

USB Stick with ARM and RS232
Elektor Electronics November 2006, p. 42.

© Copyright Segment b.v.

(additional information supplied by the author)

1. Pin descriptions for the LCD interface JP3:

REGISTER_SELECT	PA30
READ_WRITE	PA29
ENABLE	PA28
DISPLAY_D0	PA27
DISPLAY_D1	PA4
DISPLAY_D2	PA16
DISPLAY_D3	PA24
DISPLAY_D4	PA25
DISPLAY_D5	PA26
DISPLAY_D6	PA7
DISPLAY_D7	PA20

2. Description of the driver functions

Structure of commands to the ARM controller in the USB-Flashdrive.

Byte 0 instruction

Byte 1 count of entire message length (low byte)

Byte 2 count of entire message length (high byte)

Length of parameter 1 (low byte)

Length of parameter 1 (high byte)

Parameter 1 (1 - 512 Byte)

Length of parameter 2 (low byte)

Length of parameter 2 (high byte)

Parameter 2 (1 - 512 Byte)

Length of parameter 3 (low byte)

Length of parameter 3 (high byte)

Parameter 3 (1 - 512 Byte)

Length of parameter 4 (low byte)

Length of parameter 4 (high byte)

Parameter 4 (1 - 512 Byte)

The last byte (Check sum) is the result of XOR'ing all the message bytes together.

The USB Drive ARM controller replies to the command with a three byte long message:

The first byte of the reply is a repeat of the instruction byte.

The second byte = 1 if the instruction was successfully carried out, if not = 0.

The third byte is the result of XOR'ing the first and second byte together (XOR'ing all three bytes

together will always give 0 unless any bit was corrupted).

General functions used in the commands.

Name: **MsDelay**

Description: Produces a wait period.

Parameter: DelayInMilliseconds: The wait time in milliseconds.

Reply: None

Format: void MsDelay(WORD DelayInMilliseconds) ;

Name: **SendCommand**

Description: Sends an instruction to the USB Flash drive. The instruction can have up to 4 parameters. All unused parameter are omitted.

Parameter: pData: Pointer to the instruction.

Reply: None

Format: void SendCommand(uint8_t *pData) ;

Name: **GetResult**

Description: GetResult is sent to the USB-Flashdrive following a SendCommand and gives the result of the command.

Parameter: TimeOut: time in milliseconds that GetResult waits for an answer from the controller.

Reply: When the check sum is correct AND when the controller has established that the command was successfully executed GetResult will return a '1' otherwise it returns a '0'.

Format: uint8_t GetResult(uint16_t TimeOut) ;

Name: **Mount_**

Description: Parts of the SD / MMC card file system are transferred to controller system. It is necessary to perform this instruction before other file instructions can be used (OPEN, READ, WRITE, CLOSE, etc.)

Parameter: None

Reply: A 1 indicates the instruction was performed successfully, 0 indicates failure.

Format: uint8_t Mount_(void) ;

Name: **UnMount_**

Description: Any files that are open will be closed. Any parts of the file that have been buffered in the controller are rewritten to the card. The card can only be removed (without error) after an UnMount instruction has been performed.

Parameter: None

Format: uint8_t UnMount_(void) ;

Name: **CardAvailable_**

Description: The card can be accessed from the USB Port.

Parameter: None

Format: uint8_t CardAvailable_(void) ;

Name: **CardNotAvailable_**

Description: The card cannot be accessed from the USB Port.

Parameter: None

Format: uint8_t CardNotAvailable_(void) ;

Name: **GetCardStatus_**

Description: Reads the card status.

Parameter: None

Reply: A 1 indicates that the instruction was successfully carried out, 0 indicates failure.

The card status is contained in the called byte
Bit 0: '1' card present, '0' no card present
Bit 1: '1' card write protected, '0' not write protected
Bit 2: '1' Mount has been performed on the card, '0' Mount has not been performed on the card.

Format: uint8_t GetCardStatus_(uint8_t *status) ;

Name: **FileOpen_**

Description: Opens a file to read data or to read and write data or to add data.

Parameter: Handle: A number 0 to 3, connected to the file name. While the file is open instructions such as FileWrite_ referred to it using its handle.

pName: Pointer to the file name. The file name can be up to 8 characters long with a 3 character extension.

The characters A to Z, 0 to 9, #, &, %, (,), - and @ are permissible.

FileOpenMode:

FILEOPENMODE_READONLY – An existing file is opened exclusively for reading.

FILEOPENMODE_WRITE – The file is opened for reading and writing **Warning:** if the file is available the previous contents are overwritten.

FILEOPENMODE_APPEND – An existing file is opened and the data pointer is set to the file end position. The file contents can be read or data can be added to the file.

TimeAndDate: Date and time (see Appendix).

Format: uint8_t FileOpen_(uint8_t Handle, uint8_t *pName, uint8_t FileOpenMode, TimeAndDate_t TimeAndDate) ;

Name: **FileClose_**

Description: Closes the file identified by handle.

Parameter: Handle: The file handle.

TimeAndDate: Date and time.

Format: uint8_t FileClose_(uint8_t Handle, TimeAndDate_t TimeAndDate) ;

Name: **FileRead_**

Description: Reads data from the file with the handle 'Handle' data in the array pointed to by the pointer 'pData'. There are 'Count' lines of length 'size' read.

Parameter: Handle: The file handle

pData: The data array pointer

Count: the number of lines to be read.

Size: The line length.

Reply: The number of bytes returned (Count * size).

Format: uint16_t FileRead_(uint8_t Handle, uint8_t *pData, uint16_t Count, uint16_t Size) ;

Name: **FileWrite_**

Description: Data in the array pointed to by 'pData' is written to the file defined by 'Handle'. The number of lines is given by 'Count' each with a length of 'Size'.

Parameter: Handle: The file handle

pData: The data array pointer

Count: The number of lines to be written.

Size: The line length.

Format: uint16_t FileWrite_(uint8_t Handle, uint8_t *pData, uint16_t Count, uint16_t Size) ;

Name: **FileSeek_**

Description: The internal pointer indicating the next position to be read from or written to is set to the value 'Position'.

Parameter: Position: The new position for the internal pointer.

whence: The new position is taken from the start of the data (WHENCE_SEEKSET), from the current position (WHENCE_SEEKCUR) or from the end of data (WHENCE_SEEKEND).

Format: uint8_t FileSeek_(uint8_t Handle, uint16_t Position, uint8_t Whence) ;

Name: **Dir_**

Description: Displays the directory contents (files and further directories).

Parameter: pData: Pointer to the array used by the directory.

Format: uint16_t Dir_(uint8_t *pData) ;

Name: **ChangeDirectory_**

Description: Change the directory.

Parameter: pName: Pointer to the string containing the name of the new directory.

Format: uint8_t ChangeDirectory_(uint8_t *pName) ;

Name: **CreateDirectory_**

Description: Creates a new directory.

Parameter: pName: Pointer to name of the new directory.

TimeAndDate: Time and date.

Format: uint8_t CreateDirectory_(uint8_t *pName, TimeAndDate_t TimeAndDate) ;

Name: **RenameFile_**

Description: renames a file.

Parameter: pOldName: pointer to the old name

pNewName: pointer to the new name

TimeAndDate: date and time

Format: uint8_t RenameFile_(uint8_t *pOldName,uint8_t *pNewName,TimeAndDate_t TimeAndDate) ;

Name: **RemoveFile_**

Description: Deletes a file.

Parameter: pName: pointer to the name of the file to be removed.

Format: uint8_t RemoveFile_(uint8_t *pName) ;

Name: **QuickFormat_**

Description: Performs a quick format of the SD / MMC card.

Parameter: pName: pointer to the name of the SD / MMC-Card

Format: uint8_t QuickFormat_(uint8_t *pName) ;

Name: **InitDisplay_**

Description: Initialises the display attached to JP3.

Parameter: None

Format: uint8_t InitDisplay_(void) ;

Name: **ClearDisplay_**

Description: Clears the display.

Parameter: None

Format: uint8_t ClearDisplay_(void) ;

Name: **DisplayOff_**

Description: Switches the display on or off.

Parameter: OnOff:

Format: uint8_t DisplayOnOff_(uint8_t OnOff) ;

Name: **DisplayWrite_**

Description: Writes text to the display.

Parameter: CursorPosition: Position, from where the text will be written.

Count: Length of the text.

pText: pointer to the text.

Format: uint8_t DisplayWrite_(uint8_t CursorPosition, uint8_t Count, uint8_t *pText) ;

Name: **Connect_**

Description: Establishes a connection with the PC.

Parameter: None

Format: uint8_t Connect_(void) ;

Name: **Disconnect_**

Description: Disconnects the PC.

Parameter: None

Format: uint8_t Disconnect_(void) ;

Name: **WriteDisable_**

Description: Turns on write protection.

Parameter: None

Format: uint8_t WriteDisable_(void) ;

Name: **WriteEnable_**

Description: Turns off write protection.

Parameter: None

Format: uint8_t WriteEnable_(void) ;

Name: **Ping_**

Description: Tests the connection.

Parameter: None

Reply: A 1 indicates a successful test, 0 indicates test failed.

Format: uint8_t Ping_(void) ;

Name: **Baudrate_**

Description: Defines the communication speed of the serial interface. At switch on the speed is set to 9600 Bit/s (1 stop bit, no parity). Note that when the speed is changed the reply is sent at the old baud rate.

Parameter: BaudRate: The new baud rate divided by 100 (e.g. 1152 equals 115200 Bit/s). The maximum speed is 230400 Bits/s.

Reply: A '1' indicates that the speed change was successful, '0' indicates failure.

Format: uint8_t Baudrate_(uint32_t BaudRate) ;

Name: **SetDirection_**

Description: Define an I/O pin as input (1) or output (0).

Parameter: Direction: Pin number (Pin7 = Bit0, Pin8 = Bit1 up to Pin20 = Bit13).

Format: uint8_t SetDirection_(uint16_t Direction) ;

Name: **SetPullUp_**

Description: Enable internal pullup resistor for an input defined pin (pins 7 to 20 on JP3 (see circuit diagram)).

Parameter: PullUp: Pin number (Pin7 = Bit0, Pin8 = Bit1 up to Pin20 = Bit13).

Format: uint8_t SetPullUp_(uint16_t PullUp) ;

Name: **SetOutput_**

Description: Sets/resets the output pins Pin7 - Pin20 on JP3 (see circuit diagram). The pin must have been defined as an output with setDirection instruction.

Parameter: Output: Pin number (Pin7 = Bit0, Pin8 = Bit1 up to Pin20 = Bit13).

Format: uint8_t SetOutput_(uint16_t Output) ;

Name: **GetInput_**

Description: Read the status of Pin7 - Pin20.

Parameter: Data: The value of the pins (Pin7 = Bit0, Pin8 = Bit1 etc. up to Pin20 = Bit13).

Format: uint8_t GetInput_(uint16_t *Data) ;

Name: **GetFirmwareVersion_**

Description: Reads the three byte long firmware version.
(Major version – minor version – patch)

Parameter: Data: A pointer to the three byte long array which stores the firmware version.

Reply: A '1' indicates that the firmware version was read successfully, '0' indicates failure

Format: uint8_t GetFirmwareVersion_(uint8_t *Data) ;

3. Definitions and Data types

```
// FFS instructions
#define FORMAT                0x01
#define OPEN                  0x02
#define CLOSE                 0x03
#define CREATE_DIRECTORY     0x04
#define CHANGE_DIRECTORY     0x05
#define READ                  0x06
#define WRITE                 0x07
#define SEEK                  0x08
#define RENAME               0x09
#define REMOVE               0x0A
#define DIR                   0x0B
#define TELL                  0x0C

// bit masks FFS
#define READ_ONLY_FLAG       (1 << 0)
#define HIDDEN_FLAG         (1 << 1)
#define SYSTEM_FLAG         (1 << 2)
#define VOLUME_NAME_FLAG    (1 << 3)
#define DIRECTORY_FLAG      (1 << 4)
#define ACHIEVE_FLAG        (1 << 5)
#define LONG_FILENAME_ATTRIBUTE (READ_ONLY_FLAG | HIDDEN_FLAG |
SYSTEM_FLAG | VOLUME_NAME_FLAG)

// USB instructions
#define WRITE_ENABLE         0x21
#define WRITE_DISABLE       0x22
#define CONNECT              0x23
#define DISCONNECT          0x24
#define CARD_AVAILABLE      0x25
#define CARD_NOT_AVAILABLE  0x26

// MMC
#define MOUNT                0x41
#define UNMOUNT              0x42
#define GET_CARD_STATUS      0x43

// bit masks MMC
#define CARD_INSERTED        (1 << 0)
#define CARD_WRITE_PROTECT  (1 << 1)
#define CARD_MOUNTED        (1 << 2)
```

```

// DISPLAY
#define INIT_DISPLAY          0x61
#define CLEAR_DISPLAY        0x62
// #define SHOW_FILE          0x63
#define DISPLAY_WRITE        0x64
#define DISPLAY_ONOFF        0x65

// Cursor position on supported display types
#define CURSOR_1X16_LINE1_ROW1 0x00
#define CURSOR_2X16_LINE1_ROW1 0x00
#define CURSOR_2X16_LINE2_ROW1 0x40
#define CURSOR_4X20_LINE1_ROW1 0x00
#define CURSOR_4X20_LINE2_ROW1 0x40
#define CURSOR_4X20_LINE3_ROW1 0x14
#define CURSOR_4X20_LINE4_ROW1 0x54

// UPDATE
#define UPDATE                0x81

// BAUDRATE
#define BAUDRATE              0xA1
#define PING                   0xA2

// IO
#define IO_SET_DIRECTION      0xC1
#define IO_SET_PULLUP        0xC2
#define IO_GET_INPUT          0xC3
#define IO_SET_OUTPUT         0xC4

#define GET_FIRMWARE_VERSION  0xFF

typedef unsigned char uint8_t;
typedef unsigned short uint16_t;
typedef unsigned int uint32_t;

typedef struct
{
    uint8_t Second;
    uint8_t Hour;
    uint8_t Minute;
    uint8_t Day;
    uint8_t Month;
    uint16_t Year;
}
TimeAndDate_t;

typedef enum
{
    FILEOPENMODE_INVALID=0,
    FILEOPENMODE_READONLY,
    FILEOPENMODE_WRITE,
    FILEOPENMODE_APPEND,
}
FileOpenMode_t;

```

```
typedef enum
{
    WHENCE_SEEKSET=0,
    WHENCE_SEEKCUR,
    WHENCE_SEEKEND
}
Whence_t;
```