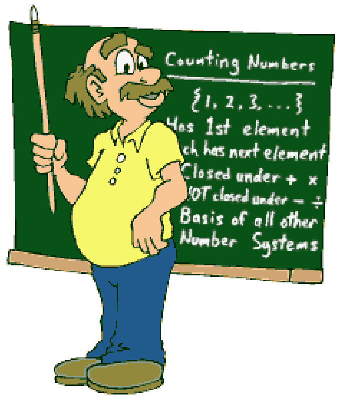
Ingeniørhøjskolen i Århus

iha.dk

AMS

Applied Microcontroller Systems

Lesson 3: The I^{2}C bus

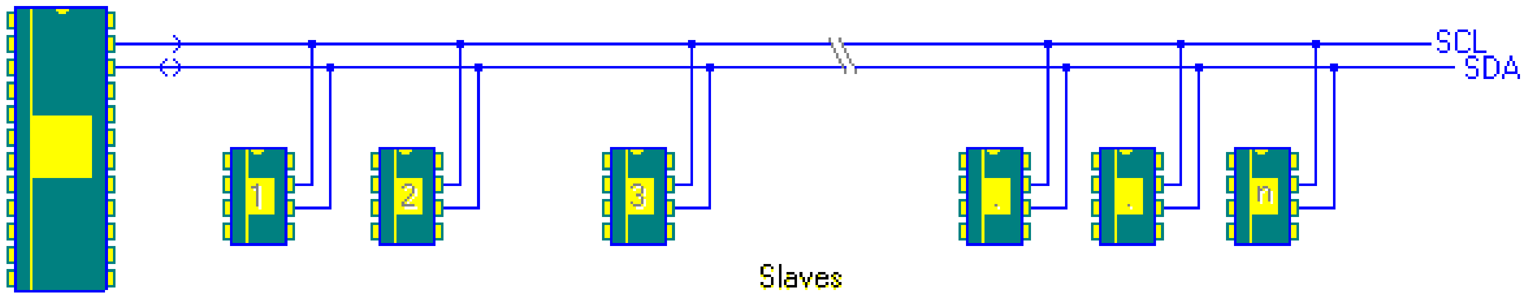


Version: February 2013, Asbjørn Baagø

I^2C (IIC)

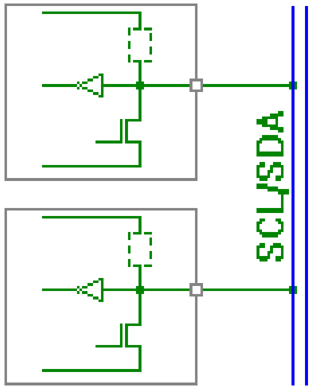
- Inter IC bus (Introduced by Philips)
- Aka “two-wire interface”
- Different versions supporting 100 kHz - 5 MHz
- Limited range (often locally at PCB)
- Built-in addressing (minor HW demands)
- Simple protocol built in (ACK / NACK)
- Real time qualified (response times can be calculated)

I^{2}C topology



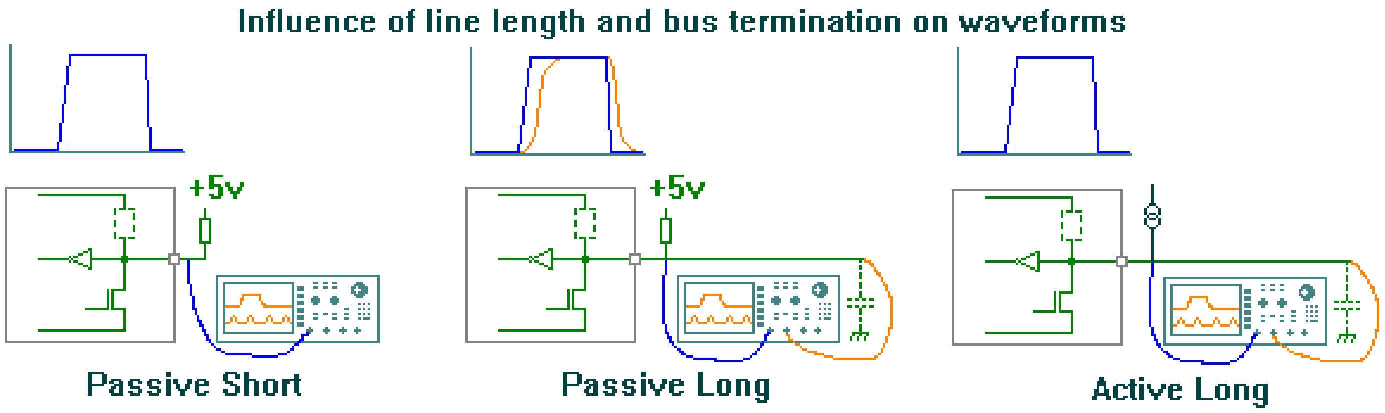
Each node has its own unique address (or ID).
By sending a START command, each node is able to become the bus master.

I^{2}C bus hardware

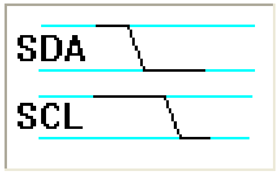


SCL and SDA are bi-directional.

Open drain => External pull-up necessary!

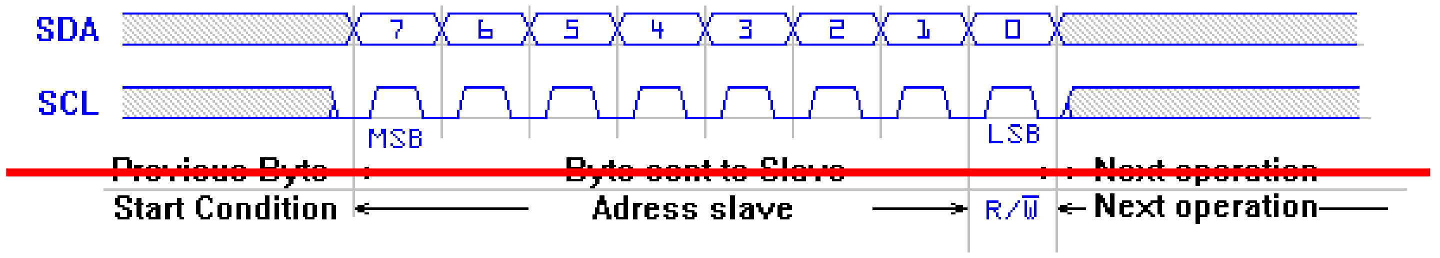


I^{2}C Master: START + address

Start		The chip issuing the Start condition first pulls the SDA (data) line low, and next pulls the SCL (clock) line low.

All other nodes now enter a state of ”listening”.

Then the master sends the address of the slave, it wants to communicate with:



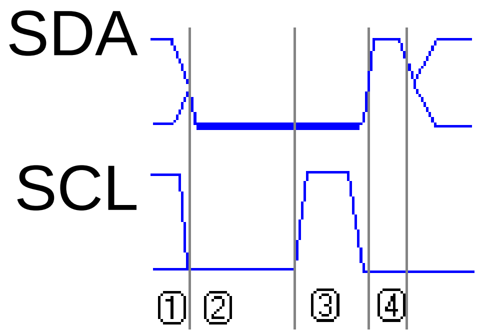
The address is 7 bits.
The last bit (R/W) informs the slave, whether the master wants to read or write.

I^{2}C Slave: ACK

After having received the START command and the address, all slaves compare the address with their own address:

* If no address match, the STOP command is awaited.

* If the address matches, the slave sends an ACK:

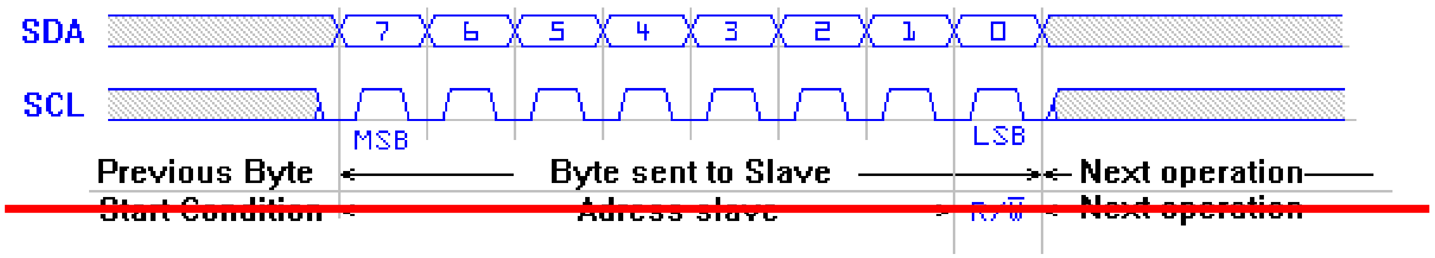


The slave holds SDA low and generates an SCL -pulse.

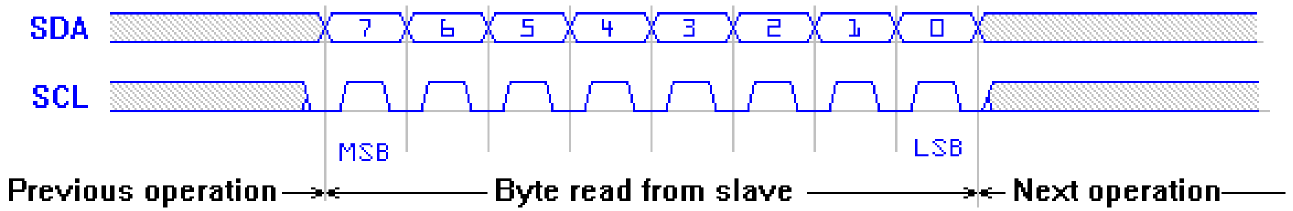
- Subsequently the master regains bus control -

I^{2}C master: Sends / receives data

1. Master sends data:



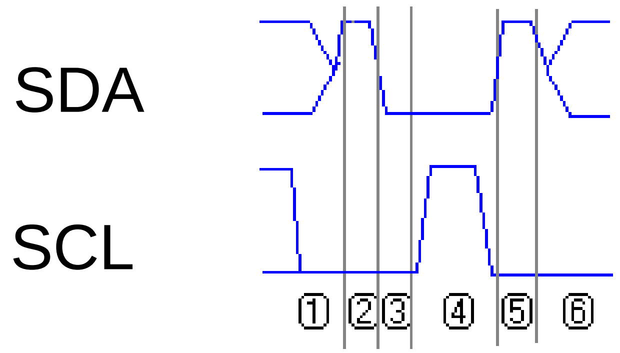
2. Master receives data:



During reception (2) the slave controls SDA, but the master controls SCL.
The slave is only allowed to change SDA while SCL is low.

I^{2}C Master: ACK after each byte received

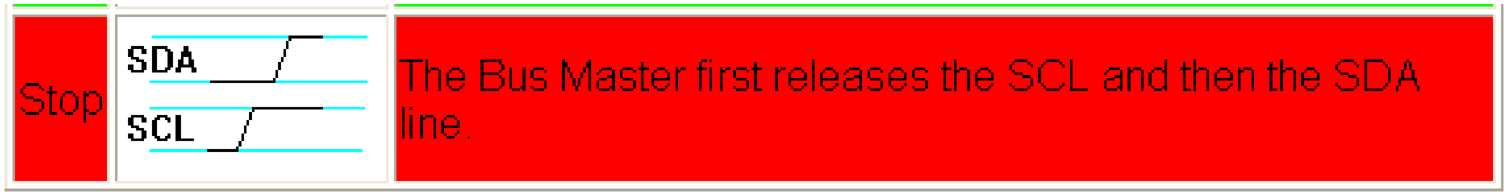
Following each byte received from a slave, the master MUST send an ACK (except after the last byte).



Master sets SDA low and generates a clock pulse. Then the slave regains control over SDA.

Following the last received byte, the master sends the STOP command (freeing the bus again).

Master: STOP


Stop

The Bus Master first releases the SCL and then the SDA line.

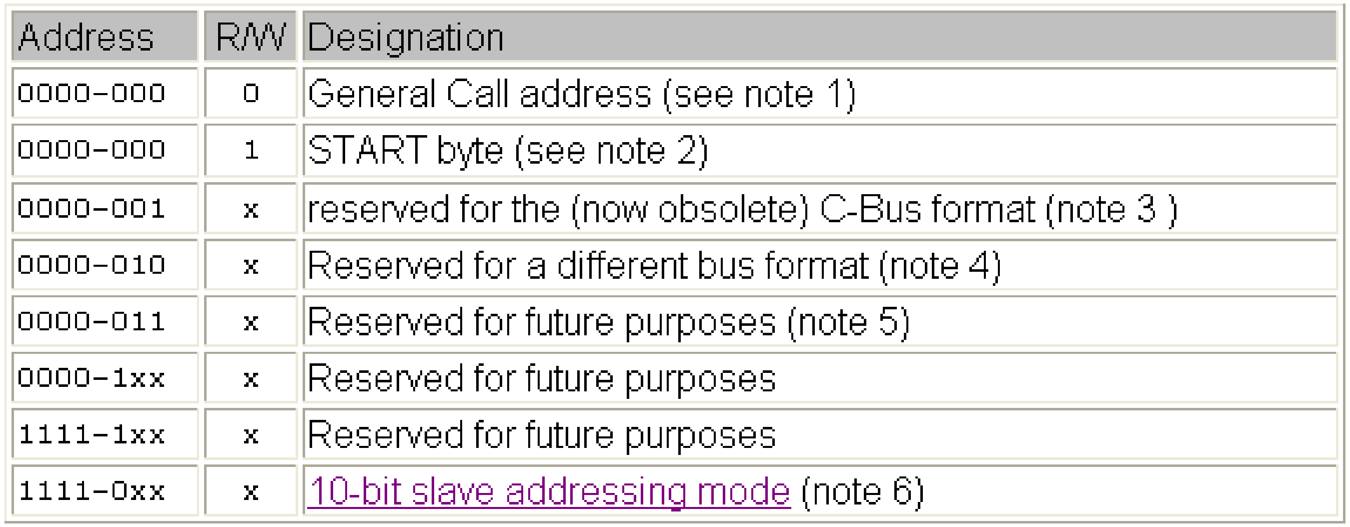
Sent by the master, when it has finished sending /receiving data.

Received by all slaves, now knowing the bus is again free.

STOP can be sent at any time (even in the middle of a data transmission).

In all cases STOP means ”END of transmission”.

Reserved addresses



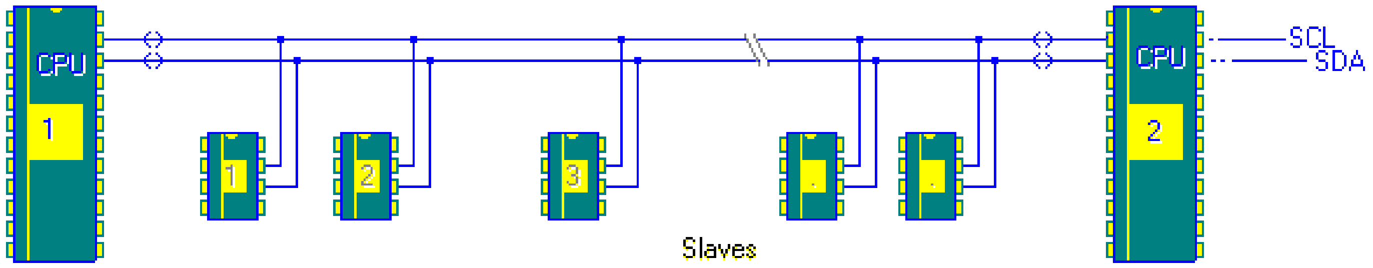
Address	R/W	Designation
0000-000	0	General Call address (see note 1)
0000-000	1	START byte (see note 2)
0000-001	x	reserved for the (now obsolete) C-Bus format (note 3)
0000-010	x	Reserved for a different bus format (note 4)
0000-011	x	Reserved for future purposes (note 5)
0000-1xx	x	Reserved for future purposes
1111-1xx	x	Reserved for future purposes
1111-0xx	x	10-bit slave addressing mode (note 6)

10-bit addressing is possible (new standard).

In this case 2 bytes are sent as address:
11110AAx + AAAAAAAA.

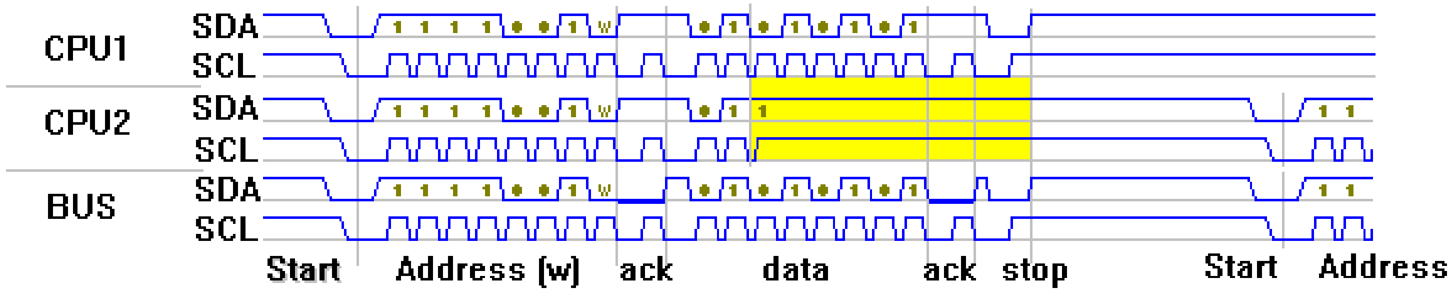
(7-bit nodes ignores 10-bit addressing).

I^{2}C Bus Arbitration



If a master is unable to set an output high (it always checks for this), it looses the bus control.

If 2 masters start transmission simultaneously, the following happens:



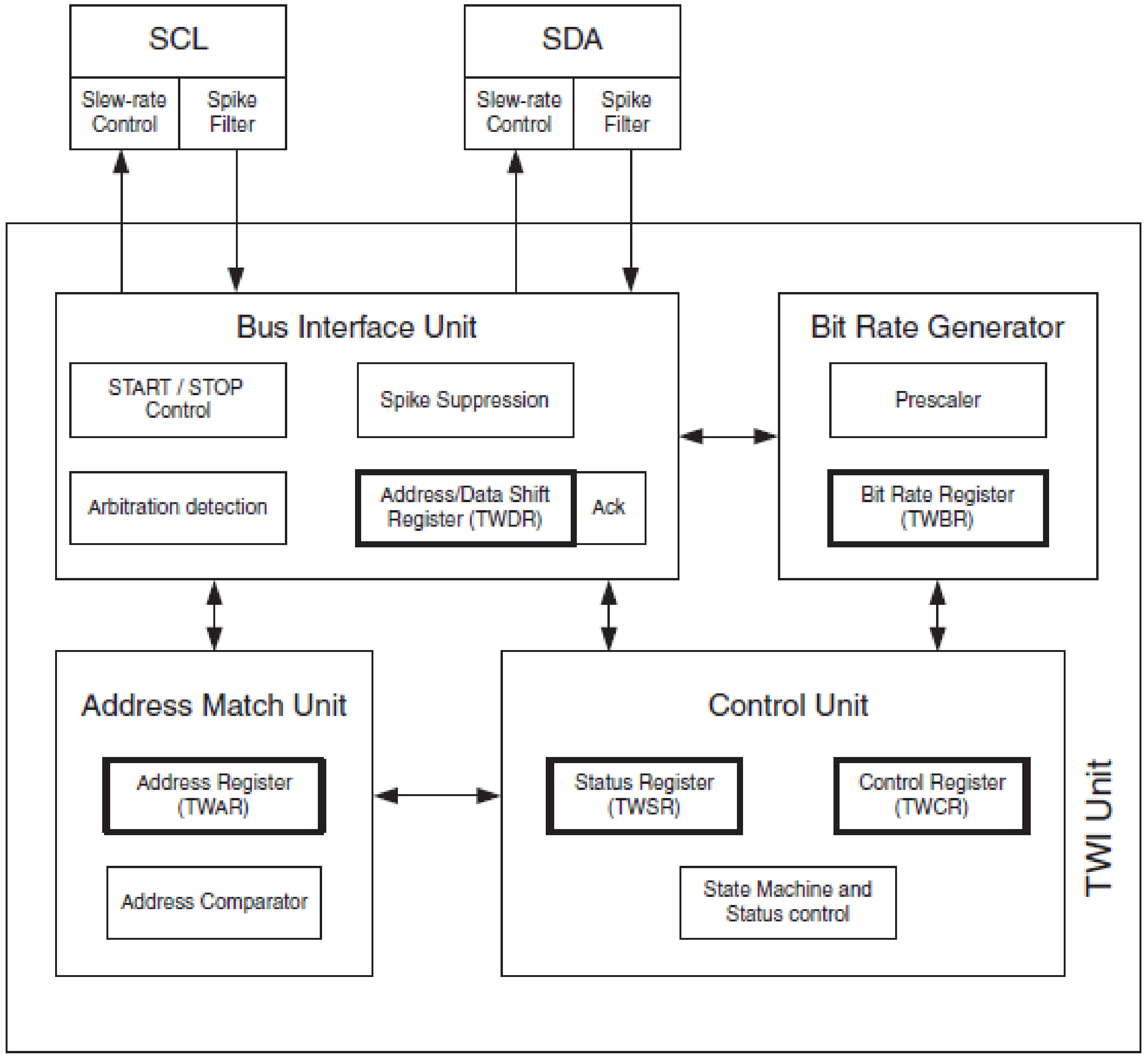
Until the yellow marking, both masters think, they own the bus.
Then CPU2 surrenders (until STOP).

Mega32: I^2C interface

- Mega32 has HARDWARE for I^2C interface.
  This is called the ”Two Wire Serial Interface”.
  Dedicated pins for SDA and SCL.

- Registers:
  TWI Bit Rate Register – TWBR
  TWI Control Register – TWCR
  TWI Status Register – TWSR
  TWI Data Register – TWDR
  TWI (Slave) Address Register – TWAR

The Mega32 I^{2}C interface (=TWI)



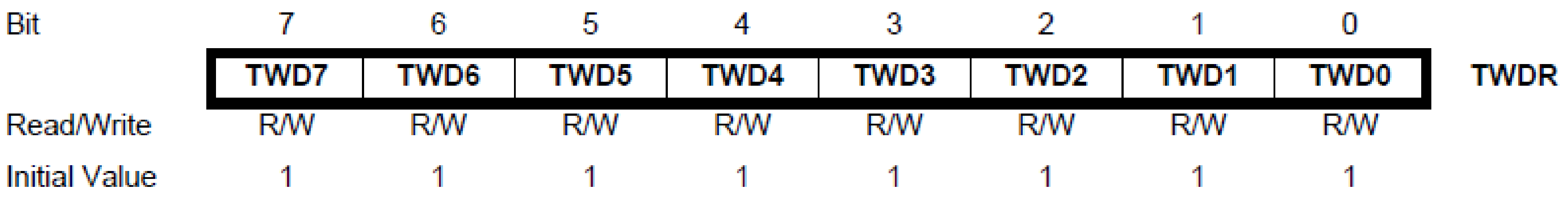
TWBR: TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

$$SCL\ frequency = \frac{CPU\ Clock\ frequency}{16 + 2(TWBR) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

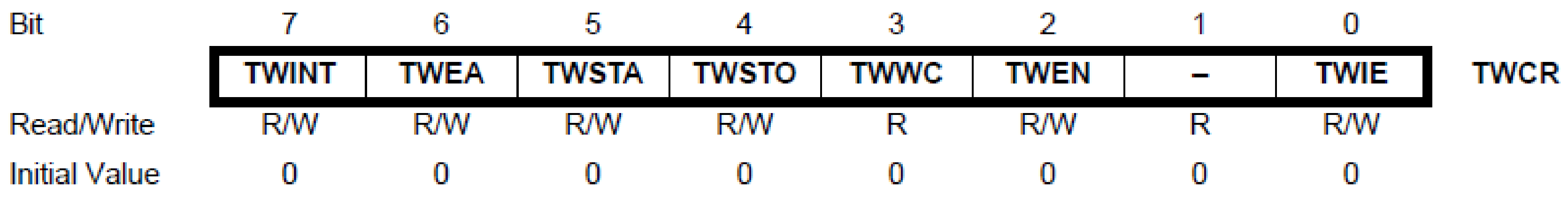
TWDR: TWI Data Register



Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

- When transmitting: Contains the next byte to be sent
- When receiving: Contains the last byte received

TWCR: TWI Control Register



Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- TWINT: Set to 1 when ANY bus operation ends.
  If TWI interrupt enabled, we get an interrupt. Must be cleared (by writing a 1) before next bus operation can start.
- TWEA: Set this bit (to 1) if ACK is to be generated on next reception.
- TWSTA: Write to 1 (together with TWINT and TWEN) to generate a START condition.
- TWSTO: Write to 1 (together with TWINT and TWEN) to generate a STOP. A STOP being sent will NOT set TWINT.
- TWWC: Set to 1, if you do a write collision (not important).
- TWEN: Must be set to 1 when you write the TWCR.
- TWIE: TWI interrupt enable (write to 1 to enable).

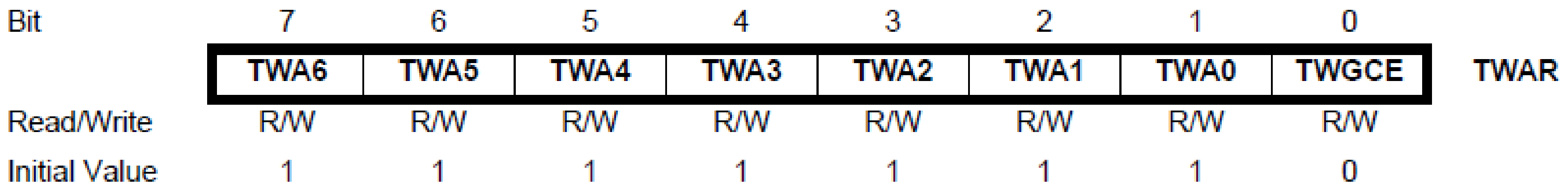
TWSR: TWI Status Register

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- Bits 7 – 3: Status code telling the status of the I2C bus. Can be used for verifying that bus operations performed all right.
  The status codes can be found in the Mega32 manual.
- Bits 1 – 0: TWI clock prescaler (see below).

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

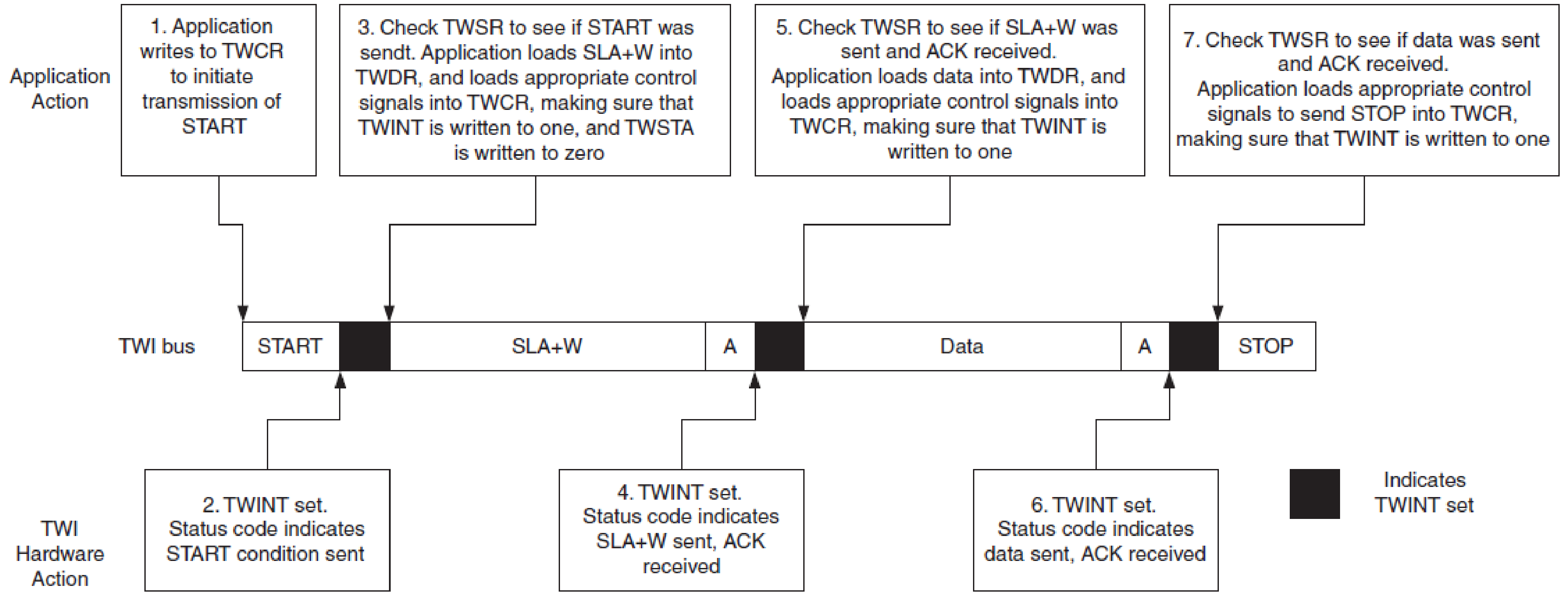
TWAR: TWI (slave) Address Register



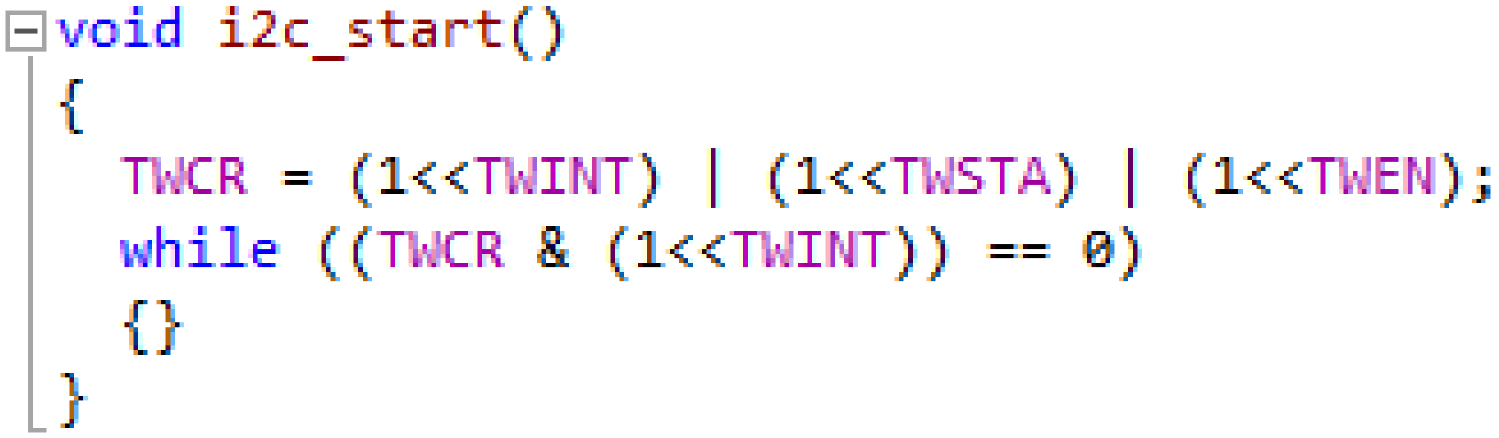
Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

- Can be used in SLAVE mode for automatic detecting the received address (compares ID).

Typical TWI transmission

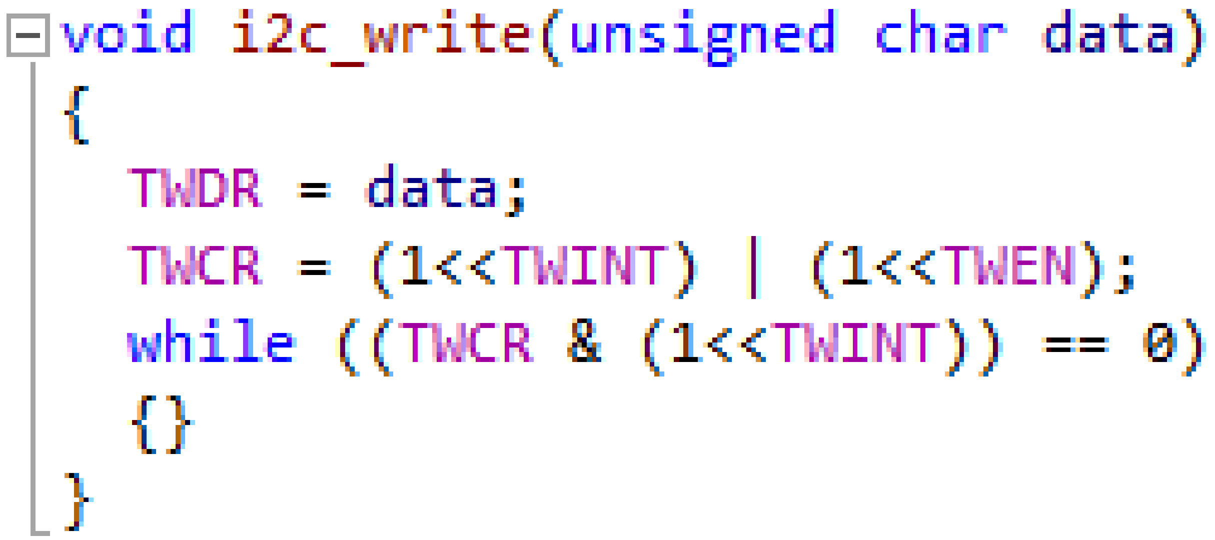

Application Action
1. Application writes to TWCR to initiate transmission of START
3. Check TWSR to see if START was sendt. Application loads SLA+W into TWDR, and loads appropriate control signals into TWCR, making sure that TWINT is written to one, and TWSTA is written to zero
5. Check TWSR to see if SLA+W was sent and ACK received. Application loads data into TWDR, and loads appropriate control signals into TWCR, making sure that TWINT is written to one
7. Check TWSR to see if data was sent and ACK received. Application loads appropriate control signals to send STOP into TWCR, making sure that TWINT is written to one
TWI bus
START
SLA+W
A
Data
A
STOP
TWI Hardware Action
2. TWINT set. Status code indicates START condition sent
4. TWINT set. Status code indicates SLA+W sent, ACK received
6. TWINT set. Status code indicates data sent, ACK received
Indicates TWINT set

i2c_start( )



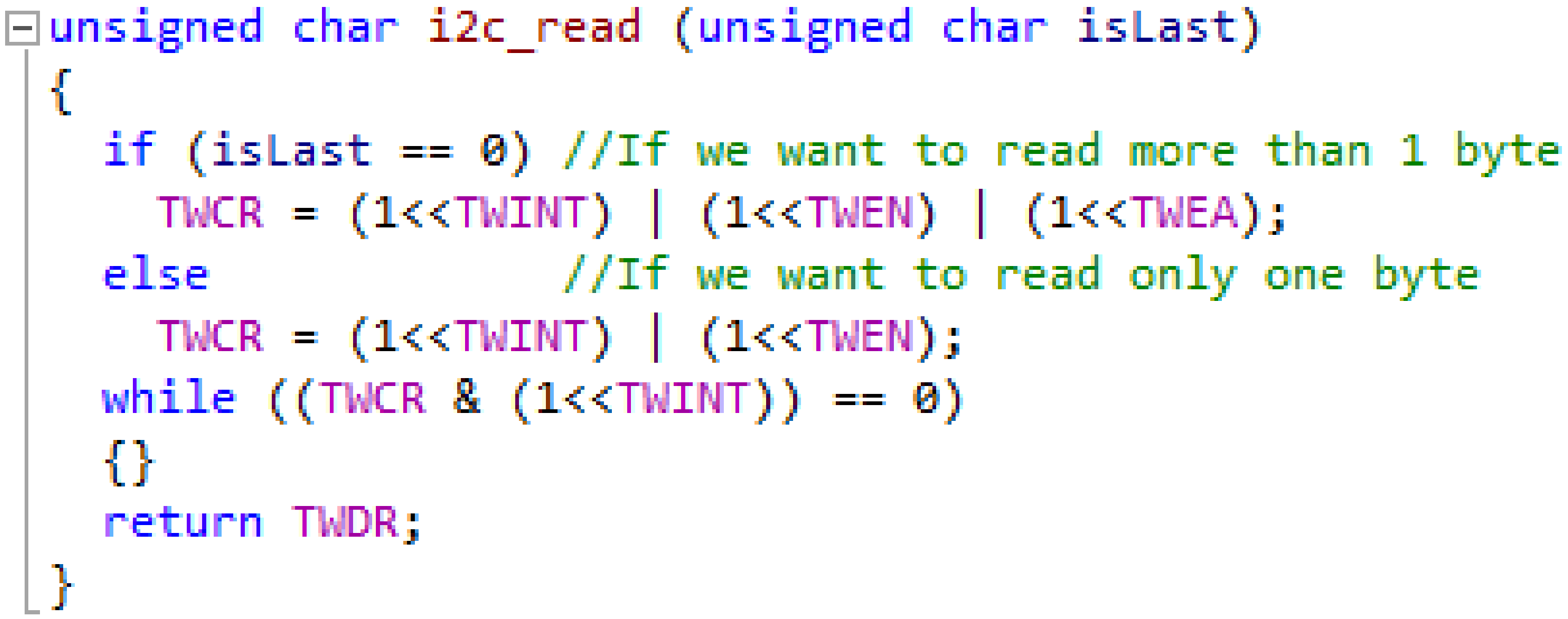
```
void i2c_start()
{
   TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
   while ((TWCR & (1<<TWINT)) == 0)
   {}
}
```

i2c_write( )



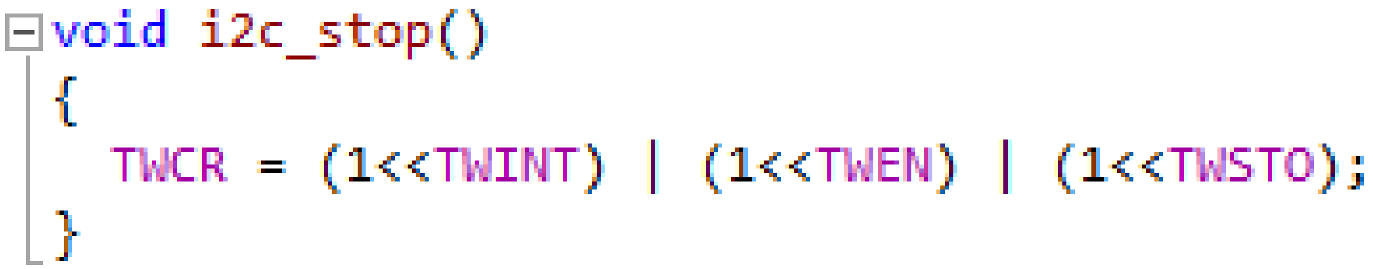
```
void i2c_write(unsigned char data)
{
    TWDR = data;
    TWCR = (1<<TWINT) | (1<<TWEN);
    while ((TWCR & (1<<TWINT)) == 0)
    {}
}
```

i2c_read( )



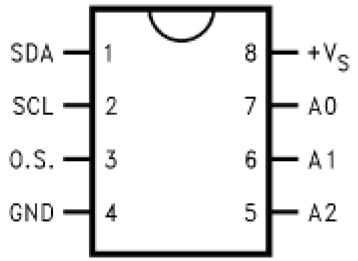
```
unsigned char i2c_read (unsigned char isLast)
{
  if (isLast == 0) //If we want to read more than 1 byte
    TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
  else             //If we want to read only one byte
    TWCR = (1<<TWINT) | (1<<TWEN);
  while ((TWCR & (1<<TWINT)) == 0)
  {}
  return TWDR;
}
```

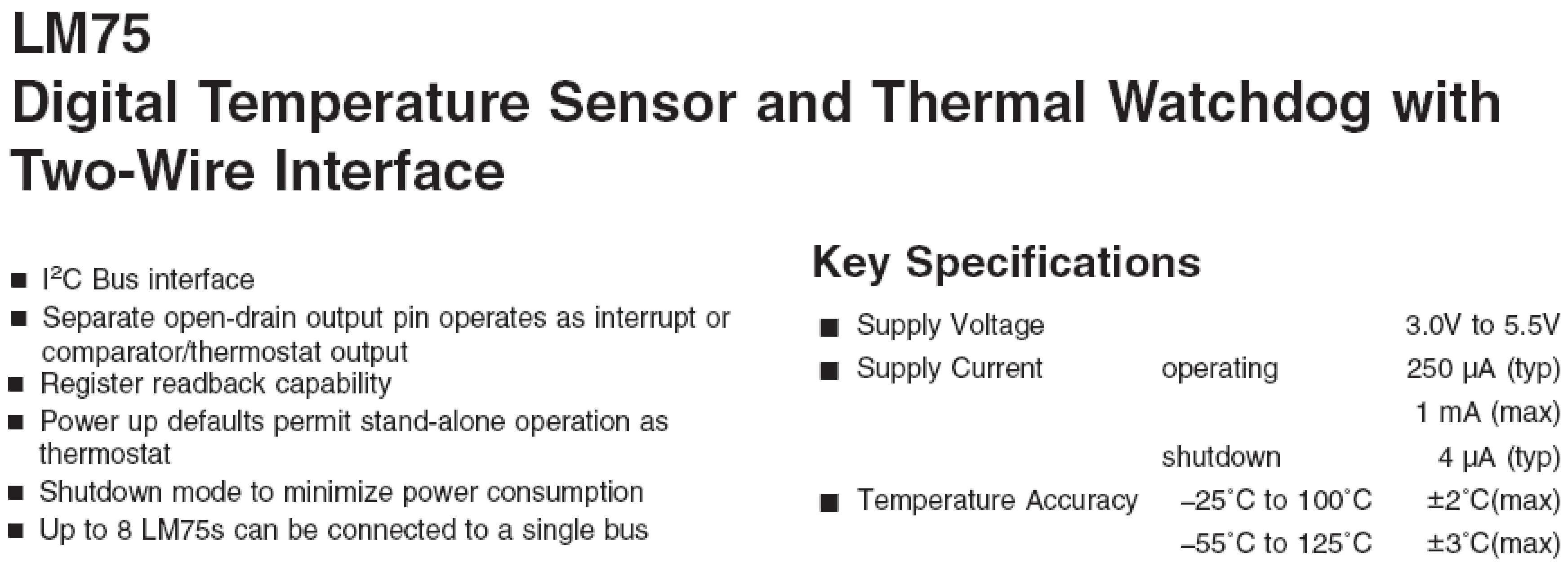
i2c_stop( )



```c
void i2c_stop()
{
   TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
}
```

I2C example: LM75





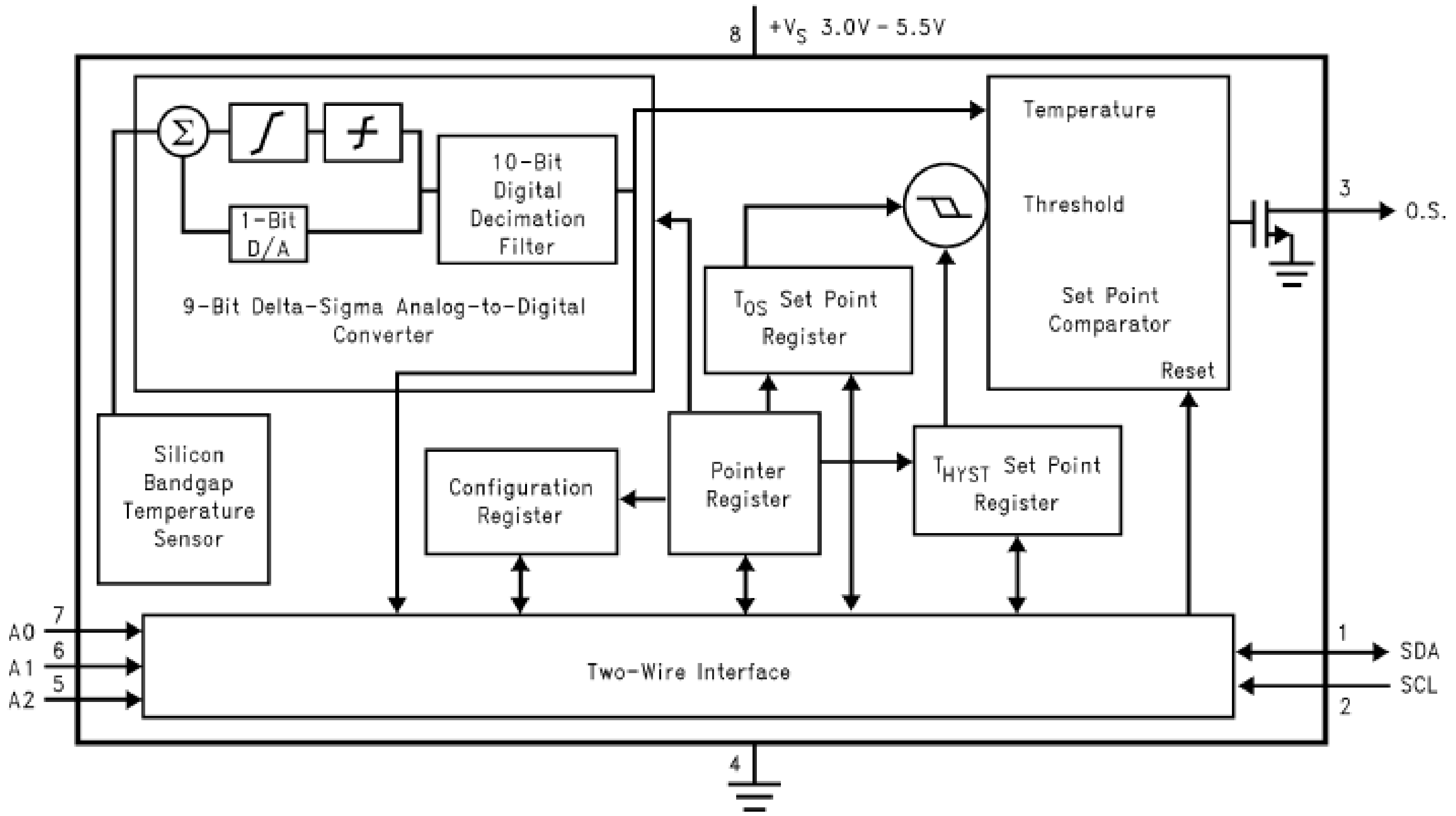
LM75
Digital Temperature Sensor and Thermal Watchdog with Two-Wire Interface

- I²C Bus interface
- Separate open-drain output pin operates as interrupt or comparator/thermostat output
- Register readback capability
- Power up defaults permit stand-alone operation as thermostat
- Shutdown mode to minimize power consumption
- Up to 8 LM75s can be connected to a single bus

Key Specifications

■ Supply Voltage		3.0V to 5.5V
■ Supply Current	operating	250 µA (typ)
		1 mA (max)
	shutdown	4 µA (typ)
■ Temperature Accuracy	–25˚C to 100˚C	±2˚C(max)
	–55˚C to 125˚C	±3˚C(max)

LM75: Block Diagram



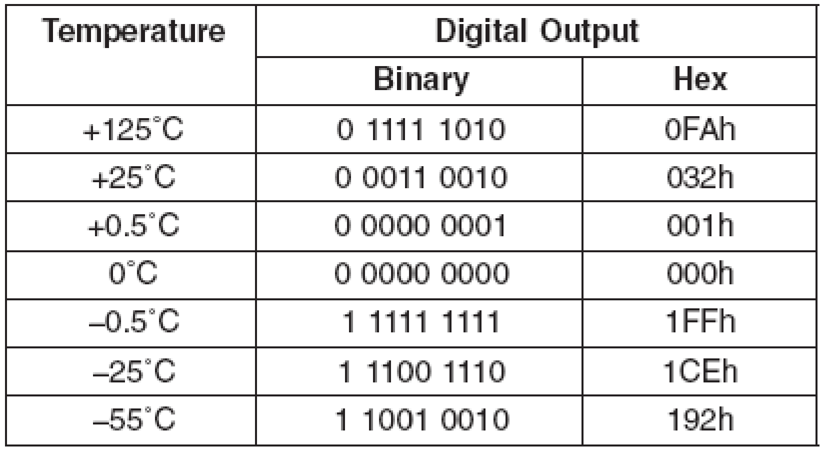
LM75 address

1	0	0	1	A2	A1	A0
MSB						LSB

LM75 is always a slave (not capable of being a master).

The address (7 bit) is composed of 4 constant bits and the external pin settings (A2-A0).

LM75: Temperature data format

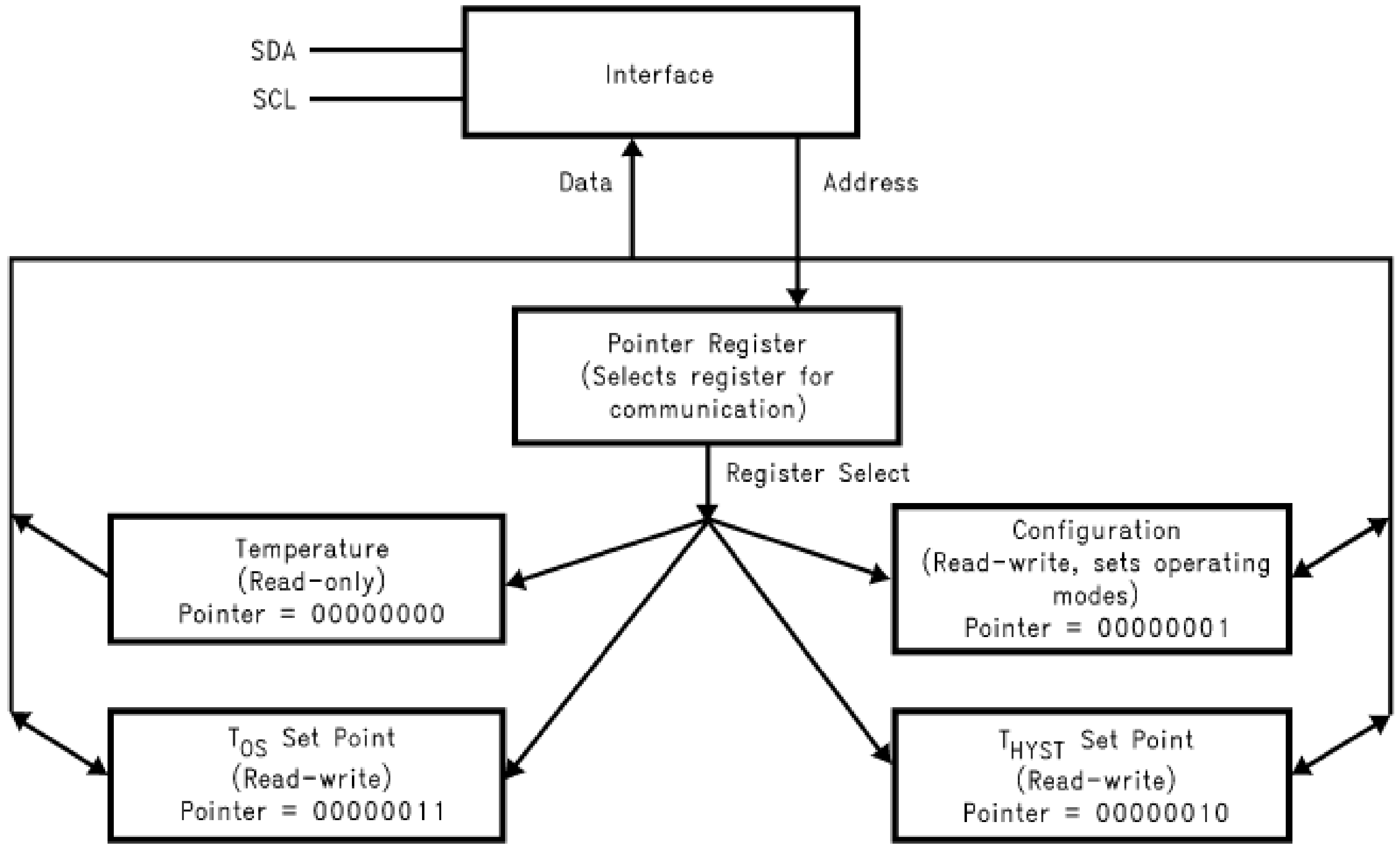


Temperature	Digital Output	
	Binary	Hex
+125˚C	0 1111 1010	0FAh
+25˚C	0 0011 0010	032h
+0.5˚C	0 0000 0001	001h
0˚C	0 0000 0000	000h
–0.5˚C	1 1111 1111	1FFh
–25˚C	1 1100 1110	1CEh
–55˚C	1 1001 0010	192h

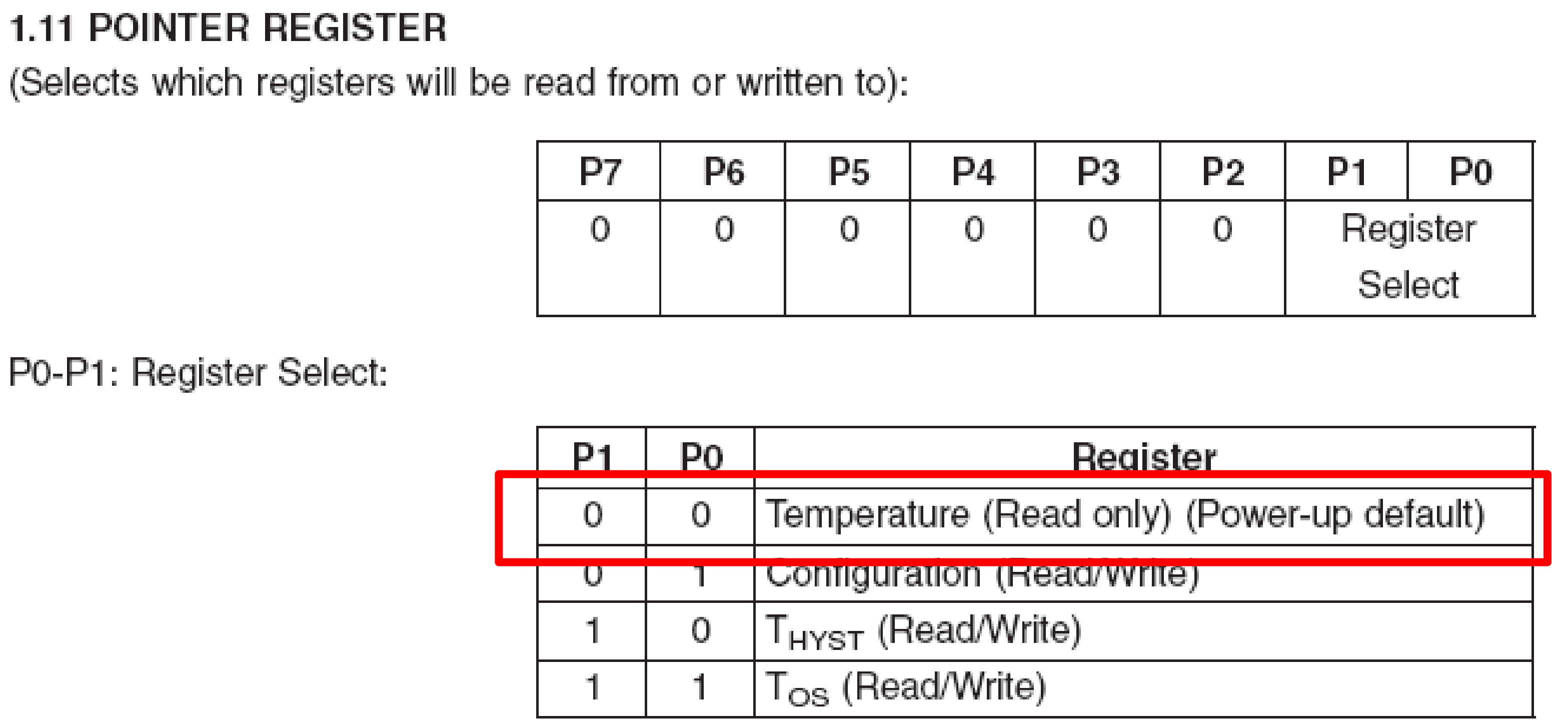
2's complement format.

LSB = 0,5 deg. Celsius.

LM75: Register structure


SDA
SCL
Interface
Data
Address
Pointer Register (Selects register for communication)
Register Select
Temperature (Read-only) Pointer = 00000000
Configuration (Read-write, sets operating modes) Pointer = 00000001
T_{OS} Set Point (Read-write) Pointer = 00000011
T_{HYST} Set Point (Read-write) Pointer = 00000010

LM75: Pointer register



1.11 POINTER REGISTER

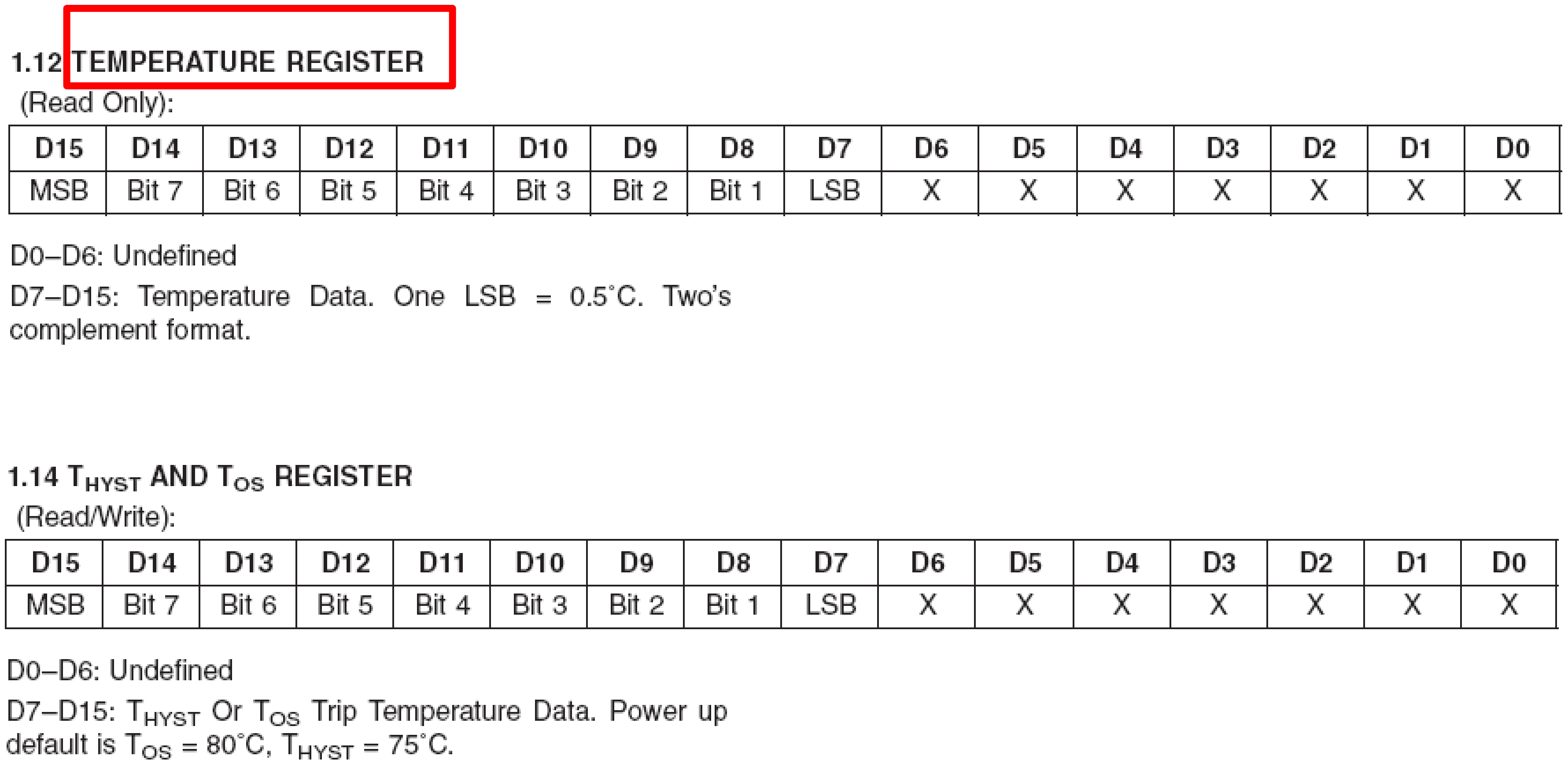
(Selects which registers will be read from or written to):

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Register Select	

P0-P1: Register Select:

P1	P0	Register
0	0	Temperature (Read only) (Power-up default)
0	1	Configuration (Read/Write)
1	0	T_{HYST} (Read/Write)
1	1	T_{OS} (Read/Write)

LM75: Temperature registers



1.12 TEMPERATURE REGISTER

(Read Only):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB	X	X	X	X	X	X	X

D0–D6: Undefined

D7–D15: Temperature Data. One LSB = 0.5˚C. Two's complement format.

1.14 T_{HYST} AND T_{OS} REGISTER

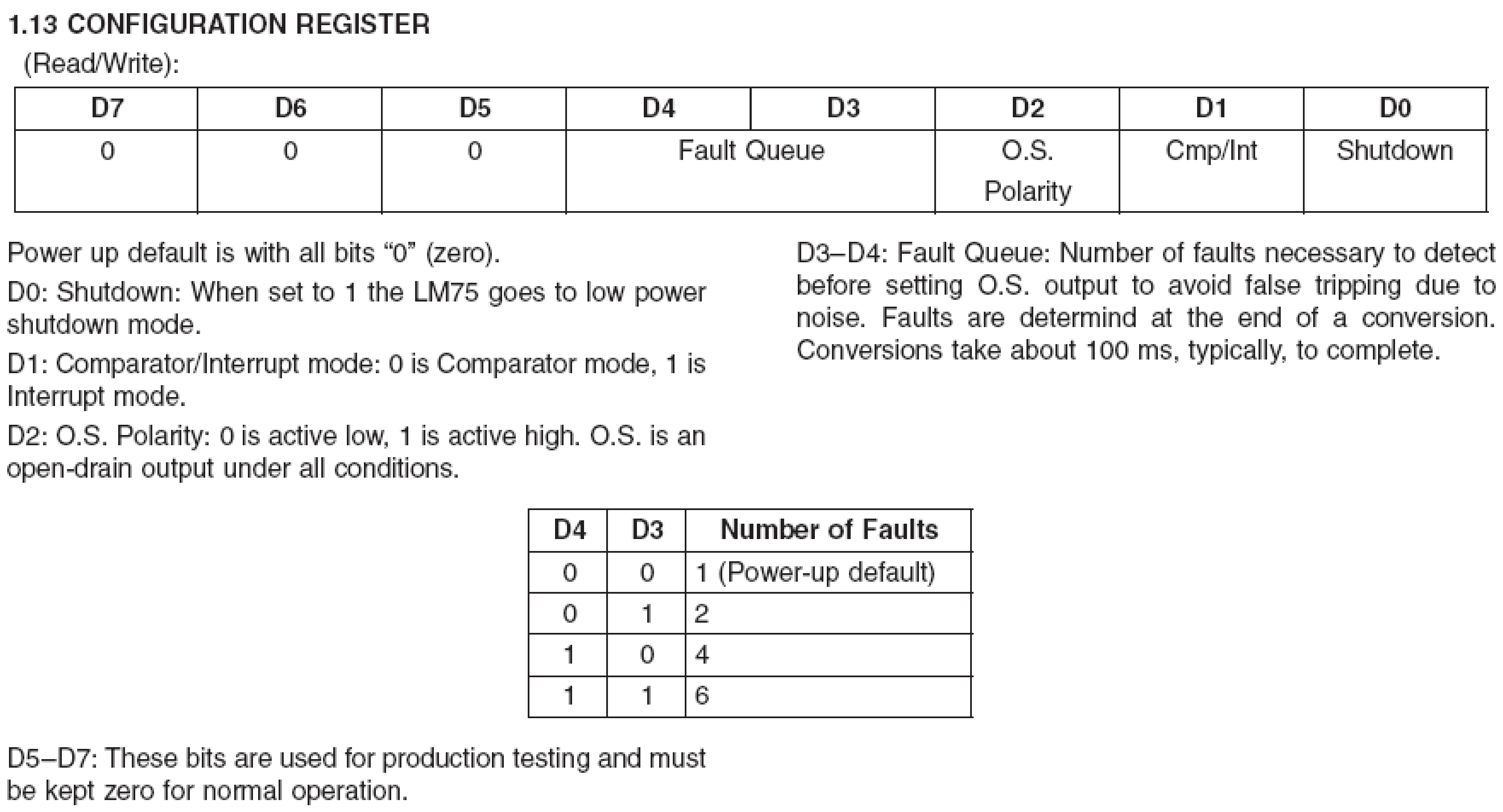
(Read/Write):

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB	X	X	X	X	X	X	X

D0–D6: Undefined

D7–D15: T_{HYST} Or T_{OS} Trip Temperature Data. Power up default is T_{OS} = 80˚C, T_{HYST} = 75˚C.

LM75: Configuration register



1.13 CONFIGURATION REGISTER

(Read/Write):

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	Fault Queue		O.S. Polarity	Cmp/Int	Shutdown

Power up default is with all bits "0" (zero).

D0: Shutdown: When set to 1 the LM75 goes to low power shutdown mode.

D1: Comparator/Interrupt mode: 0 is Comparator mode, 1 is Interrupt mode.

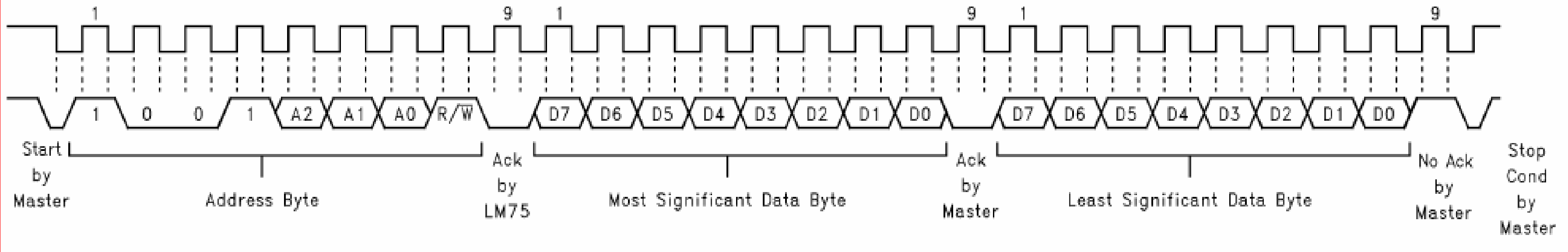
D2: O.S. Polarity: 0 is active low, 1 is active high. O.S. is an open-drain output under all conditions.

D3–D4: Fault Queue: Number of faults necessary to detect before setting O.S. output to avoid false tripping due to noise. Faults are determind at the end of a conversion. Conversions take about 100 ms, typically, to complete.

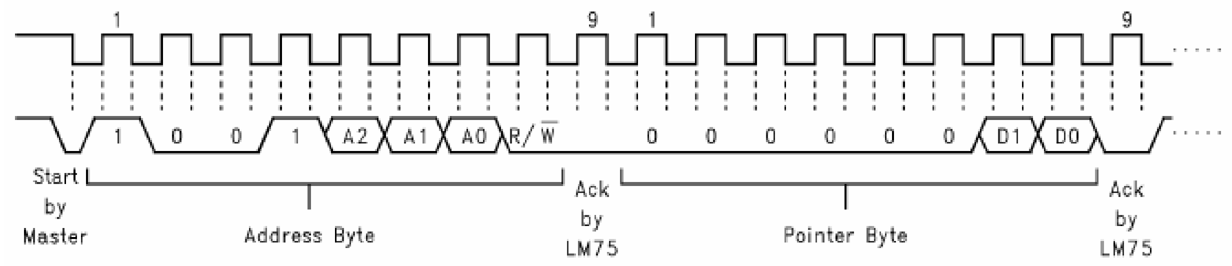
D4	D3	Number of Faults
0	0	1 (Power-up default)
0	1	2
1	0	4
1	1	6

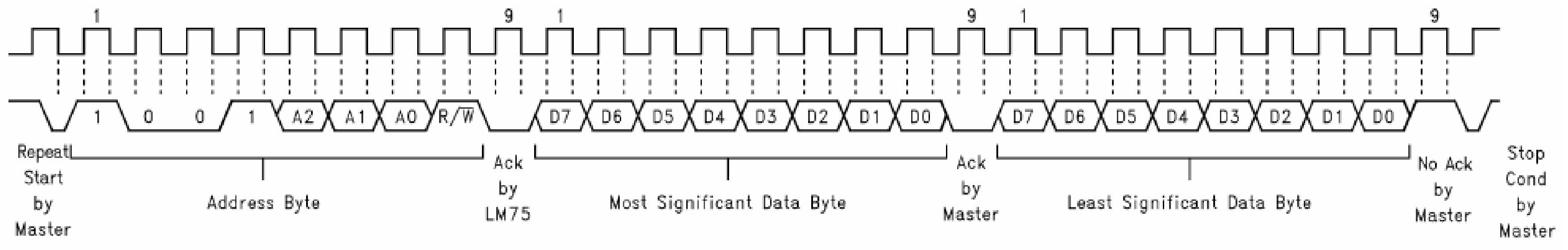
D5–D7: These bits are used for production testing and must be kept zero for normal operation.

LM75: I2C timing, 2 byte reading



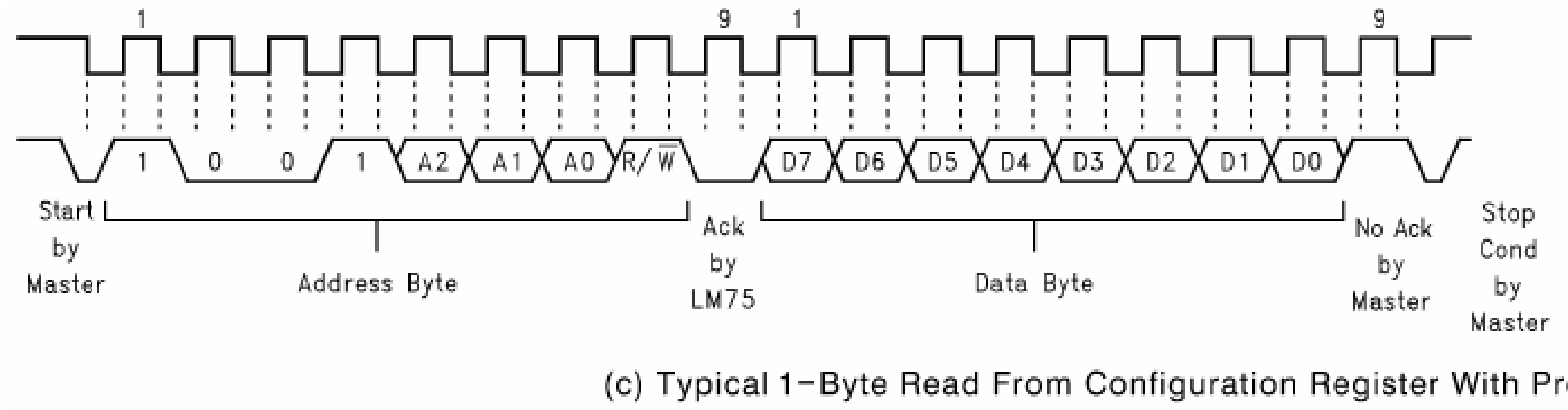
(a) Typical 2−Byte Read From Preset Pointer Location Such as Temp, T_{OS}, T_{HYST}





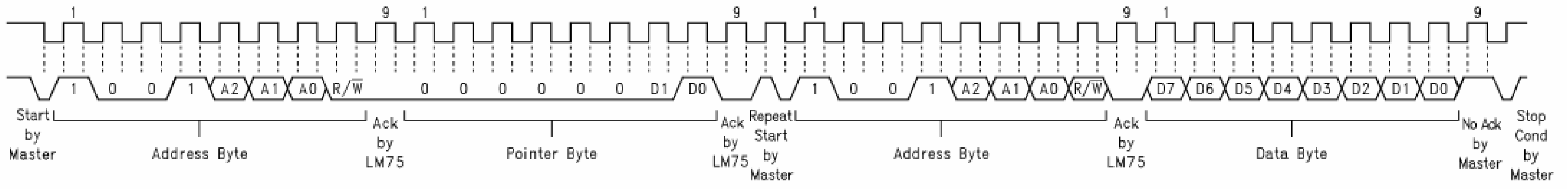
(b) Typical Pointer Set Followed by Immediate Read for 2−Byte Register such as Temp, T_{OS}, T_{HYST}

LM75: I2C timing, 1 byte reading

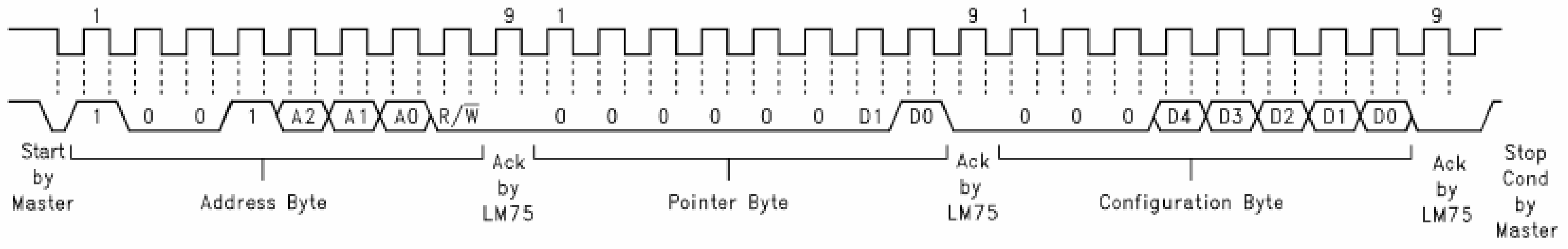


(c) Typical 1−Byte Read From Configuration Register With Preset Pointer

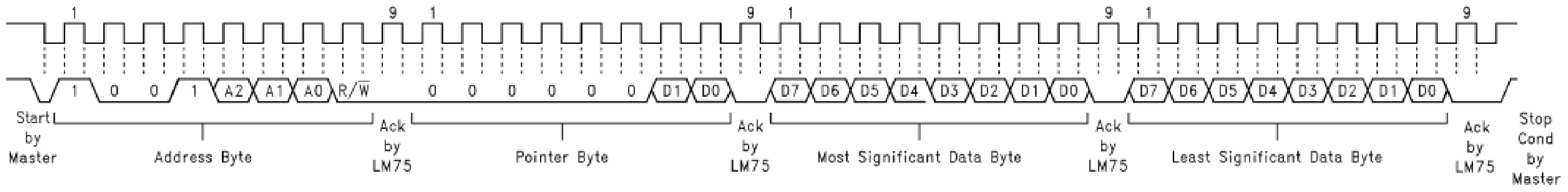
LM75: I2C timing



(a) Typical Pointer Set Followed by Immediate Read from Configuration Register

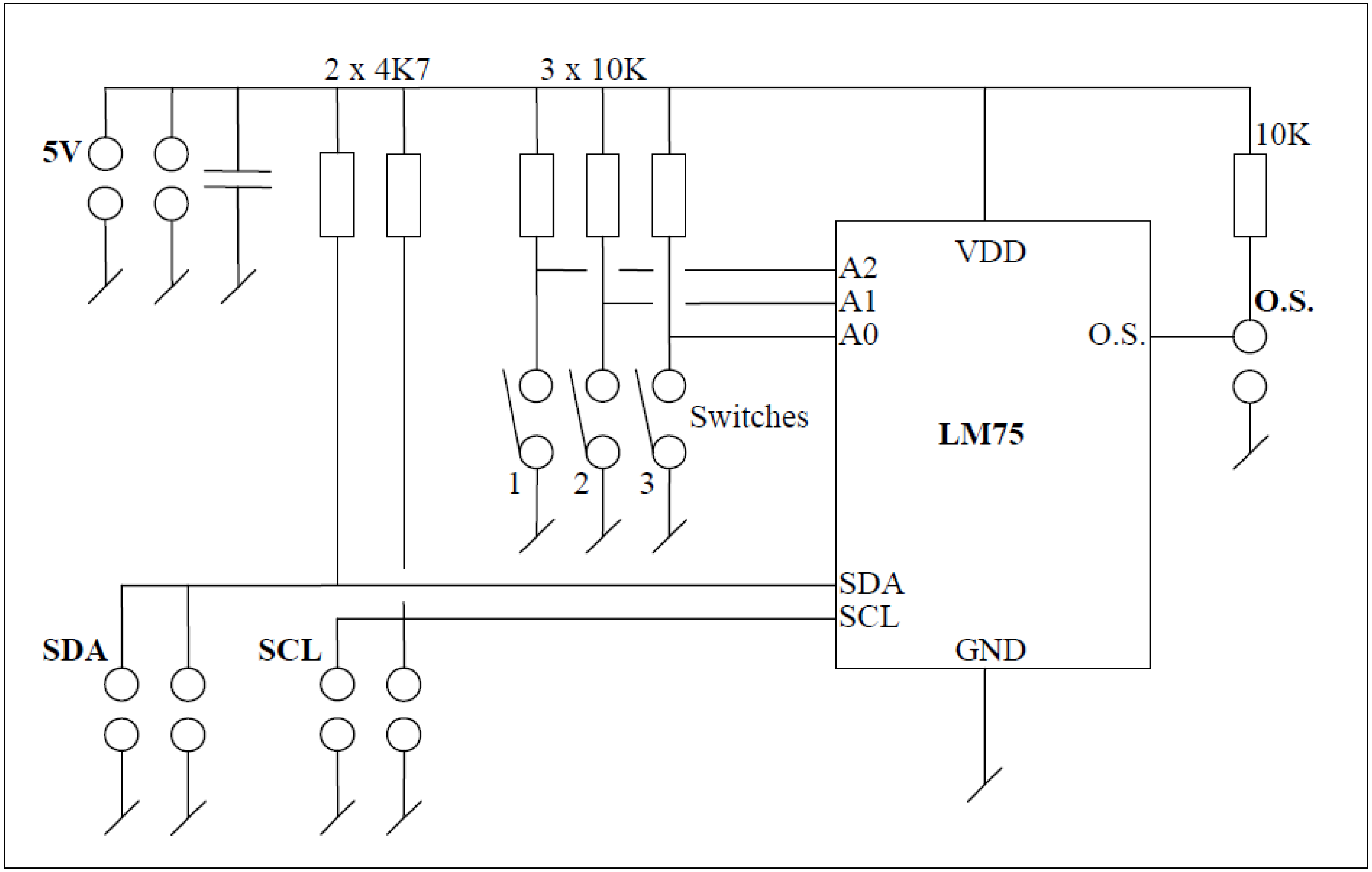


(b) Configuration Register Write


1
9
1
9
1
9
1
9
1 0 0 1 A2 A1 A0 R/W̄ 0 0 0 0 0 0 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
Start by Master
Address Byte
Ack by LM75
Pointer Byte
Ack by LM75
Most Significant Data Byte
Ack by LM75
Least Significant Data Byte
Ack by LM75
Stop Cond by Master

(c) T_{OS} and T_{HYST} Write

LAB16: LM75 I2C slave



End of lesson 3