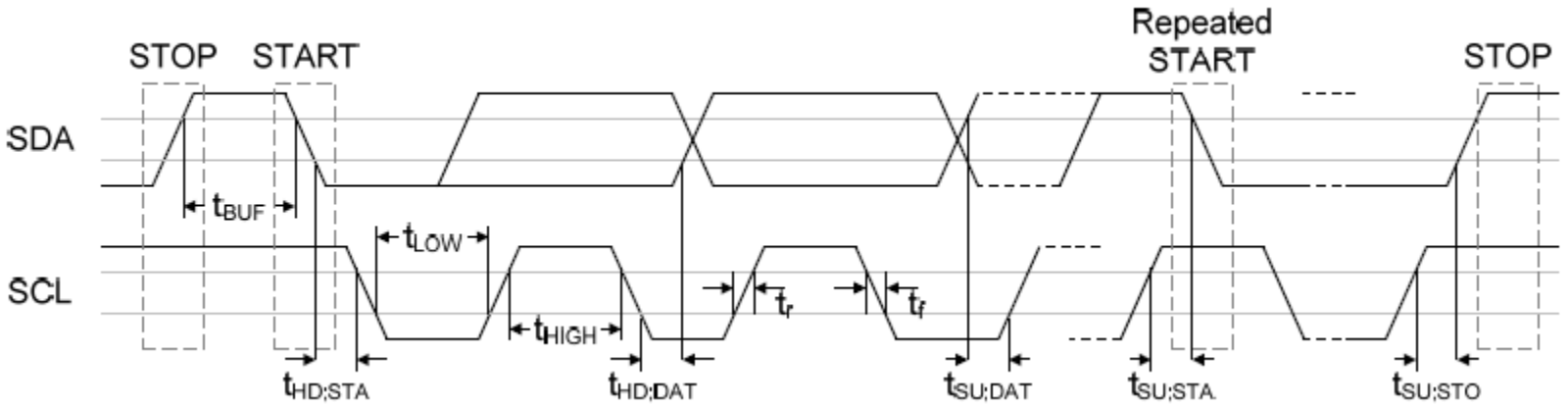
I²C Serial Interface Controller :

- is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices(Master/Slave).
- There is one SCL clock pulse for each data bit with the MSB being transmitted first. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL.



I²C : features

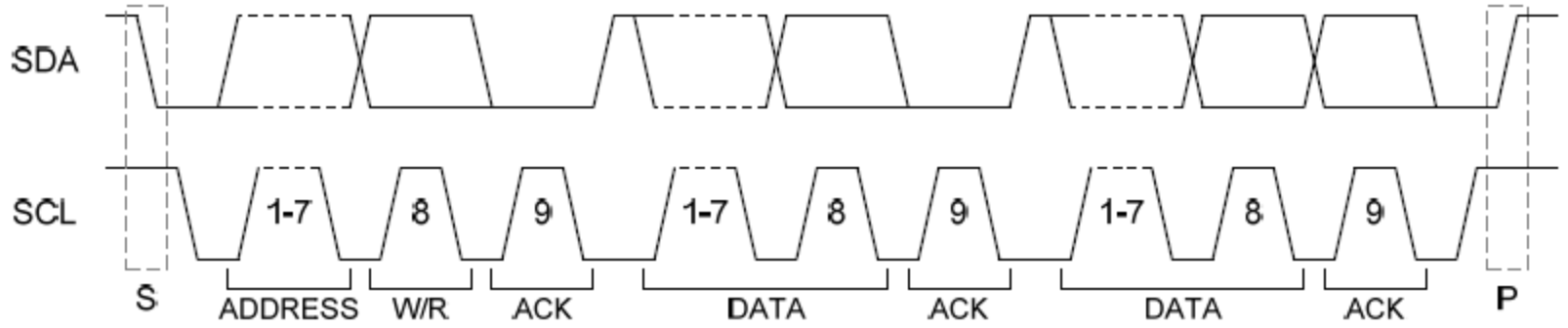
- The I2C port handles byte transfers autonomously.
- To enable this port, the bit ENS1 in I2CON should be set to '1'.
- The I2C H/W interfaces to the I2C bus via two pins: SDA and SCL.
- Pull up resistor is needed for I2C operation as these are open drain pins. When the I/O pins are used as I2C port, user must set the pins function to I2C in advance.

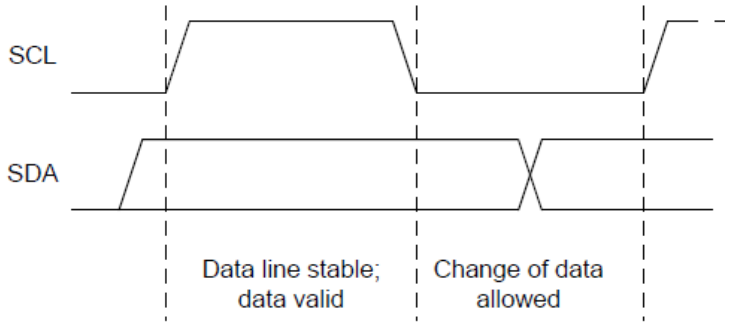
- Built-in a 14-bit time-out counter will request the I2C interrupt if the I2C bus hangs up and timer-out counter overflows.
- Supports 7-bit addressing mode

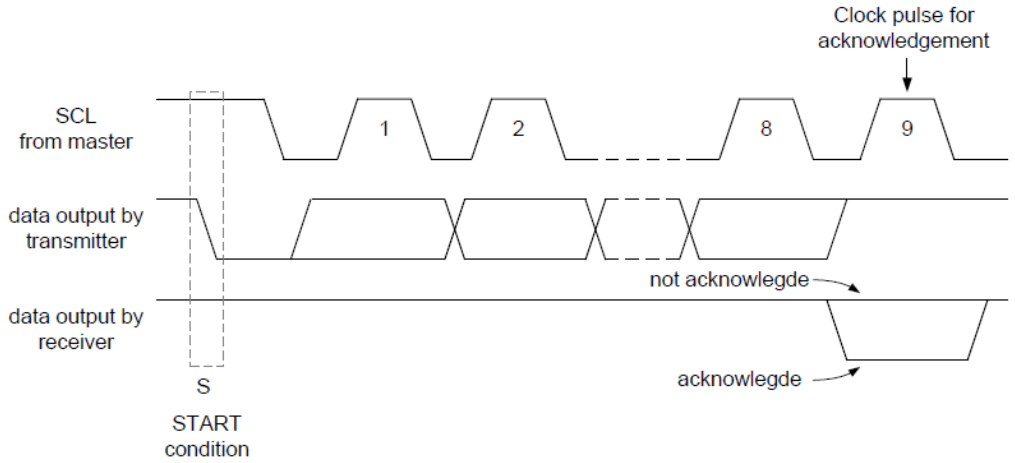
- When I2C port is enabled by setting ENS1 (I2CON [6]) to high, the internal states will be controlled by I2CON and I2C logic hardware. Once a new status code is generated and stored in I2CSTATUS, the I2C Interrupt Flag bit SI (I2CON [3]) will be set automatically. If the Enable Interrupt bit EI (I2CON [7]) is set high at this time, the I2C interrupt will be generated. The bit field I2CSTATUS[7:3] stores the internal state code, the lowest 3 bits of I2CSTATUS are always zero and the content keeps stable until SI is cleared by software. The base address is 4002_0000 and 4012_0000.

I²C : Protocol

1) START or Repeated START signal generation
2) Slave address and R/W bit transfer
3) Data transfer
4) STOP signal generation







I²C : Register Map

Register	Offset	R/W	Description	Reset Value
I2C0_BA = 0x4002_0000 I2C1_BA = 0x4012_0000				
I2CON	I2Cx_BA+0x00	R/W	I²C Control Register	0x0000_0000
I2CADDR0	I2Cx_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2CDAT	I2Cx_BA+0x08	R/W	I²C DATA Register	0x0000_0000
I2CSTATUS	I2Cx_BA+0x0C	R	I²C Status Register	0x0000_00F8
I2CLK	I2Cx_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000
I2CTOC	I2Cx_BA+0x14	R/W	I²C Time Out Control Register	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000
I2CADM0	I2Cx_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2CADM3	I2Cx_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000

I²C : Data Register (I2CDAT)

- contains a byte of serial data to be transmitted or a byte which just has been received.
- The CPU can read from or write to this 8-bit (I2CDAT [7:0]) directly while it is not in the process of shifting a byte. when I2C is in a defined state and the serial interrupt flag (SI) is set. Data in I2CDAT [7:0] remains stable as long as SI bit is set.

I²C : Control Register (I2CON)

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7]	**EI**	**Enable Interrupt** 1 = Enable I²C interrupt 0 = Disable I²C interrupt
[6]	**ENS1**	**I²C Controller Enable Bit** 1 = Enable 0 = Disable Set to enable I²C serial function controller. When ENS1=1 the I²C serial function enables. The multi-function pin function of SDA and SCL must set to I²C function first.
[5]	**STA**	**I²C START Control Bit** Setting STA to logic 1 to enter master mode, the I²C hardware sends a START or repeat START condition to bus when the bus is free.
[4]	**STO**	**I²C STOP Control Bit** In master mode, setting STO to transmit a STOP condition to bus then I²C hardware will check the bus condition if a STOP condition is detected this bit will be cleared by hardware automatically. In a slave mode, setting STO resets I²C hardware to the defined "not addressed" slave mode. This means it is NO LONGER in the slave receiver mode to receive data from the master transmit device.
[3]	**SI**	**I²C Interrupt Flag** When a new I²C state is present in the I2CSTATUS register, the SI flag is set by hardware, and if bit EI (I2CON [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI is by writing 1 to this bit.

I²C : Status Register (I2CSTATUS)

Master mode		Slave Mode	
STATUS	Description	STATUS	Description
0x08	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10	Master Repeat Start	0xA8	Slave Transmit Address ACK
0x18	Master Transmit Address ACK	0xB0	Slave Transmit Arbitration Lost
0x20	Master Transmit Address NACK	0xB8	Slave Transmit Data ACK
0x28	Master Transmit Data ACK	0xC0	Slave Transmit Data NACK
0x30	Master Transmit Data NACK	0xC8	Slave Transmit Last Data ACK
0x38	Master Arbitration Lost	0x60	Slave Receive Address ACK
0x40	Master Receive ACK	0x68	Slave Receive Arbitration Lost
0x48	Master Receive NACK	0x80	Slave Receive Data ACK
0x50	Master Receive ACK	0x88	Slave Receive Data NACK
0x58	Master Receive NACK	0x70	GC mode Address ACK
0x00	Bus error	0x78	GC mode Arbitration Lost
		0x90	GC mode Data ACK
		0x98	GC mode Data NACK
0xF8	Bus Released Note: Status "0xF8" exists in both master/slave modes, and it won't raise interrupt.		

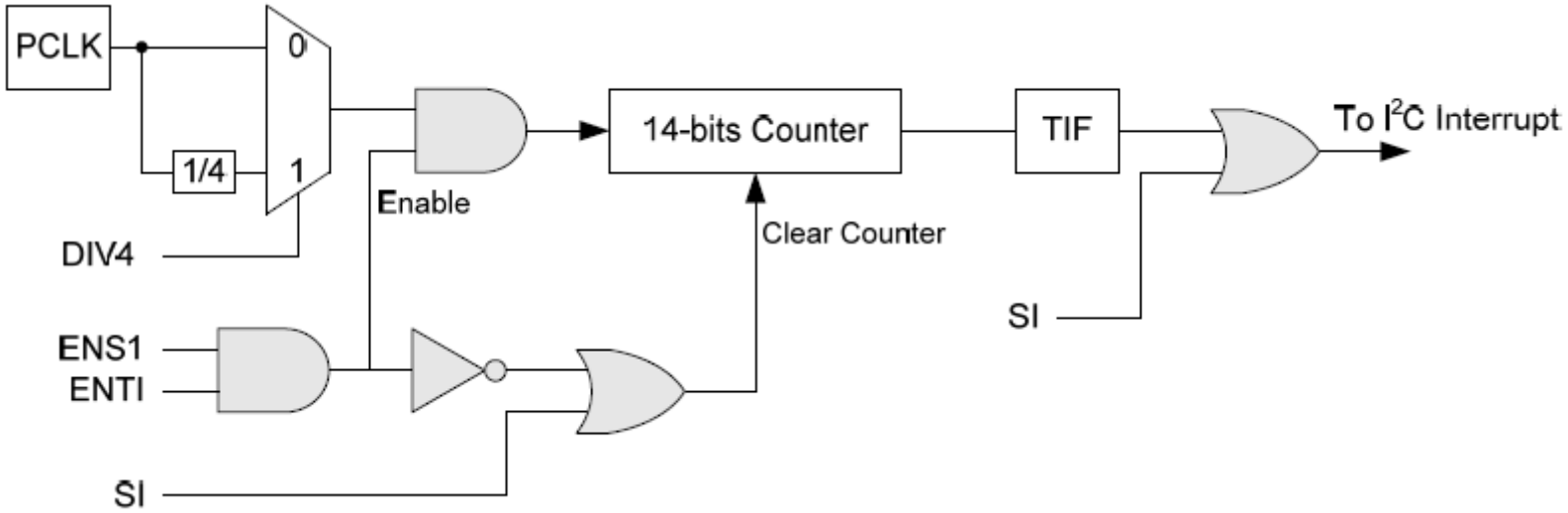
I²C : I2C Clock Baud Rate Bits (I2CLK)

- The data baud rate of I2C is determines by **I2CLK [7:0]** register when I2C is in a master mode.
- The data baud rate of I2C setting is **Data Baud Rate** of **I2C**
  = (system clock) / (4x (**I2CLK [7:0]** +1)).

If system clock = 16 MHz,
the **I2CLK [7:0]** = 40 (28H),
so
data baud rate of I2C = 16 MHz/ (4x (40 +1)) = **97.5** Kbits/sec.

I²C : Time-out Counter Register (I2CTOC)

- a 14-bit time-out counter which can be used to deal with the I2C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows (TIF=1) and generates I2C interrupt to CPU or stops counting by clearing ENTI to 0.



I²C : Address Registers (I2CADDR)

- I2C port is equipped with four slave address registers I2CADDRn (n=0~3).
- The I2C hardware will react if the contents of I2CADDRn are matched with the received slave address.

- The I2C ports support the "General Call" function. If the GC bit (I2CADDRn [0]) is set the I2C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

- I2C bus controllers support multiple address recognition with four address mask registers I2CADMn (n=0~3). When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register.

I²C : Program Example

```
void Write_24LC64(uint32_t address,uint8_t data )
{
  uint32_t i;
  SystemCoreClock = DrvSYS_GetHCLKFreq();
  //Open I2C1 and set clock = 50Kbps
  DrvI2C_Open(I2C_PORT1, 50000);

    //send i2c start
  DrvI2C_Ctrl(I2C_PORT1, 1, 0, 0, 0); //set start
  while (I2C1->I2CON.SI == 0);        //poll si flag
  //send writer command
  I2C1->I2CDAT = 0XA0;                //send writer command
    DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); //clr si flag
    while( I2C1->I2CON.SI == 0 );       //poll si flag
  //send address high
  I2C1->I2CDAT = (address>>8)&0XFF;
  DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); //clr si and set ack
  while( I2C1->I2CON.SI == 0 );       //poll si flag
  //send address low
  I2C1->I2CDAT = address&0XFF;
  DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); //clr si and set ack
  while( I2C1->I2CON.SI == 0 );       //poll si flag
  //send data
  I2C1->I2CDAT = data;                //write data to
  DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); //clr si and set ack
  while( I2C1->I2CON.SI == 0 );       //poll si flag
    //send i2c stop
  DrvI2C_Ctrl(I2C_PORT1, 0, 1, 1, 0); //send stop
  while( I2C1->I2CON.STO);
  //while( I2C1->CON.SI == 0 );
  for(i=0;i<60;i++);
  DrvI2C_Close(I2C_PORT1);

  for(i=0;i<6000;i++);
  for(i=0;i<6000;i++);
}
```

I²C : Program Example

DrvI2C_Ctrl

Prototype

void DrvI2C_Ctrl(E_I2C_PORT port, uint8_t start, uint8_t stop, uint8_t intFlag, uint8_t ack);

Description

To set I2C control bit include STA, STO, AA, SI in control register.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0 / I2C_PORT1)

start [in]

To set STA bit or not. (1: set, 0: don't set). If the STA bit is set, a START or repeat START signal will be generated when I2C bus is free.

stop [in]

To set STO bit or not. (1: set, 0: don't set). If the STO bit is set, a STOP signal will be generated. When a STOP condition is detected, this bit will be cleared by hardward automatically.

intFlag [in]

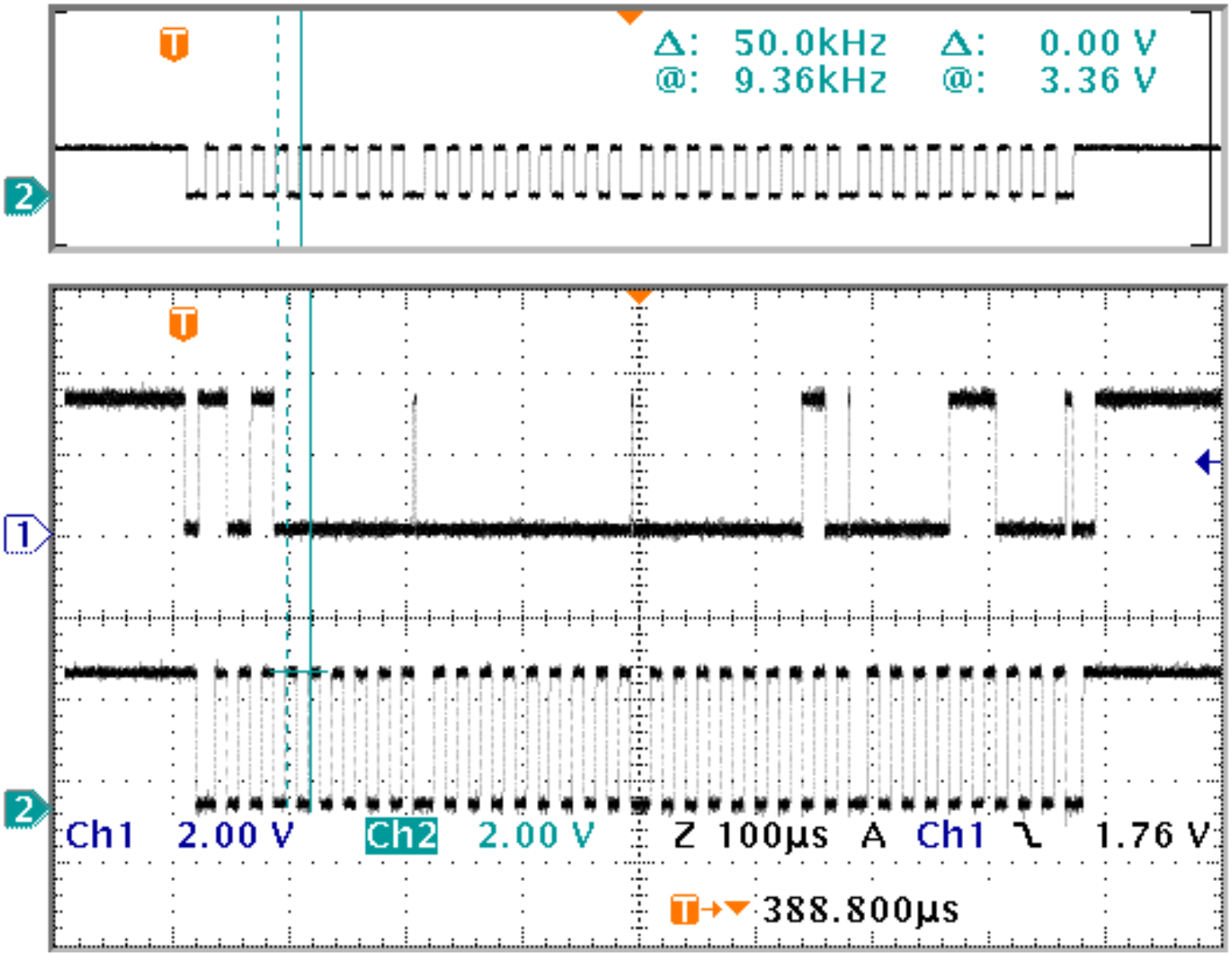
To clear SI flag (I2C interrupt flag). (1: clear, 0: don't work)

ack [in]

To enable AA bit (Assert Acknowledge control bit) or not. (1: enable, 0: disable)

I²C : Program Example

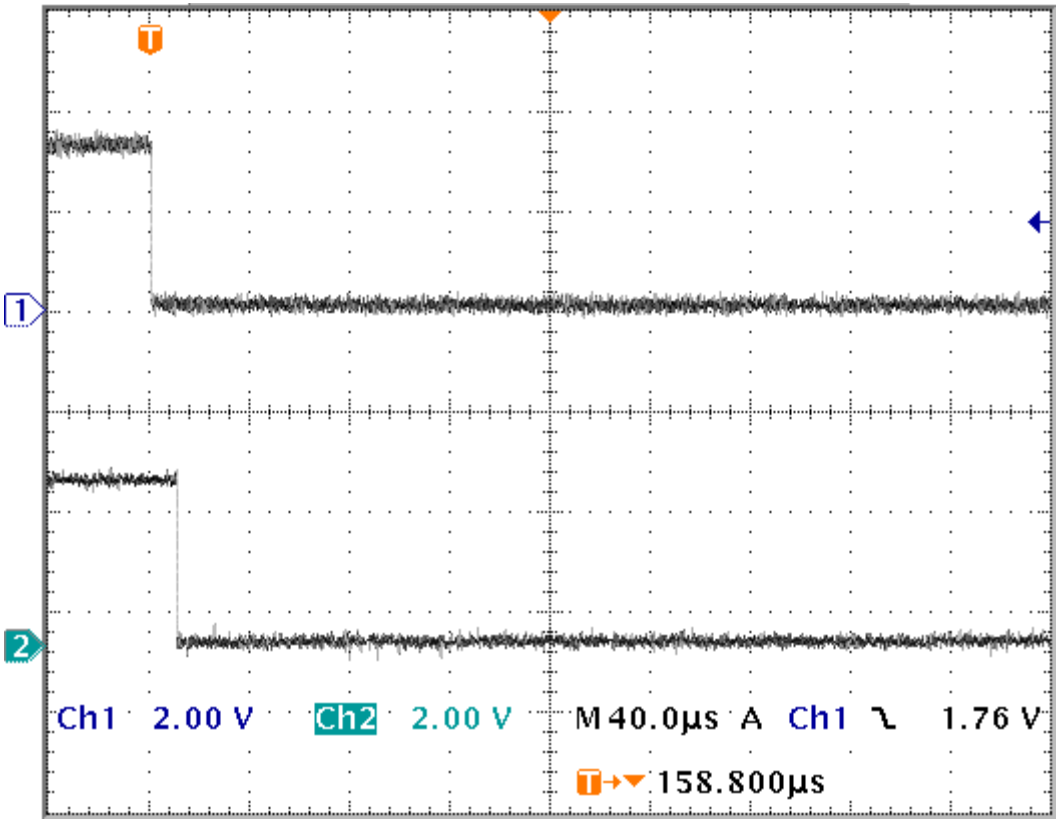
Write_24LC64(0x00000000+temp,temp+11); // temp = 1



I²C : Program Example

Write_24LC64(0x00000000+temp,temp+11); // temp = 1

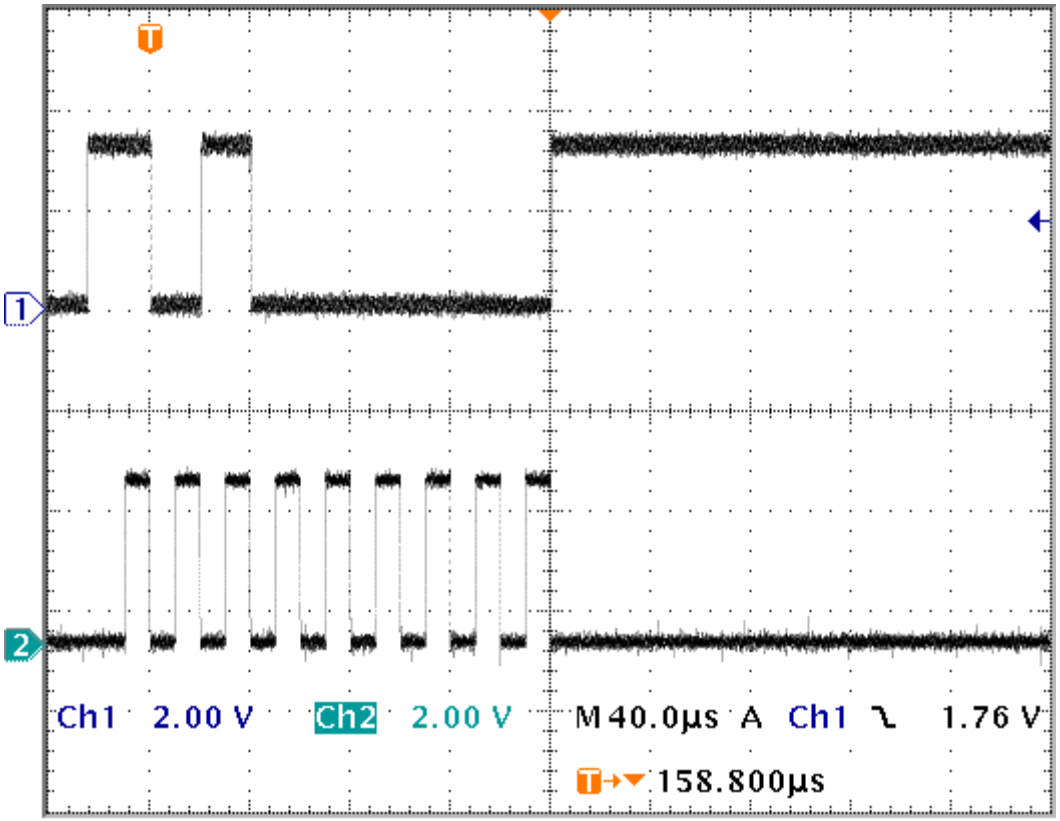
DrvI2C_Ctrl(I2C_PORT1, 1, 0, 0, 0);//set start



I²C : Program Example

Write_24LC64(0x00000000+temp,temp+11); // temp = 1

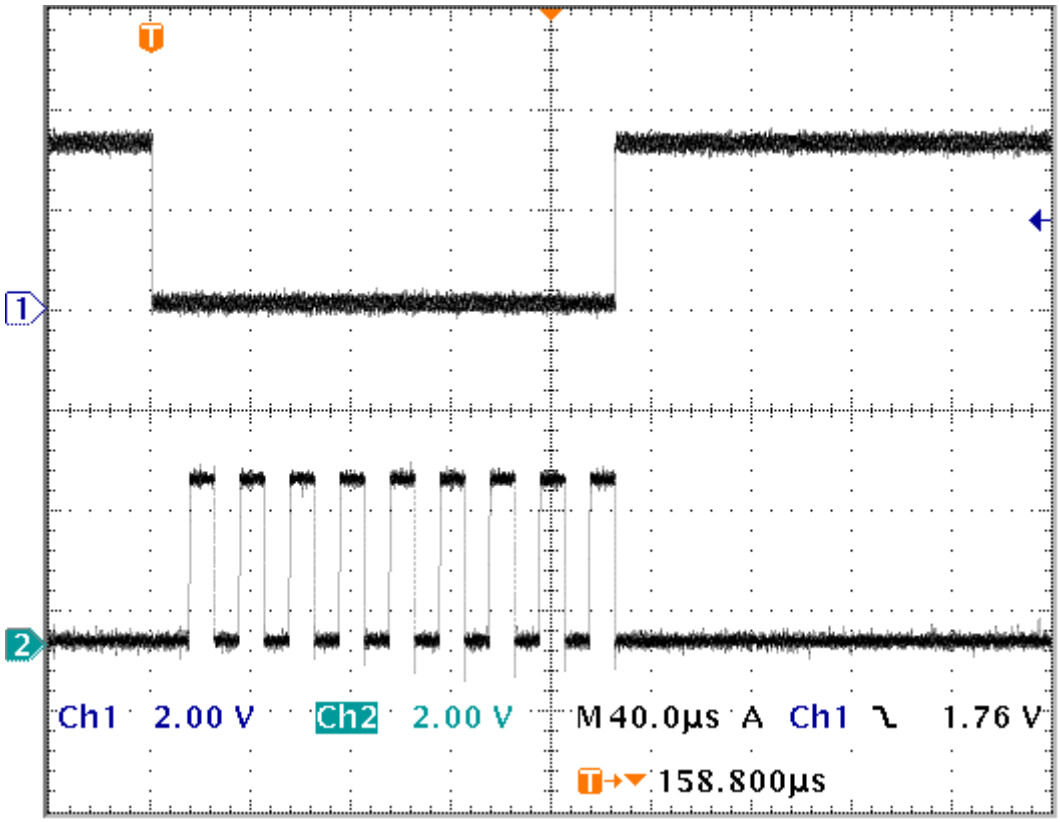
```
I2C1->I2CDAT = 0xA0;
DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 0); //clr si flag
```



I²C : Program Example

Write_24LC64(0x00000000+temp,temp+11); // temp = 1

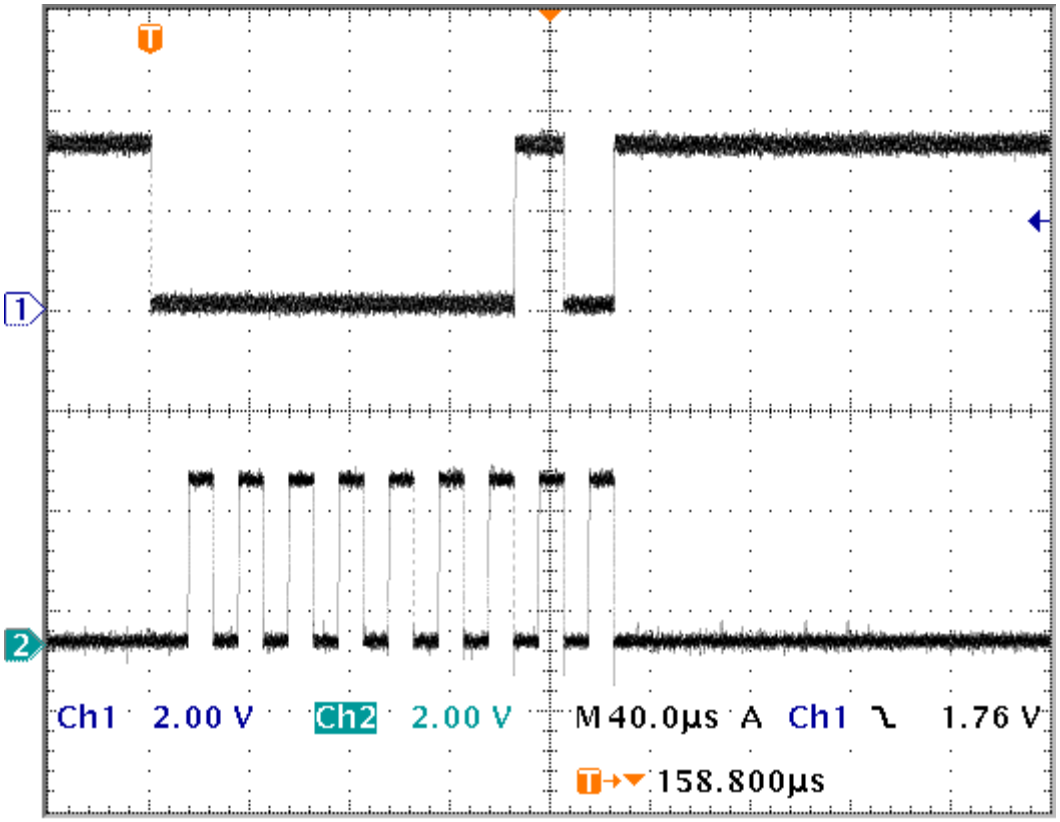
```
I2C1->I2CDAT = (address>>8)&0xFF;
DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); //clr si and set ack
```



I²C : Program Example

Write_24LC64(0x00000000+temp,temp+11); // temp = 1

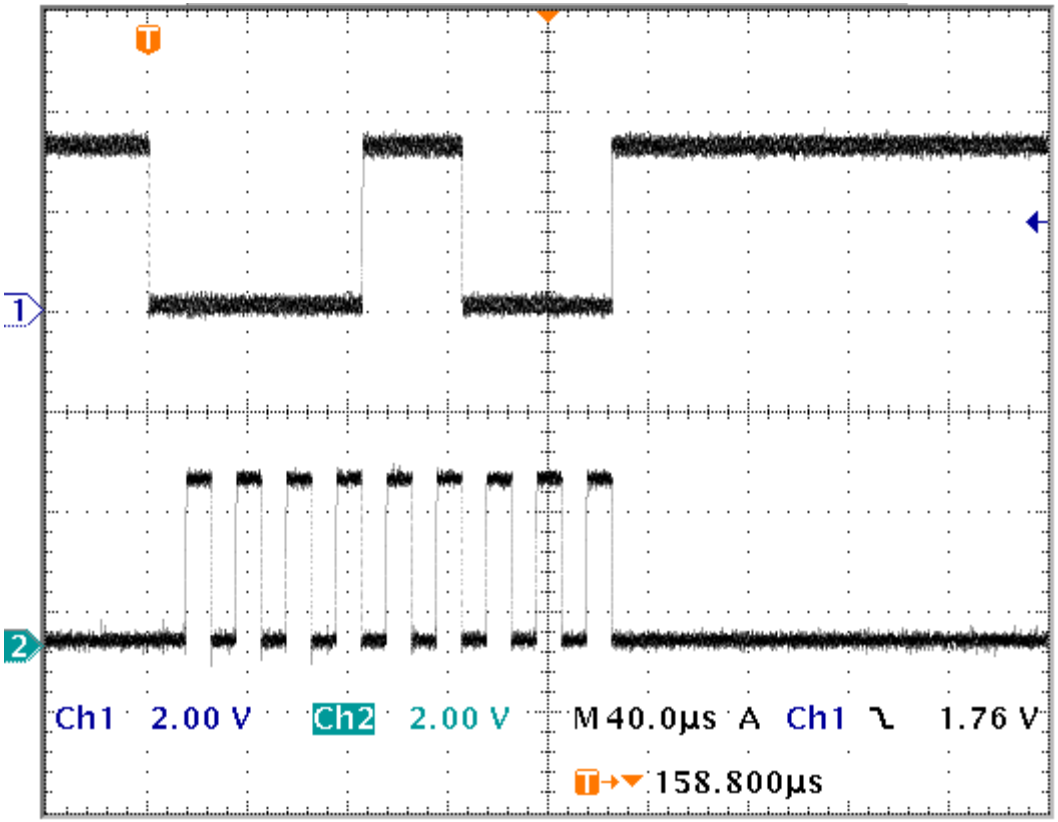
```
I2C1->I2CDAT = address&0XFF;
DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); //clr si and set ack
```



I²C : Program Example

Write_24LC64(0x00000000+temp,temp+11); // temp = 1

```
I2C1->I2CDAT = data;
DrvI2C_Ctrl(I2C_PORT1, 0, 0, 1, 1); //clr si and set ack
```



I²C : Program Example

Write_24LC64(0x00000000+temp,temp+11); // temp = 1

DrvI2C_Ctrl(I2C_PORT1, 0, 1, 1, 0); //send stop