Using SDRAM with BASCOM AVR

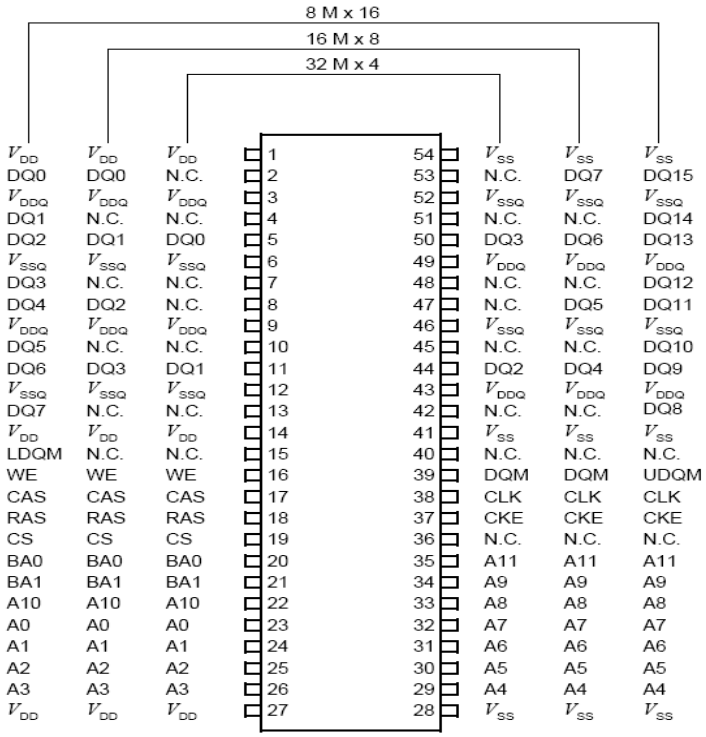
Author and library designer: Ali Taroosheh

SDRAM memory is a Dynamic RAM and they used in old PCs and it can be found in SDRAM modules. This memory is faster than AVR microcontrollers, but it has large space of RAM and this is very tempter. Below table contained details of all SDRAM types:

Module's type	Space of each IC	Array dimensions
128MByte	8MByte	4096*512*8bit*4Bank
256MByte	16MByte	4096*1024*8bit*4Bank
512MByte	32MByte	8192*1024*8bit*4Bank
1024MByte	64MByte	8192*2048*8bit*4Bank

However other types of these memories are available but we use 8bit type. (Attention to suffix part number of IC)

Below figure is a pin configuration of HYB39S128 16MByte SDRAM from Infineon Company and in this article be used 16Mx8 configurations. For more information refer to datasheets that you can find these ones into the folder of this case study.



SDRAM.lbx library in BASCOM AVR

This case study contains "SDRAM.lbx" and "config_SDram.bas" and "test.bas" files and the library most place in "lib" folder in installation path of BASCOM AVR.

"config_SDram.bas" contains some of constants and variables and declaration sub routines. The "Databus" refer to one IOport that connected to DQ7-DQ0, now porta connected to DQ7-DQ0 and if you want to change it change "Dataddr" and "Datain" to same name, for example if you want to connect DQ7-DQ0 to porc write this way:

```
Databus alias portc
dataddr alias ddrc
datain alias pinc
```

"adl" and "adh" are 2 IOports that connected to address bus. For example if you use 16MB memory these ones most to connect to A11-A0, and if you want to change it change other depended things, "adlddr" and "adhddr" most to change to the same name. For example:

```
Adl alias portb
adh alias porta
adlddr alias ddrb
adhddr alias ddra
```

Attention: other free pins of "adh" not useable.

After connecting "adh" to MSB's address bus you most to connect BA1-BA0 to the first free pins form "adh". For example about 16MB memory, it has 12 lines for address (A11-A0) and adh3-adh0 connect to A11-A8 and then BA1-BA0 most to connect to adh5-adh4, or about 64MB memory it has 13 line address bus and BA1-BA0 most to connect to adh6-adh5.

"comm" is reserved word for command lines, this memory has 6 control lines: dqm, cke, cs, ras, cas and we, in this example portd is "comm". You can change name of IOport and change constants of each pins and other free pins of "comm" can be use for other works, in this example portd.0 and portd.1 are free and use for UART.

If you want to connect portc.6 to cke write this way:

```
Comm alias portc
commddr alias ddrc
const cke=6
```

"sdram_vol" refer to memory space and it can change between 8, 16, 32 and 64.

Don't change other constants and variables.

"config_SDram.bas" was written here:

```
'###############################################################################
'                              SDram driver                                     '
'                         design with Ali taroosheh                             '
'                       email:ali.taroosheh@gmail.com                           '
'                      weblog:www.electrorc.blogfa.com                          '
'                    weblog:www.electrorc-en.blogspot.com                       '
'                                                                               '
'                                                                               '
'                                                                               '
'                               (not for compile)                               '
'###############################################################################

Databus Alias Porta                                     'Data bus
Dataddr Alias Ddra                                      'Data bus direction
Datain Alias Pina

Adl Alias Portc                                         'LSB address Bus
Adh Alias Portb                                         'MSB address Bus
Adlddr Alias Ddrc
Adhddr Alias Ddrb

Comm Alias Portd                                        'Command lines
Const Dqm = 2                                           'portd.2
Const Cke = 3                                           'portd.3
Const Cs = 4                                            'portd.4
Const Ras = 5                                           'portd.5
Const Cas = 6                                           'portd.6
Const We = 7                                            'portd.7
Commddr Alias Ddrd

Const Sdram_vol = 16                                    '16MByte

#if Sdram_vol = 8
Const Sdram_type = 12
#endif
#if Sdram_vol = 16
Const Sdram_type = 12
#endif
#if Sdram_vol = 32
Const Sdram_type = 13
#endif
#if Sdram_vol = 64
Const Sdram_type = 13
#endif

Const Not_comm = 255 Xor((2 ^ Dqm)or(2 ^ Cke)or(2 ^ Cs)or(2 ^ Ras)or(2 ^ Cas)or(2 ^ We))
Const Xor_comm =(2 ^ Dqm)or(2 ^ Cke)or(2 ^ Cs)or(2 ^ Ras)or(2 ^ Cas)or(2 ^ We)
Const Activate = Not_comm Or((2 ^ Cke) Or(2 ^ Cas)or(2 ^ We))
Const Cread = Not_comm Or((2 ^ Cke)or(2 ^ Ras)or(2 ^ We))
Const Cwrite = Not_comm Or((2 ^ Cke)or(2 ^ Ras))
Const Precharge = Not_comm Or((2 ^ Cke)or(2 ^ Cas))
```

```
Const Moderegister = Not_comm Or((2 ^ Cke))
Const Noop = Not_comm Or((2 ^ Cke)or(2 ^ Ras) Or(2 ^ Cas) Or(2 ^ We))
Const Noopr = Not_comm Or((2 ^ Cke)or(2 ^ Ras) Or(2 ^ Cas) Or(2 ^ We) Or(2 ^ Dqm))
Const Autorefresh = Not_comm Or((2 ^ Cke)or(2 ^ We))
Const Selfrefresh = Not_comm Or((2 ^ We)or(2 ^ Dqm))
Const Selfrefreshexit = Not_comm Or((2 ^ Cke)or(2 ^ Ras)or(2 ^ Cas)or(2 ^ We)or(2 ^ Dqm))
Const Deselect = Not_comm Or((2 ^ Cke)or(2 ^ Cs))
Const Burstterminate = Not_comm Or((2 ^ Cke)or(2 ^ Ras)or(2 ^ Cas)or(2 ^ Dqm))

Dim B_row As Word
Dim B_column As Word
Dim B_length As Word
Dim Sd_position As Long
Dim Sd_pointer As Word

$lib "SDRAM.lbx"
$external Sd_init , Sd_write , Sd_read
Declare Sub Sd_init()
Declare Sub Sd_write(array_pointer() As Byte , Byval Sd_position As Long , Byval Sd_length As
Word)
Declare Sub Sd_read(array_pointer() As Byte , Byval Sd_position As Long , Byval Sd_length As
Word)

Call Sd_init()
```

Programming with BASCOM AVR

In this library three sub routines worked out and you can use these sub routines for initialization, read and write to SDRAM.

1. **Sdinit()** is the first sub routine that called in "config_SDram.bas" for one time and don't need to use another time.

```
Declaration: Sub Sdinit()
Usage: Call sdinit()
```

2. **Sd_write** is the write sub routine and has three parameters; the first one is variable or variable array that will be written to the SDRAM. Second one specifies the position where the data must be written. This must be a long variable, and third one specifies how many bytes must be written to the file.

```
Declaration: Sub Sd_write(byref array_pointer() as byte,
byval sd_position as long, byval sd_length as word)
Usage: Call sd_write(array(1),position,length)
```

```
Example:
Dim arr(100) as byte
Call sd_write (arr(1),16000000,100)
```

3. **Sd_read** is the read sub routine, about parameters **sd_read** and **sd_write** are the same. In this sub routine the first parameter is variable or variable array that will be assigned with the data from the SDRAM. The second parameter specifies the position where the reading must start from. This must be a long variable, and the third parameter specifies how many bytes must be read from the SDRAM.

```
Declaration: Sub Sd_read(byref array_pointer() as byte, byval
sd_position as long, byval sd_length as word)
Usage: Call sd_read(array(1),position,length)
```

```
Example:
Dim arr(2048) as byte
Dim position as long
Positon=10000
Length=2048
Call sd_read (arr(1),position,length)
```

This is an example code that specified how to use these sub routines properly:

```
$regfile = "m32def.dat"
$crystal = 14745600
$baud = 38400
$swstack = 64
$hwstack = 64
$framesize = 64
Dim Cmd As String * 50
Dim Count As Word
Dim Pos As Long
Dim Acc As Long
Dim Leng As Word
Dim Res As Byte
Dim Arr(100) As Byte
Dim Ptr As Byte At Cmd Overlay

$include "config_SDram.bas"
Print "SDRAM driver - with AVR on BASCOM"

   Do
   For Count = 1 To 100
      Arr(count) = 0
   Next
   Print "type your command:",
   Input Cmd
      Select Case Cmd
         Case "write":
            Print "position:",
            Input Pos
            Print "type 100 characters then enter:"
            For Count = 1 To 100
               Res = Waitkey()
               If Res = 13 Then Exit For
               Arr(count) = Res
            Next
            Decr Count
            Call Sd_write(arr(1) , Pos , Count )
            Print : Print "write complete"
         Case "read":
            Print "position:",
            Input Pos
            Lencorrect:
            Print "length:" ,
            Input Leng
            If Leng > 100 Then
               Print "out of range, try again"
               Goto Lencorrect
            End If
            Call Sd_read(arr(1) , Pos , Leng)
            For Count = 1 To Leng
               Print Chr(arr(count)),
            Next
            Print : Print "read complete"
         Case "wvar":
            Print "position:",
            Input Pos
            Print "type your string in 50 characters:",
            Input Cmd
            Leng = Len(cmd)
            Call Sd_write(ptr , Pos , Leng)
            Print "write complete"
      End Select
   Loop
End                                                    'end program
```

Note: in this example **"wvar"** section is a simple way to write variables except byte array. Here written a string to SDRAM and in below sample write a long variable to SDRAM:

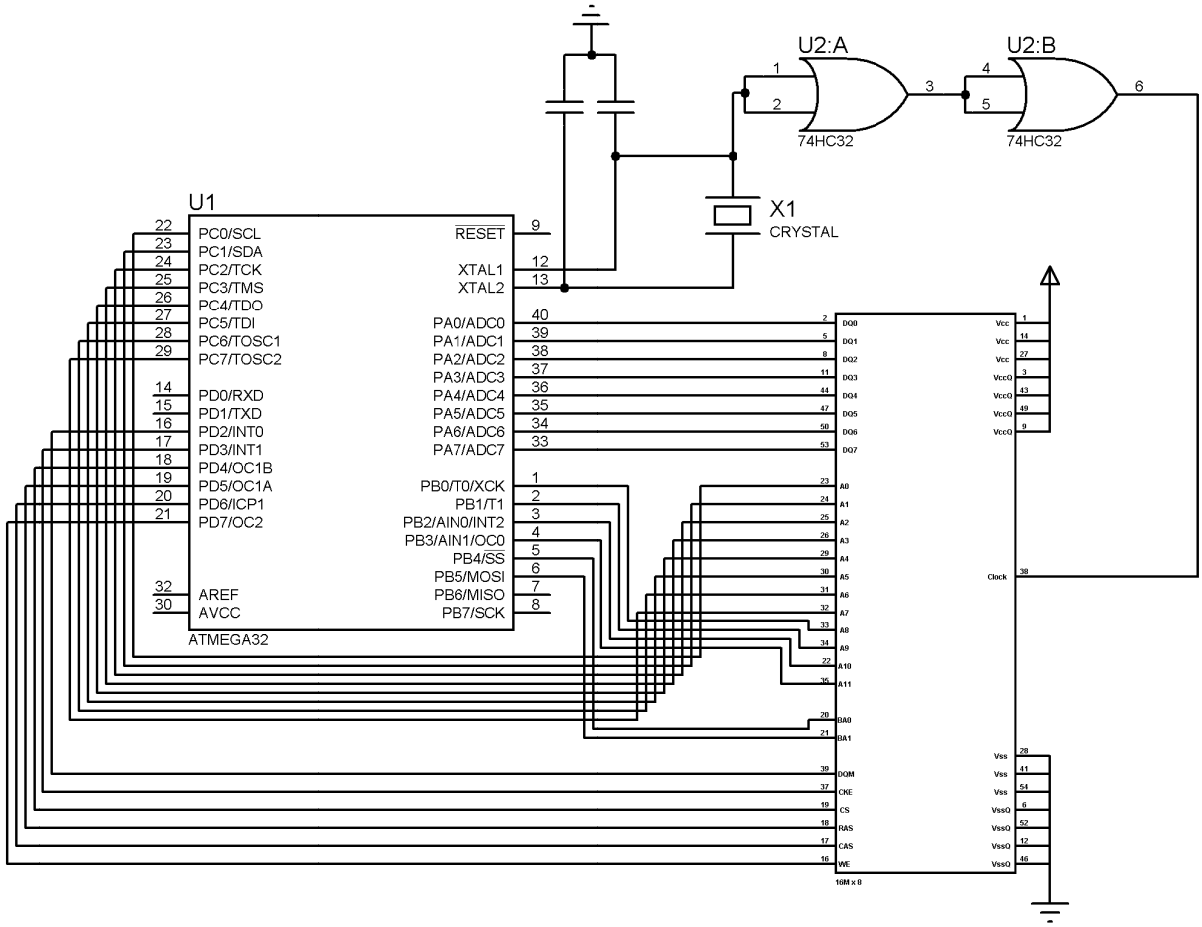
```
Dim l as long
Dim ptr as byte at l overlay
```

```
.
.
.
Call sd_read(ptr,0,4)
```

Test and Operational circuit

Because of complexity of SDRAM usage with AVR and increasing performance SDRAM clock crossed from tow or three digital gates until make a few delays about 20ns or more. Num of gates depends on propagation delay of gates and this circuit with 14.7456MHz crystal oscillator and tow gates of 74hc244 or 74hc32 works properly. This matter only for applying clock to SDRAM after executing instruction by AVR and delay most is lower than the period of Fosc.(Fosc=14.7456MHz → T≈67ns and each gate of 74hc244 has propagation delay about 7ns so using tow or three gates is great).

Note: if propagation delays of gates be more than T data maybe fail.



SDRAM DRIVER WITH BASCOM AVR

Design with Ali Taroosheh
Email: ali.taroosheh@gmail.com
Weblog: http://www.ElectroRC.blogfa.com
English weblog: http://www.ElectroRC-en.blogspot.com