HASCOM International
A.B.N. 80 264 836 521
Hardware And Software Communications
15 Marloo Lane Quinns Rocks Western Australia 6030
Tel: +61 8 9415 1311
Fax: +61 8 9415 1312
Email: hascom@hascom.com.au
Web: www.hascom.com.au

MAP27Com.ocx API Definition

Revision	Date	Author
A	24 September 1999	Chris King, HASCOM International

HASCOM International
15 Marloo Lane
Quinns Rocks
Western Australia, 6030

Contents

Methods

short OpenPort(LPCTSTR Port, short Baud, short Format);

This method opens the serial port and initialises the MAP27 protocol.

Port should point to a string that defines the port to open. For example "COM1:".
Baud is the baud rate to use. For example 9600.
Format selects the data format to use. A value of 1 (the default) signifies 8 data bits, no parity & one stop bit. A value of 2 signifies 7 data bits, even parity & one stop bit.

Returns 1 if successful. The return is –1 if the port could not be opened, -2 if the communication timeouts could not be set, -3 if the communication format could not be set and -4 if the receive thread could not be created.

BOOL ClosePort();

This method closes the serial port.

The return is TRUE if successful or FALSE if the port was not open.

short RadioManagement(short Controls);

This method allows configuration of the radio operating parameters according to the MAP27 B3h message. *Controls* is a bit map of 8 flags in the lower byte as defined in section 5.2.2.8.7 of the MAP27 document:

CONTROLSa:

8	7	6	5	4	3	2	1	
0								User does not wish to or is not capable of receiving voice calls
1								User wishes to receive voice calls
	0							User does not wish to or unit is not capable of receiving modem calls
	1							User wishes to receive modem calls
		0						User does not wish or unit is not capable to receive status messages
		1						User wishes to receive status messages
			0					User does not wish or unit is not capable to receive SST messages
			1					User wishes to receive SST messages
				0				User does not wish or unit is not capable to receive MST messages
				1				User wishes to receive MST messages
					0			Radio shall not automatically set up calls to a diversion address
					1			Radio shall automatically set up calls to a diversion address
						0		Set call-back logging inactive
						1		Set call-back logging active
							*	Reserved for further extensions, set to '0'

short Dial(short Prefix, short Ident, short CallDetails);

This method causes a MAP27 message to be sent to initiate a call. The *Prefix* and *Ident* should be set to the MPT1327 ID for the radio to be called.

The *CallDetails* parameter specifies additional parameters. The high byte signifies the MAP27 message to use, either A4h for a normal call or A5h for emergency call (A4h is assumed if this value is zero). The low byte signifies the 'call details' as described in section 5.2.2.4 of the MAP27 protocol.

CALL DETAILS:

8	7	6	5	4	3	2	1	
*								Reserved, set to '0'
	0							Non-include call
	1							Include call
		0						Individual or group call, called user(s) may reply
		1						Group call, called users are not allowed to reply
			0					Voice call
			1					Modem (Data) call
				0				High priority call
				1				Non-priority call
					0	0	0	Standard call
					*	*	1	Special customised service request
Others								Reserved

For example, to generate a voice call use a *CallDetails* value of 08h or 18h for a NPD call.

The return is 0 if the *CallDetails* high byte is invalid else it is the same as for the SendMap27Message method.

The outcome of the call attempt will be shown by either a CallSetup or ProtocolInfo event.

short Answer();

This method attempts to answer an incoming call by sending a Radio Control message (B2h) with the 'off hook' bit set. The return is the same as for the SendMap27Message method.

The outcome of the call attempt will be shown by either a CallSetup or ProtocolInfo event.

short Disconnect();

This method attempts to end a call by sending a Radio Control message (B2h) with the 'off hook' bit cleared. The return is the same as for the SendMap27Message method.

The outcome of the call attempt will be shown by either a CallCleared or ProtocolInfo event.

short SendStatus(short Prefix, short Ident, short StatusNum);

This method uses a send status message (80h) to attempt to send *StatusNum* to the radio identified by *Prefix* and *Ident*. The return is the same as for the SendMap27Message method.

The outcome of the call attempt will be shown by either a StatusDataProgress, CallCleared or ProtocolInfo event.

short SendData(short Prefix, short Ident, LPCTSTR Message);

This method uses a send data message (81h or 82h) to attempt to send *Message* to the radio identified by *Prefix* and *Ident*. The return is the same as for the SendMap27Message method.

The outcome of the call attempt will be shown by either a StatusDataProgress, CallCleared or ProtocolInfo event.

short SendNpd(VARIENT Data);

This method uses a send NPD message (A3h) to attempt to send the binary information contained in *Data*. A data call must previously have been established. The maximum number of bytes that can be contained in Data is defined as the return value from the MaxMessageSize method less one. The return is 0 if *Data* is invalid, otherwise it is the same as for the SendMap27Message method.

The outcome of this message is not provided for in the MAP27 protocol, however there are Tait specific messages that will provide this information. They cause a SendDataResult event.

BOOL ParseMap27Messages(short Enable);

This function sets or clears a flag in the control. If the flag is set, all received MAP27 packets will cause a ReceivedMap27Message event.

When the flag is set the control first attempts to process received messages, at which time events such as IncomingCall may occur, then the ReceivedMap27Message event is fired. The flag is always initially cleared when the control is loaded.

Enable should be zero to clear the flag or non-zero to set the flag.

The return is the new state of the flag.

short MaxMessageSize();

During negotiation of a MAP27 link, both parties agree on a maximum packet size. This method returns that size. If the link is not available, the return is 0. Note a typical return value, as seen with the Tait T2040, is 112.

Short SendMap27Message(VARIENT Message);

This method sends a free format MAP27 message.

Message must contain an array of bytes (VT_ARRAY) that define a MAP27 packet. The maximum number of bytes allowable is returned by the MaxMessageSize method. The control will automatically add the protocol wrapper requirements such as checksums, etc.

The return is 1 if the message has been sent successfully and 2 if the message has been queued to send. A return of 0 indicates *Message* is invalid, –1 signifies the MAP27 link is not ready to transfer messages, –2 signifies that the data is too large for the link, -3 signifies that the (5 message deep) transmit queue is full and the message will not be queued/sent.

For example, to request the status of the radio settings, *Message* should contain the two bytes B0h 02H. Or, to send a status message 10 to radio 70 1522 *Message* should contain the following bytes, 80h 46h 05h F2h 00h 0Ah.

Events

Map27Link(short Connected);

This event occurs whenever a MAP27 link is successfully negotiated or when a link is lost. The value of *Connected* is non-zero if a link has been negotiated or zero if it has been lost.

RadioLink(short InService);

This event occurs whenever the radio reports the status of it's connection to the network has changed. The radio is also polled automatically causing this event after successful MAP27 link negotiation. The value of *InService* is non-zero if the radio is in contact or zero if it is not.

IncomingCall(short Prefix, short Ident, short CallDetails);

This event occurs whenever the radio receives an incoming call. *Prefix* & *Ident* identify the calling radio. *CallDetails* contains the call type in the high byte (A4h for normal call & A5h for emergency call). The low byte contains the call details as described in the MAP27 document section 5.2.2.4.4/5

CALL DETAILS:

8	7	6	5	4	3	2	1	
*								Reserved, set to '0'
	0							Non-include call
	1							Include call
		0						Individual call
		1						Group call
			0					Voice call
			1					Data call
				0				Call has been connected (Hook signal is not needed) (GTC)
				1				Hook signal is required before connection is established (AHY)
					0	0	0	Standard call, no parameters field
					0	0	1	Reserved
					0	1	0	Reserved
					0	1	1	Reserved
					1	0	0	Reserved
					1	0	1	Customised service 1 indication
					1	1	0	Customised service 2 indication
					1	1	1	Customised service 3 indication
Others								Reserved

StatusDataProgress(short Cause);

This event is caused during transmission of a status or data message (not NPD) to show the progress of the transmission. Cause shows the message type in the high byte (C0h for successful transmission, D0h for queuing and E0h for unsuccessful). And the cause is in the low byte as documented in the MAP27 protocol section 5.2.2.1.3.

CAUSE for Message type C0h:

8	7	6	5	4	3	2	1		
0	0	0	0	0	0	0	0	ACK	Successful transaction

CAUSE for Message type D0h:

0	0	0	0	0	0	1	0	ACKQ	System busy, wait for signalling
0	0	0	0	1	0	1	0	ACKQ	Called unit engaged, wait for signalling
0	0	1	0	0	1	1	0	ACKT	Called unit's calls are diverted and radio unit tries to send message to the diversion address

CAUSE for Message type E0h:

0	0	0	0	1	0	0	0	ACK	Transaction aborted
0	0	0	0	0	0	1	1	ACKX	Invalid call, message rejected
0	0	0	0	1	0	1	1	ACKX	System or called unit overload, message rejected
0	0	0	0	0	1	0	0	ACKV	Called radio out of reach or transaction abandoned
0	0	0	0	1	1	0	0	ACKV	Called unit engaged or does not wish to accept message
0	0	0	0	0	1	1	0	ACKT	Called unit's calls are diverted
0	0	0	1	0	1	1	0	ACKT	Called unit's calls are diverted to a group address
0	0	0	0	1	1	1	0	ACKT	Called unit's calls are diverted, but the diversion address is not available

CallSetup(short Cause);

This event shows the progress of an attempt to set up or receive a call. The high byte of *Cause* shows the message type (C4h is setup positive, D4h is setup queuing, E4h is setup negative, C5h receive positive, D5h receive warning, E5h receive not connected). And the cause is in the low byte as documented in the MAP27 protocol section 5.2.2.4.3.

CAUSE for Message type C4h:

8	7	6	5	4	3	2	1		
0	0	0	0	0	0	0	0	GTC	Call connected
0	0	0	0	0	0	0	0	ACK	Include call connected

CAUSE for Message type D4h:

0	0	0	0	0	0	0	1	ACKI	Called unit alerting
0	0	0	0	0	0	1	0	ACKQ	System busy, wait for signalling
0	0	0	0	1	0	1	0	ACKQ	Called unit engaged, wait for signalling
0	0	0	0	0	1	0	1	ACKE	Emergency call is proceeding, wait for signalling
0	0	1	0	0	1	1	0	ACKT	Called unit's calls are diverted and radio unit tries to set-up call to the diversion address
0	0	1	1	0	1	1	0	ACKT	Called unit's calls are diverted to a group and radio unit tries to set-up call to the diversion address

CAUSE for Message type E4h:

0	0	0	0	1	0	0	0	ACK	Call set-up aborted
0	0	0	0	0	0	1	1	ACKX	Invalid call, call set-up rejected
0	0	0	0	1	0	1	1	ACKX	System or called unit overload, call set-up rejected
0	0	0	0	0	1	0	0	ACKV	Called radio out of reach or call set-up abandoned
0	0	0	0	1	1	0	0	ACKV	Called unit engaged or user does not wish to accept the call
0	0	0	0	0	1	1	0	ACKT	Called unit's calls are diverted
0	0	0	1	0	1	1	0	ACKT	Called unit's calls are diverted to a group address
0	0	0	0	1	1	1	0	ACKT	Called unit's calls are diverted, but the diversion address is not available
0	0	0	0	0	1	1	1	ACKB	Called unit has accepted the call for call-back

CallCleared(short Cause);

This event occurs whenever a call is ended. The high byte of *Cause* contains the message type (86h is cleared normal & A6h is cleared abnormal). And the cause is in the low byte as documented in the MAP27 protocol section 5.2.2.6.2

CAUSE for message type A6h:

8	7	6	5	4	3	2	1		
0	0	0	0	0	0	0	0	CLEAR MAINT Local	Not specified, all message transactions, call set-ups or calls are cancelled or disconnected
0	0	0	0	0	0	0	1	Local	Radio generated clear e.g. radio path protocol time-out or on-hook on the radio set
0	0	0	0	0	1	0	0	Local	Service not available (Radio unit not in radio contact)
0	0	0	0	0	1	0	1	Local	Transmission or message too long, call disconnected or message rejected
0	0	0	0	0	1	1	0	Local	Message coding not possible, message rejected
0	0	0	0	1	1	1	0	MAINT (110)	Voice or modem call disconnected, abnormal end

CAUSE for message type 86h:

0	0	0	0	1	0	1	1	CLEAR	Voice or modem call disconnected, normal end

.

ReceivedStatus(short Prefix, short Ident, short StatusNum);

This event is generated when the radio receives a status message. *Prefix* & *Ident* identify the sending radio and *StatusNum* is the status number received.

ReceivedData(short Prefix, short Ident, LPTSTR Message);

This event is generated when the radio receives a data message. *Prefix* & *Ident* identify the sending radio and *Message* is the received text (7 bit or 8 bit ascii is assumed).

ReceivedNpd(VARIENT Data);

This event is generated when the radio receives a NPD message during a data call. *Data* contains the bytes received in an array of bytes.

SendDataResult(short Status);

This event is generated when the radio sends a Tait specific MAP27 message showing the progress of a NPD data transmission. The high byte of *Status* contains the message type (C1h is successful, D1h is queuing & E1h is unsuccessful). And the low byte contains the cause as documented by Tait.

CAUSE for Message type C1h:

8	7	6	5	4	3	2	1	
0	0	0	0	0	0	0	0	Successful transmission

CAUSE for Message type D1h:

0	0	0	0	0	0	1	0	Data Queued

CAUSE for Message type E1h:

0	0	0	0	0	0	0	1	TDP format mismatch
0	0	0	0	0	0	1	0	Internal modem has failed
0	0	0	0	0	1	0	0	Data channel failure
0	0	0	0	1	0	0	0	Invalid data packet size
0	0	0	1	0	0	0	0	Data channel timeout
0	0	1	0	0	0	0	0	Modem busy
0	1	0	0	0	0	0	0	Transmission aborted

Numbering(short FleetPrefix, short RadioIdent, short LowestIdent, short HighestIdent, BOOL NumberingScheme);

After a successful link negotiation, the control automatically requests the radio numbering information. The *FleetPrefix* is the Prefix of the fleet to which the radio belongs, *RadioIdent* is the ident of the attached radio. *LowestIdent* & *HighestIdent* are the limits of individual idents in the fleet. *NumberingScheme* is FALSE for small fleets (which use 2 digit short dialing) and TRUE for large fleets (which use 3 digit short dialing).

This information can help with making calls within the same fleet as the lowest short dial ID is always 20 or 200 therefore to dial radio 32, use the FleetPrefix and add 12 to the LowestIdent.

ProtocolInfo(short Reason);

This event occurs when the radio sends a protocol information message via the MAP27 link. This is usualy to inform the DTE of some protocol error or to advise that some requested function is unsupported. *Reason* containts the reason for the message as documented in the MAP27 protocol section 5.2.2.8.9.

REASON:

8	7	6	5	4	3	2	1	
0	0	0	0	0	0	0	1	Unrecognised message
0	0	0	0	0	0	1	0	Facility or addressing not supported
0	0	0	0	0	0	1	1	Protocol state mismatch detected i.e. received message not compatible or allowed at the present state (optional)
0	0	0	0	0	1	0	0	Message coding not supported
1	*	*	*	*	*	*	*	Spare for customisation
Others								Reserved

ReceivedMap27Message(short MessageId, VARIANT Message);

This method is fired when a MAP27 packet is received. The ParseMap27Messages method must previously have been called to set the flag. The *MessageId* has the same value as the first byte contained in the *Message*, this is to allow filtering of messages without incurring the variant processing overhead.

Log(LPCTSTR LogMessage);

This event provides text log messages, useful for debugging or providing extra information to the user.