RS232LIB.DLL

Instructions on how to use the win32s-DLL RS232LIB.DLL for HP VEE
Program and document written by Tomas Tungel, Ericsson Telecom AB

Introduction

This short document describes the two functions that are included in rs232lib.dll. The problem was that if you wanted to use a RS-232-port in HP VEE you had to set all options in your instrument manager before you could use it. That made it impossible to change port-settings during runtime of a VEE-program.

This DLL will enable you to dynamically change the RS-232-settings but still use the standard functions READ, WRITE and DAV. The DLL is just an experiment and I am not to blaim for any bugs that the program might cause. Ericsson is just the company that pays my salary - don't pick on 'em! Please send me any suggestions - do not send me any complaints (you choosed to include the library into your program - not I :-)

This is my companycard:

ERICSSON TELECOM AB
Tomas Tungel
TN/ETX/PN/XAKD
126 25 STOCKHOLM

Email: Tomas.Tungel@ericsson.com

Tomas Tungel, 980129

Add your serial ports to HP VEE

To be able to use the built in serial-commands you must setup a RS-232-port in the programs delivered with HP VEE. Even if windows know that the ports exist you must configure them as follows:

1. Configure the port using I/O-config

- Start program I/O-config from HP I/O-libraries
- Check that RS-232 is available in list of available interfaces
- If none is configured - doubleclick and set it up -
  Logical Unit will be called channel and address in HP-VEE
- Save your settings by ending I/O-config (clicking OK)

2. Add the port in your Instrument Manager in HP-VEE

- Start HP-VEE
- Start Instrument Manager form the menu I/O
- Click ADD, set name, select interface (Serial) and type address (LU stated in I/O-config)
- Use advanced I/O-config to modify the default interface settings

Advanced I/O-config holds settings that you want HP VEE to use when it uses a port (these setting can not be modified during runtime (!!!) but you can change settings by changing to an instrument with same port but a different port configuration). If you would like to change instrument (change device settings) - remember that you have to execute a RESET for the device to make the new settings valid.

A simple use of serial communication in VEE

If you are attached to a serial terminal, and you do not know the size of incoming data, when it might appear and so on, this could be a good approach using the VEE-devices:

* Use advanced I/O-interface event (you get a DAV-box) to be able to check if any data comes into the buffer. I often configure this box not to wait, to be called by an "Until Break" and to send the data (number of bytes) to a multidevice direct I/O-operation READ (a read from serial device expecting text in char format and the amount of chars that is sent from the DAV-box). If you use this type of reading you might get the data from the serial port in chopped pieces. Therefor it might be good to send the output from the READ-box to a virtual buffer (actually I use a VEE-string), and let an external routine extract data in an approriate way.

* Use a multidevice direct I/O-operation WRITE (a write to serial device sending text from input pin in string format with default width) when you want to send something.

So, let us say that you are controlling an external device, and that you want to use a command to make the external device change the communication speed (baud) during runtime. In HP-VEE 4.0 (and at least all versions below that) you have to use another configured instrument (if you configure instruments in the manager called "COM1_9600", "COM1_19200", "COM1_19200_XonXoff" and so on) or use an external DLL to change these settings. That is why I have written this interface.

The DLL calls the SICL-library and primary make use of a few different SICL-functions - iserialctrl(), iserialstat() and iserialbreak(). As I programmed the new functions I used the book called "HP Standard Interface Control Library Language Reference" - therefor a lot of the variables you send and recieve from the functions look a lot like the constant-names in the SICL-library.

How to use the library in your VEE-program:

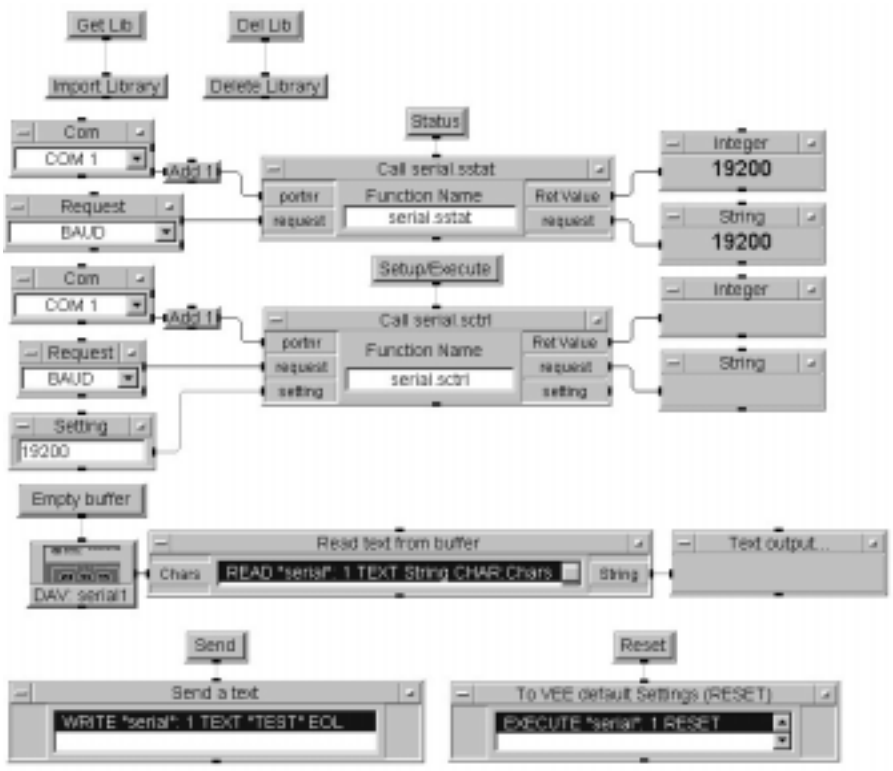
1. Import the library

Use "Import Library"-box that can be retrieved from the Device-menu. The library type shall be set to "Compiled Function" (the DLL is compiled), the name is the name of you library (so that you can import multiple libraries without using the same functionnames and so that you easily can delete a library from memory later), the filename (click on the box and allocate the file rs232lib.dll) and de-fenition-file (click on the box and allocate the file rs232lib.h). As your program executes the box it will import the compiled functions if they already are not imported).

If you for instance name your library "iserial" the imported functions will be named iserial.sstat and iserial.sctrl.

2. Using the library

Use the library as described below *if* you need to change the port settings during runtime. I strongly suggest that you call the functions within subroutines so that you can replace the DLL in the future or if the possibility of changing port settings during runtime is enabled in future versions of HP VEE). Remember that if you use EXECUTE RESET on a serial interface within a multidevice I/O you will reset the port settings to the default port settings stated in your instrument manager.



3. Using a function

When you want to use a function in your program, first import the DLL. Then the most easy way is to use your VEE program explorer to allocate the imported function, rightclick over the function-name and select "Generate Call". Then you will get a box with the in- and output-pins that are re-quired. Connect them to your dropdown lists or other boxes that can serve as input.

4. Removing the library

If you still are running a program but don't need the library any more you can use the box "Delete Library" to remove the functions from the program (the actual files will not be deleted!).

DLL functions defenition:

Common definitions

Returncodes: If the functions bumps into an error the returnvalue will be negative. These are the possible states and the texts that will be returned in the request-string:

>0	*n*	Request was performed, data *n* returned The returned data can be usefull to control lists and buttons - see if you can make use of the returned value!
0	OK	Request was performed
-1	Port access error	The port could not be accessed, is the port configured or illegaly specified?
-2	Port operation error	The port could be opened and the request was accepted by the function but the SICL-library trapped an error
-3	Undefined setting	The setting you try to send is not defined - did you try to pass an illegal setting?
-4	Undefined request	The request could not be performed be-causethe function does not recognize it

Portnr The variable portnr is a number representing your comport - pass 1 for COM1, pass 2 for COM2 and so on. If you select a port that can't be opened the function will re-turn an error code (se above).

Functionname: int sctrl (*int portnr, char *request, char *setting*);

How to use it: When you want to reconfigure an interface, pass the COM-port number, request what you want to reconfigure (see below) and setting to the function. The function will return an error number or value associated with the request and will change the inputdata request to a status message. Always use capital letters in the request- and setting-strings.

Request The request is defined as a string that is passed to the function. These are the possible request strings and settings:

"BAUD"	Changes the port speed. Available setting-strings are "50", "110", "300", "600", "1200", "2400", "4800", "7200", "9600", "19200", "38400" and "57600". The function will return the baud-setting as an integer if an error not occur.
"PARITY"	Changes the parity. Available setting-strings are "EVEN", "ODD", "NONE", "MARK" and "SPACE". The function will return an integer according to the string - EVEN results in 0, ODD returns 1, NONE returns 2 and so on.
"STOP"	Changes the number of stopbits. Use setting-string "1" or "2". The function will return the stopbits-number as an integer - 1 or 2.
"WIDTH"	Changes the number of startbits. Use setting-string "5", "6", "7" or "8". The function will return the startbits-number as an integer - 5, 6, 7 or 8.
"BUFSZ"	Changes the size of the buffer. Use a setting-string representing a integervalue between 2 and 32766. The function will return the buffersize as an integer.
"DUPLEX"	Changes the duplex-setting. The valid setting-strings are "FULL" and "HALF". The function will return an integer - "FULL" returns 0 and "HALF" returns 1.
"FLOW"	Changes the flow control. Valid setting-strings are "NONE", "XON", "RTS" and "DTR". The function will return an integer - "NONE" returns 0, "XON" returns 1 and so on.
"READ_EOI"	Changes the type of END indicator to be used for reads. Valid setting-strings are "NONE", "BIT8" or a char that indicates the read-terminator. The function will return 1 for "NONE", 2 for "BIT8" and 0 for a char terminator.
"WRITE_EOI"	Changes the type of END indicator to be used for writes. Valid setting-strings are "NONE" and "BIT8". The function will return 0 for "NONE" and 1 for "BIT8".
"RESET"	The function will perform a reset on the serial interface. Any pending writes will be aborted, the inputbuffer will be cleared and any error conditions will be reset. A break will also be sent. The function returns OK or error-code.
"BREAK"	The function will send a BREAK on the serial interface - iserialbreak(). The function returns OK or error-code.
"CLEAR"	The function will send a CLEAR on the serial interface - iclear(). The function returns OK or error-code.

Functionname: int sstat (*int portnr, char *request*);

How to use it: When you want to know the setup of a serial interface, pass the COM-port number and request (what setting you want back) to the function. The function will return an error number or value associated with the request and will change the inputdata request to a status message. Always use capital letters in the request-string.

Request The request is defined as a string that is passed to the function. These are the possible request strings:

"BAUD" Displays the port speed. The function will return the baud-setting as an integer if an error not occur. The request-string will be changed to the Baud-value as a string.

"PARITY" Displays the parity. The function will return an integer according to the string - EVEN results in 0, ODD returns 1, NONE returns 2 and so on. It will also modify the request-string to the current state ("EVEN", "ODD", "NONE", "MARK" or "SPACE").

"STOP" Display the number of stopbits. The function will return the stopbits-number as an integer - 1 or 2. The request-string will be modified to the bit-value.

"WIDTH" Displays the number of startbits. The function will return the startbits-number as an integer - 5, 6, 7 or 8. The request-string will be modified to the bit-value.

"BUFSZ" Displays the size of the buffer. The function will return the buffersize as an integer. The request-string will be modified to state the size of the buffer.

"DUPLEX" Displays the duplex-setting. The function will return an integer - "FULL" returns 0 and "HALF" returns 1. The request-string will be modified to print the current state. This request often results in an error.

"FLOW" Displays the flow control setting. The function will return an integer - "NONE" returns 0, "XON" returns 1 and so on. The request-string will be modified to the current state ("NONE", "XON", "RTS" or "DTR").

"READ_EOI" Displays the type of END indicator to be used for reads. The function will return 1 for "NONE", 2 for "BIT8" and 0 for a char terminator. If a character-terminator has been defined the request-string will be modified to contain the char-string. Otherwise it will show the current state.

"WRITE_EOI" Displays the type of END indicator to be used for writes. The function will return 0 for "NONE" and 1 for "BIT8". The request-string will be modified to "NONE" or "BIT8".

"DAV" Displays the amount of data that is available for reading from the input buffer.

These queries does not work in my environment - I can't figure out why:

"STAT" Displays the current status of the interface. The request will return a maskable integer of which the first 6 bits contain the status. The request-string is modified to represent the bit-values as a 8-character long string. Status-codes are:

Bit 1	x-------	I_SERIAL_DAV	Data is available
Bit 2	-x------	I_SERIAL_PARITY	Parity error occured
Bit 3	--x-----	I_SERIAL_OVERFLOW	Overflow error occured
Bit 4	---x----	I_SERIAL_FRAMING	Framing error occured
Bit 5	----x---	I_SERIAL_BREAK	Break has been received
Bit 6	-----x--	I_SERIAL_TEMT	Transmitter is empty

The bits will be reset after a status request has been performed.

"MSL" Displays the status of all modem status lines that are currently being asserted. The request will return a maskable integer of which the first 8 bits contain the status. The request-string is modified to represent the bit-values as a 8-character long string. Status-codes are:

Bit 1	x-------	I_SERIAL_DCD	Data carrier detect
Bit 2	-x------	I_SERIAL_DSR	Data set ready
Bit 3	--x-----	I_SERIAL_CTS	Clear to send
Bit 4	---x----	I_SERIAL_RI	Ring indicator
Bit 5	----x---	I_SERIAL_TERI	Trailing edge or RI

These bits will be reset after a status request has been performed:

Bit 6	-----x--	I_SERIAL_D_DCD	DCD-line has changed
Bit 7	------x-	I_SERIAL_D_DSR	DSR-line has changed
Bit 6	-------x	I_SERIAL_D_CTS	CTS-line has changed