Author: Yu-Han Sun
Date: August 04, 2015

Introduction

This app note explains how to use the SLG46531's I2C macro-cell as an input/output controller. This is a convenient way to add additional inputs and outputs in an MCU system such as shown in Figure 1.

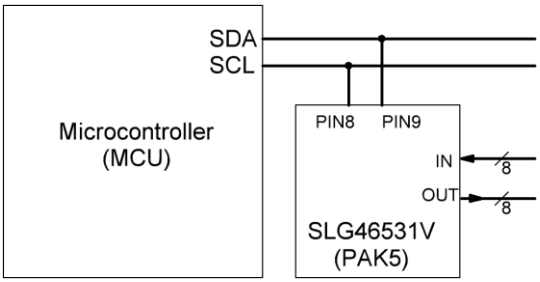

Figure 1. I/O Controller System View

Building an Output Controller-Write

The I2C interface in SLG46531 is very powerful because it can read and write all its configuration bits, including output states. However, directly writing configuration bits could be cumbersome, as they can be scattered among the 2K addressable bits. To facilitate simple, single-byte control, the SLG46531 has an I2C macro-cell which can easily be connected to other cells graphically. Figure 2 shows the I2C macro-cell outputs simply connected to output pins. The I2C macro-cell states are referred to as Virtual Inputs, and are stored contiguously at Register Bit Addresses 1952 through 1959 as documented in SLG46531's Datasheet Appendix A.

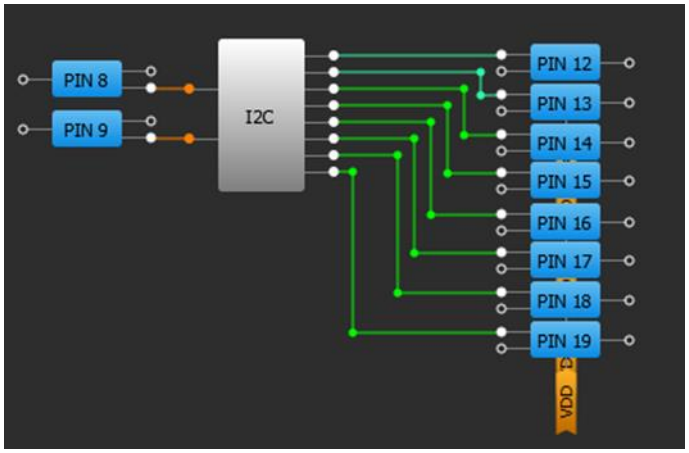

Figure 2. GUI connection of I2C macro-cell to pins

To map a Register Bit to an I2C write, simply divide by the Register Bit Address by 8 to get the Word Address, and take the modulo 8 remainder to get the bit position in the Data Byte. For example, Register Bit 1959 corresponds to Word Address 0xF4, bit position 7 for the I2C command. See Table 1 for the mapping of all the Virtual Inputs.

An I2C write can be structured as follows:

1. Start
2. Control Byte (includes device address, R/W bit=0)
3. Word Address
4. Data
5. Stop

(Please see datasheet section **19.4.1 "I2C Byte Write"** for detailed description).

Word Address		Data Bit Position	Register Bit Definition	SLG46531 Register Bit
244	0xF4	0	Virtual Input 0	1952
"	"	1	Virtual Input 1	1953
"	"	2	Virtual Input 2	1954
"	"	3	Virtual Input 3	1955
"	"	4	Virtual Input 4	1956
"	"	5	Virtual Input 5	1957
"	"	6	Virtual Input 6	1958
"	"	7	Virtual Input 7	1959

Table 1. Virtual Inputs Register Bit Addresses

Examples of I2C writes to the Virtual Inputs are shown in Table 2. Please note the following:

1. Open bracket ' [' denotes START, and close bracket ']' denotes STOP

2. The upper 4 bits of the first byte (the Control Byte) are used for device addressing, and are programmable in each SLG46531.

This allows up to 16 SLG46531 devices to be individually addressed on the same bus. The next 3 bits are hard-wired to 000 in SLG46531. The final bit of the Control Byte is R/W which is 0 for write. For example, a device programmed with "0011" is written to with 0x30 as the Control Byte.

I2C Command	Result
[0x30, 0xF4, 0xFF]	Write to device with programmed address bits "0011". Register at word address 0xF4 will be written with 0xFF, resulting in all virtual bits set HIGH.
[0x10, 0xF4, 0x00]	Write to device with programmed address bits "0001". Register at word address 0xF4 will be written with 0x00, resulting in all virtual bits set LOW.

Table 2. Example Commands

Waveforms – Writing Virtual Bits

I2C commands to write virtual bits are captured in the waveforms below. Channel 1 and 2 are the I2C clock (SCL) and data (SDA), respectively. Digital signals D0 through D7 correspond to the virtual bits connected as shown Figure 2. The I2C interface is running at 400 kHz with 10kohm resistors pulling up to 3.3 V on the SDA and SCL lines.

D0- PIN#12 (OUT0)

D1- PIN#13 (OUT1)

D2- PIN#14 (OUT2)

D3- PIN#15 (OUT3)

D4- PIN#16 (OUT4)

D5- PIN#17 (OUT5)

D6- PIN#18 (OUT6)

D7- PIN#19 (OUT7)

Channel 1 (yellow/second bottom) - PIN#8 (SCL)

Channel 2 (light blue/bottom) - PIN#9 (SDA)

The captures show the complete I2C sequence: START, Control Byte, Word Byte, Data Byte, STOP As can be seen, the outputs (pins 12 through 19) are updated at the last (8th) SCL falling edge at the end of the Data Byte.

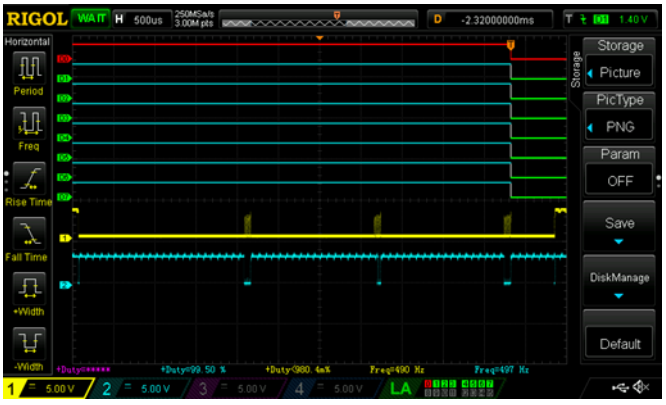

Figure 3. I2C Command [0x00, 0xF4, 0x00]

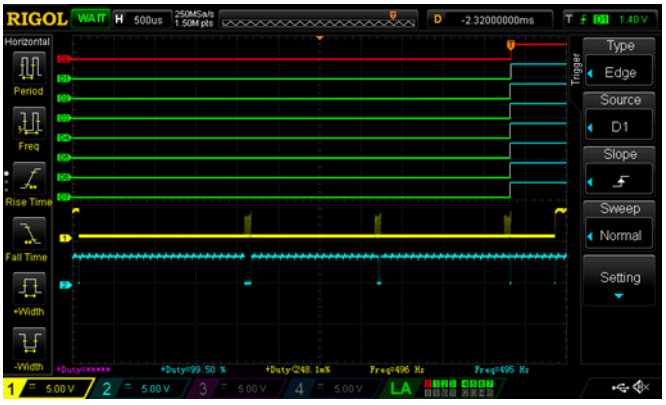

Figure 4. I2C Command [0x00, 0xF4, 0xFF]

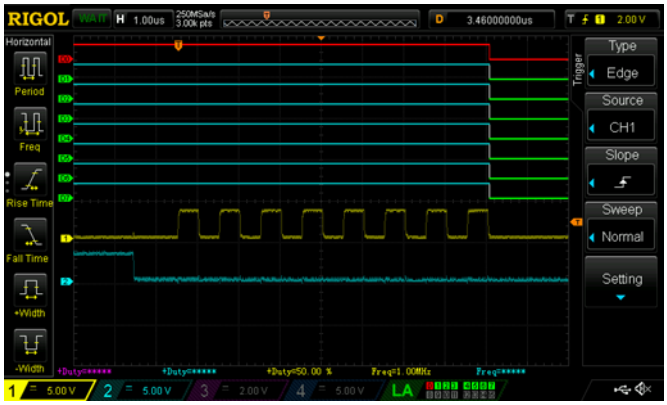

Figure 5. 1us scale of Data Byte 0x00

Figure 6. 1us scale of Data Byte 0xFF

Building an Input Controller - Read

An I2C read can be structured as follows:

1. Start
2. Control Byte (R/W=0)
3. Word Address (sets the internal address counter on device)
4. Start
5. Control Byte (R/W=1)
6. Data (from device)
7. Stop

(Please see datasheet section **19.4.4 "I2C Random Read"** for detailed description).

For an example, we will read GPIO 2 through 20 (no pin11 as it is GND). Those Digital Input bits are stored at Register bit addresses <1921:1927>, and <1968:1976> as documented in SLG46531's Datasheet Appendix A. The relevant information from that appendix is shown in Table 3 below.

Word Address		Data Bit Position	Register Bit Definition	SLG46531 Register Bit
240	0xF0	1	Pin2 Digital Input	1921
"	"	2	Pin3 Digital Input	1922
"	"	3	Pin4 Digital Input	1923
"	"	4	Pin5 Digital Input	1924
"	"	5	Pin6 Digital Input	1925
"	"	6	Pin7 Digital Input	1926
"	"	7	Pin10 Digital Input	1927
246	0xF6	0	Pin12 Digital Input	1968
"	"	1	Pin13 Digital Input	1969
"	"	2	Pin14 Digital Input	1970
"	"	3	Pin15 Digital Input	1971
"	"	4	Pin16 Digital Input	1972
"	"	5	Pin17 Digital Input	1973
"	"	6	Pin18 Digital Input	1974
"	"	7	Pin19 Digital Input	1975
247	0xF7	0	Pin20 Digital Input	1976

Table 3. GPIO Input Register Bit Address

Examples of I2C reads to the Virtual Inputs are shown in Table 4. Please note the following:

1. Open bracket ' [' denotes START, and close bracket ']' denotes STOP

2. The upper 4 bits of the first byte (the Control Byte) are used for device addressing, and are programmable in each SLG46531. This allows up to 16 SLG46531 devices to be individually addressed on the same bus. The next 3 bits are hard-wired to 000 in SLG46531. The final bit of the Control Byte is R/W which is 0 for write. For example, a device programmed with "0011" is written to with 0x30 as the Control Byte.

3. The 'r' byte is the read data sent from the slave device. Acknowledges from the master will cause the slave device to send another byte of data until there is a STOP.

The second command in Table 2 follows the same format as the previous example except two bytes are read, corresponding to Pins 12 through 20. Note that pin20 data is the 1st bit in the second data byte.

Bench testing – Reading GPIOs

For this bench test, the GPIOs 2 through 10 are read through I2C using a blank chip. First place a blank device into the development kit, start the emulator and power the SLG46531 in emulation mode. Be sure that all External Connectors for SCL and SDA, Pins 8 and 9, are enabled. Connect the master device SCL and SDA to the External Connectors and begin sending I2C commands.

The default I2C macro-cell address on a blank chip is "0000". Below is a demonstration of the first command from Table 5 if Pins 2 and 3 are HIGH. Figure 7 has the emulator configuration and Figure 8 shows the results in a terminal program. The result is as expected with a read data byte of 0x06, this indicates that bit positions 3 and 2 of 0xF0 were logic HIGH. Referring back to Table 3, it is clear that these positions belong to Pins 2 and 3.

Command	Result
[0x00, 0xF0, [0x01, r]	Read from device with programmed address bits "0000". The contents at Word Address 0xF0 will be read.
[0x30, 0xF6, [0x31, r, r]	Read from device with programmed address bits "0011". The contents at Word Address 0xF6 and the following byte 0xF7 will be read.

Table 4. Example Commands

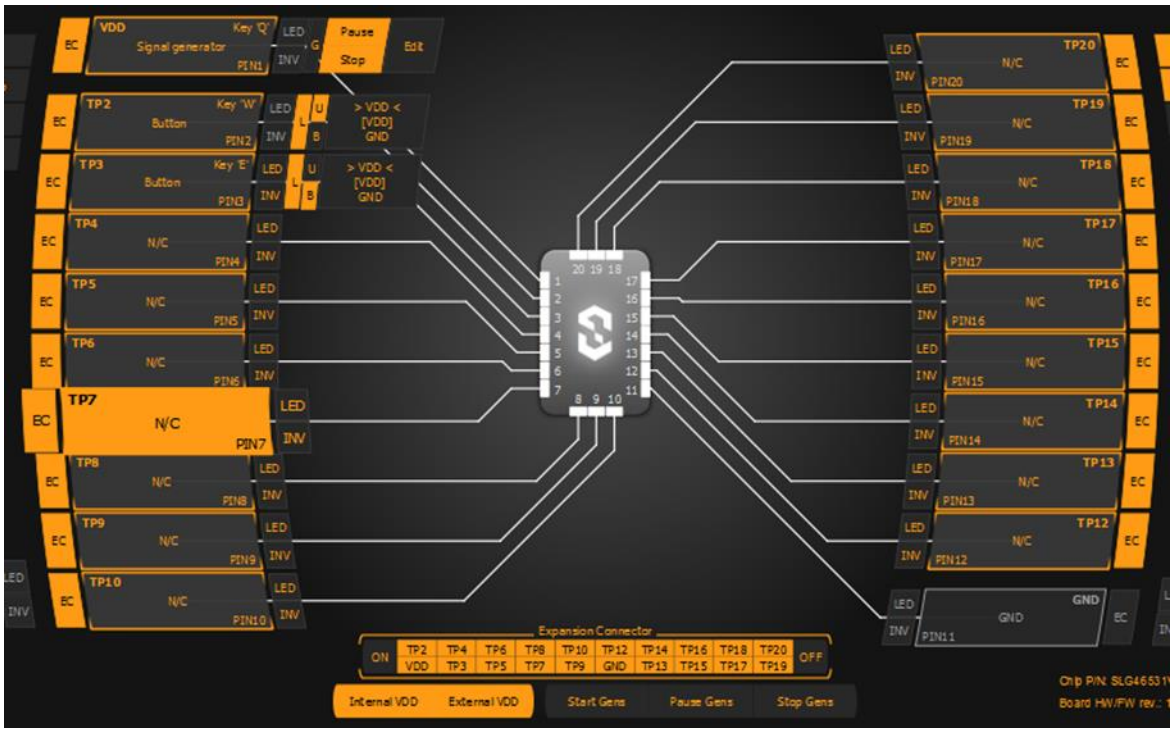

Figure 7. SLG46531 Emulator Setup for a blank chip. Pins 2 and 3 are HIGH

```
I2C>[0x00 0xF0 [ 0x01 r]
I2C START BIT
WRITE: 0x00 ACK
WRITE: 0xF0 ACK
I2C START BIT
WRITE: 0x01 ACK
READ: 0x06
NACK
I2C STOP BIT
I2C>
```

Figure 8. Terminal of Read Command

Conclusion

This App Note explained how to use the SLG46531's I2C macro-cell as an input/output controller. This technique can be useful if the user needs a way to transmit data to or from a microcontroller without using multiple GPIOs. The ability to write to Virtual Inputs 7:0 and read GPIOs via I2C makes the SLG46531 a more flexible device with more possible configurations.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), 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 OR 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 developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.