Microvision Scanner Programming Guide

eXtensible Scanner Protocol

Limited Warranty

Manufacturer warrants that this laser bar code scanner will be free of defects in material and workmanship for one (1) year from the date of shipment. Manufacturer will, at its option, either repair, replace or refund the purchase price paid by buyer for the defective Products. Such repair, replacement or refund shall be buyer's sole remedy in the event of Manufacturer's breach of this limited warranty. Repaired or replaced parts or product may include new, reconditioned or re-manufactured parts and equipment at Manufacturer's option. All costs associated with shipment to Manufacturer for warranty service, including but not limited to freight, duties, insurance and customs fees are buyer's responsibility. Manufacturer will pay the freight costs (duties, insurance, customs and any other fees are buyer's responsibility) associated with the return shipment to buyer. The method of shipment will be at Manufacturer's discretion. Repair or replacement of any parts or equipment does not extend the period of warranty provided for herein. THIS LIMITED WARRANTY IS MANUFACTURER'S ONLY WARRANTY. MANUFACTURER DOES NOT GIVE WARRANTIES OF MERCHANTABILITY OR WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. To take advantage of this warranty, buyer should contact the seller not Manufacturer. Manufacturer does not warrant that the Product or associated software will run error free or without interruptions. Manufacturer is not liable for loss of data, loss of profit, cost of cover, indirect, incidental or consequential damages.

The warranty set forth herein does not cover and Manufacturer will have no obligations hereunder if any non-conformance is caused in whole or in part by; accident, transportation, neglect, misuse, alteration, modification, or enhancement of the products or incorporation, interfacing, attachment of any feature, program, or device to the products by a person or entity other than Manufacturer, failure to provide a suitable installation environment, use of the products for other than the specific purpose for which the products are designed or any use of the product not in accordance with the User Manual or other misuse or abuse of the product.

Microvision, Inc. 6222 185th Avenue NE Redmond, WA 98052

Microvision and ROV are trademarks of Microvision, Inc. Windows and Windows Mobile are trademarks of Microsoft Corporation. The Bluetooth word, mark, and logo are owned by the Bluetooth SIG, Inc. and any use of such marks by Microvision, Inc., is under license.

Regulatory Information

UL Listed to U.S. and Canadian standards

FCC Declaration of Conformity

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference (2) this device must accept any interference received including interference that may cause undesired operations. This applies to all product options.

FCC Radio Frequency Interference Statement

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

» Reorient or relocate the receiving antenna.

» Increase the separation between the equipment and receiver

» Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

» Consult the dealer or an experienced radio/TV technician for help.

To comply with FCC RF exposure compliance requirements, a separation distance of at least 20 cm must be maintained between the antenna of this device and all persons. This transmitter must not be collocated or operating in conjunction with any other antenna or transmitter.

CAUTION: Changes or modifications to this equipment not expressly approved by manufacturer could void the user's authority to operate this equipment.

Notice for Canada

Radio Interference Notice for Canada:

This Class B digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe B est conforme á la norme NMB-003 du Canada.

Complies with 21 CFR 1040.10 and 1040.11 except for deviations pursuant to Laser Notice No. 50, dated July 26, 2001.

Compliance Information

Product Name: ROV Scanner

Product Number: MS2200

Options: All

The product herewith complies with the requirements of Low Voltage Directive 73/23/EEC, EMC Directive 89/336/EEC, WEEE Directive 2002/96/EC, and carries the "CE" mark accordingly.

Product Name: ROV Scanner with Bluetooth

Product Number: MS2200-BT

Options: All

The product herewith complies with the requirements of R&TTE Directive 99/5/EC, WEEE Directive 2002/96/EC, and carries the "CE" mark accordingly. The equipment also carries the Class 2 equipment identifier:

Laser

Type: Semiconductor laser AlGaInP

Maximum output power: 5mW

Divergence

X axis: 6-15 degrees

Y axis: 22-38 degrees

Wavelength: 650-660nm

Laser

Typ: Halbleiterlaser AlGaInP

Ausgabeleistung: 5mW

Strahlabweichung:

X Achse: 6-15 Grad

Y Achse: 22-38 Grad

Wellenlänge: 650-660nm

$C\epsilon$

FCC Tested to Comply with FCC Standards C E
FOR HOME OR OFFICE USE
COMPLIES WITH 21 CFR 1040.10 & 1040.11
EXCEPT FOR DEVIATIONS PURSUANT TO
LASER NOTICE NO.50. DATED JULY 26, 2001

CLASS 1 LASER PRODUCT

[Bar Code]

SN XXXX XXXX XXXX

MANUFACTURED YYYY/MM/DD

Class 1 Laser Product Prodotto Al Laser Di Classe 1 Klasse 1 Laserprodukt Laserprodukt Klasse 1 Luokka 1 Lasertuote Produto Laser Da Classe 1 Produit Laser De Classe 1 Producto Laser De La Clase 1 Laserprodukt Der Klasse 1 Laserprodukt Klass 1	**Disposal:** Do not dispose of this product in unsorted municipal waste. To dispose of this product, contact seller.

Table of Contents

Introduction

This document describes the communication interface used by the Microvision ROV Scanner and ROV Scanner with Bluetooth. You can also find information about many scanner features and methods of operation. This document helps you take full advantage of the scanner's utility in the software applications you develop. The intended audience is developers and system integrators who must adapt the ROV scanner to their specific needs. It is beneficial if you have general knowledge of scanner technology. Knowledge of software programming is required.

Microvision recommends the Microvision Software Development Kits (SDKs) for ROV Scanner for application devleopment; however, this document can be used as a reference on scanner functions and operation. It can also help you adapt Microvision scanners to platforms for which Microvision does not offer an SDK.

Microvision has implemented a new interface called XSP, eXstensible Scanner Protocol, for the ROV Scanner. The XSP design goals include:

- Simplify the scanner interface
- Use a single scanner operating mode
    - Make all connections behave the same
    - Bluetooth connection can be a true cable replacement and does not require a complex protocol to get data
- Provide better support for developers
    - Use an ASCII-based standard communication protocol
        - Simplify coding
        - Easy to troubleshoot
    - Base on industry standards
        - Leverages the epcGlobal Reader Protocol
        - XML formatting
- Maintain backward compatibility with Flic Scanner Legacy Mode as much as possible

Regardless of how the ROV Scanner is configured, it operates on the following basic principles:

1. The only unsolicited output from the scanner is an "attention" message triggered by an event.
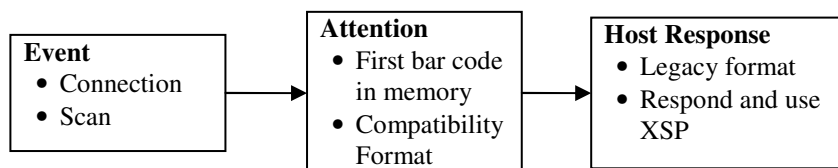
2. The scanner always responds to commands from the host with one of four message types
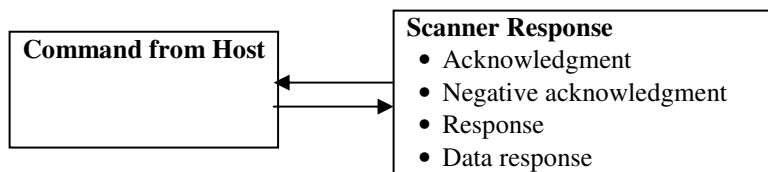
XSP Protocols

The ROV Scanner reads bar codes and then sends those bar codes from the scanner to the host device. The scanner can send each bar code to the host when it is scanned if the scanner is connected to the host device via cable or Bluetooth, or the ROV scanner can save the scanned data and send the bar codes in a group (batch mode). A ROV scanner can act as a "dumb scanner", which scans and sends the data. Or the scanner can interact with a host device to ensure that the data arrives safely and error free. There are several ways to send bar code data from the scanner to the host.

The ROV Scanner supports several protocols, that is, there are different ways to send data between the scanner and the host. ROV Scanners support most of the Legacy protocols used by the Flic Scanner. However, Microvision recommends that you use the Legacy protocols only if you want to adapt a ROV scanner to a Flic-based application. The ROV Scanner also supports the XML-based XSP (eXtensible Scanner Protocol). Microvision recommends that you use the M2M form of XSP, described in this document, to create applications designed specifically for ROV scanners.

The ROV communication protocols are listed below from the most simple to the most complex:

Protocol	Type	Comments
Compatibility	Legacy	This is the "dumb scanner" protocol. When you plug in a cable or establish a Bluetooth connection, the bar code data is sent after each scan. When not connected, the bar codes are stored in memory, and are then sent to the host when you plug in the cable or connect using Bluetooth.
ACK	Legacy	When connected to the host, the bar code data is sent after each scan. However, the host device must acknowledge each scan before the scanner can delete the bar code stored in memory. When not connected, the bar codes are stored in memory, and sent to the host when connected.
Serialization	Legacy	When connected to the host, the bar code data is sent after each scan. A four-character serial number is added to each scanned bar code. The host device must acknowledge each scan by sending the serial number and an ACK character to the scanner before the bar code is deleted.
TTY	XSP	The XSP TTY protocol is an XML-based data format. When an event occurs, the scanner sends the first bar code in memory using one of the Legacy protocols (depending on scanner configuration). The host device then replies with a TTY Null command to stop the Legacy protocol data transfer. The host device then sends TTY commands to download the data, delete data, or configure the scanner. The scanner must send a TTY response to a TTY command.
M2M	XSP	The XSP M2M protocol is similar to TTY but adds header and trailer packets to the XML data to improve data integrity. Does not require TTY command terminators within the M2M packet. Command terminators are recommended outside the M2M packet.

Note: Flic Scanners support all three Legacy protocols. Flic Cordless Scanners support only the Serialization protocol. ROV Scanner and ROV Scanners with Bluetooth support all three Legacy protocols.

Caution: Be careful when you use the Compatibility or Ack protocols over Bluetooth. Take steps to make sure that you do not lose or duplicate data.

Compatibility Protocol (Legacy)

When you use the **Compatibility Protocol**, the scanner sends the bar code data immediately after each scan. If the scanner is not connected, it saves the scanned data, and then sends the data when the scanner connects to a host device. After sending the bar code data to the host, the scanner immediately erases the bar code data from memory. The host must be ready to receive the data or the sent data will be lost. XSP provides an UndeleteBarCodes command to recover data not intended to be deleted. You should use the Compatibility Protocol only when you want to use a ROV scanner with an existing Flic-based application. Compatibility Protocol is the default for ROV scanners and is configured with the following settings:

BCDataACK = False
SeqNum=False
AutoDownload=True

ACK Protocol (Legacy)

The **ACK Protocol** gives you better data integrity than the basic Compatibility Protocol. When the scanner is connected to the host device, each scanned bar code (or the first stored bar code) is immediately sent to the host. If the scanner receives an ACK then it erases that bar code. If the scanner does not receive an ACK, then it saves the bar code data in memory. When the scanner is not connected, it saves the bar code data in its non-volatile memory. When a new bar code is scanned, the scanner sends the oldest bar code saved in memory. If the scanner receives the ACK, then the bar code the scanner sent is erased, and the scanner sends the next oldest bar code from memory. This "send and acknowledge" sequence continues until the scanner sends all the bar codes saved in memory. The ACK Protocol is configured with the following settings:

BCDataACK = True
SeqNum=False
AutoDownload=True

Serialization Protocol (Legacy)

The **Serialization Protocol** is similar to ACK Protocol except the scanner appends a four-character serial number to each scanned bar code it sends to the host device. The scanner does not clear the bar code from memory until it receives the serial number and ACK character from the host. The Serialization Protocol is normally used with a Bluetooth connection to make sure that the host does not receive duplicate bar codes. Without serialization, the host could receive two copies of the same bar code if a scanned bar code is sent and acknowledged, but the scanner does not receive the acknowledgment. This can happen if the user walks out of range before the scanner receives the acknowledgement from the host. When the user moves back into range, then the scanner resends the bar code. Serialization lets the host use the serial number to verify there are no duplicate bar codes. The Serialization Protocol is configured with the following settings:

BCDataACK = True
SeqNum=True

AutoDownload=True

TTY (XSP)

The XSP **TTY** (TeleTYpe) **Protocol** uses an XML-based format. The foundation of this protocol is the fact that the scanner sends data packets only when an event occurs or in response to TTY commands from the host device. An event is defined as either a scanned bar code or a new connection (either cable plug in or Bluetooth connection). When an event occurs, the scanner sends an Attention packet to the host. The Attention packet includes the first bar code stored in memory; this bar code uses the Compatibility format (Legacy protocol). When the host receives an attention packet, it responds by sending a TTY command, and the scanner responds to that TTY command. The host can send a TTY command at any time and the scanner always sends a TTY response.

The following Legacy Protocol settings are recommended when using TTY Protocol:

BCDataACK = True
SeqNum=False or True
AutoDownload=True

TTY provides fully human readable commands and responses. This protocol lets you use a terminal program to send commands to the ROV scanner and receive responses. TTY commands give you full control of your interaction with the scanner.

The most used scanner functions have dedicated commands, such as DownloadBarCodes or ClearBarCodes. Other scanner commands use the Set and Get commands with property names.

TTY commands work very well for configuring the scanner, reading scanner settings, downloading data, etc. Data sent using TTY protocol does not include CRCs. Therefore, TTY is excellent when typing commands. However, Microvision recommends that you use the more secure M2M protocol when applications interact with the scanner.

M2M (XSP)

The **M2M** (Machine to Machine) **Protocol** is identical to the TTY Protocol and uses the same Legacy Protocol settings. M2M gives you an additional level of security by adding a header and trailer to the TTY data packet. The header includes the packet length and message ID, and the trailer includes a Cyclic Redundancy Check (CRC). The header and trailer are especially valuable because they ensure the integrity of the downloaded bar code data. When the host sends an M2M command, the scanner always returns an M2M response. Microvision recommends using the M2M protocol when you create an interface within a ROV-based application.

XSP Protocol Details

Scanner XSP has three protocols it can use to send messages between the scanner and the host device: Legacy, TTY, and M2M. You use one of the Legacy protocols when using your ROV scanner with Flic-based applications.

Note: For more information about using the Legacy protocols, see Appendix 5.

XSP uses the following message types: command, acknowledgment (positive and negative), response, and data. The only difference between M2M protocol and TTY protocol is that the M2M uses a header and trailer in each message sent to and from the scanner.

Command messages are always sent from the host to the scanner.

The scanner sends acknowledgement, response, and data messages to the host in response to a command.

Commands and Command Terminators

The TTY and M2M use the same simple ASCII commands. This is the format of the ASCII commands:

    {CommandString}{CommandTerminator}

TTY commands must include a command terminator such as a carriage return (CR), or carriage return and line feed (CRLF).

M2M commands do not require the command terminator inside the M2M framing.

M2M commands do not **require** the command terminator at the end because the M2M header includes the length element. This element identifies the number of bytes in the command. However, on Bluetooth connections, the host shall add a Carriage Return to the end of the M2M message after the checksum. The extra carriage return is needed for the Scanner's Bluetooth module to flush the receive buffer. The extra Carriage Return is a not part of the XSP protocol and is not counted in the message length. For consistency, the extra Carriage Return can also be added to the M2M packet on a serial connection.

The following are examples of the same ClearBarCodes command and response in TTY and M2M format:

    TTY Command: `ClearBarCodes{CR}`

    TTY Response: `<ack dev="01234567" />{CRLF}`

    M2M Command: `0018C0001ClearBarCodes1AC4{CRLF}`

    M2M Response: `001FA0001<ack dev="01234567" />{CRLF}1BC6`

You can see in the M2M command that the ClearBarCodes command, (`ClearBarCodes{CR}`), is preceded by a series of alphanumeric characters (`0018C0001`); this is the header. You can also see the command is followed by another series of alphanumeric characters (`1AC4`); this is the trailer. The command terminator after the TTY command is optional. The command terminator after the M2M trailer is optional but recommended.

The header includes a number of elements, including message length, message type, and message identification number. The trailer is always a checksum function. This checksum function helps make

M2M more reliable than TTY for sending and receiving data. For a complete description of the checksum function, see Appendix 3 Using Checksum.

XSP lets you use long or short names for commands and properties. The short names are abbreviations of the long names. The following example shows the same ClearBarCodes command and response in M2M and TTY format using short command names:

>**TTY Command:** `ClBc{CR}`
>
>**TTY Response:** `<ack dev="01234567" />{CRLF}`
>
>**M2M Command:** `0018C0001ClBc1AC4{CR}`
>
>**M2M Response:** `001FA0001<ack dev="01234567" />{CRLF}1BC6`

Long names are provided for clarity. Short names are provided primarily for creating control bar codes. Control bar codes are Code128 bar codes with the first character set to FNC3, followed by a TTY command. The command terminator is not required for control bar codes.

As mentioned previously, both the TTY and M2M protocols use the same command, acknowledgment, response, and data elements. The TTY format for each message type is:

TTY Command Format	
{CommandString} Used to configure the scanner, read scanner properties, or request bar code data download	**{CommandTerminator}** All commands, acknowledgments, replies, and data must be followed by a command terminator. You can use any of the following as a command terminator: {Carriage Return} {Line Feed} {Carriage Return}{Line Feed}
{Acknowledgment} Positive response to a command or successful data transfer	
{NegativeAcknowledgment} Negative response to a command or to unsuccessful data transfer	
{Response} Property value sent by the scanner in response to a command	
{Data} Data sent by the scanner in response to a command from the host	

The following are examples of TTY commands, acknowledgments, and responses:

{CommandString}	`ClearBarCodes{CR}`
{Acknowledgment}	`<ack dev="3812272">Data</ack>{CRLF}`
{Negative Acknowledgment}	`<nack dev="3812272">Data</nack>{CRLF}`

{Response}	`<reply dev="3812272">`
	`    <prop name="BCCount">0</prop>`
	`</reply>`

{Data}	`<data dev="0761120012/1" id="6" dt="2007-07-29 15:10:22">`
	`<bc si="]XA" ts="2007-07-29 15:10:15" tt="OK">076183202104</bc>`
	`</data>`

The M2M format for each message type is:

<table>
<tr><td colspan="4" align="center">M2M Command Format</td></tr>
<tr>
<td>{**Header**}

Contains message length. type, and sequence number.</td>
<td>{**CommandString**}
{**Acknowledgment**}
{**NegativeAcknowlegment**}
{**Reply**}
{**Data**}

The same commands are used in both the M2M and TTY protocols. In M2M the commands must include the {Header} as a prefix and {Trailer} as a suffix.</td>
<td>{**CommandTerminator**}

Optional for M2M commands</td>
<td>{**Trailer**}

Checksum of the message.

For Bluetooth, a command terminator after the trailer is recommended to flush the Bluetooth module buffer.</td>
</tr>
</table>

Note: The code examples in this document may not be completely accurate. The Checksum values may not be correct.

The following are examples of M2M commands, acknowledgments, and responses:

{CommandString}   0018C0001ClearBarCodes1AC4{CR}

{Acknowledgment}   001FA0001<ack dev="01234567" />{CRLF}1BC6

{Negative Acknowledgment}   0020A0002<nack dev="01234567">0006</nack>{CRLF}3C46

{Response}
```
0367R0003
<reply dev="01234567">
    <prop name="UserId">IniTech</prop>
</reply>
1A4C
```

{Data}
```
00F7R001B
<data dev="0761120012/1" id="6" dt="2007-07-29 15:10:22">
<bc si="]XA" ts="2007-07-29 15:10:15" tt="OK">076183202104</bc>
</data>
1A4C
```

The header includes information about the message: message length, message type and message identification number. The body includes the command and command terminator. The trailer includes a checksum function that is used to verify that the message is delivered successfully. The following shows the individual elements of an M2M command:

`{Length}{MessageType}{PortNum}{MessageId}{CommandString}{CommandTerminator}{Checksum}`

You can use any number of white space characters, such as spacebar and tab, before and after M2M commands. You can also use white space between commands and parameters. However, you must include all white space characters when you calculate Length and Checksum values. For a complete description of the Checksum function, see Appendix 3 Using Checksum.

The combined M2M header elements use nine (9) characters. The following table shows the M2M command header elements you can use.

Header Element	Description
{Length} 4 chars	Four ASCII hexadecimal digits (0000-FFFF). The length is the total number of bytes in the message including the Trailer fields. Length does not include the 4 bytes of the Length field. The maximum M2M packet length is 65536 characters.
{MessageType} 1 char	ASCII character that identifies the message type: C – Command – XSP command string A – Acknowledgment – Positive or negative acknowledgment R – Response – XML bar code data or response packet N – Notification – Reserved for future use
{PortNum} 2 chars	Two ASCII hexadecimal digits (00-FF) that identifies the Port Number that the host establishes. NOTE: The PortId is for future use. Make sure that port is set to 00.
{MessageId} 2 chars	Two ASCII hexadecimal digits (00-FF). Message ID increments from "01" to "FF". "00" is considered an invalid sequence number.
Body Element	**Description**
{CommandString} {Acknowledgment} {NegativeAcknowledgment} {Reply} {Data}	A command sent by the host device to the scanner, a positive or negative acknowledgment sent by the host device or scanner, or a response sent by the scanner.
Trailer Element	**Description**
{Checksum} 4 chars	Four ASCII hexadecimal digits representing a checksum value. The checksum is calculated using the entire frame excluding this checksum field. The checksum value is defined by the CCITT standard 16-bit CRC.

The trailer in an M2M message is always a checksum function.

The same MessageID is included in the header of an M2M command and in the header of the response sent by the scanner. This means that commands and responses can be matched by MessageID.

These header and trailer elements are the only difference between TTY and M2M message formats. In the following sections, you can see examples of TTY and M2M commands, replies, acknowledgments, and data messages.

Flic Compatibility Response Format

Remember, the first message that your ROV scanner sends when it connects to a host device is in Compatibility format and uses the programmed Legacy protocol. The format for this message is:

```
[Prefix][Params]<BarcodeData>[SeqNum][Suffix]
```

Element	Description
[Prefix] 0 to 4 chars	Programmable Prefix (the Prefix is optional)
[Params] 0 to 4 chars	User defined or AIM Symbology Identifier (the Symbology Identifier is optional)
<BarcodeData> 1 to 45 chars	Bar code Data in ASCII format
[SeqNum] 4 chars	ASCII Characters that identify the sequence number of the data packet (the Seqnum is optional)
[Suffix] 0 to 4 chars	Programmable Suffix (the Suffix is optional)

Scanner XSP Command Formats

Commands are always sent from the host device to the scanner. The scanner cannot send commands; the scanner can only send a response, data, or acknowledgment (Ack or Nak).

The following are the TTY and M2M command formats:

TTY Command   `{CommandString}{CommandTerminator}`

M2M Command   `{Header}{CommandString}{CommandTerminator}{Trailer}{CommandTerminato`

The command element is a case-insensitive literal string for the command. The command terminator element tells the scanner when it reaches the end of a command. In M2M commands, the command terminators are optional. For Bluetooth, a command terminator after the trailer is recommended to flush the Bluetooth module buffer. You can use any of the following as a command terminator:

Command Terminator	Char Name	Keystrokes	Hex Value
{Carriage Return}	CR	Ctrl–M	0x0D
{Line Feed}	LF	Ctrl–J	0x0A
{Carriage Return}{Line Feed}	CRLF	Ctrl–M Ctrl–J	0x0D0A

In the CRLF table entry, the first carriage return (CR) terminates the command. The ROV scanner then ignores the subsequent line feed (LF) command.

To see all the commands that you can use with Scanner XSP, see the Scanner Commands chapter.

Recommended Command Formatting

To ensure proper communication with the scanner and to provide a consistency over Serial, USB, or Bluetooth interfaces, Microvision recommends the following command formatting.

- Preceed each command with a space character – this wakes up the scanner in the event it has gone to sleep
- Wait a 65ms to allow the scanner to wake up
- Send command – either TTY or M2M immediately follow by Command Terminator (CR)
- Wait for Scanner response to send next command
    - o If no response, wait 300ms then resend command sequence

Non-Printable Characters

Non-printable characters in Scanner XSP commands are replaced using the URL escape method. The non-printable character is replaced with *%dd*, where dd is the ASCII character value as two (2) hexadecimal digits. The percent character (%) itself is replaced with *%25*. A quote mark (") used as part of a string is replaced with *%22*. For example:

```
NUL  (x00)  =  %00
STX  (x02)  =  %02
```

```
ETX  (x03)  =  %03

%    (x25)  =  %25

"    (x22)  =  %22

'\'  (x5c)  =  %5C
```

Long and Short Command Names

Microvision Scanner XSP supports long and short names for commands and properties. The short names are abbreviations of the long command and property names. Short names let you create short control bar codes. All the tables in this document that show commands or properties include the long names and the short names.

TTY Command Examples

The following are sample of TTY commands and ROV scanner responses. For a complete list of commands and responses, see XSP Commands.

Set command	`Set UserId,"IniTech"{CR}`
Set response	`<ack dev="01234567"/>`
Get command	`Get UserId{CR}`

Get response	`<reply dev="01234567">`
	`   <prop name="UserId">IniTech</prop>`
	`</reply>`

ClearBarCodes command	`ClearBarCodes{CR}`
ClearBarCodes response	`<ack dev="01234567"/>{CRLF}`

M2M Command Examples

The following are examples of M2M commands and ROV scanner responses:

Set command	`0020C0001Set UserId,"IniTech"1AC4{CR}`
Set response	`0021A0001<ack dev="01234567"/>{CRLF}1BC6`
Get command	`0027C0003Get UserId1AC4{CR}`

Get response	`0367R0003`
	`<reply dev="01234567">`
	`   <prop name="UserId">IniTech</prop>`
	`</reply>`
	`1A4C`

ClearBarCodes command	`0018C0001ClearBarCodes1AC4{CR}`
ClearBarCodes response	`001FA0001<ack dev="01234567"/>{CRLF}1BC6`

You can use any number of white space characters, such as spacebar and tab, before and after M2M commands. You can also use white space between commands and parameters. However, you must include all white space characters when you calculate Length and Checksum values.

Scanner XSP Acknowledgment Formats

When a host device sends a command to your ROV scanner, the scanner responds by sending an acknowledgment, a response, or data. There are two types of acknowledgment: a positive acknowledgment and a negative acknowledgment.

A positive acknowledgment is referred to as an acknowledgment, or an Ack. An Ack is a positive response to a command. A negative acknowledgment is referred to as a Nak. A Nak is a negative response to a command.

The difference between TTY and M2M acknowledgments is that the M2M format uses headers and trailers. Remember, the type of acknowledgment received depends on the format of the Scanner XSP commands you use. If you use M2M commands, your scanner responds using M2M acknowledgments.

TTY and M2M Acknowledgment Format

Use TTY format to quickly configure the scanner for simple testing and prototyping. The TTY acknowledgment format is:

```
{XML Ack/Nack}
```

The M2M acknowledgement format is:

```
{Length}{MessageType}{PortNum}{MessageId}{XML Ack/Nack}{Checksum}
```

The following are TTY format acknowledgments:

TTY Acknowledge	`<ack dev="01234567"/>`
TTY Negative Acknowledge	`<nack dev="01234567"/>`
	`<nack dev="01234567">000A</nack>`

The Data element is optional. The scanner can use this field to return operation specific data, such as an error code in the case of a negative acknowledgment message (Nack).

The following are acknowledgments in M2M format:

M2M Acknowledge	`0055A001B`
	`<ack dev="3812272">`
	`   <data></data>`
	`</ack>`
	`1BC6`

M2M Negative Acknowledge	`0055A001B`
	`<nack dev="3812272">`
	`   <data></data>`
	`</nack>`
	`1BC6`

Scanner XSP Response Formats

When a host device sends a command to your ROV scanner, the scanner responds by sending a response, an acknowledgment, or data. When the host device sends a command to the scanner asking for a property value, the scanner sends a response. The response includes the property value requested in the command. Property values include the scanner user ID, Bluetooth timeout values, and bar code configuration settings.

The difference between TTY and M2M replies is that the M2M format uses headers and trailers. Remember, the type of response the scanner sends depends on the format of the Scanner XSP commands sent by the host. If you use M2M commands, your ROV scanner responds using M2M replies.

TTY and M2M Response Format

Use TTY format to quickly configure the scanner for simple testing and prototyping. The TTY reply format is:

    {Reply}

Microvision recommends using M2M formats when writing an application. The M2M response format is:

    {Length}{MessageType}{PortNum}{MessageId}{Reply}{Checksum}

The following is a TTY format response:

```
<reply dev="01234567">
  <prop name="UserId">IniTech</prop>
</reply>
```

The following is an M2M format response:

```
0367R0003
<reply dev="01234567">
    <prop name="UserId">IniTech</prop>
</reply>
1A4C
```

Scanner XSP Data Formats

When a host device sends a command to your scanner, the scanner responds by sending a response, an acknowledgment, or data. The scanner responds to a command using the same format used by the command. For example, when a ROV scanner receives a command in M2M format, it sends a data response in M2M format.

Microvision recommends using M2M format to create applications because M2M is more secure and is more reliable for sending and receiving data. M2M benefits include packet framing and CRC.

Non-printable character handling

Microvision Scanner XSP uses XML when it sends responses from the scanner to the host device. Scanner XSP uses the standard XML escape method for all non-printable and tag characters. The XML escape method replaces the non-printable or tag character with Unicode characters from the Unicode Private Use Area. The value of the non-printable character is added to the base value xE000.

The following are some non-printable and tag characters, and their escape equivalents:

quotation mark (")	`"`
apostrophe (')	`'`
ampersand (&)	`&`
less than (<)	`<`
greater than (>)	`>`

The following are some examples of the Unicode characters:

NUL (x00)	`xE0000`
STX (x02)	`xE0002`
ETX (x03)	`xE0003`

TTY and M2M Data Formats

Use TTY format to quickly configure the scanner for simple testing and prototyping. The TTY Data format is:

```
{Data}
```

Microvision recommends using M2M formats when writing an application. The M2M data format is:

```
{Length}{MessageType}{PortNum}{MessageId}{Data}{Checksum}
```

The purpose of the header and trailer is to find data communication errors and to help with data recovery.

This is an example of the TTY bar code data response to the TTY command DownloadBarCodes:

```
<data dev="3812272" id="10" dt="2007-06-24 11:20:06" bcc="3" ec="0">
   <bc si="]F0" ts="07-06-03 14:22:25" tt="ok">12345</bc>
   <bc si="]F0" ts="07-06-03 14:23:55" tt="ok">67890</bc>
   <bc si="]F0" ts="07-06-03 14:26:07" tt="ok">ABCDE</bc>
</data>
```

This is an example of the M2M bar code data response to the M2M command DownloadBarCodes:

```
00F7R001B
<data dev="3812272" id="10" dt="2007-06-24 11:20:06" bcc="3" ec="0">
   <bc si="]F0" ts="07-06-03 14:22:25" tt="ok">12345</bc>
   <bc si="]F0" ts="07-06-03 14:23:55" tt="ok">67890</bc>
   <bc si="]F0" ts="07-06-03 14:26:07" tt="ok">ABCDE</bc>
</data>
1A4C
```

In this example, the bar code data in the data response is between the data tags (<data></data>). The data response elements include the following:

<data>…</data>	XML bar code data. Variable length.
dev="..”	The unique device serial number in ASCII.
id="..”	Two digit download data ID 00-99. Uniquely identifies the download packet and the bar codes included in the packet. The ID is used by the host to request deletion of the bar codes from the scanner memory.
bcc="..”	Bar code count that has the same value as the BCCount property. This is the number of bar codes in Scanner memory available for download at the time of the download request. The value can be 0 to 9999.
dt="..”	Scanner internal date and time at the time of the download request. Only included if `Timestamp = "True"`. The timestamp format is: yyyy-mm-dd hh:mm:ss
ec="..”	Event Count that reflects the number that would be used to form the time type value (e#) if a bar code were to be scanned at the time of the download. Provides information for timestamp estimation in the case of loss of power or battery changes that may affect timestamps.

In this example, the individual bar codes in the data response are between the bar code tags (<bc></bc>). The bar code elements include the following:

<bc>…</bc>	XML bar code data for a single bar code. Variable length.
si=".."	The symbol identifier (optional)
	The value of si depends on the setting of the SymIdType element.
	If `SymIdType = "Off"`, there is no si element.
	If `SymIdType = "AIM"`, then the value equals the AIM symbology identifier (as specified in AIM ITS 02-2002 Data Carrier/Symbology Identifiers Maintenance Document for ISO 15424) for the associated bar code. Symbologies that are not otherwise assigned an AIM symbology code get the value of "`]X<alphanum>`" where <alphanum> is a digit or character (upper or lower case). The first two assigned values are:
	`]XA` – For UPC-A bar codes
	`]XE` – For UPC-E bar codes
	If `SymIdType = "Custom"`, then the value of the si attribute will be determined by the custom attribute assigned to the bar code symbology.
ts="yy‑mm‑dd hh:mm:ss"	The timestamp; uses a year-month-date-hour-minute-second format. Note: The ts attribute is included in the message only when the `TimeStamp=True`.
tt=".."	The time type. Note: tt attribute is included in the message only when the `TimeStamp=True`.
	Valid values:
	`ok` = Actual Time
	`na` = TimeStamps disabled when the bar code was scanned and no time data was recorded.
	`e1` = Estimated time (after one battery event)
	`e2` = Estimated time (after two battery events)
	`eN` = Estimated time (after N battery events)

XSP Timestamps

ROV scanners include a real-time clock (RTC) that tracks the current date and time. XSP data packets include the scanner time from the RTC in the "yyyy-mm-dd hh:mm:ss" format.

You can turn timestamps on or off using the TimeStamp property. When timestamps are set to True (on), a timestamp is recorded with every successful scan. The timestamp data is included in the data packet sent to the host device. Set timestamps to False if you do not want to record timestamp data with each scan.

The XSP data packet includes four time-related fields:

- dt="…" is the Download time attribute in the <data> element that includes the date and time that the scanner receives a download command. This value can be compared to the current system time for verification.

- ec="…" is the event count that is equal to the time type value that would be used for the next scanned bar code.

- `ts="…"` is the timestamp attribute in the <bc> element that includes the date and time the scanner assigns to a successful scan.

- `tt="…"` is the time type attribute in the <bc> element that provides the status or type of time.

  - `tt="OK"` means that the host has successfully set the scanner date and time, and that the scanner date and time is correct.

  - `tt = "na"` means that the timestamp was not turned on and no time data was recorded at the time of the scan.

  - `tt="e1"` means that the timestamp in the "ts" attribute is an estimate, and not necessarily the correct time and date. This occurs when there is a battery event after the date and time is sent to the scanner. The timestamp is an estimate because the scanner cannot calculate the amount of time the batteries were disconnected.

  - `tt="e2"` means that two battery events occurred since the host device sent the time to the scanner.

  - `tt="eN"` means that N number of battery events occurred since the host device sent the time to the scanner.

If you set timestamps to true when there are bar codes without timestamps in scanner memory, the bar codes without timestamps are assigned the tt="na" time type, and the corresponding attribute is empty ts="".

The scanner saves the date and time sent by the host device in EEPROM. The scanner also automatically saves the timestamp once every hour. If you set timestamps to true, then the date and time is saved with each scanned bar code.

After each scanner power cycle (battery event), the host device sends the current date and time to the ROV scanner. If the date and time are not set by the host, then the scanner defaults to "0001-01-01 00:00:00" or to the date and time of the last scanned bar code. The scanner then adds one (1) to the number of battery events since it received a new date and time from the host device. When the scanner receives a new date and time from the host device, the subsequent bar code packets include the attribute tt="ok".

These time attributes let the host recover from a power cycle. If tt="e1", then the host application can compare the download time to the host device time, and change the bar code times as necessary.

To correct "ts" values that have an "e1" type perform the following steps:

1. Save the host system time at the time of the download request.

2. Subtract the "dt" time returned from the scanner in response from the system time.

3. Add that difference to the "ts" time of each bar code with an "e1" type.

If a scanner sends bar codes with type "e2" or higher timestamps, then only the last set of bar code times can be recovered accurately. For example:

```
Host              Clearbarcodes
Scanner           <ack dev="0761120012"/>
```

Host	set time,"2007-01-01 12:00:00"
Scanner	<ack dev="0761120012"/>

Scan 3 bar codes

Host	DownloadBarcodes
Scanner	<data dev="0761120012" id="3" dt="2007-01-01 12:04:58">
	<bc si="]A0" ts="2007-09-00 12:04:34" tt="OK">AB0116560-001</bc>
	<bc si="]XE" ts="2007-09-00 12:04:41" tt="OK">04965802</bc>
	<bc si="]A0" ts="2007-09-00 12:04:48" tt="OK">1S000000077331</bc>
	</data>
Wait 1 min and Battery Event	
Scan bar code	
Wait 1 min and Battery Event	
Scan bar Code	
Host	DownloadBarcodes
Scanner	<data dev="0761120012" id="1" dt="2007-01-01 12:05:12">
	<bc si="]A0" ts="2007-09-00 12:04:34" tt="OK">AB0116560-001</bc>
	<bc si="]XE" ts="2007-09-00 12:04:41" tt="OK">04965802</bc>
	<bc si="]A0" ts="2007-09-00 12:04:48" tt="OK">1S000000077331</bc>
	<bc si="]XE" ts="2007-09-00 12:04:57" tt="e1">04965802</bc>
	<bc si="]XE" ts="2007-09-00 12:05:03" tt="e2">04965802</bc>
	</data>

The first three bar codes maintain an accurate timestamp value. The fourth scan shows a difference of only nine seconds from the third bar code, when the actual different was over one minute. The fifth bar code also shows a difference of only six seconds from the fourth bar code when it should be over one minute.

The system time of the last download request is about 12:07:00. The scanner time of the download request is 12:05:12. The difference is 1:48. Add 1:48 to the "ts" value of the last bar code and you can see that it was scanned at 12:06:51, just over two minutes after the third bar code.

The exact time of the fourth bar code is not recoverable.

Scanner XSP Commands

The following are the types of scanner commands you can use with your ROV scanner.

- Operation commands

- Property access commands (Get, GetAll, and Set)

- Null command

All Scanner XSP commands are case-insensitive. All string-type parameters in command messages are enclosed in quotation marks ("). Numeric-type parameters and keywords are not enclosed in quotation marks.

Operation Commands

The following are XSP Operation Commands you can use with your ROV scanner.

Operation Command Table		
Long Name	**Short Name**	**Description**
`DownloadBarCodes nrBarCodes`	`DnlBc`	Downloads nrBarcodes from memory.
		If Scanner has less than nrBarcodes bar codes in memory, downloads all the bar codes from Scanner memory.
		If nrBarCodes parameter is not sent, the scanner downloads all the bar codes from Scanner memory or the maximum number that fit into a 65K packet.
		If there are no bar codes, the scanner sends an XML packet that contains no data.
		If nrBarcodes is 0, the scanner sends an XML packet that contains no data.
		If there are bar codes in scanner memory, the scanner sends the bar code data to the host device in a continuous stream. The bar code data is sent in XML packets:
		`<data>…</data>`
		Property: `nrBarCodes`, the number of bar codes you want to download. Use this property if you do not want to download all bar codes in one continuous stream. For example, if you set nrBarCodes to 10, the scanner sends ten bar codes.
`DownloadBarCodes nrBarCodes, offset`	`DnlBc`	Downloads nrBarcodes from memory starting at bar code offset.
		If Scanner has less than nrBarcodes bar codes in memory starting from offset, then downloads all the bar codes to the end of Scanner memory.
		If nrBarcodes is 0, the scanner sends an XML packet that contains no data.
		If there are bar codes in scanner memory, the scanner sends the bar code data to the host device in a continuous stream. The bar code data is sent in XML packets:

		`<data>…</data>` offset, the starting point for downloading bar codes. Use this property to download bar codes out of order. The offset property allows you to download and verify all bar codes in Scanner memory before deleting.
`ClearBarCodes`	`ClBc`	The scanner erases all the bar codes from memory, and then sends an Ack. The scanner always returns an Ack, even if there are no bar codes in memory.
`DeleteBarCodes id`	`DlBc`	Deletes bar codes in memory from the data download with data id = "id". When the bar codes are deleted, the scanner sends an Ack. If there are no bar codes that match the data id, the scanner sends a Nak. **Property:** `id`, the data id of the bar codes you want to delete.
`DeleteAndDownloadBarCodes id,nrBarCodes`	`DlDnlBc`	Deletes bar codes in memory from the data download with data id = "id", and then downloads nrBarCodes. If the delete command succeeds, then the scanner sends the number of bar codes specified by nrBarCodes. If there are no bar codes to delete, the scanner sends a Nak.
`UndeleteBarCodes`	`UdlBc`	Restores bar codes deleted from scanner memory. Responds with Ack. You can undelete bar codes only after one of the following events occurs: • An autodownload when the scanner connects to the host device • You send a Legacy 'd' command • You send an XSP DeleteBarCodes command • You send an XSP ClearBarCodes command When the following condition exists, the scanner automatically downloads the bar codes, and then erases them from memory, when the scanner is connected to the host device: `AutoDownload = `**`True`** `BCDataAck = `**`False`** If the scanner cannot restore the bar codes, it sends a Nak to the host.
`Scan`	`Scan`	This command starts a scan by the scanner. The scan will stop when a bar code is successfully read or after 2 seconds.
`Signal`	`Sig`	This command makes the scanner beep with different tones or makes the LED blink. You can use a maximum of eight signals. Each signal uses four parameters in the following format: `<freq>,<len>,<ledN1>,<ledN2>,where` `<freq>` = the frequency in Hz. If the frequency is set to zero (0), the beeper is silent. `<len>` = the duration of the frequency in intervals of 0.5 ms.

		`<ledN1>` = defines the state of the green LED during <len>. 0 = Off. 1 = On
		`<ledN2>` = defines the state of the blue LED during <len>. 0 = Off. 1 = On 1
`RestoreUserSettings`	RUS	This command gets the User Settings from EEPROM and puts them into the scanner RAM. You use this command to replace the current scanner settings with the last saved user settings.
		A CommitUserSettings command is used to store User Settings.
		When the settings are restored, the scanner sends an Ack. If the scanner cannot restore the settings, the scanner sends a Nak.
`CommitUserSettings`	CUS	This command saves the current scanner settings in EEPROM as User Settings. These are the settings that the scanner puts into the scanner RAM when you send a RestoreUserSettings command.
		When the settings are saved, the scanner sends an Ack. If the scanner cannot save the settings, the scanner sends a Nak.
`RestoreFactorySettings`	RFS	This command gets the default factory settings and saves the settings in the scanner RAM.
		When the scanner saves the settings in RAM, then the scanner sends an Ack. If the scanner cannot restore the default settings, the scanner sends a Nak.
`EepromWrite Address,Data`	EWR	This command saves (writes) a single byte to the user settings area in EEPROM. You identify the address (0 – 127) where you want to save the data. You also identify the data, or byte value (0 – 255), that you want to save.
		The scanner has a 128-byte area reserved for user data.
		When the scanner saves the data to the correct address, the scanner sends an Ack. If the scanner cannot save the data to the correct address, the scanner sends a Nak.
`EepromRead Address`	ERD	This command gets (reads) the data stored at a specific address in user data area of EEPROM. You identify the address (0 – 127) from which you want to get the data.
		When the scanner finds the address, it sends the data in an XML packet with the following format:

		`<reply>`
		`   <mem addr="xxx">value</mem>`
		`</reply>`

		If the address is invalid, the scanner sends a Nak.

Property Access Commands

You use property access commands to change a scanner setting (Set command), retrieve a scanner setting (Get command), or retrieve all, or groups of, scanner settings (GetAll command).

Set Command

You use the Set command to change the scanner properties. You can set only one property with a single command. To set multiple properties, you must use multiple commands. The syntax of the Set command is:

```
Set {Property},{Value}{CommandTerminator}
```

The scanner responds to the Set command by sending an Ack or a Nak.

The scanner sends an Ack if the property in the command is set. If the scanner cannot set the property, it sends a Nak. The following are reasons why the scanner cannot set the property:

- The property is read only

- The property does not exist

- The value is invalid or in the incorrect format

When you set multiple properties, you must use multiple Set commands. Each Set command must be separated with a command terminator. The following is an example of multiple TTY commands separated by command terminators:

```
set LED,True{CommandTerminator}
set LED,1{CommandTerminator}
set Beep,False{CommandTerminator}
set Beep,0{CommandTerminator}
set UserId,"User"{CommandTerminator}
```

Note: Properties with Boolean types can be set to Boolean values *true* or *false*, or to corresponding numeric values 1 or 0, as shown in the above example.

Get Command

You can use the Get command to retrieve the current value of various scanner properties. The syntax of the Get command is:

```
Get {Property}{CommandTerminator}
```

The scanner responds to the Get command by sending an XML packet. The following is an example of a response to a Get command:

```
<reply dev="0780E00035">
<prop name="UserID" type="string">0000000000000000</prop>
</reply>
```

GetAll Command

You can use the GetAll command to retrieve the current values for a group of properties. You can use the long command (GetAll) or short command (GAll). The syntax of the GetAll command is:

```
GetAll {Property}{CommandTerminator}
```

The scanner responds to the GetAll command by sending an XML packet. The following is an example of a response to a GetAll Scanner command:

```
<reply dev="0780E00035">
<prop name="SWVersion" type="string">02.50.00</prop>
<prop name="BCCount" type="unsignedShort">0</prop>
```

```xml
<prop name="UserID" type="string">0000000000000000</prop>
<prop name="BCLimit" type="unsignedShort">9999</prop>
<prop name="BCDataAck" type="boolean">false</prop>
<prop name="Led" type="boolean">true</prop>
<prop name="Beep" type="boolean">true</prop>
<prop name="ForceSerializationOverBT" type="boolean">true</prop>
<prop name="TimeStamp" type="boolean">true</prop>
<prop name="Time" type="dateTime">0001-01-04 22:48:44</prop>
<prop name="PerfData" type="boolean">false</prop>
<prop name="SeqNum" type="boolean">false</prop>
<prop name="AutoDownload" type="boolean">true</prop>
<prop name="DownloadDelay" type="unsignedShort">0</prop>
</reply>
```

The GetAll Command Table shows the options for getting property information from your scanner.

<table>
<tr><th colspan="3" align="center">GetAll Command Table</th></tr>
<tr><th>Long Name</th><th>Short Name</th><th>Description</th></tr>
<tr><td><code>GetAll</code></td><td><code>GAll</code></td><td>This command gets all scanner properties. These properties include:
    Scanner
    Connection
    Formatting
    Symbology
    Identifier</td></tr>
<tr><td><code>GetAll Scanner</code></td><td><code>GAll</code></td><td>When you add the Scanner property, the scanner sends the general scanner properties.</td></tr>
<tr><td><code>GetAll Connection</code></td><td><code>GAll</code></td><td>When you add the Connection property, the scanner sends the scanner connection properties.</td></tr>
<tr><td><code>GetAll Formatting</code></td><td><code>GAll</code></td><td>When you add the Formatting property, the scanner sends the scanner formatting properties.</td></tr>
<tr><td><code>GetAll Symbology</code></td><td><code>GAll</code></td><td>When you add the Symbology property, the scanner sends the scanner symbology properties.</td></tr>
<tr><td><code>GetAll Identifier</code></td><td><code>GAll</code></td><td>When you add the Identifier property, the scanner sends the scanner identifier properties.</td></tr>
</table>

Null Command

You can send the XSP Null command to the scanner to stop the download process or wake the scanner. The Null command does not contain any settings or request any information. It is the only command you can send to the scanner to stop the download process. The Null command stops Legacy version auto-downloads and XSP version downloads. The scanner does not send a response to a Null command. The format of the null command is:

```
{CommandTerminator}
```

You can use the TTY Null command to:

- Wake up the scanner
- Stop a bar code download that is in progress

When the host device starts communication with the scanner, make sure to first send a Null command or a space character to ensure the scanner is not asleep. When responding to an Attention packet, or when executing a series of commands, be sure to conform to the protocol timeouts defined below.

Sample XSP Commands & Responses (TTY)

Task = Get the number of bar codes in scanner memory

Command	`Get BCCount{CR}`

Response	`<reply dev="0780E00035">`
	`        <prop name="BCCount" type="unsignedShort">10</prop>`
	`</reply> {CRLF}`

Task = Download all bar codes in one continuous stream with no acknowledgements (10 bar codes in scanner memory)

Command	`DownloadBarcodes {CR}`

Response	`<data dev="0780E00035" id="06" bcc="10" dt="2007-09-28 10:39:40" >`
	`        <bc ts="2007-09-28 10:38:15" tt="OK">123456789012</bc>`
	`        <bc ts="2007-09-28 10:38:16" tt="OK">123456789012</bc>`
	`        <bc ts="2007-09-28 10:38:19" tt="OK">01234565</bc>`
	`        <bc ts="2007-09-28 10:38:20" tt="OK">01234565</bc>`
	`        <bc ts="2007-09-28 10:38:21" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:38:22" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:38:23" tt="OK">12345670</bc>`
	`        <bc ts="2007-09-28 10:38:24" tt="OK">12345670</bc>`
	`        <bc ts="2007-09-28 10:38:28" tt="OK">720591307905</bc>`
	`        <bc ts="2007-09-28 10:38:29" tt="OK">4948382148773</bc>`
	`</data> {CRLF}`

Task = Download bar code data with XML escape characters

Command	`DownloadBarcodes{CR}`

Response	`<data dev="0780E00035" id="08" bcc="2" dt="2007-09-28 10:44:31" >`
	`        <bc ts="2007-09-28 10:43:41" tt="OK"><ABCDE></bc>`
	`        <bc ts="2007-09-28 10:43:42" tt="OK">"User"</bc>`
	`</data> {CRLF}`

Task = Download bar codes when the number of bar codes requested matches the bar codes in

scanner memory (3 bar codes in scanner memory)

Command	`DownloadBarcodes 3{CR}`

Response	`<data dev="0780E00035" id="10" bcc="3" dt="2007-09-28 10:46:19" >`
	`        <bc ts="2007-09-28 10:46:00" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:46:01" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:46:02" tt="OK">12345670</bc>`
	`</data> {CRLF}`

Task = Download bar codes when the number of bar codes requested is greater than the bar codes in scanner memory (3 bar codes in scanner memory)

Command	`DownloadBarcodes 10{CR}`

Response	`<data dev="0780E00035" id="10" bcc="3" dt="2007-09-28 10:46:19" >`
	`        <bc ts="2007-09-28 10:46:00" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:46:01" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:46:02" tt="OK">12345670</bc>`
	`</data> {CRLF}`

Task = Download bar codes when the number of bar codes requested is less than the bar codes in scanner memory (11 bar codes in scanner memory)

Command	`DownloadBarcodes 3{CR}`

Response	`<data dev="0780E00035" id="12" bcc="11" dt="2007-09-28 10:48:05" >`
	`        <bc ts="2007-09-28 10:46:00" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:46:01" tt="OK">1234567890128</bc>`
	`        <bc ts="2007-09-28 10:46:02" tt="OK">12345670</bc>`
	`</data>{CRLF}`

Task = Download bar codes when the number of bar codes requested is zero (11 bar codes in scanner memory)

Command	`DownloadBarcodes 0{CR}`
Response	`<data dev="0780E00035" bcc="11" dt="2007-09-28 10:49:10" />{CRLF}`

Task = Download bar codes when there are no bar codes in scanner memory

Command	`DownloadBarcodes 10{CR}`
Response	`<data dev="0780E00035" bcc="0" dt="2007-09-28 11:08:46" />{CRLF}`

Task = Delete and download bar codes (7 bar codes in scanner memory)

Command	`DownloadBarcodes 1{CR}`

Response	`<data dev="0780E00035" id="13" bcc="7" dt="2007-09-28 11:09:56" >`
	`        <bc ts="2007-09-28 11:09:43" tt="OK">123456789012</bc>`
	`</data>{CRLF}`

Command	`DeleteAndDownloadBarcodes 13,3{CR}`

Response	`<data dev="0780E00035" id="14" bcc="6" dt="2007-09-28 11:12:03" >`
	`        <bc ts="2007-09-28 11:09:44" tt="OK">123456789012</bc>`
	`        <bc ts="2007-09-28 11:09:46" tt="OK">01234565</bc>`
	`        <bc ts="2007-09-28 11:09:47" tt="OK">01234565</bc>`
	`</data>{CRLF}`

Task = Set or Get Boolean property set to True/False

Command	`Set DecodeC39,true{CR}`
Response	`<ack dev="01234567"/>{CRLF}`
Command	`Get DecodeC39{CR}`

Response	`<reply>`
	`   <prop name="DecodeC30">true</prop>`
	`</reply>{CRLF}`

Task = Set or Get Boolean property with a value of 1 or 0

Command	`Set DecodeC39,0{CR}`
Response	`<ack dev="01234567" />{CRLF}`
Command	`Get DecodeC39{CR}`

Response	`<reply dev="0780E00035">`
	`      <prop name="DecodeC39" type="boolean">true</prop>`
	`</reply>{CRLF}`

Task = Set and Get non-printable prefix string

Command	`Set PrefixStr,"%02"{CR}`
Response	`<ack dev="0780E00035"/>{CRLF}`
Command	`Get PrefixStr{CR}`

Response	`<reply dev="0780E00035">`
	`      <prop name="PrefixStr" type="string"></prop>`
	`</reply>{CRLF}`

Task = Set and Get "<" Prefix String

Command	`Set PrefixStr,"<"{CR}`
Response	`<ack dev="0780E00035"/>{CRLF}`
Command	`Get PrefixStr{CR}`

Response	`<reply dev="0780E00035">`
	`      <prop name="PrefixStr" type="string"><</prop>`
	`</reply>{CRLF}`

Task = Issue a Signal command

Command	`Signal 4000,100,1,0,4000,100,0,0{CR}`
Response	`<ack dev="0780E00035"/>{CRLF}`

Task = Write and Read EEPROM

Command	`Eepromwrite 123,25{CR}`

Response   `<ack dev="0780E00035"/>{CRLF}`

Command   `Eepromread 123{CR}`

Response
```
<reply dev="0780E00035">
      <mem addr="123">25</mem>
</reply>{CRLF}
```

XSP Application Properties

ROV scanner operations are controlled by the XSP application properties. You can use the Set command to configure XSP application property values. You can use one of the Get commands to retrieve the active XSP property values.

Microvision organizes XSP Application Properties into the following categories:

- General Properties
- Connection Properties
- Bar code Formatting Properties
- Symbology Properties
- Symbology Identifier Properties

XSP Application Property Defaults and Persistence

The operation of the scanner is controlled by the XSP application properties. The active application properties are stored in the active RAM data structure and in EEPROM. The active properties are saved in EEPROM so the scanner can use these properties during restarts. This means that each Set command sets the property value in RAM and in the Current Application Property Data area in EEPROM.

You can use the Set command to change the application property values. When you start the system, the application property values in RAM are initialized from one of three persistent areas of EEPROM. EEPROM uses the following hierarchy:

1. The **User Current Application Property** values stored in EEPROM.
2. The **User Default Application Property** values set using the CommitUserSettings command.
3. The **Factory Default Application Property** values.

The values saved in the EEPROM can be managed with the following commands:

- **RestoreUserSettings** – Copies the User Default Application Property values from EEPROM and stores the values in the scanner RAM.
- **CommitUserSettings** – Copies the Application Property values from RAM and saves the values in EEPROM as the User Default Application Property values.
- **RestoreFactorySettings** – Copies the Factory Default Application Property values from EEPROM and saves the values in the Active RAM data structure.

General Properties

The XSP Scanner General Properties control basic scanner functions and user interface behavior. The XSP General Properties table shows the property values you can configure.

General Properties Table		
Long Name	**Short Name**	**Description**
SWVersion	SwV	This property is a <string> that uses a maximum of 16 ASCII characters. The characters identify major version, minor version, and build number. A period (.) separates the versions. This is a read-only property. `<prop name="SWVersion" type="string">02.50.00</prop>`
UserID	UId	This property is a <string> that uses a maximum of 16 ASCII characters. You use quotation marks to enclose the string. The default value is "0000000000000000" For example: `Set UserID,"User"`

BCCount	BcC	This property uses four digits to identify the number of bar codes currently in scanner memory.
		
		The default value is zero (0).
		
		This is a read-only property.
		
		`get BCCount`
		
		`<prop name="BCCount" type="unsignedShort">6</prop>`

BCLimit	BcL	This property uses four digits to identify the maximum number of bar codes you can save in scanner memory. If you set the bar code limit to zero, you turn on Host Scan Control. The default value is 9999 `Set BCLimit,100`
BCDataAck	Ack	Set this property to **True** if you want Legacy bar code data with acknowledgements; set to **False** if you do not want Legacy bar code data with acknowledgements. The default setting for ROV Scanners is **False** The default setting for ROV Scanners with Bluetooth is **True** NOTE: In order to change this setting for ROV Scanners with Bluetooth, you must change the ForceSerializationOverBT setting to **False**. When using XSP protocols, set this property to **True**

`AutoDownload`	`ADl`	Set this property to **True** if you want bar codes to download automatically whenever the scanner connects to the host device. Set this property to **False** if you do not want auto-download.
		The default setting is **True**
		When using XSP protocols, set this property to **True**
`DownloadDelay`	`DlD`	This property lets you set the delay (in milliseconds) between bar codes downloaded using a Legacy protocol.
		The default value is zero (0)
		Valid values are 0, 500, 1100, 1600
		`set downloaddelay,500`
`SeqNum`	`Seq`	Set this property to **True** if you want Legacy bar code data with sequence numbers. Set this property to **False** if you do not want Legacy bar code data with sequence numbers.
		The default setting for ROV Scanners is **False**
		The default setting for ROV Scanners with Bluetooth is **True**
		NOTE: In order to change this setting for ROV Scanners with Bluetooth, you must change the ForceSerializationOverBT setting to **False**.
`ForceSerializationOverBT`	`FSOBT`	Set this property to **True** if you want the ROV Scanner with Bluetooth to use Serialization over the Bluetooth connection. If True, then regardless of the Ack and Seq property settings:
		`BCDataAck = True`
		`SeqNum = True,.`
		The default setting is **True**.
		If you want the scanner with Bluetooth to use the current Ack and SeqNum settings, set this property to **False**.
`TimeStamp`	`TSt`	Set this property to **True** if you want to add a timestamp to XSP bar code data packets. If you do not want a timestamp with each bar code, set this property to **False**.
		The default setting is **True**.
`LED`	`LED`	Set this property to **True** if you want to enable the green LED and blue LED. If you want the LEDs disabled, set this property to **False**.
		The default setting is **True**.
`Beep`	`Beep`	Set this property to **True** if you want to enable the scanner beep. If you want the scanner to operate silently, set this property to **False**.
		The default setting is **True**.
`Time`	`Time`	This property is a <string> of 19 ASCII characters that sets the date and time into the scanner. You use quotation

| | | marks to enclose the string. This string uses the following format:

`"yyyy–mm–dd hh:mm:ss"`

`set time,"2007–07–30 09:37:00"` |

Connection Properties

The XSP Scanner Connection Properties specify the parameters the ROV scanner uses to connect to a host device with a serial/USB cable or Bluetooth wireless connection. The XSP Connection Properties table shows the property values you can configure.

XSP Connection Properties Table		
Long Name	**Short Name**	**Description**
`BaudRate`	`BRate`	This property sets the baud rate the scanner uses to communicate with the host device. The default setting is 4800. Valid values are 4800, 9600, 19200, 38400, 57600, 115200 `set baudrate,4800`
`BtName`	`BtN`	This property is a <string> that uses a maximum of 16 ASCII characters to identify the Bluetooth user. You use quotation marks to enclose the string. The default setting is **Cordless**. For example: `Set BtName,"BtUser"`
`BtPIN`	`BtPn`	This property is a <string> that uses a maximum of 16 ASCII characters to identify the Bluetooth PIN number. You use quotation marks to enclose the string. The default setting is "0000" `Set BtPIN,"1234"`
`BtPartner`	`BtPt`	This property is a <string> that uses 12 ASCII characters to identify the Bluetooth device with which the scanner uses as a partner. You use quotation marks to enclose the string. The default setting is `"000000000000"`
`BtModule`	`BtM`	Set this property to **True** if you want the power in the Bluetooth module turned on. Set this property to **False** if you want the Bluetooth Module turned off. The default setting is **True**. This command is provided so that a Bluetooth host can turn off the Bluetooth module of the scanner. Used for Bluetooth batch applications. To turn on the Bluetooth module the user can press and hold the scanner button for 5 seconds or scan a control bar code with the command: Set BtM,1

BtDiscoverTimer	BtDT	This property sets the interval (in seconds) that the scanner waits to be discovered before the Bluetooth module shuts down. You use four ASCII digits and the maximum interval is 9999 seconds (approximately 2 hours 47 minutes). The default setting is **300 seconds** (five minutes).
BtConnectTimer	BtCT	This property sets the interval (in seconds) that the scanner waits for a Bluetooth connection to the host device before the scanner shuts down. You use four ASCII digits and the maximum interval is 9999 seconds (approximately 2 hours 47 minutes). The default setting is **60 seconds** (one minute).
BtRetryTimer	BtRT	This property sets the interval (in seconds) that the scanner waits after the connection is broken before it tries to make a Bluetooth connection. The scanner only tries to reconnect if scans are stored in scanner memory. You use four ASCII digits and the maximum interval is 9999 seconds (approximately 2 hours 47 minutes). The default setting is **3600 seconds** (60 minutes). If you set this property to zero, the scanner never tries to connect after the connection is broken.

After you turn off the Bluetooth module, you can turn on the module using one of the following methods:

- Scan the Set BtModule,True control bar code
- Send a Set BtModule,True command to the scanner using the serial port
- Press and hold the Scan button for five (5) seconds

A battery event does not turn on the Bluetooth module. After the Bluetooth module is turned on using one of these methods, the Bluetooth module initializes and switches to discoverable mode.

Bar Code Format Properties

The XSP Scanner Bar Code Format Properties specify how the bar code data is formatted when it is sent to the host. The XSP Bar Code Format Properties table shows the property values you can configure.

XSP Bar Code Format Table		
Long Name	**Short Name**	**Description**
Prefix	Pfx	Set this property to **True** if you want to add a prefix to the bar code data. If you set the property **False** and try to add a prefix string, the scanner sends a Nak. The default setting is **True**.
PrefixStr	PStr	You can add a prefix string to a bar code only when the Prefix property is set to **True**. This property adds a prefix string to the bar code data for Legacy protocols. The prefix uses a maximum of four (4) ASCII characters enclosed in quotation marks. Non printable characters are represented using hex code. The default setting is <STX> or "%02"

		For example to set the prefix to equal STX, type:
		set Prefix,"%02"
Suffix	Sfx	Set this property to **True** if you want to add a suffix to the bar code data. If you set the property **False** and try to add a suffix string, the scanner sends a Nak.
		The default setting is **True**.
SuffixStr	SStr	You can add a suffix string to a bar code only when the Suffix property is set to **True**. This property adds a suffix string to the bar code data in Legacy protocols. The suffix uses a maximum of four (4) ASCII characters enclosed in quotation marks. Non printable characters are represented using hex code.
		The default setting is CRLF
		For example to set the suffix to equal CRLF:
		set Suffix,"%0D%0A"
SymIdType	SIdT	This property identifies the symbology identifiers the scanner uses. Symbology identifiers are codes added to the bar code data to specify the type of bar code scanned.
		Valid values are **Off**, AIM, or Custom.
		The default setting is **Off**.

Symbology Properties

The XSP Scanner Symbology Properties specify which symbol set the scanner uses, and any specific Symbology settings.

The XSP Symbology Properties table shows the values you can configure.

Symboloby Properties table			
Long Name	**Short Name**	**Value** (default in bold)	**Description**
DecodeUPCA_EAN13	DUpAE13	**True**|False	UPC_A and EAN13 Decode On or Off
DecodeUPCE	DUpE	**True**|False	UPC_E Decode On or Off
DecodeEAN8	DE8	**True**|False	EAN8 Decode On or Off
DecodeC128	DC128	**True**|False	Code128 Decode On or Off
DecodeC39	DC39	**True**|False	Code39 Decode On or Off
DecodeITF	DITF	True|**False**	Code 2of5 (ITF) Decode On or Off
DecodeRSSL	DRL	True|**False**	RSS Limited Decode On or Off
DecodeRSSS	DRS	True|**False**	RSS Stacked Decode On or Off
DecodeRSS14	DR14	True|**False**	RSS_14 Decode On or Off
DecodeGS1_128	DG128	**True**|False	GS1_EAN128 (UCC) Decode On or Off
DecodeCodabar	DCB	True|**False**	Codabar Decode On or Off
VerifyC39CheckSum	VC39Ck	True|**False**	Verify Code39 Checksum On or Off
StripC39CheckSum	SC39Ck	True|**False**	Strip Code39 Checksum On or Off Note: Strip On means Verify & Strip
StripUPCCheckSum	SUpCk	True|**False**	Strip UPC Checksum On or Off
VerifyITFCheckSum	VITFCk	True|**False**	Verify ITF Checksum On or Off
StripITFCheckSum	SITFCk	True|**False**	Strip ITF Checksum On or Off Note: Strip On means Verify & Strip
DecodeSupps	DSup	**True**|False	Decode Supplemental Bar codes
RequireSupps	RqSup	True|**False**	Require Supplemental Bar codes
CombineSupps	CmSup	True|**False**	Combine Supplemental Bar codes with base bar code
ExpandUPCEToUPCA	ExA	True|**False**	Expand UPCE to UPCA
ExpandUPCAToEAN13	Ex13	True|**False**	Expand UPCA to EAN13
ExpandUPCEANTo14	Ex14	True|**False**	Expand UPC or EAN to 14
ITFMinLen	ITFMnL	**0** to 44	ITF (I2of5) bar code Minimum Length
ITFMaxLen	ITFMxL	0 to **44**	ITF (I2of5) Bar code Maximum Length
CodabarMinLen	CBMnL	**0** to 44	Codabar bar code Minimum Length
CodabarMaxLen	DUpAE13	0 to **44**	Codabar bar code Maximum Length

Symbology Identifier Properties

You can use the Symbology Identifier Properties to configure the Custom Symbology Identifiers for each Bar code Symbology that Microvision supports.

The XSP Symbology Identifier Properties table shows the values you can configure.

XSP Symbology Identifier Table		
Long Name	**Short Name**	**Description**
`SymIdUPCA`	`SIdUpA`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for UPC_A. The default value is **"A"**
`SymIdUPCE`	`SIdUpE`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for UPC_E. The default value is **"E"**
`SymIdEAN8`	`SIdE8`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for EAN_8 . The default value is **"FF"**
`SymIdEAN13`	`SIdE13`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for EAN_13. The default value is **"F"**
`SymIdDig2Supp`	`SidSup2`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for 2 Digit Supplemental. The default value is the same as the base bar code. For example: if the base bar code is a UPC-A, the value transmitted will be **"A"**.
`SymIdDig5Supp`	`SidSup5`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for Digit5 Supplemental. The default value is the same as the base bar code. For example: if the base bar code is a UPC-A, the value transmitted will be **"A"**.
`SymIdC128`	`SIdC128`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for Code 128. The default value is **"f"**
`SymIdC39`	`SIdC39`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for Code 39. The default value is **"a"**
`SymIdITF`	`SIdITF`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for Code 2of5 (ITF). The default value is **"b"**
`SymIdRSS14`	`SIdR14`	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for RSS-14.

		The default value is **"r"**
SymIdRSSL	SIdRL	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for RSS Limited. The default value is **"r"**
SymIdRSSS	SidRS	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for RSS Stacked. The default value is **"r"**
SymIdGS1_128	SIdG128	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for UCC/EAN-128 (GS1-128). The default value is **"f"**
SymIdCodabar	SIdCB	This property uses a <string> with a maximum of four ASCII characters as the Custom Symbology ID for Codabar. The default value is **"c"**

If you combine supplemental bar codes, then the supplemental bar codes do not their own symbology ID. Instead, the base bar code's symbology ID applies to both the base and the supplemental bar code. If the supplemental bar codes are not combined, then any UPC or EAN supplemental uses the same custom symbology ID for any UPC or EAN variant.

Protocol Specifications

The protocol specifications describe the behavior of the protocol, namely, the message sequences and timeout values.

The following three general properties control the basic operating behavior of the scanner:

- BCLimit (Bar code Limit): This parameter sets the number of bar codes that can be stored in memory, range 0 – 9999 (the maximum number of scans stored by the scanner is approximately 4000 UPC-A bar codes, but this number will vary based on type of bar code and use of timestamps).

- BCDataAck: This parameter sets whether downloaded bar code data packets must be acknowledged by the host

- AutoDownload: This parameter sets whether the bar code data is downloaded automatically when the scanner connects

The Protocol Specification table shows how these three settings control the operating behavior of your scanner.

Protocol Specification Table				
Scanner Behavior	BCLimit	BCDataAck	AutoDownload	Description
Classic Scanner Emulation	`1-9999`	`False`	`True`	These are the factory default settings for ROV Scanner. When the scanner is connected to the host device, it starts downloading bar codes in a continuous stream. The scanner does not wait for the host device to send an acknowledgment before sending the next bar code. This configuration is the same as the behavior of a Flic Scanner.
Classic Bluetooth Scanner Emulation	`1-9999`	`True`	`True`	These are the factory default settings for ROV Scanner with Bluetooth. When the scanner connects to a host or scans a bar code, the scanner sends the first bar code in memory. The scanner then waits for an acknowledgement from the host. If the scanner receives a Legacy protocol acknowledgement (with Serialization), the scanner deletes that bar code and sends the next bar code in memory. If the host sends an XSP response instead of a Legacy response, the scanner responds in XSP. This configuration is the same as the Flic Cordless Scanner with the addition of XSP support.
Reliable Transfer	`1-9999`	`True`	`True`	Same as Classic Bluetooth Scanner

				Emulation with two differences. • Does not require Serialization • This applies to either Standard or Bluetooth scanners
Host Scan Control	0	True	True	Host scan control sends an alert to the user when a proper connection to an application is established. When the scanner is connected (either cable or Bluetooth), scanned data is captured and sent to the host. When the host responds with an acknowledgement, the scanner emits a second "good" beep, which indicates the data was received. If the scanner does not receive an acknowledgement from the host application within three seconds, the scanner emits a "bad" beep. This may indicate the connection has been lost, or that the application has detected an error, and requires user intervention. Bar codes that are not acknowledged are not saved by the scanner.
Polled Batch	1-9999	NA	False	When you set Autodownload to **False**, the scanner ignores connection events. If the scanner is connected after scanning and storing a batch of bar codes, then the scanner does not send out an attention packet. To retrieve data from the scanner, the host must send a DownloadBarcodes command, or the user can scan another bar code to generate an Attention packet.

Note: The Host Scan Control setting for the ROV scanner is different from the setting of the same name for Flic scanners. To emulate Flic Host Scan Control with a ROV scanner, set BCLimit to 1 (one), BCDataAck to True, and AutoDownload to True.

Timeouts

The XSP protocol uses two timeouts:

- Inter-character timeout (5 seconds)
- Message response timeout (3 seconds)
- Scanner wake timeout (4 seconds)

The *inter-character timeout* is the maximum interval between two characters in a message. If the scanner does not receive the next character in a message before the end of the inter-character timeout, the receive buffer is flushed. The characters received before the timeout are dropped and the system waits for next message.

The *message response timeout* refers to the maximum interval allowed before the ROV scanner receives the first byte after a response. If the host does not receive a response during the inter-message timeout, the requestor resends the message packet using the same sequence number. If the sender does not receive a response before the end of the timeout to the second request message, then the sender considers the communication failed.

During a connection oriented or reliable transfer, the scanner sends each bar code and then waits for an acknowledge message from the host. If the scanner does not receive an acknowledge message from the host during the inter-message timeout then the scanner resends the bar code data packet (with same sequence number). If there is no response to the resent packet, then the scanner stops sending bar code data. The scanner then attempts to send the bar codes again only when the ROV scanner reconnects or when the scanner receives a download request from the host.

The Scanner w*ake timeout* refers to the maximum period of inactivity before the scanner goes into low-power (sleep) mode. To prevent the scanner from going to sleep during a slow TTY communication session, any character received from the host resets the wake timer.

Consequently, when manually entering TTY commands with a terminal emulator (such as HyperTerminal) the user cannot pause for more than four (4) seconds between characters. A four-second delay between characters causes the Inter-character Timeout to expire and makes the scanner go into sleep mode.

Scanner Wakeup

As described in the Timeout section, the ROV scanner goes into a low-power (sleep) mode if there is no activity during the Wake timeout. The first character received while the scanner is in sleep mode wakes the scanner. However, the first character is lost and not received by the Scanner. Because of this, the host device must send an extra wake-up character to the scanner.

When you use XSP commands, Microvision recommends that the scanner is first awakened by sending a space character. After sending a space character, the scanner wake timeout is reset to four (4) seconds and the scanner is ready to receive TTY or M2M commands.

Spontaneous Scanner to Host Messages

When the ROV scanner completes a scan, it sends the scanned bar code data to the host in Flic Compatiblity format and protocol according to the scanner settings. The bar code data the ROV scanner sends also acts as an attention packet to the host for XSP communication.

Host to Scanner Commands

The host device sends commands to your ROV scanner. Your ROV scanner sends data responses and acknowledgments to the host device. ROV scanners do not send commands to the host device.

Positive Acknowledgments (Ack) and Negative Acknowledgments (Nak)

Both your ROV scanner and the host device send positive acknowledgments (Ack) and negative acknowledgments (Nak).

The scanner sends an Ack when:

- The command sent to the scanner has a valid syntax

   AND

- The command sent to the scanner is supported

   AND

- The requested operation is completed successfully

The scanner sends a Nak when:

- The command sent to the scanner has an invalid syntax

   OR

- The command sent to the scanner is not supported or is not implemented in the current context

   OR

- The requested operation is not completed successfully

The optional error code associated with a Nak is a 16-bit integer, represented with four hexidecimal digits. The currently supported error codes are defined in the Error Codes table below.

Error code (HEX)	Error Name	Description
0000	NO_ERROR	
0001	ERROR_UNKNOWN	Unspecified error.
0002	ERROR_COMMAND_NOT_SUPPORTED	The command is not supported, or is unknown to the scanner. Unsupported commands always return: ERROR_COMMAND_NOT_SUPPORTED, when the command exists but is not activated, not

		implemented, unknown, or invalid.
0003	ERROR_PARAMETER_INVALID_FORMAT	A parameter has the wrong format.
0004	ERROR_PARAMETER_MISSING	A required parameter is not specified.
0005	ERROR_PARAMETER_INVALID_DATATYPE	The parameter type is invalid.
0006	ERROR_PARAMETER_ILLEGAL_VALUE	The command includes a parameter value that is illegal.
0007	ERROR_PARAMETER_OUT_OF_RANGE	A parameter is outside the valid range for this parameter,
0008	ERROR_PARAMETER_NOT_SUPPORTED	A parameter is not supported by this scanner.
0009	ERROR_PARAMETER_LENGTH_EXCEEDED	The length of the given parameter was too long,
000A	ERROR_INVALID_STATE	The requested operation is not allowed or cannot be completed in the current state of the scanner.

Acknowledgment and Negative Acknowledgment Examples

TTY Format (successful Set command):

```
Host:   Set UserId,"IniTech"{CR}
Scanner: <ack dev="01234567"/>
```

M2M Format (successful Set command):

```
Host: 0020C0001Set UserId,"IniTech"1AC4{CR}
Scanner: 0021A0001<ack dev="01234567"/>{CRLF}1BC6
```

TTY Format (successful Set command, commands are case in-sensitive):

```
Host:   set userID,"IniTech"{CR}
Scanner: <ack dev="01234567"/>{CR}
```

M2M Format (successful Set command, commands are case in-sensitive):

```
Host: 0020C0002set userID,"IniTech"1AC4{CR}
Scanner: 0021A0002<ack dev="01234567"/>{CRLF}1BC6
```

TTY Format (successful Get command):

```
Host:   Get UserId{CR}
Scanner:
<reply dev="01234567">
     <prop name="UserId">IniTech</prop>
</reply>
```

M2M Format (successful Get command):

```
Host: 0027C0003Get UserId1AC4{CR}
Scanner:
0367R0003
<reply dev="01234567">
     <prop name="UserId">IniTech</prop>
</reply>
1A4C
```

TTY Format (Invalid command, 0002 = ERROR_COMMAND_NOT_SUPPORTED):

```
Host:   Set UserId,"IniTech"{CR}
Scanner: <nack dev="01234567">0002</nack>{CRLF}
```

M2M Format (Invalid command, 0002 = ERROR_COMMAND_NOT_SUPPORTED):

```
Host: 0027C001BSett UserId,"IniTech"1AC4{CR}
Scanner: 0011A001B<nack dev="01234567">0002</nack>{CRLF}1BC6
```

TTY Format (Invalid command, ERROR_PARAMETER_MISSING):

```
Host:   Get{CR}
Scanner: <nack dev="01234567">0004</nack>{CRLF}
```

M2M Format (Invalid command, ERROR_PARAMETER_MISSING):

```
Host: 0027C001BGet1AC4{CR}
Scanner: 0011A001B<nack dev="01234567">0004</nack>{CRLF}1BC6
```

TTY Format (valid command, ERROR_PARAMETER_INVALID_DATATYPE):

```
Host:   Set BCLimit,"InitTech"{CR}
Scanner: <nack dev="01234567">0005</nack>{CRLF}
```

M2M Format (valid command, ERROR_PARAMETER_INVALID_DATATYPE):

```
Host: 0027C001BSet BCLimit,"IniTech"1AC4{CR}
Scanner: 0011A001B<nack dev="01234567">0005</nack>{CRLF}1BC6
```

TTY Format (valid command, ERROR_PARAMETER_OUT_OF_RANGE):

```
Host:   Set BCLimit,15000{CR}
Scanner: <nack dev="01234567">0006</nack>{CRLF}
```

M2M Format (valid command, ERROR_PARAMETER_OUT_OF_RANGE):

```
Host: 0027C001BSet BCLimit,150001AC4{CR}
Scanner: 0011A001B<nack dev="01234567">0006</nack>{CRLF}1BC6
```

TTY Format (Invalid parameter, ERROR_PARAMETER_NOT_SUPPORTED):

```
Host:   Set UsirId,"IniTech"{CR}
Scanner: <nack dev="01234567">0008</nack>{CRLF}
```

M2M Format (Invalid parameter, ERROR_PARAMETER_NOT_SUPPORTED):

```
Host: 0027C001BSet UsirId,"IniTech"1AC4{CR}
Scanner: 0011A001B<nack ev="01234567">0008</nack>{CRLF}1BC6
```

TTY Format (Invalid parameter, ERROR_PARAMETER_LENGTH_EXCEEDED):

```
Host:   Set UserId,"This is way too long User Identification"{CR}
Scanner: <nack dev="01234567">0009</nack>{CRLF}
```

M2M Format (Invalid parameter, ERROR_PARAMETER_LENGTH_EXCEEDED):

```
Host: 0027C001BSet UsirId,"This is way too long User
Identification"1AC4{CR}
Scanner: 0011A001B<nack dev="01234567">0009</nack>{CRLF}1BC6
```

Protocol User Scenarios

This chapter includes sample user scenarios for ROV scanners.

- Legacy Scanner Emulation (Default) with unreliable bar code transfer

- Reliable bar code transfer

- Host scan control

User Scenario: Legacy Scanner Emulation (Default) with unreliable bar code transfer

BCLimit = **1 - 9999**, BCDataAck = **False** , AutoDownload = **True**

Connected to Host:

On Scan: (always send the bar code and then delete it)

```
If (BC_Last – BC_First < BCLimit) {
  Beep GoodScan;
  Store bar code (BC_Last++);
  Send bar code to Host (BC_First);
  Free Storage (BC_First++);
} else {
    Beep MemoryFull;
}
```

On Host Command:

```
Respond to Command;
```

Not Connected to Host:

On Scan: (always store bar code until memory full)

```
If (BC_Last – BC_First <= BCLimit) {
  Beep GoodScan;
  Store bar code (BC_Last++);
} else {
    Beep MemoryFull;
}
```

On Connection: (Download and delete all bar codes unless interrupted by Host Command)

```
While (BC_First < BC_Last) {
  Send bar code to Host (BC_First);
  if (Host Command) {
      Respond to Command;
  }
  Free Storage (BC_First++);
}
```

User Scenario: Reliable bar code transfer

BCLimit = **1 - 9999**, BCDataAck = **True**, AutoDownload = **True**

Connected to Host:

On Scan: (Store bar code, send it, wait for ACK)

```
If (BC_Last – BC_First <= BCLimit) {
  Beep GoodScan;
  Store bar code (BC_Last++);
  Send bar code to Host (BC_First);
} else {
    Beep MemoryFull;
}
```

On Host Command: (if ACK, delete bar code and send next bar code)

```
If (Host Response == ACK) {
  Free Storage (BC_First++);
  Send bar code to Host (BC_First);
} else{
  Respond to Command;
}
```

Not Connected to Host:

On Scan: (Store bar codes until memory full)

```
If (BC_Last – BC_First <= BCLimit) {
  Beep GoodScan;
  Store bar code (BC_Last++);
} else {
    Beep MemoryFull;
}
```

On Connection: (Download first bar code, wait for ACK…)

```
While (BC_First < BC_Last) {
  Send bar code to Host (BC_First);
  If (Host Response == ACK) {
    Free Storage (BC_First++);
  } else{
    Respond to Command;
  }
}
```

User Scenario: Host Scan Control

Host Scan Control configuration sends an alert to the user whether or not the scanner is connected to the host. After a scan, if the user gets a second "good" beep, then the host received and accepted the bar code. If the host does not respond, then the scanner emits a "bad" beep, which alerts the user to investigate. If the scanner does not detect a connection to a host, then it emits a "not connected" beep alerting the user to investigate.

BCLimit = 0 (Host should acknowledge each bar code)

- Connected to Host:

    - On Scan:

      ```
      Beep GoodScan;
      Store bar code (BC_First = BC_Last);
      Send bar code to Host (BC_Last);
      if (!Host Response) {
        Beep BadBeep;
        Free Storage (BC_First++; BC_Last++);
      } else if (Host Response == ACK) {
        Beep GoodBeep;
        Free Storage (BC_First++; BC_Last++);
      } else {
        Respond to Command;
      }
      ```

    - On Host Command

      ```
      Respond to Command;
      ```

- Not Connected to Host:

    - On Scan:

      ```
      Beep BadBeep;
      ```

    - On Connection:

      ```
      Nop;
      ```

Control Bar codes

The Microvision Legacy and Plus Mode Control bar codes are obsolete and no longer supported by Microvision. The new Control Bar Codes are encoded as Code128 Character Set A or B with Fnc3 as the first character. Control bar codes use the same format as the XSP TTY Set commands:

The Syntax of the Set command is:

```
Set {Property},{Value}{CommandTerminator}
```

All Set commands have an equivalent control bar code. Each control bar code can have only one command. ROV scanners do not download a control bar code string to host as a normal bar code. A successful beep is generated for a valid command interpretation and an "invalid command beep" is generated for an incorrect control bar code string.

Note: To reduce the length of Control Bar codes, Boolean properties can be set with numeric values 1 and 0 instead of True and False. Also, to reduce length, short command and parameter names are used.

Appendix 1 Abbreviations

Abbreviation	Definition
AIM	Association for Automatic Identification and Mobility
ASCII	American Standard Code for Information Interchange
BCS	Bar Code Scanner
BT	Bluetooth
CR	Carriage Return (0x0D)
EAN	European Article Number
EEPROM	Electrically Erasable Programmable Read only memory
GS1	Global Standard 1
ITF	Interleaved 2 of 5 Symbology
LED	Light Emitting Diode
LF	Line Feed (0x0A)
M2M	Machine-to-Machine protocol
MEMS	Micro-Electro-Mechanical System
RSS	Reduced Space Symbology
SDK	Software Development Kit
TTY	Tele TYpewriter (Human-to-Machine protocol)
UPC	Universal Product Code
XML	eXtensible Markup Language
XSP	eXtensible Scanner Protocol

Appendix 2 Legacy Commands

Microvision supports the following commands to give you backward compatibility with Legacy Flic Applications. The Flic Legacy commands are only supported in TTY (Human-Machine Language) format.

Flic Legacy Command Format

The Flic Legacy commands are supported by the ROV scanner for application backwards compatibility. The Flic Legacy command format is as follows:

    {LegacyCommandCode}{CR|ETX}

A Legacy Command may also include optional data, in which case the command format is:

    {LegacyCommandCode} {Data}{CR|ETX}

The Legacy Command table shows the commands that you can use

Legacy Command Table			
Cmd Code	**Cmd Name**	**Scanner Response**	**Cmd Description & Notes**
Any	Wakeup	None	Any single character will wakeup the system.
W	Who	Configuration Info	The response format is specified in a table below.
Sxxxx	Send data	None	Transmit device format and configuration data. The data format is specified in table below.
C	Clear	None	Clear All Bar code data from the device
D	Download	Bar code Data in TTY format	Download stored bar code data
T	Signal Good Scan	None	LED & PIEZO enabled based on user setting
T0			LED & PIEZO enabled based on user setting
T1			LED False, PIEZO True
T2			LED True, PIEZO False
T3			LED True, PIEZO True
F	Factory Settings	None	Restore Factory Settings
F1			Restore Factory Settings

Ixxxx	Initial Time	None	Set Initial Time in format IYYMMDDHHMMSS
BM	Get/Set Bluetooth Pairing Mode	1 or 2 (Current Pairing Mode)	1 = Normal Pairing, any host can pair with proper PIN code. 2 = Strong pairing. Only communicate with the current host.
BM1		ACK	
BM2		ACK	
BT	Read Bluetooth Timers	Timer values: $T_{Discovery}$ $T_{Connect}$ T_{Sleep}	
BTLxxxx or BTDxxxx	Set Discoverable Timer: $T_{Discovery}$	ACK	Specifies the amount of time that an unpaired system will remain in discoverable mode before shutting down. The parameter is four ASCII characters representing a 16-bit data value with a resolution of 1 second per bit.
BTCxxxx	Set Connection Timer: $T_{Connect}$	ACK	Specifies the amount of time that the system will remain connected to a Bluetooth host without any communications activity. The parameter is four ASCII characters representing a 16-bit data value with a resolution of 1 second per bit. A value of 0000 will prevent the system from shutting down due to inactivity.
BTSxxxx	Set Bsleep/Retry Timer: T_{Sleep}	ACK	Specifies the amount of time that the system will periodically try to connect to a host and to download stored bar codes. The parameter is four ASCII characters representing a 16-bit data value with a resolution of 1 second per bit. A value of 0000 will disable this feature.
BX	Reset Bluetooth Module	ACK	Returns all of the internal Bluetooth parameters to their original value

Legacy W Command Response Format

The legacy W command returns the device configuration data you see in the W Command Response Format table.

W Legacy Command Response Format Table			
Start address	**Stop address**	**Description**	**User Access**
0	7	User ID Data	R/W
8	B	Device Configuration Data	R/W
C	F	Device ID	R
10	11	Hardware Version	R
12	14	Firmware Version	R

A sample response for user ID A123B456C789D012, default configuration data, device ID 3664, Hardware version 1.7, Firmware version 2.0.0 would look like this:

```
Microvision® xxxx® Bar code Scanner, ID: 00003664, FW: 2.0.0<CR><LF>

   <SOH>WA123B456C789D01269000000000003664010702000<CR><LF>
```

Legacy S Command Data Format

The S command data is formatted as you see in the Legacy S Command Data Format table:

Legacy S Command Data Format		
Address or Bit	**Function**	**Flic and ROV Default**
0/0 .. 7/7	User ID String	00000000
Address 8 Bit 0	Flic: Beep Enable 0=Beep,Off ; 1=Beep, **True**	1 (On)
Address 8 Bits 1 & 2	TimeStamp Mode: B2 B1 0 0 Flic: TimeStamp **False** (default) ROV: TimeStamp,Off 0 1 Flic: TimeStamp **True** – store timestamp for every valid bar code ROV: TimeStamp,True 1 0 Flic: TimeStamp **True** – store timestamp for every button press ROV: N/A - timestamp on button press, not supported	Flic: 00 (Off) ROV: 01 (On)

	1 1 Flic: (reserved) TimeStamp **False** – bits automatically cleared ROV: N/A	
`Address 8 Bit 3`	Flic: LED Enable	1 (On)
	ROV: 0=LED,False ; 1=LED, **True**	
`Address 8 Bit 4`	Flic: STX at start of bar code in compatibility mode	1 (send STX)
	ROV: 0 = Prefix,Off 1 = Prefix,On; PrefixStr,"%02"	
`Address 8 Bit 5`	Flic: Send AIM code	0 (Off)
	ROV: 0=SymIdType,Off ; 1= SymIdType,AIM	
`Address 8 Bit 6`	Flic: LF Enable	1 (send CR/LF)
	ROV: Works with Bit 9/1 9/1 8/6 0 0 Suffix,On; SuffixStr,"%0D" (CR) 0 1 Suffix,On; SuffixStr,"%0D%0A" (CRLF) 1 0 Suffix=On, SuffixStr="%03" (ETX) 1 1 N/A	01 (CRLF)
`Address 8/7`	Flic: ACK/NAK Enable	0 (Off)
	ROV: 0=Ack,Off; 1=Ack=On	TBD: 0 for Connected 1 for BT
`Address 9 Bit 0`	Flic: Download format (0 for compatibility mode, 1 for native/XML	0 (compatible)
	ROV: N/A (Deprecated)	0
`Address 9 Bit 1`	Flic: ETX Enable	0 (Disabled)
	ROV: Works with Bit 9/1 9/1 8/6 0 0 Suffix,On; SuffixStr,"%0D" (CR) 0 1 Suffix,On; SuffixStr,"%0D%0A" (CRLF) 1 0 Suffix=On, SuffixStr="%03" (ETX) 1 1 N/A	01 (CRLF)
`Address 9 Bit 2`	Flic: Disable automatic bar code	0 (Enabled)

	download over RS-232	
	ROV: 0=AutoDownload,On 1=AutoDownload,Off	0 (On)
`Address 9 Bit 3`	Flic: Enable NCR identifiers	0 (Disabled)
	ROV: 0=SymIdType,Off; 1=SymIdType,Custom	0 (Off)
`Address 9 Bit 4`	Flic: Expand UPC-E bar codes to UPC-A format	0 (Don't expand)
	ROV: 0= ExpandUPCEToUPCA,Off 1= ExpandUPCEToUPCA,On	0 (Off)
`Address 9 Bit 5`	Flic: CTS Hardware Handshake	0 (Disabled)
	ROV: N/A	
`Address 9 Bit 6`	Flic: Code 39 and ITF stripped check character	0 (Disabled)
	ROV: 0=StripC39CheckSum,Off 1=StripC39CheckSum,On NOTE: Applies to Code39 only, not ITF, see StripITFCheckSum	0 (Off)
`Address 9 Bit 7`	Flic: Code 39 and ITF checksum enable	0 (Disabled)
	ROV: 0=VerifyC39CheckSum,Off 1=VerifyC39CheckSum,On NOTE: Applies to Code39 only, not ITF, see VerifyITFCheckSum	0 (Off)
`Address A/0`	Flic: Disable Code 39 decoding	0 (Enabled)
	ROV: 0=DecodeC39,On; 1=DecodeC39,Off	0 (On) NOTE: Reverse Polarity
`Address A Bit 1`	Flic: Disable Code 128 decoding	0 (Enabled)
	ROV: 0=DecodeC128,On; 1=DecodeC128,Off	0 (On) NOTE: Reverse Polarity
`Address A Bit 2`	Flic: Disable UPC-E decoding	0 (Enabled)
	ROV: 0=DecodeUPCE,On; 1=DecodeUPCE,Off	0 (On) Note: Reverse Polarity
`Address A Bit 3`	Flic: Disable EAN-8 decoding	0 (Enabled)
	ROV: 0=DecodeEAN8,On; 1=DecodeEAN8,Off	0 (On) NOTE: Reverse Polarity
`Address A Bit 4`	Flic: Disable UPC-A/EAN-13 decoding	0 (Enabled)

	ROV: 0=DecodeUPCA; 1=DecodeUPCA,Off NOTE: Applies to UPCA Decode only, not EAN-13	0 (On) NOTE: Reverse Polarity
Address A Bit 5	Flic: Enable combined bar codes	0 (Disabled)
	ROV: 0=CombineSupps,Off; 1=CpmbineSupps,On	0 (Off)
Address A Bit 6	Flic: Require supplemental bar codes	0 (Not required)
	ROV: 0=RequireSupps,Off 1=RequireSupps,On	0 (Off)
Address A Bit7	Disable decoding of supplemental bar codes	0 (Enabled)
	ROV: 0=DecodeSupps,On 1=DecodeSupps,Off	0 (On) NOTE: Reverse Polarity
Address B Bit 0	Flic: Enable bar code serialization	0 (Disabled)
	ROV: 0=SeqNum,Off; 1=SeqNum,On	TBD: **False** for Connected **True** for BT
Address B Bit 1	Flic, ROV: Reserved for internal use only	0
Address B Bit 2	Flic: Expand UPC-A bar codes to EAN-13 format	0 (Don't expand)
	ROV: 0= ExpandUPCAToEAN13,Off 1= ExpandUPCAToEAN13,On	0 (Off)
Address B Bit 3	Flic: Disable Code ITF decoding (not in Bluetooth)	0 (Enabled)
	ROV: 0= DecodeITF,On; 1= DecodeITF,Off	0 (On) NOTE: Reverse Polarity
Address B Bit 5, Bit 4	Flic, ROV: Bar code Delay B/5 B/4 Delay 0 0 none 0 1 500 ms 1 0 1100 ms 1 1 1600 ms	00 (Disabled)
Address B Bit 6	Disable automatic download over	0 (Enabled)

	Bluetooth	
Address B Bit 7	Flic: CTS Hardware Handshake Active Level	0 (Active High)
	ROV: N/A (Deprecated)	0

The following Flic legacy commands are obsolete and are no longer supported:

Obsolete Legacy Commands	
Address or Bit	**Description**
F2	Set NCR Factory Defaults
MI	Set Minimum ITF Decode Length (Function is supported by the new extended TTY commands.)
MX	Set Maximum ITF Decode Length (Function is supported by the new extended TTY commands.)
T5	Turn off Bluetooth module (This command not supported in X.36. Functionality is supported by the new extended TTY commands.)

Legacy Wakeup Behavior

To support backward compatibility with Flic Scanners, Flic wakeup behavior is supported by Scanner XSP. When you use a ROV scanner with a Flic-based application, the system wakes up after receipt of any initialization character. If the initialization character is followed by a single '*W*' character (case insensitive, no command terminator), the '*W*' response TTY format banner is sent to the host to confirm that communications is established.

If a "*W*" is not received within 4 seconds after the initialization character, the command buffer will be flushed and the scanner will return to low-power sleep mode.

Appendix 3 Using Checksum

The checksum value gives you an extra measure of data security and validation. The checksum is the sum of all the character values transmitted. Some data modes include checksum values. An application (or the XSP) calculates the checksum value when the data is read and then compares the calculated value with the checksum value sent. If the values match then all the data was received properly. If the values are different, then an error occurred when the data was sent or received. In the following example, XML data stream the check sum value is B7 (at the end of data):

Flic v: 1.9.7<CR><LF>

<?xml version="1.0" encoding="us-ascii" standalone="yes"?><CR><LF>

<upload><CR><LF>

<!-- Microvision Flic Bar code Scanner --><CR><LF>

<device type="Flic" id="3812272E" hwv="1.0" fwv="1.9.7" ud="0000000000000000" it="040106080206"><CR><LF>

<tag type="bc" dt="0327" ct="E" bc="111111111117"/><CR><LF>

<tag type="bc" dt="01" ct="E0" bc="2222222222222"/><CR><LF>

<tag type="bc" dt="01" ct="E" bc="03333333"/><CR><LF>

<tag type="bc" dt="01" ct="C0" bc="4444444"/><CR><LF>

<tag type="bc" dt="02" ct="A0" bc="555"/><CR><LF>

<tag type="bc" dt="01" ct="E4" bc="66666660"/><CR><LF>

<tag type="t" dt="08"/><CR><LF>

</device><CR><LF>

</upload><CR><LF>

chk: B7!<CR><LF>

The checksum (the chk value in the last line) is the modulo 256 sum of the characters in the transmission. The calculation starts with (and includes) the v in the first line and ends with the colon (:) character after the chk in the final line. Sum the ASCII value of all the data character, divide that number by 256, and the remainder will be the checksum.

--------------------It is either the above or the below algorithm------------------

Appendix 4 C AMS Standard Cyclic Redundancy Check - CRC16

In several places a Cyclic Redundancy Check (CRC) is used for error detection in data transport and storage. Except where indicated we use the CRC calculation below which is based on 16 bit words and is referred to as CRC16. This information was taken from Coding for Error Detection in AMS Data Transport -- CRC16, drafted in September 1997 by V. Koutsenko. It appears that the polynomial matches the "X.25" standard.

Essentials

"The encoding procedure accepts an (n-16)-bit data block and generates a systematic binary (n,n-16) block code by appending a 16-bit Frame Check Sequence (FCS) as the final 16 bits of the codeblock."

$$FCS(x) = [x^{16} * M(x) + x^{(n-16)} * L(x)] \text{ modulo } G(x)$$

$M(x)$ is the (n-16)-bit data to be encoded expressed as a polynomial with binary coefficients, $L(x) = x^{15} + x^{14} + ... + x^2 + x + 1$ (all "1" polynomial), $G(x) = x^{16} + x^{12} + x^5 + 1$ is the generator polynomial, All addition operators are the modulo 2 additions (Exclusive OR).

Error Detection

The error detection syndrome $S(x)$ will be zero if no error is detected.

$$S(x) = [x^{16} * C'(x) + x^n * L(x)] \text{ modulo } G(x)$$

$C'(x)$ is the received block in polynomial form. Big Endian bits and bytes order.

References

"Telemetry Concept and Rationale", CCSDS 100.0-G-1, <u>Green Book, Consultative Committee for Space Data Systems,</u> December 1987, Annex D, "Telemetry Transfer Frame Error Detection Encoding/Decoding Guideline".

"Advanced Orbiting Systems, Network and Data Links: Summary of Concept, Rationale, and Performance", CCSDS 700.0-G-3, <u>Green</u>

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	NULL	35	23	#	70	46	F	105	69	i
1	01	SOH	36	24	$	71	47	G	106	6A	j
2	02	STX	37	25	%	72	48	H	107	6B	k
3	03	ETX	38	26	&	73	49	I	108	6C	l
4	04	EOT	39	27	'	74	4A	J	109	6D	m
5	05	ENQ	40	28	(	75	4B	K	110	6E	n
6	06	ACK	41	29	)	76	4C	L	111	6F	o
7	07	BEL	42	2A	*	77	4D	M	112	70	p
8	08	BS	43	2B	+	78	4E	N	113	71	q
9	09	HT	44	2C	,	79	4F	O	114	72	r
10	0A	LF	45	2D	−	80	50	P	115	73	s
11	0B	VT	46	2E	.	81	51	Q	116	74	t
12	0C	FF	47	2F	/	82	52	R	117	75	u
13	0D	CR	48	30	0	83	53	S	118	76	v
14	0E	SO	49	31	1	84	54	T	119	77	w
15	0F	SI	50	32	2	85	55	U	120	78	x
16	10	DLE	51	33	3	86	56	V	121	79	y
17	11	DC1	52	34	4	87	57	W	122	7A	z
18	12	DC2	53	35	5	88	58	X	123	7B	{
19	13	DC3	54	36	6	89	59	Y	124	7C	|
20	14	DC4	55	37	7	90	5A	Z	125	7D	}
21	15	NAK	56	38	8	91	5B	[	126	7E	~
22	16	SYN	57	39	9	92	5C	\	127	7F	
23	17	ETB	58	3A	:	93	5D	]	128	80	
24	18	CAN	59	3B	;	94	5E	^			
25	19	EM	60	3C	<	95	5F	_			
26	1A	SUB	61	3D	=	96	60	`			
27	1B	ESC	62	3E	>	97	61	a			
28	1C	FS	63	3F	?	98	62	b			
29	1D	GS	64	40	@	99	63	c			
30	1E	RS	65	41	A	100	64	d			
31	1F	US	66	42	B	101	65	e			
32	20	SP	67	43	C	102	66	f			
33	21	!	68	44	D	103	67	g			
34	22	"	69	45	E	104	68	h			

Appendix 5 Legacy Protocols

This section describes the formats for commands, acknowledgements, and responses you use with the Flic Compatibility Protocol. You use this protocol only

Flic Compatibility Acknowledgment Format

Use the Flic Compatibility format to work with Flic-based applications. You can use Flic Compatibility acknowledgments with or without sequence numbering.

Flic Compatibility Acknowledgment

The following are examples of Flic Compatibility acknowledgments:

Acknowledge: `ACK = Ctrl-F = 0x06`

Negative Acknowledge: `NAK = Ctrl-U = 0x15`

Flic CompatibilityAcknowledgment with Sequence Numbering

The following are examples of Flic Compatibility acknowledgments when sequence numbering is True:

Acknowledge: `XXXXCtrl-F`

Negative Acknowledge: `XXXXCtrl-U`

Where *XXXX* is the four-digit ASCII sequence number assigned to the acknowledgment and the negative acknowledgment.