PROJECT REPORT

SD-Support for Tmote Sky

Submitted To

Computer Science Department
(Course: Network Embedded Sensing Network)

Johns Hopkins University

By

Santosh Bahir
Roshan Krishnan

12-Dec-2008

Acknowledgement

We first thank to Dr. Andreas Terzis for giving us opportunity to work on this project. We also thank to Razvan Musaloiu-E for guidance, feedback, comments and suggestions which were immensely helpful throughout the entire project without which project would not have been possible.

Table of contents

1. Introduction

The architecture of sensor node is fairly simple consisting of different hardware unit for computation, storage, communication, etc. Local storage is required on the sensor node to store the data before forwarding to base station, to capture detailed information of the event, to meet the memory constraints. In case where distributed storage/database is maintained, the need of storage at node is further increased.

2. Motivation/Problem Statement

SD/MMC card provides inexpensive and large persistent storage. It helps application to log data for long term. It can also serve as temporary buffer.

In this project, we have interfaced SD Card to Tmote Sky using the SPI interface. It adds one new module to support this SD card.

We have provided Block Storage and Log Storage abstraction for SD Card.

3. Related Work

Shimmer: Shimmer is wireless sensor platform. Shimmer supports an interface with SD card with size up to 2GB for offline data capture. The microcontroller in shimmer is MSP430 (version F1611) which is same as that of Tmote Sky. Shimmer communicates with SD card using SPI protocol. The USART port required for this SPI communication is made available to the microSD Card in SHIMMER.

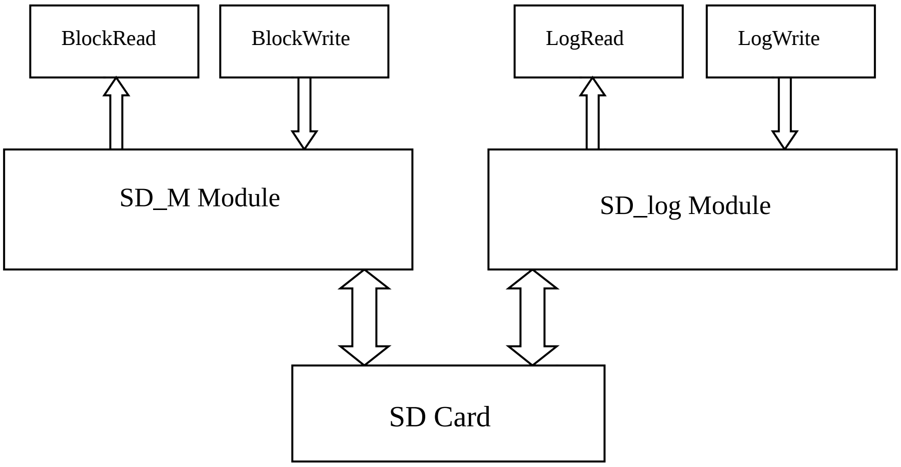
But in Tmote Sky, the USART port is not available on the external expansion slot and hence it cannot communicate utilizing the USART port of the microcontroller. For this purpose a technique called Bit Banging is used to simulate the operation of the shift registers required for SPI communication.

4. Architectural/Technical Description

1. Architecture

The architecture of this component follows the component concept of TinyOS. The new components BLOCKSTORAGEP and SD_log have been written. They provide an interface with the SD Card similar to that available for flash storage present on the mote. Specifically, interfaces provided are BlockWrite, BlockRead for Block Storage abstraction and LogWrite and LogRead for Log Storage abstraction.

Configuration component BLOCKSTORAGEC provides the wiring for the BLOCKSTORAGEP component.



Module Architecture Diagram

1.1. Interfaces used

Boot:

This interface is used to initialize the SD card on booting the mote.

GeneralIO:

This interface is used to communicate to SD card on Microcontrollers pins. Total of 5 microcontroller pins are used which are also connected to the expansion slot. The details of the pins are

Microcontroller Pin No	Expansion Slot Pin.	Direction	Functionality	Description
60	ADC0	Out	mmcCD	Card Detection
34	UART0TX	Out	Data Out	Data Output Channel Input to the Card
35	UART0RX	In	Data In	Data Input Channel Input to the Card
61	ADC1	Out	Clock	Clock
62	ADC2	Out	mmcCS	Card Selection

1.2. Interface provided

BlockWrite

This interface provides following commands and interfaces

Commands: write, erase, sync

events: writeDone, eraseDone, syncDone

The handler for these events generators and commands are present in BLOCKSTORAGEP components.

BlockRead:

This interface provides following commands and interfaces

Commands: read, computeCrc, getSize

events: readDone, computeCrcDone

The handler for these events and commands are present in BLOCKSTORAGEP components

LogWrite:

This interface provides following commands and interfaces

Commands: append, currentOffset, erase, sync

events: appendDone, eraseDone, syncDone

The handler for these events and commands are present in SD_log components

LogRead

This interface provides following commands and interfaces

Commands: read, currentOffset, seek, getSize

events: readDone, seekDone

The handler for these events and commands are present in SD_log components

SD

This interface provides following commands and interfaces

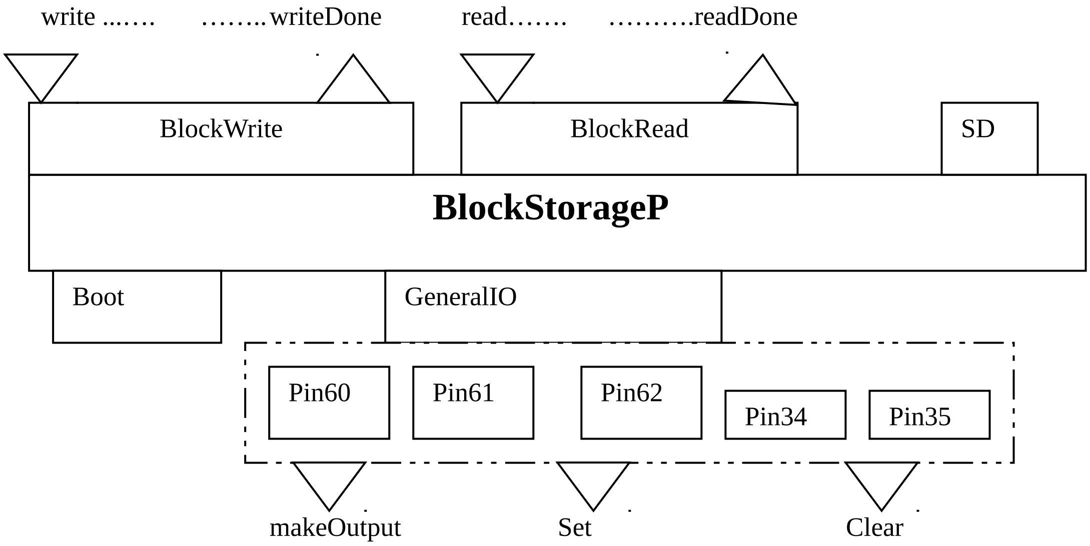
Commands: init, setIdle, setBlockLength

The handlers for these commands are present in BLOCKSTORAGEP components. Note that, this interface is exclusively used for the implementation of BLOCKSTORAGEP module.

2. Technical Description

2.1. Block Abstraction:

The new component BLOCKSTORAGEP, related components and interacting interfaces are shown in below diagram. All the above mentioned commands and events are handled in the BLOCKSTORAGEP module The below diagram shows that BLOCKSTORAGEP modules provides BlockRead and BlockWrite nterfaces and uses Boot and GeneralIO interfaces. Downward triangle shows that the commands of these interfaces used by other modules or components. Upwards triangle shows the event signaled by these interfaces.



SD Block Storage Component and Interface Diagram

BLOCKSTORAGEP module provides the commands for interfacing with SD Card. Based on these commands the BLOCKSTORAGEP module sets/resets the output pins with help of GeneralIO module. The BLOCKSTORAGEP is also responsible for generating the appropriate events in response to the commands. The data transfer with the SD Card is always performed in blocks of 512bytes.

The functions and tasks descriptions of block abstractions (BLOCKSTORAGEP module) are below:

1. **error_t BlockRead.computeCrc(storage_addr_t addr, storage_len_t len, uint16_t crc)**

   CRC is not calculated here as sending data to SD card does not require it. Though computeCrcDone is signaled to above layer.

2. **error_t BlockWrite.sync()**

   This syncs the data present in the buffer to SD card

   Parameter: none

   Return: Status Success/Error code

3. **uint8_t SD.init()**

   Initialize the SD card on boot-up or power on. It sets the direction of all the ports used and put the SD card in idle status.

   Parameter: None

   Return: Status Success/Error code.

4. **uint8_t spiSendByte (uint8_t data)**

   It sends one byte data on pin 34.

   Parameter: data one byte data to be sent

   Return: Status Success/Error code

5. **uint8_t setIdle ()**

   It sets SD card in idle mode.

   Parameter: None

   Return: Status Success/Error code

6. **void sendCmd(const uint8_t cmd, uint32_t data, const uint8_t crc)**

   It sends the Commands to SD card. Commands are always 6 bytes. First byte is actual command, next 4 bytes are argument to commands and last byte is crc for this command.

   Parameter: cmd Command to be sent to SD card

   data Argument to command

   crc checksum for command

   Return: None

7. **uint8_t getResponse()**

   It gets the response for the command sent

   Parameter: None

   Return: Status Success/Error code

8. **task void SDWrite()**

   It writes the data in blocks to SD card.

9. **error_t BlockWrite.write(uint32_t address, void * buffer,uint32_t count)**

   This post the task to write *count* bytes of data present at *buffer* to SD card at *address*

   Parameter: address Start address where data should be written on card

   buffer pointer to write buffer

   count number of bytes to be written

   Return: Status Success/Error code

10. **uint8_t SD.setBlockLength (const uint16_t len)**

    It sets the blocklength to len size. The default block size is 512 bytes.

    Parameter: len Size of block

    Return: Status Success/Error code

11. **uint8_t getXXResponse(const uint8_t resp)**

    it gets the response for the command sent and checks if it is equal to *resp.* If it is equal it returns success else fail

    Parameter: resp expected response

    Return: Status Success/Error code

12. **uint8_t checkBusy()**

    It checks if the card is busy.

    Parameter: None

    Return: Status Success/Error code

13. **task void SDRead()**

    This reads the data from SD card.

14. **uint8_t BlockRead.read(uint32_t address, void* buffer, uint32_t count)**

    It posts the task to read *count* bytes at address buffer from starting address *address* from SD card.

    Parameter: address Start address of data to read from SD card.

    buffer pointer to read buffer

count Number of bytes to be read.

Return: Status Success/Error code

15. uint32_t BlockRead.getSize()

It reads the size of cards in bytes.

Parameter: None

Return: Card Size

16. task void SDErase()

This task erases the SD card. The range used to erase blocks is 15 blocks at a time.

17. uint8_t BlockWrite.erase()

This posts the task to erase the SD card.

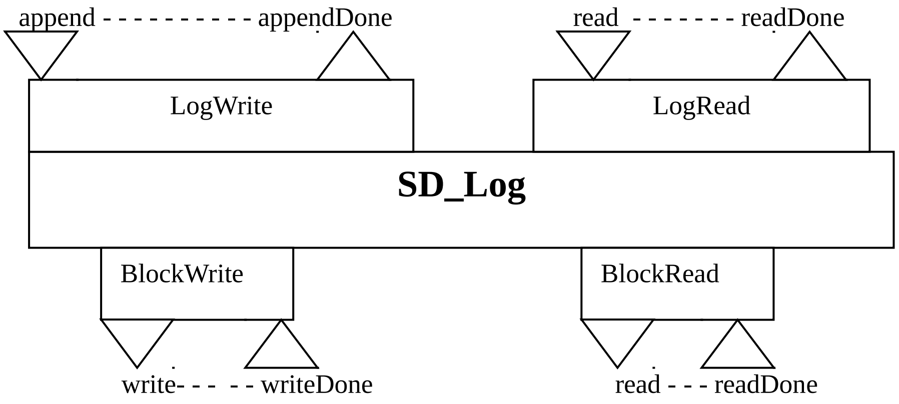
Parameter: None

Return: Status Success/Error code

2.2. Log Abstraction:

The new component SD_log, related components and interacting interfaces are shown in below diagram. All the above mentioned commands and events, present in Log Abstraction, are handled in the SD_log module.

The below diagram shows that SD_Log modules provides LogRead and LogWrite interfaces and uses BlockWrite and BlockRead interfaces. Downward triangle shows that the commands of these interfaces used by other modules or components. Upward triangle shows the event signaled by these interfaces.



SD Block Log Component and Interface Diagram

SD_log module contains handler for the commands and events to interact with the SD card. Internally SD_log uses Block abstraction to read, write data from SD Card.

The functions and tasks descriptions of block abstractions (SD_log module) are below:

1. **void BlockWrite.writeDone(storage_addr_t addr, void* buf, storage_len_t len, error_t error)**
   No action is performed.
2. **void BlockWrite.eraseDone(error_t error)**
   No action is performed.
3. **void BlockWrite.syncDone(error_t error)**
   No action is performed.
4. **error_t LogWrite.append(void* buf, storage_len_t len)**
   Data present at *buf* of length *len* is appended to log

Parameter:	buf	Write data pointer
	len	Number of bytes to be appneded
Return:	Status	Success/Error code

5. **storage_cookie_t LogWrite.currentOffset()**
   No action is performed.
6. **error_t LogWrite.sync()**
   No action is performed.
7. **error_t LogWrite.erase()**
   No action is performed.
8. **void BlockRead.readDone(storage_addr_t addr, void* buf, storage_len_t len,error_t error)**
   No action is performed.
9. **BlockRead.computeCrcDone(storage_addr_t addr, storage_len_t len,uint16_t crc, error_t error)**
   No action is performed.
10. **task void readDone()**
    We are signaling event as readDone.
11. **getBlocks(storage_len_t len)**
    This function calculates the total number of blocks to be written /read for data of size *len*
12. **error_t LogRead.read(void* buf, storage_len_t len)**
    read the data of length *len* in read buffer *buf*
13. **storage_cookie_t LogRead.currentOffset()**
    No action is performed.

14.error_t LogRead.seek(storage_cookie_t offsetloc)

No action is performed.

15.storage_len_t LogRead.getSize()

No action is performed.

Note: Log Abstraction is partially implemented. Commands sync, erase, seeks are not implemented.

5. Evaluation

The time duration for various operations on SD Card are summarised in below table

Operations	Time/Block (in ms)
Read Block	96
Write Block	199
Erase	15

The above result can be interpreted as -
To write single block on SD card takes 199 ms.

Note: The time required for Write operation is more because before writing the data, we are handling the case wherein offset address where data to be written is not in multiple of block size. In that case we are reading the data of the block where current data is to be written. And along with the new data existing data is overwritten again.

6. Conclusion

This project has accomplished its goal to extend the Tmote Sky memory by providing support SD card. Persistent storage limit of Tmote is limited by the capacity of SD card. Currently, it varies from 512k to 4GB.

7. References

- SanDisk Secure Digital (SD) Card Product Manual, Rev. 1.9 © 2003 SANDISK CORPORATION
- Physical Layer Simplified Specification Version 2.00
- Tmote Sky : Datasheet
- Shimmer overview present at http://docs.tinyos.net/index.php/Intel_SHIMMER
- SHIMMER Hardware Guide