dsPIC30F Peripheral Module

Inter-Integrated Circuit

or

I^2C Module

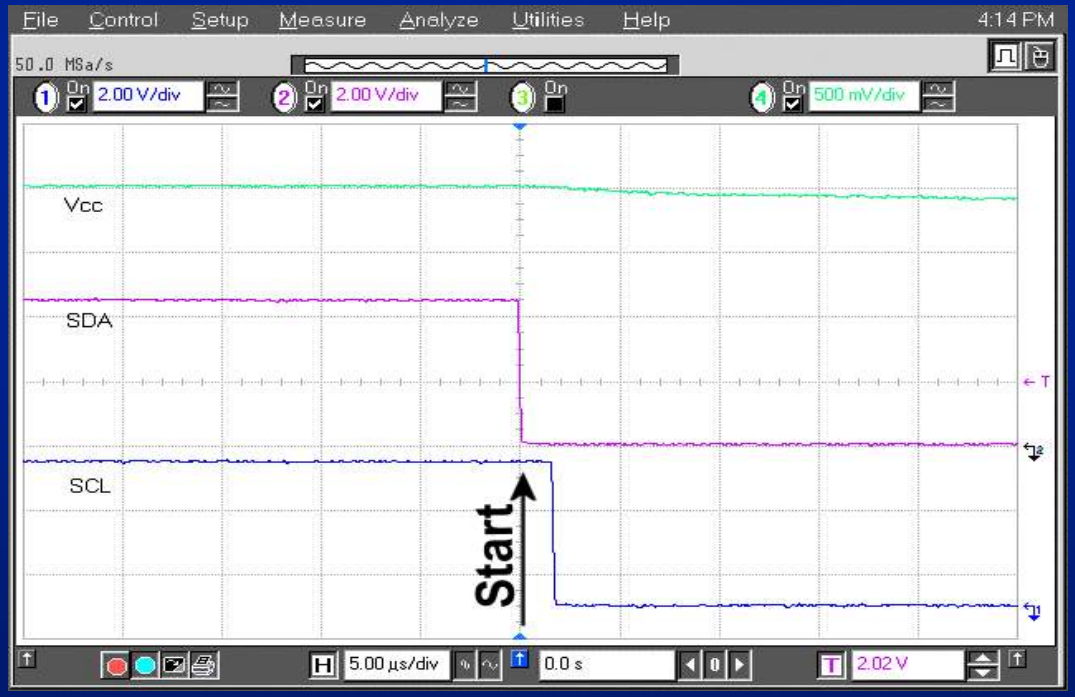
I²C™ Bus Specification

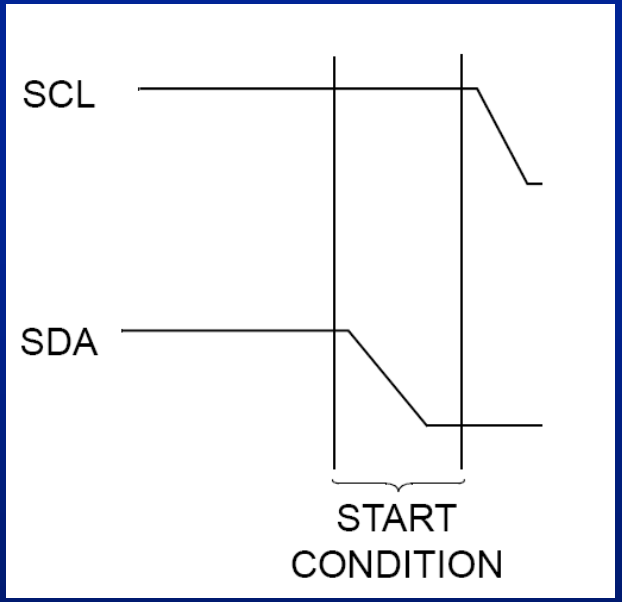
- Developed by Philips.
  - Addressable 2-wire bus capable of addressing 8 serial EEPROM devices in addition to other devices.
  - Defined Start and Stop Conditions.
  - Defined Bus Arbitration.
  - Synchronous Master/Slave system.
  - 100 kHz/400 kHz and 1 MHz bus speeds.
  - SDA – Serial Data.
  - SCL – Clock Line.
  - Both SDA and SCL require pull-up resistors.

I²C™ Start Condition

I²C™ Start Condition

- Used to Initiate the Start of a period of Bus Activity.
  - Defined as a High to Low Transition of SDA while the SCL line is Held High.

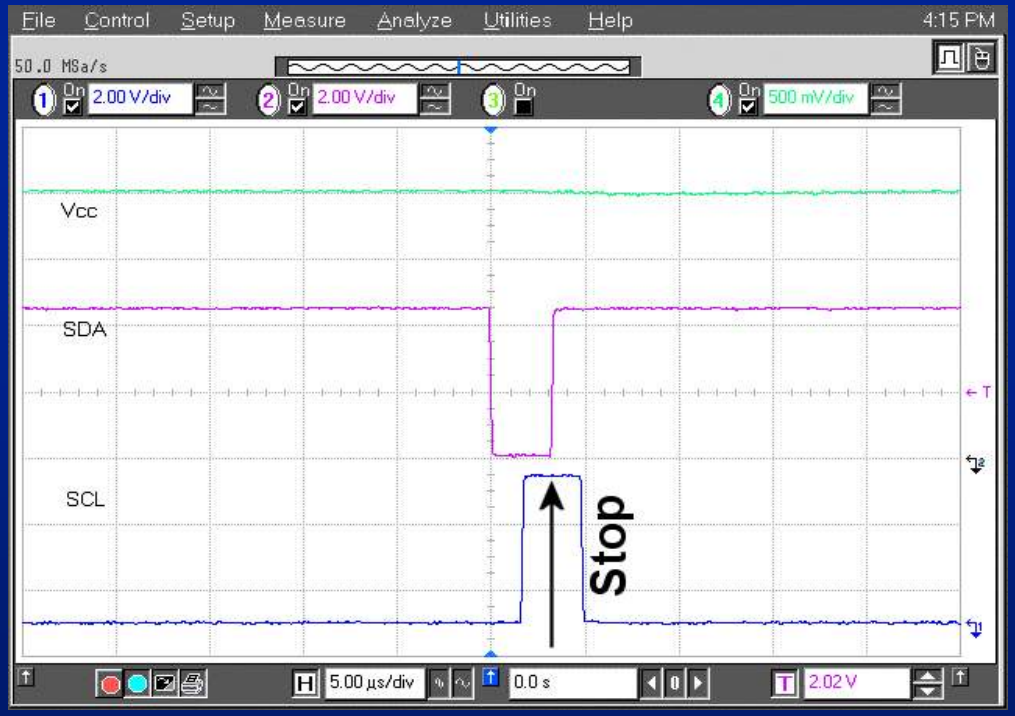


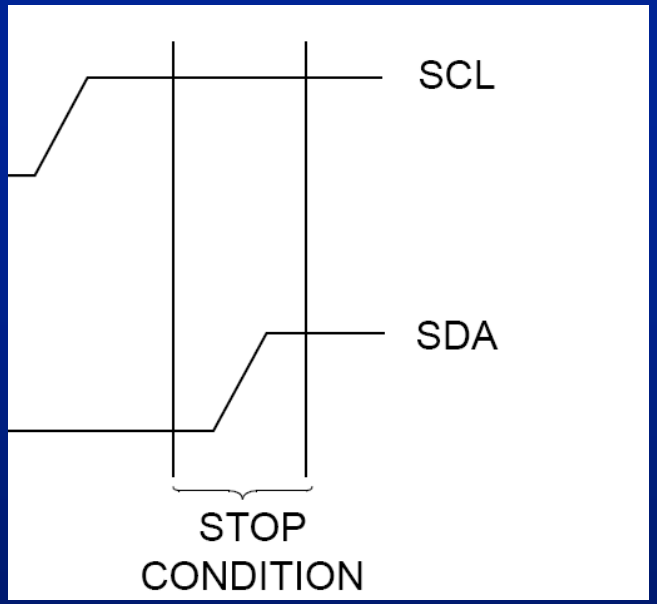


I²C™ Stop Condition

I²C™ Stop Condition

- Used to Signal the End of a period of Bus Activity.
  - Defined as a Low to High Transition of SDA while the SCL line is Held High.

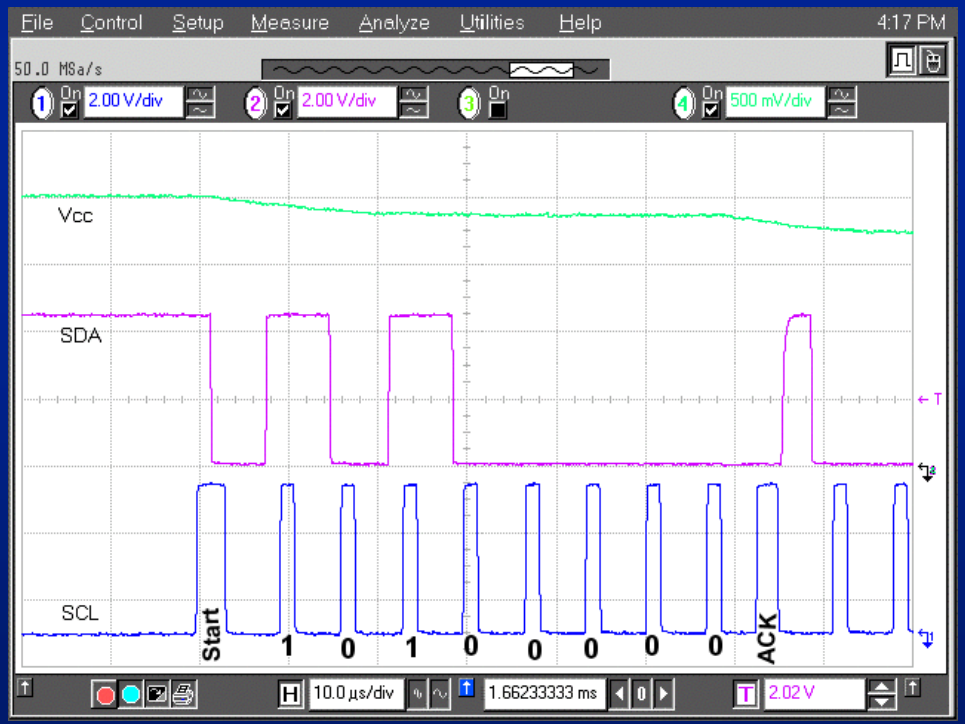


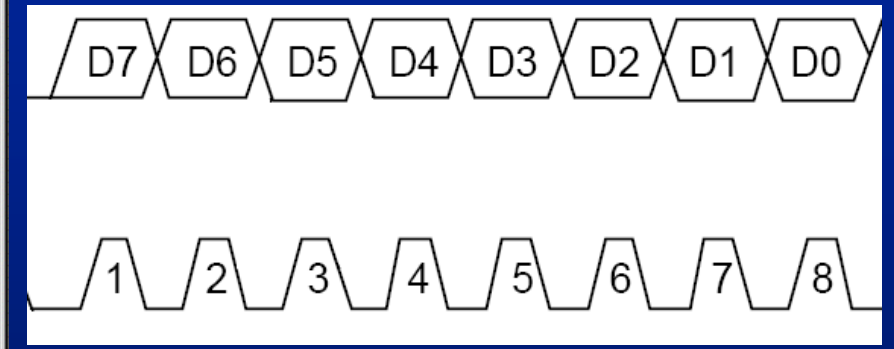


I²C™ Data Transfer

I²C™ Data Transfer

- Data is transferred from the Master to the Slave in blocks of 8 bits.
  - Data is transferred on the rising edge of SCL.

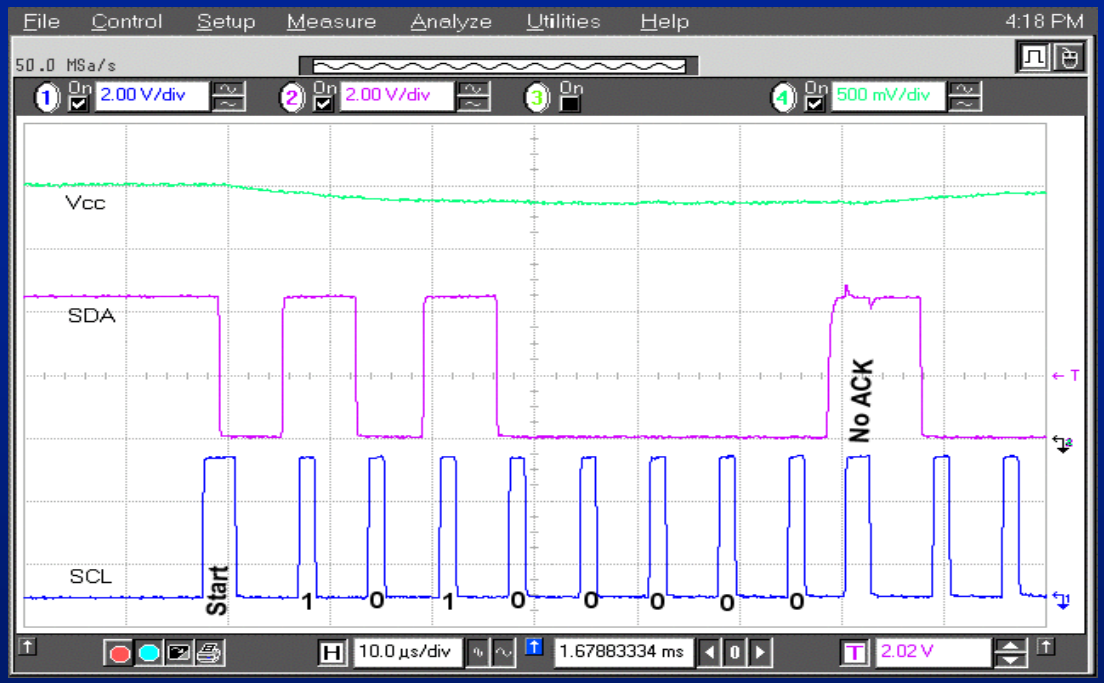


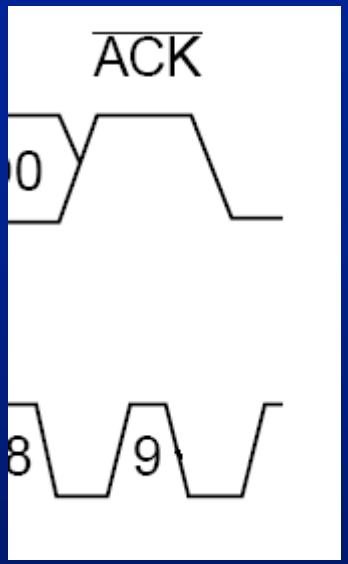


I²C™ ACK/NACK Condition

I²C™ ACK/NACK Condition

- Used to indicate the Success or failure of a data transmission or continuation of an operation.
  - Generated by the Master or the Slave by holding SDA on the 9th rising SCL.

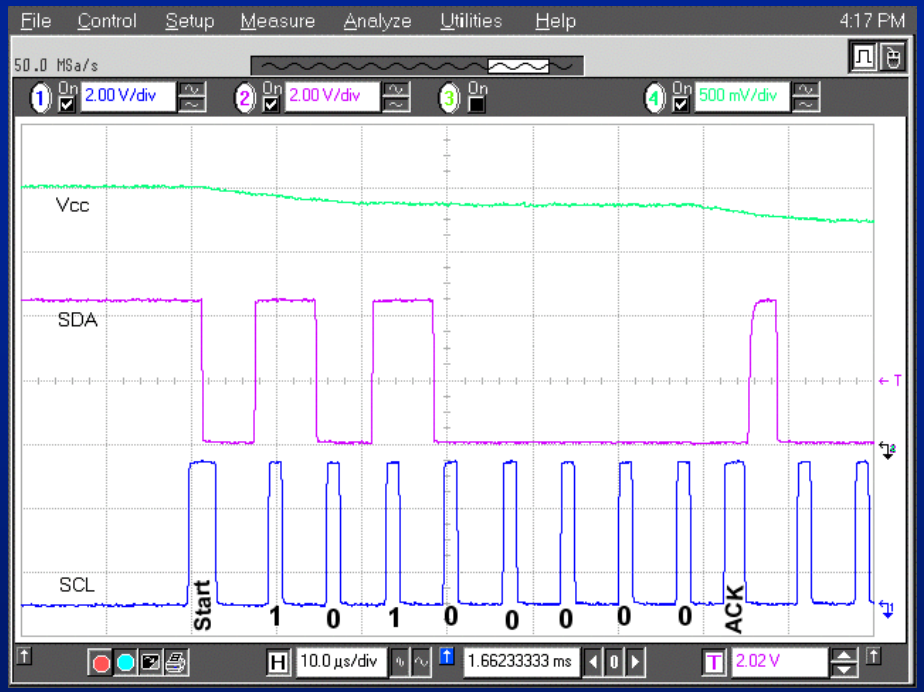


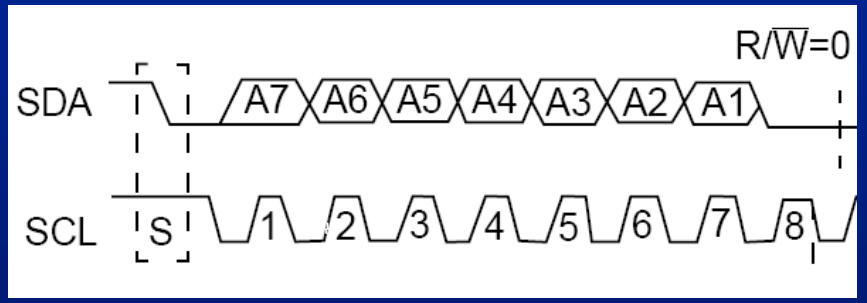


I²C™ Device Addressing

I²C™ Device Addressing

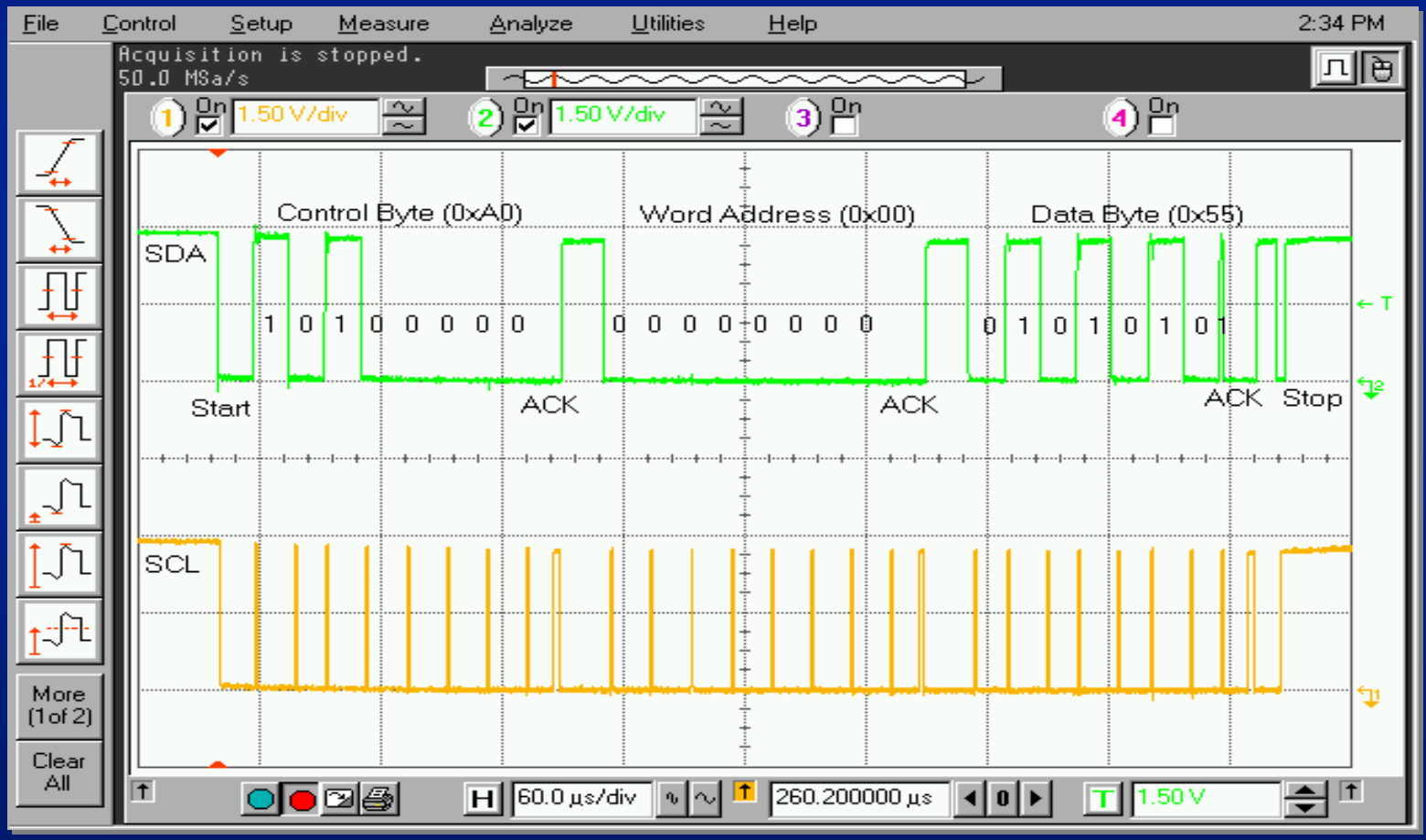
- The first byte received following the Start condition is the control byte.
  - It contains the control code, block- or chip-select bits, and the R/W bit





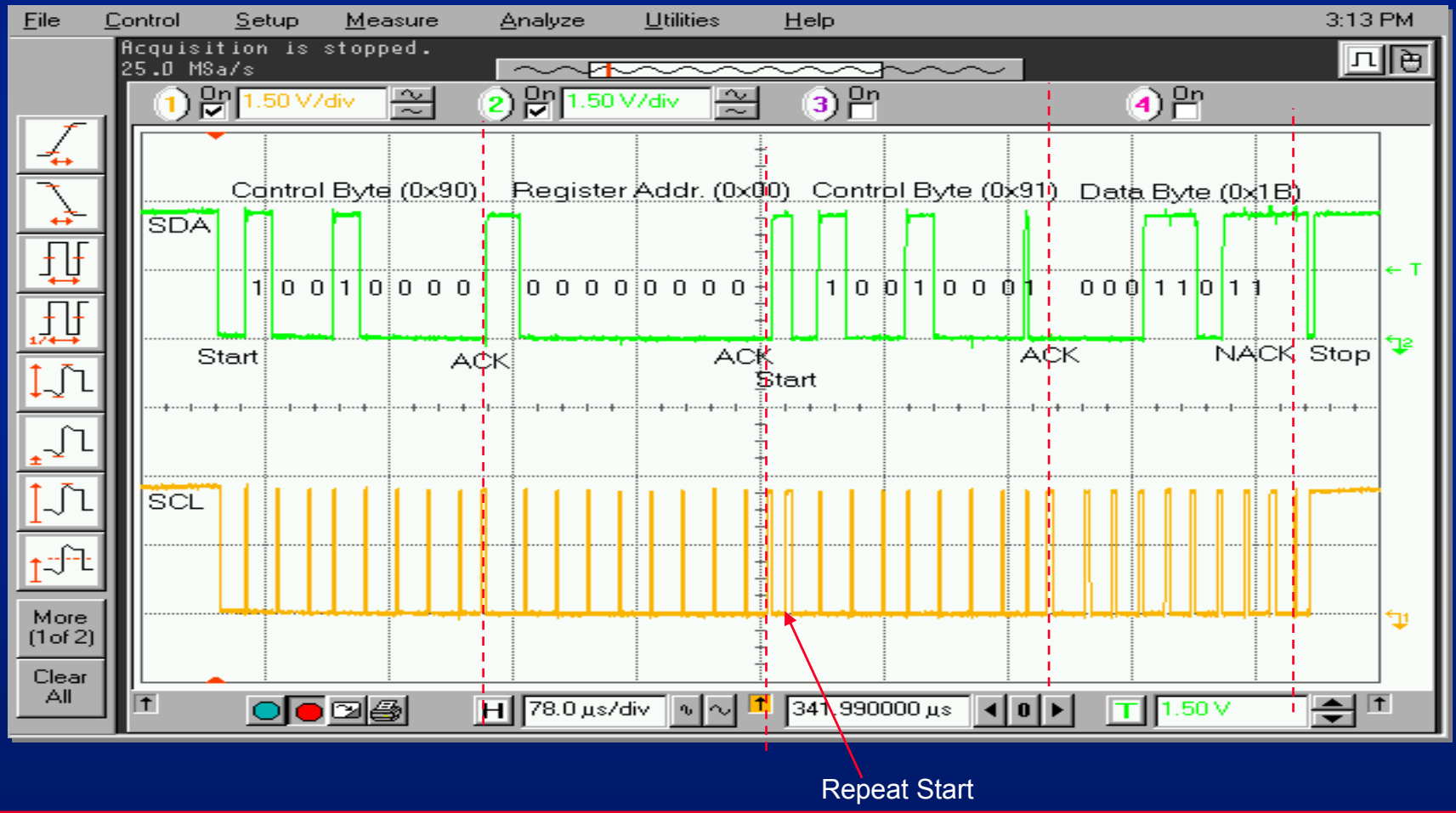
I²C™ Full Data Transfer Example 1

- Below is a full bus cycle for writing to the external EEPROM.



I²C™ Full Data Transfer Example 2

- Below is a full bus cycle for reading from the Slaver.



I²C™ Summary

- Two Wire Bus.
  - DC – 100/400 kHz bus speed up to 1 MHz extended bus speed.
  - Master/Slave configuration
- Start Condition.
  - SDA – High to Low whilst SCL is High
- Stop Condition.
  - SDA – Low to High whilst SCL is High
- ACK/NACK Condition.
  - SDA – held by Master/Slave on 9th Rising SCL

dsPIC30F I²C Module

- dsPIC30F I²C module are
  - As a Signal Master Device
  - As a Master/Slaver in a Multi-Master System
    - Bus collision detection and arbitration available
  - As a Slave Device
- I²C Master and Slaver logic each generated Interrupt independent.
- Dedicate I²C baud rate generator for the Master

dsPIC30F I²C Features

- Independent Master and Slaver design
- Multi-Master support, No Message lost in arbitration
- Detects 7-bit and 10-bit address mode
- Automatic SCL stretching for process to respond of the slave data request
- Supports both 100KHz and 400KHz bit rate
- Detects general call address

I²C Registers

- I²C module has the following memory-mapped registers:
  - I2CTRN - Data Transmit Register (8 bits)
  - I2CRCV - Data Receive Register (8 bits)
  - I2CBRG - Baud Rate Generator (9 bits)
  - I2CCON - I²C Control Register (16 bits)
  - I2CSTAT - I²C Status Register (16 bits)
  - I2CADD - Slaver Address Register (10 bit)

I²C - Baud Rete Generator

- Dedicated 9-bit Baud Rate Generator
- Baud Rate set by I2CBRG register
  - **Baud Rate = Fcy /(I2CBRG + 1)**
  - Fosc=120MHz, 400KHz bps = 30 Fcy / (I2CBRG + 1)
    - I2CBRG = 299 = 0x12B
- Bits are transmitted and/or received at
- the rate defined by the Baud Rate as
- calculated above
  - Maximum Baud Rate is 1 MHz

I2CBRG Register

-	-	-	-	-	-	-	BR8
bit15	14	13	12	11	10	9	bit8

BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
bit7	6	5	4	3	2	1	bit0

I2CCON Register

I2CCON Register

I2CEN	-	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit15	14	13	12	11	10	9	bit8

- I2CEN : I2C Enable bit
- I2CSIDL : STOP or IDLE mode selection
- SCLREL : SCL Release Control bit, write “0” to stretch wait delay (Slaver)
- IPMIEN : Enable IPMI support module bit
- A10M : 10-bit Slave Address bit
- DISSLW : Disable Slew Rate Control bit
- SMEN : SMBus Input Level bit
  - SMEN = 1 , Thresholds compliant with SMBus specification

I2CCON Register (cont.)

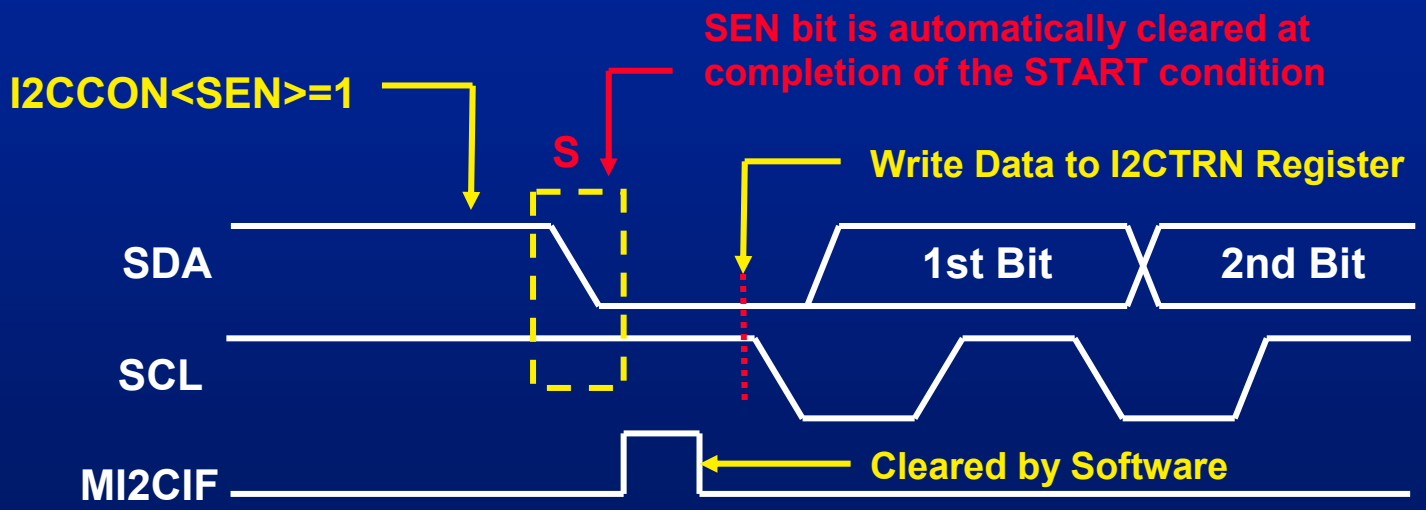
I2CCON Register

GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit7							bit0

- GCEN : General Call Enable (Slaver Mode)
- STREN : SCL clock stretch Enable bit (Slaver Mode)
- ACKDT : Generate Acknowledge Data bit (Master mode)
- ACKEN : Acknowledge Sequence Enable bit (Master Mode)
- RCEN : Enable Receive mode for I2C
- PEN : STOP Condition Enable (Master Mode)
- RSEN : Repeated START Condition Enable bit (Master Mode)
- SEN : START Condition Enable bit (Master Mode)

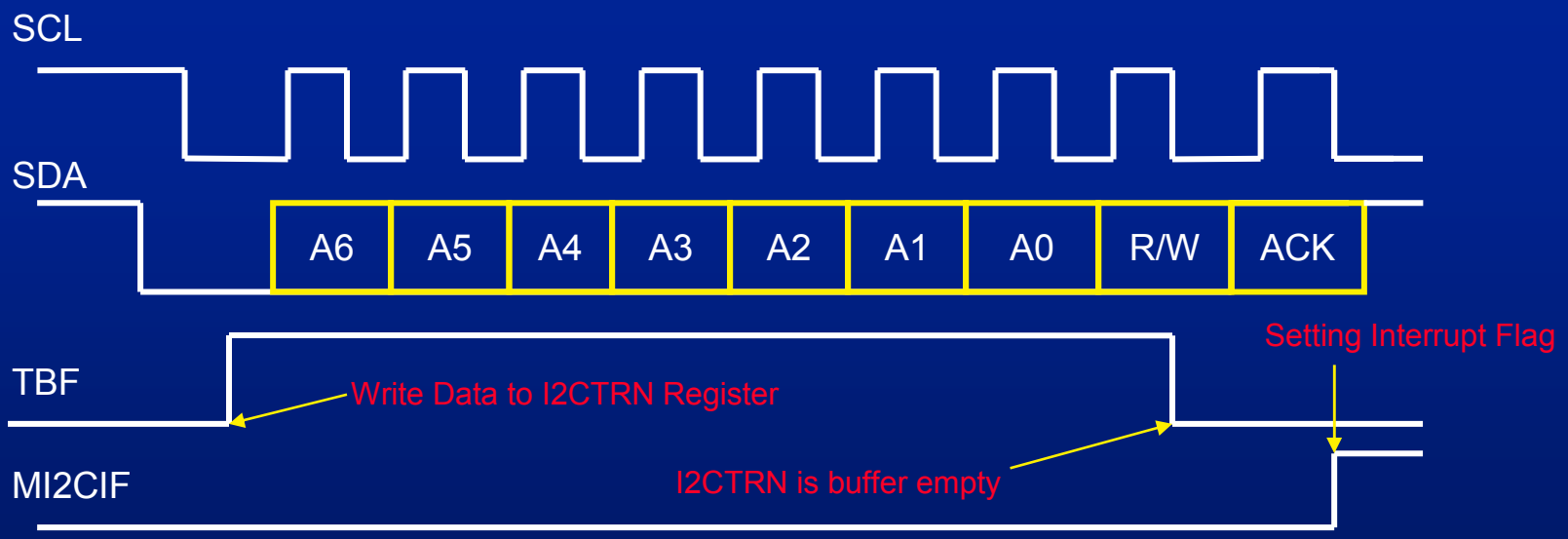
I2C - Message Sequence

- I2C module enabled by setting theI2CEN bit in I2CCON register
- Message begins with START condition
  - SDA pin goes high to low, SCL stays high
  - MI2CIF will be generated at completion



I2C - Message Sequence (contd.)

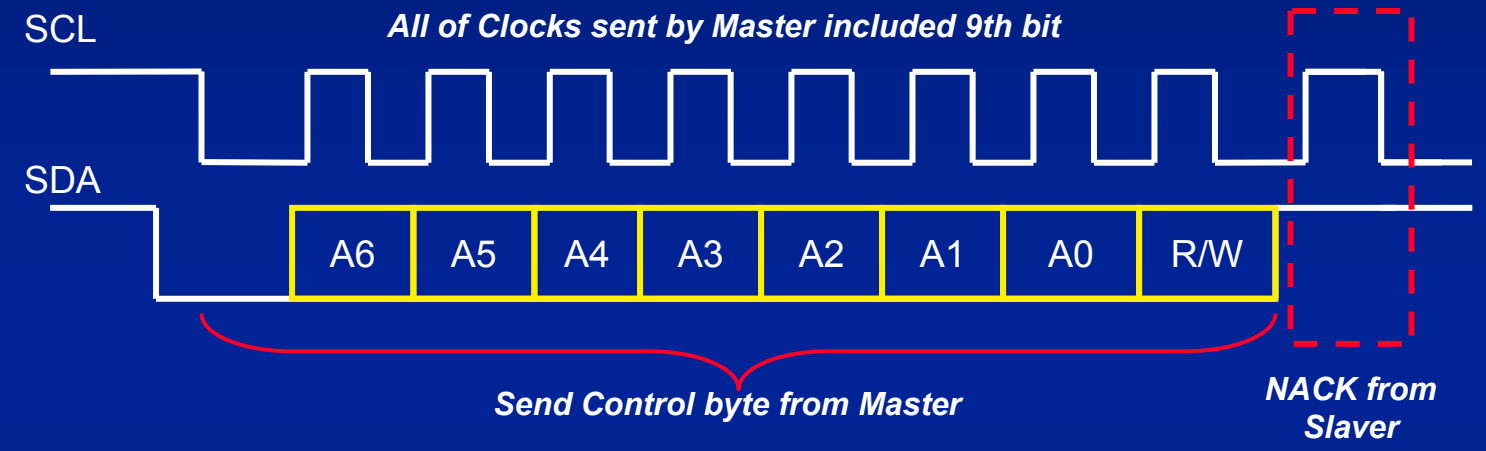
- Slave is activated when Master writes the7-bit Slave address to I2CTRN
  - Transmitted R/W bit indicates data direction
    - Slave Transmission (R/W = 1) or Slave Reception (R/W = 0)
  - The byte in I2CTRN is transmitted

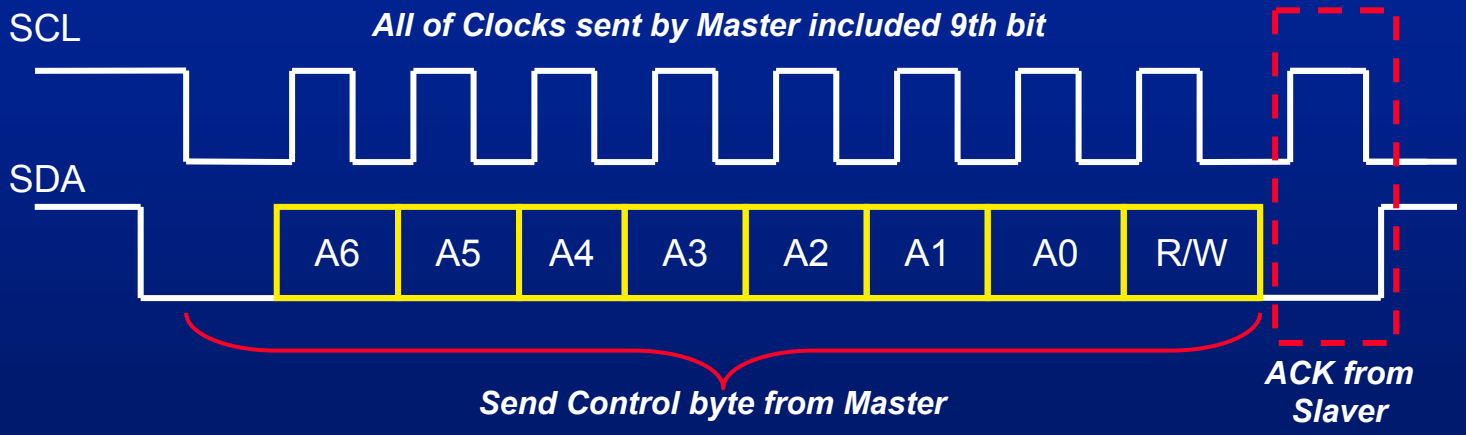


I2C - Message Sequence (contd.)

- Slave address is defined by the value written to the I2CADD register of Slave
  - Address received by Slave in I2CRCV register only if it matches I2CADD
  - Slave automatically acknowledges
- every byte received from Master
  - If address match occurred, Slave pulls SDA low during 9th bit time (ACK)
  - If address did not match, Slave lets SDA remain high (NACK)

I²C – Control Byte Timing



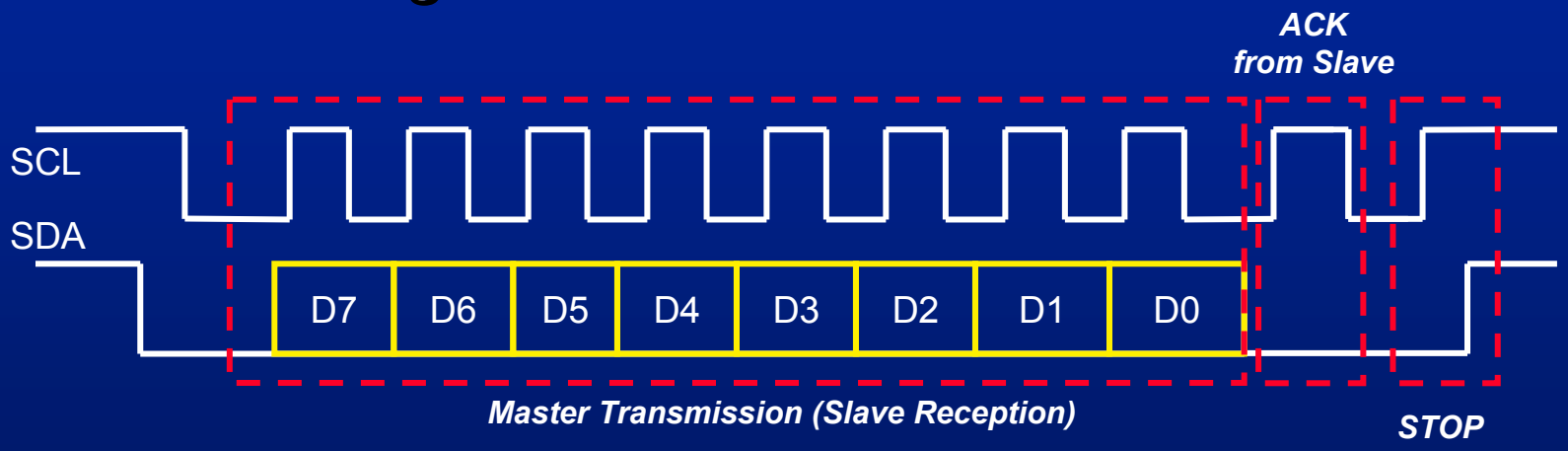


ACK / NACK Status

- Master can determine ACK/NACK status by polling the ACKSTAT bit in I2CSTAT register
- Slave can determine if last byte received was address or data, by polling the D_A bit in I2CSTAT
  - If it was an address, Slave can determine value of received R/W bit by polling the R_W bit in I2CSTAT

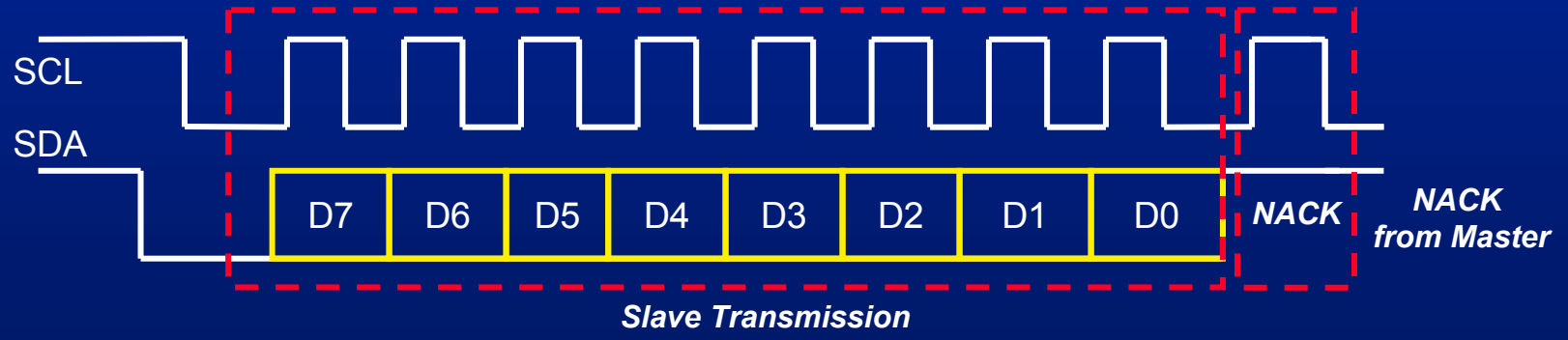
Master Send a Data to Slave

- Slave Reception (if R/W = 0)
  - Slave reads each data byte from its I2CRCV register
  - Slave continues receiving bytes until Master initiates a STOP condition by setting the PEN bit in I2CCON



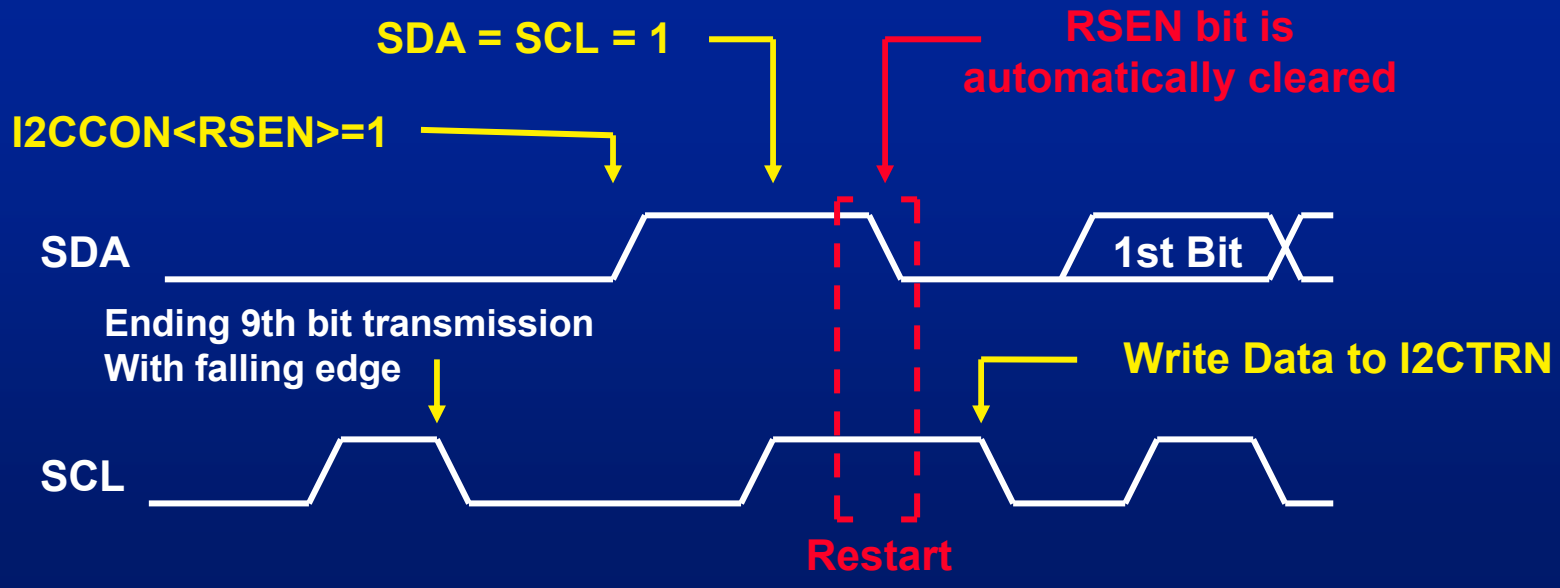
Slave Send Back a Data to Master

- Slave Transmission (if R/W = 1)
  - Slave writes each data byte to I2CTRN
  - Master software sets ACKEN bit in I2CCON
    - ACKDT bit in I2CCON selects ACK/NACK
    - ACKEN bit is automatically cleared
    - A MI2CIF interrupt will be generated
  - If Slave receives ACK, it transmits next byte
  - If Slave receives NACK, it stops transmitting



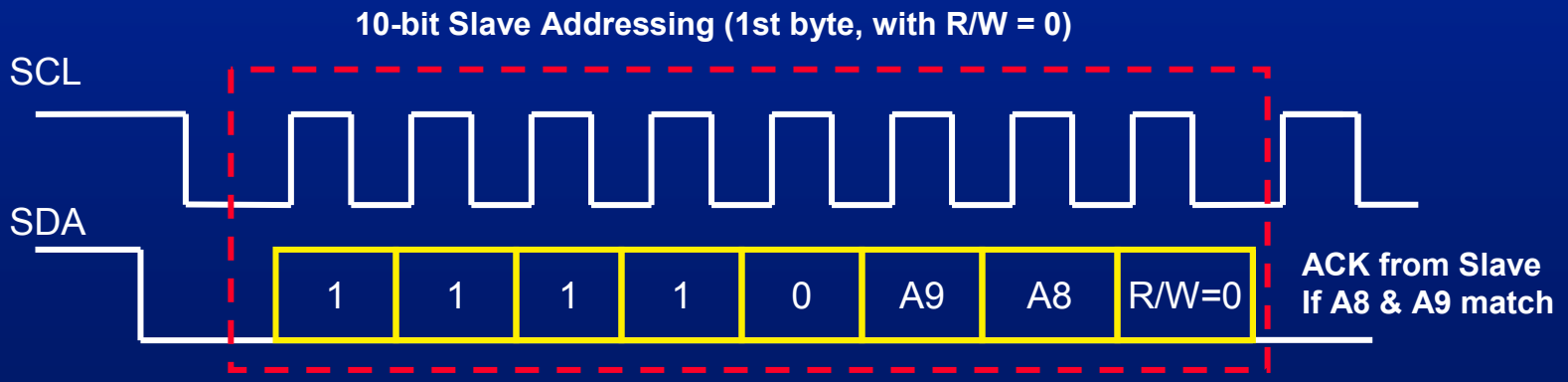
Repeated START Condition

- Direction of data transfer can be changed by Master initiating a 'Repeated START' condition followed by a Slave Address byte with suitable R/W bit
  - Repeated START condition is initiated by setting the RSEN bit in I2CCON



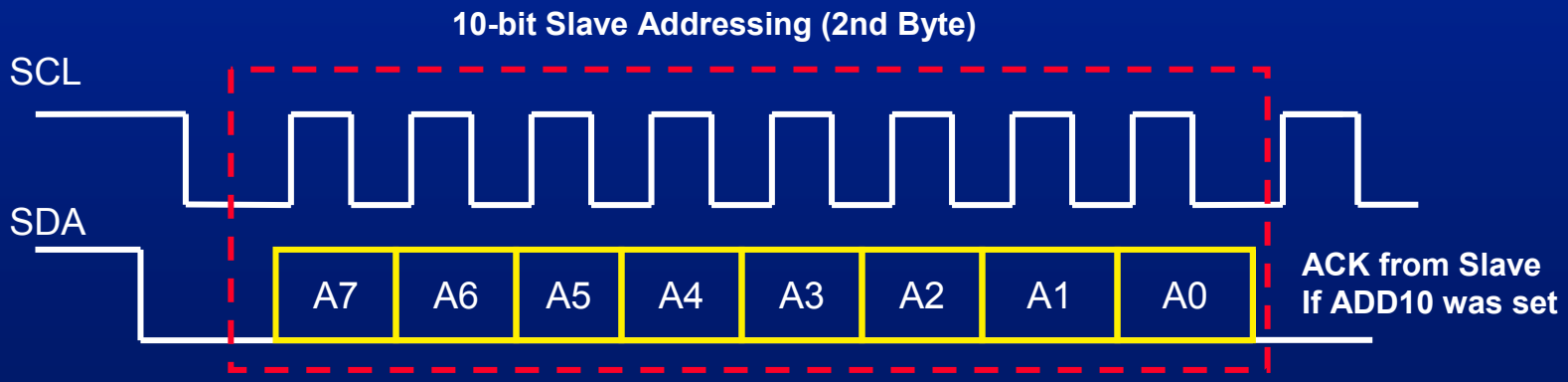
I²C – 10-bit Address Mode the Upper Byte (R/W=0)

- 10-bit Addressing enabled when Master sets A10M control bit in I2CCON
- Master can start address sequence by transmitting upper 2 bits of 10-bit address
- If A9 and A8 match the corresponding bits in Slave's I2CADD, it waits for next byte
  - pre-defined upper address preamble '11110', the upper two bits of the 10-bit Slave address, and a cleared Read-Write bit.



I2C – 10-bit Address Mode the Lower Byte

- Master then transmits lower byte of address
  - If this byte matches lower 8 bits in I2CADD, ADD10 bit in I2CSTAT is set
  - For Slave Reception, Master can start sending data bytes
  - For Slave Transmission, Master must initiate a Repeated START condition

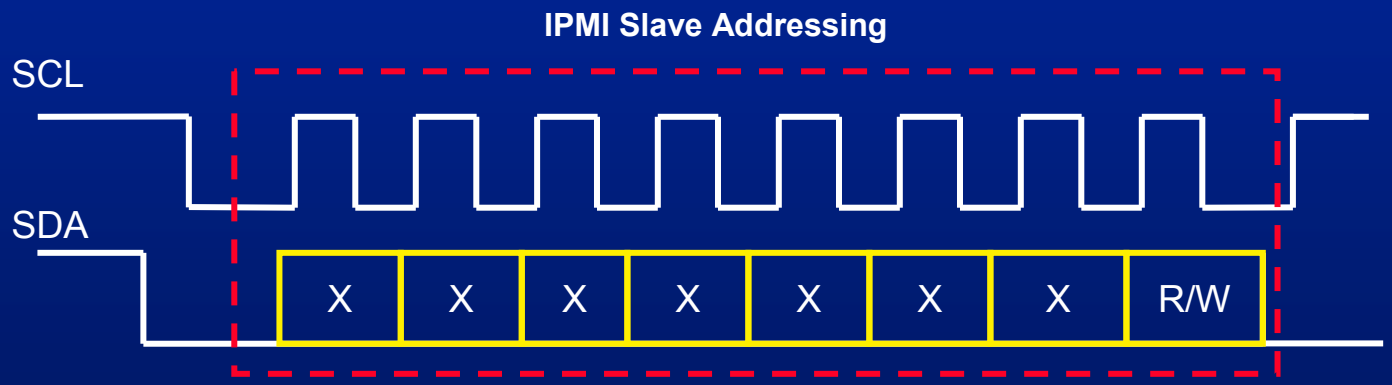


I2C - General Call Addressing

- General Call (GC) Addressing
  - General Call Address (0x00) can be used by Master to address all Slaves, irrespective of I2CADD values
  - Enabled for a Slave by setting the GCEN bit in I2CCON
  - Master must transmit R/W = 0 (Slave Reception) in this mode
  - Address Match status can be determined by polling GCSTAT bit in I2CSTAT

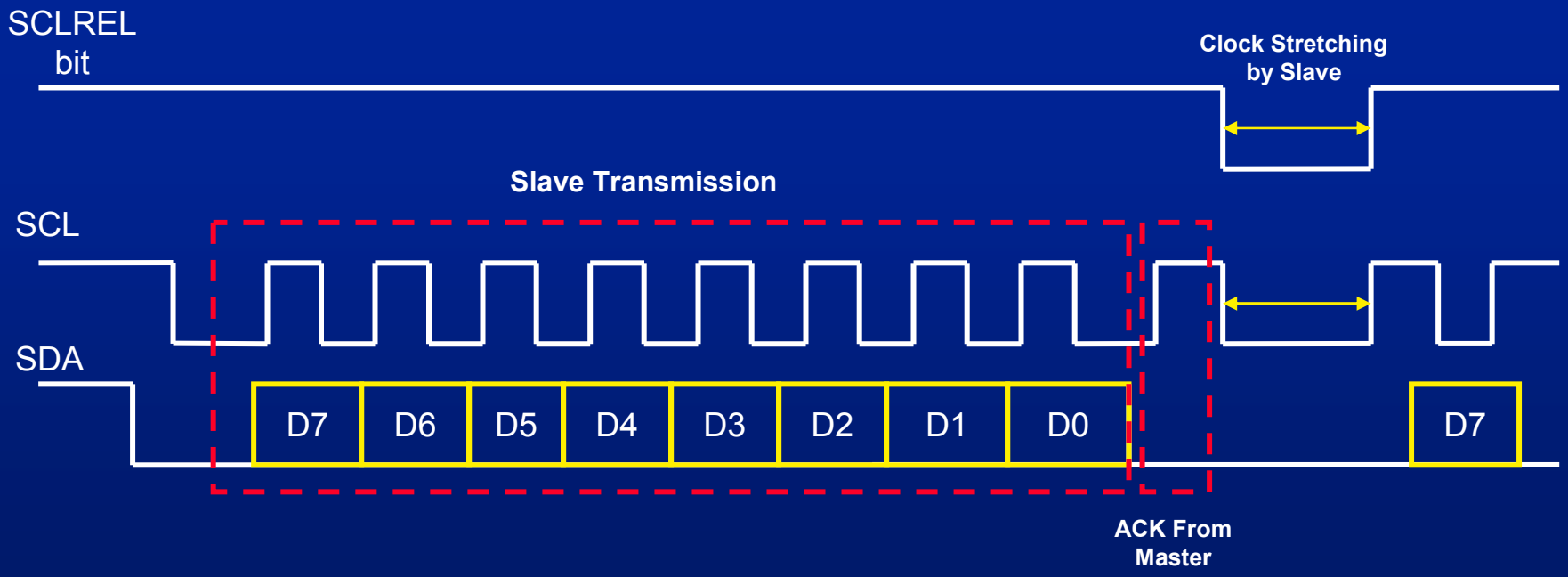
I2C - IPMI Addressing

- Intelligent Peripheral Management Interface (IPMI) Addressing
  - Can be used to configure an I2C Slave to accept and respond to all addresses
  - Enabled for a Slave by setting the IPMIEN bit in I2CCON



Clock Stretching during Slave Transmission

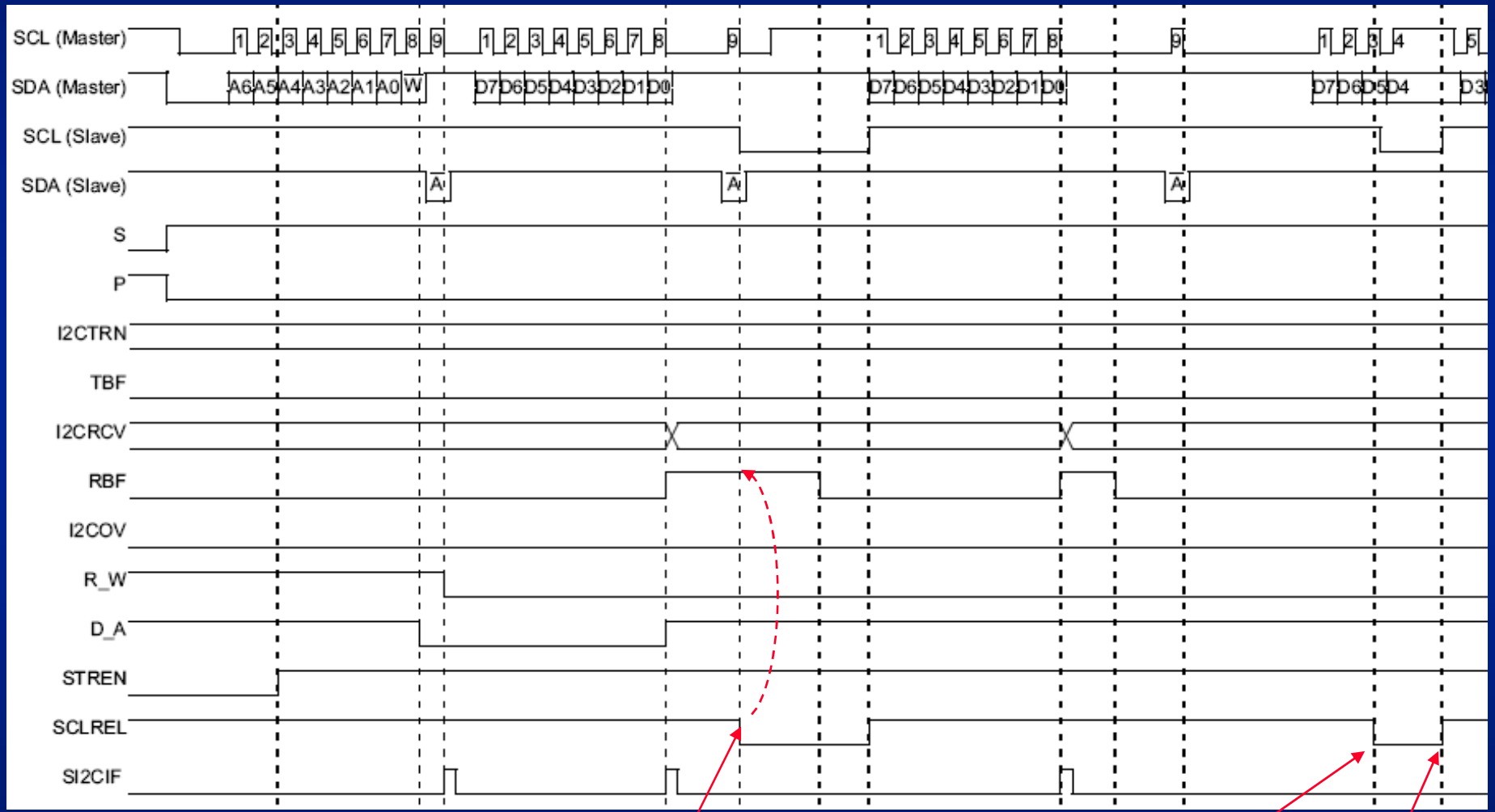
- STREN bit must be set “1” to Enable
  - When Slave receives an Address Byte with R/W = 1 or an ACK, SCLREL bit in I2CCON gets cleared, which holds SCL low
  - After writing I2CTRN, Slave must set SCLREL



I2C - Clock Stretching during Slave Reception

- If STREN bit in I2CCON = 1, when Slave receives a byte, the SCLREL bit gets cleared, which holds SCL low
  - This prevents any further data transmission by the Master, thereby giving the Slave software time to read the received data
  - After reading the data in I2CRCV, Slave software must release SCL by setting the SCLREL bit
  - STREN = 1 also allows software-controlled stretching of the clock

Clock Stretching Enabled



I2C - Error Conditions

- Write Collision
  - Occurs if the user attempts to write to I2CTRN while I2C module is busy
  - Indicated by IWCOL bit in I2CSTAT
  - Avoided by writing to I2CTRN only when TBF bit in I2CSTAT is clear
- Receive Overflow
  - Occurs if a new byte has been received whileI2CRCV is still holding previous byte
  - Indicated by I2COV bit in I2CSTAT
  - Avoided by reading I2CRCV as soon as RBF bit in I2CSTAT is set, before new byte is shifted in

I2C - Error Conditions (contd.)

- Master Bus Collision
  - Bus Collision typically occurs in a Multi-Master configuration, when one Master lets the SDA line float high and another Master drives SDA low
    - Only one Master can control the bus at a time
    - The Master that first pulls SDA low will win Bus Arbitration
    - The other Master will go into an idle state and set its BCL bit in I2CSTAT
  - Bus Arbitration and Collision Detection is performed by the module during START, Repeated START,STOP, ACK, Address and Data sequences

I2C - Interrupts

- Master Interrupt
  - Indicated by MI2CIF bit and enabled by MI2CIE bit
  - Following events cause MI2C interrupt
    - Successful completion of START, RESTART or STOP condition
    - Transmission of ACK/NACK by Master
    - Data byte transmitted or received
    - Bus Collision

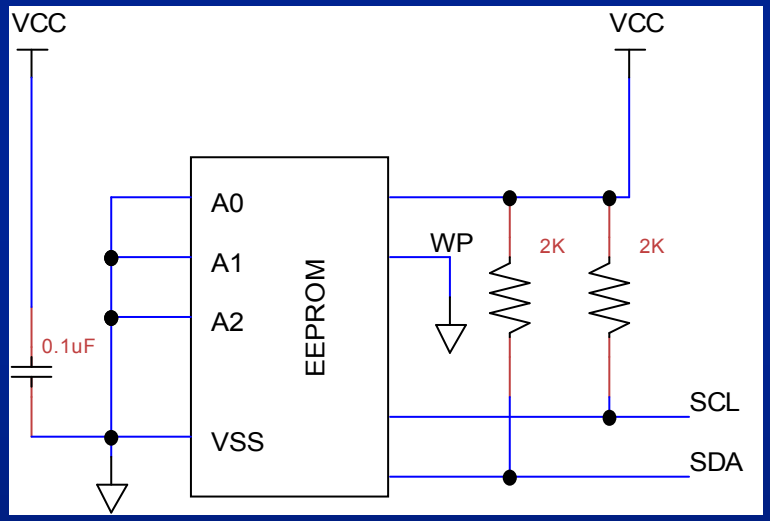
I2C - Interrupts (contd.)

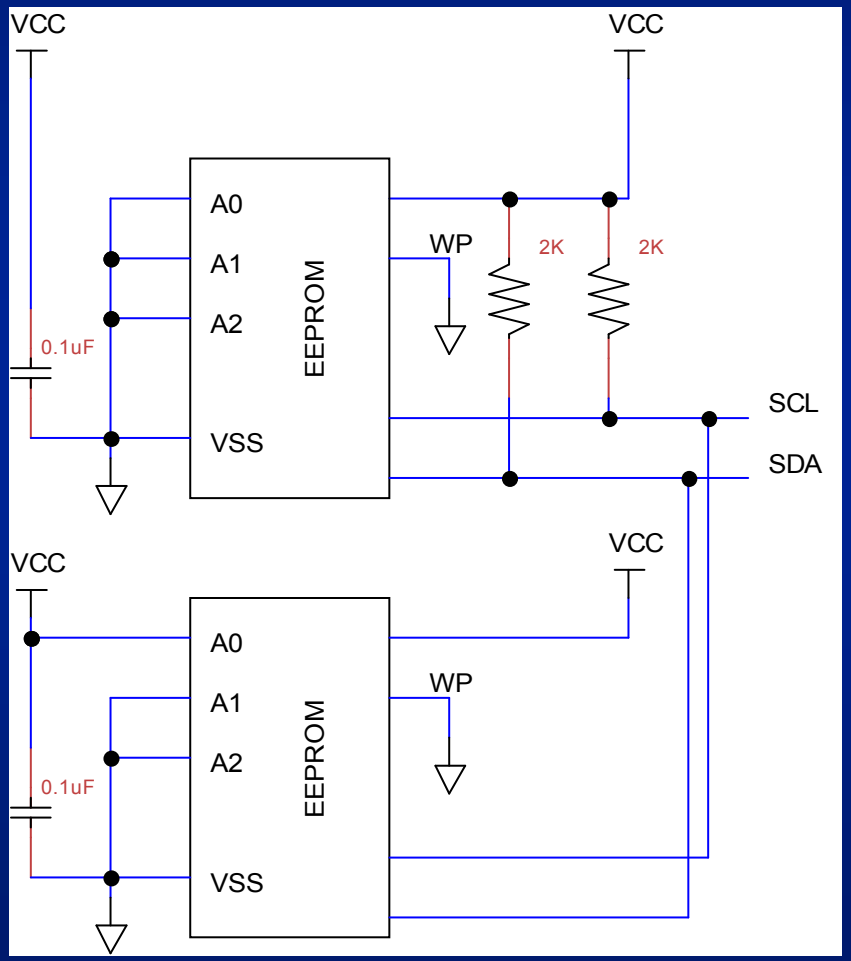
- Slave Interrupt
  - Indicated by SI2CIF bit and enabled by SI2CIE bit
  - Following events can cause a Slave I2C interrupt
    - Detection of a valid Slave address byte
    - Reception of Slave Transmission request from Master
    - Reception of a data byte from Master

I2C - Integrated Signal Conditioning

- Slew Rate Control
  - For 400 kHz operation, bus specification requires slew rate control of device pin out
  - Automatic Slew Rate Control
  - Disabled by setting DISSLW bit in I2CCON
- SMBus Support
  - For System Management Bus (SMBus),voltage levels are 0.2*VDD and 0.8*VDD
  - Module automatically complies with SMBus input levels if SMEN bit in I2CCON is set

I²C™ Connect with EEPROM





I²C™ EEPROM H/W

- Pull up Resistor on SCL ensures power up into standby (and is required for multimaster)
- Address pins can be any combination
  - all ‘0’ is easiest
  - Extra address pin for expand devices
- WP=0 allows for writes to occur
  - WP high makes a serial ROM
- Pull up on SDA needs to be set for bus speed
  - Use 10KΩ for 100 kHz or less
  - Use 2KΩ for higher speed (or less if bus capacitance is high)

Jumper Setting for Lab1

- Caption !!
  - I2C, SPI, UART and PGC/PGD share both pin 25 & 26
  - Need to use the jumper to change the debug pin to EMUC1 and EMUD1 (default are PGC & PGD)
  - On the Configuration Bits select “Use the EMUC1 and EMUD1)
- So,
  - Program Mode : Set DSW1 1&2 to ON, 3&4 are OFF position
  - Debug Mode : Set DSW1 1&2 to OFF, 3&4 are ON position

I²C Lab1 (Master Mode)

- Initialize the dsPIC30F4011 to an I²C Master at 100K bps
- Set 1 Second time base using Timer1 to read the ADC result form both VR1 and VR2
- Put ADC Value on the LCD module (MSB Only)
- Use the EEPROM command to write the ADC value to the external EEPROM (24LC04B) at location 0x10 (VR1) and 0x20 (VR2) address
- Read 24LC04B with I2C Read Command to read EEPROM data from 0x10 and 0x20 then display on the LCD Module at Second Line

I²C Lab2 (Slave Mode)

- Initialize the dsPIC30F4011 to an I²C Slave Device
- Use the Interrupt to receive the I2C data
- Master send out both value for VR1 and VR2 to Slave, Slave side has to receive the date with 64 bytes receiver buffer (simulate EEPROM data area)
- Slave display received data on the LCD line 2
- Slave has capability simulate the 24LC04B that cab be Read/Write thorough I2C command