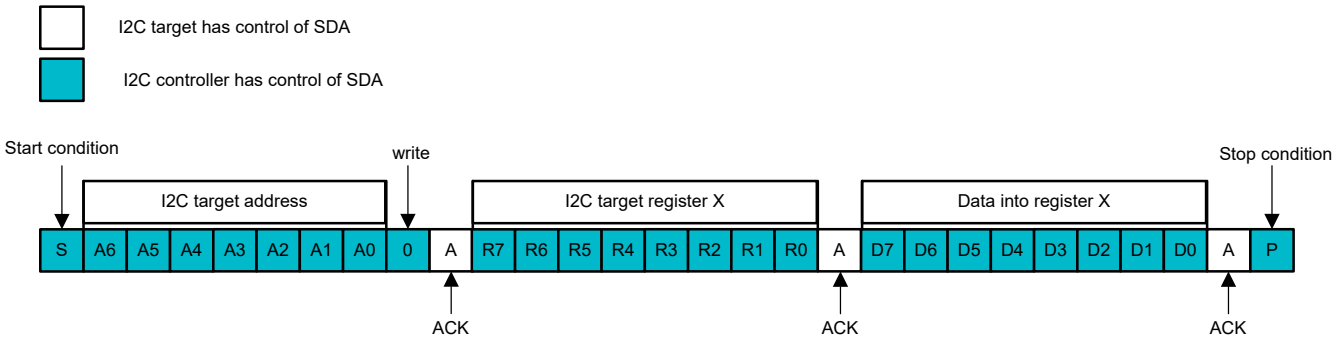

Figure 1-1. I2C Single Write

For read transactions, the I2C transactions are more involved since reads require a prior write to set the pointer or address of the read register. Figure 1-2 shows a generic example of a single read transaction which begins with a write to set the required register to read from first.

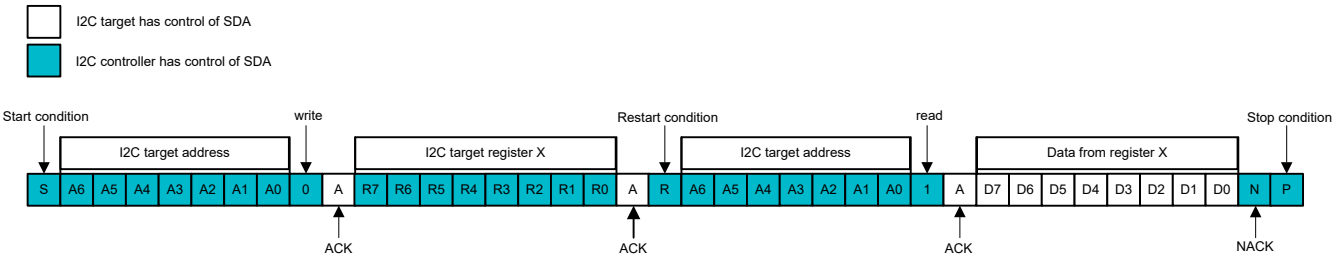

Figure 1-2. I2C Single Read

2 How is Auto Increment Useful?

Using the auto increment feature allows for the user to read or write data without needing to perform a stop condition. This can save time and computation cycles for the I2C controller during the initialization sequence for the I2C bus. An example of a burst write using the auto increment function is shown in Figure 2-1. The example showcases 4 bytes of data being written to the I2C target. Each byte of data is written to the next register in the stack beginning with the first register (register 0). The total amount of bytes sent to the I2C target is 6. If the auto increment function was not used, the number of bytes needed to perform the same transaction can total 12. This transaction cut the time required in half (for this example).

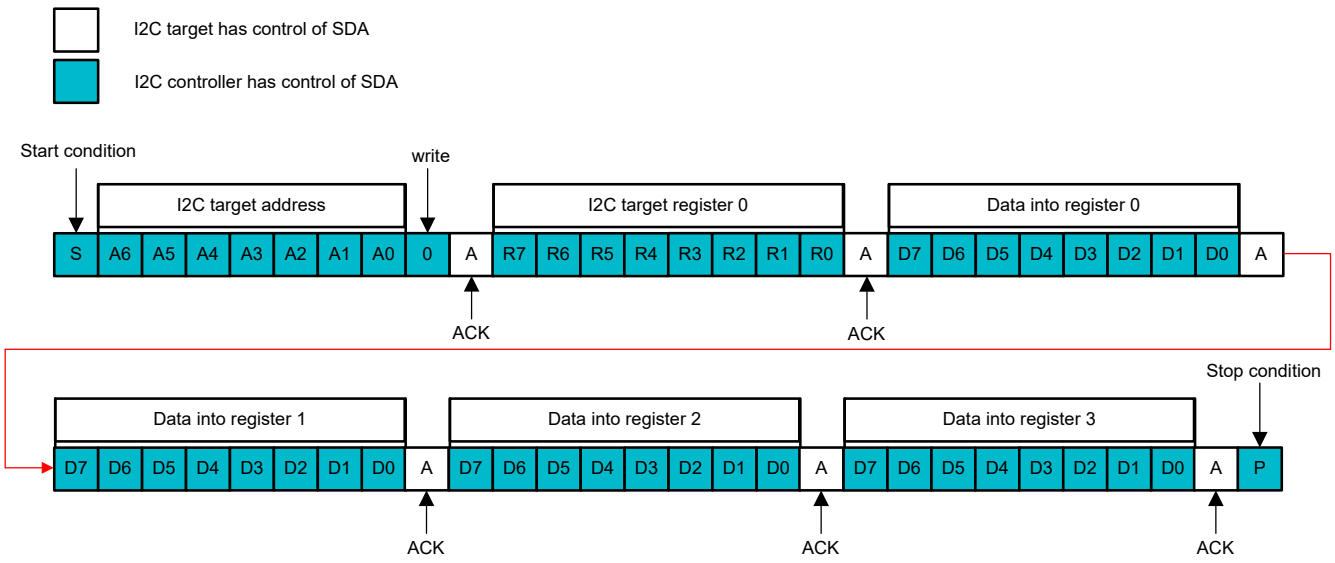

Figure 2-1. I2C Auto Increment Burst Write

A read transaction example with an auto increment is shown in Figure 2-2. This example shows the beginning register is register 5 (it does not need to start at register 0) and increments up to register 9. The total amount of bytes read from the I2C device is 5 and the number of bytes required to do this was 8. If the I2C controller (or user) had chosen to not use the auto increment, an additional 3 bytes are needed to set up to read the next register (register 6 in this example). This increment totaled 8 bytes to read 2 bytes of data. Seeing this, it is clear that auto increment provides the user with more efficient programming methods.

As an additional note, the I2C controller in this example can have theoretically continued to read bytes from this I2C target had it ACK'd after receiving data from register 9 and there was data in register 10 that it needed to collect. Essentially, read transactions can continue until the controller ends the transaction.

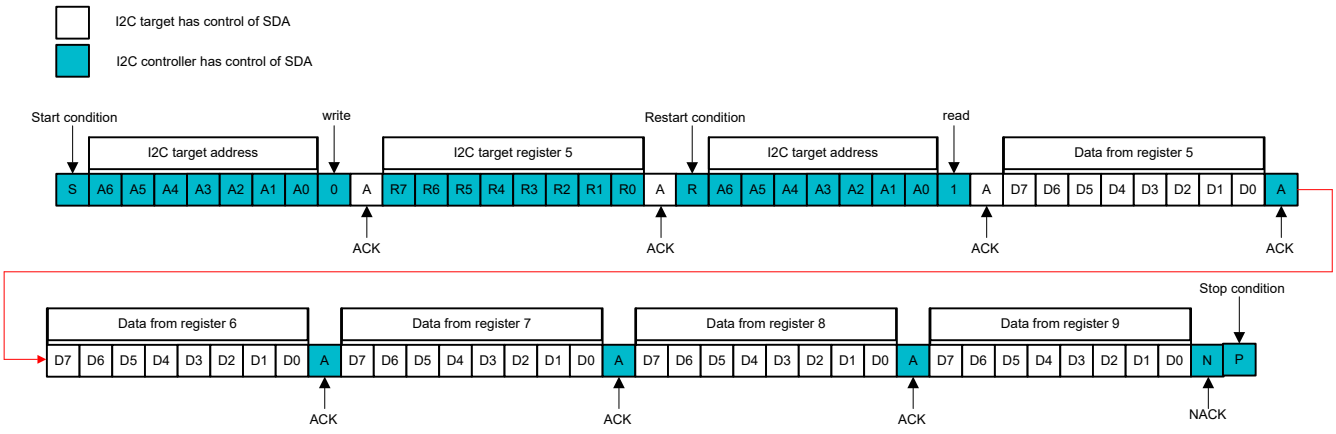

Figure 2-2. I2C Auto Increment Burst Read

3 Do All I2C Devices Support Auto Increment?

No, the auto increment feature is only supported by devices that are designed to include the feature. The auto increment function was not part of the I2C standard when function was first conceived and legacy I2C devices might not include or allow for continuous writes or reads to the registers.

4 Are There Different Types of Auto Increment?

Yes, some devices can only auto increment within a certain set of registers. An example of this is with TI I2C IO expander. For IO expander with 2 ports (16 bits of expansion), the auto increment can occur within the set of port registers. An example of this special auto increment case is shown in Figure 4-1 using a 16-bit IO expander like TCAL6416. Doing consecutive writes after setting the register to 0x02h (the output port 0 register) can result in the data written into the register to alternate between register 0x02h (output port 0 register) and 0x03h (output port 1 register).

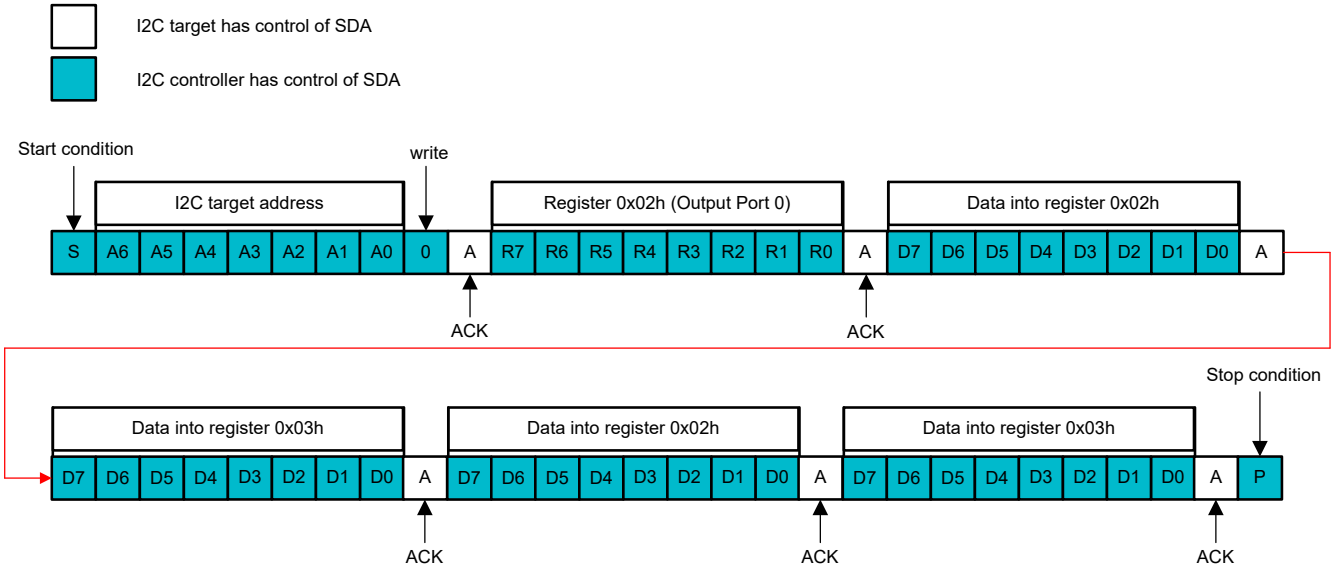

Figure 4-1. Auto Increment IO Expander

This type of auto increment feature results in an initialization sequence that requires multiple I2C transactions (with start/restart and/or stop conditions) since the user cannot write to all the registers in one go. While this is a disadvantage from an initialization perspective, this auto increment approach does come with an advantage. Being able to write or read from only one set port allows for better latency in receiving/outputting useful data.

Figure 4-2 shows a 4-bit IO expander (TCA9536) is used to toggle bits 2 and 3. The auto increment feature is being used to immediately toggle the bits after each ACK. Since the device only has one port, the auto increment feature continuously increments into the same register (output register 0x01h) so every time new data is written, the device updates at each ACK. This approach lowers the latency of the outputs toggling because after sending the register byte (0x01h) the device can freely change the output state. The alternative is to always write to the device by sending the device address, the register, and then the data in between start and stop conditions.

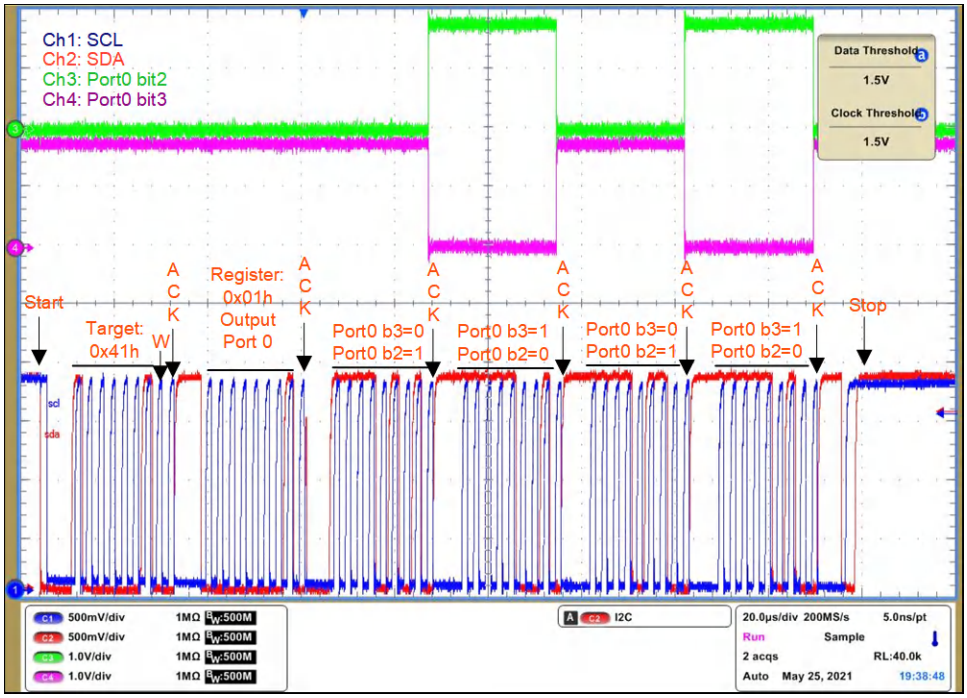

Figure 4-2. Scope Example Write Burst

From a read perspective, applications where the inputs of an IO expander need to be polled make this kind of auto increment efficient because the device does not need to increment across all of the registers before looping back to the one of interest.

5 Does Auto Increment Need to be Enabled?

It depends. In the prior example, the IO expander within TI's portfolio are all designed to automatically auto increment or loop when a continuous write or read occurs on the data frame. Other I2C devices require a write to a certain register to set a bit to enable the auto increment feature. An example of this is TI's keypad scanner (TCA8418/TCA8418E) where bit 7 in register 0x01h (configuration register) must be set to 1 to enable. For this device, the auto increment feature can fully loop through all of the registers unlike the I2C IO expander which saves a lot of time since the device has 46 total registers.

6 References

- Texas Instruments, *TCA8418 I2C Controlled Keypad Scan IC With Integrated ESD Protection*, data sheet.
- Texas Instruments, *TCA8418E I2C Controlled Keypad Scan IC With Integrated ESD Protection*, data sheet.
- Texas Instruments, *TCA9536 Remote 4-Bit I2C and SMBus I/O Expander with Configuration Registers*, data sheet.
- Texas Instruments, *TCAL6416 16-Bit Translating I2C-Bus, SMBus I/O Expander With Interrupt Output, Reset, and Agile I/O Configuration Registers*, data sheet.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265