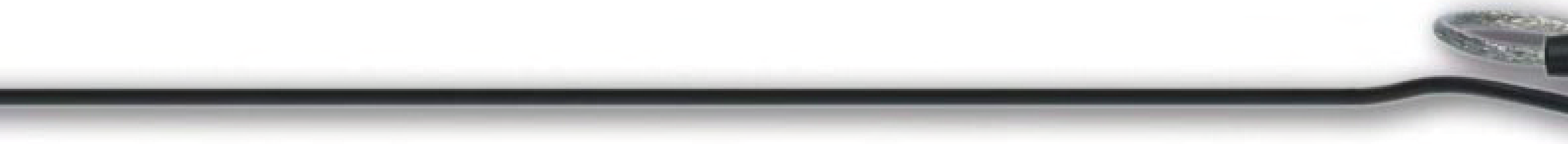
USB Stick with ARM

Gigabyte Flash drive for microcontroller



Ursula Engelmann-Schrader and Jürgen Engelmann

This neat stand-alone memory stick can store or transfer data from a microcontroller system in the field to a PC using its built-in USB and RS232 ports. Add to that an LCD and the simple to use datalogging mode is just the icing on the cake!

Some electronic systems are of necessity sited at remote locations where they may be collecting data from natural events such as wind strength or solar powered installations where measurements are performed and stored on site in the memory of a microcontroller system. Occasionally the data requires transferring to a PC where they can be fully evaluated and archived. Without the luxury of a radio link or telephone line to transfer the data it would seem a good solution to use a versatile memory stick which can both plug into a serial port of a microcontroller system and a USB port on a PC, well there is no need to look any further, that's exactly the function of the device described here!

A Janus memory

When the design for this 'microcontroller USB stick' was first sketched out on the back of an envelope it was decided not to use hard-wired memory chips for data storage but instead provide a slot for an MMC or SD-Card (FAT16 format). The flexibility of this approach means that your choice of memory for the card is not confined to just one supplier and allows you to take advantage of the falling cost and increased (gigabyte) capacity of the most recent memory cards.

Just like the Roman god Janus, the USB flash drive presents two faces to the world:

On the USB side it looks like a Windows and Linux compatible USB memory stick. Once plugged into a PC all files stored in the flash card can be viewed or edited on the screen. The PC user is free to begin work interpreting the data stored in the card previously by a microcontroller system (which may also still be attached).

The second face of the drive allows data stored in the memory by a PC to be read by a microcontroller system. The on-board ARM processor together with its firmware enables an external microcontroller system connected to the RS232 port to have simple read/write access to the flash memory. An illustration at the end of the article shows the USB Flashdrive connected to the ATmega controller board (050176-71) which was featured in the May 2006 edition of *Elektor Electronics.*

Access to the memory card by the microcontroller is performed by a driver which interprets a set of predefined instructions. The instructions are simple commands such as FileOpen, FileRead, FileWrite, FileClose, etc. The driver is contained in the software and not only controls access to the memory but also interprets commands for control of peripherals such as the LCD.

The C source code for the drivers together with a few examples of code suitable for 8051 compatible controllers are included in the free software download available at the *Elektor Electronics* website [1]. A number of additional Pascal files are also included for information — these were written for an earlier application of the unit. From the DOS point of view the memory card is treated just like an external drive connected to the serial interface. A Windows program ('Testsuite') is also included along with the other files and can be used to test the PCB. The flash memory card is also extremely easy to use in data logging mode (see text box).

The ARM7 Controller

The heart of the circuit shown in **Figure 1** is the ARM7 Controller AT91SAM7S64 (IC1). It is a 32-bit controller with a RISC core. The CPU instruction set includes instructions which allow switching between 16 or 32-bit instructions to enable optimum use of the processor for each application. The processor has a 64 kB Flash memory and 16 kB RAM. An on-board PLL multiplies the 12 MHz crystal frequency up to 48 MHz used by the processor.

The ARM7 controller is equipped with a

and RS232

s

whole range of additional features including a complete USB port integrated onboard so it is only necessary to connect to the D– and D+ signals on the USB port. Resistors R9 and R10 form a potential divider network which the controller uses to determine if a device is plugged into a USB port. A brief overview of the ARM7 microcontroller features are shown in the text box.
Those of you who would like to delve deeper into the workings of the ARM7 controller can use either a GNU compiler or any of the other available compilers. ARM technology was discussed in several articles in *Elektor Electronics*.

Main Features

Memory media:	MMC or SD card up to 2 GB
File system:	FAT16 (A maximum of four files may be open at the same time)
Interfaces:	1 x USB (2.0 and 1.1), 2 x RS232
USB Data rate:	12 Mbit/s
RS232 Data rate:	9600 bit/s to 230 kbit/s
Power supply:	5 V derived from the USB connector or external mains adapter.
Current consumption:	50 mA (approx).
Options:	LCD connector, TTL level serial interface.
Dimensions (approx):	41 mm x 77 mm x 18 mm (including connectors and memory card)

The interfaces

The USB interface (ST2) has a raw bit rate of 12 Mbit/s and can be used with either a USB 1.1 or USB 2.0 compliant USB port on a PC. High-speed operation is indicated by the PC via R11 on D+.
The RS232 interface is quite conventional, using a MAX232 (IC11) to perform the necessary level shifting for signals on the 9-way Sub-D connector CON2 providing ±12 V nominal on the RS232 side and TTL levels on the ARM controller side.
A second serial interface is available on JP2. The RS232 interface can handle data rates ranging from 9.6 up to 230 kbit/s. On reset the communications rate is set to 9600 bit/s (Default). The serial receive and transmit signals are also available on pins 2 and 3 of JP1 at TTL levels. When this option is used it is necessary to remove solder jumper SJ1 to prevent any possible contention in the signal levels produced by the received data on CON2.
A 21-way pin header connector, JP3, provides connection for a standard LCD display using the HD44780 or compatible controller. Support is provided for displays using one, two or four lines of text up to 16 or 20 characters long. Power is derived directly from the USB bus. If an LCD display is fitted it will be necessary to fit the SMD preset R1 to allow control of the contrast. The display uses 11 pins on JP3; eight are for data, one for Register Select (RS), one for Read/Write (R/W) and one for Enable (E). With no display fitted, PA7 to PA20 can be used as general I/O pins.
The ARM controller also has an SPI interface for connection of a Flash memory card; the following paragraph looks at this interface in more depth.

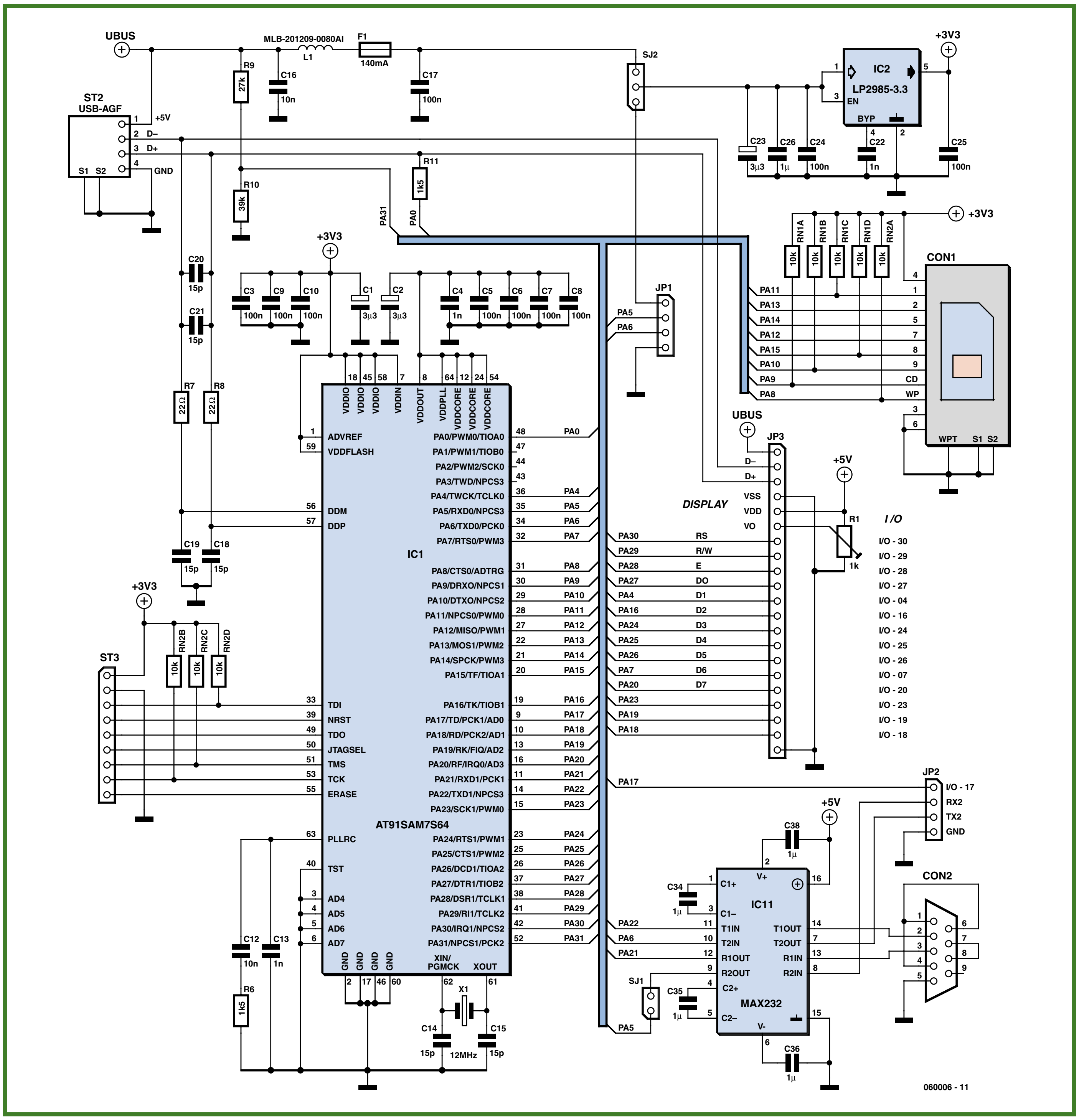


Figure 1. The core of the unit is an ARM7 controller which contains an integrated USB interface. Its firmware allows a microcontroller connected to the RS232 port read/write access to data in the flash memory.

The memory card

The Flash card connector on the PCB (CON1) can accommodate both MMC and SD cards despite their differing thickness. Both types of card are used for many other commercial applications such as digital cameras, palm tops etc. they are competitively priced and offer good memory capacity. The circuit supports cards with a capacity up to 2 GB.

Flash memory does have a limited number of erase cycles (which obviously limits the write cycles also). This 'endurance' figure is largely dependant on the type of technology used by the card (NOR or NAND Flash). Standard MMC and SD cards (NOR Flash) quote an endurance figure of 100,000 write cycles while industrial grade variants boast 400,000 cycles with an extended operating temperature range.

When a card is inserted into the connector the controller determines the status of the write protection switch. The card memory space is organised in sectors of 512 bytes, this is also the size of memory which can be read or written to in one step. The file system used is the industry standard FAT16 format so the card can also be re-

ARM7 Controller

Main features of the AT91SAM7S64

- 32-bit RISC Architecture
- 64 kB Flash, 16 kB SRAM
- Reset Controller, Brownout Detector
- 32 I/O Pins (Pullups)
- 2 UART, SPI (8 to 16-bit), TWI
- USB 2.0 Full Speed (12 Mbit/s), DMA controller
- several timers, e.g. 16-bit Timer/Counter, PWM Controller
- 8 channel 10-bit A/D converter
- JTAG, connection for emulation and debug
- 5 V tolerant I/Os, 4 I/O pins providing up to 16 mA

moved and used in other equipment such as card readers etc.
Communication with the memory cards can be performed using either MMC or SD mode and both types of card support the SPI interface mode. The ARM 7 controller uses SPI mode and needs just four connections to the card: MOSI (Master Out Slave IN), MISO (Master In Slave Out), SCK (Serial Clock), ~~SS~~ (Slave Select (active low)). The card communicates at a raw data rate of 12 Mbit/s. It is important

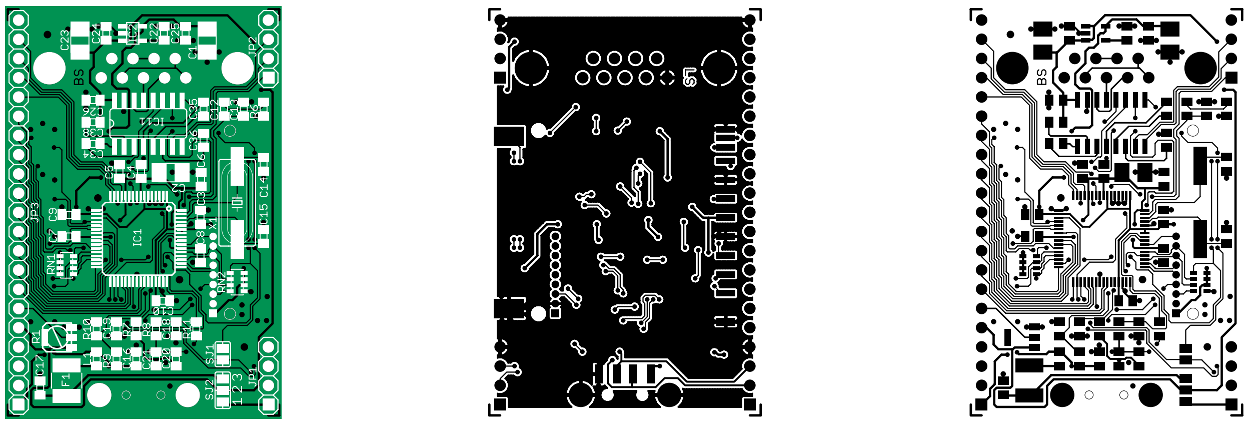


Figure 2. **SMD component placement on the PCB. This board is available ready populated and tested.**

COMPONENTS LIST

Resistors
R1 = 1kΩ potentiometer, SMD, 1% (optional, see text)
RN1,RN2 = 4-way 10kΩ resistor array, 5%
R6, R11 = 1kΩ5 (SMD 0805, 1%)
R7,R8 = 22Ω (SMD 0805, 1%)
R9 = 27kΩ (SMD 0805, 1%)
R10 = 39kΩ (SMD 0805, 1%)

Capacitors
C1,C2,C23 = 3µF3 (SMD 3528, tantalum, 20%)
C3,C5,C6,C7,C8,C9,C10,C24,C25 = 100nF (SMD 0805, 10%)
C4,C13,C17,C22 = 1nF (NP0, SMD 0805, 5%)
C12,C16 = 10nF (SMD 0805, 10%)
C14,C15,C18,C19,C20,C21 = 15pF (SMD 0805, 5%)
C26,C34,C35,C36,C38 = 1µF (SMD 0805, 10%)

Semiconductors
IC1 = AT91SAM7S64 (programmed, order code **060006-41**)
IC2 = LP2985A-33DBVT (SOT23, Texas Instruments)
IC11 = MAX232 (SO16, Maxim)

Miscellaneous
X1 = 12MHz quartz crystal (SM49)
F1 = 140mA Polyswitch
L1 = choke, MLB-201209-0080AI (Kitagawa)
CON1 = card holder for SD/MMC cards
CON2 = 9-way sub-D socket (female), angled pins, PCB mount
ST1 = USB-A plug, PCB mount, e.g. Assmann A-USB-A-SMT (Reichelt USB AGF)
JP1,JP2 = 4-way SIL pinheader, lead pitch 0.1 in
JP3 = 21-way pinheader, lead pitch 0.1 in

PCB, order code **060006-1** (unpopulated) or **060006-91** (assembled and tested, see Elekor SHOP on in this magazine or on website)
Project software, file **060006-81.zip**, free download from www.elektor.com (month of publication)

Table 1. Overview of commands

USB commands:

Mount	UnMount	CardAvailable	CardNotAvailable
Connect	DisConnect	WriteDisable	WriteEnable
Ping			

File related commands:

FileOpen	FileClose	FileRead	FileWrite
FileSeek	FileTell	Dir	ChangeDir
CreateDir	RenameFile	RemoveFile	QuickFormat

Commands to the peripherals:

InitDisplay	ClearDisplay	DisplayOff	DisplayWrite
Baudrate	SetDirection	SetOutput	GetInput
SetPullup	GetCardStatus	GetFirmwareVersion	

Example

MOUNT command to the controller:

```
41h  04h  00h  45h
 |    |    |    |
 |    |    |    |__________ Checksum 41h XOR 04h XOR 00h gives 45h.
 |    |    |_______________ High-Byte of the sum of bytes in the command sequence.
 |    |____________________ Low-Byte of the sum of bytes in the command sequence.
 |_________________________ The command byte.
```

Controller response:

```
41h  01h  40h
 |    |    |
 |    |    |_______________ Checksum 41h XOR 01h gives 40h.
 |    |____________________ The value 1 indicates that the MOUNT request was successful.
 |_________________________ Echo the command byte.
```

to ensure that this card is not inserted or removed during operation!

Supplying the power

The complete unit draws its 5 V supply from the from either the USB connector or from an external mains adaptor via connector JP1.

A solder jumper (SJ2) is used to define which power supply option is used. Voltage regulator (IC2) produces the 3.3 V required by the microcontroller and the majority of the remaining circuitry. In normal operation just two pins of SJ2 will be bridged to define the source of power for the unit. It is not recommended but it is also possible to bridge all three pins together so that power can be supplied from either source but with both bridged you must make certain that power is not supplied to the unit from both the adapter **and** USB port at the same time!

Coil L1 and capacitor C17 form a low pass filter to remove any interference on the USB power supply line. The capacitors fitted across the USB data lines and also around the voltage regulator perform the same function. A 150 mA PolySwitch fuse (F1) prevents the circuit from drawing too much current if a fault occurs.

Printed circuit board

The PCB (**Figure 2**) shows that all the SMD components are mounted on one side of the board while the other side is used to mount the connectors (**Figure 3**).

The largest devices on the component side (**Figure 4**) are the AT91SAM7S64 microcontroller, the MAX232 and the 12-MHz quartz crystal.

Soldering SMD components can be tricky (and expensive) if you are not experienced in this procedure, both the 64-pin LQFP package used for the controller and the 1206 outline resistor network require particular care. For those of you who are less confident, a fully populated and tested PCB can be ordered from the Elektor SHOP or alternatively the unpopulated PCB is also available for the more adventurous.

Software

All the software for this project is contained in an archive file no.060006-81.zip at www.elektor-electronics.com. The text box entitled 'Software files' gives an overview of all the program files and documentation. It contains mainly driver routines and example files for a microcontroller system attached to the RS232 port of the flash memory.

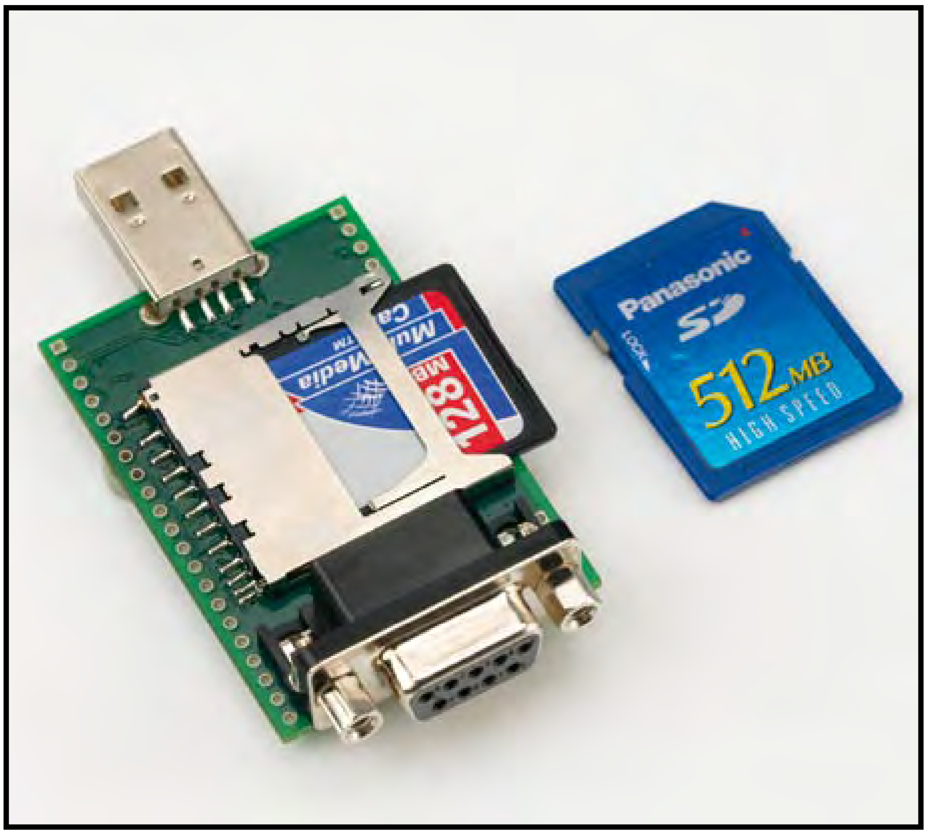


Figure 3. The USB socket, RS232 connector and MMC/SD card holder are the only components mounted on one side of the board.

Software for the ARM controller (device available pre-programmed) takes care of the file system and executes commands sent over from the USB or RS232 interface. Commands originating from the operating system of the PC are received over the USB port and 'see' the flash memory as an additional drive. Commands from an external microcontroller system are received and responses are sent over the RS232 interface. The overview in **Table 1** shows that the commands can be divided into three groups:

- USB commands

All the instructions dealing with communication between the USB port, RS232 port, the PC and ARM7 controller.

- File commands

These allow files to be opened and closed, renamed and for the attached MMC/SD memory card to be formatted.

- Peripheral commands

Commands to control the display, the free I/O pins and communications speed.

All communications conform to the same basic structure; a command is sent to the controller followed by a reply from the controller. Each message to the controller contains a command field, a parameter field (if the command has any parameters associated with it) and a check sum. The main difference is in the number of parameters sent.
Instructions to the controller therefore have the following format:

[Command byte - [0 to 4 parameters] - Checksum]

Byte 1:	command bye
Byte 2:	LowByte of total length of command
Byte 3:	HighByte of total length of command
Byte 4:	LowByte of length of 1st parameter
Byte 5:	HighByte of length of 1st parameter
Byte 6:	data of 1st parameter (1-512 bytes)

[…] up to three more parameters

Byte *n*:	8-bit XOR checksum across full command

The controller reply has a similar format, typically consisting of a three-byte sequence. The first byte is a repeat (echo) of the command byte sent to the controller; the second byte is a token indicating either a successful or unsuccessful outcome of the command while the third byte

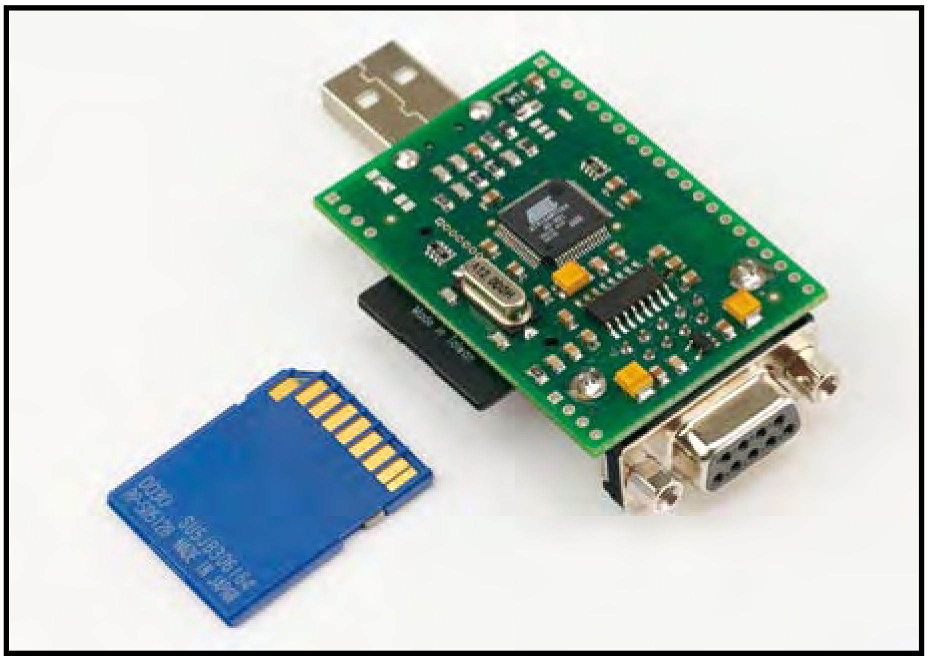


Figure 4. The AT91SAM7S64, MAX232 and 12 MHz crystal are the largest components on the underside of the PCB.

NAVTEX

NAVTEX meteorological radio broadcasts have proved useful to sailors for many years and also served as the source of inspiration for this project. The international NAVTEX (Navigational Warnings by Telex) network sends out information on sea conditions and storm warnings as situations develop. The station transmits internationally on 518 KHz or nationally (in the local language) on 490 KHz [2].

These text broadcasts can be picked up by dedicated receivers (see photo) some of them use a thermal printer to provide a hard copy of the messages while others use a flash memory to store the incoming message where it can be read simultaneously (or later) by a PC via a USB port.

Expanding on this concept the author produced this design which can store data from a microcontroller system and transfer it to a PC using an RS232 port, a USB port and a low cost, high capacity MMC/SD card as the storage medium.

Software Files

060006-81.zip
(Download from www.elektor.com
or order the CD):

Source text/drivers for C
File Stickdrv:
double.cpp Commands to the stick
double.h Header file
serial.cpp serial interface
serial.h Header file
File Stick51:
C examples for 8051 compatible devices

Source text and drivers for DOS-like commands:
File Stickdos:
COPYS.PAS Copy command
DELS.PAS Delete command
DIRS.PAS Dir command
TYPES.PAS Type command
MISC.PAS odds and ends
PARAM.PAS parameter handling of the above instructions
RS232.PAS serial interface
STICK.PAS commands to the stick

Windows Program (incl. source code)
File Sticktest with Testsuite for testing purposes

Documentation:
File Stickdok: Additional information describing the commands

is a checksum. When the command requests data this is transmitted before the sequence.
The controller response therefore has the following format:

[Command byte - success/failure token - Checksum]

Byte 1: repeat of command
Byte 2: '1' if command successfully completed; else '0'
Byte 3: 8-bit XOR checksum across above two bytes

To better illustrate the communication process the MOUNT command is detailed in Table 1. MOUNT is generally the first command in the program and does not have any parameters associated with it.
This command instructs the controller to load parts of the filing system (FAT, index) of the MMC/SD card into its RAM. It is necessary to retrieve this information from the memory card before some of the other commands can be used.
When all the bytes in the command and reply sequence are XORed together the result should equal 00h this indicates (with a high level of probability) that the communication sequence was not corrupted.
To check whether the command was carried out successfully it is necessary to examine the middle byte of the received sequence, if the byte is 01h it indicates that it was successful whereas a 00h indicates that it was not possible to carry out the command.
A more detailed description of each command is included amongst the files that can be downloaded [1] for this article (060006-81.zip).

(060006-I)

Web links

[1] www.elektor.com
(month of publication)

[2] http://en.wikipedia.org/wiki/Navtex

Data logger

The USB-Flash memory can also be used as a data logger. Firmware in the ARM controller switches into data logging mode when it detects a jumper connecting pins 17 and 18 of the pin header strip JP3. Three additional components are required for this mode of operation:

- A pushbutton wired between pin 4 (GND) and Pin 7 (PA30),
- A 750 Ω resistor between Pin 8 (PA29) and Pin 12 (PA16),
- A red low-current LED (2 mA) with the anode connected to Pin 12 (PA16) and the cathode to Pin 13 (PA24).

Once the jumper is in place the LED lights continuously to indicate that it has entered data logging mode. When the pushbutton is pressed the message DATEN appears (if data is not available). The LED blinks and the controller waits for data from the RS232 port (9600 bit/s). When data arrives it is written to a file which is given the name 'DATEN001.TXT', if this file already exists it creates another with the name 'DATEN002.TXT' and so on up to 'DATEN999.TXT'.

Once the file has been transferred, another press of the pushbutton closes the file and the LED again lights continuously ready for the next file.

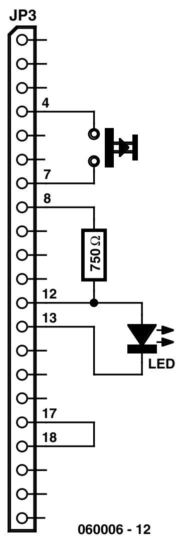


Diagram of the 21-pin header strip JP3 when used in data logging mode. Pin 1 is indicated by a square outline printed on the PCB.