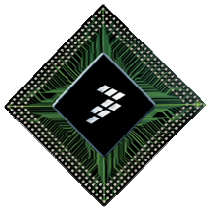
SD Card training with the Flexis ™ JM family and the DEMOFLEXISJMSD card



freescale™
semiconductor

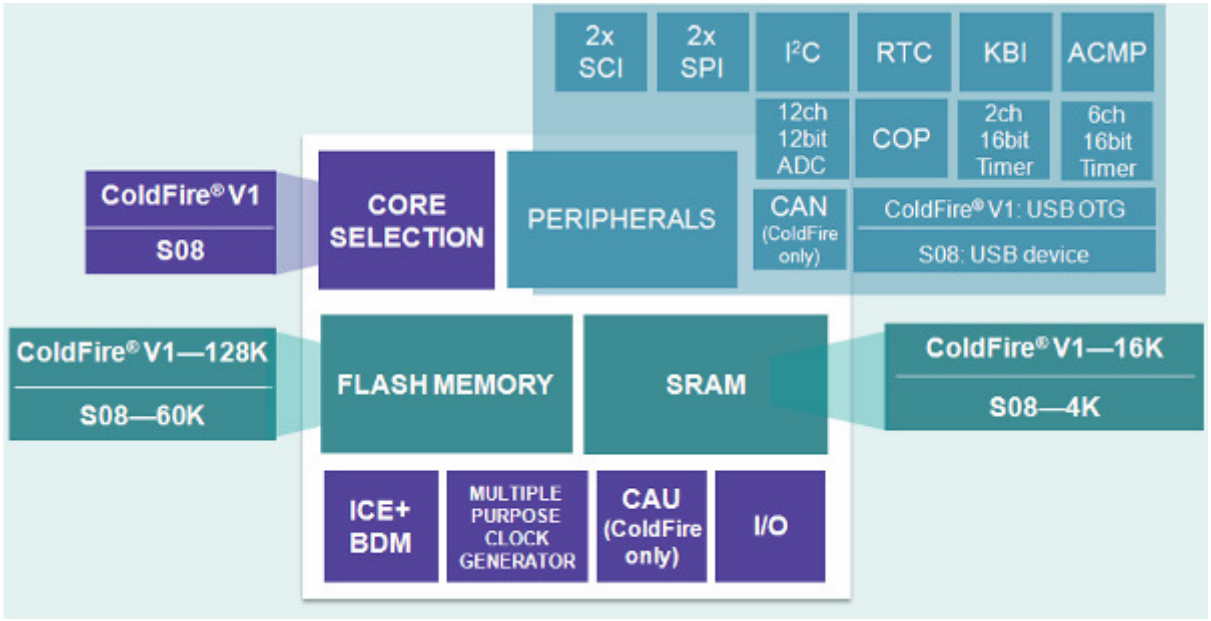
Agenda

- Introduction to Flexis JM Family
- USB and Software Stacks
- SD Cards
- FAT-lite
- Practical examples (Hands on)

The Flexis™ USB Family: JM128 and JM60
ColdFire V1 and S08 JM

2.7—5.5V operating range

►***Memory***
S08
48MHz S08 or ColdFire V1 core
24MHz bus frequency
Up to 4KBytes SRAM; Up to 60KB flash
ColdFire V1
50.33MHz S08 or ColdFire V1 core
25.16MHz bus frequency
Up to 16KBytes SRAM; Up to 128KB flash



►***Features***
2x SCI, I2C, 2x SPI
8 channel KBI
16-bit timers: 1 x 2-ch, 1 x 6-ch
12-bit 12 channel A-to-D converter
Analog comparator
Up to 51 general purpose I/O
Multiple Purpose Clock Generation
- PLL
- FLL
- On-chip oscillator
- External crystal support

Integrated CAN Module (ColdFire V1 only)
Cryptographic Acceleration Unit (ColdFire V1 with 80LQFP only)

►***Complete USB Solution***
Integrated Full Speed USB device (S08) or USB on-the-go (ColdFire V1)
Complimentary USB SW Stack
CodeWarrior for Microcontrollers
Processor Expert

ColdFire JM128 Packages
80LQFP, 64LQFP, 64QFP, 44LQFP

S08JM60 Packages
64LQFP, 64QFP 48QFN, 44LQFP

Temperature Range
JM60 -40C to 85C JM128 -40C to 105C

freescale™ semiconductor

USB and Software Stacks

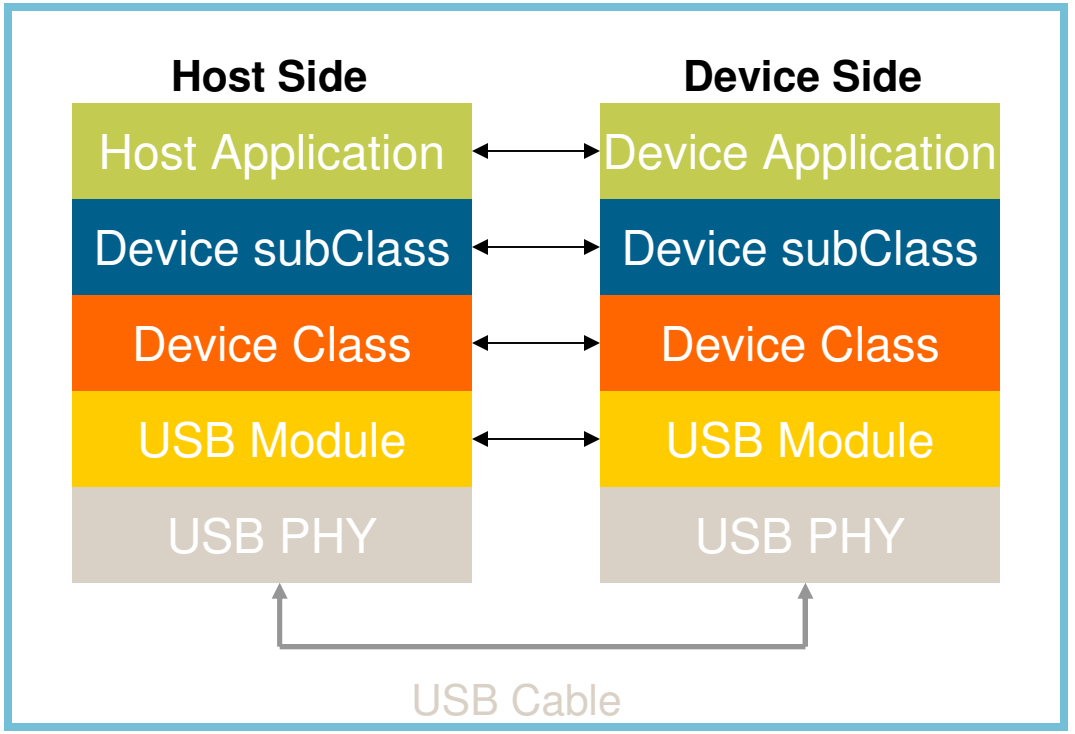
USB Layers

Device subClass
- HID: Joystick, Mouse, Keyboard, etc
- MSD: Hard Disks, Thumb drives, etc
- CDC: All communication related devices

Device Class
- Human Interface Devices
- Mass Storage Device
- Communication device class

USB Module
- Bulk Endpoints
- Interrupt Endpoints
- Isochronus Endpoints
- Control Endpoint

Physical Interface
- Internal
- External



Don't waste Time anymore!!

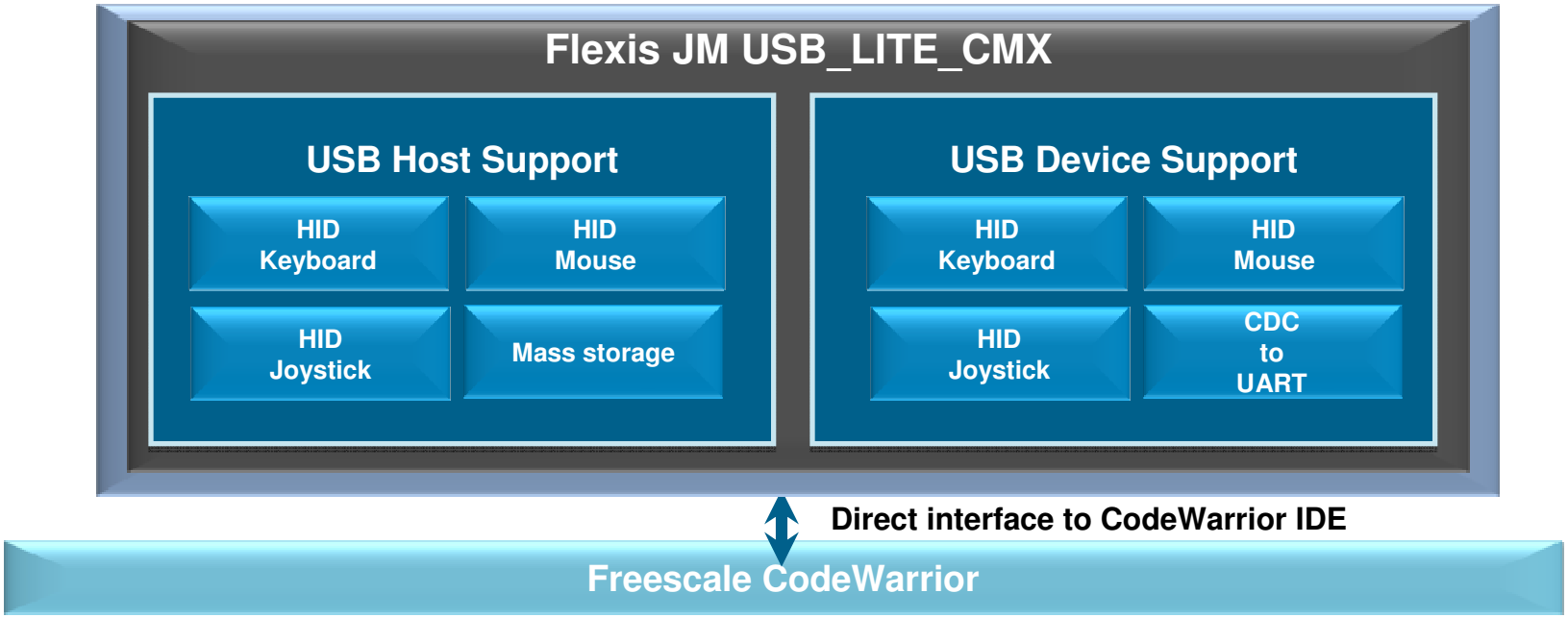
Layer 1 & 2 are provided in Flexis JM family

Stacks handle layers 3 & 4

Just care about Layer 5!!!

freescale semiconductor

Freescale USB_LITE Stack by CMX



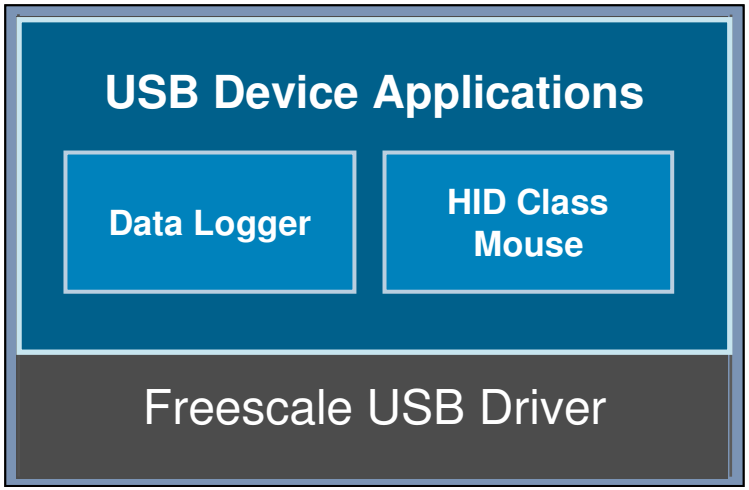
ColdFire V1 CMX stack: support USB host/device/OTG, used for MCF51JMxx

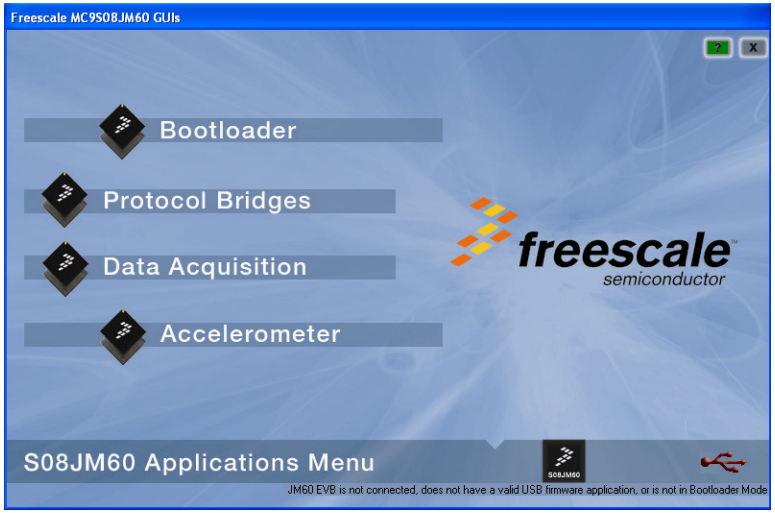
S08 CMX stack: Support USB device, used for MC9S08JMxx

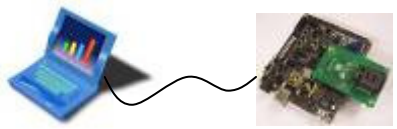
USB_Mini stack for MC9S08JMxx from Freescale

Additional to the CMX stack, Freescale provides a second complimentary and open source USB Mini stack.

- ►Freescale USB Driver + PC Driver + GUI´s
  - USB Driver Designed for JM60 Device
  - Freescale USB PC Driver
    - WinUSB
  - GUI
    - Demo code (VB .NET)
- ►Applications
  - HID Class (Mouse demo)
  - General Application
    - USB – SCI/SPI/IIC Bridge
    - USB Data Logger
- ►USB Boot Loader
  - Update Firmware by USB
  - Project template and library are provided

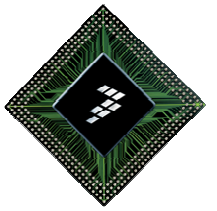






freescale™ semiconductor

Read and Write SD Cards using SPI Module



freescale™
semiconductor

SD Cards

SD card is currently the most popular storage card in the market

- Low Cost
- Low power consumption
- Easy Handling
- Portable
- High capacity

High-end products!
High-performance processors!

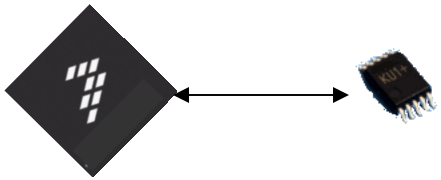
Some include SD Card module!

freescale™ semiconductor

Low-end MCU

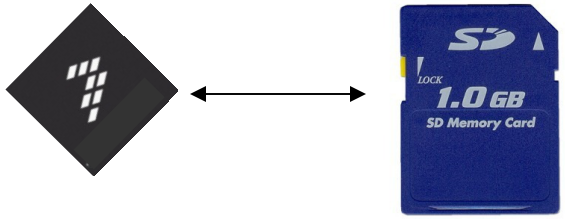
What happens with large amounts of data in low end MCU

Probably use an external serial flash (SPI,IIC,etc) right?



Problem: MCU is needed for external access to data in memory!

Why not use a removable memory like SD Card?

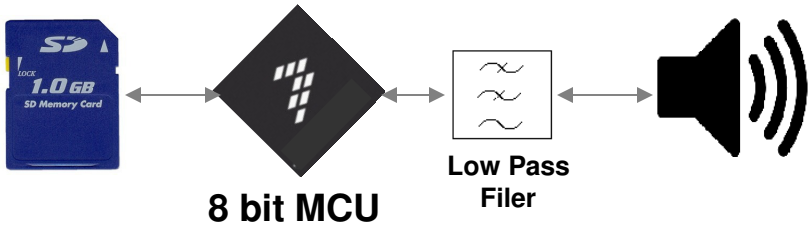


8 bit MCU with SD card Support?

freescale™ semiconductor

Sound generator

- Store wav files in SD card
- Use PWM´s to generate audio signal (No DAC's needed)
- To change sounds (music) only need to replace the SD card
- Low power consumtion
- Low Cost

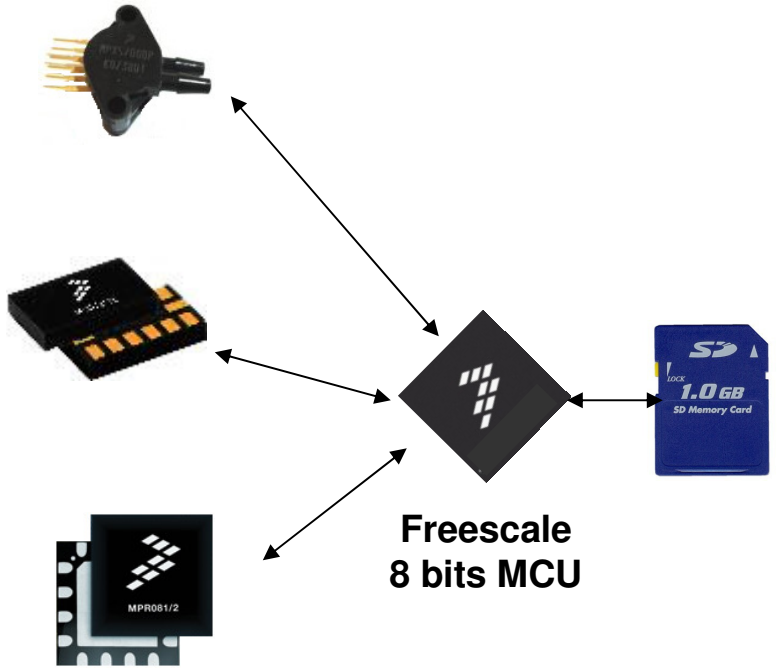


MCU Requirements

1 SPI module	-	4 pins
1 PWM output	-	1 pin
1 timmer	-	NA

Data adquisition

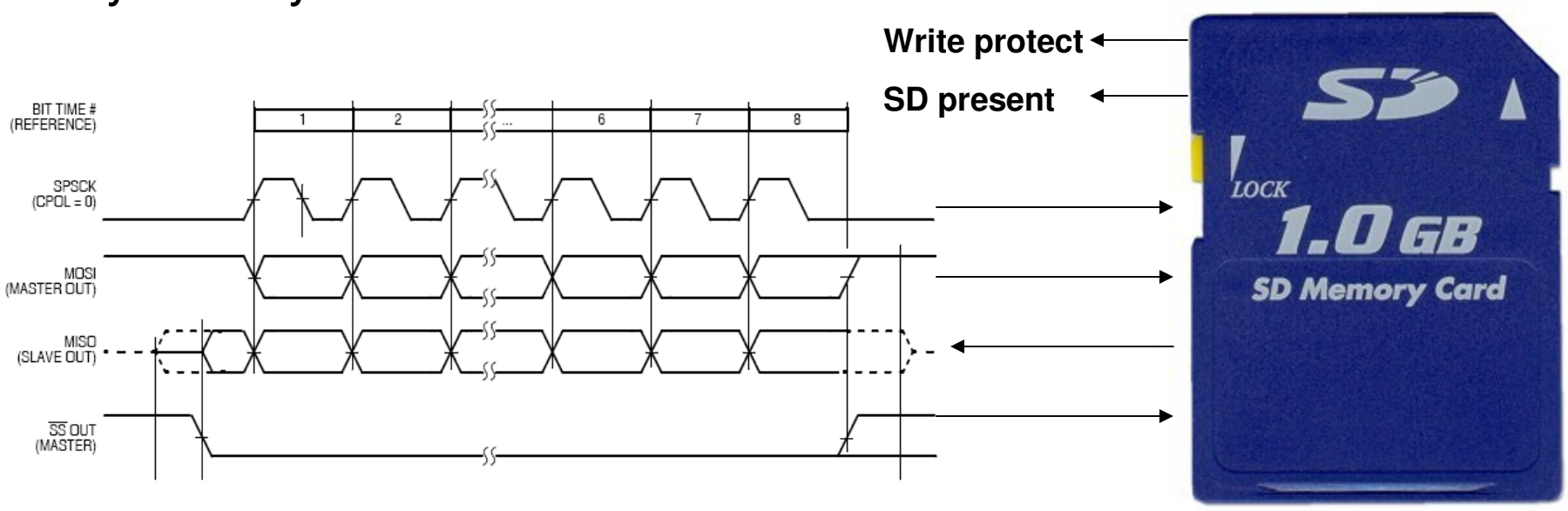
- Acceleration (inclination)
- Proximity
- Humidity
- Light
- Wind speed
- Pressure
- Speed
- Data network
- Etc



SD Card Operation: Physical layer

- SD Card may communicate in 4 bit mode or SPI.
  - SPI is, by standard, forcibly supported by SD and mini SD but not necessarily by Micro SD.
  - SPI mode was developed specifically to support MCU.

- Physical layer for SPI:

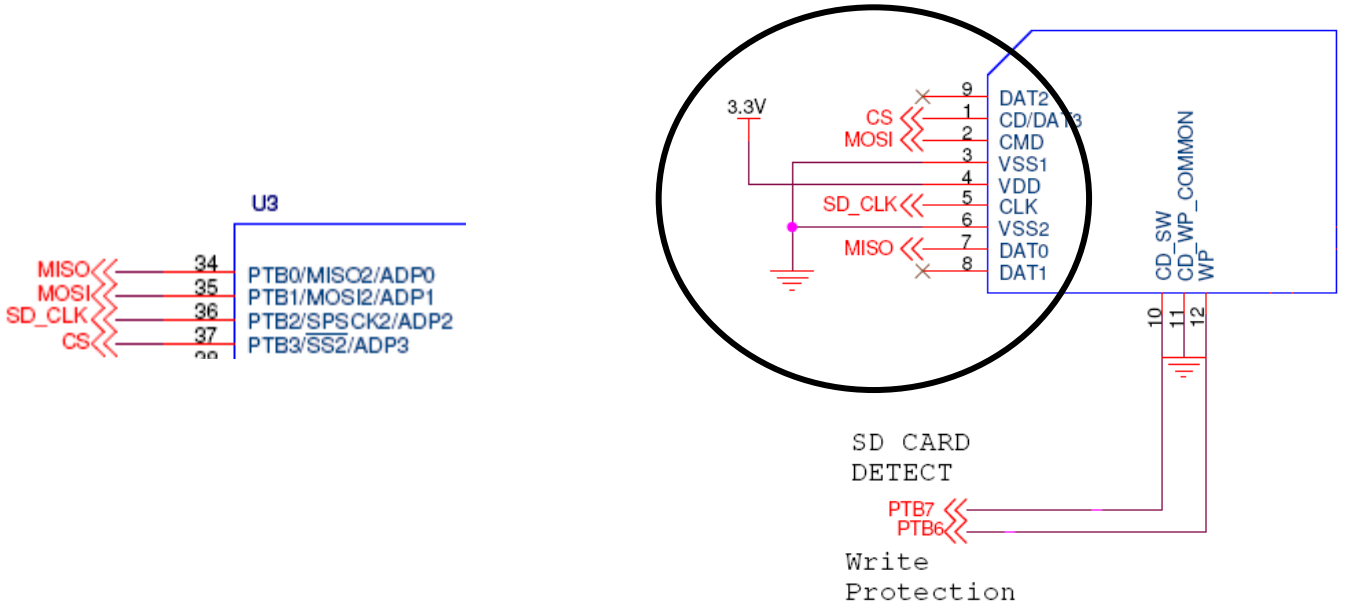


freescale semiconductor

Communicating to the SD Card with an SPI module

► S08 MCU have SPI modules that are fully compatible with the SD card SPI mode.

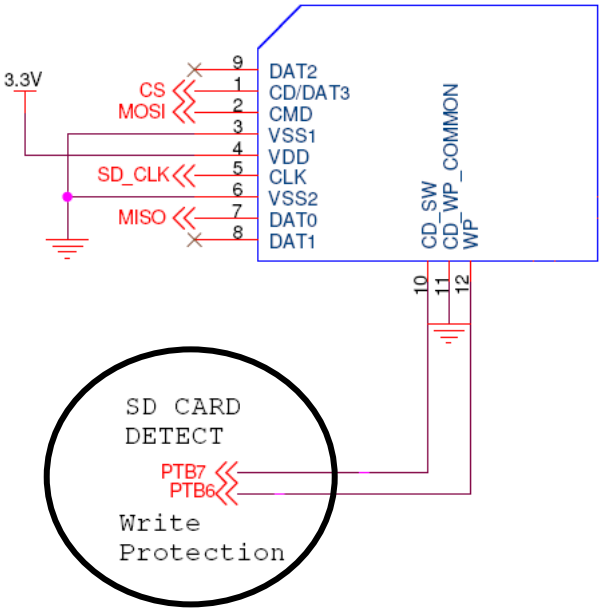
► Master mode, clock baud rate, polarity and phase are all configurable with the SPI module registers.



Additional SD card Signals

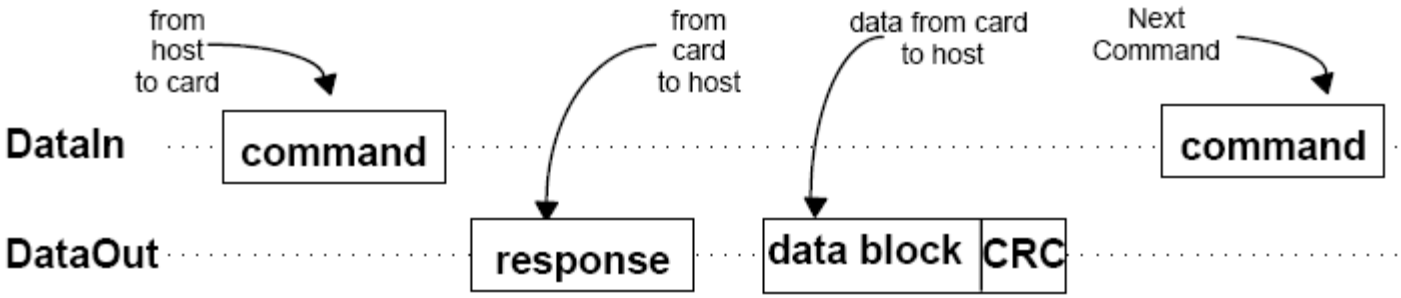
► SD card requires two extra signals for proper communications:

- WP: write protect.
  - Connected to the MCU port (internal pull-up), it provides the capability to detect if the “Lock” switch in the card is on.
- Card detect
  - Provides the signal to detect connection of an SD card in the socket.



SD Card Operation: Logical layer

- Command driven interface

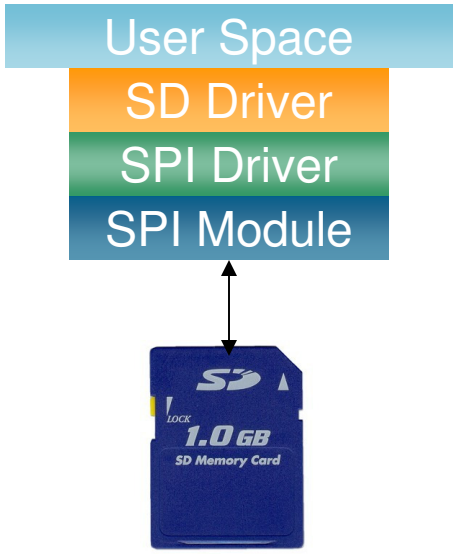


CMD24	Yes	[31:0] data address[10]	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.[4]
CMD25	Yes	[31:0] data address[10]	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until 'Stop Tran' token is sent (instead 'Start Block').
CMD26	No				

Easy to Use

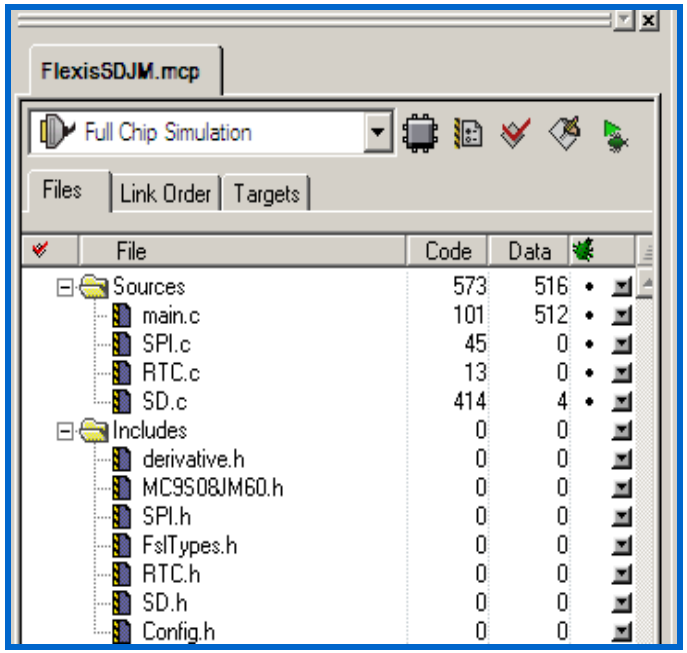
SPI module:	Freescale SPI/QSPI Module
SPI Driver:	Initialization, I/O SPI Logic (SPI.c & SPI.h)
SD Driver:	Initialization, SD Card info,I/O SD Logic (SD.c & SD.h)
User Space:	Data Buffers

Freescale provides this driver for free!



freescale semiconductor

SD Card Driver



SD Card API

SD_Init(void)

SD_Write_Block(Address, Data Pointer)

SD_Read_Block(Address, Data Pointer)

SD_GetCID(void)

SD_GetCSD(void)

freescale semiconductor

FAT-Lite Library

Connectivity!

Share!! music, video,
Photos, files

File System

Different devices
Speaking the same language

freescale™ semiconductor

FAT crash course

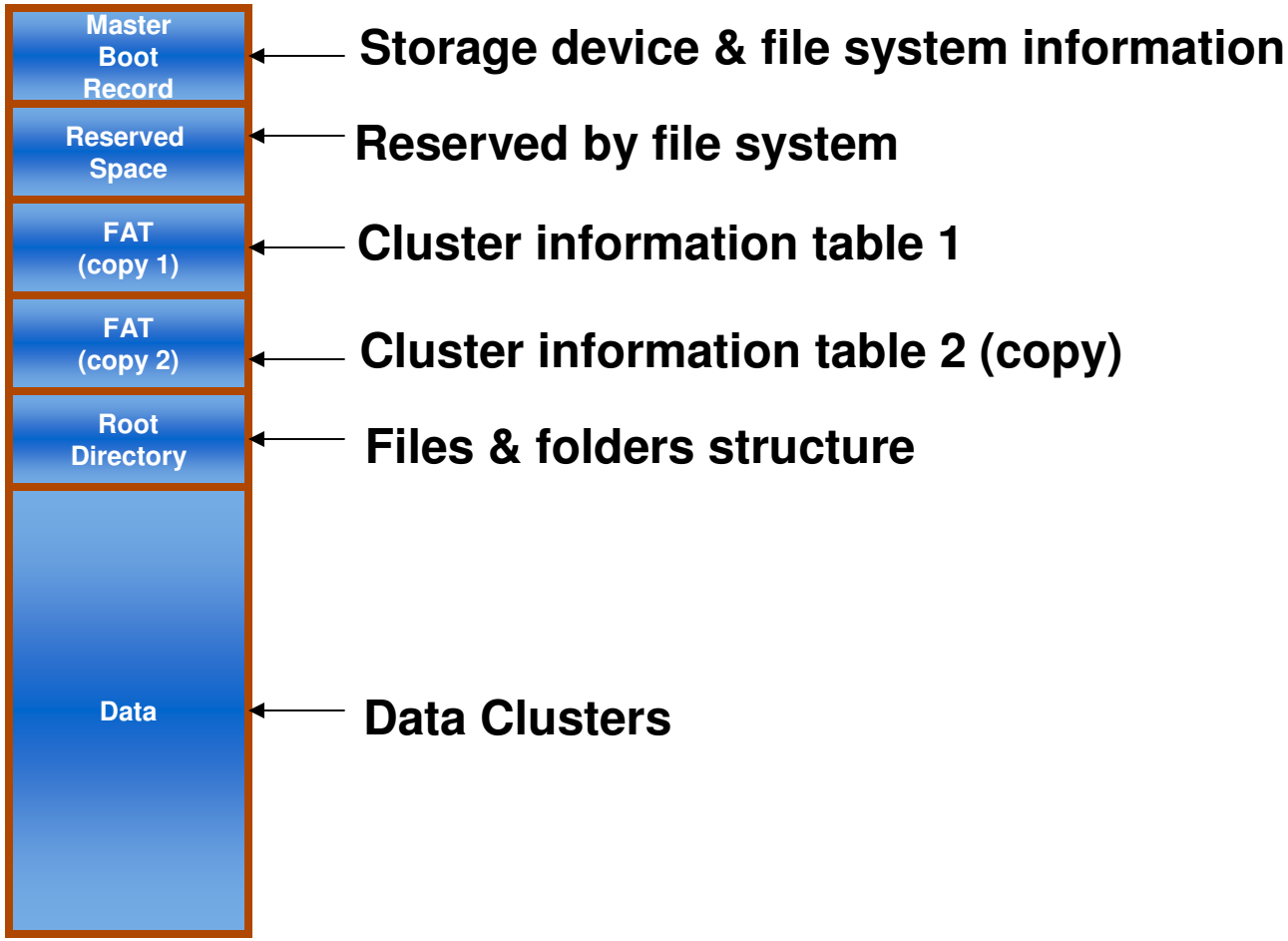
What is it?

- The **File Allocation Table** is a series of addresses that is accessed as a lookup table to see which cluster comes next.
- **Cluster** is single unit of data storage at the FATxx file system logic level.

Developed by Microsoft®

- FAT12 developed in 1977
- FAT16 developed in 1987
- FAT32 developed in 1996 (when Windows® 95 came out)

FAT File system basic structure



FAT versions

FAT12

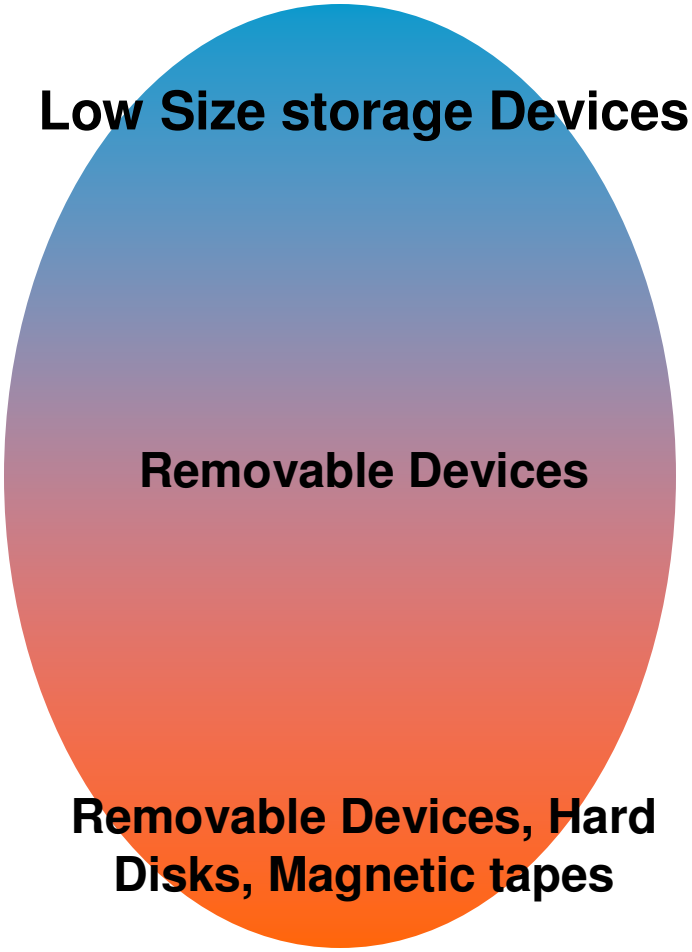
- 2^{12} FAT entries
- 32 MB volume size
- 0.5 KB to 4 KB Cluster Size

FAT16

- 2^{16} FAT entries
- 2 GB volume size
- 2 KB to 32 KB Cluster Size

FAT32

- 2^{28} FAT entries
- 8 TB volume size
- 4 KB to 32 KB Cluster Size



freescale semiconductor

FAT16 & FAT32 support (8 bit)

FAT32 support

- Large amounts of RAM for 32 bit FAT handling
- All cluster/Sector/size algorithms are 32 bit math operations
- Code Size will increased
- More CPU load / more power consumption

FAT16 support

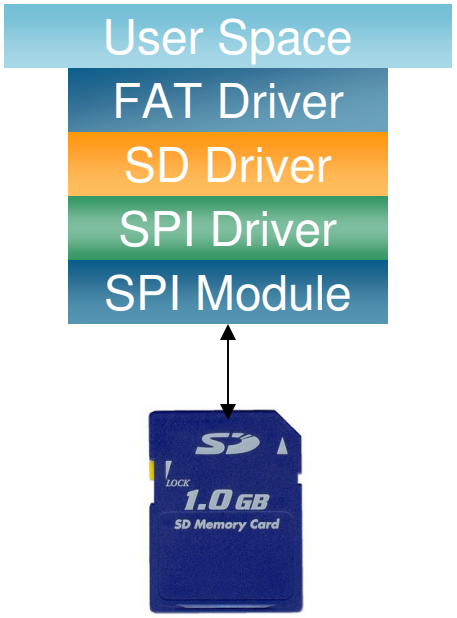
- 16 bit FAT handling is easy for S08 arquitecture
- Cluster/Sector 16 bit operations, 32 bit size algorithms
- Less amount of program memory is needed
- CPU load is for user application not for FAT driver
- Low power operation (less CPU load)

Remember:
FAT 16 supports up to 2 GB!!!!!!

Directly to a PC!

SPI module: Freescale SPI/QSPI Module
SPI Driver: Initialization, I/O SPI Logic (SPI.c & SPI.h)
SD Driver: Initialization, SD Card info,I/O SD Logic (SD.c & SD.h)
FAT Driver: File Open, File Write, File Read functions (FAT.c & FAT.h)
User Space: Data Buffers

How to change drivers?



freescale semiconductor

Software Interface Layer

Path: D:\Projects\DEMOFLEXISJMSD\DEMOFLEXISJMSD_DataLogger\Sources\Fat.h

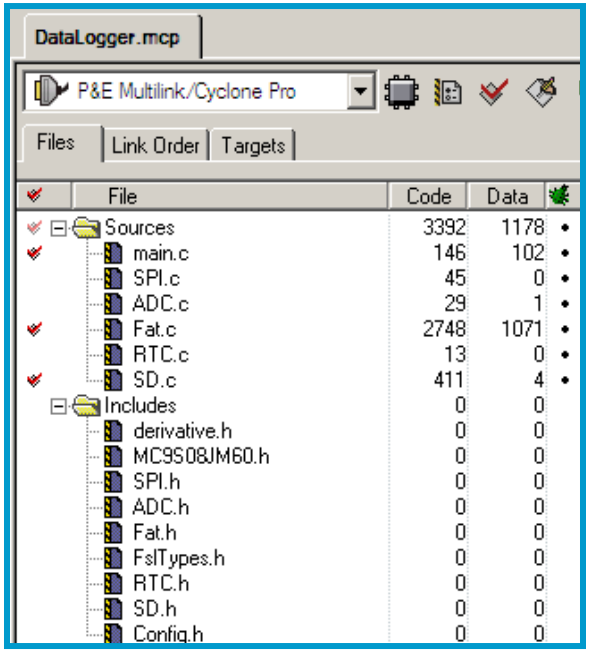
```
#ifndef __Fat__
#define __Fat__

/* Includes */

#include "FslTypes.h"
/************************* HIL ****************************/
/**********************************************************/

/* Includes */
#include "SD.h"      // SD Card Driver


/* Storage HIL */
#define GetPhysicalBlock(A,B)        (void)SD_Read_Block(A,B);
#define StorePhysicalBLock(A,B)      (void)SD_Write_Block(A,B);

/**********************************************************/
/**********************************************************/
```

Transparent communication between layers

Including your own driver Is very easy!

Path: D:\Projects\DEMOFLEXISJMSD\DEMOFLEXISJMSD_DataLogger\Sources\SD.h

```
#ifndef __SD__
#define __SD__

/* Includes */

#include "FslTypes.h"
/************************* HIL ****************************/
/**********************************************************/

/* Includes */
#include "SPI.h"      // SPI Driver


/* HIL */
#define ReadSPIByte      SPI_Receive_byte
#define WriteSPIByte     SPI_Send_byte

/**********************************************************/
/**********************************************************/
```



FAT API

FAT_Read_Master_Block(void)
FAT_FileTableSearch(FileName,Function)
FAT_File_Write(DataPointer, Size)
FAT_File_Read(DataPointer)
FAT_Close(void)

Using USB, SD Card and FAT on Flexis MC9S08JM60

Practical examples

Hands on

- SD Card Reader demo
- FAT demo using USB terminal
- Data logger example

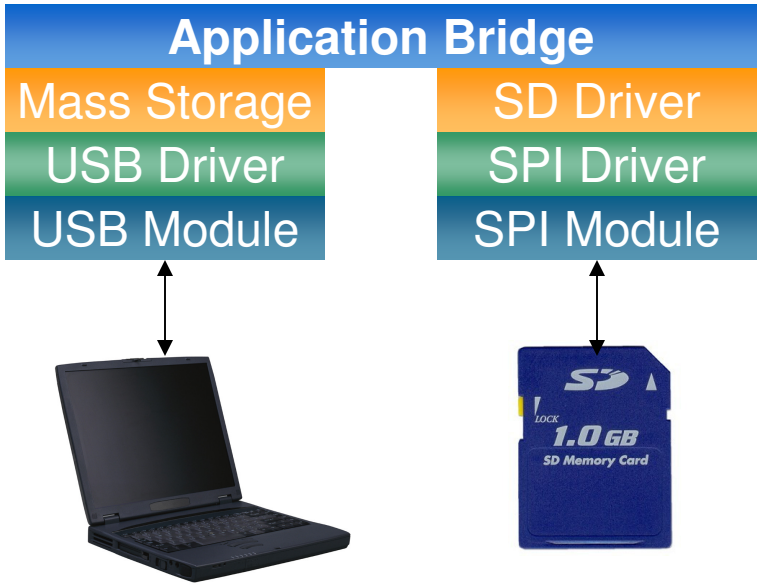
DEMOFLEXISJMSD board

freescale™ semiconductor

SD Card Reader

This example shows how to implement a commercial SD Card reader using the Flexis JM microcontrollers

Host system (PC) send the blocks that the native file system needs. In other words, the microcontroller is a logical bridge between host file system and external physical memory



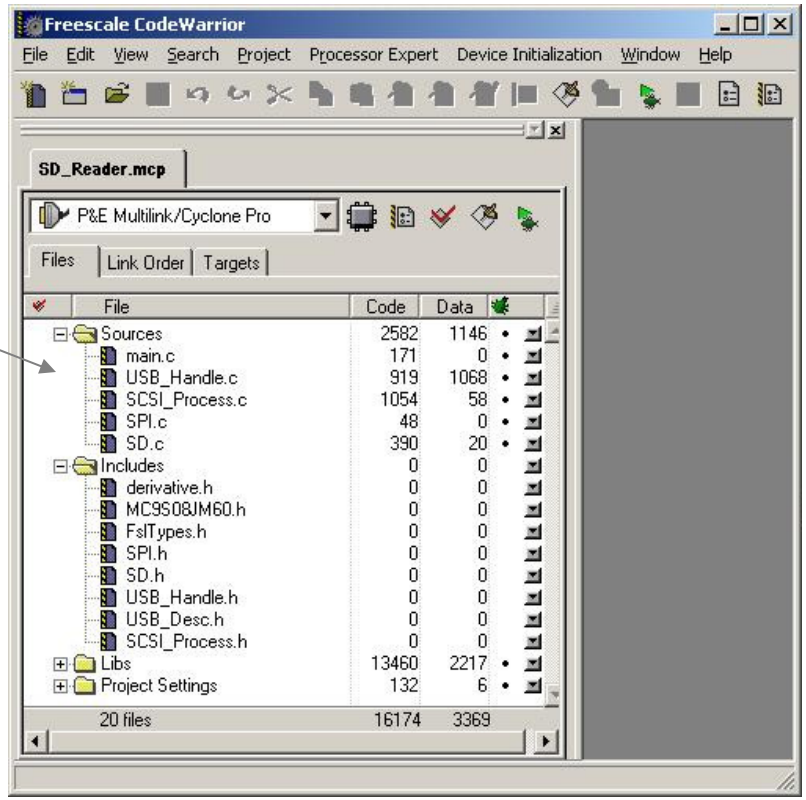
freescale semiconductor

What's inside SD Card Reader

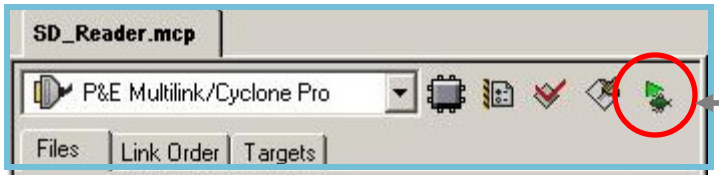
- Open CodeWarrior 6.x (6.1 and above)
- Open project SD Card Reader project

Source Files

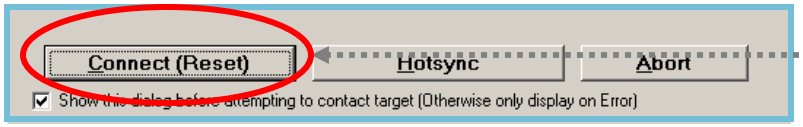
Main	main routine
USB_Handle	USB Layer control
SCSI_process	Mass Storage and SCSI
SPI	SPI driver
SD	SD Card Driver



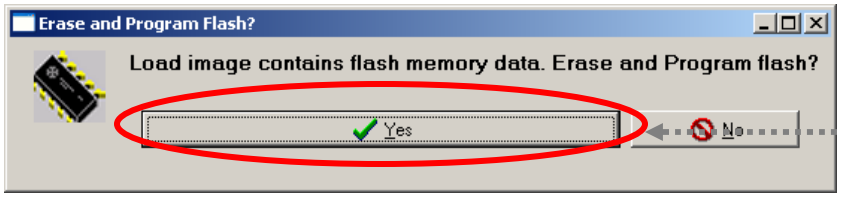
Downloading SD Card Reader…



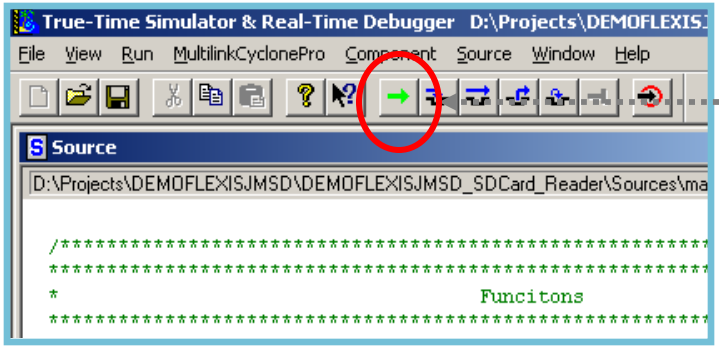
Clic on “Debug” icon



Clic “connect” on PEMICRO Connection manager



Load the new project in memory



Run the application

freescale semiconductor

SD Card Reader - Result

Did you hear the USB sound?

Open a Windows Explorer

A new Removable Disk has
Been placed.

**Open it, and drag & drop files
In it!**

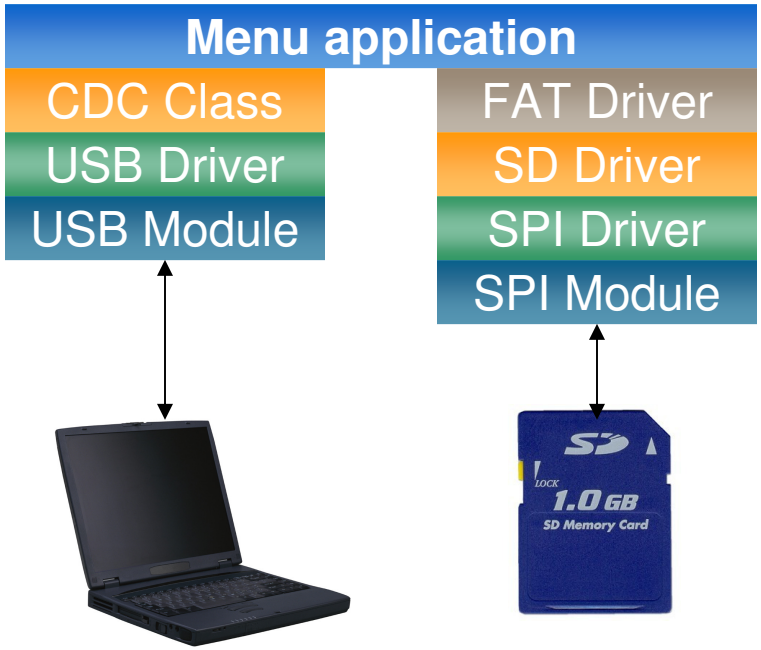
FAT Demo Using USB Terminal

The purpose of this demo is:

- Read/Write files in the SD Card using FAT-lite driver
- Use the USB CDC class to emulate a PC COM port

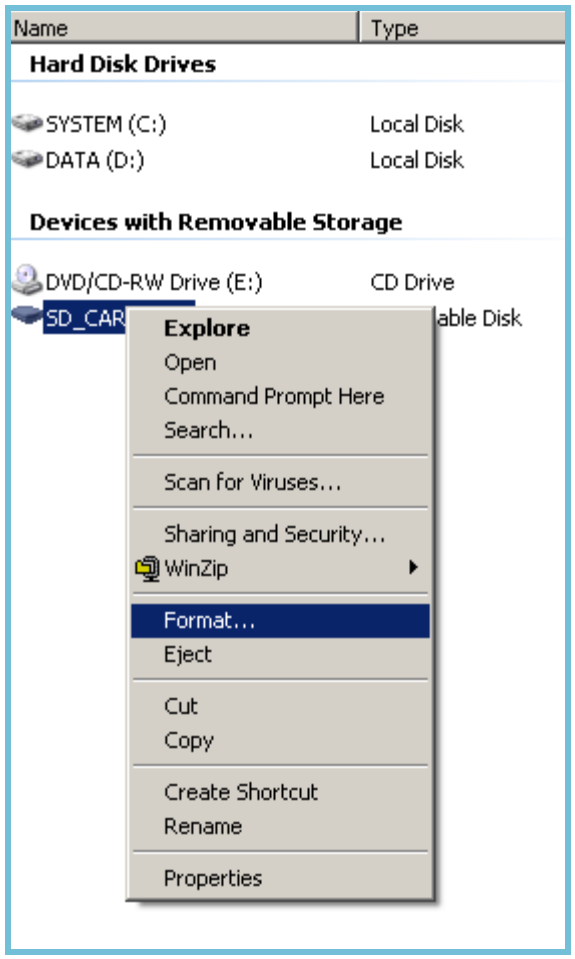
What we need?

SD card driver
FAT16 driver
USB / CDC class drivers



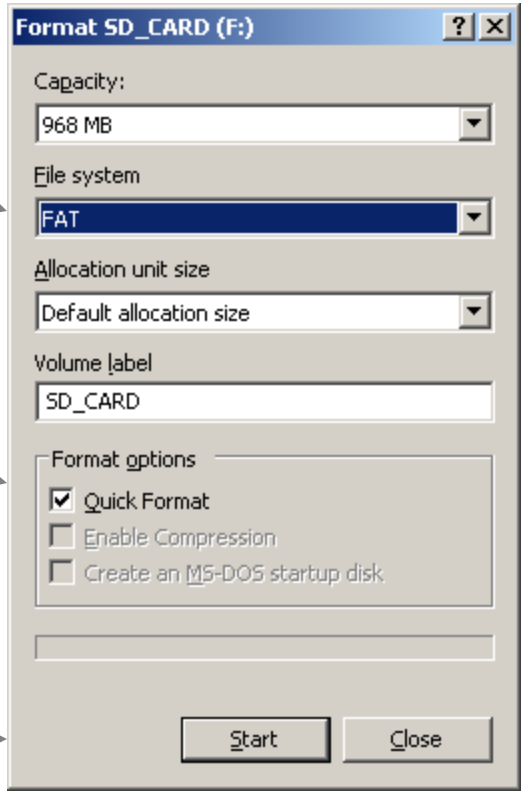
Formatting the SD card to FAT16

Before start, let's format the SD Card using the SD Card Reader Demo



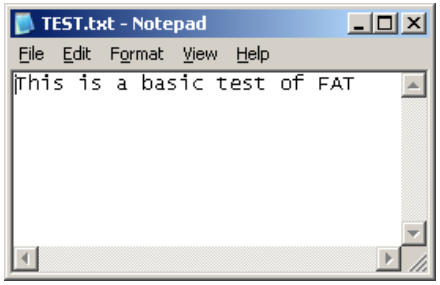
Change to FAT File system

Use Quick format option

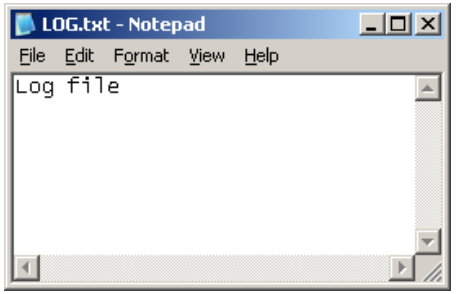


freescale semiconductor

Files for Labs

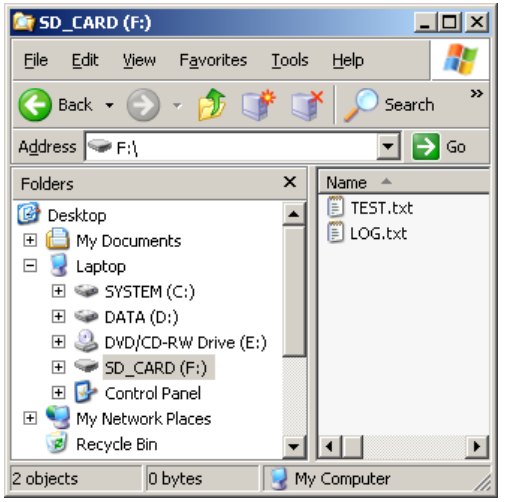


Create TEST.txt file, write something in it and save it in the SD Card



Create LOG.txt file, write “Log file” in it and save it in the SD Card

SD Card content must be



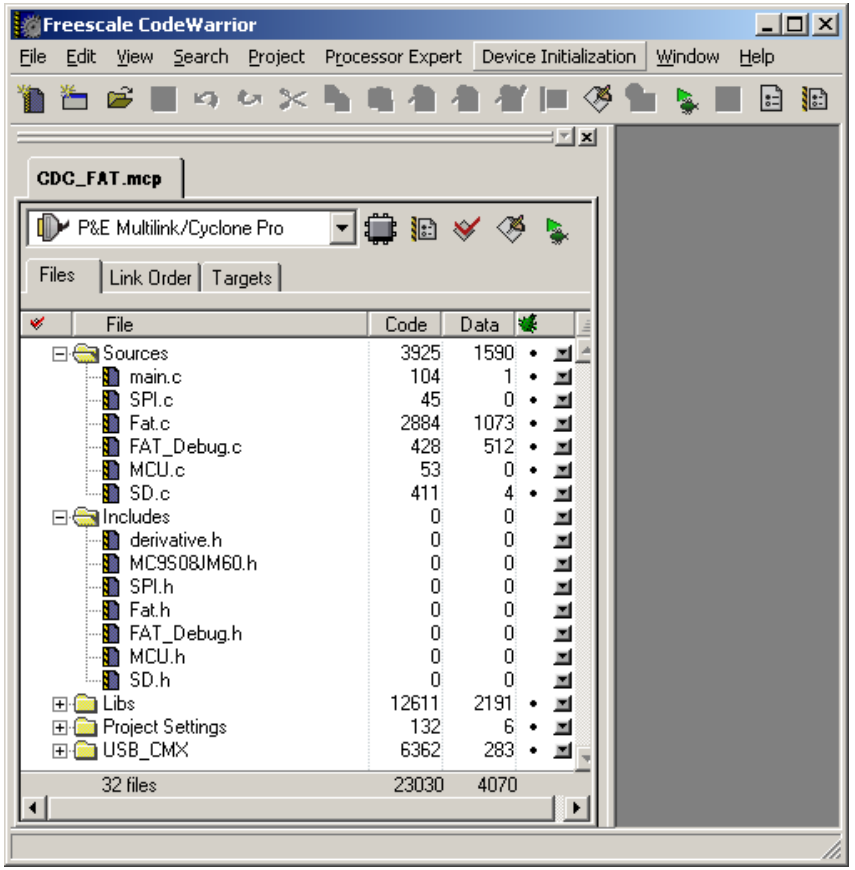
freescale semiconductor

What's inside FAT/CDC demo

- Open project CDC Reader project

Source Files

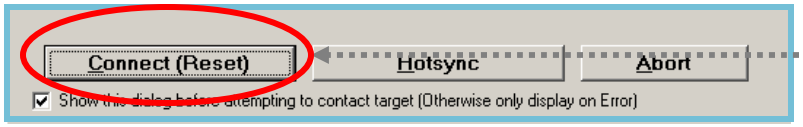
Main	main routine
USB_CMX	USB / CDC layers
SPI	SPI driver
SD	SD Card Driver
FAT	FAT driver
FAT_Debug	terminal demo



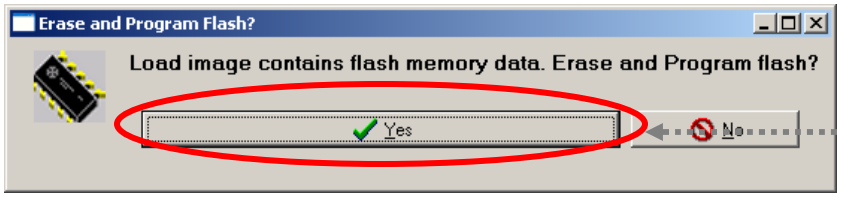
freescale semiconductor

Downloading FAT/CDC demo

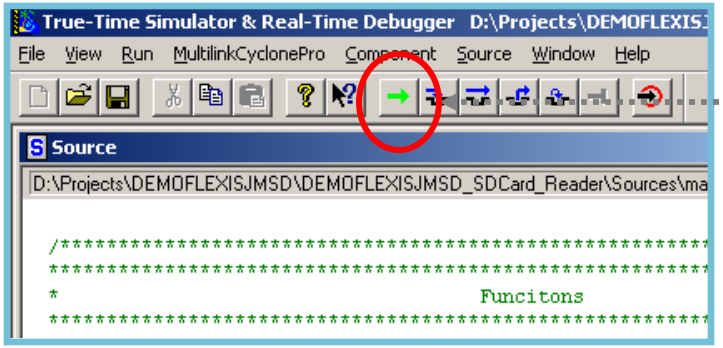
Clic on “Debug” icon



Clic “connect” on PEMICRO Connection manager



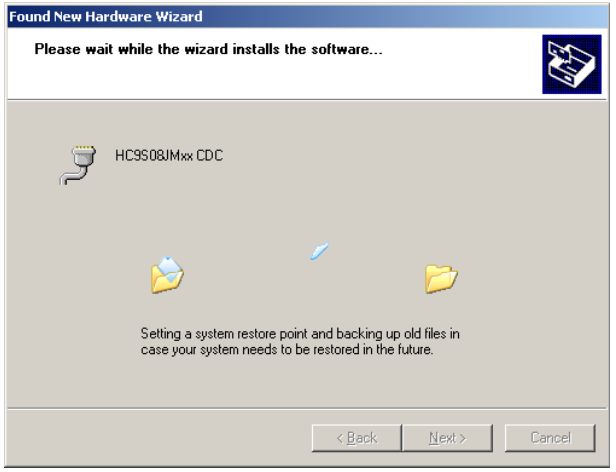
Load the new project in memory

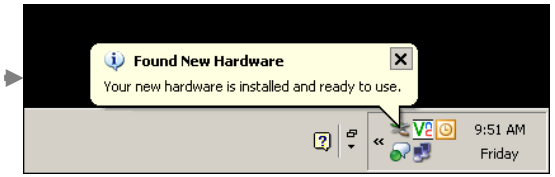


Run the application

USB Driver Request

Driver:
HC9S08JMxx.inf

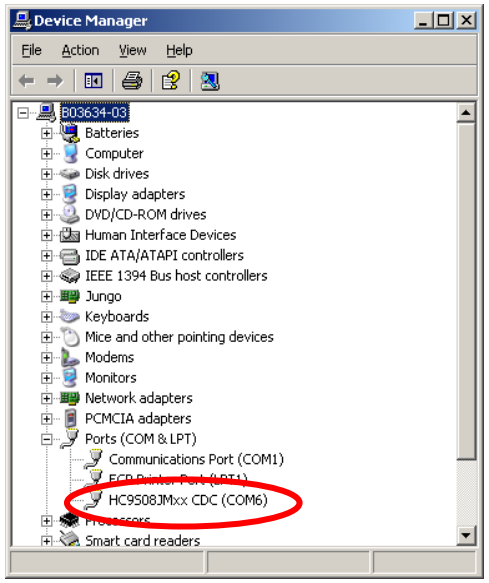




Done!

freescale semiconductor

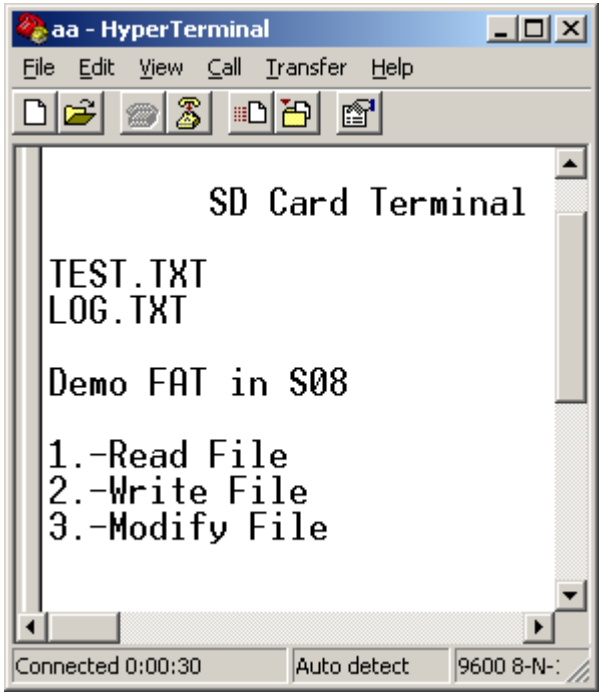
Open COM port



Open an Hyperterminal:
9600 8N1 no flow control

Press the button on the board!

Look the file list of the SD Card!



Go to Device Manager (Windows key + Pause)
Open the Ports (COM & LPT)
and check that one COM was assigned to DEMOFLEXISJMSD board

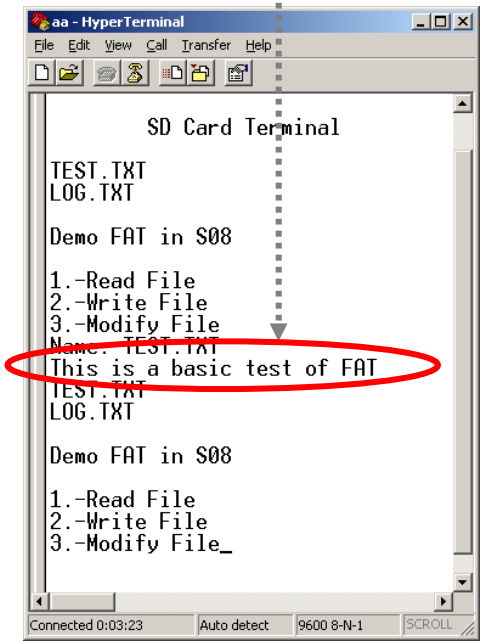
freescale semiconductor

Let's FAT

Read

- Press 1 for "Read file"
- type TEST.TXT (upper case please)

File content!

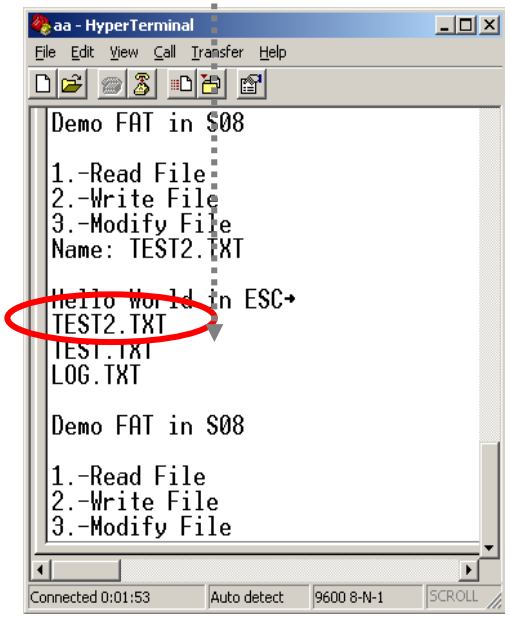


Read TEST2.TXT
For double check

Write (create a file)

- Press 2 for "Write file"
- type TEST2.TXT (upper case please)
- Enter some text and press "ctrl+z"

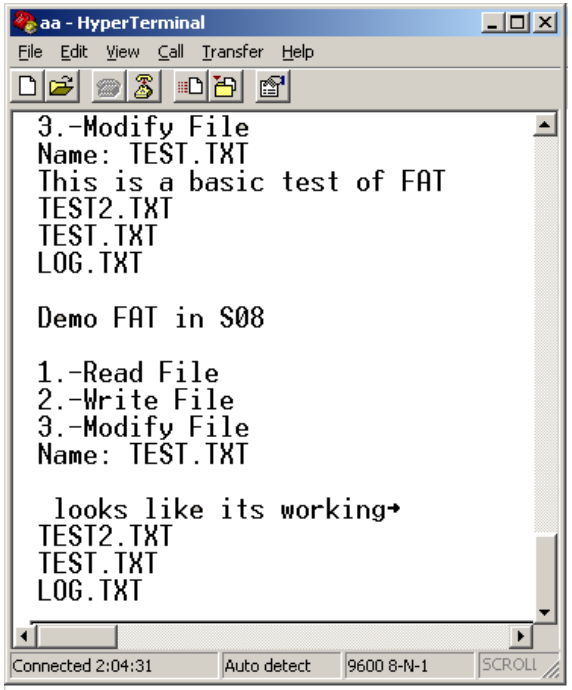
New File!



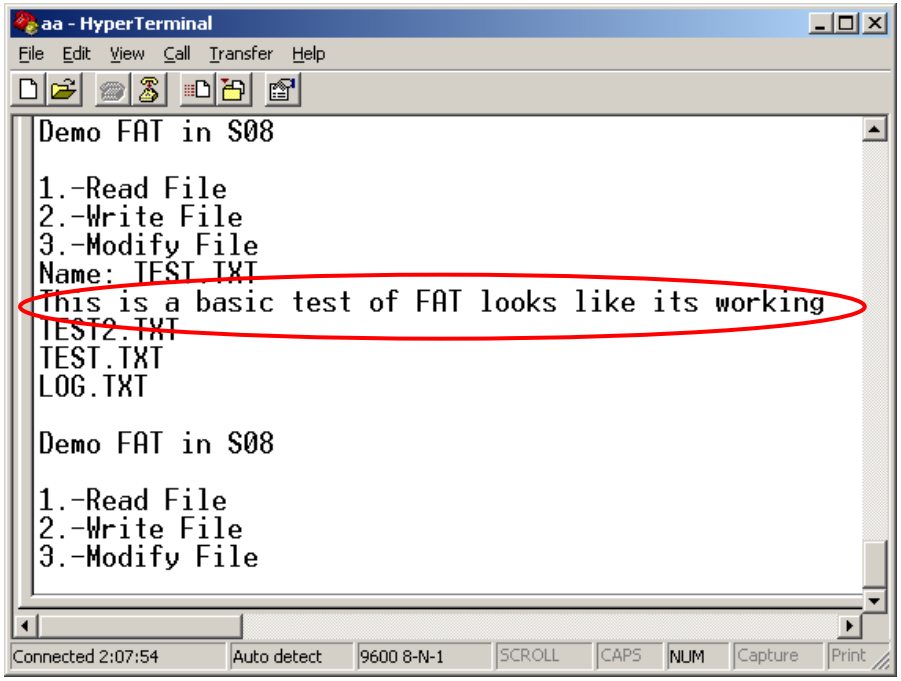
freescale semiconductor

Modify (add content to file)

- Press 3 for “Modify file”
- type TEST.TXT (upper case please)
- Enter text to add and press “ctrl+z”

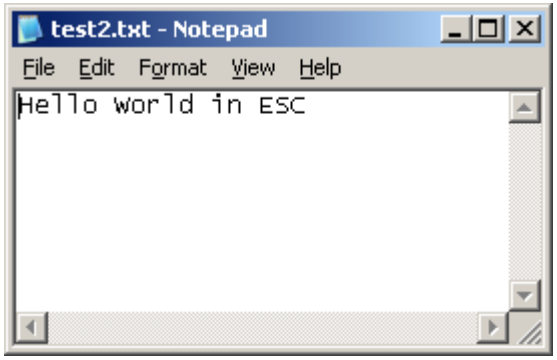


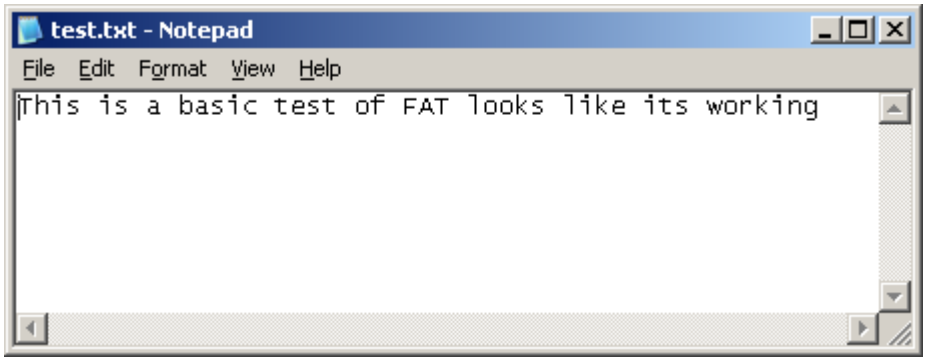
Read again TEST.TXT



FAT/CDC conclusion

Load SD Card Reader Code in DEMOFLEXISJMSD and Check the files inside the SD Card





Conclusion

Read files

Create files

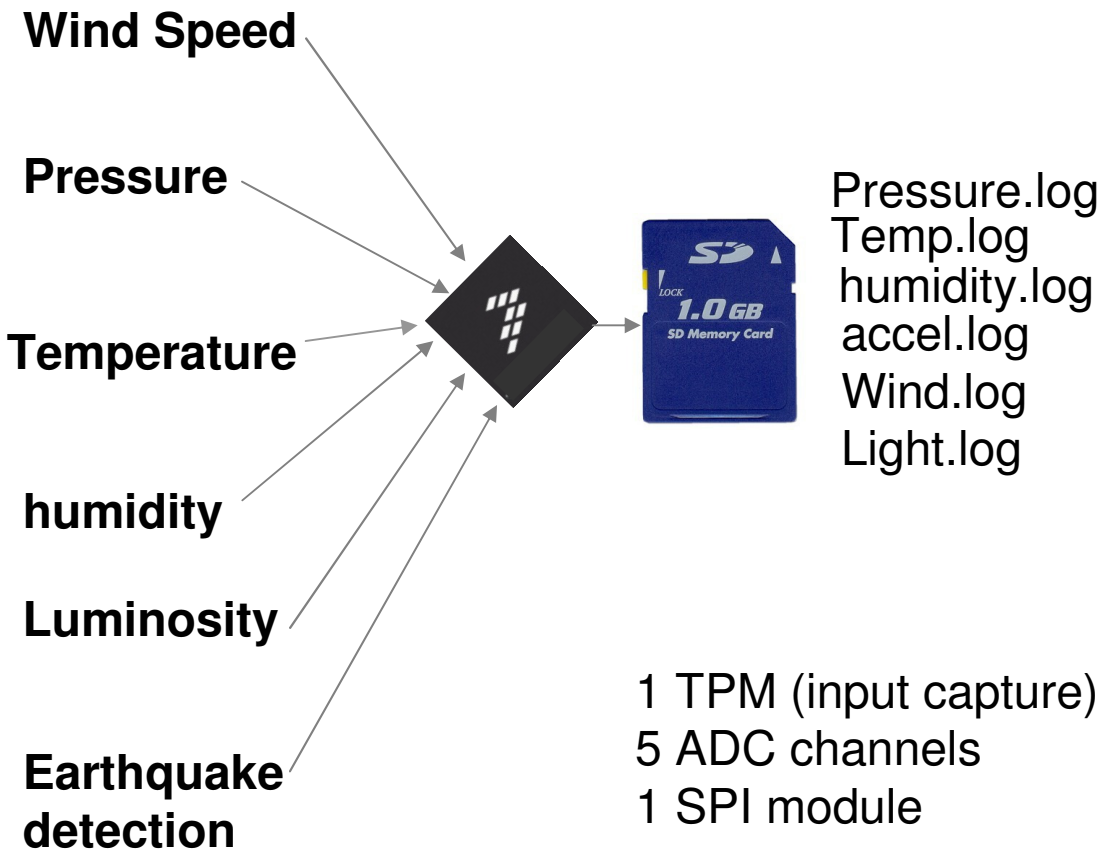
Open files and add content

freescale semiconductor

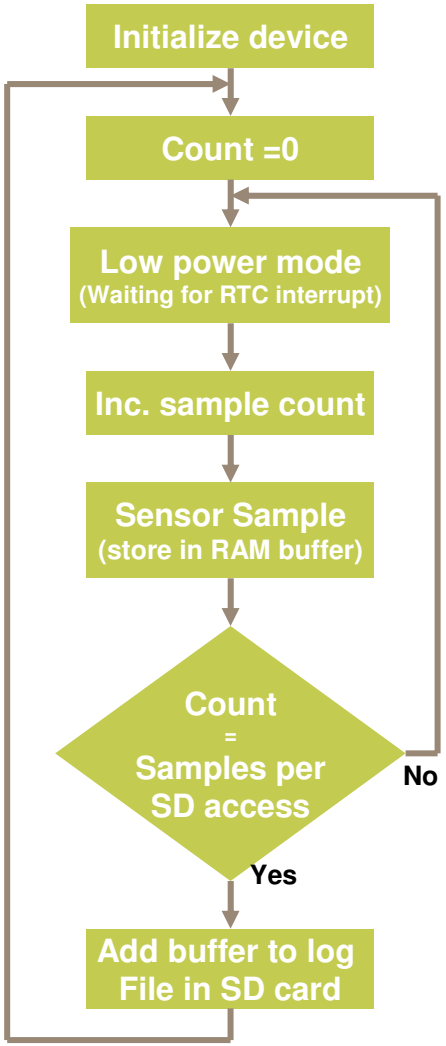
Low Cost Data Logger with mass data storage

Use the SD card to store large amount of data from different sensors

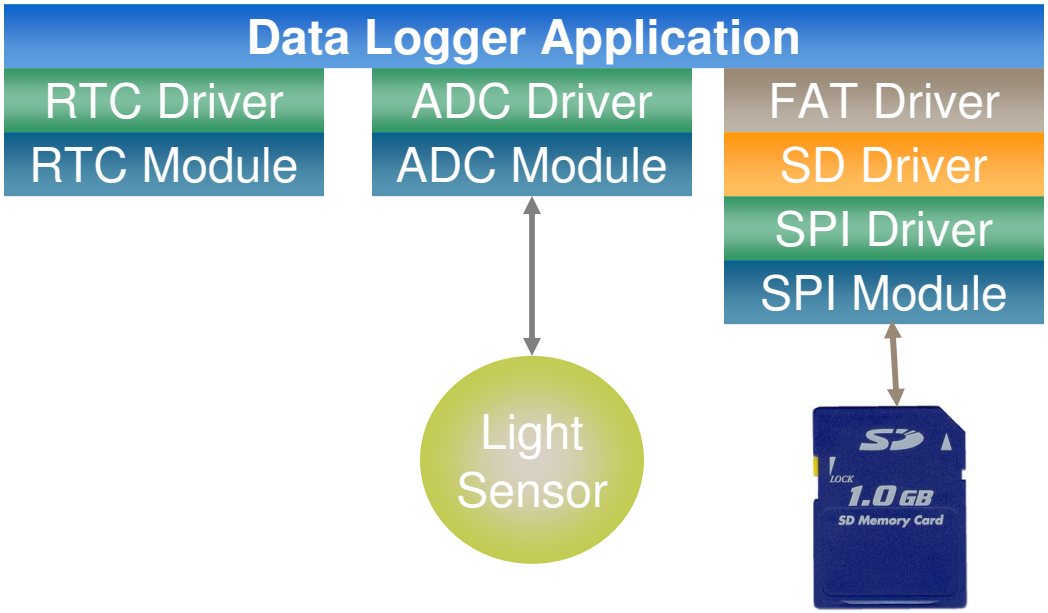
- **Low power**
  MCU will be in low power mode, just wake up to take sensors sample periodically
- **Low cost**
  Sensors, SD Card socket and Low cost Freescale MCU is needed
- **Organized data**
  Each sensor can have a different log file



Data logger example



- Use the RTC period as a sampling time base
- Configure how many samples for each SD Card access is good for your application (depends of battery life time and data importance)



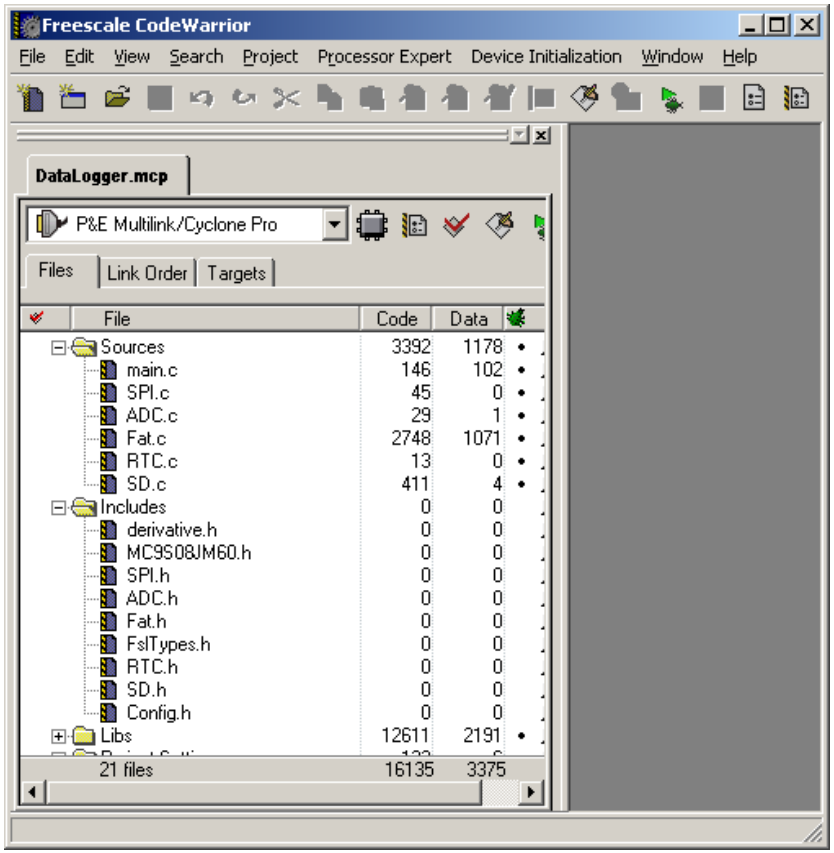
freescale semiconductor

Inside data logger demo

- Open project Data logger project

Source Files

Main	main routine
SPI	SPI driver
SD	SD Card Driver
FAT	FAT driver
ADC	ADC driver
RTC	RTC Driver



Changing Data Logging Parameters

Open Config.h file

Number of samples that MCU will store before write SD Card

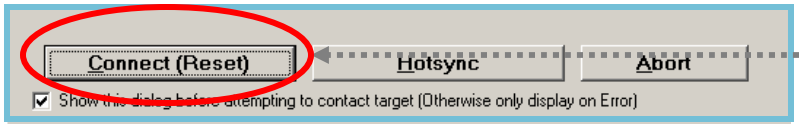
Sampling Time

File name
In use

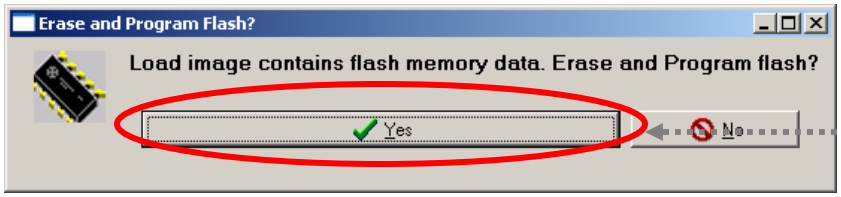
```
/****** User Defines *****************/
#define DATA_SAMPLES     50      /* How many samples will read before store the buffer
                                 in the SD Card */

#define SAMPLE_TIME_05_SEG       /* SAMPLE_TIME_05_SEG Sampling Interval 0.5 seg*/
                                 /* SAMPLE_TIME_1_SEG Sampling Interval 1 seg*/
                                 /* SAMPLE_TIME_2_SEG Sampling Interval 2 seg*/
                                 /* SAMPLE_TIME_3_SEG Sampling Interval 3 seg*/
                                 /* SAMPLE_TIME_6_SEG Sampling Interval 6 seg*/


#define FILE_NAME    "LOG.TXT"  /* LOG file name (Previously created)*/


/******* Please Do not Modify ********/
#define SAMPLES_WRITE   DATA_SAMPLES *2


#endif /* __Config__ */
```

Downloading Data logger demo

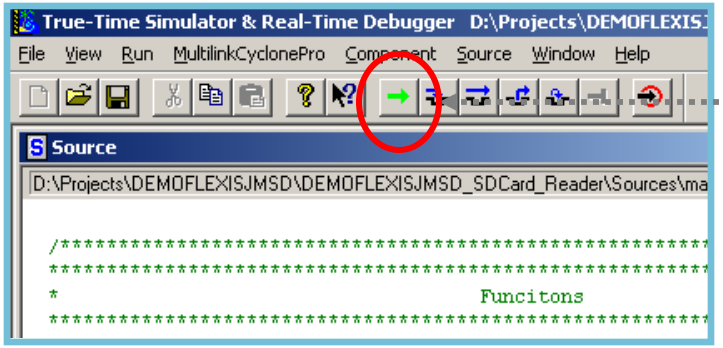
Clic on “Debug” icon



Clic “connect” on PEMICRO Connection manager



Load the new project in memory



Run the application

freescale semiconductor

Data logger results

User LED (red) will toogle each SD Card access.

SD Card Access period = Number of samples x Sample period

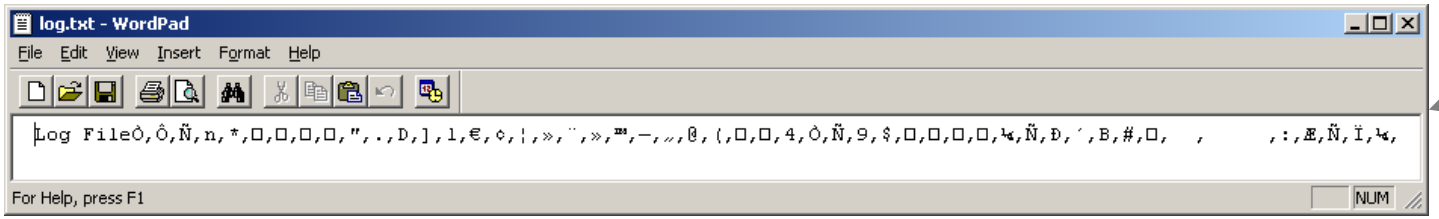
Move your hand over light sensor while waiting for one or two LED blinks

Remove the SD Card from DEMOFLEXISJMSD board

Load the SD Card reader code again

Use to read the LOG.txt file in the SD Card

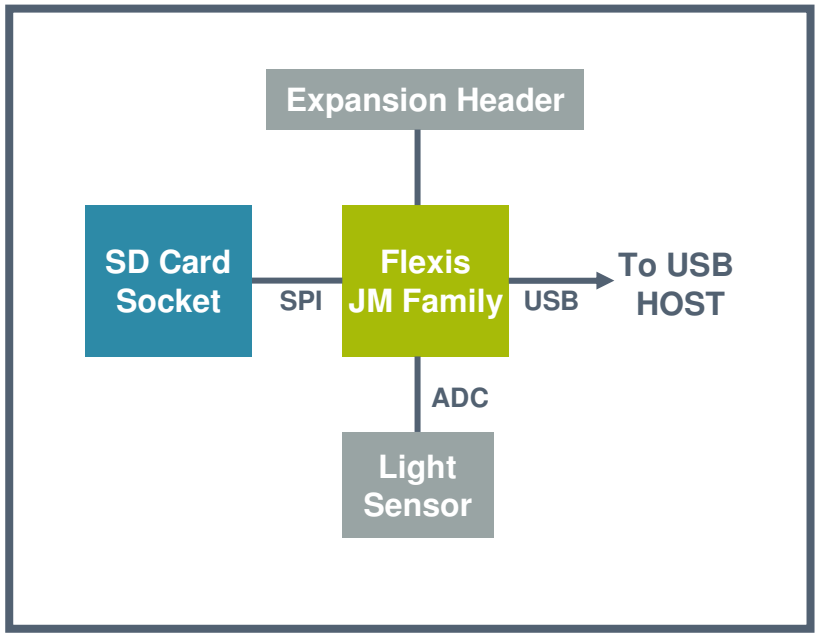
Each sample is separated by a coma (,)



Examples information

- **Data Logger**
  - Flash 3.9KB
  - RAM 1.4KB including SD Read/Write buffers
- **SD Card Reader**
  - Flash 4.1KB
  - RAM 1.4KB including SD Read/Write buffers
- **CDC_FAT terminal Demo**
  - Flash 9.4KB
  - RAM 2KB including SD Read/Write buffers and USB stack

DEMOFLEXISJMSD



DEMOFLEXISJMSD block diagram

More information on Reference Design 104, look for DRM104 at:
www.freescale.com

Conclusions

► Small MCU systems can handle simple file systems to interface with operating systems.

► File systems allow applications to exchange data without compatibility issues.

► USB enabled MCU will open up possibilities of data storage and exchange applications.

► FAT and SD card libraries for low cost MCU aren't necessarily complex if you are willing to sacrifice some performance.