BOSCH

Bosch Security Systems B.V.

Allegiant Satellite SDK
User Guide

Table of Contents

1. Overview

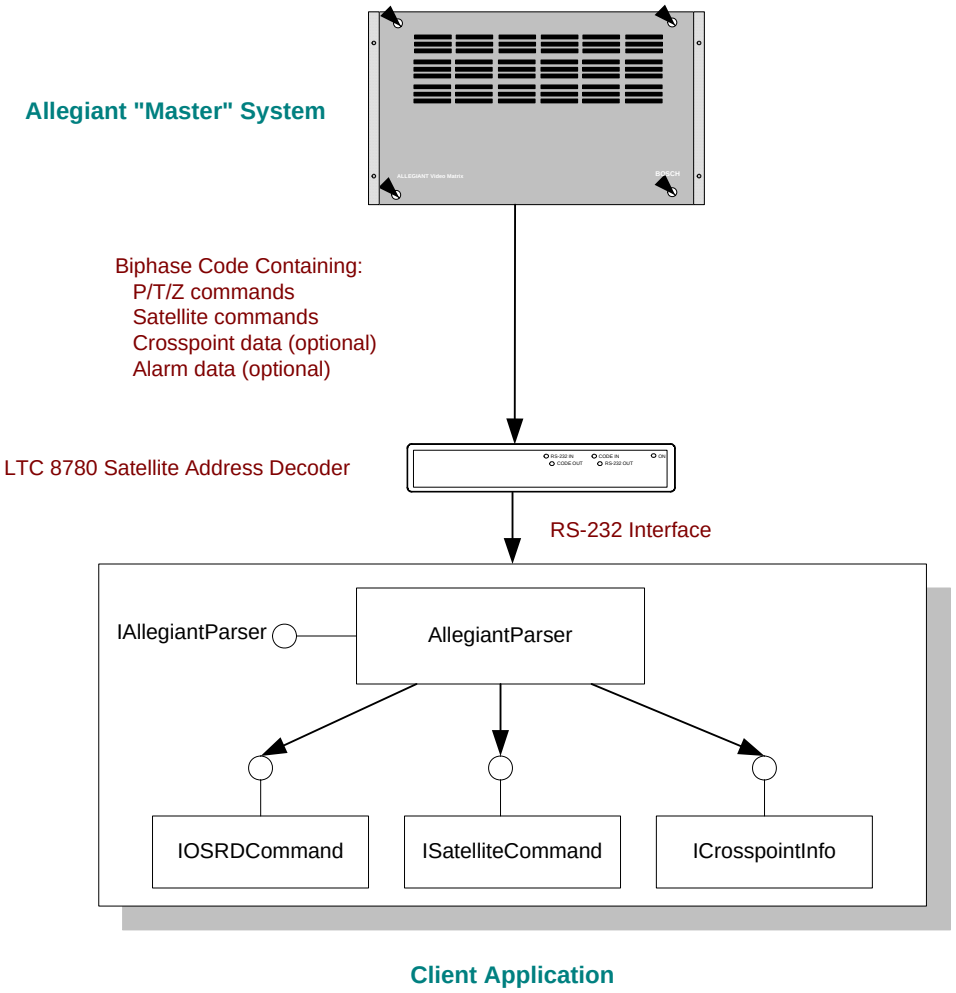
1.1 Introduction

The *Allegiant SDK User Guide* provides details on how a client application can use the Allegiant Satellite SDK to retrieve commands that are forwarded from a master Allegiant system to the LTC 8780 Data Converter units.

1.2 Features Provided by the SDK

- Convert Allegiant satellite commands from a binary protocol to ActiveX interface methods
- Ability to communicate with LTC 8780 under several settings and deployment scenarios
- Translate continuous PTZ commands to their indefinite equivalents
- Support for various Parser modes that dictate command delivery format
- Scales to allow multiple LTC 8780 units within the same Application
- Asynchronous mode of operation during command parsing
- Integrates into any ActiveX enabled development environment

1.3 System Diagram



1.4 Allegiant Version Compatibility

Allegiant CPU(8900)	Version 8.4
Allegiant CPU(Non-8900)	Version 8.2
Allegiant MCS (LTC 8059)	Version 2.60 or higher

1.5 Supported Containers

The Allegiant Satellite SDK is designed to support the following ActiveX containers. Other containers may be able to use the SDK, but the primary consumers are listed here.

- ActiveX containers developed with Microsoft Visual C++ 6.0
- Microsoft Visual Studio .NET applications

1.6 SDK Library Files Descriptions

The SDK consists of one binary file its described below see Section 3 for installation instructions for a client machine.

Filename	Description
LTC8780_CommandParser.dll	This is the main ActiveX/COM DLL for the SDK. This DLL must be registered for use by a client application.

1.7 Sample Applications Included

The SDK includes four sample Applications, three that use C++ in separate Windows Application frameworks Console, ATL and MFC. In addition, a sample that uses Microsoft's .NET framework all the samples are found in the SDK Examples directory.

1.7.1 Installing the .NET Example

The .NET examples requires for the correct version of Microsoft's .NET framework installed on the host machine prior to running it, this sample was built using MS Visual Studio .NET 2003 and the .NET Framework version 1.1.

The framework can be downloaded from Microsoft's website using the following link;

http://msdn.microsoft.com/netframework/downloads/updates/default.aspx

2. Service

If the unit ever needs repair service, the customer should contact the nearest Bosch Security Systems, Inc. Service Center for return authorization and shipping instructions.

Service Centers

USA

Phone: 800-366-2283 or 717-735-6638

fax: 800-366-1329 or 71-735-6639

CCTV Spare Parts

Phone: 800-894-5215 or 408-956-3853 or 3854

fax: 408-957-3198

e-mail: BoschCCTVparts@ca.slr.com

Canada

Phone: 514-738-2434

Europe, Middle East & Asia Pacific Region

Phone: 32-1-440-0711

For additional information, see

www.boschsecuritysytems.com.

3. SDK Redistribution

To redistribute the Allegiant Satellite SDK along with a client application, the rules noted below must be followed. If these rules are not followed, applications that use the SDK may not function properly.

1. When the file LTC8780_CommandParser.dll is redistributed to end-user PCs, it must be copied to the <*common files*>*Bosch Security Systems* folder. The <common files> value is located in the registry at:

   key: HKEY_LOCAL_MACHINE \ SOFTWARE \ Microsoft \ Windows \ CurrentVersion

   value: CommonFilesDir

2. Existing copies of LTC8780_CommandParser.dll should only be overwritten if their version numbers are lower than those of the copies being redistributed. Furthermore, LTC8780_CommandParser.dll must be registered.
3. The LTC8780_CommandParser.dll can be registered by running the following command at the DOS command prompt in the same directory where the DLL is located;

   regsvr32 LTC8780_CommandParser.dll

4. Using the SDK

The SDK features are presented in a logical order in this section, from most basic to advanced. Refer to the *Allegiant Satellite SDK Interface Definition* for details about the object model.

4.1 Configuring the LTC 8780 Data Converter Unit

This section briefly covers the dip switch settings for the LTC 8780 unit for further details and description please refer to the *LTC 8780 Series Instruction Manual*.

4.1.1 Hardware Configuration

The hardware configuration consists of setting up the Allegiant Master Satellite system and setting up the Data converter unit.

4.1.1.1 Allegiant Master Satellite System

The SDK is designed to work with an Allegiant matrix/controller setup in a satellite configuration. It requires an Allegiant matrix/switcher, an LTC 8780 Data Converter Unit and a PC work station.

You should be familiar with the Allegiant Matrix Switch control system, Windows-based software, and have a general knowledge of networking.

Refer to the Allegiant Instruction Manual for setting up an Allegiant Satellite System, which can be found on the Bosch website:

www.boschsecurity.com.

4.1.1.2 LTC 8780 Data Converter Unit

- The LTC 8780 Data Converter is required to convert Allegiant Satellite commands into a format that matches the software settings.
- Connect the LTC 8780 CODE IN connector to the biphase port of the Allegiant.
- Refer to the LTC 8780 manual for biphase connections, which can be found on the Bosch website: www.boschsecurity.com.
- The LTC 8780 is connected to the biphase COM port using a Bosch S1385 or Industry standard Null modem RS-232 data cable.
- The data rate of the RS-232 interface must match the software setting.
- See *Table 1* and *Table 2* for DIP Switch settings.

Table 1: DIP Switch Settings

DIP Switch	Switch Number	Recommended Setting	Function
S101	1 (BAUD)	ON	RS-232 Baud Rate of 38400
	2 (BAUD)	ON	
	3 (CODE DIST)	OFF	Biphase input to Biphase output
	4 (ADDR FUNCT)	OFF	CCL Address Select
S102	1 (RCVR DRVR)	ON	Enable transmission of R/D messages
	2 (XPOINT)	OFF	Disable transmission Crosspoint messages
	3 (CCL)	ON	Enable transmission of CCL messages
	4 (MATCH)	ON	CCL message address not checked
S103	1 – 4 (ADDR LOW)	(See *Table 2*)	Address or Block Select Low Order Bits (Must be set to Satellite Address)
S104	1 – 4 (ADDR HIGH)	(See *Table 2*)	Address or Block select high order bits (Must be set to Satellite Address)
S105	1 (MODEM CMD)	OFF	Not applicable
	2 (ORIGINATE)	OFF	Modem command + ATA
	3 (CHECK CTS)	ON	Check CTS before transmitting data
	4	OFF	Not used

Table 2: CCL Address Select

Address DIP Switches								CCL Address Select
S104				S103				
4	3	2	1	4	3	2	1	
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	1
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	2
OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	3
OFF	OFF	OFF	OFF	OFF	OFF	ON	ON	4
OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	5
OFF	OFF	OFF	OFF	OFF	ON	OFF	ON	6
OFF	OFF	OFF	OFF	OFF	ON	ON	OFF	7
OFF	OFF	OFF	OFF	OFF	ON	ON	ON	8
X	X	X	X	X	X	X	x	(continue series)
ON	ON	ON	ON	ON	ON	ON	OFF	255
ON	ON	ON	ON	ON	ON	ON	ON	256

4.1.2 Software Configuration

The software configuration must match the hardware settings above otherwise the parser will not be able to receive and decode the incoming commands. A majority of the configuration items are exposed as properties of the parser component.

4.1.2.1 Configuring the RS-232 Serial Port

The baud rate, RS-232 handshaking mode and COM port name must match the settings from above, the following properties can be used to apply the settings;

Using the *IAllegiant* interface, which is exposed by the *AllegiantParser* coclass set the following properties;

PortName(…) - RS-232 PC port that is connected to code converter.

BaudRate(…) - Baud rate setting using jumpers S101 on the Data converter.

HandShake(…) - Handshake mode as set using jumper S105 on the Data converter.

Once configured the serial port can be opened and closed using the *OpenCOMM*(…) and *CloseCOMM*(…) methods on the *IAllegiantParser* interface.

4.1.2.2 Using the Data Converter Unit with Address matching Enabled

The Data Converter unit can be configured to either forward all CCL commands that are received from the Master Allegiant system or only forward commands that match the Address of the Data Converter unit. Switch #4 on dip switch S101 enables the Address function and Switch #4 on dip

switch S102 enables Address matching, the format of CCL messages will vary based on how your configure the dip switches.

The Command Parser exposes a property *UseMessageHeader*(…) on the *IAllegiantParser* interface which needs to be set to true when Address matching is disabled and false otherwise.

4.2 Selecting the Parser Mode

The command parser supports three independent modes, *ParserModeOSRD*(…) for OSRD commands, *ParserModeSatellite*(…) for Satellite commands and *ParserModeCrosspoint*(…) for crosspoint commands, all these properties are members of the *IAllegiantParser* interface.

Each property is of type *PARSER_MODE* that takes one of the following values;

PARSER_MODE_PROCESS - process all commands and forward using interface methods

PARSER_MODE_USE_RAW_METHOD - semi process the command and forward using raw method that includes device address

PARSER_MODE_RAW - no processing required just forward all commands to the raw interface

4.3 Starting the Allegiant Command Parser

Once all the settings have been applied starting the command parser is very straight forward, the client application first needs to implement at least one of the callback interface – *IOSRDCommand*, *ISateliteCommand* or *ICrossPointInfo*. Once the interface is available just call the *StartParser*(…) method from the *IAllegiantParser* interface supplying at least one callback interface pointer, this method is asynchronous and will return once the parser has been started, incoming messages will be pushed to the appropriate callbacks as they arrive on the wire.

The command parser can be stopped at any moment by invoking the *Stop*(…) method.

4.4 Handling Camera Switching commands

The camera switching commands are sent to the *ISateliteCommand* interface through the *SetCameraOnLogicalMonitor*(…) method, once a client application has implemented this interface and supplied it as an argument to *StartParser*(…) it should be called periodically with the camera switching instructions.

4.5 Processing PTZ commands

PTZ commands are forwarded to the *IOSRDCommand* interface, again this interface needs to be implemented and supplied to *StartParser*(…). Based on the *ParserModeOSRD* property setting the commands will either be decomposed and forwarded to interface methods or they will be sent to one of the raw methods.

When the *ParserModeOSRD* is set *to USE_RAW_METHOD* all commands will be channeled to the *PutRawOSRDCommand*(…) method, the second parameter (*arrRawCommand*) contains the raw PTZ payload that can be decoded by an Autodome Camera hence it is probably the most efficient way for the client application to transparently transmit PTZ commands.

4.6 Translating continuous PTZ operations to their indefinite equivalents

The motion of the AutoDome camera tends to be jittery when continuous PTZ operations are used, this is mainly due to network latencies that slow down the transmission of packets furthermore a lot of bandwidth is utilized when sending out repetitive commands. The command parser can be instructed to convert continuous PTZ operations to their indefinite equivalents, in this mode the Application will only receive two commands, one to start the indefinite PTZ operation and a last one to cancel the last PTZ operation. When using variable speed PTZ commands, the client application will only get speed updates.

The *EnableRepetitiveCommands*(…) properties of interface *IAllegiantParser* enables the PTZ command translation when set to false.

4.7 Error Handling

Every interface method returns an *HRESULT*, when an error condition occurs the low order bytes of the *HRESULT* will contain the error code once this value has been extracted it will map to one of the values in *PARSER_ERROR* enum. The following code snippet displays how to extract the error code;

```
.
.
HRESULT hr = S_OK;
//attempt to open the serial port
hr = pIAllegiantParser->OpenCOMM(_bstr_t("COM1"));
// check the return value
if(FAILED(hr))
{
        PARSER_ERROR eError = (PARSER_ERROR)HRESULT_CODE(hr);
        if(PARSER_ERROR_PORT_OPEN_FAILED  ==  eError)
        {
                std::cout << "Failed to open the specified Communication port";
        }
}
```

4.8 Connecting to multiple LTC 8780 units

For each LTC 8780 unit that the client application wishes to connect to, one instance of the *AllegiantParser* object must be created and each unit needs a unique serial port. For example, if the client wants to connect three Data converters then three *AllegiantParser* objects instances and three serial ports will be required. In addition the client application may need separate callback objects so that they can determine what parser instance fired the callback.

4.9 SDK Version Information

The LTC8780_CommandParser.dll version information can be retrieved by using Windows Explorer, simply right click on the file, select properties and go to the Version tab.

5. REGIONAL BOSCH TECHNICAL SERVICE CONTACT INFORMATION

Region	Phone Number
Austria	0810001038
Bangladesh	6568825902
Belgium	078152380
Brunei	6568825902
Canada	1-800-326-1450
Denmark	70120680
Finland	010802340
France	0810810123
Germany	01803000321
Greece	0080046121007
Hong Kong/China	85226210532
India	91207120516 / 91207120541
Ireland	01850789389
Indonesia	6568825902
Italy	0147897410
Japan	6568823089
Korea	6568825902
Latin America	011-717-735-6540
Luxembourg	80024084
Malaysia	6568823529
Nepal	6568825902
Netherlands	09002020195
Norway	81000054
Pakistan	6568825902
Philippines	6568823089
Portugal	800846900
Singapore	6568823851
Spain	902119827
Sri Lanka	6568825902
Sweden	087505193
Switzerland	0848803123
Thailand	6626311879. ext. 518
Taiwan	886221342602
United Kingdom	08458400018
United States	1-800-326-1450
Vietnam	6568825902
English Speaking - Other	+31 40 2786473
French Speaking - Other	+31 40 2786473
German Speaking - Other	+31 40 2786473