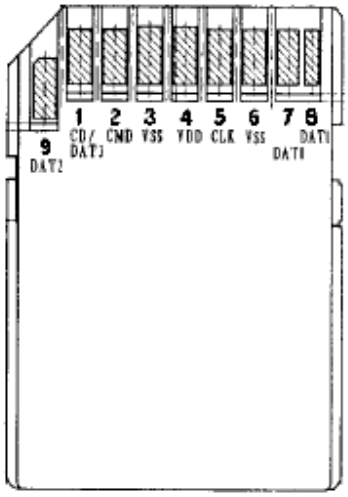
راهنمای استفاده از AVR-DOS در BASCOM

همانطور که می‌دانیم برنامه Bascom یکی از پرقدرتترین کامپایلرهای موجود در زمینه میکروکنترلرهای AVR می‌باشد. این کامپالر هرچند که به زبان Basic می‌باشد و هر چند که این زبان برنامه نویسی زبان مرسوم برنامه نویسی برای میکروکنترلر نمی‌باشد اما توانسته با ارائه امکانات بسیار گسترده طرفداران زیادی را ایجاد کند. یکی از امکانات خارق العاده این کامپایلر کتابخانه AVR-DOS و MMC می‌باشد. در مورد این دو کتابخانه که به نوعی مکمل یکدیگر هستند می‌توان به جرأت گفت کاملترین کتابخانه در زمینه کارتهای حافظه SD و MMC و استانداردهای FAT32, FAT16, FAT12 می‌باشند(البته ترکیب کتابخانه AVR-DOS و FlashCardDrive توانایی کار با حافظه‌های CF را ایجاد می‌کند).

حافظه بزرگ، یک نیاز: یکی از مشخصات برجسته هر میکروکنترلر وجود حافظه بیشتر است. شاید این مشخصه که یک نیاز بزرگ است تا کنون با تولید انواع زیادی از میکروکنترلرها برطف نشده است اما شرکت‌های تولید کننده دست از کار نکشیده و انواع حافظه‌های حجیمی را تا کنون ارائه داده‌اند. بطور مثال شرکت ATMEL نوعی حافظه Flash با نام At45DBxxx را ارائه داد که تا سقف ۱۶ مگابایت آن تولید شده و استفاده گسترده‌ای از آن نیز شد، اما این فضا بدلیل جدا بودن از محیط کامپیوتر باز هم ارضأ کننده این نیاز نیست. وجود حافظه-های SD و MMC برای این نیاز پاسخ کاملا قانع کننده‌ای می‌باشد چون از سویی حجم بسیار عظیمی را در اختیار کاربر قرار می‌دهد و هم از سویی با استانداردهای مرسوم کامپیوتر سازگار بوده و استفاده از آن به مراتب از هر نوع حافظه‌ای آسانتر می‌باشد.

Memory cards: دو بخش عمده این حافظه‌ها شامل SD و MMC بوده که تفاوت چندانی با یکدیگر نداشته مگر در سرعت که SD سرعت بیشتری داشته و نیز کتابخانه‌های مذکور سازگاری بیشتری با این نوع دارند. در شکل زیر شکل و ترتیب پایه‌های حافظه SD را نشان می‌دهد:



Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

جدول زیر شرح پایه‌های این حافظه می‌باشد:

Pin No.	Name	Type[1]	Description
SD Mode			
1	CD/DAT3[2]	I/O[3], PP	Card detect/Data line [Bit 3]
2	CMD	I/O, PP	Command/Response
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DAT0	I/O, PP	Data line [Bit 0]
8	DAT1	I/O, PP	Data line [Bit 1]
9	DAT2	I/O, PP	Data line [Bit 2]
SPI Mode			
1	CS	I	Chip Select (active low)
2	DataIn	I	Host-to-card Commands and Data
3	V_{SS1}	S	Supply voltage ground
4	V_{DD}	S	Supply voltage
5	CLK	I	Clock
6	V_{SS2}	S	Supply voltage ground
7	DataOut	O	Card-to-host Data and Status
8	RSV[4]	---	Reserved
9	RSV[5]	---	Reserved

همانطور که در جدول فوق ملاحظه می‌کنید این‌گونه حافظه‌ها توانایی ارتباط تحت دو پروتکل را دارا می‌باشند. پروتکل ارتباطی اول SD mode که نوع پرسرعت ارتباطی بوده و این کتابخانه از آن پشتیبانی نمی‌کند و SPI mode که تحت پروتکل نام آشنای SPI با دنیای خارج ارتباط برقرار کرده و بسیاری از میکروکنترلرهای موجود فعلی این ارتباط را به صورت سخت افزاری در خود دارند و در AVR بطور معمول این ارتباط با سرعت 8mbps قابل راه‌اندازی است.

نکته: ولتاژ کاری این نوع حافظه بین ۲.۷ ولت تا ۳.۶ ولت می‌باشد و برای راه‌اندازی آن پیشنهاد می‌شود ولتاژ کاری میکروکنترلر و این حافظه ۳.۶ ولت تنظیم کرد.

نکته: پایه‌های DataIn به MOSI و CLK به CSK و DataOut به MISO از میکرو متصل می‌شود.

Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

BASCOM و کارت حافظه

برای اینکه در BASCOM کارت حافظه را راه‌اندازی کنیم باید فایل "Config_MMC.bas" را در اختیار داشته باشیم. این فایل در مسیر نصب برنامه BASCOM موجود است. مسیر زیر را پیگیری کنید:

C:\Program Files\MCS Electronics\BASCOM-AVR\SAMPLES\AVRDOS

این فایل حاوی تنظیماتی برای پروتکل ارتباطی SPI می‌باشد. در خط ابتدایی برنامه ثابتی به نام Cmmc_soft تعریف شده است که در صورت صفر بودن این پروتکل به صورت سخت افزاری پیکره‌بندی می‌شود و اگر عددی بغیر از صفر باشد SPI بصورت نرم افزاری پیکره‌بندی می‌شود. پیکره‌بندی نرم افزاری به مراتب دارای سرعت کمتری بوده اما می‌توان از هر پایه ورودی/خروجی مورد نظر برای ارتباط با حافظه استفاده کرد که این موضوع در نوع سخت افزاری آن صادق نبوده و اماکن پذیر نیست. در صورت قرار دادن عدد غیر صفر تنظیمات پایه‌های مورد نظر بعد از خط زیر در این فایل انجام می‌شود:

```
#else                                   ' Config here SPI pins, if not using HW
```

BASCOM و استاندارد FAT

حال برای اینکه بتوانیم در برنامه BASCOM طبق استاندارد FAT استفاده کنیم، باید به فایل "CONFIG_AVR-DOS.bas" دسترسی داشته باشیم که این فایل در مسیر فوق الذکر موجود می‌باشد و در اینجا دوباره این مسیر را نشان می‌دهیم:

C:\Program Files\MCS Electronics\BASCOM-AVR\SAMPLES\AVRDOS

این فایل حاوی تعداد زیادی متغیر و ثابت بوده و ما برای سطح مبتدی و حتی متوسط کار چندانی با آنها نداشته و فقط کامپایلر با آنها مفهوم FAT را راه‌اندازی و پشتیبانی می‌کند. اما در این فایل دو ثابت بسیار مهم موجود است که ذکر توضیح در مورد آنها خالی از لطف نیست. اولین خط برنامه موجود در این فایل تعریف ثابت cfilehandles می‌باشد که عدد متناظر با آن نشان دهنده تعداد فایلی است که می‌توان همزمان باز کرد و ثابت بعدی csepfathandle می‌باشد که تعداد بافر برای کار با این استاندارد در آن مشخص می‌شود و بافر بزرگتر بشکل محدود روی سرعت کار تاثیر دارد.

نکته: توجه کنید که اعداد بالا هر کدام با دریافت ضریب، حجم قابل توجهی از SRAM داخلی میکروکنترلر را در اختیار می‌گیرد. برای تغییر دادن این اعداد به توضیحاتی که بالای هر ثابت نوشته شده کاملا دقت کنید زیرا رعایت نکردن این نکته باعث بروز خطا در کامپایلر می‌شود(out of sram space).

توابع کتابخانه‌ای و برنامه نویسی

قبل از هر چیزی ذکر این نکته اهمیت دارد که میکروکنترلر در کار با FAT نیاز به یک ساعت دارد. البته این موضوع فقط برای modify کردن فایل و فولدرهاست و بطور معمول بوسیله RTC تامین می‌شود و توصیه می‌شود که از این امکان استفاده شود.

بدلیل گستردگی کدهای خطا در این دو کتابخانه به "“AVR-DOS File System” موجود در help برنامه BASCOM مراجعه کنید.

ابتدا برای هر کاری باید فایل‌های مذکور را در برنامه خود اضافه و پیوست کنیم برای این کار پیشنهاد می‌شود این دو فایل را از مسیر مذکور کپی کرده و در مسیر پروژه اصلی خود قرار دهید و به صورت زیر این دو فایل را در برنامه اصلی پیوست کنید:

```
$include "config_mmc.bas"

$include "CONFIG_AVR-DOS.bas"
```

بعد از مرحله فوق باید در برنامه از وجود کارت حافظه در مدار یقین حاصل کرد که این کار با دستور زیر انجام می‌شود:

```
Drivecheck()
```

همانطور که در مثال زیر مشاهده می‌کنید در صورتیکه موری کارت در مدار وجود نداشته باشد خروجی تابع فوق عددی غیر از صفر خواهد شد:

```
A2:
If Drivecheck() <> 0 Then Goto A2
```

تابع بعدی تابعیست که در فایل‌های پیوست شده وجود داشته اما ذکر آن الزامی می‌باشد. این تابع که در مثال زیر استفاده شده است کارت حافظه را از نظر نرم افزاری ریست و init می‌کند:

```
Temp1 = Driveinit()
```

در صورت بروز هر گونه خطا، خروجی این تابع غیر صفر خواهد شد.

تابع بعدی بررسی فایل سیستم موجود در کارت حافظه را انجام می‌دهد. این امر اطلاعاتی از قبیل نوع FAT – ظرفیت حافظه و... را در اختیار AVR-DOS قرار می‌دهد و انجام آن برای کار تحت پروتکل FAT ضروریست. این تابع نیز مانند بقیه توابع در صورت بروز خطا در خروجی عدد غیر صفر ایجاد می‌کند.

```
Temp1 = Initfilesystem(1)
```

*. تا به این لحظه کارت حافظه آماده برای کار می‌باشد و شما از این جا به بعد می‌توانید برای کار با آن برنامه نویسی را شروع کنید. برای کار با کارت حافظه توابع گسترده‌ای را طراحی کرده‌اند که در جداول زیر بشکل مجزا به ذکر کاربردی ترین توابع خواهیم پرداخت:

Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

توابع کار با فولدرها

function	Description	sample
chdir	Change directoy	Chdir "ETC."
Chdir "\"	Go back to root directory	Chdir "\"
Chdir "."	Come back to previous directory	Chdir "."
mkdir	Make new directory	Mkdir "ETC."
rmdir	Remove empty directory	Rmdir "ETC."

مثال:

```
Mkdir "ali."

Chdir "ali."
```

در این مثال برنامه یک فولدر با نام "ali." تولید کرده و وارد آن می-شود.

توابع کار با فایلها

function	Description	sample
open	Open new file	*.refer to example
close	Close opened file	*.refer to example
flush	Save current file	*.refer to example
freefile	Return a free file number	*.refer to example
kill	Delete file	kill "ETC.txt"
filedate	Read date of generation file	Temp=filedate("ETC.txt")
filetime	Read time of generation file	Temp=filetime("ETC.txt")
filelen	Reverse length of file	Temp=filelen("ETC.txt")
dir	Show container of path	*.refer to example

توضیح: برای کار با فایلها ابتدا باید به آن فایل یک عدد نسبت داد. این کار فقط برای راحتی و کاهش حجم برنامه می‌باشد بطوریکه هر بار بجای اسم فایل عدد مربوط به آن را بکار می‌بریم. برای دریافت یک عدد آزاد که به فایل دیگری نسبت داده نشده باشد از تابع freefile بصورت زیر استفاده می-شود:

```
Ff=freefile()
```

پس از این کار این عدد که در متغییر FF قرار گرفته به فایل نسبت می-دهیم. این کار با دستور open انجام می‌شود:

```
Open "ETC.txt" for binary as #ff
```

مثال فوق باعث ایجاد یک فایل با شماره ff و نوع دسترسی binary می‌شود. در توضیح نوع دسترسی می‌توان به جدول زیر اشاره کرد:

Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

	Open mode			
Action	Input	Output	Append	Binary
Attr	●	●	●	●
Close	●	●	●	●
Put				●
Get				●
LOF	●	●	●	●
LOC	●	●	●	●
EOF	●	1)	1)	●
SEEK	●	●	●	●
SEEK-Set				●
Line Input	●			●
Print		●	●	●
Input	●			●
Write		●	●	●

همانطور که ملاحظه می‌کنید چهار حالت مختلف وجود دارد: binary, input, output, append که هر کدام توانایی پشتیبانی تعدادی از توابع مذکور را دارند.

فایل از نوع binary کامل ترین حالت دسترسی می‌باشد. فایل از نوع append فقط برای باز کردن فایل موجود می‌باشد. فایل نوع input فقط برای خواندن از یک فایل موجود است و نوع output برای ایجاد فایل جدید و نوشتن در آن است. مثال زیر یک فایل موجود را برای خواندن باز می‌کند:

```
Ff=freefile()

Open "ali.txt" for input as #ff
```

نکته: در مثال بالا منظور اینستکه فایل قبلا وجود داشته و در این صورت فایل به صورت read only باز می‌شود و دستورات نوشتاری روی آن ایجاد خطا می‌کند.

مثال:

```
Ff=freefile()

Open "ali.txt" for binary as #ff

.

.

.

Flush #ff
```


Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

```
Close #ff
```

در این مثال بعد از کار با فایل با دستور flush تغییرات را ذخیره سازی کرده و با دستور close آنرا می‌بندیم(دستور close به تنهایی شامل دستور flush می‌باشد یعنی با این دستور تغییرات ذخیره سپس فایل بسته می‌شود که برای توضیح بیشتر به فایل "config_avr-dos.bas" مراجعه کنید).

دستور dir: این دستور محتوی مسیر فعلی را نمایش می‌دهد. برای درک موضوع در مثال زیر تابعی را نوشته‌ایم که بدنبال فایل مورد نظر گشته و در صورت پیدا کردن آن عدد یک را برمی‌گرداند:

```
Function search (byval name as string) as byte
Local st as string*12
St=""
St=dir("*.*")
While len(st)>0
      If st=name then
           Search=1
           Exit function
      End if
     St=dir()
Wend
Search=0
End function
```

توابع ویرایش فایل‌ها

function	Description	sample
print	Write to current file	Print #ff,"hello"
lineinput	Read a line of current file	Lineinput #ff,st
get	Read from one point	Get #ff,st,pos,le
put	Write data to file	Put #ff,st,pos,le
seek	Return next position of read/write	Temp=seek(#ff)
seek	Change next position of read/write	Seek #ff,temp
eof	End of file status	*.refer to example

توضیحات: دستورات print و lineinput فقط برای قالب ASCII می‌باشد و توانایی جابجایی داده از نوع string را داشته و در صورت جابجایی انواع دیگر داده امکان از بین رفتن آن وجود دارد.

Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

مثال:

```
Dim ff as byte
Dim le as long
Dim pos as word
Dim st as string*85
Ff=freefile()
Open "ali.txt" for binary as #ff
Print #ff,"my name is ali taroosheh"
Seek #ff,20000
Lineinput #ff,st
Close #ff
```

در مثال فوق متن "my name is ali taroosheh" در ابتدای فایل نوشته شد و از آدرس ۲۰۰۰۰ به بعد یک خط خوانده شد.

کاربرد eof: این تابع وضعیت پایان فایل را مشخص می‌کند. در مثال زیر برنامه از ابتدا تا انتهای فایل را خوانده و روی UART ارسال می‌کند:

```
$baud=115200
Dim ff as byte
Dim count as long
Dim le as long
Dim acc as byte
Ff=freefile()
Open "ali.txt" for binary as #ff
Count=1
Do
Le=seek(#ff)
Get #ff,acc,le,count
Print chr(acc);
Loop until eof(#ff)<>0
Print "EndOFFile"
Close #ff
```

دستورات put و get: این دستورات برای خواندن و نوشتن از هر جای فایل به طول مشخص ایجاد شده‌اند. در مثال زیر یک آرایه ۵۱۲ بایتی در آدرس یک تا ۵۱۲ نوشته شده و همان آرایه از فایل خوانده می‌شود:

```
Dim ff as byte
Dim le as long
Dim count as word
Dim arr(512) as byte
Ff=Freefile()
Open "ali.txt" for binary as #ff
Le=1
Count=512
Put #ff,arr(1),le,count
Get #ff,arr(1),le,count
Close #ff
```

توابع کار مستقیم با کارت حافظه

function	Description	sample
Drivecheck	Check the memory card	Temp=drivecheck()
driveinit	Init memory	Temp=driveinit()
drivereset	Reset memory	Temp=drivereset()
drivereadsector	Read a sector from memory	Temp=drivereadsector(wSRAMPointer, lSectorNumber)
drivewritesector	Write a sector to memory	Temp=drivewritesector(wSRAMPointer, lSectorNumber)

نکته: در صورتی‌که حرفه‌ای نیستید با این توابع کار نکنید چون ممکن است کارت حافظه شما آسیب ببیند.

خواندن و نوشتن سکتور: هر سکتور در اینگونه حافظه‌ها ۵۱۲ بایت است مثال زیر خواندن و نوشتن بصورت سکتور به سکتور را نشان می‌دهد:

```
Dim ptr as word
Dim arr(512) as byte
Dim le as long
Dim err as byte
```


Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

```
Le=1000

Ptr=varptr(arr(1))

Err=drivereadsector(ptr,le)

Err=drivewritesector(ptr,le)
```

در این مثال یک بار سکتور ۱۰۰۰ خوانده شد و در آرایه arr نوشته شد و دوباره همان آرایه در همان سکتور نوشته شد.

Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com

امیدوارم با این مطلب توانسته باشید این نوع حافظه‌ها را به کار ببرید

Weblog: www.ElectroRC.blogfa.com
Email: Ali.taroosheh@gmail.com