برمجة المتحكمات المصغرة

التجارب العملية

الجلسة الحادية عشرة

Programming

Embedded Systems Microcontroller

You Can Practice Microcontroller Programming Easily Now!

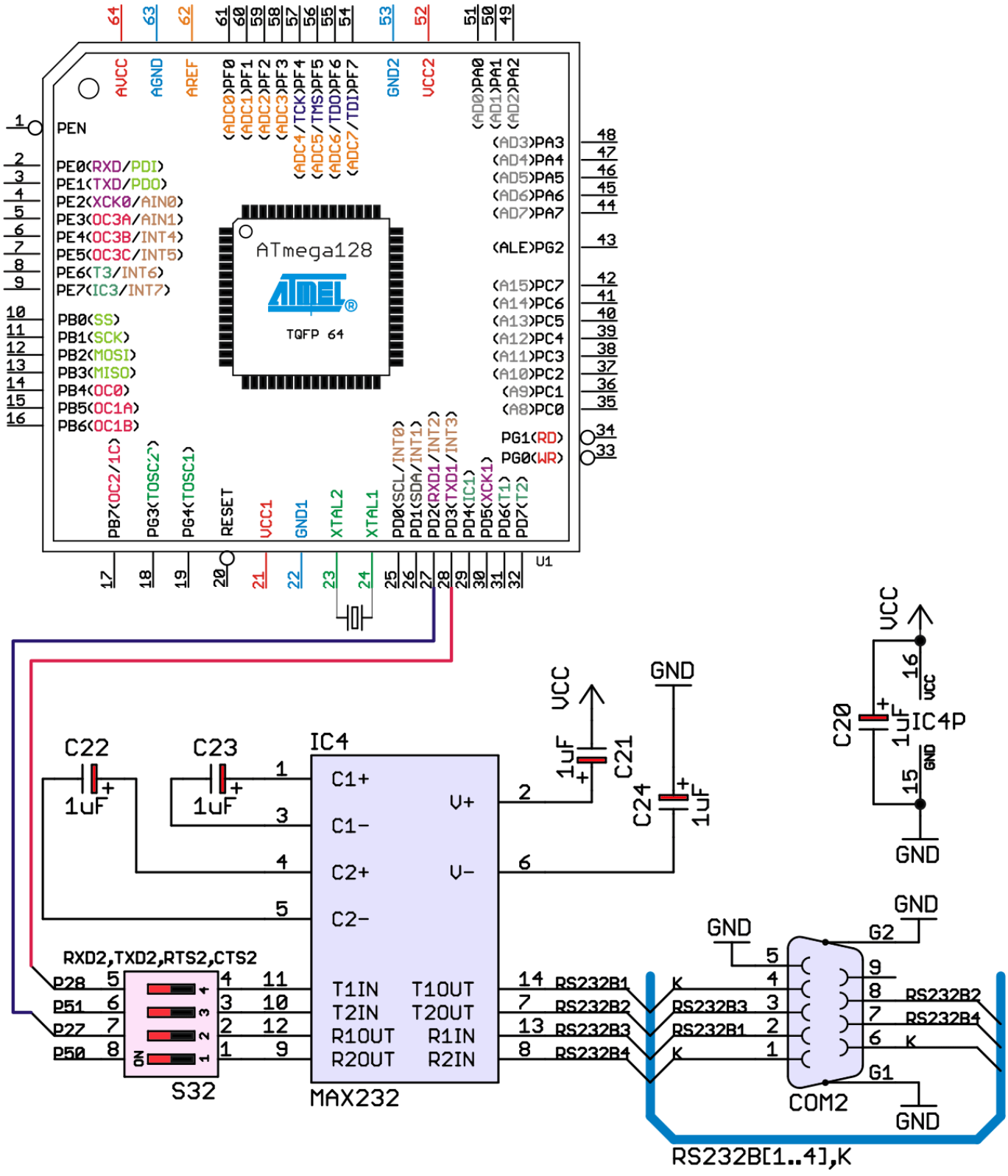
WALID BALID, Tuesday, December 15, 2009

Exp.30: Programming Internal EEPROM التجربة الثلاثون: برمجة الذاكرة الداخلية EEPROM

الغاية من التجربة:

استثمار وبرمجة ذاكرة المعطيات الداخلية (EEPROM) للمعالج.

مخطط الدارة:



شرح عمل الدارة:

تحوي الدارة أعلاه على دارة ملائمة بين النافذة التسلسلية UART1 للمتحكم المصغر ومنفذ الاتصال التسلسلي RS232 للحاسب. سوف نقوم بكتابة برنامج للقراءة والكتابة من وإلى ذاكرة المعطيات الداخلية للمعالج (EEPROM) باستخدام النافذة التسلسلية.

التعليمات الجديدة:

التعليمة	شرح التعليمة
Dim Var **As** [**xram**|Sram|**Eram**] Type [**at** Location][**overlay**] *Examples:* Dim Sram_var As Sram Byte At &H10 Dim Eprm_var As Eram Byte At &H80	تعريف متحول في ذاكرة (معطيات خارجية|معطيات داخلية|ذاكرة دائمة) مع خيار تحديد موقع المتحول في الذاكرة. **overlay**: خيار يجعل المتحول المعرف متحول خيالي في الذاكرة أي أنه لن يتم حجز مساحة في الذاكرة ولكن تكمن الخطورة في كتابة متحول على العنوان.
Writeeeprom Var , Address	كتابة قيمة (Var) إلى الذاكرة EEPROM عند العنوان Address.
Writeeeprom Var , Label	كتابة قيمة (Var) إلى الذاكرة EEPROM لتتوضع عند اللافتة Label.
Readeeprom Var , Address	قراءة قيمة (Var) من الذاكرة EEPROM عند العنوان Address.
Readeeprom Var , Label	قراءة قيمة (Var) من الذاكرة EEPROM المتوضعة عند اللافتة Label.
$eeprom	توجيه المترجم إلى تخزين البيانات الموجودة في التعليمة DATA والتي هي مباشرة بعد التوجيه في الذاكرة EEPROM، وسيتم توليد ملف ثنائي EPP.
$data	توجيه المترجم إلى أن البيانات التالية سيتم تخزينها في ذاكرة البرنامج.
$eepromhex	توجيه المترجم إلى تخزين البيانات في الملف EPP بصيغة Intel HEX. هذا التوجيه يجب أن يستخدم مع التعليمة $eeprom.
$eepleave	توجيه المترجم إلى عدم توليد ملف EPP وعدم محي الذاكرة EEPROM.
$default Sram | Xram | Eram *Examples:* $default Sram Dim A As Byte , B As Byte $default Eram Dim C As Byte , D As Byte	تعيين موقع حجز جميع المتحولات التالية للتوجيه $default.
$end $default	استعادة الإعدادات الافتراضية التي تم تغييرها باستخدام التوجيه السابق.
$noramclear	توجيه المترجم إلى عدم تصفير محتوى الذاكرة SRAM عند التهيئة.
$romstart = address	توجيه المترجم إلى تخزين البرنامج ابتداءً من العنوان المحدد.

ملاحظة: يوصى في الوثيقة الفنية لعائلة AVR بعد استخدام البايت الأول من الذاكرة EEPROM المتوضع عند العنوان صفر لأنه يمكن أن تتغير قيمة هذا البايت أثناء تصفير المعالج أو أي حالة عابرة.

برنامج تشغيل الدارة (1):

```
$regfile = "m128def.dat"
$crystal = 4000000
$baud = 9600
'------------------
Dim B  As Byte , I As Byte
Dim W  As Word , S As String * 5

Dim Eb As Eram Byte At 13
Dim Ei As Eram Integer At 14
Dim El As Eram Long At 16
Dim Es As Eram String * 5 At 20
'------------------
Do
   S = "ABCDE" : Es = S
   S = ""
   S = Es : Print S

   B = 10 : Eb = B
   B = 0
   B = Eb : Print B
'/------

   For I = 0 To 4
      Readeeprom B , I
      Print B
   Next I
'/------

   S = "abcde" : W = 10000
   Writeeeprom S , 5
   Writeeeprom W , 11

   S = "" : W = 0
   Readeeprom S , 5 : Print S
   Readeeprom W , 11 : Print W
'/------

   Restore Lbl
   Read B : Print B
   Read B : Print B
Loop
End
'------------------

Lbl:
Data 10 , 12
'------------------

$eeprom
   Data 1 , 2 , 3 , 4 , 5
$data
```

التوجيهات.

تعريف متحولات في الذاكرة SRAM

تعريف متحولات في الذاكرة EEPROM

إسناد قيمة متحول في الذاكرة SRAM إلى متحول في الذاكرة EEPROM.

قراءة قيمة عند عنوان محدد في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

إسناد قيمة متحول في الذاكرة SRAM إلى عنوان محدد في الذاكرة EEPROM.

قراءة قيمة عند عنوان محدد في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

تحميل قيم إلى الذاكرة SRAM عند لافتة محددة ومخزنة في ذاكرة البرنامج ROM.

تخزين قيم في الذاكرة ROM.

تخزين قيم في الذاكرة EEPROM.

برنامج تشغيل الدارة (2):

التوجيهات.

```
$regfile = "m128def.dat"
$crystal = 4000000
$baud = 9600
$eepromhex
$eepleave
'------------------
Dim Var As Sram Byte At &H200
'------------------
$eeprom
   Label1:
   Data 1 , 2 , 3 , 4 , 5

   Label2:
   Data 10 , 20 , 30 , 40 , 50
$data
'------------------
   Readeeprom Var , Label1
      Print Var
         Readeeprom Var
            Print Var
'/------


   Readeeprom Var , Label2
      Print Var
         Readeeprom Var
            Print Var
'/------


   Var = 100
   Writeeeprom Var , Label1
   Var = 101
   Writeeeprom Var

   Readeeprom Var , Label1
      Print Var
   Readeeprom Var
      Print Var
'/------


   Var = 0
   Writeeeprom Var , 3
      Readeeprom Var , 3
         Print Var
End
```

تخزين قيم في الذاكرة EEPROM عند لافتات محددة.

قراءة قيمة عند لافتة محددة في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

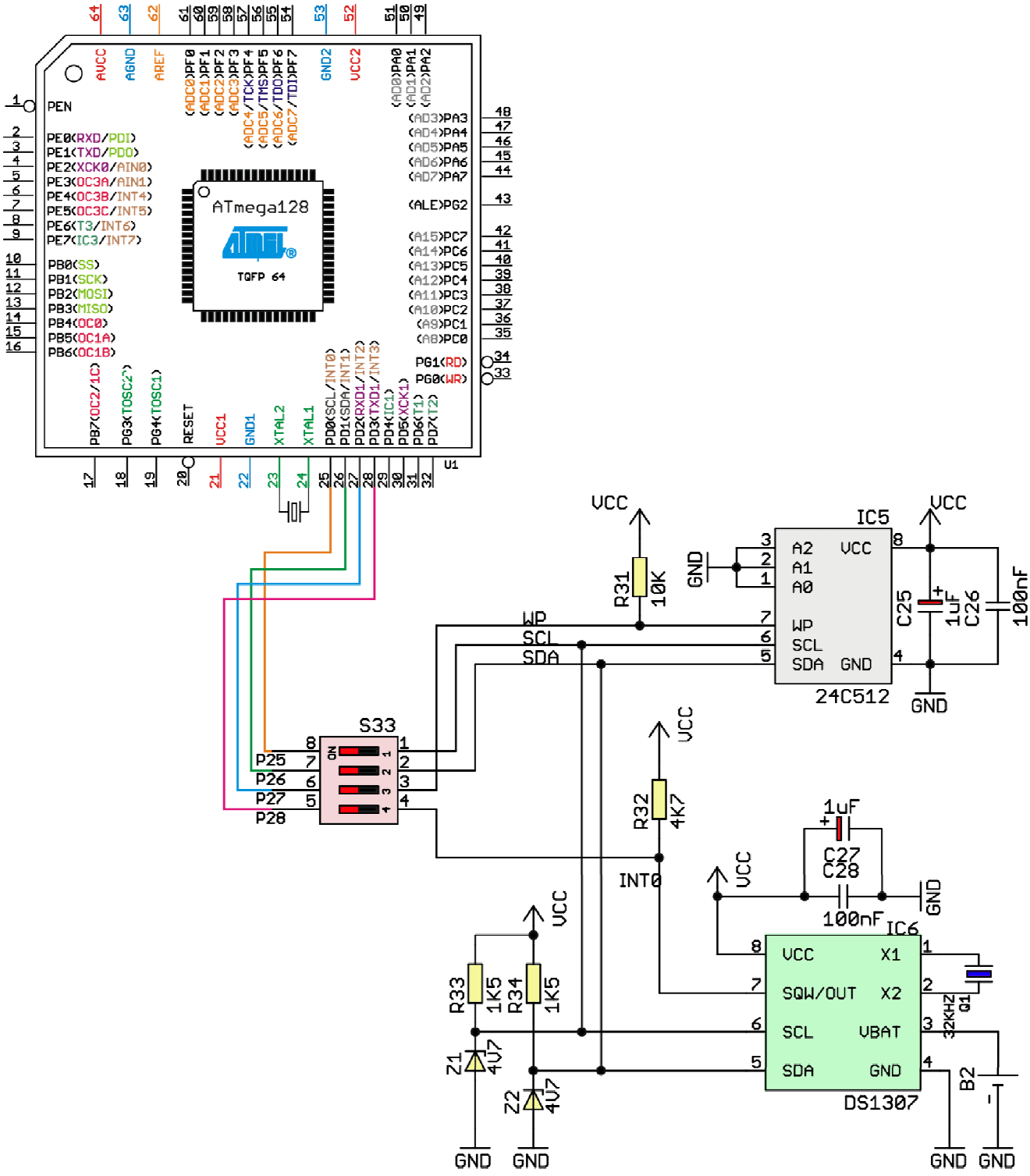
قراءة قيمة عند لافتة محددة في الذاكرة EPROM إلى متحول في الذاكرة SRAM.

إسناد قيمة متحول في الذاكرة SRAM إلى الذاكرة EEPROM.عند لافتة محددة.

الغاية من التجربة:

دراسة مبدأ عمل البروتوكول I²C من خلال برمجة ذاكرة معطيات تسلسلية EEPROM ودارة توليد الزمن الحقيقي RTC.

مخطط الدارة:



مبدأ عمل البروتوكول I^2C:

تم تطوير ويعتبر البروتوكول I^2C (Inter-Integrated Circuit) في أوائل عام 1980 من قبل شركة Philips بهدف تخفيض كلفة تصنيع المنتجات الإلكترونية (TV) ومن أجل ربط متحكم مع بعض المحيطيات في أجهزة التلفاز ويعتبر الأكثر استخداماً في الأجهزة الإلكترونية ويسمى أيضاً TWI (Two Wire Interface).

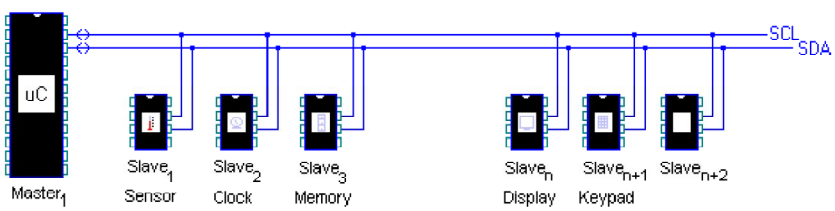
يصنف البروتوكول I^2C من بروتوكولات الاتصال التسلسلي المتزامنة التي تعتمد على خطين (SDA, SCK)، الخط SDA دائماً ثنائي الاتجاه، فإما أن يكون اتجاه البيانات من المرسل إلى المستقبل أو العكس، أما الخط SCK فهو أحادي الاتجاه في الأنظمة التي تعتمد مبدأ One-Master<>Multi-Slaves ويكون ثنائي الاتجاه في الأنظمة التي تعتمد على مبدأ Multi-Master<>Multi-Slaves.

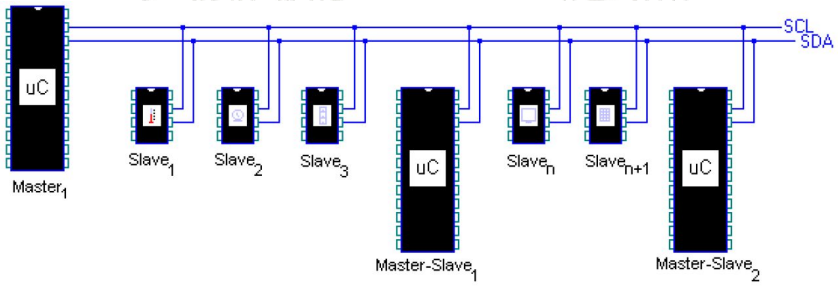
يستخدم هذا البروتوكول من أجل ربط Chip-to-Chip عند سرعات نقل منخفضة ومتوسطة، ويعمل من أجل Multi-Drop Bus حيث يمكن أن يتواجد على خط النقل قائد وحيد (Master) وأكثر من مقاد (Multi-Slave) ويتم التخاطب بين جهازين فقط معاً في نفس اللحظة (Master<>Slave)، كما يمكن إعداده من أكثر وجود أكثر من قائد (Multi-Master) وأكثر من مقاد (Multi-Slave).

خرج الأجهزة في البروتوكول I^2C هو من نوع Open collector، لذلك يتم ربط مقاومات رفع لكلا SDA, SCK، وفي حال البطالة تكون حالة كلا الخطين "1".

كل جهاز يعمل وفق البروتوكول I²C يملك عنوان فريد (Unique Address) مؤلف من 7-Bit (112 nodes)، أو يمكن أن يتوفر بعنوان 10-bit (1008 nodes) يستخدم هذا العنوان لتحديد الجهاز المقاد المراد التخاطب معه من قبل القائد.

إن عدد الأجهزة على خط النقل يعتمد مباشرة على سعة الخط حيث أن القيمة الأعظمية للسعة يجب أن لا تتجاوز 400pF، وغالباً تكون سعة كل جهاز بحدود 10pF.





المميزات والمساوئ لبرتوكول الاتصال I^2C:

المساوئ (Disadvantages)	المحاسن (Advantages)
× مسافة الاتصال قصيرة (3meter).	ü يمكن التخاطب مع أكثر من Slave باستخدام خطين.
× سرعات منخفضة لا تتجاوز 400Khz.	ü كلفة البناء منخفضة ، وسهل التطبيق.
× محدودية في عناوين الأجهزة.	ü شائع الاستخدام، ومتوفر ككيان صلب وبرمجياً.
	ü يمكن فصل ووصل أي جهاز من الناقل دون أي تأثير على النظام أو الحاجة لأي تغيير.

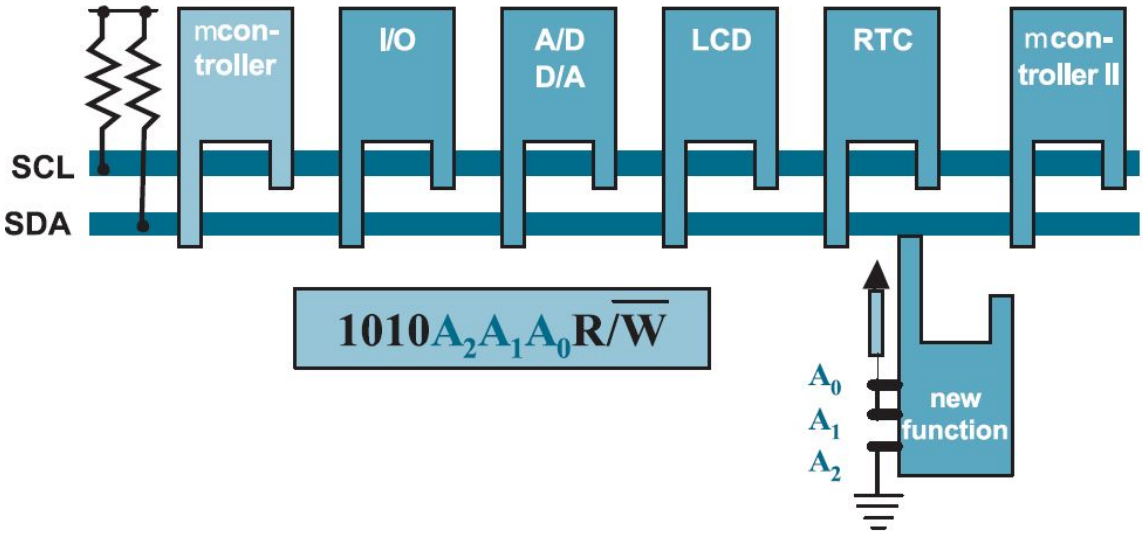
Interfacing I2C EEPROM

Description:

I^2C is an abbreviation of Inter Integrated Circuit and is a protocol for serial communication between Integrated Circuits, it is also called Two Wire Interface (TWI). The bus is used for communication between microcontrollers and peripheral devices like memories, temperature sensors and I/O expanders. An EEPROM is a Electrically Erasable and Programmable Read Only Memory.

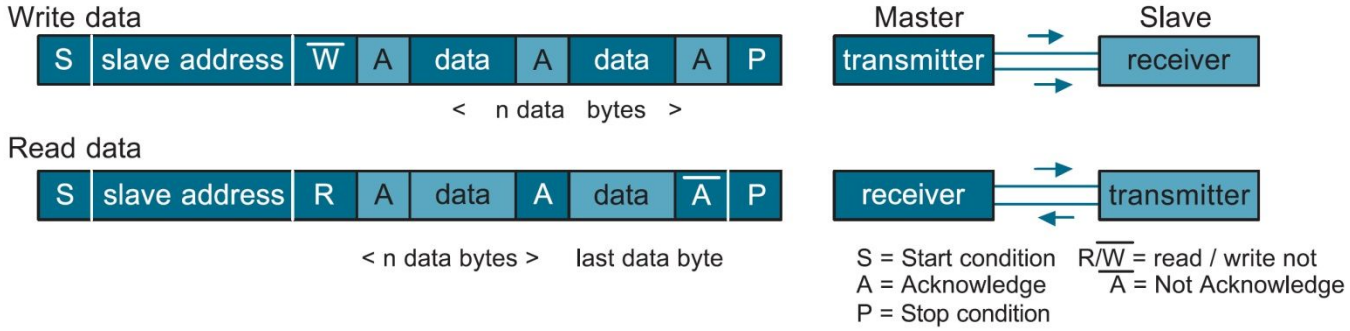
EEPROM Model	Size	Internally Organized	Address (hex)
AT24C01	128 Bytes	128 x 8 = 1024 bits	00000 >> 0007F
AT24C02	256 Bytes	256 x 8 = 2048 bits	00000 >> 000FF
AT24C04	512 Bytes	512 x 8 = 4096 bits	00000 >> 001FF
AT24C08	1 Kbyte	1024 x 8 = 8192 bits	00000 >> 003FF
AT24C16	2 Kbyte	2048 x 8 = 16384 bits	00000 >> 007FF
AT24C32	4 Kbyte	4096 x 8 = 32768 bits	00000 >> 00FFF
AT24C64	8 Kbyte	8192 x 8 = 65536 bits	00000 >> 01FFF
AT24C128	16 Kbyte	16384 x 8 = 131072 bits	00000 >> 03FFF
AT24C256	32 Kbyte	32768 x 8 = 262144 bits	00000 >> 07FFF
AT24C512	64 Kbyte	65536 x 8 = 524288 bits	00000 >> 0FFFF
AT24C1024	128 Kbyte	131072 x 8 = 1048576 bits	00000 >> 1FFFF

*The communication of the bus goes along two lines: **SDA** (Serial Data) and **SCL** (Serial Clock). Each I^2C device has a unique **7-bit address** (Device Select Code). The most significant bits are fixed and assigned to a specific device category (e.g. b1010 is assigned to serial EEPROMS). The three less significant bits (**A2,A1 and A0**) are programmable and used to address the device. The three bits allows eight different I2C address combinations and therefore allowing up to eight different devices of that type to operate on the same I2C-bus. The I2C address is send in the 1st byte, the lest significant bit of the first byte is used to indicate if the master is going to **write(0) or read(1)** from the slave.*



*The device that sends data along the bus is called **master**, a device that receives the data is called **slave**. The master starts the transmission with a start signal and stops the transmission with a stop signal on the SDA line. During the start and stop signals the SCL line has to be high. After the master has started the data-transmission with a start signal, the master writes a device address byte to the*

slave. Each data byte has to have a length of 8 bits. The slave has to acknowledge the reception of the data byte with a acknowledge-bit (ACK).

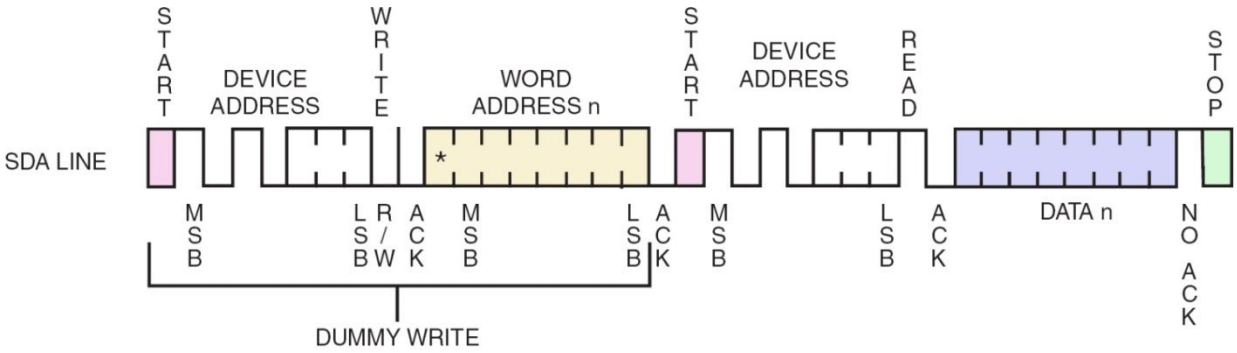


A write operation requires a device address bytes, two address bytes and the data-byte. Upon receive of the address the EEPROM sends an ACK and then clocks in the data-byte. The EEPROM sends again an ACK and the microcontrollers sends a stop-signal to terminate the write sequence.

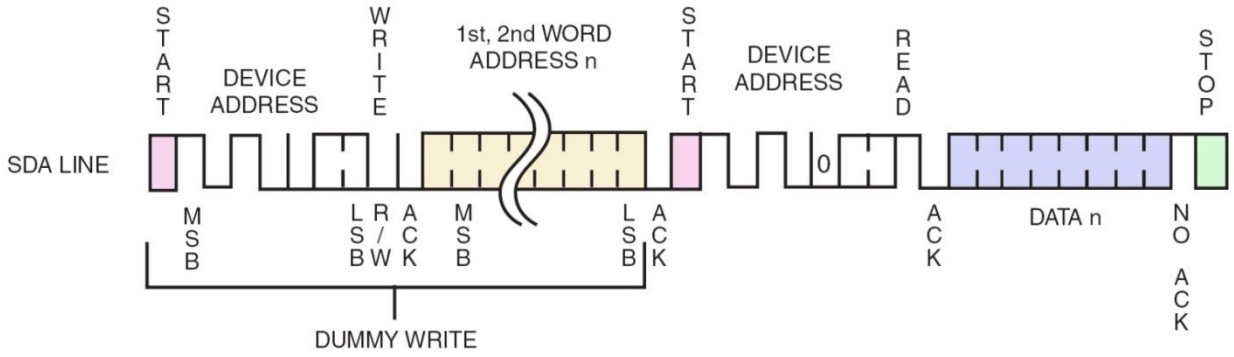
	Device Address / Page Byte	Word Address Bytes	Maximum Devices on Bus
1K/2K	1 0 1 0 A_2 A_1 A_0 R/W	1	8
4K	1 0 1 0 A_2 A_1 P_0 R/W	1	4
8K	1 0 1 0 A_2 P_1 P_0 R/W	1	2
16K	1 0 1 0 P_2 P_1 P_0 R/W	1	1
32K-512K	1 0 1 0 A_2 A_1 A_0 R/W	2	8
1M	1 0 1 0 A_2 A_1 P_0 R/W	2	4
	MSB LSB		

All devices from 32K – 512K will require no system changes and can be interchanged with only the page size differences to consider.

Low Density Random Read:



Medium and High Density Random Read:



AT24C32 (4 Kbyte)

*4096 * 8 = 32768 bits* ***0000 >> 0FFF*** *32 byte page*

&H0000	Saturday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H01FF
&H0200	Sunday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H03FF
&H0400	Monday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H05FF
&H0600	Tuesday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H07FF
&H0800	Wednesday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H09FF
&H0A00	Thursday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H0BFF
&H0C00	Friday [128 Set] 128 x 4 = 215 Bytes *512Bytes*	&H0DFF
&H0E00	NON USED AREA *512Bytes*	&H0FFF

Software:

The BASCOM-AVR compiler is used to make a program that writes and reads one byte from the EEPROM. BASCOM has several embedded commands to control the I2C bus.

In BASCOM-AVR you first have to configure the ports you use for the SDA and SCL lines of the I2C bus. Then you send the device address to select the EEPROM that is connected to the I2C bus. After that you send two bytes to the EEPROM to select the address in the EEPROM to which you want to write the data. The last byte to send in a write sequence is the data byte.

```
$regfile = "m16def.dat"
$crystal = 2000000
$lib "I2C_TWI.LBX"
$baud = 9600
'----------------------------
Config Scl = Portc.0
Config Sda = Portc.1
Config Twi = 100000                                    '100KHZ
'----------------------------
Const Addressw = 160                                  '&B10100000 slave write address
Const Addressr = 161                                  '&B10100001 slave read address
'----------------------------
Dim Adres_h As Byte , Adres_l As Byte
Dim Rd_value As Byte , Wr_value As Byte
'----------------------------
Do
   Input "Wr_value:" , Wr_value
   Input "Adres_l:" , Adres_l
   Input "Adres_h:" , Adres_h

   Gosub Write_eeprom
   Gosub Read_eeprom

   Print "Error W: " ; Err
   print "Wr_value: " ; Wr_value

   Print "Error R: " ; Err
   Print "Rd_value: " ; Rd_value
Loop
End
'----------------------------
Write_eeprom:
    I2cstart                                           'Start condition
    I2cwbyte Addressw                                  'Slave address
    I2cwbyte Adres_h                                   'H address of EEPROM
    I2cwbyte Adres_l                                   'L address of EEPROM
    I2cwbyte Wr_value                                  'Value to write
    I2cstop                                            'Stop condition
    Waitms 10                                          'Wait for 10 milliseconds
Return
'----------------------------
Read_eeprom:
   I2cstart                                            'Generate start
   I2cwbyte Addressw                                   'Slave adsress
   I2cwbyte Adres_h                                    'H address of EEPROM
   I2cwbyte Adres_l                                    'L address of EEPROM
   I2cstart                                            'Repeated start
   I2cwbyte Addressr                                   'Slave address (read)
   I2crbyte Rd_value , Nack                            'Read byte
   I2cstop                                             'Generate stop
Return
'----------------------------
```