**Using the Logger32 External Interface**

# Bob Furzer K4CY

## 1.0 GENERAL

A series of Windows messages have been coded that allow message exchange between Logger32 and a user-developed application, and data transfer using ADI formatted strings.

A skeleton VB application (with source code) is provided as a starting point for those who wish to write interfaces.

## 2.0 ESTABLISHING COMMUNICATION

The first step in establishing communication between your application and Logger32 your application must identify the hWnd of the Logger32 MDI:

Dim L32hWnd as long

L32hWnd = FindWindow(vbNullString, "Logger32")

If Logger32 is not running, your application must retry periodically until Logger32 is running.

### 2.1 Initial WIndows Messages to Logger32

Second, your application must register/identify an unused Windows message (Logger32 supports up to 5 simultaneous external applications):

### 2.1.1 Message 1

Dim L32Msg as long

L32Msg = RegisterWindowMessage("Logger32 1")

PostMessage L32hWnd, L32Msg, 1, GhW ‘ GhW is the hWnd of your main form

Logger32 will reply to you with a message number 0. If the L32Msg you send is in use by another external application, the lParam will be set to 0. If the L32Msg is unused, then the lParam will be set to 1. If your connect request was rejected (lParam is 0), then your application must retry using a different L32Msg:

L32Msg = RegisterWindowMessage("Logger32 2")

PostMessage L32hWnd, L32Msg, 1, GhW ‘ GhW is the hWnd of your main form

Logger32 currently supports up to 5 external interface applications. Your application can retry the connection using a RegisteredWindowMessage “Logger32 1” to “Logger32 5”.

### 2.1.2 Message 2

When your application has successfully connected to Logger32 (you receive a message number 0 with an lParam of 1), you must now send a message number 2 to Logger32 with an lParam of the TextBox.hWnd Logger32 is to send text strings to. If your application does not want unsolicited data from Logger32, then you set the lParam to 0:

PostMessage L32hWnd, L32Msg, 2, TextBox.hWnd

### 2.1.3 Message 3

Immediately following the message 2 to Logger32 you have the option to send a message 3 to Logger32. Setting the lParam to 0 (or simply not sending a message 3) will tell Logger32 to not send any DX Spot information to your application (I suspect most applications will not want to receive DX Spots from Logger32). Setting the lParam to 1:

PostMessage L32hWnd, L32Msg, 3, 1

will tell Logger32 to send [ADIF](#ADIF) formatted text to your application for each DX Spot received by Logger32. If you have previously set the message 2 lParam to 0 (no TextBox hWnd provided to Logger32) and you set the message 3 lParam to 1 (telling Logger32 to send your DX Spot information) Logger32 will complain, generate a warning message and turn off the request.

Logger32 will respond by doing three things:

* + 1. Logger32 sends your application a message number 3. The lParam of this message is the hWnd of a TextBox in Logger32 your application sends ADI formatted text strings to.
    2. Logger32 sends your application a message 99. The IParam of this message is the radio currently in use (1 or 2).
    3. Logger32 sends your application a message 100. The lParam of this message is the radio frequency (in Hz).

If your application has OK’d the receipt of unsolicited text strings from Logger32 (you sent a TextBox hWnd as the lParam of your message number 2), Logger32 will send your application a WM\_SETTEXT message and put the current Radio Mode in the TextBox you have identified. The format of the text is (say) <APP\_RADIO\_MODE:3>SSB

## 3.0 ADDITIONAL MESSAGES TO LOGGER32

The basic connection and synchronization between Logger32 and your application is complete. Logger32 will respond to the following additional messages from your application.

**Message 4**. lParam is 0. Your application tells Logger32 it has stopped. Logger32 will free up the RegisteredWindowMessage your application was using.

**Message 5**. lParam is 0. Your application tells Logger32 to shut down.

**Message 6**. lParam is 0. Your application relinquishes PTT control. Logger32 assumes PTT control and initialized PTT ports/keying lines.

**Message 7**. lParam is 0. Your application assumes control of the PTT.

**Message 8**. lParam is 0. Your application tells Logger32 to key the PTT.

**Message 9**. lParam is 0. Your application tells Logger32 to unkey the PTT.

**Message 10**. lParam is the hWnd of a TextBox in your application to write to. Logger32 will respond with an ADIF formatted text string in the following format <APP\_SET\_FREQ\_MODE:27>18132.012|CW The frequency is in KHz and the decimal separator will be in the correct format for the PCs regional settings.

**Message 11**. lParam is 0. Your application tells Logger32 to release CAT control of the Radio.

**Message 12**. lParam is 0. Your application tells Logger32 to take CAT control of the Radio.

**Message 13**. lParam is 0. If the Logger32 Mode controlled by Sound Card option is checked, then <APP\_FORCE\_MODE:x> messages from your application will change Logger32 Mode.

**Message 14**. lParam is 0. Disables the feature turned on by message 13. Remember to turn this off when your application closes.

**Message 15**. If lParam is 1 then a flag is set so that Logged QSOs will be marked for QSLing. If lParam is 0 then the flag is turned off.

**Message 16**. If lParam is 1 then a flag is set so that Logged QSOs will be marked for eQSLing. If lParam is 0 then the flag is turned off.

**Message 17**. If lParam is 1 then a flag is set so that Logged QSOs will be marked for LoTW. If lParam is 0 then the flag is turned off.

**Message 18**. If IParam is 1 then Logger32 radio polling is turned off. If IParam is 0 then Logger32 radio polling is turned on. If your application

disables Logger32 radio polling, remember to turn it back on when your application closes.

**Message 19**. If lParam is 0 then Logger32 Mode from BandPlan option is turned off. If lParam is 1 then Logger32 Mode from BandPlan is turned on. If lParam is 2 then Logger32 will respond with a message 103 indicating the Mode from BandPlan in the lParam. If your application uses message 19, remember to set the value back to the original setting when your application closes.

**Message 20**. lParam is ignored. An ExternalInternetCallsignLookup module sends this message on lookup completion. On receipt of this message, Logger32 looks for and processes the data file written by the external lookup module.

**Message 21**. lParam is ignored. When Logger32 receives this message, the export LoTW file process is started. On completion of the export, Logger32 sends unsolicited messages to the app: <APP\_LoTW\_RECORDS:xx> and <APP\_LoTW\_FILENAME:xx>.

**Message 22**. lParam is ignored. When Logger32 receives this message the export eQSL file process is started. On completion of the export, Logger32 sends unsolicited messages to the app: <APP\_eQSL\_RECORDS:xx> and <APP\_eQSL\_FILENAME:xx>.

**Message 23**. lParam is ignored. When Logger32 receives this message, the sync LoTW process is started. Prior to sending this message the application should send a <APP\_IMPORT\_SYNC\_FILENAME:xx>xxx message to Logger32 to advise the filename of the file to be imported.

**Message 24**. lParam is ignored. When Logger32 receives this message, the sync eQSL process is started. Prior to sending this message, the application should send a <APP\_IMPORT\_SYNC\_FILENAME:xx>xxx message to Logger32 to advise the filename of the file to be imported.

**Message 25** An lParam of 1 flags Logger32 to write an ADIF formatted record to a file ‘eQSL Dump File.txt’ in the application directory for each QSO logged. If the lParam is 2, the ADIF record is written and the EQSL\_QSL\_SENT field in the Logbook is set to ‘Y’. If the lParam is 0, the function is turned off and no ADIF formatted records are written. Logger32 remembers any previous setting of this function. It is not necessary to send this message every time the external application is executed, only when the user wishes to change the setting. Logbook informational entries like G3NPA=) are not written.

**Message 26**. An lParam of 1 flags Logger32 to write an ADI formatted record to a file 'ADIF audit trail.txt' in the application directory as each QSO is logged or modifies. An lParam of 0 flags Logger32 to not write the record. It is not necessary to send this message every time the external application is executed, only when the user wishes to change the setting. New QSOs added to the Logger32 Logbook, and written to the 'ADIF audit trail.txt' file, are prefixed with the text <APP\_QSO\_LOGGED:0>. QSOs deleted from the Logger32 Logbook, and written to the 'ADIF audit trail' file, are prefixed with the text <APP\_QSO\_DELETED:0>. QSOs in the Logger32 Logbook that are modified, and written to the 'ADIF audit trail' file, are written as two records. The first, as the unmodified QSO prefixed with the text <APP\_QSO\_DELETED:0>, the second as the modified QSO prefixed with <APP\_QSO\_LOGGED:0>. It is the responsibility of the external application to do whatever file maintenance/cleanup of the 'ADIF audit trail.txt' as necessary. All logbook entries (including informational entries like G3NPA=) are written. The user must remove these if they are not required for your application.

**Message 27**. Same as message 26, but enables/disables writing to the "ClubLog Dump File.txt" in the application directory. Logbook informational entries like G3NPA=) are not written.

**Message 28**. Sending this message to Logger32 (params are ignored) opens the External Interface Debug window to allow message sequences to be monitored. This window can also be turned on by adding this entry to the Logger32.ini file:

[Globals]

External Interface Debug=1

**Message 29**: Sending this message to Logger32 (params are ignored) makes Logger32 send all current QSO information from the Logbook Entry Window.

**Message 120**. Send this message to Logger32 if you have written data to a disk file as the result of an internet lookup.

## 4.0 ADDITIONAL MESSAGES FROM LOGGER32

Logger32 will send the following additional messages to your application:

**Message 99** This message is sent prior to message 100. The IParam contains the number of the currently in use radio (1 or 2).

**Message 100** This message is sent whenever the radio frequency changes. The lParam contains the radio frequency in Hz. Message 100 is always preceeded by message 99.

**Message 101** This message is sent if Logger32 has calculated a valid beam heading and the user types CTL\_A or ALT\_A. The iParam contains the Short Path direction (if the user typed CTL\_A) or Long Path direction (if the user typed ALT\_A). The directions are calculated to include any/all Logger32 corrections the user has set. No checking is done to ensure that direction is within the range of 0 to 360 degrees; this checking/correction must be done by the external application.

**Message 102** This message is sent following a user click on a DX Spot.

**Message 103** This message is sent in response to receipt of a message 19 with IParam of 2. The message 103 will have an IParam of 0 if the Logger32 Mode from BandPlan option is off and an IParam value of 1 if the Logger32 Mode from BandPlan is on.

**Message 104** This message is sent whenever the secondary radio frequency changes. The IParam contains the radio frequency in Hz. Message 104 is always preceded by message 99.

**Message 105** Logger sends this message to the application every time a QSO is written to the log and an ADIF formatted message is written to the eQSL Dump File. When an ADIF file is imported into Logger32 a single message 105 is sent on completion of the import. This function is enabled/disabled when Logger32 receives a message 25. Prior to sending message 105, Logger32 sends a <APP\_eQSL\_DUMP\_FILENAME:xx>filename to the application.

**Message 106** Logger32 sends this message every time a record is written to the ‘ADIF audit trail.txt’ file. When a logbook QSO is deleted, and ADIF formatted is added to the file. The record has the additional field of <APP\_QSO\_LOGGED:0> to indicate why the record was added to the file. Logger32 sends your application a message 106 with an lParam of 0. When a logbook QSO is added, an ADIF formatted record is added to the file. The record has the additional field of <APP\_QSO\_LOGGED:0> to indicate why the record was added to the file. Logger32 sends your application a message 106 with an lParam of 1. When a logbook QSO is modified, an ADIF formatted record is added to the file. The record has the additional fields of <APP\_QSO\_MODIFIED:0> and <APP\_HRD\_APIKEY:xx>call|date|time (call, date and time are the values prior to modification) to indicate why the record was added to the file and to provide the necessary keys for the application to identify the original QSO record that was modified. Logger32 sends your application a message 106 with an lParam of 2.

**Message 107** Logger sends this message to the application every time a QSO is written to the logbook and an ADIF formatted message is written to the ClubLog Dump File. When an ADIF file is imported into Logger32 a single message 105 is sent on completion of the import. This function is enabled/disabled when Logger32 receives a message 27. Prior to sending message 107, Logger32 sends a <APP\_CLUBLOG\_DUMP\_FILENAME:xx>filename to the application.

**Message 108** If the currently selected rotor is “External Interface” Logger32 sends this message to the application to request azimuth rotor position. This is a poll message. The application will reply with a message 108 that has the position in the lParam

**Message 109** Logger32 sends this message when the auto-lookup process (callsign lookup) is triggered.

**Message 110** Logger32 sends this message when the antenna selector is changed. The lParam contains the antenna number (either decimal of BCD).

**Message 111** Logger32 sends this message to external applications when a QSO is logged.  The message is not sent if QSOs are imported or added from the ADD Window.

## 5.0 ADIF TEXT STRINGS

Data is exchanged between your application and Logger32 by writing ADIF formatted text strings to identified TextBoxes. It is suggested/recommended that the user utilize the TextBox\_Change event as a trigger to process the ADIF data received.

### 5.1 Logger32 recognizes the following ADIF strings

**<EOR>** If received without additional text, Logger32 will clear all current entries in the Logbook Entry Window.

**<APP\_TAB>** If received without additional text, this simulates moving the focus from the Callsign field of the Logger32 Logbook Entry Window. Automatic functions (like QRZ lookup, Previous QSOs lookup, auto-populate of data from previous QSOs, etc. are triggered.

**<CALL:x>** Callsign is placed on the Logbook Entry window.

**<RST\_RCVD:x>** The RST Received is placed on the Logbook Entry window.

**<RST\_SENT:x>** The RST Sent is placed on the Logbook Entry window.

**<NAME:x>** The name is placed on the Logbook Entry window.

**<APP\_TIME\_ON:x>** is in the Microsoft timestamp format (example: 39470.6737384259) and sets the ADIF QSO\_DATE and TIME\_ON fields.

**<APP\_TIME\_OFF:x>** is in the Microsoft timestamp format (example: 39470.6737384259) and sets the ADIF TIME\_OFF field.

**<APP\_QSL:1>Y** sets the Logger32 QSL flag on. Any other character(s) turns the flag off.

**<APP\_eQSL:1>Y** sets the Logger32 eQSL flag on. Any other character(s) turns the flag off.

**<APP\_LoTW:1>Y** sets the Logger32 LoTW flag on. Any other character(s) turns the flag off.

**<COMMENT:x>** the comment is placed on the Logbook Entry window.

**<QTH:x>** the QTH is placed on the Logbook Entry window.

**<ADDRESS:x>** the address is placed on the Logbook Entry window.

**<STATE:x>** if the user has assigned STATE as a user field on the Logbook Entry window then the Primary Administrative Subdivision is placed on the Logbook Entry window.

**<CNTY:x>** if the user has assigned CNTY as a user field on the Logbook Entry window then the Secondary Administrative Subdivision is placed on the Logbook Entry window.

**<GRIDSQUARE:x>** if the user has assigned GRIDSQUARE as a user field on the Logbook Entry window then the gridsquare is placed on the Logbook Entry window.

**<IOTA:x>** if the user has assigned IOTA as a user field on the Logbook Entry window then the IOTA is placed on the Logbook Entry window.

**<STX:x>** if the user has assigned STX as a user field on the Logbook Entry window then the transmitted serial number is placed on the Logbook Entry window.

**<SRX:x>** if the user has assigned SRX as a user field on the Logbook Entry window then the received serial number is placed on the Logbook Entry window.

**<QSL\_VIA:x>** if the user has assigned QSL\_VIA as a user field on the Logbook Entry window then the QSL information is placed on the Logbook Entry window.

**<QSLMSG:x>** if the user has assigned QSLMSG as a user field on the Logbook Entry window then the QSL message is placed on the Logbook Entry window.

**<SAT\_NAME:x>** if the user has assigned SAT\_NAME as a user field on the Logbook Entry window then the satellite name is placed on the Logbook Entry window.

**<SAT\_MODE:x>** if the user has assigned SAT\_MODE as a user field on the Logbook Entry window then the satellite mode is placed on the Logbook Entry window.

**<PROP\_MODE:x>** if the user has assigned PROP\_MODE as a user field on the Logbook Entry window then the propagation mode is placed on the Logbook Entry window.

**<FREQ\_RX:x>** if the user has assigned FREQ\_RX as a user field on the Logbook Entry window then the receive frequency is placed on the Logbook Entry window.

**<TEN\_TEN:x>** if the user has assigned TEN\_TEN as a user field on the Logbook Entry window then the 10x10 number is placed on the Logbook Entry window.

**<USER\_1:x>** if the user has assigned USER\_1 as a user field on the Logbook Entry window then the user defined data is placed on the Logbook Entry window.

**<USER\_2:x>** if the user has assigned USER\_2 as a user field on the Logbook Entry window then the user defined data is placed on the Logbook Entry window.

**<USER\_3:x>** if the user has assigned USER\_3 as a user field on the Logbook Entry window then the user defined data is placed on the Logbook Entry window.

**<APP\_LOGQSO:x>** will log data currently on the Logbook Entry window. This may be sent as standalone data, or part of a complete QSO to be logged..

**<FREQ:x>** if the Logger32 Radio Type is set to none, then this data (in KHz) will simulate a change of radio frequency in Logger32.

**<MODE:x>** if the data received does not match the mode of the Logbook Entry window, a warning message is generated.

**<APP\_FORCE\_FREQ:x>** If a message 18 with IParam of 1 has been sent to Logger32 (radio polling has been turned off) this will change the frequency in the Logbook Entry window.

**<APP\_FORCE\_MODE:x>** This will change the mode in the Logbook Entry window. The message is intended to be used when Logger32 does not have control of the radio. If Logger32 has control of the radio, the Mode you have forced may be overwritten by the mode from the next poll.

**<APP\_SET\_FREQ\_MODE:x>** Logger32 will set the radio frequency and radio mode to the data received. The format is (say) <APP\_SET\_FREQ\_MODE:14>14003.451|CW-R Note that the frequency is in KHz.

**<APP\_SET\_FREQ:x>** Logger32 will set the radio frequency to the data received. The format is (say) <APP\_SET\_FREQ:9>14003.451 Note that the frequency is in KHz.

**<APP\_SET\_MODE:x>** Logger32 will set the radio mode to the data received. The format is (say) <APP\_SET\_MODE:3>USB. For this to work, the Logger32 Mode from BandPlan or Mode from Radio must be enabled

**<APP\_SET\_MODE\_DIGITAL:x>** Logger32 will set the radio mode to the data received. The format is (say) <APP\_SET\_MODE\_DIGITAL:3>USB. This is used with radios that have modes like USB-D. For this to work, the Logger32 Mode from BandPlan or Mode from Radio must be enabled.

**<APP\_CLICK\_DXSPOT:x>** This simulates clicking on a DX Spot in Logger32. Your application must pass both the frequency (in KHz) and the DX station callsign. The format is (say) <APP\_CLICK\_DXSPOT:13>14003.01|K4CY.

**<APP\_IMPORT\_SYNC\_FILENAME:xx>** This is sent to Logger32 prior to sending message 23 or 24 to update Logger32 on the file to be imported.

**<APP\_CLICK\_RCP\_BUTTON:x>y** Your application sends this ADIF message to Logger32 to click a Radio Control Panel macro button. y is the button number in the range of 1 to 48. The Radio Control Panel does not have to be open for this to work.

**<APP\_QSO\_UPDATE:0>&ADIF field data&<EOR>** Logger32 update ADIF field in the Logbook (NAME, ADDRESS, STATE, CNTY, IOTA, GRIDSQUARE, QSL\_VIA and QTH) ADIF field data will be like this.

<IOTA:6>AS-007<GRIDSQUARE:4>PM95

**<APP\_OVERWRITE:1>0 or <APP\_OVERWRITE:1>1** 　0 meaning only populate empty fields and 1 meaning overwrite existing fields with new data.

### 5.2 Logger32 sends the following ADIF strings to your application (if you have allowed unsolicited data to be sent)

**<CALL:x>** is sent if the Callsign is changed on the Logbook Entry window

**<RST\_SENT:x>** is sent when the RST\_SENT field is changed on the Logbook Entry window

**<RST\_RCVD:x>** is sent when the RST\_RCVD field is changed on the Logbook Entry window

**<NAME:x>** is sent when the NAME field is changed on the Logbook Entry window

**<MODE:x>** is sent when the MODE of the Logbook Entry window is changed.

**<APP\_RADIO\_MODE:x>** is sent when the radio mode is changed.

**<APP\_RADIO\_MODE\_SECONDARY:x>** is sent when the secondary radio mode is changed.

**<APP\_DXSPOT\_CALLSIGN:x>** is sent when a DX Spot is received.

**<APP\_DXSPOT\_DXCC:x>** is sent when a DX Spot is received.

**<APP\_DXSPOT\_FREQ:x>** is sent when a DX Spot is received.

**<APP\_DXSPOT\_BAND:x>** is sent when a DX Spot is received. The band is derived from the Logger32 BandPlan.

**<APP\_DXSPOT\_MODE:x>** is sent when a DX Spot is received. The operation mode (i.e. SSB, CW, RTTY, etc.) mode is derived from the Logger32 BandPlan.

**<APP\_DXSPOT\_TIME:x>** is sent when a DX Spot is received.

**<APP\_DXSPOT\_COLOR:x>** is sent when a DX Spot is received if the DX Spot is highlighted.

**<APP\_DXSPOT\_CLICKED:x>** is sent when a DX Spot has been clicked on. The format is: <APP\_DXSPOT\_CLICKED:18>HB9RDE|14225.0|SSB

**<APP\_PTT\_STATUS:x>** is sent when Logger32 receives a message 6, 7, 8 or 9. Additional message may be sent if messages are ignored (as an example if PTT is OFF and Logger32 receives a message 9).

**<APP\_LoTW\_RECORDS:x>** is a count of exported QSOs sent when Logger32 completes a LoTW file export.

**<APP\_LoTW\_FILENAME:xx>** is the filename of a LoTW export.

**<APP\_eQSL\_RECORDS:x>** is a count of exported QSOs sent when Logger32 completes a eQSL file export.

**<APP\_eQSL\_FILENAME:xx>** is the filename of a eQSL export.

**<APP\_QSO\_UPDATE:x>y <EOR>** y is callsign for lookup. It will be like this.

<APP\_QSO\_UPDATE:6>JA1NLX<EOR>

**<EOR>** is sent when Logger32 clears the Logbook Entry window.

## 6.0 THINGS TO CONSIDER

If you write an application or interface to Logger32, please check the code handles frequency strings where the decimal separator may be either a period or a comma (depending on the users PC Regional Settings). Also, please ensure you handle the connect sequence correctly so that you don’t tie up more than one RegisteredWindowMessage. The sample code provides a working sample.

Finally, please make sure you release any resources (including your assigned RegisteredWindowMessage when your application closes.