

Interface Description of the Par devices

Version 0.5

2005-10-8

Copyright www.true-random.com.
This document is without warrenty.

Contents

Table of Contents	1
1 Introduction	3
I USB and Serial Port	4
2 Communication Parameters	5
3 Command List	6
3.0.1 Remarks	8
4 Detailed Description of the Commands	9
4.0.2 Remarks	9
4.1 Command 0x10	9
4.2 Command 0x11	10
4.3 Command 0x12	10
4.4 Command 0x13	10
4.5 Command 0x14	11
4.6 Command 0x19	11
4.7 Command 0x20	11
4.8 Command 0x22	11
4.9 Command 0x23	12
4.10 Command 0x30	12
4.11 Command 0x32	12
4.12 Command 0x37	13
4.13 Command 0x38	13
4.14 Command 0x39	13
4.15 Command 0x3d	14
4.16 Command 0x40	14
4.17 Command 0x41	14
4.18 Command 0x50	15
4.19 Command 0x51	15
4.20 Command 0x59	15
4.21 Command 0x5a	16
4.22 Command 0x5f	16
4.23 Command 0x60	16

4.24 Command 0x61	17
4.25 Command 0x62	17
4.26 Command 0x63	17
4.27 Command 0x64	18
4.28 Command 0x65	18
4.29 Command 0x66	18
4.30 Command 0x67	19
4.31 Command 0x70	19
4.32 Command 0x71	19
4.33 Command 0x72	20
4.34 Command 0x73	20
4.35 Command 0x74	21
4.36 Command 0x77	21
II Parallel Port	22
5 Overview and Communication Parameters	23
6 Description of the parallel port modes	24
6.1 Gray Code Mode	24
6.2 Interrupt Mode	25
III Appendix	26
A Example	27
A.1 serial number	27

Chapter 1

Introduction

This Document describes the ports of the Par devices from www.true-random.com, e. g. RW3USB.

Generally the conventions of ANSI-C, IEEE 754 and SI are used.

Part I
USB and Serial Port

Chapter 2

Communication Parameters

From the computer's point of view the Par devices are UARTs which are connected via an USB-UART adapter. This adapter uses an FTDI FT232BM which is also used e. g. in many USB-RS232 adaptors. The driver for the FT232BM is usually automatically loaded directly after plug-in because the driver is in the Linux kernel since vesion 2.4. For other supoorted operating systems, MacOS and Microsoft-Windows, go to the driver page on www.ftdichip.com. For other operating systems, e. g. BSD, you have to look into there documentation.

The communication parameters are 1,500,000 8N1: 1.5 Mbaud, 8 Bit/Byte, no parity, one stopbit and no handshake.

The Par device usually works as a slave which sends data only as an answer to a command from the computer ("master"). The only exception is going into a sleep mode when the supply voltage of the USB bus goes down: Than the Par device sends a quit message. But that should never happen because the USB supply voltage usually can only go down during a plug out and at plug out the data lines are disconnected before the power supply lines.

Every correctly received command is acknowledged from the Par with a data packetwhich starts with the same command. In case of an error nothing is send from the Par device. In future firmware versions it's possible to send an error message (command 0x77, see next two chapters) when there are enough requests to do so.

The data are transfered half duplex and the Par device usually answers within one millisecond. If the device does not answer within 100 milliseconds you can be shure that the data packet has been lost (packet loss) or that the device is down or has closed the connection (command 0x22, see next two chapters).

The data are transfered in big endian format, so you can read them easily by printing the data hexadecimal.

Chapter 3

Command List

The following table shows all commands which can be used or occur at the communication between a Par device and a computer.

The command is always the first byte of a data packet and this byte determines a) the length (2..256 Byte) and b) the structure of the data packet (next chapter).

The data packets always do have this structure:

Byte_0 (command), ... Byte_N (XOR checksum).

The XOR checksum, the checksum for transfer errors, is the exclusive or of the Bytes_0 til Byte N-1: $\text{Byte}_0 \oplus \text{Byte}_1 \dots$, and the Par devices do ignore all packets with a faulty checksum.

The checksum check is simply $\text{Byte}_0 \oplus \dots \oplus \text{Byte}_N$ and the result is zero when the checksum is ok.

The XOR checksum detects all one-bit-errors and most other errors.

When the data packet is longer than 2 Bytes than the second byte must be the address of the device:

Byte_0 (command), Byte_1 (address), ... Byte_N (XOR checksum).

Par devices do ignore all packets which do have another address.

The default address is 0x13 but the address can be questioned and set. The address is stored in an EEPROM and therefore not altered by a power down, reset or update of the firmware.

List of the commands

Command	Status	Availability	Name
0x10	ok	+	CHECK
0x11	ok	+	GET_ADR
0x12	ok	+	SET_ADR
0x13	ok	+	SOFT_RESET
0x14	ok	+	HARD_RESET
0x20	ok	+	SLEEP
0x22	ok	+	HANGUP
0x23	ok	+	SET_MODES
0x30	ok	+	MODE_AM
0x31	ok	+	MODE_LPM3
0x32	ok	+	GET_TEMPERATUR
0x37	ok	+	GET_CONFIG
0x38	ok	+	GET_SOFTWARE_CONFIG
0x39	ok	+	GET_AUTODETECT_CONFIG
0x3d	ok	+	GET_STATUS
0x40	ok	o	GET_ADC_DATA0
0x41	ok	o	GET_ADC_DATA1
0x49	ok	o	ONLINE_TEST
0x50	ok	o	SET_DAC_DATA0
0x51	ok	o	SET_DAC_DATA1
0x59	ok	o	GET_RAND_SEED
0x5a	ok	o	GET_RAND_BLOCK
0x5f	ok	+	FLASH_CHECK
0x60	ok	+	STACK_CHECK
0x61	ok	o	GET_UPTIMES
0x62	ok	o	GET_TIME
0x63	ok	o	SET_TIME
0x64	ok	o	GET_DATE
0x65	ok	o	SET_DATE
0x66	ok	o	GET_SECONDS
0x67	ok	o	SET_SECONDS
0x70	ok	+	READ_MEM
0x71	ok	+	WRITE_MEM
0x72	ok	o	READ_MSD
0x73	ok	o	WRITE_MSD
0x74	ok	o	GET_MSD_CONFIG
0x77	p	+	ERROR

3.0.1 Remarks

Name: The name is the define which is used by the Software/Firmware.

Availability: A “+” means that this command is available for all Par devices.

A “o” means this command is available for some Par devices. Examples: The date/time commands are only available for Par devices with a real time clock, the ONLINE_TEST is only available for the Par devices with a separate true random number generator.

The manual for each device contains a list of the commands which are available for that device.

Status: An ok means that the command is implemented and successfully tested.

A p means it is planned and it can be available after an update of the firmware.

The time/date of the par devices is without leap seconds and without daylight saving time.

The Par devices without a real time clock do ignore all time/date commands.

Chapter 4

Detailed Description of the Commands

4.0.2 Remarks

Echo means that the Acknowledgment is the data packet which has been send.

The data are send in big endian format. Example: The string 0x010203 is transmitted as 0x01, 0x02, 0x03 with 0x01 as the first byte.

The data types used (uint8_t, uint16_t, ...) are described in the ANSI C standard (see www.ansi.org).

The default address (second byte if the command/acknowledgment is longer than two bytes) is 19.

4.1 Command 0x10

The CHECK command is for checking the connection like a small TCP/IP ping.

Length: 2 Bytes.

Structure: 0x1010

Acknowledgment length: 2 Bytes.

Acknowledgment Structure: 0x1010

4.2 Command 0x11

The GET_ADR command is for getting the actual address (uint8_t) from the Par device.

Length: 2 Bytes.

Structure: 0x1111

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x11<address byte><XOR checksum byte>

4.3 Command 0x12

The SET_ADR command is for setting the address (uint8_t) of the par device. The address is store in an EEPROM an is therefore not changed by an reset or power down.

Length: 4 Bytes.

Structure: 0x12<address byte><new address byte><XOR checksum byte>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x12<new address byte><XOR checksum byte>

4.4 Command 0x13

The SOFT_RESET command is for resetting some variables of the par device and is not fully implemented yet.

Length: 3 Bytes.

Structure: 0x13<address byte><XOR checksum byte>

Acknowledgment: Echo.

4.5 Command 0x14

The HARD_RESET command is for resetting the Par device.

After the acknowledgment the code at the RESET interrupt vector is executed in the Par device.

Length: 3 Bytes.

Structure: 0x13<address byte><XOR checksum byte>

Acknowledgment: Echo.

4.6 Command 0x19

The BIG_CHECK command is for checking the connection like a big size TCP/IP ping.

The data bytes can be any data bytes, e. g. 0, 1, 3, ... 0xfd.

Length: 256 Bytes.

Structure: 0x19<address byte><data byte 252>...<data byte 0><XOR checksum>

Acknowledgment: Echo.

4.7 Command 0x20

The SLEEP_MODE command is setting the Par device into sleep mode where only the real time clock is active (if the device has a RTC).

Length: 3 Bytes.

Structure: 0x20<address byte><XOR checksum>

Acknowledgment: Echo.

4.8 Command 0x22

The HANGUP command kills the UART communication (UART interrupt) to ensure that the work done by the Par device is not interrupted by an interrupt.

Length: 3 Bytes.

Structure: 0x22<address byte><XOR checksum>

Acknowledgment: Echo.

4.9 Command 0x23

The SET_MODES command sets the main mode (`uint16_t`) and sub mode (`uint16_t`) of the Par device.

Length: 7 Bytes.

Structure: 0x23<address byte><high byte of the main mode><low byte of the main mode><high byte of the sub mode><low byte of the sub mode><XOR checksum>

Acknowledgment: Echo.

4.10 Command 0x30

The ONLINE_TEST command is for starting the TRNG online test which is done from the Par device.

The acknowledgment is the test result (appendix).

Length: 3 Bytes.

Structure: 0x30<address byte><XOR checksum>

Acknowledgment length: 13 Bytes.

Acknowledgment Structure: 0x30<address byte><test result byte 9>...<test result byte 0><XOR checksum>

4.11 Command 0x32

The GET_TEMPERATUR is for getting the temperature (`uint16_t`) of the Par device. The temperature is in Kelvin and therefore unsigned.

Length: 3 Bytes.

Structure: 0x32<address byte><XOR checksum>

Acknowledgment length: 5 Bytes.

Acknowledgment Structure: 0x32<address byte><temperature high byte><temperature low byte><XOR checksum>

4.12 Command 0x37

The GET_CONFIG command is for getting the configuration (see Appendix) which is stored in the Par device EEPROM.

Length: 3 Bytes.

Structure: 0x37<address byte><XOR checksum>

Acknowledgment length: 23 Bytes.

Acknowledgment Structure: 0x37<configuration byte 19>...<configuration byte 0><XOR checksum>

4.13 Command 0x38

The GET_SOFTWARE_CONFIG command is for getting the configuration (see Appendix) which is hard coded in the Par device firmware.

Length: 3 Bytes.

Structure: 0x38<address byte><XOR checksum>

Acknowledgment length: 23 Bytes.

Acknowledgment Structure: 0x37<configuration byte 19>...<configuration byte 0><XOR checksum>

4.14 Command 0x39

The GET_AUTODETECT_CONFIG command is for getting the autodetected configuration (see Appendix) of the Par device firmware.

The acknowledgment only returns the things which could be really detected.

Length: 2 Bytes.

Structure: 0x39<address byte><XOR checksum>

Acknowledgment length: 23 Bytes.

Acknowledgment Structure: 0x37<configuration byte 19>...<configuration byte 0><XOR checksum>

4.15 Command 0x3d

The GET_STATUS command is for getting the status (see appendix) of the par device.

Length: 3 Bytes.

Structure: 0x3d<address byte><XOR checksum>

Acknowledgment length: 23 Bytes.

Acknowledgment Structure: 0x10<address byte>...<data byte 253><XOR checksum>

4.16 Command 0x40

The GET_ADC_DATA0 command is for getting the data (voltage, uint16_t) of the ADC pin 0 (first ADC input).

Length: 3 Bytes.

Structure: 0x40<address byte><XOR checksum>

Acknowledgment length: 5 Bytes.

Acknowledgment Structure: 0x40<address byte><data high byte><data low byte><XOR checksum>

4.17 Command 0x41

The GET_ADC_DATA1 command is for getting the data (voltage, uint16_t) of the ADC pin 1 (second ADC input).

Length: 3 Bytes.

Structure: 0x40<address byte><XOR checksum>

Acknowledgment length: 5 Bytes.

Acknowledgment Structure: 0x40<address byte><data high byte><data low byte><XOR checksum>

4.18 Command 0x50

The SET_DAC_DATA0 command is for setting the output of the DAC0 to that value (uint16_t).

Length: 5 Bytes.

Structure: 0x50<address byte>...<data byte 253><XOR checksum>

Acknowledgment length: 256 Bytes.

Acknowledgment Structure: 0x50<address byte><XOR checksum>

4.19 Command 0x51

The SET_DAC_DATA1 command is for setting the output of the DAC0 to that value (uint16_t).

Length: 5 Bytes.

Structure: 0x50<address byte>...<data byte 253><XOR checksum>

Acknowledgment length: 256 Bytes.

Acknowledgment Structure: 0x50<address byte><XOR checksum>

4.20 Command 0x59

The GET_RANDOM_SEED command is for getting a true random number (uint16_t). This number is truly random because it contains e. g. the noise from the ADC but the quality is untested.

Length: 3 Bytes.

Structure: 0x59<address byte><XOR checksum>

Acknowledgment length: 5 Bytes.

Acknowledgment Structure: 0x59<address byte><high random byte><low random byte><XOR checksum>

4.21 Command 0x5a

The GET_RAND_BLOCK command is for getting a block of 252 bytes of high quality true random numbers with an associated status byte.

The status byte is zero if the random byte do have a successfully tested high quality.

Length: 3 Bytes.

Structure: 0x5a<address byte><XOR checksum>

Acknowledgment length: 256 Bytes.

Acknowledgment Structure: 0x5a<address byte><random byte 251><random byte 0><status byte><XOR checksum>

4.22 Command 0x5f

The FLASH_CHECK command is for getting the XOR checksum (uint16_t) of the whole flash of the Par device microcontroller.

Length: 3 Bytes.

Structure: 0x5f<address byte><XOR checksum>

Acknowledgment length: 5 Bytes.

Acknowledgment Structure: 0x5f<address byte><flash XOR checksum high byte><flash XOR checksum low byte><XOR checksum>

4.23 Command 0x60

The STACK_CHECK command is for getting the size of the RAM (uint16_t) in bytes and the number of bytes which are reserved for the stack and currently unused (uint16_t). This number is always greater zero because a stack overflow causes a hard reset.

Length: 3 Bytes.

Structure: 0x60<address byte><XOR checksum>

Acknowledgment length: 7 Bytes.

Acknowledgment Structure: 0x60<address byte><stack size high byte><stack size low byte><RAM size high byte><RAM size low byte><XOR checksum>

4.24 Command 0x61

The GET_UPTIMES command is for getting the uptime in seconds (uint32_t) a) since the last (hard) reset and b) the last sleep mode.

Length: 3 Bytes.

Structure: 0x61<address byte><XOR checksum>

Acknowledgment length: 11 Bytes.

Acknowledgment Structure: 0x61<address byte><uptime a MSB>...<uptime a LSB><uptime b MSB>...<uptime b LSB><XOR checksum>

4.25 Command 0x62

The GET_TIME command is for getting the time (hour (0...23), minute (0...59), second (0...59)) of the real time clock.

Length: 3 Bytes.

Structure: 0x62<address byte><XOR checksum>

Acknowledgment length: 6 Bytes.

Acknowledgment Structure: 0x62<hour byte><minute byte><second byte><XOR checksum>

4.26 Command 0x63

The SET_TIME command is for setting the time (hour (0...23), minute (0...59), second (0...59)) of the real time clock.

Length: 6 Bytes.

Structure: 0x63><hour byte><minute byte><second byte><XOR checksum>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x63<address byte><XOR checksum>

4.27 Command 0x64

The GET_DATE command is for getting the date (year -2000 (0...255), month (1...12), day (1...31)) of the real time clock.

Length: 3 Bytes.

Structure: 0x64<address byte><XOR checksum>

Acknowledgment length: 6 Bytes.

Acknowledgment Structure: 0x64<year byte><month byte><day byte><XOR checksum>

4.28 Command 0x65

The SET_DATE command is for setting the date (year -2000 (0...255), month (1...12), day (1...31)) of the real time clock.

Length: 6 Bytes.

Structure: 0x64<year byte><month byte><day byte><XOR checksum>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x64<address byte><XOR checksum>

4.29 Command 0x66

The GET_SECONDS command is for reading the time and date of the Par device by reading the seconds (uint32_t) since 2000-01-01 0:00:00.

Remark: In TAI time the difference between 2000-01-01 0:00:00 and 1970-01-01 0:00:00 is 946684800 (see “date -ud 'Jan 1, 2000 0:00:00' +%s”).

Length: 7 Bytes.

Structure: 0x66<address byte><XOR checksum>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x66<address byte><MSB of the 32 bit second counter>...<LSB of the 32 bitsecond counter><XOR checksum>

4.30 Command 0x67

The SET_SECONDS command is for setting the time and date of the Par device by setting the seconds (uint32_t) since 2000-01-01 0:00:00.

Remark: In TAI time the difference between 2000-01-01 0:00:00 and 1970-01-01 0:00:00 is 946684800 (see “date -ud 'Jan 1, 2000 0:00:00' +%s”).

Length: 7 Bytes.

Structure: 0x66<address byte><MSB of the 32 bit second counter>...<LSB of the 32 bitsecond counter><XOR checksum>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x66<address byte><XOR checksum>

4.31 Command 0x70

The READ_MEM command is for reading the 32 byte long personal string from the EEPROM of the Par device.

Length: 3 Bytes.

Structure: 0x70<address byte><XOR checksum>

Acknowledgment length: 35 Bytes.

Acknowledgment Structure: 0x70<address byte><string byte 31><string byte 0><XOR checksum>

4.32 Command 0x71

The WRITE_MEM command is for writing the 32 byte long personal string into the EEPROM of the Par device.

Length: 35 Bytes.

Structure: 0x71<address byte><string byte 31><string byte 0><XOR checksum>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x71<address byte><XOR checksum>

4.33 Command 0x72

The READ_MSD command is for reading 253 Bytes from the Mass Storage Device (MSD). The 32 bit MSD Address (MSD_Address) is the address where the first data Byte (Byte 0) is read. The other Bytes are read from the current address, so Byte 1 is read from MSD_Address+1 and so on.

The data bytes are transferred in little endian format.

Length: 7 Bytes.

Structure: 0x72<Address Byte><MSD Address Byte 3><MSD Address Byte 3><MSD Address Byte 1><MSD Address Byte 0><XOR checksum>

Acknowledgment length: 256 Bytes.

Acknowledgment Structure: 0x72<Address Byte><Byte 0><Byte 1><Byte 2>...<Byte 252><XOR checksum>

4.34 Command 0x73

The WRITE_MSD command is for writing 1..253 Byte into the Mass Storage Device (MSD). The 32 bit MSD Address (MSD_Address) is the address where the first data Byte (Byte 0) is written to. The other Bytes are written to the current address, so Byte 1 gets written to MSD_Address+1 and so on.

The data bytes are transferred in little endian format.

Length: 256 Bytes.

Structure: 0x73<Address Byte><MSD Address Byte 3><MSD Address Byte 3><MSD Address Byte 1><MSD Address Byte 0>Byte 0><Byte 1><Byte 2>...<Byte 248><XOR checksum>

Acknowledgment length: 3 Bytes.

Acknowledgment Structure: 0x73<address byte><XOR checksum>

4.35 Command 0x74

The GET_MSD_CONFIG command is for reading the available space (for commands READ_MSD, WRITE_MSD) of the Mass Storage Device (MSD) and for reading the registers (CID, CSD, ...).

The size is returned as the 32 bit number of blocks (of 512 Byte).

Length: 3 Bytes.

Structure: 0x74<address byte><XOR checksum>

Acknowledgment length: 163 Bytes.

Acknowledgment Structure: 0x74<address byte><MSD size Byte 3><MSD size Byte 3><MSD size Byte 1><MSD size Byte 0><MSD CID Byte 3>...<MSD CID Byte 0><MSD CSD Byte 3>...<MSD CSD Byte 0><8 Bytes reserved for other registers><XOR checksum>

4.36 Command 0x77

The ERROR command is for signaling an error. The error byte is the number of the error. This command can also be send from the Par device.

Remark: The error command is unused and error messages are dropped. In the current implementation a missing Acknowledgment is used as error message.

Length: 256 Bytes.

Structure: 0x10<address byte>...<data byte 253><XOR checksum>

Acknowledgment length: 0 Bytes.

Part II

Parallel Port

Chapter 5

Overview and Communication Parameters

The Par devices do have a parallel port for plug in to a computer parallel port. The parallel port is for transferring data with hard real time, e. g. Linux +RTAI, up to 400,000 transfers per second.

The parallel port can be used in two main modes: Gray Code mode and Interrupt mode. In Gray Code mode the data are transferred Gray Code coded and asynchronously. In Interrupt mode the data are transferred interrupt driven via a line which causes an interrupt at the receiver. The receiver then toggles a handshake line to acknowledge the data and allow the sender to send the next interrupt.

In Gray Code mode up to 400,000 transfers per second can be done while in Interrupt mode approx. 100,000 transfers per second can be done.

The parallel port has 17 data lines: 8 bidirectional data lines, 4 output control lines and 5 input status lines. The function of the associated Par device lines depends on the main mode (Gray Code mode / Interrupt mode) and the sub-mode which can be set via USB (previous chapter).

Chapter 6

Description of the parallel port modes

6.1 Gray Code Mode

6.2 Interrupt Mode

Part III
Appendix

Appendix A

Example

Data packet from the computer	Data packet from the par device	comment
0x1010	0x1010	the computer sends a ping the Par device answers the ping
0x1111	0x111302	the computer is querying the address the Par device sends it's address

A.1 serial number

The serial number has the same structure as an ISBN number and can be checked online at <http://www.isbn-check.com/> .