Ein starker Partner
für Automation

ET Instrumente GmbH
Neckarauer Str. 2
D-68766 Hockenheim
Telefon 06205 / 396 910
Telefax 06205 / 396 911
E-Mail-Adresse Info@ETTGmbH.de
Internet http://www.ETTGmbH.de

Interface and Tools Description Devices with ET1307 Controller

Version: **V1.2 R3**
Date: **27.2.2012**
Editor: **S.Hefele**

Relevant Software

ebhost.dll	Host DLL (V1.2.54 or higher)
gpib32.dll	GPIB Driver
etusb.sys	USB Driver
epret.exe	Host Interpreter (V1.2.54 or higher)

Content

1. Interfaces

The devices with ET1307 Controller can communicate with the PC over different interfaces. The following interfaces are available:

USB
RS232
Ethernet
GPIB

The library "EBHOST.DLL" can be used to control a device via USB, RS232 or Ethernet interface. For the USB interface an additional driver ("ETUSB.SYS" and "ETUSB.INF") must be installed. The RS232 interface can be used directly from a terminal program (i.e. SerCom) or an application. To use the GPIB interface, a National Instruments compatible PC card with associated drivers and the library "GPIB32.DLL" is necessary.

The connection and operation of the device over one of that interfaces are described in the following chapters.

USB

In order to control the device via USB the device must be connected to a USB port using a USB cable with type A connector on one side and type B connector on the other. Then the driver must be installed. You will be asked for the driver for the "ET1307" device when first connecting the device to the PC. The driver “ETUSB.SYS” is on the support CD in a folder called “USB-DRIVER”. After the driver installation an "ET1307"-device appears in the system.

The control software does not need a special installation. It can simply be copied from the “ET-INSTRUMENTE”-folder on the CD on your PC. That folder also contains the library “EBHOST.DLL”, which realizes the communication of an application with the USB-driver of the device. The application “EPRET.EXE” is an interpreter, sending plain text inputs directly to the device.

Setting the Unit Number

If multiple devices are connected to a single PC, they have to be separated by their unit numbers. Therefore every connected device must have an unique unit number. A maximum of 127 devices can be connected to a single USB Host. The unit number n can be set by the following command:

SYSTem:UNITNUMBer n

The command can be send over each available interface. To send the command the terminal program EPret can be used.
A valid unit number has to be in the range between 0 and 255. A change requires a restart of the device to get active.

In initial state unless otherwise noted every device has unit number 0. Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules. If only a single device is used on PC, there will be no change necessary.

RS232

If the device should be controlled via the serial interface, the device must not be connected via USB. It has to be connected to an unused COM-Port on the PC via a null modem cable (cross over RS232 cable). The transfer settings are the following:

Baud rate:	9600 Bit/s
Data bits:	8 Bit
Stop bits:	1 Bit
Parity:	None
Protocol:	None

Every command has to be terminated with an ASCII line feed character <LF> (ASCII-Code 10). A response is also terminated by a line feed character.
To send and receive commands the terminal program EPret can be used.

RS232 connector pinning

Function	PIN	PIN	Function
		9	N.C.
CTS	8	7	RTS
N.C.	6	5	COM (GND)
N.C.	4	3	TxD
RxD	2	1	N.C.

Ethernet

To control the device via Ethernet (LAN) interface, the device must be connected to an Ethernet port with an Ethernet cable (LAN cable) with RJ-45 connectors (module connector 8P8C). The communication follows the norm 10BASE-T / 100BASE-T. The connection can also be made over a router or hub. If an Ethernet connection is made, the USB and RS232 interfaces are inactive.

The device must be initialized with its unit number and its IP address (IPv4). In initial state unless otherwise noted every device has unit number 0 and the IP address "192.168.172.180". Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules and the IP addresses "192.168.172.181" for module 1 and "192.168.172.182" for module 2. If only a single device is used on PC, there will be no change necessary.

The IP address can be changed by command, if necessary and the communication to the device is established. The command is:

SYSTem:IP <IP-Address>

The command can be send over each available interface. To send the command the terminal program EPret can be used.
Devices with successive unit numbers should have successive IP addresses. This allows device systems to be initialized automatically. All valid IP addresses can be used, but it is strongly recommended to use the subnet addresses 192.168.x.x. It is necessary that the IP address of the device fits to the subnet of the LAN the device is connected to. A change requires a restart of the device to get active.

GPIB

The IEEE488.2-BUS interface is implemented with a National Instruments GPIB-Controller. To use the GPIB interface a specific driver and the library "GPIB32.DLL" is necessary. The following interface functions according to the standard IEEE 488.2 are available:

SH1	Source Handshake	:	all functions
AH1	Acceptor Handshake	:	all functions
T6	Talker	:	basic equipment, serial-Poll, End-addressing by MLA
L4	Listener	:	basic equipment End-addressing by MLA
SR1	Service Request	:	all possibilities
RL1	Remote Local	:	all abilities (with closing LLD)
PP1	Parallel Poll	:	adjustment remote control
DC1	Device Clear	:	all abilities
DTO	Device Trigger	:	no abilities
CO	Controller	:	no abilities

The connection to the BUS happens over a 24-pol plug connection by IEEE-488.2 norm.

Setting the DIP switch

The operating mode is selected with the six fold DIP switch next to the IEEE488 connector. The switch setting is read once at power up and because of that fact the device has to be power down and power up again when you want to change the settings.

Function assignment of the switch:

Switch 6(MSB): OFF Operating via the RS232-interface
ON Operating via the IEEE488.2-bus.

The function of the five switches depends on the operating mode (RS232/V24 or IEEE488.2). They are only in IEEE488.2 mode active.

GPIB Address

With the switches 5 down to 1 the device address at the IEEE488.2-bus is adjusted. Addresses in the range from 1 to 30 are permitted. When an invalid address is selected no connection is possible and the GPIB driver or software will react with a “no listener” error. A new address is only assigned after completed reboot (power down and power up). The device address has to be set binary coded. Switch 1 has the value 1, the second switch value 2, the third value 4 and so on. In position ON the value is added to the address, in position OFF that switch does not influence the address.

Example: The unit address 6 has to be set.
6 = 4 + 2,
i.e. switch three (value 4) and switch 2 (value 2) have to be active (position ON). The other ones have to be turned off.

All possible settings are listed below to simplify the adjustment.

Factory-made address adjustment: **5** (IEEE488.2-Bus)

Sw. 5	Sw. 4	Sw. 3	Sw. 2	Sw. 1	Device-address	Listener-address	Talker-address
OFF	OFF	OFF	OFF	ON	1	!	A
OFF	OFF	OFF	ON	OFF	2	"	B
OFF	OFF	OFF	ON	ON	3	#	C
OFF	OFF	ON	OFF	OFF	4	$	D
OFF	***OFF***	***ON***	***OFF***	***ON***	***5***	***%***	***E***
OFF	OFF	ON	ON	OFF	6	&	F
OFF	OFF	ON	ON	ON	7	'	G
OFF	ON	OFF	OFF	OFF	8	(	H
OFF	ON	OFF	OFF	ON	9	)	I
OFF	ON	OFF	ON	OFF	10	*	J
OFF	ON	OFF	ON	ON	11	+	K
OFF	ON	ON	OFF	OFF	12	,	L
OFF	ON	ON	OFF	ON	13	-	M
OFF	ON	ON	ON	OFF	14	.	N
OFF	ON	ON	ON	ON	15	/	O
ON	OFF	OFF	OFF	OFF	16	0	P
ON	OFF	OFF	OFF	ON	17	1	Q
ON	OFF	OFF	ON	OFF	18	2	R
ON	OFF	OFF	ON	ON	19	3	S
ON	OFF	ON	OFF	OFF	20	4	T
ON	OFF	ON	OFF	ON	21	5	U
ON	OFF	ON	ON	OFF	22	6	V
ON	OFF	ON	ON	ON	23	7	W
ON	ON	OFF	OFF	OFF	24	8	X
ON	ON	OFF	OFF	ON	25	9	Y
ON	ON	OFF	ON	OFF	26	:	Z
ON	ON	OFF	ON	ON	27	;	[
ON	ON	ON	OFF	OFF	28	<	
ON	ON	ON	OFF	ON	29	=	]
ON	ON	ON	ON	OFF	30	>	^

2. Description Library EBHOST.DLL

The library “EBHOST.DLL” can be implemented in own applications. It supplies interface functions to communicate with the device via the USB driver.

In the file “EBDEF.h” every prototypes, functions and constants, you can call in your application are defined. This file is written in C-Syntax and can be used in C- or C++ programs. However it is also possible use Visual Basic or another programming language. You can use the USB, RS232, Ethernet or GPIB-Interface to communicate. The commands are SCPI-conform. Maximal 256 devices can be handled by the library.

In the following section the constant values of the error codes, defined in the file “EBDEF.H” according to the SCPI standard, are quoted:

```
cEbAck = 0;                 // no error {device and DLL}
cEbErrGeneric = -102;       // syntax error {device only}
cEbErrUnknown = -113;       // undefined header (unknown command or function)
                            // {device only}
cEbErrBusy = -190;          // device still busy {device and DLL}
cEbErrSetting = -221;       // settings conflict (command not valid in the current state)
                            // {device only}
cEbErrRange = -222;         // out of range {device only}
cEbErrParam = -224;         // illegal parameter value (digital value) {device only}
cEbErrSlot = -228;          // address error (slot or channel) {device only}
cEbErrInstrFormat = -232;   // invalid format (instruction format) {device and DLL}
cEbErrHw = -240             // hardware error {device only}
cEbErrHwOvp = -241          // hardware error: over voltage {device only}
cEbErrHwOt = -242           // hardware error: over temperature {device only}
cEbErrHwMod = -243          // hardware error: module not present {device only}
cEbErrHwHv = -244           // hardware error: HV fail {device only}
cEbErrHwPow = -245          // hardware error: over power {device only}
cEbErrHwPowPred = -246;     // hardware error: over power prediction {device only}
cEbErrHwOvc = -247;         // hardware error: over current {device only}
cEbErrHwOvcPred = -248;     // hardware error: over current prediction {device only}
cEbErrHwRegFail = -249;     // hardware error: load mode regulation error {device only}
cEbErrHwTime = -250;        // hardware error: output on timeout error {device only}
cEbErrTooMany = -350;       // too many errors (error queue overrun) {device only}
cEbErrCom = -360;           // communication error (rs232, usb, gpib) {device and DLL}
cEbErrInstrQueue = -363;    // input buffer overrun (instruction queue) {device only}
cEbErrInstrSeq = -400;      // query error (instruction transfer sequence) {device only}
```

```
cEbErrUnit = -901;              // unit not present {DLL only}
cEbErrDllException = -910;      // dll exception {DLL only}
cEbErrMemException = -911;      // out of memory error {DLL only}
cEbErrFileException = -912;     // file handling error {DLL only}
cEbErrSocketException = -913;   // socket error {DLL only}
cEbErrUnitLocked = -920;        // device locked {DLL only}
```

In the following section function as defined in the file “EBDEF.H” are quoted:

long EbInit(long nUnit, const char pID, long nIDSize, long nComPort);*

This function initializes the device with the unit number nUnit and the specified ID at the position pID. If that device does not have the ID, it will not be initialized. The port number of the RS232 Interface has to be set in nComPort. If USB communication should be used that variable has to be set to zero. The return value is an error code.

long EbInitNet(long nUnit, const char pIP, long nIPSize);*

This function initializes the device with the unit number nUnit and the IP address pIP. In initial state unless otherwise noted every device has unit number 0 and the IP address "192.168.172.180". Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules and the IP addresses "192.168.172.181" for module 1 and "192.168.172.182" for module 2. The return value is an error code.

long EbReset(long nUnit);

This function resets the communication of device with the unit number nUnit. After a reset the communication needs to be initialized again. The return value is an error code.

long EbCommand(long nUnit, const char* pCmd, long nCmdSize);

This function sends a SCPI-command pCmd with the length nCmdSize to the device with the unit number nUnit. That device has to be initialized. The return value is an error code.

long EbQuery(long nUnit, const char* pCmd, long nCmdSize, char* pAnswer, long nAnswerSize);

This function sends a SCPI-query at pCmd to the device with the unit number nUnit. This device has to be initialized. The answer is written in pAnswer. The buffer array should be big enough to contain the expected answer. If the size is unknown, a 64 Byte huge buffer can be used. The return value is an error code.

long EbLock(long nUnit);

With this function a lock state can be set. This state marks the device with unit number nUnit as locked from another application. The function does not block other commands and queries to be executed. It is just a flag to make synchronization between applications easier. The return value is an error code.

long EbUnlock(long nUnit);

With this function a lock state can be reset. This state marks the device with unit number nUnit as unlocked to free it for another application. The function does not block other commands and queries to be executed. It is just a flag to make synchronization between applications easier. The return value is an error code.

long EbGetLockState(long nUnit);

With this function the lock state can be queried. If the device with unit number nUnit is marked as locked, the return value is cEbErrUnitLocked, else the return value is cEbAck. The function does not block other commands and queries to be executed. It is just a flag to make synchronization between applications easier. The return value is an error code.

An application can use the lock flag to mark a sequence of several commands and queries to be executed without being disturbed or interrupted by another application. Therefor all application have to query the lock state of the device first, befor they try to send a command or query. If the device is not lock, an application can lock it for its own use. Otherwise it has to wait, until the device is unlocked again. After the critical sequence of commands and queries the application who has locked the device must unlock it again.

3. Description Interpreter EPret

With the interpreter software EPret commands and queries can be send and the answers of the device can be received. EPret is using the library “EBHOST.DLL" to communicate with the device via USB, RS232 or Ethernet interface and it is using the library "GPIB32.DLL" to communicate via GPIB interface. The commands and queries of the specific device are documented in the device manual and are always conform with the SCPI standard.

The interpreter provides some additional commands for initializing and scripting. Scripts are sets of commands and queries that can be executed in a one sequence. Scripts are stored in files, so that they can be used every time needed. The files are simple ASCII files with the extension "*.SBI". The commands and queries of a executed script sequence along with the query answers are stored in a result file. The result files are simple ASCII files with the extension "*.SBA". The files are stored to the same folder the scripts are stored. If the execution of a scripts is repeated, an existing older result file will be replaced.

Initializing must be done before you can send device commands and queries to the device. The Initialization must be done each time the device is switched on or the interpreter EPret is started.

USB

To initialize a device on the USB interface, following command can be used:

InitUsb(<UnitNr>)

The unit number <UnitNr> is used to recognise several devices in systems. Each connected device must have an unique unit number. In initial state unless otherwise noted every device has unit number 0. Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules.

If the unit number of a specific device is unknown, it can be found by connecting this device only and use the following command:

FindUnit

The found unit number will be displayed in the protocol window. The device will be initialized with the found unit number with that command too. If the unit number could not be found, an error message is displayed.

If the device is initialized, the unit number can be set by the following command:

SYSTem:UNITNUMBer <UnitNr>

The command can be send over each available interface.
A valid unit number has to be in the range between 0 and 255. A change requires a restart of the device to get active.

Several devices with different unit numbers can be initialized. The devices can be selected to communicate with by the following command

SelUsb(<UnitNr>)

This commands selects the device with unit number <UnitNr> and all following commands and queries will be send to that device.

RS232

To initialize a device on the RS232 interface, following command can be used:

InitCom(<ComPort>)

The number of the physical COM port the device is connected to must be transfered in <ComPort>.

Several devices with different unit numbers can be initialized. The devices can be selected to communicate with by the following command

SelCom(<ComPort>)

This commands selects the device with COM port <ComPort> and all following commands and queries will be send to that device.

Ethernet

To initialize a device on the Ethernet (LAN) interface, following command can be used:

InitNet(<UnitNr>, <IP-Address>)

The unit number <UnitNr> is used to recognise several devices in systems. Each connected device must have an unique unit number and an unique IP address. In initial state unless otherwise noted every device has unit number 0 and the IP address "192.168.172.180". Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules and the IP addresses "192.168.172.181" for module 1 and "192.168.172.182" for module 2.

The IP address can be changed by command, if necessary and the communication to the device is established. The command is:

SYSTem:IP <IP-Address>

The command can be send over each available interface. To send the command the terminal program EPret can be used.
Devices with successive unit numbers should have successive IP addresses. This allows device systems to be initialized automatically. All valid IP addresses can be used, but it is strongly recommended to use the subnet addresses 192.168.x.x. It is necessary that the IP address of the device fits to the subnet of the LAN the device is connected to. A change requires a restart of the device to get active.

Several devices with different unit numbers and IP addresses can be initialized. The devices can be selected to communicate with by the following command

SelNet(<UnitNr>)

This commands selects the device with unit number <UnitNr> and all following commands and queries will be send to that device.

Close

This command closes the connection to the active device.

GPIB

To communicate with a device over the GPIB interface (IEEE488.2-BUS) a National Instruments compatible GPIB card must be installed in the PC. The appropriate GPIB driver along with the library "GPIB32.DLL" must be installed.
To initialize a device on the GPIB interface, following command can be used:

InitGpib(<GpibAddr>)

The GPIB address < GpibAddr > is used to recognise several devices in systems. Each connected device must have an unique GPIB address.

Several devices with different GPIB addresses can be initialized. The devices can be selected to communicate with by the following command

SelGpib(<GpibAddr>)

This commands selects the device with GPIB address < GpibAddr > and all following commands and queries will be send to that device.

Extended Funktions

The extended functions, that can be used to write scripts, are described in the following:

Step(<Nr>, <Count>, <Var1 startvalue> ,<Var1 inc>, <Var2 startvalue>, <Var2 inc>)

Definition of a Step in a script. A step can be executed one time or can be repeated. It can be repeated automatically <Count> times. While running a sequence the execution can be continued with the previous or the next step. Therefor the step must have an unique and successive number <Nr>.

<Nr>	Number of step
<Count>	Count of step repeats (loops)
<Var1 startvalue>	Start value according to the variable <1> in the script. The variable can replace a slot number or channel.
<Var1 inc>	If this value is '1', the value of the variable <1> will be increased by 1 each repetition of the step.
<Var2 startvalue>	Start value according to the variable <2> in the script. The variable can replace a slot number or channel.
<Var2 inc>	If this value is '1', the value of the variable <2> will be increased by 1 each repetition of the step.

Prompt("<text>")

The text <text> will be displayed in the protocol window and in the result file.

MsgBox("<text>")

The text <text> will be dusplayed in a dialog window during the script execution.

Delay(<Zeit in [ms]>)
Wait(<Zeit in [ms]>)

Delay of the script execution for <time in [ms]> at the point of this command.

Condition("<Condition char>", "<Condition string>", "<Fail comment>")

Definition of a condition. The string or value of the string <Condition string> must fit a specified condition. The condition is defined by the <Condition char>. If the condition does not fit, the test is FAIL and the fail message <Fail comment> will be prompted to the user in the protocol window and in the result file. If all conditions in a scipts fit, the test is PASS.
The <Condition string> is compared to the answer string or answer value of the previous query. This could be a measurement query. The <Condition char> defines the condition type. The following types can be used:

"$"	Compare the result string with the <Condition string> as string
"="	Result equal to value of <Condition string>
"<"	Result less than value of <Condition string>
"<="	Result less or equal to value of <Condition string>
">"	Result greater than value of <Condition string>
">="	Result greater or equal to value of <Condition string>

4. Generic functions of devices with ET1307 Controller

The following commands and queries are common to all devices with ET1307 controller. In addition every device type has its own set of commands and queries which are described in the specific device manual. All commands and queries are conform with the SCPI standard.

**IDN?*
**VER?*
**RST*
**CLR*
**CLS*

SYSTem:ERRor?
SYSTem:VERsion?
SYSTem:UNITNUMBer n
SYSTem:UNITNUMBer?
SYSTem:IP ip
SYSTem:IP?

Meaning of placeholders:

n	integer value	// positive integer value
ip	ip address string	// IPv4 address in the form xxx.xxx.xxx.xxx

**IDN?*

Query of devices identifier (ID).

**VER?*

Query of firmware version.

**RST*
**CLR*

Reset the device to the switch-on state. All outputs will be turned off.

**CLS*

Reset error queue.

SYSTem:ERRor?

Query of next error in the error queue. The primarily error is returned from the queue (fifo). If the error queue contains no more errors, a '0' (no error) is returned.

SYSTem:VERsion?

Query of the firmware version.

SYSTem:UNITNUMBer n
SYSTem:UNITNUMBer?

Setting the unit number of the device.
The command can be send over each available interface.
A valid unit number has to be in the range between 0 and 255. A change requires a restart of the device to get active.
In initial state unless otherwise noted every device has unit number 0. Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules.
If multiple devices are connected to a single PC, they have to be separated by their unit numbers. Therefore every connected device must have an unique unit number. If two devices with the same unit number are connected to the PC, the devices could not be recognized and a communication is not possible.

SYSTem:IP <IP-Address>

Setting the IP address of the device.
The command can be send over each available interface.
In initial state unless otherwise noted every device has unit number 0 and the IP address "192.168.172.180". Dual devices normally has the unit numbers 1 and 2 for the two integrated device modules and the IP addresses "192.168.172.181" for module 1 and "192.168.172.182" for module 2.
Devices with successive unit numbers should have successive IP addresses. This allows device systems to be initialized automatically. All valid IP addresses can be used, but it is strongly recommended to use the subnet addresses 192.168.x.x. It is necessary that the IP address of the device fits to the subnet of the LAN the device is connected to. A change requires a restart of the device to get active.

5. Troubleshooting

Problem	Solution
Communication over RS232 does not work.	Make sure that no other interface is connected. Make sure you are using a null modem cable (cross over RS232 cable). Check the COM port number, the device is connected to and make sure to initialize the device with that COM port number. In case of an existing GPIB interface make sure to set DIP switch 6 to off position.
Communication over USB does not work.	Check the unit number. Make sure that the USB driver is installed correctly and the library EBHOST.DLL is on your system. Make sure that the optional Ethernet or GPIB interface is not connected. In case of an existing GPIB interface make sure to set DIP switch 6 to off position.
Communication over Ethernet (LAN) does not work.	Check the unit number. Make sure that the IP address is unique in your network (LAN) and the subnet address of the router or PC fits to devices IP address. If no router is connected to your LAN, turn of DHCP of the control PC and give it a fix IP address. After installation or changing the device configuration in your LAN, turn all hubs off and on again to reset all stored MAC addresses.
Communication over GPIB does not work.	Check the unit number. Make sure to set DIP switch 6 to on position and to set the correct GPIB address to DIP switches 1 – 5.

If all checks are made, please restart the device. If the problem still remains, please contact the manufacturer ET Instrumente GmbH.