AN10397

How to use the SC18IM700 to control any I²C-bus device

Semiconductors

Document information

Info	Content
Keywords	I2C, I2C controller, Master I2C, UART
Abstract	This application note shows how the SC18IM700 can be used in a system to control other I²C-bus slave devices.

PHILIPS

Revision history

Rev	Date	Description
01	20051205	Application note; initial version.

Contact information

For additional information, please visit: **http://www.semiconductors.philips.com**

For sales office addresses, please send an email to: **sales.addresses@www.semiconductors.philips.com**

AN10397_1

Application note **Rev. 01 — 5 December 2005** **2 of 10**

1. Introduction

Philips Semiconductors recently introduced a whole new family of devices called 'the Bridges'. These devices are intended to transform data from one serial bus to another serial bus, and this transformation allows the host to control devices that have serial host bus interfaces that are not native to the system.

One such scenario is the I²C-bus interface. There are wide ranges of devices that have an I²C-bus interface to communicate with a host. Some of such devices are: EEPROMs, temperature sensors, analog-to-digital or digital-to-analog converters, LED blinkers and I/O expanders. To use these devices in the system, a host must have an integrated I²C-bus controller on-board, or it must have an external, stand-alone I²C-bus controller.

Philips Semiconductors' SC18IM700—an I²C-bus controller with a UART host interface—is the perfect choice in the case where the host does not have an integrated I²C-bus controller on-board, and the system designer wishes to use I²C-bus related devices in the system. The SC18IM700 does not require any programming at all other than the code to write and read from the host's UART port. This non-programming is possible because SC18IM700 communicates with the host through a series of messages that are based on ASCII characters.

This application note shows how a host can control, setup, read and write to any I²C-bus devices through this UART-to-I²C controller. The examples used in this application note are built around an LED blinker from Philips Semiconductors, the PCA9531. Besides the I²C-bus control function, the SC18IM700 also contains a general-purpose 8-bit programmable I/O port. These eight general purpose I/O pins can be configured to the following modes: input only, open-drain output, push-pull output or quasi-bidirectional input/output.

2. PCA9531: LED blinker

The PCA9531 is an 8-bit I²C-bus and SMBus I/O expander optimized for dimming LEDs in 256 discrete steps for Red/Green/Blue (RGB) color mixing and backlight applications.

The initial setup sequence programs the two blink rates and duty cycles for each individual PWM. From then on, only one command from the bus master is required to turn individual LEDs ON, OFF, BLINK RATE 1 or BLINK RATE 2. Based on the programmed frequency and duty cycle, BLINK RATE 1 and BLINK RATE 2 will cause the LEDs to appear at a different brightness or blink at periods up to 1.69 second. The open-drain outputs directly drive the LEDs with maximum output sink current of 25 mA per bit and 100 mA per package.

Please refer to the PCA9531 data sheet for more detail about this device and its internal registers' usage. The following steps to program the part are extracted from the example in the data sheet, page 11.

2.1 Programming example

The following example will show how to set LED0 to LED3 on. It will then set LED4 and LED5 to blink at 1 Hz at a 50 % duty cycle. LED6 and LED7 will be set to be dimmed at 25 % of their maximum brightness (duty cycle = 25 %).

Table 1: **PCA9531 programming example**

Programming step	I2C-bus data[1]
START	S
PCA9531 address with A0, A1, A2 = LOW	C0h
PSC0 subaddress + auto-increment	11h
Set prescaler PSC0 to achieve a period of 1 second: $\text{blink period} = 1 = \dfrac{PSC0 + 1}{152}$ PSC0 = 151	97h
Set PWM0 duty cycle to 50 %: $\dfrac{PWM0}{256} = 0.5$ PWM0 = 128	80h
Set prescaler PCS1 to dim at maximum frequency: $\text{blink period} = max$ PSC1 = 0	00h
Set PWM1 output duty cycle to 25 %: $\dfrac{PWM1}{256} = 0.25$ PWM1 = 64	40h
Set LED0 to LED3 on	55h
Set LED4, LED5 to PWM0, and LED6, LED7 to PWM1	FAh
STOP	P

[1] This column shows the steps needed to be performed by the host to set the LED to blink at a specified rate.

AN10397_1

Application note **Rev. 01 — 5 December 2005** **4 of 10**

3. SC18IM700: UART to I²C-bus controller

The SC18IM700 is designed to serve as an interface between a standard UART of a microcontroller/microprocessor and the serial I²C-bus. This allows the microcontroller/microprocessor to communicate directly with other I²C-bus devices. The SC18IM700 can operate as an I²C-bus master and can be a transmitter or a receiver. The SC18IM700 controls all the I²C-bus specific sequences, protocol, arbitration and timing. The host communicates with the SC18IM700 with ASCII messages protocol; this makes the control sequences from host to SC18IM700 become very simple.

The host initiates I²C-bus data transfer, read from and write to SC18IM700 internal registers through a series of ASCII commands. Table 2 lists the ASCII commands supported by SC18IM700, and also their hex value representation. Unrecognized commands are ignored by the device.

Table 2:   ASCII commands supported by the SC18IM700

ASCII command	Hex value	Command function
S	0x53	I²C-bus START command
P	0x50	I²C-bus STOP command
R	0x52	read SC18IM700 internal register command
W	0x57	write to SC18IM700 internal register command
I	0x49	read GPIO port command
O	0x4F	write to GPIO port command
Z	0x5A	power-down

Please refer to the SC18IM700 data sheet for more descriptive detail of the above commands.

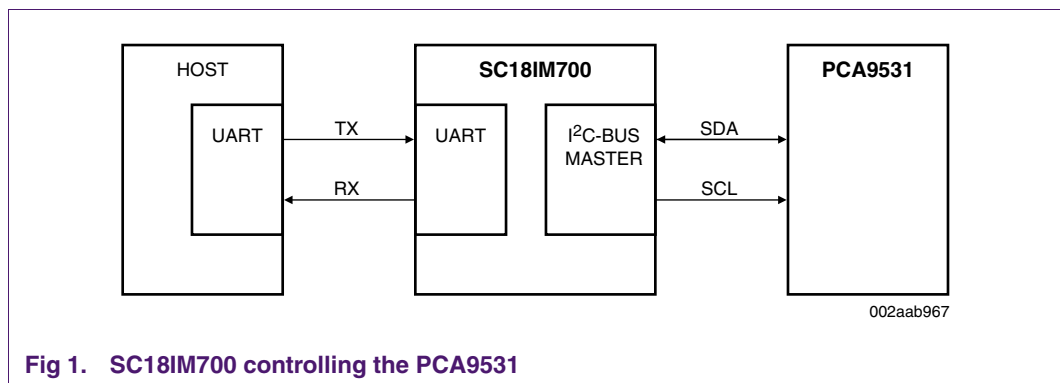## 4. Using the SC18IM700 to control the PCA9531

Fig 1.   SC18IM700 controlling the PCA9531

The 'S' command can be used to setup the PCA9531's internal registers to blink the LED at specified frequency and rate. The format for the 'S' command is shown in Figure 2.
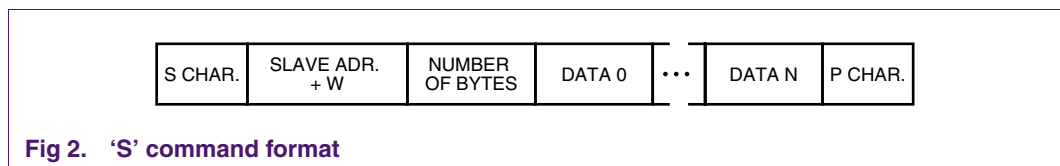

Fig 2.   'S' command format

The command begins with an 'S' character as the first byte, the second byte specifies the I^{2}C-bus slave address of the I^{2}C-bus device, where 'W' is the least significant bit and it is set to a logic 0. The third byte indicates the number of data bytes in the message (DATA 1 to DATA N), and the last byte ended with the 'P' character.

To turn on and blink the LED, the host would send a message to the SC18IM700 as follows: **S C0 07 11 97 80 00 40 55 FA P**.

Once the SC18IM700 receives this message from host it will translate and will send the same message through the I^{2}C-bus to the PCA9531. The LED should be now blinking.

If the host wishes to read the PCA9531 internal registers to make sure they are correctly set after the host has written to them, the 'Read after Write' command can be used for that purpose. The format for the 'Read after Write' command is as shown in Figure 3.

S CHAR.	SLAVE ADR. + W	NUMBER OF BYTES	DATA 1	...	DATA N	S CHAR.	SLAVE ADR. + R	NUMBER OF BYTES	P CHAR.

Fig 3. 'Read after Write' command format

Using the first example the host sends the following message to the SC18IM700:

S C0 07 11 97 80 00 40 55 FA **S C1 07 P**

SC18IM will send the message to PCA9531, then will read seven bytes from PCA9531 and sends them to the host automatically. The host will receive the following byte from SC18IM700 through its UART port:

11 97 80 00 40 55 FA

The host can also ask SC18IM700 to automatically send an I^{2}C-bus' transaction status information to the host after SC18IM700 has sent the host's message to an I^{2}C-bus slave. The host can do this by sending a 'Register Read' command to read the I2Cstatus register after any normal I^{2}C-bus message. Using the above example, the host would send the following message to SC18IM700:

S C0 07 11 97 80 00 40 55 FA P **R 0A P**

Once the message is sent to the I^{2}C-bus device at address 0xC0, SC18IM700 will send the I2C_OK (hex code 0xF0) status to the host. See *Table 9* of the data sheet for other I2C Status Code.

Upon receiving the I2C_OK status the host can send another message to SC18IM700.

AN10397_1

Application note   Rev. 01 — 5 December 2005   6 of 10

5. Using the 8-bit programmable GPIO port

GPIO 0 to GPIO 7 ports may be configured by the host to one of four types. These are: quasi-bidirectional, push-pull, open-drain, and input-only. Two bits are used to select the desired configuration for each port pin. PortConf1 is used to select the configuration for GPIO[3:0], and PortConf2 is used to select the configuration for GPIO[7:4]. Each port pin has Schmitt triggered input that also has a glitch suppression circuit.

Table 3:   Port configurations

IOx.1	IOx.0	Port configuration
0	0	quasi-bidirectional output configuration
0	1	push-pull output configuration
1	0	input-only configuration
1	1	open-drain output configuration

Normally to read the actual state of the input pins the host must read the IOState register, or to write to any of the output pins the host would perform a write to the IOState register. But, there are two commands that can be used to simplify the process of controlling the GPIO port: they are the 'I' and the 'O' commands. Instead of reading and writing to the IOState register, the host would send these two commands in form of a message.

Let's say GPIO0 to GPIO3 are programmed as inputs, and GPIO4 to GPIO7 are programmed as outputs, and the host wants to read the state of all of the input pins and to set all the output pins to 1.

To read the input pins the host sends this message to SC18IM700:

   I P

The SC18IM700 will return a byte to the host through the UART port, bit 3 to bit 0 of this byte indicate the actual state of the input pins; bit 7 to bit 4 indicate the actual state of the output pins.

To set the output pins, the host sends this message to SC18IM700:

   O FX P

Where the 'X' can be any value and will not have any effect because GPIO3 to GPIO0 are programmed as inputs. GPIO7 to GPIO4 should all go HIGH once this message is sent.

AN10397_1

Application note **Rev. 01 — 5 December 2005** **7 of 10**

6. Conclusion

As shown in the previous examples, the SC18IM700 can be used to control any I^{2}C-bus device. This device does not require any programming at all because the commands are sent from a host in the form of ASCII characters. The 8-bit general purpose programmable port offers the system great flexibility, and the device can be used as an expandable 8-bit port with a UART interface.

7. Abbreviations

Table 4: **Abbreviations**

Acronym	Description
ASCII	American Standard Code for Information Interchange
EEPROM	Electrically Erasable Programmable Read Only Memory
GPIO	General Purpose Input/Output
I^{2}C-bus	Inter IC bus
LED	Light Emitting Diode
PWM	Pulse Width Modulator
SMBus	System Management Bus
UART	Universal Asynchronous Receiver/Transmitter

AN10397_1

Application note **Rev. 01 — 5 December 2005** **8 of 10**

8. Disclaimers

Life support — These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes in the products - including circuits, standard cells, and/or software - described or contained herein in order to improve design and/or performance. When the product is in full production (status 'Production'), relevant changes will be communicated via a Customer Product/Process Change Notification (CPCN). Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no licence or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

9. Trademarks

Notice — All referenced brands, product names, service names and trademarks are the property of their respective owners.
I2C-bus — logo is a trademark of Koninklijke Philips Electronics N.V.

10. Contents

PHILIPS