Support for Communication Ports in Imagine

Preliminary documentation V1 20th October 2002 Peter Tomcsanyi

To support communication ports (like serial ports and parallel ports) there is a new class CommPort implemented in Imagine. This document briefly documents it.

The CommPort Class

The ComPort class' descendants can be used to denote one particular serial or port. It contains settings, which allow the user to set the characteristics of that port and it contains procedures, which allow executing the actual communication.

Settings

port

Its value is the name of the actual serial or parallel port, which the object refers to. The typical values are COMn or LPTn where n is the number of the port (COM1, COM2, LPT1, etc.). Other port names can be used only if the particular computer has such a port implemented.
The default value is COM2.

baudrate

Gives the speed of communication in Bd. The value is the actual number (like 9600, 2400, ...).
The default value is 9600.

databits

Number of data bits transmitted for each character. Valid values are 5, 6, 7 and 8.
The default value is 8.

stopbits

Number of stop bits used during transmission. Valid valies are 1, 1.5 and 2.
The default value is 1.

parity

Parity used during transmission. Valid values are 0 to 4. The meaning of these numbers is:

Value	Meaning
0	no parity
1	odd parity
2	even parity
3	"mark parity"
4	"space parity"

The default value is 0 (no parity)

RTSCTS

Defines the way in which hardware flow control protocol is used. Valid values are "true, "false and 1. The meaning is:

Value	Meaning
true	use RTS/CTS hardware flow control
false	do not use RTS/CTS hardware flow control
1	do not use RTS/CTS hardware flow control but set the RTS line on and keep it during the whole communication

The default value is 1.

XONXOFF

Defines whether the software flow control protocol is used. Valid values are "true and "false.
The default value is "false.

asyncInput

Defines the way in which the user wants to handle input from the port.
If asyncInput is set to false, then the input must be retrieved by calls to the Receive procedure, which may wait for it if the input is not available. In this mode it is possible to check WaitingChars to know if there is any unread input waiting in the port's buffer.
If asyncInput is set to true then the CommPort object itself polls the port to know when data has arrived and if so then it executes an onReceive event. Inside that event the user can use the Receive procedure to get the data, which caused the execution of onReceive event. In such a case the Receive procedure never waits for its input and it ignores the required number of chars to read (if given). Note that onReceive events need not be executed each time when one single character arrives. One onReceive event may signal the arrival of a bigger chunk of data.
The default value is false.

connected

It is true if the port is connected otherwise it is false.
The typical use is that the user creates a CommPort descendant, then sets all the characteristics of the port (or they may be contained in the NEW command), and then sets its connected setting to true.
If the setting is false even after setting it to true then it means that the port was not able to connect (the Logo program can react by an error message if appropriate).
The default value is False.

Procedures

send ***text***

(send ***text timeout*****)**

Sends the *text* to the port. *Text* can be any Logo value, which is converted to text in the same way as the print command does. If it is used with one input then it is a command (gives no output value). If the output port does not accept the characters then the send command waits infinitely until the port becomes ready. The Logo process, which is running such a waiting send command, can be stopped by the usual means (like the stop button on the toolbar or by a cancel command). I.e. the user can abort the waiting if in error cases.

The optional *timeout* may give a timeout value in milliseconds. If the port doe not accept all the characters during the timeout then the procedure exits prematurely. If the optional timeout input is used then the procedure is an operation and its output is the number of characters, which were actually sent.

receive

(receive *number*)
(receive *number timeout*)

It is an operation, which receives characters from the port. Its output is always a word consisting of received characters.
The optional input *number* defines the number of characters to read.
If no *number* is given then it is assumed to be 1.
If the characters are already available in the port's input buffer, then receive returns immediately. Otherwise it waits until the desired number of characters arrive. If there is no *timeout* input given then the waiting time is infinite. If there is a *timeout* given (in milliseconds) then the receive command returns after the timeout has elapsed even if there were less than *number* characters read.
Note that during a long waiting in receive command the process containing the command can be stopped by the usual means (like the stop button on the toolbar or by a cancel command). I.e. the user can abort the waiting if in error cases.
If the receive procedure is called during an onReceive event's execution then the *number* input is ignored and the output of receive is always formed using all characters received from the last call to receive procedure. In such a case the receive command never waits for input characters, so the *timeout* input is ignored, too.
When asyncInput is set to true then the program should not call the receive procedure in situations when it was not notified by the onReceive event that there are characters available. The safest rule to keep to is not to call the *receive* procedure outside the onReceive event.

purgeInput

Purges the input queue of the port. It means that all characters, which were received by the operating system but not by the Logo program will be discarded and a pending receive procedure (if any) will be finished immediately.

purgeOutput

Purges the output queue of the port. It means that all characters, which were sent by the send procedure from the Logo program to the operating system but the operating system did not pass them to the physical port will be discarded and a pending send procedure (if any) will be finished immediately.

waitingChars

An operation, which result is the number of characters in the input queue of the port, i.e. the number of characters, which are "waiting" for the receive procedure.

Events

onReceive

The event is triggered if the setting asyncInput is set to true and some input characters arrive. It is not triggered for each individual character, if the input comes quickly then it may be triggered once for several characters received. The actual characters

received can be retrieved by a call to the procedure receive. Its result gives all received characters in one word. Note that in this situation the inputs to receive procedure are ignored.