The JRTalk extension.

The Dynamic Link Library, JRTalk.dll, located in the Windows\System directory regulates the communication between Serf or XL-Plot and a program that you may have created in Visual Basic or in C. It also gives access to a part of the Serf/XL-Plot math library. When calling one of the functions in the list below, the dll verifies if a copy of the program is already running. If not it launches the program before executing the function. The dll supposes that the executable is in the default directory: 'C:\serf' or 'C:\xlplot'. If the executable is in an other directory you need either to add a PATH command in the C:\autoexec.bat file or run the **S_SetProgramDirectory(**"some other directory"**)** function each time your program starts.

Most of the functions return a 32 bit integer (int in C, long in VBasic) usually containing the ID of the document (spreadsheet or drawing sheet). If a negative value is returned, an error condition has been encountered. Calling **S_Error(**negative integer**)** will give details of the error in question.

Contrary to Serf and XL-Plot, which are freeware, the JRTalk add-on is shareware. You are granted trial period, after which the dll needs to be registered. The dll's registration number is required for registration. Your registration number can be obtained by running S_Error(-99). The licence will be valid for a single installation on a computer. Future updates will be free. See the licence agreement for details.

In a **Visual Basic** program, each dll function has to be declared before it can be used. This can be done by using the ordinal number of the function as an alias as in:
Declare Function S_ColsToRows Lib "JRTalk" Alias "#2" (ByVal clos As Integer) As Long ,
or by using the visual basic alias in the list above:
Declare Function S_ColsToRows Lib "JRTalk" Alias "@S_ColsToRows$qqss" (ByVal clos As Integer) As Long.
The latter method is recommended, since the ordinal numbers in future versions of JRTalk.dll will almost certainly be different. The complete list of function names, aliases and ordinal numbers is to be found in the file JRTalk.def in the program directory. The list at the end of this document may be used to paste the list of **VB** function declarations in a program.
In **C**, the functions are called by their C names. To import the functions, create a header file with entries of the form: **long _import _stdcall** S_ColsToRows(short clos);. For compilation with Borland C++, a library file, 'JRTalk.lib', is available in the program directory.

Visual Basic alias	**C name**
@S_ClearClipboard$qqsv	int S_ClearClipboard()
@S_ColsToRows$qqss	int S_ColsToRows(short)
@S_CreatePlot$qqsv	int S_CreatePlot()
@S_Error$qqsl	int S_Error(long)
@S_GetCellFloat$qqsrfss	int S_GetCellFloat(float&,short,short)
@S_GetCellText$qqspcss	int S_GetCellText(char*,short,short)
@S_InsertColumn$qqspfsspc	int S_InsertColumn(float*,short,short,char*)
@S_MenuItem$qqss	int S_MenuItem(short)
@S_NewDrawSheet$qqsv	int S_NewDrawSheet()
@S_NewSpreadSheet$qqsv	int S_NewSpreadSheet()
@S_SelectCell$qqsss	int S_SelectCell(short,short)
@S_SelectColumns$qqsss	int S_SelectColumns(short,short)
@S_SelectRange$qqsssss	int S_SelectRange(short,short,short,short)
@S_SelectRows$qqsss	int S_SelectRows(short,short)
@S_SetAsEColumn$qqss	int S_SetAsEColumn(short)
@S_SetAsXColumn$qqss	int S_SetAsXColumn(short)
@S_SetAsYColumn$qqss	int S_SetAsYColumn(short)
@S_SetDocument$qqsi	int S_SetDocument(int)
@S_SetProgramDirectory$qqspc	int S_SetProgramDirectory(char*)
@S_SortColumns$qqsssss	int S_SortColumns(short,short,short,short)
@S_TextAtCell$qqspcss	int S_TextAtCell(char*,short,short)
@S_XY2Sheet$qqspft1spct4	int S_XY2Sheet(float*,float*,short,char*,char*)
@S_XYPlot$qqspft1spct4	int S_XYPlot(float*,float*,short,char*,char*)
@S_Y2Sheet$qqspfspc	int S_Y2Sheet(float*,short,char*)

C: int S_SetProgramDirectory(char* text)

VB: S_SetProgramDirectory(ByVal text As String) As Long

Sets the directory containing serf.exe or XL-Plot.exe. On success, it returns 0, on error it returns a negative value.
Example:
ret = S_SetProgramDirectory("C:\Programs\serf")

C: int S_Error(short err)
VB: S_Error(ByVal err As Integer) As Long

This function shows a messagebox containing an error message if the argument 'err' is negative. It returns the value 0 (zero).

C: int S_NewDrawSheet()
VB: S_NewDrawSheet() As Long

Creates a new (empty) drawing sheet and returns a positive document ID or a negative error number.

C: int S_NewSpreadSheet()
VB: S_NewSpreadSheet() As Long

Creates a new (empty) spreadsheet and returns a positive document ID or a negative error number.

C: int S_XYPlot(float *x,float *y,short len,char* xs,char* ys)
VB: S_XYPlot (x As Single, y As Single, ByVal ilen As Integer, ByVal xs As String, ByVal ys As String) As Long

This function takes two floating point arrays, x and y, of length len and two null terminated ASCII strings of text. It creates a XYplot on the current drawing sheet with the xstring (xs) and ystring (ys) as axis units. If no drawing sheet is available or if the current drawing sheet already contains 6 plots, it will create a new one. On success, it returns the ID of the drawing sheet, on error it returns a negative value.
Example:
id = S_XYPlot(x(0), y(0), 50, "s", "nM")
S_Error (id)

C: int S_XY2Sheet(float *x,float *y,short len,char* xs,char* ys)
VB: S_XY2Sheet (x As Single, y As Single, ByVal ilen As Integer, ByVal xs As String, ByVal ys As String) As Long

This function takes two floating point arrays, x and y, of length ilen and two null terminated ASCII strings of text. It creates a **new** spreadsheet with two columns (1 and 2) of data. The top most cells of the first and second columns will contain the xstring (xs) and ystring (ys) respectively. On success, it returns the ID of the spreadsheet, on error it returns a negative value.

C: int S_Y2Sheet(float *y,short len,char *ystring)
VB: S_Y2Sheet (y As Single, ByVal ilen As Integer, ByVal text As String) As Long

This function takes a floating point array, y, of length ilen and a null terminated ASCII string of text. It **adds** a new column of data to the current spreadsheet. The top most cell of the column will contain the ystring. On success, it returns the ID of the spreadsheet, on error it returns a negative value.
Example:
id = S_Y2Sheet(y(0), 100, "mV")

C: int S_InsertColumn(float *y,short len,short col,char *string)
VB: S_InsertColumn (y As Single, ByVal ilen As Integer, ByVal col As Integer, ByVal text As String) As Long

This function takes a floating point array, y, of length ilen and a null terminated ASCII string of text. It **inserts** a new column of data at column col of the current spreadsheet. The original column and those following it will be shifted to the right. The top most cell of the column will contain the ystring. On success, it returns the ID of the spreadsheet, on error it returns a negative value.

C: int S_CreatePlot()
VB: S_CreatePlot() As Long

This function creates a new plot using the currently selected spreadsheet columns. If no drawing sheet is available or if the current drawing sheet contains already 6 plots, a new drawing sheet will be created. On success, it returns the ID of the drawing sheet, on error it returns a negative value.

Prior to using the S_CreatePlot() instruction, at least 1 column of numerical data in the current spreadsheet needs to be selected using one of the following functions:

C: int S_SelectColumns(short col1,short col2)
VB: S_SelectColumns (ByVal col1 As Integer, ByVal col2 As Integer) As Long

The columns col1 through col2 will be selected. If only one column is selected (col1=col2) before issuing the S_CreatePlot() instruction, the column is considered to contain y data and the x-axis will run from 1 through N, where N is the number of entries in the column. If multiple columns are selected, the first will furnish the x coordinates, while the following columns are treated as y coordinates.

```
C:      int S_SetAsXColumn (short col)
        int S_SetAsYColumn (short col)
        int S_SetAsEColumn (short col)
VB:     S_SetAsXColumn (ByVal col As Integer) As Long
        S_SetAsYColumn (ByVal col As Integer) As Long
        S_SetAsEColumn (ByVal col As Integer) As Long
```

These three functions set column, col, as the column containing X data, Y data and Errors respectively. On success, they return the ID of the spreadsheet, on error they return a negative value. The data in the "Errors" column will be plotted as error bars in the new histogram resulting from a subsequent call to the S_CreatePlot() function.
Example:
```
ss_id = S_SetAsXColumn (5)
ss_id = S_SetAsYColumn (2)
ds_id = S_CreatePlot()
```

Functions to select spreadsheet cells, to change and to retrieve their contents are:

C: int S_SelectRows(short row1,short row2)
VB: S_SelectRows (ByVal row1 As Integer, ByVal row2 As Integer) As Long

This function selects rows 'row1' through 'row2'. On success, it returns the ID of the spreadsheet, on error it returns a negative value.

C: int S_SelectCell(short col,short row)
VB: S_SelectCell (ByVal col As Integer, ByVal row As Integer) As Long

This function selects the spreadsheet cell at column 'col' and row 'row'. On success, it returns the ID of the spreadsheet, on error it returns a negative value.

C: int S_SelectRange(short col1,short row1,short col2,short row2)
VB: S_SelectRange (ByVal col1 As Integer, ByVal row1 As Integer, ByVal col2 As Integer, ByVal row2 As Integer) As Long

This function selects the range of cells from [col1,row1] through [col2,row2]. On success, it returns the ID of the spreadsheet, on error it returns a negative value.

C: int S_TextAtCell(char* text,short col,short row)
VB: S_TextAtCell (ByVal text As String, ByVal col As Integer, ByVal row As Integer) As Long

Sets the contents of the spreadsheet cell at column 'col' and row 'row'. On success, it returns the ID of the spreadsheet, on error it returns a negative value. To enter a floating-point number in a spreadsheet cell proceed as in the following
Example:
ret = S_TextAtCell("17.45",2,3)

C: int S_GetCellText(char* text,short col,short row)
VB: S_GetCellText (ByVal text As String, ByVal col As Integer, ByVal row As Integer) As Long

Retrieves the contents of the spreadsheet cell at column 'col' and row 'row' as a character string. The string 'text' needs to have a length of at least 256 bytes. In VB it has to have been dimensioned as a fixed-length string. On success, the function returns the ID of the spreadsheet, on error it returns a negative value.
Example:
Dim result As String *256
ret = S_GetCellText(result,3,4)
upon return the string 'result' may contain a formula, e.g. "=sqrt(4)", plain text or a number, e.g. "3.41"

C: int S_GetCellFloat(float& x,short col,short row)
VB: S_GetCellFloat (x As Single, ByVal col As Integer, ByVal row As Integer) As Long

Retrieves the floating point number located at cell [col,row] and stores it in the variable x that is passed by reference. If the spreadsheet cell is empty or does not contain a number or a formula, x will be 0 (zero). Hence, if the spreadsheet cell contains "=sqrt(4)", x will be 2. On success, the function returns the ID of the spreadsheet, on error it returns a negative value.
Example:
Dim result As Single
ret = S_GetCellFloat(result,3,4)

C: int S_SortColumns(short c1,short c2,short ckey,short ascend)
VB: S_SortColumns(ByVal c1 As Integer, ByVal c2 As Integer, ByVal ckey As Integer, ByVal ascend As Integer) As Long

This function sorts the numerical values contained in the spreadsheet column ckey in acending order if ascend is 1 and in decending order if ascend is 0. The entries in the columns c1 through c2 will be displaced amongst rows along with the entries in column ckey. On success, the function returns the ID of the spreadsheet, on error it returns a negative value.

C: int S_ColsToRows(short doclose)
VB: S_ColsToRows (ByVal doclose As Integer) As Long

Copies the spreadsheet cells contained in the currently selected range of cells while interchanging columns and rows. It subsequently pastes the data in a new spreadsheet. If doclose is 1, the original (donor) spreadsheet will be closed without saving. If doclose is 0, the donor spreadsheet will remain open.
Example:
ret = S_SelectRange (1, 1, 16, 10)
ret = S_ColsToRows(0)

C: int S_SetDocument(int id)
VB: S_SetDocument (ByVal id As Long) As Long

Sets the document (spreadsheet or drawing sheet) as the currently active one. The program keeps track of the active spreadsheet and the active drawing sheet seperately, so it it not necessary to reactivate the current drawing sheet after a manipulation on a spreadsheet for example. It takes the document's ID as an argument. On success, the function returns the ID of the document, on error it returns a negative value.

C: int S_ClearClipboard()
VB: S_ClearClipboard() As Long

Clears the clipboard and returns 0.

```
C:	int S_MenuItem(short command)
VB:	S_MenuItem (ByVal command As Integer) As Long
```

This function simulates the selection of an menu item from the document's menu bar. It takes a menu item identifier as argument. The following commands may be used:

name	**identifier**	**action**
CM_TXTCOPY	1211	Copies the selected range of spreadsheet cells to the clipboard
CM_TXTPASTE	1212	Pastes the clipboard onto the currently active spreadsheet
CM_TXTINSERT	1213	Inserts empty columns, rows or range of cells onto the currently active spreadsheet.
CM_TXTCLEAR	1215	Clears the currently selected range of spreadsheet cells.
CM_TXTDELETE	1216	Deletes the currently selected range of spreadsheet cells.
CM_TXTVCOPY	1289	Copies the current histogram to the clipboard.
CM_SETASX	1227	Sets the currently selected spreadsheet column as the X column.
CM_SETASY	1228	Sets the currently selected spreadsheet column as the Y column.
CM_SETASERROR	1229	Sets the currently selected spreadsheet column as the error column.
CM_REMLINKS	1268	Removes the links between spreadsheet and drawing sheet.
CM_INVMAT	1306	Carries out inversion of an augmented matrix.
CM_FILLITEM1	1276	Fill down Copy menu item.
CM_FILLITEM2	1277	Fill down Increment menu item.
CM_FILLITEM3	1278	Fill down Decrement menu item.
CM_FILLITEM4	1279	Fill down Interpolation menu item.
CM_FILLITEM10	1285	Fill down Value menu item.
CM_FILLITEM5	1280	Fill right Copy menu item.
CM_FILLITEM6	1281	Fill right Increment menu item.
CM_FILLITEM7	1282	Fill right Decrement menu item.
CM_FILLITEM8	1283	Fill right Interpolation menu item.
CM_FILLITEM11	1284	Fill right Value menu item.
CM_VECPASTE	1233	Pastes the clipboard onto the currently active drawing sheet.

The following codes (may) require user intervention (dialogue box)

CM_LINEPLOT	1261	menu item: Modify/Data>Line Plot
CM_BARPLOT	1260	menu item: Modify/Data>Bar Plot
CM_CREAPOLYG	1336	menu item: Modify/Data>Create polygon
CM_GETSTATS	1315	menu item: Modify/Data>Get Column Stats
CM_RESIZE	1300	menu item: Modify/Data>Resize column/line length
CM_FIT	1110	menu item: Modify/Data>Do Fit
CM_SETFITFUNC	1109	menu item: Modify/Data>Set Fit Function
CM_SETCOLW	1220	menu item: Modify/Data>Set Column Widths
CM_SETLINH	1221	menu item: Modify/Data>Set Line Heights
CM_FOURIER	1318	menu item: Modify/Data>Fourier transform
CM_TXTCONVOLVE	1325	menu item: Modify/Data>(de-)Convolution
CM_TXTCOPY	1211	menu item: Edit>Copy
CM_TXTVCOPY	1289	menu item: Edit>Copy values
CM_COPYLC	1364	menu item: Edit>Copy L <--> C
CM_TXTPASTE	1212	menu item: Edit>Paste
CM_TXTINSERT	1213	menu item: Edit>Insert
CM_TXTPANDI	1214	menu item: Edit>Insert/Paste
CM_TXTCLEAR	1215	menu item: Edit>Clear
CM_TXTDELETE	1216	menu item: Edit>Delete
CM_REPLACE	1290	menu item: Edit>Replace

The math library

Visual Basic alias	**C name**
@S_Binom$qqsii	float S_Binom(int,int)
@S_Correl$qqspft1i	float S_Correl(float*,float*,int)

@S_Cov$qqspft1	float S_Cov(float*,float*,int)
@S_Erfc$qqsf	float S_Erfc(float)
@S_Extrapol$qqspfif4bool	float S_Extrapol(float*,int,float,bool)
@S_FDens$qqsfii	float S_FDens(float,int,int)
@S_FT$qqspft1is	short S_FT(float*,float*,int,short)
@S_Gamma$qqsf	float S_Gamma(float)
@S_InvertMatrix$qqspdii	short S_InvertMatrix(double*,int,int)
@S_Mean$qqspfi	float S_Mean(float*,int)
@S_PChi$qqsfi	float S_PChi(float,int)
@S_PFTest$qqsfii	float S_PFTest(float,int,int)
@S_PTTest$qqsfii	float S_PTTest(float,int,int)
@S_PolyInterpol$qqspfiif	float S_PolyInterpol(float*,int,int,float)
@S_PolyRegres$qqspfiit1	short S_PolyRegres(float*,int,int,float*)
@S_QProb$qqsffff	float S_QProb(float,float,float,float)
@S_Random$qqsf	float S_Random(float)
@S_Reshuffle$qqspfipi	short S_Reshuffle(float*,int,int*)
@S_SortFloat$qqspfipi	short S_SortFloat(float*,int,int*)
@S_TDens$qqsfi	short S_TDens(float,int)
@S_Zero$qqsf	short S_Zero(float)
@S_copyf$qqspft1i	short S_copyf(float*,float*,int)
@S_copyfs$qqspspfi	short S_copyfs(short*,float*,int)
@S_copyi$qqspit1i	short S_copyi(int*,int*,int)
@S_copys$qqspst1i	short S_copys(short*,short*,int)
@S_copysf$qqspfpsi	short S_copysf(float*,short*,int)
@S_integrate$qqspfi	short S_integrate(float*,int)
@S_swapff$qqspft1i	short S_swapff(float*,float*,int)
@S_swapss$qqspst1i	short S_swapss(short*,short*,int)
@S_vadds$qqspffi	short S_vadds(float*,float,int)
@S_vaddv$qqspft1i	short S_vaddv(float*,float*,int)
@S_vadfs$qqspfpsi	short S_vadfs(float*,short*,int)
@S_vadsf$qqspspfi	short S_vadsf(short*,float*,int)
@S_vadss$qqspssi	short S_vadss(short*,short,int)
@S_vadsv$qqspst1i	short S_vadsv(short*,short*,int)
@S_vdivv$qqspft1i	short S_vdivv(float*,float*,int)
@S_vinv$qqspfi	short S_vinv(float*,int)
@S_vinvs$qqspsi	short S_vinvs(short*,int)
@S_vmax$qqspfi	int S_vmax(float*,int)
@S_vmaxs$qqspsi	int S_vmaxs(short*,int)
@S_vmin$qqspfi	int S_vmin(float*,int)
@S_vmins$qqspsi	int S_vmins(short*,int)
@S_vmuls$qqspffi	short S_vmuls(float*,float,int)
@S_vmulv$qqspft1i	short S_vmulv(float*,float*,int)
@S_vsets$qqspffi	short S_vsets(float*,float,int)
@S_vsetsi$qqspiii	short S_vsetsi(int*,int,int)
@S_vsetss$qqspssi	short S_vsetss(short*,short,int)
@S_vsqr$qqspfi	short S_vsqr(float*,int)
@S_vsubv$qqspft1i	short S_vsubv(float*,float*,int)

Statistical functions

C:	float S_Binom(int n,int k)
VB:	S_Binom(ByVal n As Long, ByVal k As Long) As Single
Returns the coefficient of the k'th term of an n-binomial.

C:	float S_Gamma(float x)
VB:	S_Gamma(ByVal x As Single) As Single
Returns gamma(x). Gamma has the properties: gamma(n+1)=n! and x*gamma(x) = gamma(x+1).

C:	float S_TDens(float x,int d)
VB:	S_TDens(ByVal x As Single, ByVal d As Long) As Single

Returns the double sided probability-density of the t-function for t=x at d degrees of freedom.

C: float S_PTTest(float t,int d,int s)
VB: S_PTTes(ByVal t As Single, ByVal d As Long, ByVal s As Long) As Single
Returns the probability of a t-value at d degrees of freedom, single sided (s=1) or double sided (s=2). For s=2, it is the integral of TDens(t,df).

C: float S_FDens(float x,int d1,int d2)
VB: S_FDens(ByVal x As Single, ByVal d1 As Long, ByVal d2 As Long) As Single
Returns the probability density of Fisher's F-function for x at d1 (data sets) and d2 (total N-1) degrees of freedom.

C: float S_PFTest(float x,int d1,int d2)
VB: S_PFTes(ByVal x As Single, ByVal d1 As Long, ByVal d2 As Long) As Single
Returns the probability of a F-value (x) at d1 and d2 degrees of freedom. It is the integral of FDens(x,df1,df2).

C: float S_QProb(float q,float rr,float cc,float df)
VB: S_QProb(ByVal q As Single, ByVal rr As Single, ByVal cc As Single, ByVal df As Single) As Single
Returns the probability for a q-value at df1 and df2 degrees of freedom (multiple comparison test).

C: float S_PChi(float chisqr,int df)
VB: S_PChi(ByVal chisqr As Single, ByVal df As Long) As Single
Returns the probability of a chi-2 at df degrees of freedom.

C: float S_Erfc(float y)
VB: S_Erfc(ByVal y As Single) As Single
Returns the complementary error function (1-erf(y)), where erf(y) is error function of y. erf(y) is twice the integral of the Gaussian distribution with 0 mean and variance of ½.

C: float S_Random(float max)
VB: S_Random(ByVal max As Single) As Single
Returns a random value between 0 and max.

C: float Cov(float *x,float *y,int npts)
VB: S_Cov(x As Single, y As Single, ByVal npts As Long) As Single
Returns the covariance of the data in the floating point arrays x and y of lenght npts.

C: float S_Correl(float *x,float *y,int npts)
VB: S_Correl(x As Single, y As Single, ByVal npts As Long) As Single
Returns the correlation coefficient of the data in the floating point arrays x and y of length npts.

Floating point functions

C: short S_copyf(float *x, float *y, int len)
VB: S_copyf(x As Single, y As Single, ByVal len As Long) As Integer
Copies the contents of the floating point array, y, of lenght len to array x. The function returns 0.

C: short S_swapff(float *x, float *y, int len)
VB: S_swapff(x As Single, y As Single, ByVal len As Long) As Integer
Swaps the contents of the floating point arrays x and y. Each array has a length of at least len. The function returns 0.

C: short S_vsets(float *x, float fac, int n)
VB: S_vsets(x As Single, ByVal fac As Single, ByVal n As Long) As Integer
Sets the first n elements of the floating point array, x, to the scalar value fac. The function returns 0.

C: short S_vadds(float *x, float fac, int n)
VB: S_vadds(x As Single, ByVal fac As Single, ByVal n As Long) As Integer

Adds the scalar value fac to the first n elements of the floating point array x. The function returns 0.

C: short S_vaddv(float *x, float *y, int n)
VB: S_vaddv(x As Single, y As Single, ByVal n As Long) As Integer
Replaces the first n elements of x by the sum of the elements of the two floating point arrays, x and y. Hence, x(i)=x(i)+y(i) for i=0 through n-1. The function returns 0.

C: short S_vsubv(float *x, float *y, int n)
VB: S_vsubv(x As Single, ByVal y As Single, ByVal n As Long) As Integer
Replaces the first n elements of x by the difference of the elements of the two floating point arrays, x and y. Hence, x(i)=x(i)-y(i) for i=0 through n-1. The function returns 0.

C: short S_vmuls(float *x, float fac, int n)
VB: S_vmuls(x As Single, ByVal fac As Single, ByVal n As Long) As Integer
Multiplies the first n elements of the floating point array, x, by the scalar fac. The function returns 0.

C: short S_vmulv(float *x, float *y, int n)
VB: S_vmulv(x As Single, y As Single, ByVal n As Long) As Integer
Replaces the first n elements of x by the product of the elements of the two floating point arrays, x and y. Hence, x(i)=x(i)*y(i) for i=0 through n-1. The function returns 0.

C: short S_vdivv(float *x, float *y, int n)
VB: S_vdivv(x As Single, y As Single, ByVal n As Long) As Integer
Replaces the first n elements of x by the quotient of the elements of the two floating point arrays, x and y. Hence, x(i)=x(i)/y(i) for i=0 through n-1. The function returns 0.

C: short S_vsqr(float *x, int n)
VB: S_vsqr(x As Single, ByVal n As Long) As Integer
Takes the square root of each element of the floating point array of length n. The function returns 0.

C: short S_vinv(float *x, int n)
VB: S_vinv(x As Single, ByVal n As Long) As Integer
Replaces the first n elements of x by their square roots. Hence, x(i)=sqrt(x(i)) for i=0 through n-1. The function returns 0.

C: short S_integrate(float *x, int n)
VB: S_integrate(x As Single, ByVal n As Long) As Integer
Carries out numerical integration of the first n elements of the array x. The function returns 0.

C: int S_vmin(float *x, int len)
VB: S_vmin(x As Single, ByVal n As Long) As Integer
Returns the index of the element of the floating point array, x, of length len that containins the minimum value. To obtain the minimum value itself write: min = x(S_vmin(x,len)).

C: int S_vmax(float *x, int len)
VB: S_vmax(x As Single, ByVal n As Long) As Integer
Returns the index of the element of the floating point array, x, of length len that containins the maximum value. To obtain the maximum value itself write: max = x(S_vmax(x,len)).

C: float S_Mean(float *x,int len)
VB: S_Mean(x As Single, ByVal npts As Long) As Single
Returns the mean value of the floating point array, x, of length len.

C: short S_Zero(float x)
VB: S_Zero(ByVal x As Single) As Integer
Returns 1 if x is between -1e-15 and +1e-15. Returns 0 otherwise.

C: short S_SortFloat(float *x,int len,int *indx)
VB: S_SortFloat(x As Single, ByVal len As Long, indx As Long) As Integer

Sorts the elements of the floating point array, x, in ascending order. A list of indices is maintained in the integer array indx for subsequent use with the S_Reshuffle() function. If the pointer to indx equals NULL, the index array is ignored. The function returns 0.

C: short S_Reshuffle(float *y,int len,int *indx)
VB: S_Reshuffle(y As Single, ByVal len As Long, indx As Long) As Integer
Sorts the contents of the floating point array, y, in an order that is determined by the indx array. If thecontents of the indx array have been obtained by a previous call to S_SortFloat(), then the elementsof y are moved as the elements of x in the call to S_SortFloat(x,len,indx). The function returns 0.
Example:
```
Dim x(4) As Single, y(4) As Single, indx(4) As Long
x(0)=0          x(1)=3  x(2)=2  x(3)=1
y(0)=8  y(1)=5  y(2)=6  y(3)=7
ret= S_SortFloat(x(0),4,indx(0))
ret= S_Reshuffle(y(0),4,indx(0))
result: x: 0 1 2 3         y: 8  7  6  5
```

C: short S_FT(float* real,float* imag,int np,short inverse)
VB: S_FT(real As Single, imag As Single, ByVal np As Long, ByVal inverse As Integer) As Integer
Takes the Fourier transform of the floating point arrays real and imag of length np. If inverse=0 the forward transformation is carried out, if it is 1, the inverse transform is carried out. If np equals a power of 2, then the FFT algorithm is used. The contents of the arrays real and imag are replaced by the transform.

C: short S_InvertMatrix(double* mat,int nmat,int dim)
VB: S_InvertMatrix(mat As Double, ByVal nmat As Long, ByVal mdim As Long) As Integer
Inverts the augmented matrix, mat. mat is a nmat*nmat square matrix, augmented with a column of dimension nmat. The matrix, mat, is passed as a linear double precision array. In this linear array ndim elements are reserved for each row. ndim should be at least nmat+1.
Example:
invert the 3*3 matrix augmented with the vector [5,6,5]:
```
        2  3  -1   5
        1  1   1   6
        0  -2  3   5
Dim M(12) as Double
M(0)=2    M(1)=3    M(2)=-1    M(4)=5   M(6)=1 ...... M(10)=3   M(11)=5
ret= S_InvertMatrix(M(0),3,4)
```

C: float S_Extrapol(float *x,int num,float where,short log)
VB: S_Extrapol(x As Single, ByVal num As Long, ByVal where As Single, ByVal log As Integer) As Single
This function carries out linear (log=0) or logarithmic (log=1) regression using the data in the floating point array of length num in order to estimate the ordinate at abscis 'where'. It returns the estimate.

C: short S_PolyRegres(float *x,int num,int degree,float *result)
VB: S_PolyRegres(x As Single, ByVal num As Long, ByVal degree As Long, result As Single) As Integer
This function calculates the coefficients of a polynomial of degree 'degree' that fits best the data in the floating point array x of length num. The coefficients are stored in the array 'result' that should have a length of at least degree+1. The function returns 0.

C: float S_PolyInterpol(float *x,int num,int degree,float where)
VB: S_PolyInterpol(x As Single, ByVal num As Long, ByVal degree As Long, where As Single) As Single
This function returns the estimate of the ordinate at abscis 'where' by passing a polynomial of degree 'degree' through the data in the floating point array x of length num.

short data functions

C: short S_copys(short *x, short *y, int n)
VB: S_copys(x As Integer, y As Integer, ByVal n As Long) As Integer
Copies the contents of the 16 bit integer array, y, to the 16 bit integer array, x, of length n. The function returns 0.

C:	short S_vsetss(short *x, short fac, int n)
VB:	S_vsetss(x As Integer, ByVal fac As Integer, ByVal n As Long) As Integer
Sets the contents of the 16 bit integer array, x, of length n to the scalar 'fac'. The function returns 0.

C:	short S_vinvs(short *x, int n)
VB:	S_vinvs(x As Integer, ByVal n As Long) As Integer
Inverts the contents of the 16 bit integer array x of length n. The function returns 0.

C:	short S_vadss(short *x, short fac, int n)
VB:	S_vadss(x As Integer, ByVal fac As Integer, ByVal n As Long) As Integer
Adds the scalar 'fac' to each element of the 16 bit integer array, x, of length n. The function returns 0.

C:	short S_vadsv(short *x, short *y, int n)
VB:	S_vadsv(x As Integer, y As Integer, ByVal n As Long) As Integer
Adds, element by element, the contents of the 16 bit integer arrays, x and y of lengths n. The result is stored in x. The function returns 0.

C:	short S_swapss(short *x, short *y, int n)
VB:	S_swapss(x As Integer, y As Integer, ByVal n As Long) As Integer
Interchanges the contents of the 16 bit integer arrays x and y of lengths n. The function returns 0.

C:	int S_vmins(short *x, int n)
VB:	S_vmins(x As Integer, ByVal n As Long) As Integer
The function returns the index of element containing the minimum value of the 16 bit integer array x of length n. To obtain the minimum value itself write: ret=x(S_vmins(x,n)).

C:	int S_vmaxs(short *x, int n)
VB:	S_vmaxs(x As Integer, ByVal n As Long) As Integer
The function returns the index of element containing the maximum value of the 16 bit integer array x of length n. To obtain the maximum value itself write: ret=x(S_vmaxs(x,n)).

integer data functions

C:	short S_copyi(int *x, int *y, int n)
VB:	S_copyi(x As Long, y As Long, ByVal n As Long) As Integer
Copies the contents of the 32 bit integer array y of length n to the 32 bit integer array x. The function returns 0.

C:	short S_vsetsi(int *x, int fac, int n)
VB:	S_vsetsi(x As Long, ByVal fac As Long, ByVal n As Long) As Integer
Sets the contents of the 32 bit integer array x of length n to the scalar 'fac'. The function returns 0.

mixed format functions

C:	short S_copysf(float *x, short *y, int n)
VB:	S_copysf(x As Single, y As Integer, ByVal n As Long) As Integer
Copies the contents of the 16 bit integer array of length n to the floating point array x. The function returns 0.

C:	short S_copyfs(short *x, float *y, int n)
VB:	S_copyfs(x As Integer, y As Single, ByVal n As Long) As Integer
Copies the contents of the floating point array y of length n to the 16 bit integer array x. The function returns 0.

C:	short S_vadfs(float *x, short *y, int n)
VB:	S_vadfs(x As Single, y As Integer, ByVal n As Long) As Integer
Adds the contents of the 16 bit integer array y of length n to the floating point array x. The function returns 0.

C:	short S_vadsf(short *x, float *y, int n)
VB:	S_vadsf(x As Integer, y As Single, ByVal n As Long) As Integer

Adds the contents of the floating point array y of length n to the 16 bit integer array x. The function returns 0.

Serial port communication (RS232)

The following routines implement simple serial communication without handshaking. The routines all return 0 unless an error occurs (a negativenumber).

Visual Basic alias	**C name**
@S_SendCOM$qqsspc	int S_SendCOM(short,char*)
@S_SetCOM$qqsssss	int S_SetCOM(short,short,short,short)
@S_SetEndStr$qqspc	int S_SetEndStr(char*)
@S_WaitForReplyCOM$qqsspc	int S_WaitForReplyCOM(short,char*)
@S_WriteCOM$qqsspcs	int S_WriteCOM(short,char*,short)

C: int S_SetCOM(short baudrate,short parity,short stopbits,short bytesize)
VB: S_SetCOM(baudrate As Integer, parity As Integer, stopbits As Integer, bytesize As Integer) As Long

Sets the serial communication parameters to baudrate,parity,stopbits and bytesize. The parameter stopbits may take the values 0,1 and 2 for one, one and a half and two stopbits respectively.

C: int S_SetEndStr(char* endstr)
VB: S_SetEndStr(ByVal endstr As String) As Long

Sets the string of characters that signals the end of the transmitted sequence (often a line feed character, Hex(10) or Hex(13), is used).

C: S_SendCOM(short com,char* Buffer)
VB: S_SendCOM(ByVal com As Integer, ByVal buffer As String) As Long

Sends a the contents of a "0" terminated string, Buffer, to com port number "com" and waits for a reply that should contain at least the end-of-message character string as defined in S_SetEndStr. Buffer should be large enough to contain the outgoing and incoming messages. The reply will replace the contents of Buffer. If no reply arrives within 2 seconds, error -94 will be returned. The short integer parameter "com" can take the values 1 through 4 (for COM1 through COM4). The routine returns a negative value upon error.
example:
```
Dim command As String * 64
command = Chr$(10)
ret = S_EndStr(command)
command = "Oh sole mio " + Chr$(10)
ret = S_SetCOM(9600, 0, 2, 8)          ' baudrate 9600, no parity, 2 stopbits, bytesize 8
ret = S_SendCOM(2, command)            ' send something useless to COM2 and wait for reply
```

C: int S_WriteCOM(short com, char* buffer, short wait)
VB: S_WriteCOM (ByVal com As Integer, ByVal command As String, ByVal wait As Integer) As Long

This function opens the com port "com" and writes the null terminated string. If "wait" equals 1, the port remains open and is supposed to be closed by a subsequent call to S_WaitForReplyCOM. If "wait" equals 0, the port will be closed and any reply will be ignored. The short integer parameter "com" can take the values 1 through 4 (for COM1 through COM4). The routine returns a negative value upon error. This function may be used either to write something to a serial port without expecting a reply or to write first something to the serial port, handle some other tasks and then check for a reply. The latter is useful to control slow devices, so the program has not to idle while waiting for a reply.

C: int S_WaitForReplyCOM (short com, char* buffer)
VB: S_WaitForReplyCOM(ByVal com As Integer, ByVal buffer As String) As Long

This function is used in conjunction with the S_WriteCOM function. After the serial port "com" has been opened by S_WriteCOM it waits for a reply. The char string "buffer" should be large enough to hold the reply, which should be terminated by the "endstr" as defined by a call to S_SetEndStr. The short integer parameter "com" can take the values 1 through 4 (for COM1 through COM4). The routine returns a negative value upon error.

example:

```
Dim command As String * 64
command = Chr$(13)
ret = S_EndStr(command)
command = "something" + Chr$(13)
ret = S_SetCOM(9600, 0, 2, 8)           ' baudrate 9600, no parity, 2 stopbits, bytesize 8
ret = S_WriteCOM(2,command,1)           ' send something to COM2
ret = some_other_function()             ' do something else
ret = S_WaitForReplyCOM (2, command)    ' wait for reply
```

Visual Basic function decarations

```
Declare Function S_Binom Lib "JRTalk" Alias "@S_Binom$qqsii" (ByVal n As Long, ByVal k As Long) As Single
Declare Function S_ClearClipboard Lib "JRTalk" Alias "@S_ClearClipboard$qqsv" () As Long
Declare Function S_ColsToRows Lib "JRTalk" Alias "@S_ColsToRows$qqss" (ByVal clos As Integer) As Long
Declare Function S_copyf Lib "JRTalk" Alias "@S_copyf$qqspft1i" (x As Single, y As Single, ByVal n As Long) As
        Integer
Declare Function S_copyfs Lib "JRTalk" Alias "@S_copyfs$qqspspfi" (x As Integer, y As Single, ByVal n As Long)
        As Integer
Declare Function S_copyi Lib "JRTalk" Alias "@S_copyi$qqspit1i" (x As Long, y As Long, ByVal n As Long) As
        Integer
Declare Function S_copys Lib "JRTalk" Alias "@S_copys$qqspst1i" (x As Integer, y As Integer, ByVal n As Long)
        As Integer
Declare Function S_copysf Lib "JRTalk" Alias "@S_copysf$qqspfpsi" (x As Single, y As Integer, ByVal n As Long)
        As Integer
Declare Function S_Correl Lib "JRTalk" Alias "@S_Correl$qqspft1i" (x As Single, y As Single, ByVal npts As
        Long) As Single
Declare Function S_Cov Lib "JRTalk" Alias "@S_Cov$qqspft1i" (x As Single, y As Single, ByVal npts As Long) As
        Single
Declare Function S_CreatePlot Lib "JRTalk" Alias "@S_CreatePlot$qqsv" () As Long
Declare Function S_EndStr Lib "JRTalk" Alias "@S_SetEndStr$qqspc" (ByVal endstr As String) As Long
Declare Function S_Erfc Lib "JRTalk" Alias "@S_Erfc$qqsd" (ByVal y As Single) As Single
Declare Function S_Error Lib "JRTalk" Alias "@S_Error$qqsl" (ByVal err As Integer) As Long
Declare Function S_Extrapol Lib "JRTalk" Alias "@S_Extrapol$qqspfifs" (x As Single, ByVal num As Long, ByVal
        where As Single, ByVal log As Integer) As Single
Declare Function S_FDens Lib "JRTalk" Alias "@S_FDens$qqsfii" (ByVal x As Single, ByVal d1 As Long, ByVal
        d2 As Long) As Single
Declare Function S_FT Lib "JRTalk" Alias "@S_FT$qqspft1is" (real As Single, imag As Single, ByVal np As Long,
        ByVal inverse As Integer) As Integer
Declare Function S_Gamma Lib "JRTalk" Alias "@S_Gamma$qqsf" (ByVal x As Single) As Single
Declare Function S_GetCellFloat Lib "JRTalk" Alias "@S_GetCellFloat$qqsrfss" (x As Single, ByVal col As
        Integer, ByVal row As Integer) As Long
Declare Function S_GetCellText Lib "JRTalk" Alias "@S_GetCellText$qqspcss" (ByVal text As String, ByVal col
        As Integer, ByVal row As Integer) As Long
Declare Function S_InsertColumn Lib "JRTalk" Alias "@S_InsertColumn$qqspfsspc" (y As Single, ByVal ilen As
        Integer, ByVal col As Integer, Optional ByVal text As String = 0) As Long
Declare Function S_integrate Lib "JRTalk" Alias "@S_integrate$qqspfi" (x As Single, ByVal n As Long) As Integer
Declare Function S_InvertMatrix Lib "JRTalk" Alias "@S_InvertMatrix$qqspdii" (mat As Double, ByVal nmat As
        Long, ByVal mdim As Long) As Integer
Declare Function S_Mean Lib "JRTalk" Alias "@S_Mean$qqspfi" (x As Single, ByVal npts As Long) As Single
Declare Function S_MenuItem Lib "JRTalk" Alias "@S_MenuItem$qqss" (ByVal command As Integer) As Long
Declare Function S_NewDrawSheet Lib "JRTalk" Alias "@S_NewDrawSheet$qqsv" () As Long
Declare Function S_NewSpreadSheet Lib "JRTalk" Alias "@S_NewSpreadSheet$qqsv" () As Long
Declare Function S_PChi Lib "JRTalk" Alias "@S_PChi$qqsfi" (ByVal chisqr As Single, ByVal df As Long) As
        Single
```

```
Declare Function S_PFTest Lib "JRTalk" Alias "@S_PFTest$qqsfii" (ByVal x As Single, ByVal d1 As Long, ByVal
        d2 As Long) As Single
Declare Function S_PolyInterpol Lib "JRTalk" Alias "@S_PolyInterpol$qqspfiif" (x As Single, ByVal num As Long,
        ByVal degree As Long, where As Single) As Single
Declare Function S_PolyRegres Lib "JRTalk" Alias "@S_PolyRegres$qqspfiit1" (x As Single, ByVal num As Long,
        ByVal degree As Long, result As Single) As Integer
Declare Function S_PTTest Lib "JRTalk" Alias "@S_PTTest$qqsfii" (ByVal t As Single, ByVal d As Long, ByVal s
        As Long) As Single
Declare Function S_QProb Lib "JRTalk" Alias "@S_QProb$qqsffff" (ByVal q As Single, ByVal rr As Single, ByVal
        cc As Single, ByVal df As Single) As Single
Declare Function S_Random Lib "JRTalk" Alias "@S_Random$qqsf" (ByVal max As Single) As Single
Declare Function S_Reshuffle Lib "JRTalk" Alias "@S_Reshuffle$qqspfipi" (x As Single, ByVal n As Long, index
        As Long) As Integer
Declare Function S_SelectCell Lib "JRTalk" Alias "@S_SelectCell$qqsss" (ByVal col As Integer, ByVal row As
        Integer) As Long
Declare Function S_SelectColumns Lib "JRTalk" Alias "@S_SelectColumns$qqsss" (ByVal col1 As Integer, ByVal
        col2 As Integer) As Long
Declare Function S_SelectRange Lib "JRTalk" Alias "@S_SelectRange$qqsssss" (ByVal col1 As Integer, ByVal
        row1 As Integer, ByVal col2 As Integer, ByVal row2 As Integer) As Long
Declare Function S_SelectRows Lib "JRTalk" Alias "@S_SelectRows$qqsss" (ByVal row1 As Integer, ByVal row2
        As Integer) As Long
Declare Function S_SendCOM Lib "JRTalk" Alias "@S_SendCOM$qqsspc" (ByVal com As Integer, ByVal buffer
        As String) As Long
Declare Function S_SetAsEColumn Lib "JRTalk" Alias "@S_SetAsEColumn$qqss" (ByVal col As Integer) As Long
Declare Function S_SetAsXColumn Lib "JRTalk" Alias "@S_SetAsXColumn$qqss" (ByVal col As Integer) As Long
Declare Function S_SetAsYColumn Lib "JRTalk" Alias "@S_SetAsYColumn$qqss" (ByVal col As Integer) As Long
Declare Function S_SetCOM Lib "JRTalk" Alias "@S_SetCOM$qqsssss" (ByVal baud As Integer, ByVal parity As
        Integer, ByVal stopbits As Integer, ByVal bytesize As Integer) As Long
Declare Function S_SetDirectory Lib "JRTalk" Alias "@S_SetProgramDirectory$qqspc" (ByVal text As String) As
        Long
Declare Function S_SetDocument Lib "JRTalk" Alias "@S_SetDocument$qqsi" (ByVal id As Long) As Long
Declare Function S_SortColumns Lib "JRTalk" Alias "@S_SortColumns$qqsssss" (ByVal c1 As Integer, ByVal c2
        As Integer, ByVal ckey As Integer, ByVal ascend As Integer) As Long
Declare Function S_SortFloat Lib "JRTalk" Alias "@S_SortFloat$qqspfipi" (x As Single, ByVal n As Long, index As
        Long) As Integer
Declare Function S_swapff Lib "JRTalk" Alias "@S_swapff$qqspft1i" (x As Single, y As Single, ByVal n As Long)
        As Integer
Declare Function S_swapss Lib "JRTalk" Alias "@S_swapss$qqspst1i" (x As Integer, y As Integer, ByVal n As
        Long) As Integer
Declare Function S_TDens Lib "JRTalk" Alias "@S_TDens$qqsfi" (ByVal x As Single, ByVal d As Long) As Single
Declare Function S_TextAtCell Lib "JRTalk" Alias "@S_TextAtCell$qqspcss" (ByVal text As String, ByVal col As
        Integer, ByVal row As Integer) As Long
Declare Function S_vadds Lib "JRTalk" Alias "@S_vadds$qqspffi" (x As Single, ByVal fac As Single, ByVal n As
        Long) As Integer
Declare Function S_vaddv Lib "JRTalk" Alias "@S_vaddv$qqspft1i" (x As Single, y As Single, ByVal n As Long)
        As Integer
Declare Function S_vadfs Lib "JRTalk" Alias "@S_vadfs$qqspfpsi" (x As Single, y As Integer, ByVal n As Long)
        As Integer
Declare Function S_vadsf Lib "JRTalk" Alias "@S_vadsf$qqspspfi" (x As Integer, y As Single, ByVal n As Long)
        As Integer
Declare Function S_vadss Lib "JRTalk" Alias "@S_vadss$qqspssi" (x As Integer, ByVal fac As Integer, ByVal n
        As Long) As Integer
Declare Function S_vadsv Lib "JRTalk" Alias "@S_vadsv$qqspst1i" (x As Integer, y As Integer, ByVal n As Long)
        As Integer
Declare Function S_vdivv Lib "JRTalk" Alias "@S_vdivv$qqspft1i" (x As Single, y As Single, ByVal n As Long) As
        Integer
Declare Function S_vinv Lib "JRTalk" Alias "@S_vinv$qqspfi" (x As Single, ByVal n As Long) As Integer
Declare Function S_vinvs Lib "JRTalk" Alias "@S_vinvs$qqspsi" (x As Integer, ByVal n As Long) As Integer
Declare Function S_vmax Lib "JRTalk" Alias "@S_vmax$qqspfi" (x As Single, ByVal n As Long) As Integer
Declare Function S_vmaxs Lib "JRTalk" Alias "@S_vmaxs$qqspsi" (x As Integer, ByVal n As Long) As Integer
Declare Function S_vmin Lib "JRTalk" Alias "@S_vmin$qqspfi" (x As Single, ByVal n As Long) As Integer
Declare Function S_vmins Lib "JRTalk" Alias "@S_vmins$qqspsi" (x As Integer, ByVal n As Long) As Integer
Declare Function S_vmuls Lib "JRTalk" Alias "@S_vmuls$qqspffi" (x As Single, ByVal fac As Single, ByVal n As
        Long) As Integer
Declare Function S_vmulv Lib "JRTalk" Alias "@S_vmulv$qqspft1i" (x As Single, y As Single, ByVal n As Long)
        As Integer
Declare Function S_vsets Lib "JRTalk" Alias "@S_vsets$qqspffi" (x As Single, ByVal fac As Single, ByVal n As
        Long) As Integer
Declare Function S_vsetsi Lib "JRTalk" Alias "@S_vsetsi$qqspiii" (x As Long, ByVal fac As Long, ByVal n As
        Long) As Integer
Declare Function S_vsetss Lib "JRTalk" Alias "@S_vsetss$qqspssi" (x As Integer, ByVal fac As Integer, ByVal n
        As Long) As Integer
Declare Function S_vsqr Lib "JRTalk" Alias "@S_vsqr$qqspfi" (x As Single, ByVal n As Long) As Integer
Declare Function S_vsubv Lib "JRTalk" Alias "@S_vsubv$qqspft1i" (x As Single, ByVal y As Single, ByVal n As
        Long) As Integer
Declare Function S_XY2Sheet Lib "JRTalk" Alias "@S_XY2Sheet$qqspft1spct4" (x As Single, y As Single, ByVal
        ilen As Integer, Optional ByVal xs As String = 0, Optional ByVal ys As String = 0) As Long
Declare Function S_XYPlot Lib "JRTalk" Alias "@S_XYPlot$qqspft1spct4" (x As Single, y As Single, ByVal ilen As
        Integer, Optional ByVal xs As String = 0, Optional ByVal ys As String = 0) As Long
Declare Function S_Y2Sheet Lib "JRTalk" Alias "@S_Y2Sheet$qqspfspc" (y As Single, ByVal ilen As Integer,
        Optional ByVal text As String = 0) As Long
Declare Function S_WaitForReplyCOM Lib "JRTalk" Alias "@S_WaitForReplyCOM$qqsspc" (ByVal com As
        Integer, ByVal buffer As String) As Long
Declare Function S_WriteCOM Lib "JRTalk" Alias "@S_WriteCOM$qqsspcs" (ByVal com As Integer, ByVal
        command As String, ByVal wait As Integer) As Long
Declare Function S_Zero Lib "JRTalk" Alias "@S_Zero$qqsf" (ByVal x As Single) As Integer
```

JRTalk licence

The JRTalk extension to Serf/XL-Plot and the math lib is shareware.
It may be used free of charge for a trial period of 100 days, after
which your copy needs to be registered in order to function.
The licence is valid for installation on a single computer.
After registration, future updates (contained in the program installation
file) will be free of charge. Hence, deinstallation and subsequent
reinstallation of a more recent version of Serf/XL-Plot will not invalidate
the licence.

The licence may be ordered from: **http://order.kagi.com/?8XT**
The registration number of your copy of JRTalk is required for the
purchase. Your registration number may be obtained by either
issuing S_Error(-99) or running «Register_JRTalk.exe » located
in the program directory.