Massive Storage in a Miniature (Embedded) Package

Anthony Massa, MNW Tech

Abstract. Data, data is everywhere…and we want ways to get it, store it, transmit it, and mine it. Many different options exist from solid state drives to embedded data recorders. This article takes a look at the fundamentals of an embedded data storage system, the thoughts behind the design decisions and different features to incorporate in an embedded data storage system.

Introduction

The article delves into what data to store, where to store it, and how to get it off the device. I will also get into the considerations of power management in order to extend the battery life of the storage system and ruggedizing the system so that the data is reliable, as well as the different file system options available.

Data Storage System Design

The design of a data storage system has several basic features to consider…and some not so basic. The first question is: What data is going to be stored? Followed by: Where will the data be stored?

In our initial application, the data to store is information that determines what an object is doing as it floated on the ocean's surface. To accomplish this, an inertial measurement unit (IMU) is used to generate the data. It is important to consider other input data sources, typically from various sensors. In this case, having an accommodating platform that can receive analog (via ADC) or digital data is important. For digital inputs, many sensors are designed with widely used serial interfaces such as serial peripheral interface (SPI) or inter-integrated circuit (I2C). Including these serial interfaces in the data storage system design is important to allow easier integration with a wide variety of sensors.

In order to determine where to record the data, the storage speed needs to be determined as well as the capacity for the data. Along with these considerations, the facilities supported by our microcontroller (MCU) are important to consider.

In this case, the MCU included a secure digital host controller (SDHC) interface for SD/MMC/µSD cards. Therefore, the first design uses an SD memory card for data storage. This offers the ability to have removable media for data extraction, varying sizes (including large storage space) of storage capacity by simply changing the card, and a compact size.

In a rugged environment using removable media may not be a good idea because of the connector as a potential issue where the card becomes dislodged. There is also the extra cost associated with a connector if price is the key concern.

In rugged environments, soldered flash might be a better alternative such as serial or NAND flash. Considerations of storage capacity and speed still need to be understood in order to ensure the system specifications are met by the hardware. There is also the consideration of how the data is stored to these "hardwired" alternatives where a file system may not accommodate soldered in flash. In that case, a custom driver needs to be developed. Other concerns with soldered flash are wear-leveling and bad-block handling.

Retrieving the Data

After determining the data storage issues, next is to design how the data is extracted from the storage system. Several options exist depending on how the data storage system is going to be deployed. If the system is going to be returned to a lab environment for data extraction and a removable memory card (such as the previously mentioned microSD card is used), most PCs come with interfaces that can accept these types of memory cards directly. This type of interface provides a level of data integrity by reducing the chances of corrupted data during the transfer.

Another option if the system is returned to a lab environment, a serial port such as a UART could be used to send the data for post processing. Easily interfaces to PCs (albeit typically with a USB-to-serial cable) and can include common serial protocols X, Y, or ZMODEM to add a layer of data integrity with packet checksums.

If the system cannot be returned, but has network access, such as an Ethernet connection, then the data can be retrieved using a standard network protocol. This requires the data storage system to incorporate a network stack in order to be able to communicate over the network. A protocol such as the file transfer protocol (FTP) can be used to allow the data to be accessed remotely.

If a wired connection is not possible, then a wireless connection can be designed to get the data off the unit. Wireless connectivity offers a lot of flexibility as far as data removal goes. Several off-the-shelf modules exist that provide the commonly supported (Wi-Fi, Zigbee, Bluetooth) and low power protocols in use today. There are some transmission range concerns depending on how remote the unit is deployed but these types of wireless protocols offer low power capabilities for remote data extraction. Other wireless options are also available including various modem modules such as cellular or satellite. There are also power as well as added cost concerns not only for the modules themselves but also the cost to use the satellite or cellular network to retrieve the data.

Power Management

A key concern for remote systems is power management. If a system is battery powered and needs to last for months or even years without intervention, one of the biggest design considerations is power management. Many MCUs nowadays have several different sleep modes which conserve power. The power management module can then be designed so that the system only wakes up when it is time to collect data samples

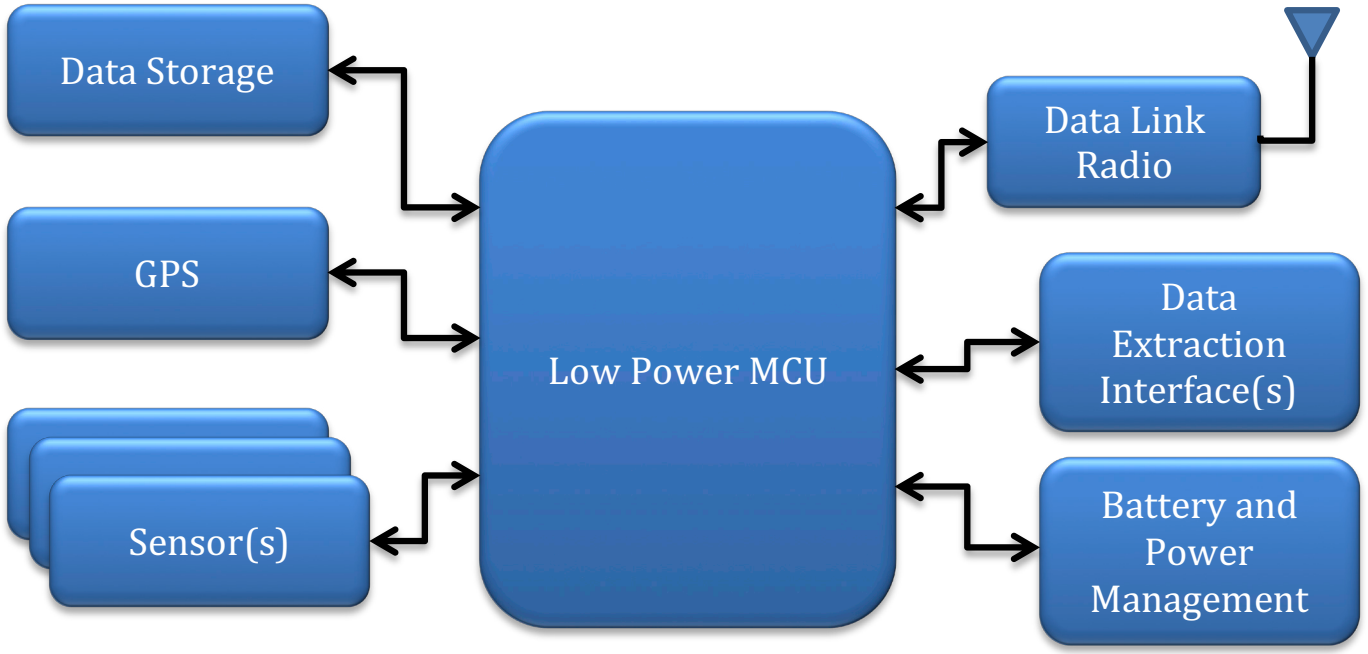

Figure 1. Data storage system block diagram.

from the sensors or time to transmit the data. If a time base is needed to synchronize when data is sampled or transmitted, a GPS module can be included in the system to synchronize time across several units in the network as well as providing location information. A real-time clock (RTC, which can be a module in many MCUs out on the market today or as a standalone chip) can also be used, but needs to be programmed initially with the correct date and time value.

Figure 2 shows the MNW Tech SD card-based data storage system. This system provides a serial interface to extract the data from the system, as well as the capability to remove the SD card directly.

Data Security

In particular applications it may be necessary to secure the data stored to a device for example in medical applications where patient information must be secured. Data security can be a major concern in systems where the storage device is removable and can be tampered with by unknown sources. In these cases the data is encrypted before storing it to the memory device. Various security algorithms exist that provide the necessary security including many open source solutions. In many cases, the microcontroller can assist in the data security by providing a cryptographic acceleration unit. For example, some versions of the Freescale Kinetis ARM-based microcontroller include such a cryptographic accelerator that assists in many different popular cryptographic algorithms including DES, 3DES, AES, MD5, SHA-1, and SHA-256.

System Configuration

In order to develop a general design and product for several different applications, runtime system configuration is needed. Therefore, the data storage system is able to adapt to the specific needs of a particular customer by allowing the customer to select various configuration parameters such as how often the data is stored or data sampling rate, what data is overwritten when the limit is reached, when a new data file is started, and even what data is stored in the file.

There are several different options for runtime system configuration such as a loadable text file which can be contained on the memory card, if present, or downloaded serially. The user can then customize the various data storage parameters by hand-editing the system configuration file. Once deployed, it can be difficult to modify the configuration file unless remote access, such as via wireless download, is available on the system.

Another method which is more elegant is a web interface providing the configuration information. In this case, a network stack and web server need to be incorporated into the data storage system and the customized web page interface needs to be developed. Using a web interface is common on many products today for example on nearly all routers. The user is then able to select the various configuration parameters from drop down lists and radio buttons to customize the operation of the data storage system.

File System

To use a file system, or not…that is a question. For certain,

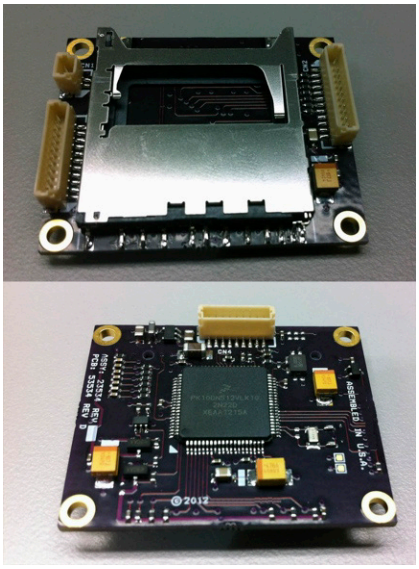

Figure 2. MNW Tech SD card-based data storage system.

more basic data storage applications, a file system can be unnecessary and instead directly writing to the storage module can be used. For example, if a serial flash is used to store the data gathered, only the serial flash driver need be developed and the data can be written directly to the device at the specified addresses. This eliminates layers of complexity with a file system and file management.

In other applications, a file system provides the additional features and capabilities that are necessary. A file system provides help with numerous features including organizing data through the usage of files, timestamp on file data, as well as wear-leveling and bad block management.

Most memory devices have a finite number of program-erase cycles that they are specified to meet. In order to extend the life of the storage device, different memory sectors are used to store the data, essentially mapping the data to different sectors rather than repeatedly storing data to the same sector location in the device. This technique is called wear-leveling. Another technique called bad block management verifies data written to the storage device and keeps track of sectors that are damaged and therefore unusable to avoid data loss.

Several open source file systems are available for use. One popular file system used is FatFS (http://elm-chan.org/fsw/ff/00index_e.html). FatFS is ideal for many embedded systems because it has a small footprint, is written it C making the source code platform independent, and has a very modular design allowing different configuration options. Many of the basic API calls are provided such as open, close, read, write, and mount.

Conclusion

Many different applications exist that require data storage systems from collecting the body's physiological characteristics data to monitoring the temperature of a remote location to tracking the path of an unmanned vehicle. This article has presented a number of key design considerations for an embedded data storage system along with possible solutions based on the type of application in which the data storage system is deployed.

ABOUT THE AUTHOR

Anthony Massa is the Director of Software Engineering at MNW Tech <https://www.mnwtech.com> in San Diego, CA. Anthony has over 20 years of experience in all aspects of engineering, focusing on embedded systems and has worked on several successful products. He has taught courses on embedded software development and written extensively on the subject including the books Programming Embedded Systems: with C and GNU Development Tools (O'Reilly) and Embedded Software Development with eCos (Prentice Hall PTR) as well as several articles.

Phone:619-252-7010
E-mail: amassa@san.rr.com