

MAP27API

Programmer's Manual

**© 2006 – 2009 Barley Mountain, LLC.
All rights reserved.**

www.map27.net

TABLE OF CONTENTS

1. Overview

2. Architecture

3. Function Reference

- 3.1 MAP_Initialize
- 3.2 MAP_Start
- 3.3 MAP_Stop
- 3.4 MAP_Reset
- 3.5 MAP_GetRadioInformation
- 3.6 MAP_MakePrivateCall
- 3.7 MAP_MakeGroupCall
- 3.8 MAP_AnswerCall
- 3.9 MAP_DisconnectCall
- 3.10 MAP_PTT
- 3.11 MAP_SendStatus
- 3.12 MAP_SendMessage
- 3.13 MAP_SetMaxLen
- 3.14 MAP_SendMAP27Command

4. Callback Events

- 4.1 Callback Function Declaration
- 4.2 Event Codes

5. Result Codes

6. Addressing

1. Overview

The MAP27API provides easy access to the majority of functions as defined in the MAP27 interface for trunked radio systems (MPT1327 and MPT1343).

The original purpose of MAP27 was to allow universal access to a variety of trunked radios, by using the same set of commands and events. The complicated data-link layer however is an obstacle for many developers. Most applications only need a subset of the entire MAP27 functionality; therefore it is not always cost effective to develop an in house solution by building a broad implementation of the MAP27 protocol stack.

With this API, any developer has easy access to a very simple set of functions. It is possible to send status and text messages as well as establishing private and group calls with just a few lines of code.

This document describes the programmer's interface to use the MAP27API which is embedded in the MAP27API.DLL file. All functions that are referenced in this document are directly exported from the DLL. They can be used from any programming language (for example C, C++, VB, VB.Net, C#.NET, Delphi, Assembly) as long as DLL function calls are supported.

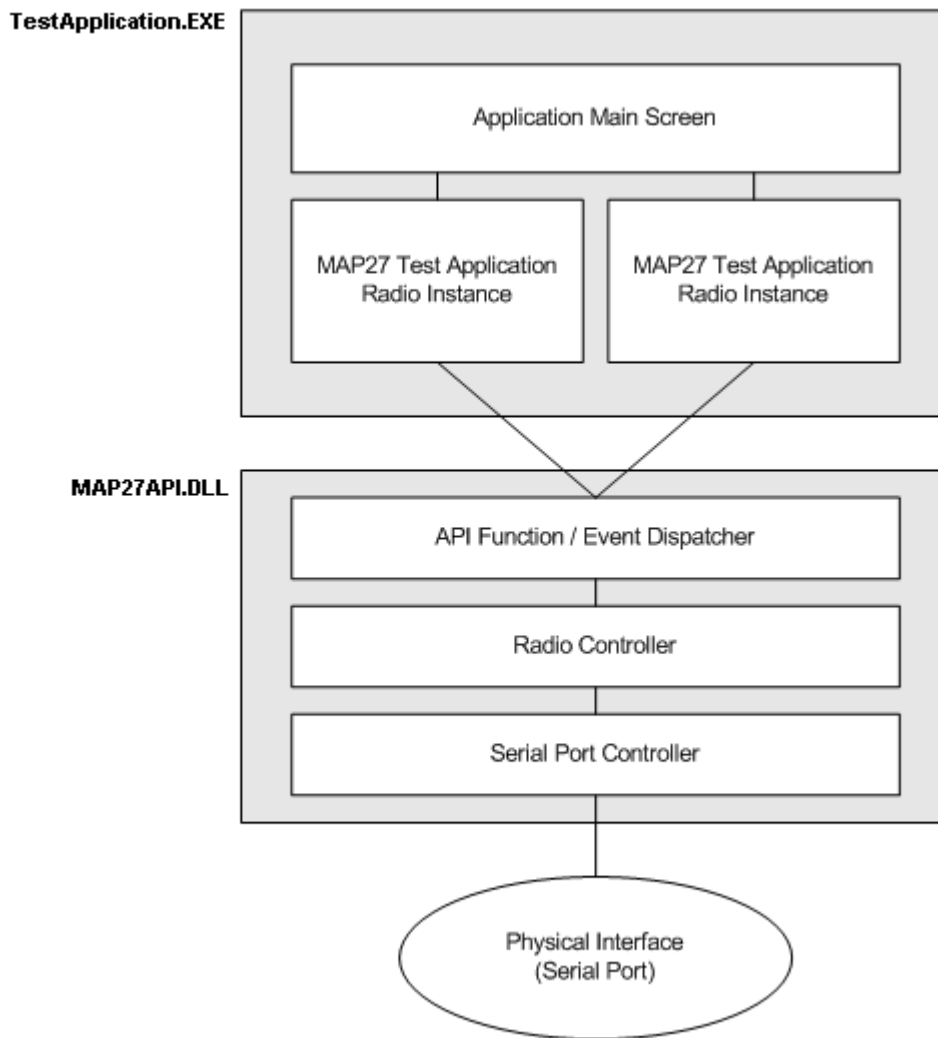
Up to 8 radios can be controlled simultaneously with this DLL (located at serial ports COM1 through COM8)

The application life-cycle for using this API:

1. Application Startup
2. Call MAP_Initialize(...)
3. Call MAP_Start(...) for all necessary ports
4. Run Application (call other MAP_* functions)
- [...]
5. Call MAP_Stop(...) for all necessary ports
6. Exit Application

2. Architecture

The diagram below shows how the MAP27API.DLL is used by an application. The Radio instances from the test application call the API functions in the DLL. The DLL dispatches the function calls to the correct Radio controller object that handles all MAP27 low level protocol functions and communicates directly with the radio via RS232 (serial port). Incoming events are signaled to the application by calling a callback function that is defined in the MAP_Initialize(...) function at application start up.



3. Function Reference

3.1 MAP_Initialize

Purpose:

Initialize the DLL, must be called before any other functions are called.

Function declaration:

```
int MAP_Initialize(int mapEventSize, MAPCallbackFunc callback, short  
radioOffset, short groupOffset, char* licenseKey);
```

Parameters:

mapEventSize	sizeof MAPEvent
callback	Pointer to callback function that will be called for any events (see chapter 4 for details).
radioOffset	Logical offset that is added to the internal numbering of the radio IDs. This offset is used in every address related operation of this API (see chapter 6 for details)
groupOffset	Logical offset that is added to the internal numbering of the group IDs. This offset is used in every group ID related operation of this API (see chapter 6 for details)
licenseKey	zero-terminated string of license-key as provided by vendor. If this parameter is set to NULL, the API is run in evaluation mode, and will shut down the application after 10 minutes.

Possible function results:

```
MAP_SUCCESS  
MAP_SYSTEM_ALREADY_INITIALIZED  
MAP_INVALID_KEY  
MAP_INVALID_PARAMETER
```

3.2 MAP_Start

Purpose:

Open a serial communication channel to a connected radio and establish a MAP27 link to it.

Function declaration:

```
int MAP_Start(int serialPort, int speed);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
speed	Serial port baud rate (valid values: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200)

Possible function results:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_ALREADY_OPEN
MAP_UNABLE_TO_OPEN_PORT
MAP_INVALID_BAUD_RATE

3.3 MAP_Stop

Purpose:

Close a serial communication channel.

Function declaration:

```
int MAP_Stop(int serialPort);
```

Parameters:

serialPort Physical serial port number. Range 1 – 8.

Possible function results:

```
MAP_SUCCESS  
MAP_SYSTEM_NOT_INITIALIZED  
MAP_PORT_OUT_OF_RANGE  
MAP_PORT_NOT_IN_USE
```

3.4 MAP_Reset

Purpose:

Reset a MAP27 link (reestablish connection)

Function declaration:

```
int MAP_Reset(int serialPort);
```

Parameters:

serialPort Physical serial port number. Range 1 – 8.

Possible function results:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE

3.5 MAP_GetRadioInformation

Purpose:

Retrieve information about the connected radio

Function declaration:

```
int MAP_GetRadioInformation(int serialPort, int size, TRadioInformation*
    pTRadioInformation);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
size	sizeof TRadioInformation
pTRadioInformation	Pointer to a TRadioInformation structure (see below)

Possible function results:

```
MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_INVALID_PARAMETER
MAP_RADIO_NOT_INITIALIZED
```

```
struct TRadioInformation
```

```
{
    unsigned char    Prefix           // connected unit's PFI
    short            Ident            // connected unit's INDENT
    short            LogicalAddress;  // connected unit's logical address
    short            FirstRadio;      // base radio address
    short            FirstGroup;      // base group id
    short            NumberOfUnits;   // number of radios
    short            NumberOfGroups   // number of groups
    unsigned char    ManufacturersCode;
    unsigned char    ModelNumber;
    int              SerialNumber;
    TSupportedFacilities SupportedFacilities;
    TSupportedCodings SupportedCodings;
};
```

```
struct TSupportedFacilities
{
    bool          VoiceCalls;
    bool          ModemCalls;
    bool          StatusMessages;
    bool          SST;
    bool          MST;
    bool          AutoDiversion;
    bool          CallBackLog;
};
```

```
struct TSupportedCodings
{
    bool          BCD_ASCII;
    bool          CCITT2_ASCII;
    bool          Binary;
    bool          BCD;
    bool          Telex;
    bool          CCITT5_ASCII;
    bool          PC_CHAR;
};
```

3.6 MAP_MakePrivateCall

Purpose:

Establish connection to other radio

Function declaration:

```
int MAP_MakePrivateCall(int serialPort, short address);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
Address	Logical radio address of destination radio

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_RADIO_NOT_INITIALIZED
MAP_RADIO_ADDRESS_OUT_OF_RANGE

3.7 MAP_MakeGroupCall

Purpose:

Establish group call connection

Function declaration:

```
int MAP_MakeGroupCall(int serialPort, short groupID);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
groupID	ID of talkgroup

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_RADIO_NOT_INITIALIZED
MAP_GROUP_ID_OUT_OF_RANGE

3.8 MAP_AnswerCall

Purpose:

Answer incoming call

Function declaration:

```
int MAP_AnswerCall(int serialPort);
```

Parameters:

serialPort Physical serial port number. Range 1 – 8.

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_RADIO_NOT_INITIALIZED
MAP_NO_INCOMING_CALL

3.9 MAP_DisconnectCall

Purpose:

Disconnect active call or transmission

Function declaration:

```
int MAP_DisconnectCall(int serialPort);
```

Parameters:

serialPort Physical serial port number. Range 1 – 8.

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_RADIO_NOT_INITIALIZED

3.10 MAP_PTT

Purpose:

Disconnect active call or transmission

Function declaration:

```
int MAP_PTT(int serialPort, int push);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
push	1 = PTT pressed (sending), 0 = PTT not pressed

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_RADIO_NOT_INITIALIZED
MAP_RADIO_NO_ACTIVE_CALL

3.11 MAP_SendStatus

Purpose:

Send status code (0-31) to other radio

Function declaration:

```
int MAP_SendStatus (int serialPort, short address, unsigned char status);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
address	Logical radio address of destination radio
status	Status code to transmit (0-31)

Function Result:

```
MAP_SUCCESS  
MAP_SYSTEM_NOT_INITIALIZED  
MAP_PORT_OUT_OF_RANGE  
MAP_PORT_NOT_IN_USE  
MAP_RADIO_NOT_INITIALIZED  
MAP_RADIO_NOT_IDLE  
MAP_INVALID_STATUS_CODE  
MAP_RADIO_ADDRESS_OUT_OF_RANGE
```

3.12 MAP_SendMessage

Purpose:

Send text message to other radio. A text message is transmitted as CCITT-5 ASCII message (7bit). It will be sent as SST or MST, depending on the message size (> 25 characters will be MST). The maximum message size is 100 characters.

Function declaration:

```
int MAP_SendMessage (int serialPort, short address, char* message);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
address	Logical radio address of destination radio
message	Zero terminated message string

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE
MAP_PORT_NOT_IN_USE
MAP_RADIO_NOT_INITIALIZED
MAP_RADIO_NOT_IDLE
MAP_MESSAGE_TOO_LONG
MAP_RADIO_ADDRESS_OUT_OF_RANGE

3.13 MAP_SetMaxLen

Purpose:

Set the “Maximum length” field for the MAP27 data link layer to a different value than the default (=10). Some radios do not initialize unless this value is set to something lower.

Function declaration:

```
int MAP_SetMaxLen (int serialPort, int maxLen);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
maxLen	“Maximum length” (as defined in MAP27 section 4.3.2.1.)

Function Result:

MAP_SUCCESS
MAP_SYSTEM_NOT_INITIALIZED
MAP_PORT_OUT_OF_RANGE

3.14 MAP_SendMAP27Command

Purpose:

Sends a raw network layer command to the radio. Using this function, the user has the ability to call functions that are in the MAP27 specification but not supported by the SDK, or functions that are proprietary to certain radios.

Function declaration:

```
int MAP_SendMAP27Command (int serialPort, char* command, int size);
```

Parameters:

serialPort	Physical serial port number. Range 1 – 8.
command	Pointer to a memory buffer that contains the raw data to be sent to the radio
size	Size of buffer in bytes

Function Result:

```
MAP_SUCCESS  
MAP_SYSTEM_NOT_INITIALIZED  
MAP_PORT_OUT_OF_RANGE  
MAP_PORT_NOT_IN_USE  
MAP_RADIO_NOT_INITIALIZED  
MAP_RADIO_NOT_IDLE
```

Example:

```
// increasing the volume level:  
char cmd[3];  
cmd[0] = 0xB6;           // VOLUME_CONTROL command (5.2.2.8.10)  
cmd[1] = 0x01;           // "Normal speech audio path"  
cmd[2] = 0x01;           // "Up"  
MAP_SendMAP27Command(1, cmd, sizeof(cmd));
```

4. Callback Events

4.1 Callback Function Declaration

The API notifies the application by executing the callback function that is provided by the `MAP_Initialize(...)` function call (See chapter 3.1).

The calling convention is defined as followed:

```
typedef void (*MAPCallbackFunc) (MAPEvent* event);
```

It passes a single parameter, which is a pointer to a `MAPEvent` structure:

```
struct MAPEvent
{
    unsigned char    EventCode;    // type of event, see below
    unsigned char    PortNumber;   // port number
    unsigned char    RadioState;   // current or new radio state
    unsigned char    SignalStrength; // current or new signal strength
    short            RemoteAddress; // current peer radio address
    char             Data[102];    // status or text message
};
```

4.2 Event Codes

- MAP_EVENT_STATE_CHANGED = 0
 The radio state has changed. The "RadioState" parameter of the event structure contains the new radio state, which is one of the following:
- RADIO_STATE_DISCONNECTED=0 MAP 27 link is down
 - RADIO_STATE_INITIALIZING = 1 Radio is being initialized
 - RADIO_STATE_IDLE = 2 Radio connected and idle
 - RADIO_STATE_CALLING = 3 Outgoing call in progress
 - RADIO_STATE_RINGING = 4 Incoming call pending
 - RADIO_STATE_CALL_ACTIVE= 5 Call in progress
 - RADIO_STATE_SENDING = 6 Sending data
- If a call is being established, in progress, or data is being sent, the "RemoteAddress" parameter of this event carries the logical radio address of the peer radio. The "Data" parameter contains a reason or progress description, if a call was disconnected, or transaction performed.
- MAP_EVENT_SIGNAL_STRENGTH = 1
 The signal strength level has changed. The "SignalStrength" parameter of the event structure contains the new value. The range is 0 to 255.
- MAP_EVENT_STATUS_RECEIVED = 2
 A status code was received from a peer radio. The code is stored as ASCII number in the "Data" parameter of the event structure. The "RemoteAddress" parameter of this event carries the logical radio address of the sending radio.
- MAP_EVENT_TEXT_MESSAGE_RECEIVED = 3
 A text message was received from a peer radio. The message is stored as ASCII text in the "Data" parameter of the event structure (zero terminated). The "RemoteAddress" parameter of this event carries the logical radio address of the sending radio.

5. Result Codes

All API functions return a single integer value that reports success or failure of the operation. The codes are defined as followed:

MAP_SUCCESS	= 0
Function completed successfully	
MAP_SYSTEM_NOT_INITIALIZED	= 1
Unable to perform function because MAP_Initialize(...) has not been called.	
MAP_PORT_OUT_OF_RANGE	= 2
The physical port number is out of range (valid range is 1 - 8)	
MAP_UNABLE_TO_OPEN_PORT	= 3
Unable to open serial port with given COM-port number	
MAP_PORT_NOT_IN_USE	= 4
Cannot close port, because it has not been opened	
MAP_INVALID_PARAMETER	= 5
At least one parameter is invalid. Usually the size of a structure is either invalid or incompatible.	
MAP_RADIO_NOT_INITIALIZED	= 6
The operation can not be performed, because the radio has not been fully initialized yet	
MAP_RADIO_ADDRESS_OUT_OF_RANGE	= 7
The provided address is out of range. The range is determined by the radio programming as well as the offset provided in MAP_Initialize(...)	
MAP_GROUP_ID_OUT_OF_RANGE	= 8
The provided group ID is out of range. The range is determined by the radio programming as well as the offset provided in MAP_Initialize(...)	
MAP_INVALID_STATUS_CODE	= 9
The status code must be in the range of 0 – 31	
MAP_MESSAGE_TOO_LONG	= 10
The maximum number of characters is 100	
MAP_SYSTEM_ALREADY_INITIALIZED	= 11
The API has already been initialized	

- MAP_INVALID_KEY = 12
The provided license key is invalid
- MAP_INVALID_BAUD_RATE = 13
The provided baud rate is not valid. It can only be one of the following:
1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- MAP_PORT_ALREADY_OPEN = 14
Port can not be opened, because it is already open. To change the baud rate, or reset the port, the port must be closed first.
- MAP_RADIO_NO_ACTIVE_CALL = 15
Operation cannot be performed, because there is no incoming (ringing) and no active ongoing call at this time
- MAP_NO_INCOMING_CALL = 16
Operation cannot be performed, because there is no incoming (ringing) call at this time
- MAP_RADIO_NOT_IDLE = 17
Operation cannot be performed, because the radio is not in idle state.

6. Addressing

In MAP27, radio units are addressed by a code that consists of a 7 bit “PFIX” code, plus a 13 bit “IDENT” code. This code uniquely identifies the radio throughout the entire trunking system. In order to hide this fact, radios are programmed with a base number that is subtracted from the actual radio ID. All radios of one fleet are programmed with the same base number, therefore the radios can communicate with each other, using a simple 2 or 3 digit code (address). The same goes for the group IDs, which are PFIX/IDENT numbers as well.

Since after subtracting the base number, the address and group ranges would be 0..[max radios - 1] and 0..[max groups - 1], it is possible to configure offsets for radio IDs and group IDs, when programming the radios. That way it is possible to include all radio and group IDs within one numbering scheme.

For example, if a fleet has 10 radios and 3 groups, it can be set up to the following scheme:

10..19 = Radio IDs (radio id offset is 10)

50..52 = Group IDs (group id offset is 50)

Now the user can access all radios and groups with a 2 digit code from the radio keyboard.

This API accepts the same code in its function calls. It is not necessary to know the PFIX and IDENT base codes, because this API retrieves them automatically from the radio unit. However, the radio ID offset and group ID offset must be manually set in the MAP_Initialize(...) function. If they are not known, the recommended way is leaving them set to zero, and addressing all radios and groups in the fleet by using address ranges that are starting at 0.

Note: The MAP27 protocol also provides means for additional address coding information (“ADESC” field), which is not supported by this API.