



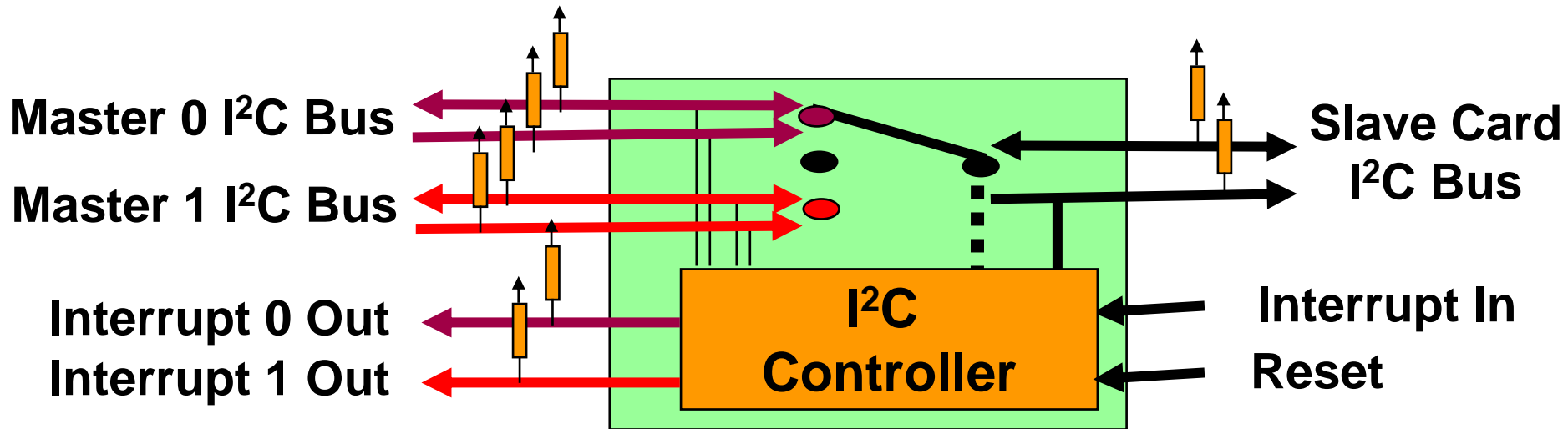
I2C 2005-1 Demonstration Board

I²C-bus Master Selector

Oct, 2006



2 to 1 I²C Master Selector w/Interrupt Logic and Reset



FEATURES

- Select one of two I²C masters to a single channel
- I²C/SMBus commands used to select channel
- Reset or Power On Reset (POR) resets state machine
- Interrupt outputs also report demultiplexer status
- Sends 9 clock pulses and stop condition to clear slave card prior to transferring master

KEY POINTS

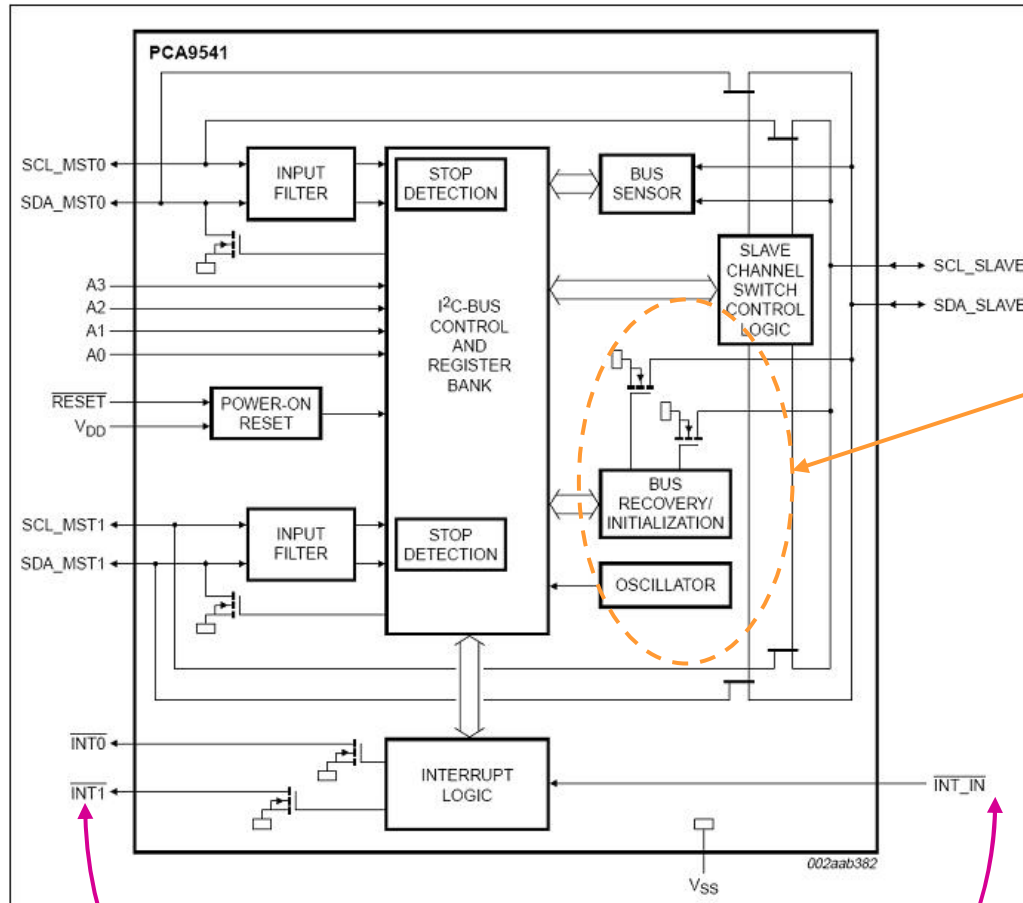
- Allows primary and backup master to communicate to one downstream slave card.
- Arbitration circuit between bus masters
- Doesn't isolate bus capacitance
- Allows voltage translation between 1.8 V, 2.5 V, 3.3 V and 5 V
- Idle detect for live insertion protection

- **PCA9541/01** - defaults to channel 0 on start-up/reset
- **PCA9541/02** - defaults to channel 0 on start-up/reset after stop condition
- **PCA9541/03** – defaults to off on start-up/reset, master commands channel

PCA9541 – In a nutshell

- Two upstream I²C channels (Masters) + One downstream channel (Slave)
- One master at a time has control of the downstream channel: allows a primary and a backup master to communicate with one downstream slave I²C bus
- Two separate state machines with same control scheme on both channels (same code)
- The other master is completely isolated from the downstream channel
- Switch commands done through the I²C bus (no additional pin required)
- Interrupt pins provide feedback & channel status to the two masters
- Interrupt input pin allows downstream interrupt sources to be forwarded to the two masters
- Each Interrupt source can be masked if not required in the application
- Bus recovery sequence allows the channel switch to be performed on a clean & idle bus
- Reset or Power On Reset (POR) resets state machines
- Allows voltage translation between 1.8 V, 2.5 V, 3.3 V and 5 V
- 2.5 V to 5.5 V power supply with 5 V tolerant I/O's
- Idle detect circuitry for live insertion protection
- Does not isolate bus capacitance (no buffer capability)

Device Hardware



I²C MASTER 0

Programmable
I²C Addresses

16 devices in
the same bus
Device powers
up and resets to
its default state

I²C MASTER 1

I²C SLAVE

Bus recovery:
PCA9541 =
master device

- PCA9541/01:
Channel 0 on (default)
- PCA9541/02:
Channel 0 on after a
Stop (default)
- PCA9541/03:
no Channel enabled
(default)

Interrupt function:

- Input = collect interrupts from slave bus
- Output = information to the masters

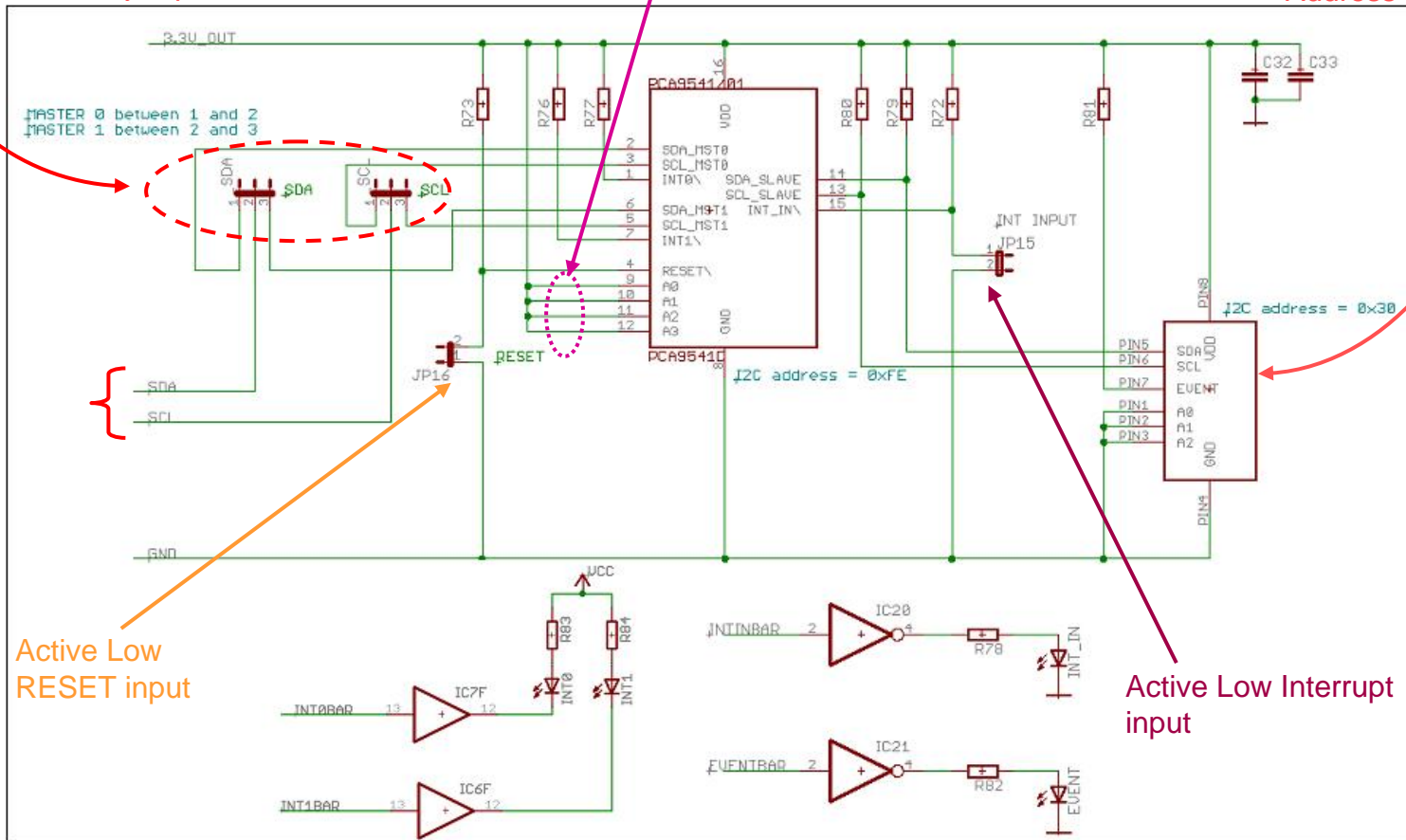
Demonstration Board: Hardware Introduction

MASTER 0 and MASTER 1 emulated with jumper SDA and jumper SCL

PCA9541/01:
I²C address: A[3:0] = 1111 → Address = 0xFE

SE98: slave device on the downstream bus
Address = 0x30

MAIN I²C bus – One single Master

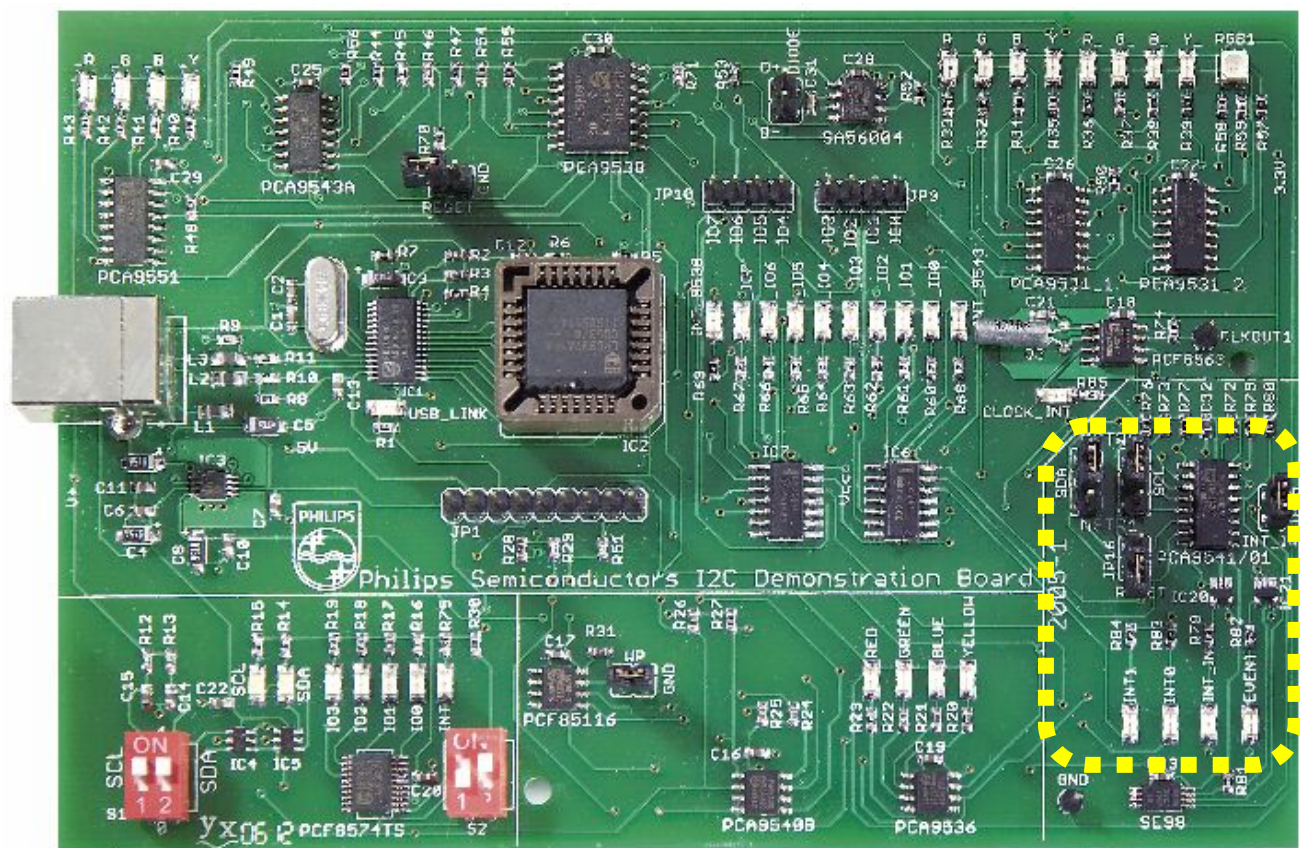


Active Low
RESET input

Active Low Interrupt
input

Visual status: Interrupt input and outputs

Demonstration Board: Hardware Introduction



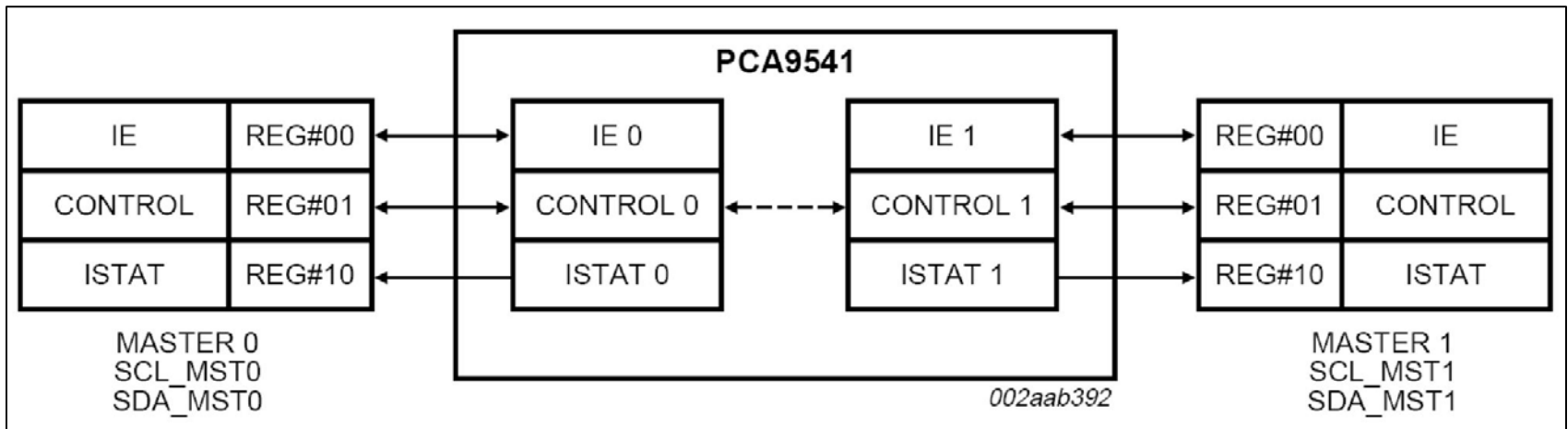
CONTROL Registers – Introduction

- PCA9541 has 2 state machines
- Each state machine has 3 registers
 - CONTROL Register (R/W):

Allows channel control and selects the different modes
 - INTERRUPT STATUS Register (R only):

Gives information about the source of an Interrupt
 - INTERRUPT ENABLE Register (R/W):

Enables or disables the possible Interrupt sources



CONTROL Register (1): Bus Control & Switch

- Bus Control & Switch done through the 4 LSB's in the CONTROL register (Reg# 01)

7	6	5	4	3	2	1	0
NTESTON	TESTON	0	BUSINIT	NBUSON	BUSON	NMYBUS	MYBUS

- 2 bits for Bus Control: **MYBUS** (R/W) and **NMYBUS** (R only)
- 2 bits for Channel ON/OFF Switch: **BUSON** (R/W) and **NBUSON** (R only)
- The two Read Only bits (information from the other state machine) associated with the two Read/Write bits allow the requestor to enable/disable the channel and take/give up control of the I²C bus
- One of the 2 masters is ALWAYS in control of the I²C bus
- The 3 versions (/01, /02 and /03) work the same way – Only power up configuration will be different
- Truth Tables (As a master reads its Control Register):

NMYBUS R only	MYBUS R/W	DOES THE MASTER CONTROL THE I ² C BUS?
0	0	YES
0	1	NO
1	0	NO
1	1	YES

To give up control of the I²C bus

To take control of the I²C bus

NBUSON R only	BUSON R/W	ARE THE UP/DOWNSTREAM CHANNELS CONNECTED?
0	0	NO
0	1	YES
1	0	YES
1	1	NO

To disable the I²C bus

To enable the I²C bus

CONTROL Register (2): Bus Initialization

- This feature is controlled by the bit BUSINIT in the CONTROL Register (Reg# 01)

7	6	5	4	3	2	1	0
NTESTON	TESTON	0	BUSINIT	NBUSON	BUSON	NMYBUS	MYBUS

- When set (= 1), the PCA9541 becomes a master to initialize the I²C-bus before switching to the master that requested bus control
- When reset (= 0), the PCA9541 performs the bus switch without initiating a bus initialization/recovery function. The master located in the upstream channel will have to initiate it
- Bus Initialization sequence:
 - 1) PCA9541 sends 9 clock pulses and SDA is released High all the time
 - 2) PCA9541 sends a STOP command
- This sequence will complete any read transaction which was previously in process and the downstream slave configured as a slave-transmitter should release the SDA line because the PCA9541 did not acknowledge the last byte (SDA High)

CONTROL Register (3): Interrupt Lines Test

- Feature is controlled by the bits TESTON and NTESTON in the CONTROL Register (Reg# 01)

7	6	5	4	3	2	1	0
NTESTON	TESTON	0	BUSINIT	NBUSON	BUSON	NMYBUS	MYBUS

- Allows test of the Interrupt output lines (/INT0 and /INT1). Can be part of power up checks to be sure that the Interrupt lines work fine.
 - When a master sets its TESTON bit (= 1), its Interrupt line is asserted (goes low)
 - When a master sets its NTESTON bit (=1), the other Interrupt line is asserted
 - Interrupt assertion can also be checked in the Interrupt Status Register (Reg #03)
 - Interrupt is cleared by resetting the bit that has been previously set to 1
- Other application: Handshake or “Bus Busy” signal between masters
 - Useful feature for applications that need to share a common downstream resource
 - When NTESTON is set, it allows the other master to know that the bus is busy with the other master or that a specific action needs to be taken

PCA9541 Software GUI:

Device → Master Selector (2-to-1 demux) → PCA9541

Interrupt Enable register Control register Interrupt Status register

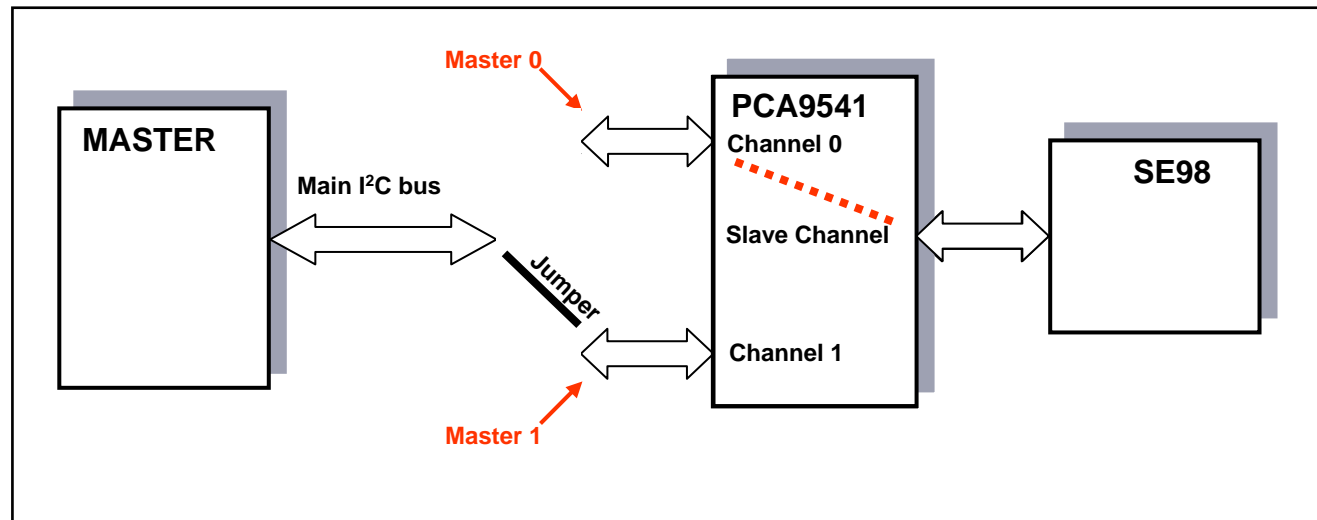
I²C address = 0xFE

Auto Write Feature

NMYBUS	MYBUS	Slave channel	NBUSON	BUSON	Slave channel
0	0	The master reading this combination has control of the bus.	0	0	off
1	0	The master reading this combination does not have control of the bus.	1	0	on
0	1	The master reading this combination does not have control of the bus.	0	1	on
1	1	The master reading this combination has control of the bus.	1	1	off

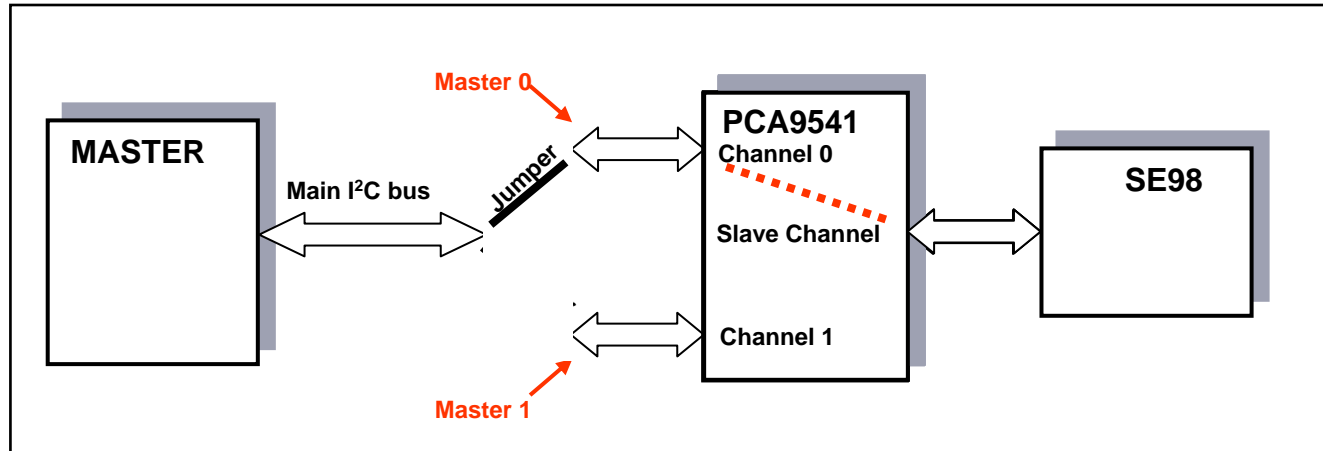
Hardware Detected 3.3V Off 5.0V Off 100 kHz

Hands-On 1 : Bus Control & Switch (1)



1. a) Jumpers SDA and SCL are connected to Master 0 (Upper position)
b) Power sequence the board → Board OFF then ON
2. Open the PCA9541 and SE98 GUI's:
PCA9541: Device → Master Selector (2-to-1 demux) → PCA9541
SE98: Device → Thermal Management → SE98
3. a) PCA9541: Read the CONTROL Register. What is the result and what does it mean?
b) SE98: Read the Manufacturer ID. What do you see?
PCA9541: Reg#1 = 04h: MASTER 0 controls the slave bus and the switch is ON
SE98: Transmission is successful and Manufacturer ID = 1131h
4. Connect jumpers SDA and SCL to Master 1 (Lower position)

Hands-On 1 : Bus Control & Switch (2)



5. a) PCA9541: Read the CONTROL Register. What is the result and what does it mean?
b) SE98: Read the Device ID. What do you read?

PCA9541: Reg#1 = 0Ah: MASTER 1 doesn't control the slave bus and the switch is ON
SE98: Transmission is unsuccessful (NACK) and Device ID can't be read

6. a) Program the CONTROL Register so that MASTER 1 takes control of the bus in order to communicate with SE98
b) SE98: Read the Device ID. What do you read?

SE98: Transmission is now successful (ACK) and Device ID = 0A101h

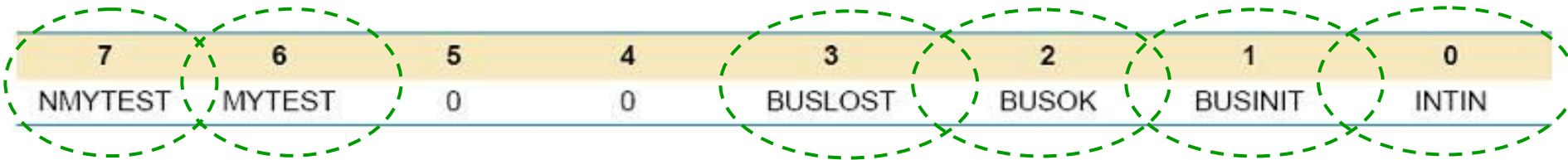
7. Connect jumpers SDA and SCL to Master 0 (Upper position)

8. **Redo Step 5 and then Step 6 (Read SE98 Capability instead of Device ID)**

PCA9541: Reg#1 = 04h: MASTER 0 doesn't control the slave bus and the switch is ON
After taking control, SE98 Capability = 0015h

INTERRUPT STATUS Register

- INTERRUPT STATUS Register (Reg# 02) is a Read Only Register that gives information about the cause of an Interrupt event
- Interrupt sources that are not masked will generate an Interrupt
- A read of the register will clear the Interrupt (INTx will go high again unless a new Interrupt occurs) and the bit(s) set to 1
- MYTEST and NMYTEST are not cleared when a read is performed



When set (= 1), the bits in that register have the following meanings:

- INTIN: An Interrupt has been detected in the Interrupt input pin (INT_IN)
- BUSINIT: The bus initialization/recovery sequence has been executed
- BUSOK: The bus was not idle when the switch occurred (PCA9541 did not perform the bus initialization/recovery sequence before the actual switch)
- BUSLOST: The other master took control of the bus
- MYTEST: The Interrupt line has been asserted with the Interrupt Line Test bit (MYTEST bit = 1 in the CONTROL Register)
- NMYTEST: The other master asserted the Interrupt line with the Interrupt Line Test bit (NMYTEST = 1 in the CONTROL Register)

INTERRUPT ENABLE Register

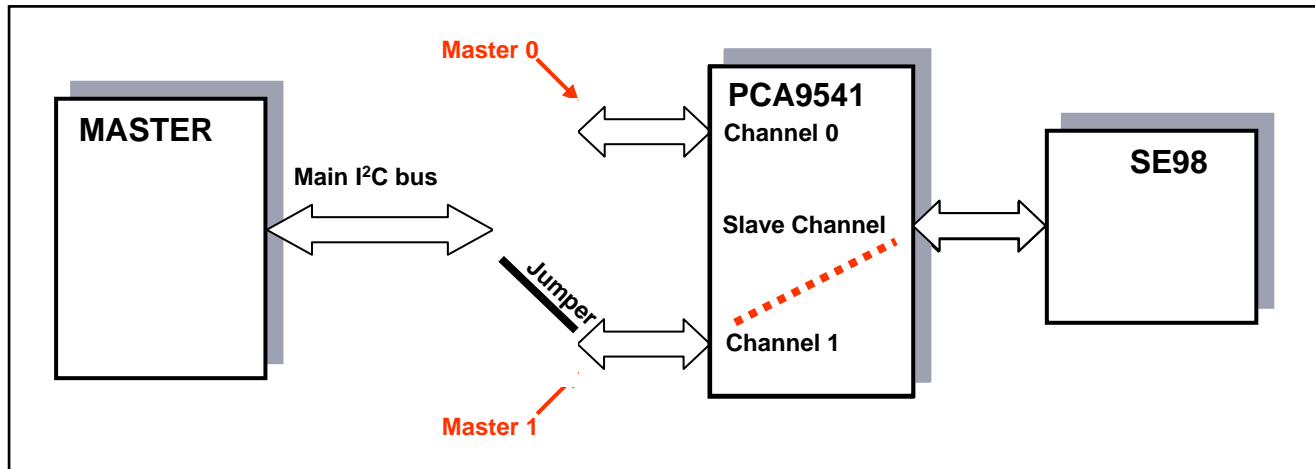
- INTERRUPT ENABLE Register (Reg# 00) allows to enable / disable the different Interrupt sources
- The Interrupt Lines Test cannot be masked
- At power up or after a RESET pulse, all the Interrupts are enabled

7	6	5	4	3	2	1	0
0	0	0	0	BUSLOSTMSK	BUSOKMSK	BUSINITMSK	INTINMSK

When set (= 1), the bits in that register have the following meanings:

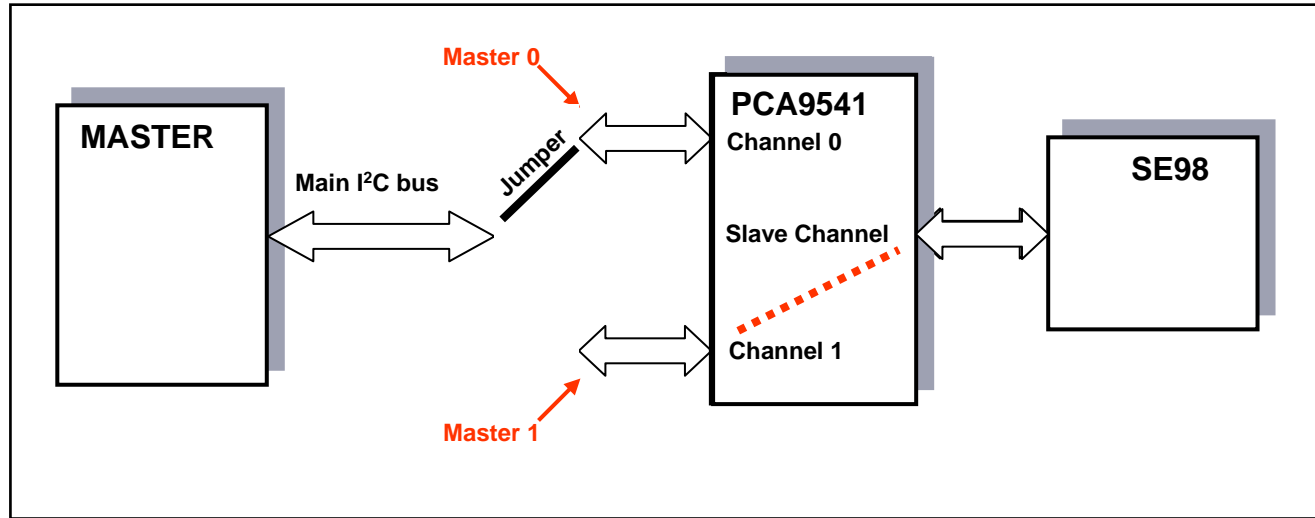
- INTINMSK: No Interrupt generated when INT_IN goes low
- BUSINITMSK: No Interrupt generated after the bus initialization/recovery sequence has been executed
- BUSOKMSK: No Interrupt generated after a switch in a non idle situation happened (PCA9541 did not perform the bus initialization/recovery sequence)
- BUSLOSTMSK: No Interrupt generated after the other master took control of the bus

Hands-On 2: Interrupts and Interrupt Control (1)



1. Remove the Jumper JP15 (INT_IN) and connect Jumpers SDA and SCL to Master 0 (Higher position)
2. Power sequence the board → Board OFF then ON
3. Open the PCA9541 GUI: Device → Master Selector (2-to-1 demux) → PCA9541
4. Read the INTERRUPT STATUS and the Interrupt ENABLE Registers. What do you read?
INT STATUS Reg# = 0x00 → No Interrupt ($\overline{\text{INT0}} = 1$ – LED OFF)
INT ENABLE Reg# = 0x00 → All the interrupt sources are enabled
5. Connect jumpers SDA and SCL to Master 1 (Lower position)
6. Program the CONTROL Register so that MASTER 1 takes control of the bus
What do you see?
LED INT0 is now ON → $\overline{\text{INT0}}$ is Low

Hands-On 2: Interrupts and Interrupt Control (2)



7. A) Connect jumpers SDA and SCL to Master 0 (Upper position)
B) Read the INTERRUPT STATUS Register
What do you see and read?

INT STATUS Reg# = 08h
LED INTO is now OFF

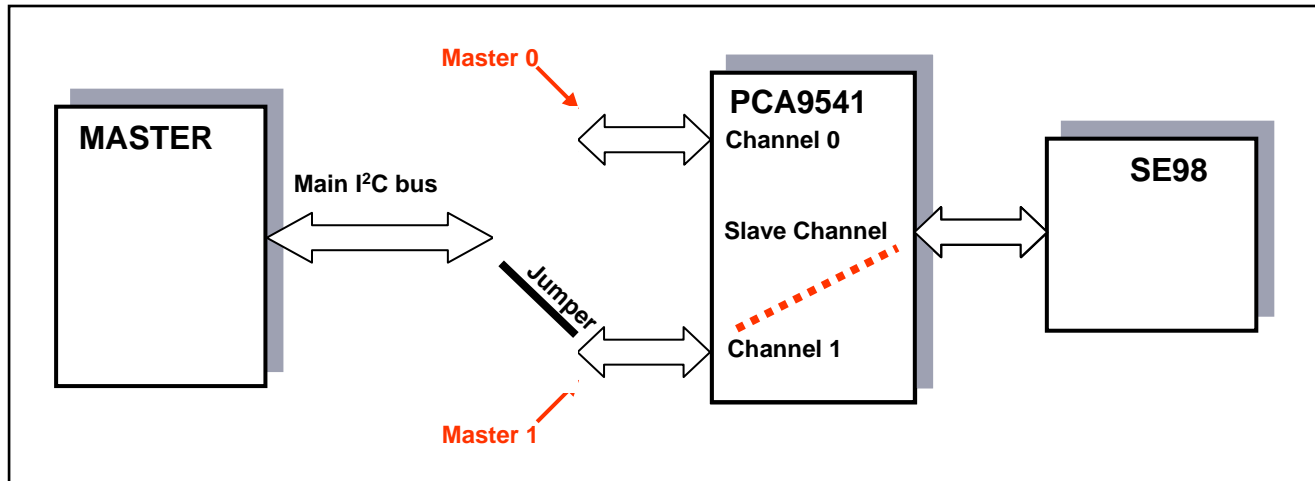
→ BUSLOST = 1 – f Master 0 lost control of the bus
→ Interrupt INTO is cleared after the read sequence

8. Read the INTERRUPT STATUS Register one more time. What do you read?

INT STATUS Reg# = 00h

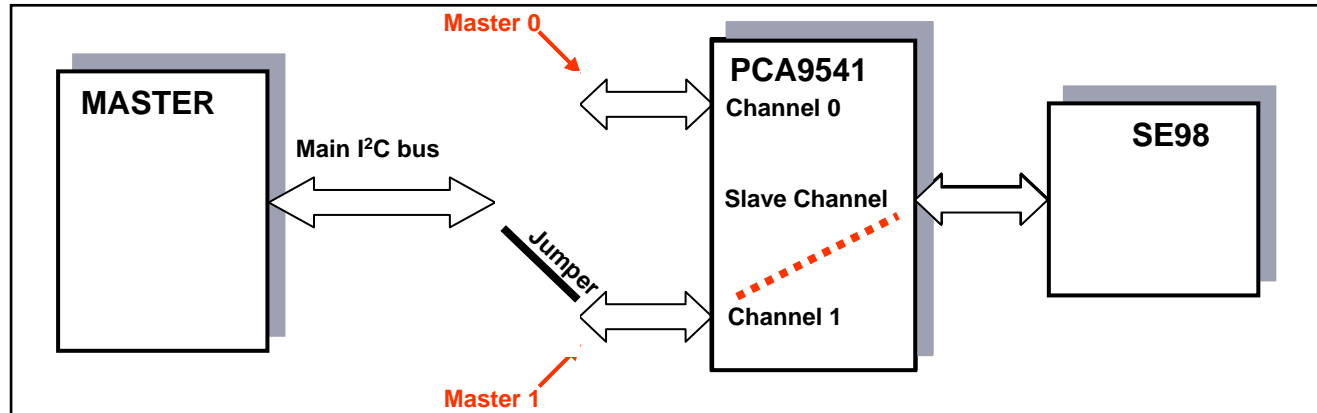
→ BUSLOST bit has been previously cleared

Hands-On 2: Interrupts and Interrupt Control (3)



9. a) Check BUSLOSTMSK in IE Register and Write the I²C message
b) Program the CONTROL Register so that MASTER 0 takes control of the bus
What do you see?
LED INT1 is now ON → $\overline{\text{INT1}}$ is Low
10. a) Connect jumpers SDA and SCL to Master 1 (Lower position)
b) Read the INTERRUPT STATUS Register
What do you see and read?
INT STATUS Reg# = 08h → BUSLOST = 1 – Master 1 lost control of the bus
LED INT1 is now OFF → Interrupt in INT1 is cleared after the read sequence
11. Program the CONTROL Register so that MASTER 1 controls the bus
What do you see?
NOTHING LED INT0 stays OFF because the Interrupt $\overline{\text{INT0}}$ that is asserted when the master lost control of the bus has been previously masked (BUSLOSTMSK = 1)

Hands-On 2: Interrupts and Interrupt Control (4)



12. a) Enable Jumper JP15 (INT_IN) → LED INT_IN should be ON
b) Read the INTERRUPT STATUS Register

What do you see and read?

INT STATUS Reg# = 01h

LED's INT0, INT1 are ON

→ $\overline{\text{INTIN}} = 1$ means that $\overline{\text{INT_IN}}$ pin is Low

→ $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are Low

13. a) Remove Jumper JP15
b) Check INTINMSK in IE Register and Write the I2C message
c) Enable Jumper JP15 (INT_IN) → LED INT_IN should be ON
d) Read the INTERRUPT STATUS Register

What do you see and read?

INT STATUS Reg# = 00h

LED INT1 is ON

→ $\overline{\text{INTIN}} = 0$

→ $\overline{\text{INT1}}$ is Low

INTINMSK bit has been masked for Master 1 so no Interrupt is asserted when $\overline{\text{INT_IN}}$ goes Low

Hands-On 3: Using the Expert Mode

- Write the following I²C sequences (Red) using the Expert Mode and execute it step by step

1. Connect Jumpers SDA and SCL to Master 0 (Higher position) and enable Jumper JP15 (INT_IN) → LED INT_IN should be ON
2. Power sequence the board → Board OFF then ON
3. Write to the CONTROL Register to test Master 0's Interrupt Line and give control to Master 1 (Hint: at power up, CONTROL REGISTER = 04h)
4. Read the CONTROL and the INTERRUPT STATUS Registers – Use the Notes field to explain the INTERRUPT STATUS Register read
5. Write to the INTERRUPT ENABLE Register to mask the interrupt generated by $\overline{\text{INT_IN}}$
6. Read the INTERRUPT STATUS Register – Use the Notes field to explain the INTERRUPT STATUS Register read

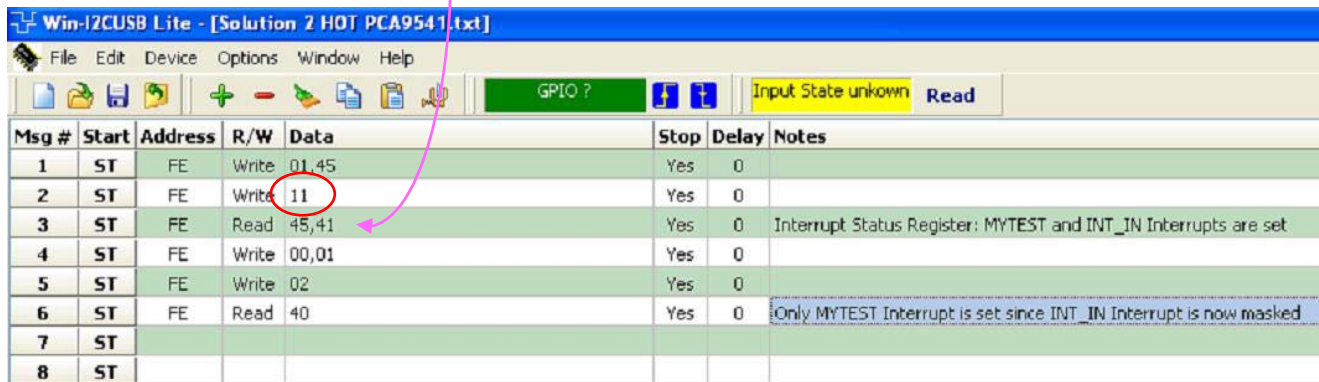
Hands-On 3: Using the Expert Mode – Solution

■ Solution 1

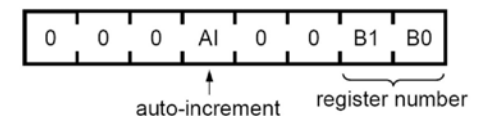


Msg #	Start	Address	R/W	Data	Stop	Delay	Notes
1	ST	FE	Write	01,45	Yes	0	
2	ST	FE	Write	01	Yes	0	
3	ST	FE	Read	45	Yes	0	
4	ST	FE	Write	02	Yes	0	
5	ST	FE	Read	41	Yes	0	Interrupt Status Register: MYTEST and INT_IN Interrupts are set
6	ST	FE	Write	00,01	Yes	0	
7	ST	FE	Write	02	Yes	0	
8	ST	FE	Read	40	Yes	0	Only MYTEST Interrupt is set since INT_IN Interrupt is now masked

■ Solution 2 - Optimized code using the Auto Increment feature



Msg #	Start	Address	R/W	Data	Stop	Delay	Notes
1	ST	FE	Write	01,45	Yes	0	
2	ST	FE	Write	11	Yes	0	
3	ST	FE	Read	45,41	Yes	0	Interrupt Status Register: MYTEST and INT_IN Interrupts are set
4	ST	FE	Write	00,01	Yes	0	
5	ST	FE	Write	02	Yes	0	
6	ST	FE	Read	40	Yes	0	Only MYTEST Interrupt is set since INT_IN Interrupt is now masked
7	ST						
8	ST						



Allows Read & Write of several consecutive registers within the same I²C command

