Introduction
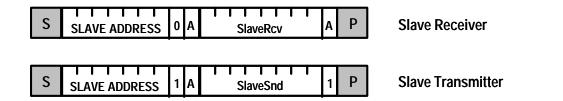
This technical note shows an I2C slave software example for the Philips Semiconductors LPC2000 microcontroller family. The software is written for the LPC2138 and tested on an MCB2130 board. It supports interrupt driven I2C slave message transfers.

I2C slave demo

I2C slave mode functions are very specific to the system design, and therefore, very difficult to make generic. In the example below an I2C interrupt is generated if the interface recognizes its own slave address (0x20 or 0x21). When addressed as "slave transmitter", byte *SlaveSnd* (which is actually the analog value at AIN1 = P0.28) is transmitted. When addressed as "slave receiver", byte *SlaveRcv* from the bus master is received. This byte is reflected to port pins P1.16 to P1.23.

Slave Receiver

Slave Transmitter

```c
#include "LPC213x.h"

extern void I2C0_Init(void);
extern unsigned char SlaveSnd;


static unsigned char ADC_Read(void)
{
  unsigned int i;

    AD0CR  = 0x00200302;                // Init ADC (Pclk = 12MHz) and select channel AD0.1
    AD0CR |= 0x01000000;                // Start A/D Conversion
    do
    {
       i = AD0DR;                       // Read A/D Data Register
    } while ((i & 0x80000000) == 0);    // Wait for end of A/D Conversion

    return (i >> 8) & 0x00FF;           // bit 8:15 is 8-bit AD value
}

void main(void)
{
    PINSEL1 |= 0x01000000;              // P0.28 = AD0.1
    IODIR1   = 0x00FF0000;              // P1.16..23 defined as Outputs
    I2C0_Init();                        // initialize I2C bus

    while (1)
    {
       SlaveSnd = ADC_Read();           // convert and send channel AD0.1
    }
}
```

PHILIPS

```c
#include "LPC213x.h"

unsigned char SlaveRcv = 0xAA;
unsigned char SlaveSnd;


void I2C0_Isr(void) __irq
{
  unsigned char st;

    st = I2C0STAT;
    I2C0CONCLR = 0x2C;                  // clear STA, AA and SI
    switch(st)
    {
      case 0x60:                        // own SLA+W received, Ack returned (slave receiver)
      case 0x68:                        // Addressed as slave
        I2C0CONSET = 0x04;              // set AA, return ACK on first byte
        break;
      case 0x80:                        // Data received, ACK returned
        SlaveRcv = I2C0DAT;             // read and store data, NACK on next byte
        IOCLR1 = 0x00FF0000;            // Turn off LEDs P1.16..23
        IOSET1 = SlaveRcv << 16;        // Turn on LED
        break;
      case 0x88:                        // data received, NACK returned
      case 0xA0:                        // STOP or REP.START received while addressed as slave
      case 0xC0:                        // Data transmitted, NOT ACK received
      case 0xC8:                        // Last data transmitted, ACK received
        I2C0CONSET = 0x04;              // set AA, switch to not addressed slave mode
        break;
      case 0xA8:                        // own SLA+R received, Ack returned (slave transmitter)
      case 0xB8:                        // Data transmitted, ACK received
        I2C0DAT = SlaveSnd;             // Transmit last data AA = 0
        break;
      default:
        break;
    }
    VICVectAddr = 0;                    // reset VIC
}

void I2C0_Init(void)
{
    PINSEL0   |= 0x50;                  // P0.3 = SDA, P0.2 = SCL
    I2C0ADR    = 0x20;                  // set I2C slave address
    I2C0CONSET = 0x44;                  // enable I2C hardwar and set AA (ack)

    VICVectAddr0  = (unsigned int) &I2C0_Isr;
    VICVectCntl0  = 0x29;              // Channel1 on Source#9 ... enabled
    VICIntEnable |= 0x200;            // 9th bit is the I2C
}
```