Pic16F876 used as an 8+8 IO expander & motor controller working as I2C slave

By a.ratti

This I2C 8+8 IOs expander has been built using a pic16F876. It works as a slave device with his own address, responding to several commands.

The pinout of the slave system can be found in schematic # 1. There are 8 outputs that can be controlled with one byte (where 0 = all outs off and 255 = all outs on).
The 8 inputs can be read at any time, the byte returned represent the inputs status. The inputs are all internally pulled up and are activated when external switches pull them down to zero. So all open inputs will yield 0 as an answer, while all inputs pulled down to zero, will yield 255 as an answer

Without any modification, outputs 1 & 2 can be used to drive stepper motor # 1, where output # 1 is the clock (steps) and output # 2 is the direction.

The same apply for outputs 3 & 4 that can be used to drive stepper motor # 2, where output # 3 is the clock (steps) and output # 4 is the direction.

Naturally to control stepper motors, you need suitable optoisolated bipolar chopper drivers.

If outputs are used to drive stepper motors then 4 inputs will serve the motors in the following way:

Input 1 is for homing motor # 1 (when homing is activated it will stop when input # 1 goes high)
Input 2 is for homing motor # 2 (when homing is activated it will stop when input # 2 goes high)
Input 3 is for synchro start for motor # 1 (keeping low the input motor is hold till input goes high)
Input 4 is for synchro start for motor # 2 (keeping low the input motor is hold till input goes high)

If the motor capabilities of the expander are used, then the busy line will result extremely useful, in timing the master, because a motor command can take several seconds to be executed. Remember that during the motor travel no other commands can be sent to the device. So if busy is high no command can be sent when busy line goes low device is ready.

In addition the device has two pwm ports that can be controlled both in frequency and duty cycle

To write to or read from slave, Master must use the following codes

WriteSlave:
```
I2CWrite SDA,SCL,Address,[str TxBuffer\3], TimeOut
 pause 5
 Return
```

```
ReadSlave:
I2CRead SDA,SCL,Addess,[str TxBuffer\3], TimeOut
 pause 5
 Return
```

Where address is the number of the device 2; 3 or 4 (see software)

TxBuffer is an array of 3 bytes for command code and value, so organized:

TxBuffer[0] = command code

TxBuffer[1] = Low byte and TxBuffer[2] = High byte

LIST OF WRITING COMMAND CODES:

Command code	Description
20	Write to eeprom the speed value for motor # 1
21	Write to eeprom the speed value for homing motor # 1
22	Write to eeprom the ramp value for motor # 1
25	write to eeprom the speed value for motor # 2
26	Write to eeprom the speed value for homing motor # 2
27	Write to eeprom the ramp value for motor # 2
30	Write to eeprom the frequency for PWM #1 & PWM #2
50	Write the system configuration byte (see note 1)

LIST OF EXECUTING COMMAND CODES

Command code	Description
100	Set outputs as per low byte (byte=0 all outs off - byte=255 all outs on)
110	Start motor # 1 in CW direction for the number of steps given with low & high byte
113	Start the homing of motor # 1 in CW direction.(see note 2)
115	Start motor # 1 in CCW direction for the number of steps given with low & high byte
118	Start the homing of motor # 1 in CCW direction.
120	Start motor # 2 in CW direction for the number of steps given with low & high byte
123	Start the homing of motor # 2 in CW direction.
125	Start motor # 2 in CCW direction for the number of steps given with low & high byte
128	Start the homing of motor # 2 in CCW direction.
130	Syncro start of motor # 1 in CW direction. (see note 3)
135	Syncro start of motor # 1 in CCW direction.
140	Syncro start of motor # 2 in CW direction.
145	Syncro start of motor # 2 in CCW direction.
170	Duty cycle of PWM # 1
175	Duty cycle of PWM # 2

LIST OF READING COMMAND CODES

Command code	Description (Reading commands)
200	Return status of inputs and outputs on low and high byte
210	Return speed value of motor # 1
212	Return homing speed of motor # 1
215	Return ramp value of motor # 1
220	Return speed value of motor # 2
222	Return homing speed of motor # 2
225	Return ramp value of motor # 2
230	Return frequency of PWM # 1 & PWM # 2
250	Return configuration byte

EXAMPLES:

TURNS FIRST 4 OUTPUTS HIGH:

Command : TxBuffer[0]=100 TxBuffer[1]=15 TxBuffer[2]=0

TURN OUTPUT # 8 HIGH ALL THE OTHER LOWS

Command : TxBuffer[0]=100 TxBuffer[1]=128 TxBuffer[2]=0

TURNS ALL OUTPUTS HIGH:

Command : TxBuffer[0]=100 TxBuffer[1]=255 TxBuffer[2]=0

TURNS ALL OUTPUTS LOW:

Command : TxBuffer[0]=100 TxBuffer[1]=0 TxBuffer[2]=0

ROTATE MOTOR # 1 10 TURNS CW (800 steps per turn)

Command : TxBuffer[0]=110 TxBuffer[1]=64 TxBuffer[2]=31

CHANGE ACCELERATION RAMP TO MOTOR # 1 (new ramp = 800 steps)

Command : TxBuffer[0]=110 TxBuffer[1]=32 TxBuffer[2]=3

CHANGE SPEED TO MOTOR # 1 (speed= 300 units of delay)

Command : TxBuffer[0]=20 TxBuffer[1]=44 TxBuffer[2]=1

READ STATUS OF INPUTS

Command : TxBuffer[0]=200 TxBuffer[1]=0 TxBuffer[2]=0

immediatly after inwoke a I2C reading and it will return:

TxBuffer[0]= command code

TxBuffer[1]= Outputs status

TxBuffer[1]= Inputs status

All the reading command must be sent with a I2CWRITE command, followed immediatly by a I2CREAD command wich will return the reading.

MOTOR UNITS.

The units used for moving the motors are:

STEPS FOR TRAVEL (rotation) the number of steps per turns will depend from motor and driver.

STEPS FOR RAMP (Acceleration) ¾ of motor turn is a good ramp value

UNITS OF DELAY FOR SPEED CONTROL (one unit = 3 microseconds)
This delay will control the 50% duty cycle clock . So shorter is the delay faster will be the clock, longer is the delay slower will be the clock . (one pulse = one motor step)

Unit of delay = 33333/steps per seconds

If I need to rotate the motor at 1600 steps per second then : 333333/2000=208.3

Command : TxBuffer[0]=20 TxBuffer[1]=208 TxBuffer[2]=0

Now a 800 steps per turn motor will make two turns per second or (120 turns per min)

CONFIGURATION

This is a byte used by the system to avoid corruption of the motors position when an output command is sent.

When the configuration byte is set to 2, then any outputs writing, will skip outputs 1 & 2 .
So motor # 1 will not be unintentionally moved by any wrong output command.

When the configuration byte is set to 4, then any output writing, will skip outputs 1; 2; 3 & 4 protecting positon of both motors.

If homing command is executed then the config byte will be set automatically otherwise it must be set via command code:

Command : TxBuffer[0]=50 TxBuffer[1]=2 TxBuffer[2]=0 (for motor # 1 only)

Command : TxBuffer[0]=50 TxBuffer[1]=4 TxBuffer[2]=0 (for both motors)

HOMING

There are two commands for homing, CW & CCW. Homing should always travel towards the zero position switch, which when activated will immediatly stop the motor. This position is known as HOME POSITION. It is good rule to approch zero point, traveling at a slow speed.
If you activate the homing in the wrong direction the system will not stop till you activate the zero switch manually.

PWM SETTING

PWM frequency can be set for both channels from a minimum of 1250 Hz up to 32000 Hz maximum.
Duty cycle can be set for each channel individually value from 0 to 255

Command : TxBuffer[0]=30 TxBuffer[1]=184 TxBuffer[2]=11

(Set PWM frequency to 3 Khz)

Command : TxBuffer[0]=170 TxBuffer[1]=128 TxBuffer[2]=0

(Set duty cycle of PWM # 1 to 128)

Command : TxBuffer[0]=175 TxBuffer[1]=64 TxBuffer[2]=0

(Set duty cycle of PWM # 2 to 64)

SOFTWARE

There are three different hex files. They differ for the different device address in the following way:

I2C_Expander_2 Device address = 2

I2C_Expander_3 Device address = 3

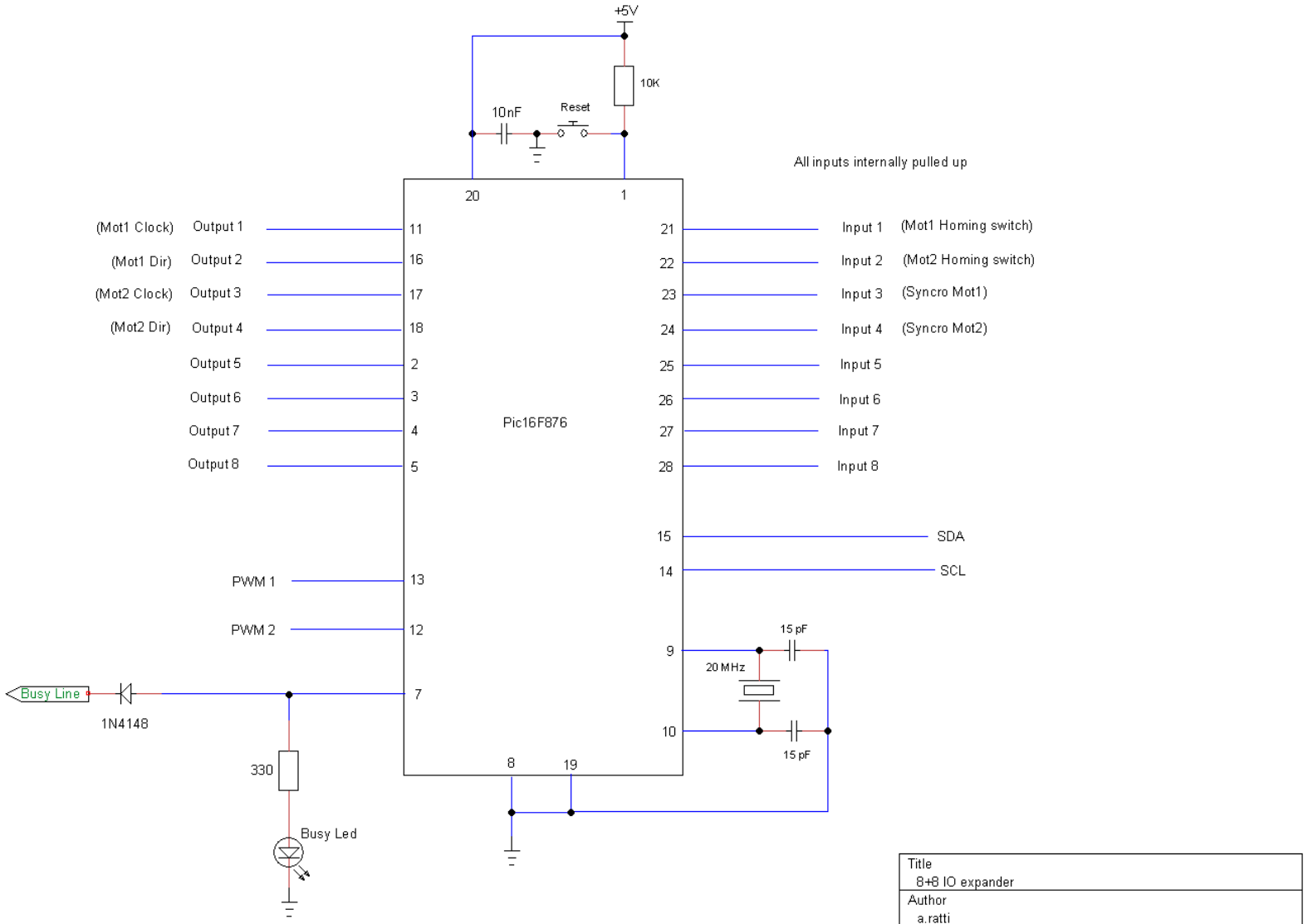
I2C_Expander_4 Device address = 4

With these three files you can put three devices on the same I2C bus with 24 + 24 IOs plus 6 PWM channels available.

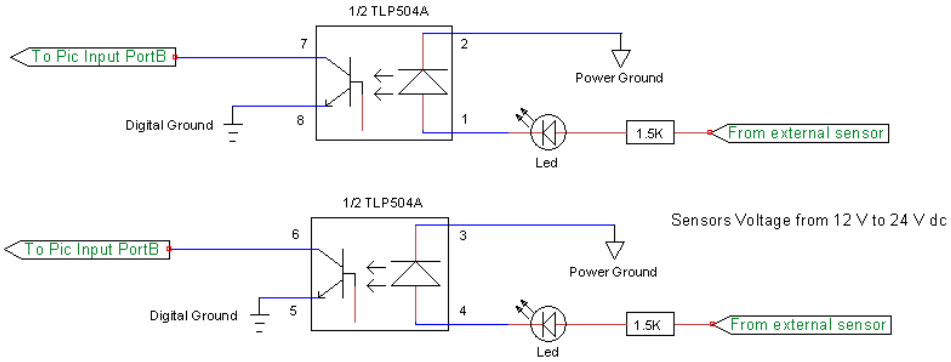
Remember to label the device with the proper address, once programmed, to avoid confusion.

I hope you could find this device useful for your future projects.

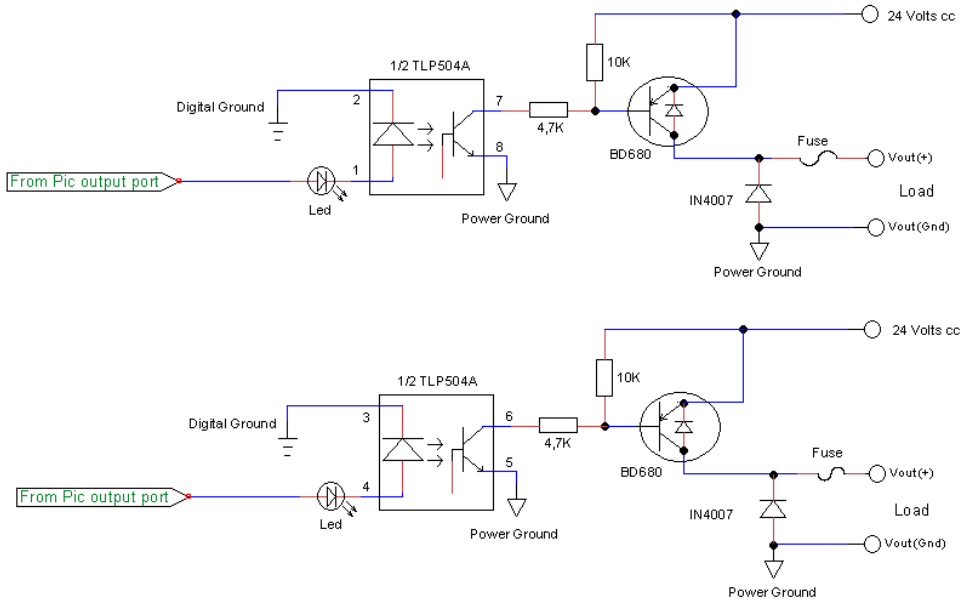
a.ratti



Title 8+8 IO expander		
Author a.ratti		
File I Settings\Alberto\Desktop\8+8_IO_Expander.dsn		Document 1
Revision 1.0	Date 15-05-2006	Sheets 1 of 1

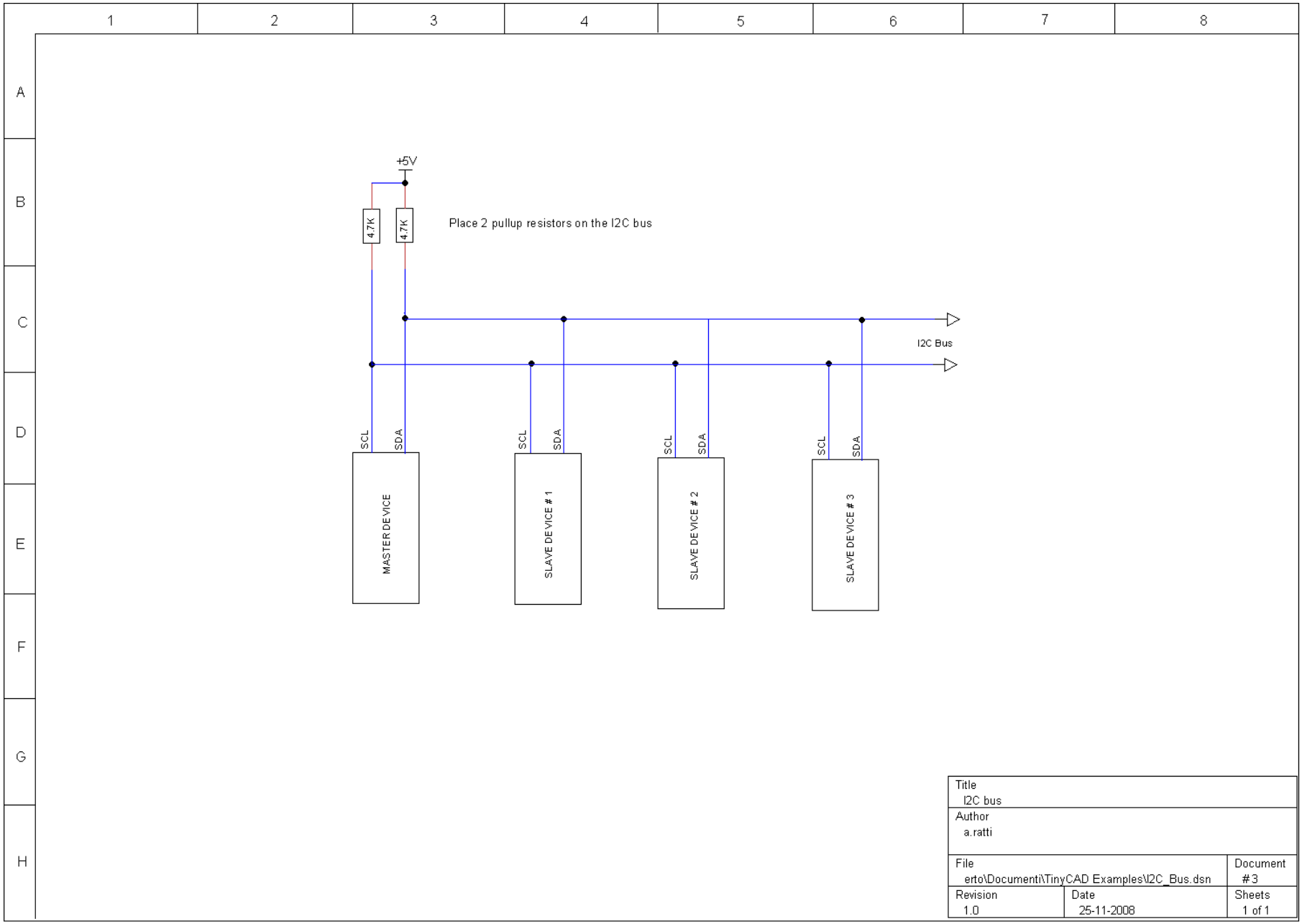


INPUTS SECTION



OUTPUTS SECTION

Title Optoisolate Inputs & Outputs		
Author a.ratti		
File ettings\Alberto\Desktop\Optoisolate_Inputs_a.dsn		Document #2
Revision 1.0	Date 01-01-2008	Sheets 1


1
2
3
4
5
6
7
8
A
B
C
D
E
F
G
H
+5V
4.7K
4.7K
Place 2 pullup resistors on the I2C bus
I2C Bus
SCL
SDA
MASTER DEVICE
SCL
SDA
SLAVE DEVICE # 1
SCL
SDA
SLAVE DEVICE # 2
SCL
SDA
SLAVE DEVICE # 3
Title
I2C bus
Author
a.ratti
File
erto\Documenti\TinyCAD Examples\I2C_Bus.dsn
Document
#3
Revision
1.0
Date
25-11-2008
Sheets
1 of 1