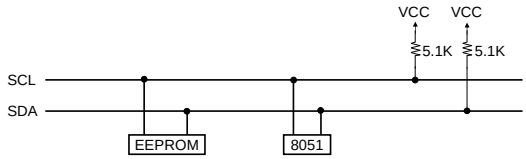
I²C EEPROM

- Why EEPROM?
  - Non-volatile memory
  - In-circuit programming/erasing
  - Inexpensive
- Applications
  - Speed dial on phone
  - Retain settings on car, TV, other devices
- Device contains a tunneling dielectric
- Erased state is when the bit cell is storing a '1'
- ~20V is generated by internal charge pump in order to program the device

I²C Overview

- I²C = IIC = Inter-Integrated Circuit
- Synchronous bi-directional protocol
- 2-wire interface
  - SCL – Clock
  - SDA – Data
- With standard I²C, up to 16kbits (2KB) of EEPROM maximum on a single I²C bus
- Supports devices such as EEPROM and data converters
- I²C protocol from 1980's
  - Standard (100 kbps)
  - Fast (400 kbps)
  - High-Speed (3.4 Mbps)
- Other related serial interfaces
  - SPI, Microwire, 3-wire, 1-wire, SMBus, CANBUS

8051 Connections



SCL pull-up somewhat optional in our implementation since 8051 is the only master. If more than one master, must use open drain.

Implementation

- Hardware
  - Use decoupling capacitor
  - Use pull-up resistors for SCL and SDA
  - Pick two port pins on the 8051
- Firmware
  - Check generation of SCL with oscilloscope, ensure it doesn't exceed maximum bus speed (e.g. 100 kHz)
  - I²C functions for SDCC available on web
  - Can use ACK polling to determine device readiness

I²C Terms

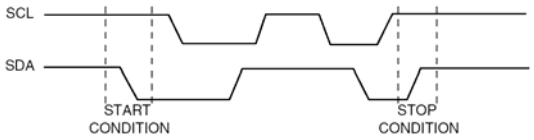
- Master (8051)
  - Device which initiates all activity on the bus
  - Generates clock for the transfer
  - Generates start and stop conditions
- Slave (EEPROM)
  - Responds to the master's request
- Receiver
  - Reads data on the bus. Can be Master or Slave.
- Transmitter
  - Writes data to the bus. Can be Master or Slave.
- Page Block
  - 2Kbits of data → 256 bytes → 16 pages
- Page
  - 16 bytes (8 byte pages for small EEPROMs like 24C02)

I²C Protocol – Bus Conditions

Four bus conditions

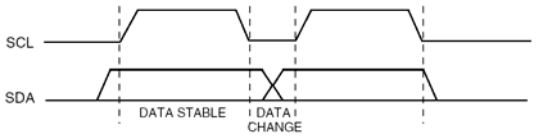
- Start
  - High to low transition of SDA when SCL is high. Generated by Master only.
- Stop
  - Low to high transition of SDA when SCL is high. Generated by Master only.
- Data
  - Data must be valid when SCL is high
  - Data can change when SCL is low
- ACK
  - Receiver drives this condition

Start and Stop Definition



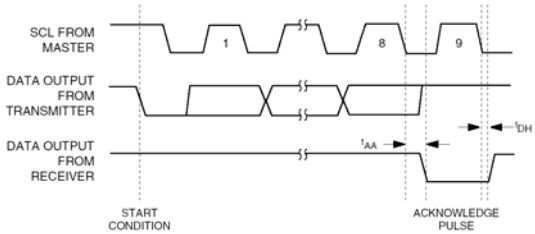
- Start
  - High to low transition of SDA when SCL is high. Generated by Master only.
- Stop
  - Low to high transition of SDA when SCL is high. Generated by Master only.

Data Validity



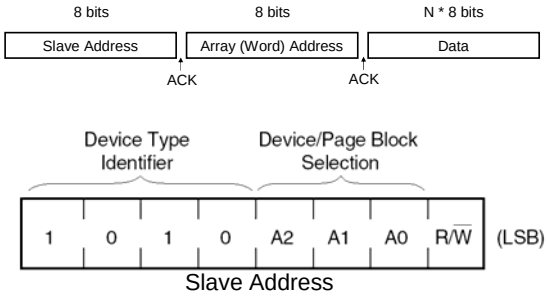
Data must be valid when SCL is high
Data can change when SCL is low

Acknowledge Response from Receiver

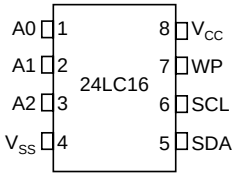


• Lack of an acknowledge from the receiver can indicate an error or that the receiver is busy

I²C Protocol – Addressing



EEPROM Page Block Selection



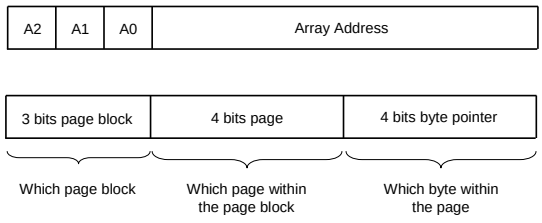
24XX16: 2KB (8 page blocks)
24XX08: 1KB (4 page blocks)
24XX04: 512 bytes (2 page blocks)
24XX02: 256 bytes
24XX01: 128 bytes
24XX00: 16 bytes

24XX16: A2, A1, and A0 are NC (no-connect). Page block is selected using 3 address bits in slave address field of command.

24XX04: A0 is NC (A2:A1 pins identify one of four devices possible on the I²C bus). Device is selected using most significant 2 address bits in slave address byte (these must match the strapping of A2 and A1 pins on the chip). Page block is selected using A0 address bit in slave address field of command.

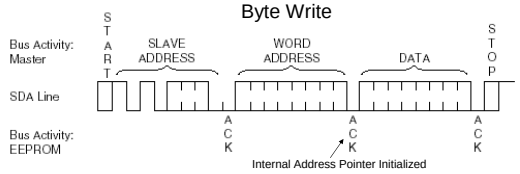
EEPROM Internal Address Pointer

For the 24XX16, an 11-bit internal address pointer uniquely identifies the addressed memory cell within the EEPROM

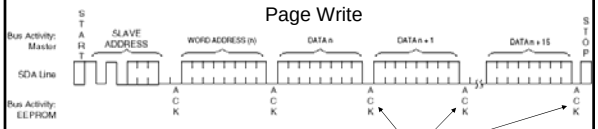


Write Operations

Byte Write

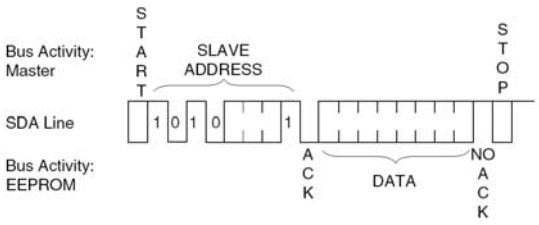


Page Write



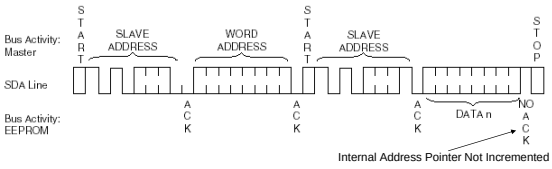
Read Operations

Current Address Read

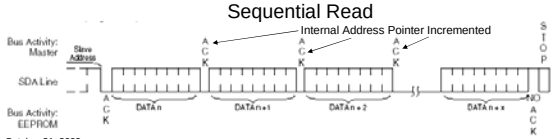


Read Operations

Random Read



Sequential Read



I2C Bit Banging

```
*-------------------------
* I2C BUSS Interface routines
*-------------------------
*
* Generate a START condition & TX char in A
*
?ICstrt  SETB   P3.4     SDA = 1
         SETB   P1.0     SCL = 1
         NOP
         CLR    P3.4     SDA = 0 - START
         NOP
         CLR    P1.0     SCL = 0 - Ready for
                         first bit
*
* Send 8 bit character in A
*
?ICsend  MOV    R7,#8    8 bits per byte
?1       RLC    A        Get next bit to send
         MOV    P3.4,C   Write bit to data line
         SETB   P1.0     Toggle CLOCK high
         NOP             Waste time
         CLR    P1.0     Toggle CLOCK low
         DJNZ   R7,?1    Send all bits
         SETB   P3.4     Release DATA line
         SETB   P1.0     9'th clock
         NOP
         MOV    C,P3.4   Get ACK bit
         CLR    P1.0     Return clock LOW
         RET
```

```
*
* Read 8 bits of data into ACC
*
?ICread  MOV    R7,#8    8 bits per byte
?2       SETB   P1.0     Set clock high
         NOP
         MOV    C,P3.4   Get bit
         RLC    A        Shift it over
         CLR    P1.0     Reset bit
         DJNZ   R7,?2    Do them all
         RET
*
* Send an ACK to the remote
*
?ICack   CLR    P3.4     Zero DATA
         SETB   P1.0     Raise CLOCK
         NOP
         CLR    P1.0     Lower CLOCK
         SETB   P3.4     Release DATA
         RET
```

Note: Other I2C libraries for SDCC are available on the Internet