

**SIMPLE SERIAL
INTERFACE
PROGRAMMER'S GUIDE**

SIMPLE SERIAL INTERFACE PROGRAMMER'S GUIDE

72E-40451-03

Revision A

May 2015

No part of this publication may be reproduced or used in any form, or by any electrical or mechanical means, without permission in writing from Zebra. This includes electronic or mechanical means, such as photocopying, recording, or information storage and retrieval systems. The material in this manual is subject to change without notice.

The software is provided strictly on an “as is” basis. All software, including firmware, furnished to the user is on a licensed basis. Zebra grants to the user a non-transferable and non-exclusive license to use each software or firmware program delivered hereunder (licensed program). Except as noted below, such license may not be assigned, sublicensed, or otherwise transferred by the user without prior written consent of Zebra. No right to copy a licensed program in whole or in part is granted, except as permitted under copyright law. The user shall not modify, merge, or incorporate any form or portion of a licensed program with other program material, create a derivative work from a licensed program, or use a licensed program in a network without written permission from Zebra. The user agrees to maintain Zebra’s copyright notice on the licensed programs delivered hereunder, and to include the same on any authorized copies it makes, in whole or in part. The user agrees not to decompile, disassemble, decode, or reverse engineer any licensed program delivered to the user or any portion thereof.

Zebra reserves the right to make changes to any software or product to improve reliability, function, or design. Zebra does not assume any product liability arising out of, or in connection with, the application or use of any product, circuit, or application described herein.

No license is granted, either expressly or by implication, estoppel, or otherwise under any Zebra Technologies Corporation, intellectual property rights. An implied license only exists for equipment, circuits, and subsystems contained in Zebra products.

Revision History

Changes to the original manual are listed below:

| Change | Date | Description |
|------------|--------|---|
| -03 Rev. A | 5/2015 | This guide was updated to include SSI Command changes. This guide is electronic only and replaces p/n 72-40451-02. Added CMD_ACK_ACTION; changed all references of EAN-128 to GS1-128; various additional modifications. |

Table of Contents

| | |
|------------------------|-----|
| Revision History | iii |
|------------------------|-----|

About This Guide

| | |
|-----------------------------|------|
| Introduction | xi |
| Chapter Descriptions | xii |
| Notational Conventions..... | xii |
| Related Documents | xii |
| Service Information..... | xiii |

Chapter 1: Introduction to SSI

| | |
|--|------|
| Introduction | 1-1 |
| Serial Parameter Settings | 1-2 |
| Hardware Signals | 1-2 |
| Hardware Handshaking | 1-3 |
| Host Transmission to Decoder | 1-3 |
| Host Transmission Sample Code | 1-3 |
| Decoder Reception of Host Transmission | 1-4 |
| Decoder Reception Sample Code | 1-4 |
| Decoder Transmission to Host | 1-4 |
| Decoder Transmission Sample Code | 1-5 |
| Host Reception of Decoder Transmission | 1-6 |
| Host Reception Sample Code | 1-6 |
| Software Handshaking | 1-7 |
| Transfer of Decode Data | 1-7 |
| ACK/NAK Enabled and Packeted Data | 1-7 |
| ACK/NAK Enabled and Unpacketed ASCII Data | 1-7 |
| ACK/NAK Disabled and Decode Data of Any Type | 1-7 |
| Unsolicited ACK/NAK | 1-7 |
| Expected Responses | 1-8 |
| Message Packets | 1-9 |
| Multipacketing | 1-9 |
| Multipacketing, Option 1 | 1-9 |
| Multipacketing, Option 2 | 1-9 |
| Multipacketing, Option 3 | 1-10 |
| Packet Format | 1-10 |

Chapter 2: SSI Commands

| | |
|------------------------------|------|
| Introduction | 2-1 |
| SSI Command Lists | 2-1 |
| ABORT_MACRO_PDF | 2-6 |
| AIM_OFF | 2-7 |
| AIM_ON | 2-8 |
| BEEP | 2-10 |
| CAPABILITIES_REQUEST | 2-13 |
| CAPABILITIES_REPLY | 2-14 |
| BATCH_DATA | 2-17 |
| Bar Code String | 2-17 |
| BATCH_REQUEST | 2-18 |
| CHANGE_ALL_CODE_TYPES | 2-19 |
| CMD_ACK | 2-20 |
| CMD_ACK_ACTION | 2-22 |
| CMD_NAK | 2-24 |
| CUSTOM_DEFAULTS | 2-27 |
| DECODE_DATA | 2-28 |
| EVENT | 2-40 |
| FLUSH_MACRO_PDF | 2-42 |
| FLUSH_QUEUE | 2-43 |
| ILLUMINATION_OFF | 2-44 |
| ILLUMINATION_ON | 2-45 |
| IMAGE_DATA | 2-46 |
| IMAGER_MODE | 2-48 |
| LED_OFF | 2-49 |
| LED_ON | 2-50 |
| PAGER_MOTOR_ACTIVATION | 2-51 |
| PARAM_DEFAULTS | 2-52 |
| PARAM_REQUEST | 2-53 |
| PARAM_SEND | 2-56 |
| REPLY_REVISION | 2-58 |
| REQUEST_REVISION | 2-59 |
| SCAN_DISABLE | 2-60 |
| SCAN_ENABLE | 2-61 |
| SLEEP | 2-62 |
| SSI_MGMT_COMMAND | 2-63 |
| START_SESSION | 2-64 |
| STOP_SESSION | 2-65 |
| VIDEO_DATA | 2-66 |
| WAKEUP | 2-68 |

Appendix A: Transaction Examples**Appendix B: Mandatory Parameter**

About This Guide

Introduction

The Simple Serial Interface (SSI) Programmer's Guide provides system requirements and programming information about Zebra's Simple Serial Interface, which enables decoders (e.g., SE955 scan engine, hand-held scanners, 2D scanners, etc.) to communicate with a serial host.

Chapter Descriptions

Topics covered in this guide are as follows:

- *Chapter 1, Introduction to SSI* provides an overview of SSI, including signal lines, protocol, and packeting information. Protocol layers are described in a “bottom-up” manner, from the hardware layer up through software handshaking.
- *Chapter 2, SSI Commands* describes each command supported by SSI.
- *Appendix A, Transaction Examples* illustrate sample transactions.
- *Appendix B, Mandatory Parameter* describes the Parameter Scanning option which must be used with each product using SSI.

Notational Conventions

The following conventions are used in this document:

- “User” refers to anyone using an SSI product.
- “You” refers to the End User, System Administrator or Programmer using this manual as a reference for SSI
- *Italics* are used to highlight the following:
 - Chapters and sections in this and related documents
 - Dialog box, window and screen names
 - Drop-down list and list box names
 - Check box and radio button names
- **Bold** text is used to highlight the following:
 - Key names on a keypad
 - Button names on a screen.
- bullets (•) indicate:
 - Action items
 - Lists of alternatives
 - Lists of required steps that are not necessarily sequential
- Sequential lists (e.g., those that describe step-by-step procedures) appear as numbered lists.

Related Documents

Refer to the Product Reference Guide for your product for product-specific information on SSI.

For the latest version of this guide and all guides, go to: <http://www.zebra.com>.

Service Information

If you have a problem with your equipment, contact Zebra Technologies support for your region. Contact information is available at: <http://www.zebra.com>.

When contacting support, please have the following information available:

- Serial number of the unit
- Model number or product name
- Software type and version number

Zebra responds to calls by e-mail, telephone or fax within the time limits set forth in service agreements.

If your problem cannot be solved by Zebra Technologies support, you may need to return your equipment for servicing and will be given specific directions. Zebra is not responsible for any damages incurred during shipment if the approved shipping container is not used. Shipping the units improperly can possibly void the warranty.

If you purchased your Zebra business product from a Zebra business partner, please contact that business partner for support.

Chapter 1 Introduction to SSI

Introduction

This chapter describes the system requirements of the Simple Serial Interface (SSI), which provides a communications link between Zebra Technologies decoders and a serial host. Information is provided from the perspective of both the host and the decoder.

The following must be understood before using SSI:

- The SSI interface provides a means for the host to control the decoder.
- SSI is a half-duplex communication protocol.
- SSI is transaction-based, that is, the host commands and the decoder responds. For example, the host commands “beep the beeper” and the decoder both beeps and “ACKs” as a response. Acknowledgments are vital for maintaining synchronization.

The following sections describe the basic hardware layer (signals and handshaking) first, followed by software protocol, and finishing with a description of message packets.

Serial Parameter Settings

For communication to occur serial parameters must match between the host and the decoder. These parameters can be set using information in the Product Reference Guide supplied with your decoder.

The default parameters are:

- Baud Rate: 9600 Baud
- Data Bits: 8 bits
- Number of parity bits: 1 bit
- Parity: None
- Stop Bits: 1
- Hardware Handshaking: Always
- Software Handshaking: On
- Inter-Packet Delay: 0 milliseconds
- Multi-Packet Option: Option 1

Settings for other parameters related to image capture, video capture and other decoder performances should also be reviewed before using SSI.

Hardware Signals

The hardware layer of SSI consists of four signals: Transmit Data (TXD), Receive Data (RXD), Request to Send (RTS) and Clear to Send (CTS).

From the decoder's perspective:

- TXD: Serial data transmit output. Drives the serial data receive input of the host.
- RXD: Serial data receive input. Driven by the serial data transmit output of the host.
- RTS: Drives host CTS, Decoder Output. Acknowledges host demand to transmit.
- CTS: Driven by HOST RTS. Decoder Input. Host Demand / Interrupt to receive.

From the host's perspective:

- HOST RXD: Serial data receive input. Driven by the serial data transmit output of the decoder.
- HOST TXD: Serial data transmit output. Drives the serial data receive input of the decoder.
- HOST CTS: Driven by decoder RTS. Host Input. Decoder ACK of host demand to transmit.
- HOST RTS: Host demand / interrupt to transmit. Must be honored by decoder.

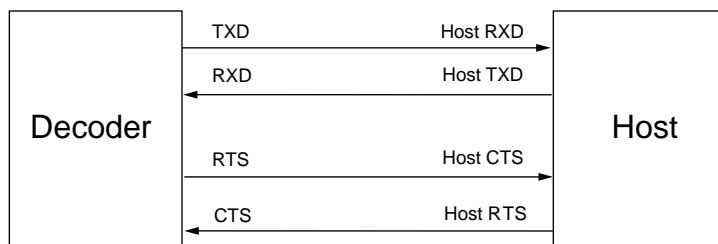


Figure 1-1 Host/Decoder Interconnection

Hardware Handshaking

The four hardware signals are used to perform host transmission to decoder, and decoder transmission to host. The host is the Bus Master and controls the actions of SSI. In cases of collision or arbitration, the decoder always defers to the actions of the host. The host programmer should adhere to these specifications closely.

Host Transmission to Decoder

When the host wants to transmit data to the decoder:

1. The host changes the state of the HOST RTS line from inactive to active.
2. The decoder replies by changing the state of the HOST CTS line from inactive to active, within the time frame set by the response timeout parameter.
3. Upon detecting the active HOST CTS, the host transmits the message on the HOST TXD line, verifying that HOST CTS remains active on a character-by-character basis.
4. After sending all characters in the message, the host must change the state of the HOST RTS line from active to inactive.
5. The decoder responds by changing the state of the HOST CTS line from active to inactive.

If the decoder does not respond with HOST CTS when it detects HOST RTS within the programmable (via parameters) response timeout, the host may retry the message. Only after multiple failure attempts spread over some period of time should the host declare the interface 'non-viable'. The decoder may be performing a time-intensive function which precludes talking to the host, resulting in the failure of the HOST CTS to go active.

Only one message per HOST RTS/CTS handshake is recognized, so if the host sends two (or more) messages without changing the state of the HOST RTS line, the decoder ignores the second (and subsequent) messages. Once the number of characters indicated by the host is received (plus the two bytes required for a checksum), further characters are ignored. The host may briefly toggle the state of the RTS line during this transaction, as long as the inter-character timeout is not exceeded.

While the host is not required to continuously stream characters, the maximum character-to-character delay cannot exceed the host Intercharacter Timeout parameter. If this timeout is exceeded, the decoder waits for the host to de-assert HOST RTS. Once the HOST RTS is in-active, the decoder issues a NAK message, which causes the host to re-try the entire message.

Host Transmission Sample Code

```
boolean host_transmit()
  request permission to send [HOST RTS active]
  WHILE (the last character has not been sent) DO
    set up serial response time out
    WHILE (permission has not been granted [HOST CTS is inactive]) DO
      IF (serial response time out expired) THEN
        remove request to send /* transmit failed */
        /* calling function may retry transmit */
        RETURN (FALSE)
      END
    END
    transmit a character
  END
  remove request to send [HOST RTS inactive]
  RETURN (TRUE) /* transmit successful */
```

Decoder Reception of Host Transmission

The decoder constantly monitors the CTS line for activity:

1. When CTS is made active by the host, the decoder responds by making the RTS line active, even if the decoder was attempting to transmit, deferring to host action.
2. The decoder monitors the RXD line and receives the characters comprising the message.
3. When all characters are received (length of message plus the two byte checksum) the decoder acts upon the message/command and ignores further characters.
4. When the decoder detects that CTS is inactive, it makes RTS inactive and transmits the response.

If the host exceeds the inter-character timeout delay, the decoder waits for de-assertion of the CTS line to send a NAK message to the host.

There are two cases where the host may make the CTS line inactive before the message is complete. If the host did not send any characters, there is no message, so the decoder does not reply. If, however, the first character is sent, an incomplete message results in a NAK.

Decoder Reception Sample Code

This pseudo code assumes that the receiving is enabled.

```
void decoder_receive()
  IF (host is requesting to send [CTS active]) THEN
    give host permission to send [RTS active]
    WHILE (no characters received) DO
      IF (host not requesting to send [CTS Inactive]) THEN
        remove host's permission to send
        RETURN /* NULL xmit - do not NAK */
      END
    END
    set up host character time out
    WHILE (not timed out AND not the last character) DO
      IF (a character was received) THEN
        reset host character time out
      END
    END
    WHILE (host is requesting to send [CTS active]) DO
      wait /* for host to end handshake */
    END
    remove host's permission to send [RTS inactive]
    process received message and prepare response
  END
  RETURN
```

Decoder Transmission to Host

The decoder is sometimes required to send data to the host. This transaction is more complicated than the host transmission.

The decoder first ensures that the host is not attempting to send by examining the state of CTS. If the host is attempting to send, the decoder defers transmission and permits the host to send as described previously. The response to the host transmission is given by the decoder before the transmission pending. For example, if the

decoder is sending decode data (as a result of scanning a bar code) and the host interrupts with a beep the beeper message, the beeper message is acted upon and acknowledged prior to transmitting the decode data message.

The host may temporarily hold off the decoder transmission by making the HOST RTS line active until it is ready to receive. If the host does not send any characters, the decoder resumes transmission when the HOST RTS line becomes inactive. If, however, the host sends one or more characters, the decoder resends the entire packet from the start when the HOST RTS line becomes inactive.

If the host is not trying to send, the decoder sends one character of the transmission on the TXD line. The decoder again checks the state of the CTS line, so the host may interrupt on a character-by-character basis. While the host should try to avoid interrupting the decoder transmission, bus collisions are possible and must defer to the host.

Decoder Transmission Sample Code

```

boolean decoder_xmit()
  /* insure that the host is not trying to send something */
  IF (host is requesting to send [CTS active]) THEN
    /* host attempt to send..grant permission */
    enable receiving
    give host permission to send [RTS active]
    set up serial response time out
    WHILE (host is still requesting to send [CTS active]) DO
      IF (character was received OR timed out) THEN
        /* abort transmission..try again later */
        RETURN (FALSE)
      END
    END
    disable receiving
    remove host's permission to send [RTS inactive]
  END
  WHILE (there are characters to send) DO
    IF (host is not requesting to send [CTS inactive]) THEN
      send next character
    ELSE
      /* the host is either holding us off or beginning */
      /* a transmission to the decoder */
      enable receiving
      give host permission to send [RTS active]
      WHILE (host is still requesting to send [CTS active]) DO
        IF (character was received) THEN
          /* beginning of a host transmission */
          /* abort transmission..try again later */
          RETURN (FALSE)
        END
      END
      /* by virtue of code flow to here the host */
      /* was merely holding off the decoder */
      disable receiving
      remove host's permission to send [RTS inactive]
    END /* resume transmit */
  END
  RETURN (TRUE)

```

Host Reception of Decoder Transmission

The host must be ready to receive data from the decoder anytime the host is not transmitting. The host can temporarily hold off the decoder transmission by keeping the HOST RTS line asserted. The host can also interrupt an incoming message by asserting HOST RTS.

Since the decoder does not change the state of the HOST CTS line when transmitting, the host is aware of a decoder transmission only when it receives the first character. For each subsequent character, the host sets up an intercharacter timeout. If this timeout has not expired and the host has not received the last character of the transmission, the host receives the next character.

Host Reception Sample Code

```
void host_receive()
  IF (a character has been received) THEN
    set up intercharacter time out
    WHILE (not timed out AND not the last character) DO
      IF (host can receive right now) THEN
        deassert HOST RTS /* in case host was holding off decoder */
        IF (a character was received) THEN
          reset intercharacter time out
        END
      ELSE
        IF (host wants to send to decoder) THEN
          RETURN /* so host can transmit */
        ELSE
          /* host does not want to send but needs some time */
          assert request to send [HOST RTS active] /*hold off
          decoder */
          set up new intercharacter time-out
        END
      END
    END
    process received message and prepare response
  RETURN
END
RETURN
```

Software Handshaking

Software handshaking provides an ACK/NAK response for commands that do not have a natural response. For example, the command “tell me your parameters” is followed by the response “my parameters are X”. A “start a decode session” command, however, has no natural response, so software handshaking provides an ACK/NAK response.

ACK/NAK handshaking may be enabled (default) or disabled. If enabled, all packeted messages must have a response in the form of an ACK, or a NAK of various types. We recommend this handshaking remain enabled to provide feedback to the host.

Raw decode data and the WAKEUP command do not use ACK/NAK handshaking since they are not packeted data.

Transfer of Decode Data

The Decode Data Packet Format parameter controls how decode data is sent to the host. When this parameter is enabled, the data is sent in a DECODE_DATA packet. When disabled, data is transmitted as raw ASCII data.



NOTE When decode data is transmitted as raw ASCII data, ACK/NAK handshaking does not apply even if it is enabled.

ACK/NAK Enabled and Packeted Data

The decoder sends a DECODE_DATA message after a successful decode. The decoder waits for a programmable time-out for a CMD_ACK response. If it does not receive the response, the decoder tries to send two more times before issuing a host transmission error. If the decoder receives a CMD_NAK from the host, it may attempt a retry depending on the cause field of the CMD_NAK message.

ACK/NAK Enabled and Unpacketed ASCII Data

The decoder sends a RAW DATA message after a successful decode. No ACK/NAK handshaking occurs even though it is enabled because data is unpacketed.

ACK/NAK Disabled and Decode Data of Any Type

The decoder sends the decode data. No response is expected from the host since ACK/NAK handshaking is disabled. The security of this transaction is not guaranteed.

Unsolicited ACK/NAK

An unsolicited ACK or NAK is an unexpected message, and is ignored since the decoder can not interpret the message.

CMD_NAK Cancel is a special case of transmission by the host, so is considered a solicited message. This message halts (and discards) an unwanted transmission by the decoder. For example, a large image sent by the decoder to the host is typically multi-packeted, and consists of a large transmission. The host may cancel this by transmitting a CMD_NAK Cancel.

Expected Responses

The following tables list allowable decoder and host responses.

Table 1-1 *Decoder Responses to Host Transmission*

| Host Transmission | Allowable Decoder Responses |
|------------------------|-----------------------------|
| AIM_OFF | CMD_ACK / CMD_NAK |
| AIM_ON | CMD_ACK / CMD_NAK |
| BATCH_REQUEST | BATCH_DATA / CMD_NAK |
| BEEP | CMD_ACK / CMD_NAK |
| CAPABILITIES_REQUEST | CAPABILITIES_REPLY |
| CHANGE_ALL_CODE_TYPES | CMD_ACK/CMD_NAK |
| CMD_ACK | None |
| CMD_ACK_ACTION | None |
| CMD_NAK | None |
| CUSTOM_DEFAULTS | CMD_ACK / CMD_NAK |
| FLUSH_QUEUE | CMD_ACK / CMD_NAK |
| ILLUMINATION_OFF | CMD_ACK / CMD_NAK |
| ILLUMINATION_ON | CMD_ACK / CMD_NAK |
| IMAGER_MODE | CMD_ACK / CMD_NAK |
| LED_OFF | CMD_ACK / CMD_NAK |
| LED_ON | CMD_ACK / CMD_NAK |
| PAGER_MOTOR_ACTIVATION | CMD_ACK / CMD_NAK |
| PARAM_DEFAULTS | CMD_ACK / CMD_NAK |
| PARAM_REQUEST | PARAM_SEND |
| PARAM_SEND | CMD_ACK / CMD_NAK |
| REQUEST_REVISION | REPLY_REVISION |
| SCAN_DISABLE | CMD_ACK / CMD_NAK |
| SCAN_ENABLE | CMD_ACK / CMD_NAK |
| SLEEP | CMD_ACK / CMD_NAK |
| SSI_MGMT_COMMAND | SSI_MGMT_COMMAND or CMD_NAK |

Table 1-1 Decoder Responses to Host Transmission (Continued)

| Host Transmission | Allowable Decoder Responses |
|-------------------|---|
| START_SESSION | CMD_ACK / CMD_NAK Note that once the decoder gathers the appropriate data, it sends this data unsolicited. |
| STOP_SESSION | CMD_ACK / CMD_NAK |
| WAKEUP | None |

Table 1-2 Host Responses to Decoder Transmission

| Decoder Transmission | Allowable Host Responses |
|----------------------|--------------------------|
| CAPABILITIES_REPLY | None |
| CMD_ACK | None |
| CMD_NAK | None |
| DECODE_DATA | CMD_ACK / CMD_NAK * |
| EVENT | CMD_ACK / CMD_NAK * |
| IMAGE_DATA | CMD_ACK / CMD_NAK * |
| PARAM_SEND | None |
| REPLY_REVISION | None |
| VIDEO_DATA | CMD_ACK / CMD_NAK |

* Multipacketed data; the host may ACK/NAK only the last packet of a multi-packeted message. Intermediate packets get no response. Intermediate packets always have the continuation bit set (1). The last packet has the continuation bit cleared (0). See *Multipacketing on page 1-9* for multi-packeting options.

Message Packets

All communications between the host and the decoder are exchanged in the form of packets. A packet is a collection of bytes framed by the proper SSI protocol formatting bytes. The maximum length of a packet is 257 bytes, consisting of a checksum (two bytes), a header (four bytes), and up to 251 characters of data. Note that the length field in the header does NOT include the length of the checksum, but DOES include the length of the header itself.

Multipacketing

SSI supports multiple packets for one message for cases when size is insufficient to transfer a complete message. Bit1 of the status byte in the message header is set to one for all packets except the last to indicate another packet is to follow. In the last packet, this bit is set to zero. The host must re-assemble these packets into one message. The decoder sends each packet in order.

Multipacketing, Option 1

The host ACK/NAKs each packet in a strict transaction-based method. If a CMD_NAK checksum message occurs, the decoder retransmits the packet that was NAK'd.

Multipacketing, Option 2

The decoder sends data packets continuously, with no ACK/NAK handshaking to pace the transmission. If the host is overrun, it can use hardware handshaking to temporarily hold off the decoder.

At the end of transmission, the decoder waits for a CMD_ACK or CMD_NAK. The host acknowledges the transmission, or requests the entire multi-packet message be resent from the first packet.

The host may stop the transmission from the decoder at any time by asserting hardware handshaking. The host then transmits either CMD_NAK, resend to instruct the decoder to resend the entire message, or CMD_NAK, cancel to cancel the transmission completely. Note that because these NAKs are unexpected, interruption of transmission must occur first.

Multipacketing, Option 3

Option 3 is similar to Option 2, except there is a programmable inter-packet delay. The decoder waits a programmed period after sending each packet. This may be faster than Option 1 because the host receives data on a periodic basis without attempting to send the ACK/NAK, but slower than Option 2 since the inter-packet delay transpires on each packet. However, it helps prevent host receiver overrun.

Packet Format

The general packet format for SSI messages is as follows:

| | | | | | |
|--------|--------|----------------|--------|----------|----------|
| Length | Opcode | Message Source | Status | Data.... | Checksum |
|--------|--------|----------------|--------|----------|----------|

Table 1-3 *Field Descriptions*

| Field Name | Format | Sub-Field | Meaning |
|----------------|--------|---|---|
| Length | 1 Byte | Length | Length of message not including the check sum bytes. Maximum value is 0xFF. |
| Opcode | 1 Byte | See SSI Command Lists on page 2-1 . | Identifies the type of packet data sent. |
| Message Source | 1 Byte | 0 = Decoder, 04 = Host | Identifies where the message is coming from. |

Note: The checksum is a 2 byte checksum and must be sent as HIGH BYTE followed by LOW BYTE.

Table 1-3 *Field Descriptions (Continued)*

| Field Name | Format | Sub-Field | Meaning |
|------------|--------------------------|---|---|
| Status | Bit 0 | Retransmit | 0 = First time packet is sent 1 = Subsequent transmission attempts |
| | Bit 1 | Continuation Bit | 0 = Last frame of a multipacket message 1 = Intermediate packet of a multipacket message |
| | Bit 2 | Reserved | Always set to zero |
| | Bit 3 | Change Type (applies to parameters) | 0 = Temporary change 1 = Permanent change |
| | Bits 4 - 7 | | Unused bits must be set to 0. |
| Data... | Variable number of bytes | See individual sections for details. | |
| Checksum | 2 Bytes | 2s complement sum of message contents excluding checksum. | Checksum of message formatted as HIGH BYTE LOW BYTE |

Note: The checksum is a 2 byte checksum and must be sent as HIGH BYTE followed by LOW BYTE.

Chapter 2 SSI Commands

Introduction

This chapter describes each available SSI command, including field descriptions and host and decoder requirements.

SSI Command Lists

The following table lists the available SSI commands alphabetically.

Table 2-1 SSI Commands

| Name | Type | Opcode | Description | Page |
|-----------------------|------|--------|---|----------------------|
| ABORT_MACRO_PDF | H | 0x11 | Terminates MacroPDF sequence and discards segments. | 2-6 |
| AIM_OFF | H | 0xC4 | Deactivates aim pattern. | 2-7 |
| AIM_ON | H | 0xC5 | Activates aim pattern. | 2-8 |
| BATCH_DATA | D | 0xD6 | Transmits stored decode data. | 2-10 |
| BATCH_REQUEST | H | 0xD5 | Requests stored decode data. | 2-10 |
| BEEP | H | 0xE6 | Sounds the beeper. | 2-10 |
| CAPABILITIES_REQUEST | H | 0xD3 | Requests commands which decoder will perform. | 2-13 |
| CAPABILITIES_REPLY | D | 0xD4 | Lists commands which decoder will perform. | 2-14 |
| CHANGE_ALL_CODE_TYPES | H | 0xC9 | Enables / Disables all code types. | 2-19 |
| CMD_ACK | H/D | 0xD0 | Positive acknowledgment of received packet. | 2-20 |

Note: D = Decoder, H = Host, H/D = Host/Decoder

Table 2-1 SSI Commands (Continued)

| Name | Type | Opcode | Description | Page |
|-------------------------------|------|--------|--|----------------------|
| CMD_ACK_ACTION | H | 0xD8 | This is a positive acknowledgment of a received packet and can be used in place of the CMD_ACK command to allow users to control the beeper, pager motor (i.e., vibration feedback) and LEDs after receiving decoded data or any other SSI command. Note: This command is not supported by all scanners. | 2-22 |
| CMD_NAK | H/D | 0xD1 | Negative acknowledgment of received packet. | 2-24 |
| CUSTOM_DEFAULTS | H | 0x12 | Host command to update Custom Defaults Buffer. | 2-27 |
| DECODE_DATA | D | 0xF3 | Decode data in SSI packet format. | 2-28 |
| EVENT | D | 0xF6 | Event indicated by associated event code. | 2-40 |
| FLUSH_MACRO_PDF | H | 0x10 | Terminates MacroPDF sequence and transmits captured segments. | 2-42 |
| FLUSH_QUEUE | H | 0xD2 | Tells the decoder to eliminate all packets in its transmission queue. | 2-43 |
| ILLUMINATION_OFF | H | 0xC0 | Deactivates Illumination | 2-44 |
| ILLUMINATION_ON | H | 0xC1 | Activates Illumination. | 2-45 |
| IMAGE_DATA | D | 0xB1 | Data comprising the image. | 2-46 |
| IMAGER_MODE | H | 0xF7 | Commands imager into operational modes. | 2-48 |
| LED_OFF | H | 0xE8 | Extinguishes LEDs. | 2-49 |
| LED_ON | H | 0xE7 | Activates LED output. | 2-50 |
| PAGER_MOTOR_ACTIVATION | H | 0xCA | Actuates the vibration feedback. | 2-51 |
| PARAM_DEFAULTS | H | 0xC8 | Sets parameter default values. | 2-52 |
| PARAM_REQUEST | H | 0xC7 | Requests values of certain parameters. | 2-53 |
| PARAM_SEND | H/D | 0xC6 | Sends parameter values. | 2-56 |
| REPLY_REVISION | D | 0xA4 | Replies to REQUEST_REVISION with decoder's software/hardware configuration. | 2-58 |
| REQUEST_REVISION | H | 0xA3 | Requests the decoder's configuration. | 2-59 |
| SCAN_DISABLE | H | 0xEA | Prevents the operator from scanning bar codes. | 2-60 |
| SCAN_ENABLE | H | 0xE9 | Permits bar code scanning. | 2-61 |
| SLEEP | H | 0xEB | Requests to place the decoder into low power. | 2-62 |

Note: D = Decoder, H = Host, H/D = Host/Decoder

Table 2-1 SSI Commands (Continued)

| Name | Type | Opcode | Description | Page |
|-------------------------|------|--------|--|----------------------|
| SSI_MGMT_COMMAND | H/D | 0x80 | RSM command to read/set some scanner attributes. | 2-63 |
| START_SESSION | H | 0xE4 | Tells decoder to attempt to decode a bar code. | 2-64 |
| STOP_SESSION | H | 0xE5 | Tells decoder to abort a decode attempt. | 2-65 |
| VIDEO_DATA | D | 0xB4 | Data comprising the video. | 2-66 |
| WAKEUP | H | N/A | Wakes up decoder after it's been powered down. | 2-68 |

Note: D = Decoder, H = Host, H/D = Host/Decoder

Table 2-2 lists the SSI commands by Opcode.

Table 2-2 SSI Commands by Opcode

| Opcode | Name |
|--------|------------------------|
| 0x10 | FLUSH_MACRO_PDF |
| 0x11 | ABORT_MACRO_PDF |
| 0x12 | CUSTOM_DEFAULTS |
| 0x80 | SSI_MGMT_COMMAND |
| 0xA3 | REQUEST_REVISION |
| 0xA4 | REPLY_REVISION |
| 0xB0 | Reserved |
| 0xB1 | IMAGE_DATA |
| 0xB4 | VIDEO_DATA |
| 0xC0 | ILLUMINATION_OFF |
| 0xC1 | ILLUMINATION_ON |
| 0xC4 | AIM_OFF |
| 0xC5 | AIM_ON |
| 0xC6 | PARAM_SEND |
| 0xC7 | PARAM_REQUEST |
| 0xC8 | PARAM_DEFAULTS |
| 0xC9 | CHANGE_ALL_CODE_TYPES |
| 0xCA | PAGER_MOTOR_ACTIVATION |
| 0xD0 | CMD_ACK |
| 0xD1 | CMD_NAK |
| 0xD2 | FLUSH_QUEUE |
| 0xD3 | CAPABILITIES_REQUEST |
| 0xD4 | CAPABILITIES_REPLY |
| 0xD5 | BATCH_REQUEST |
| 0xD6 | BATCH_DATA |
| 0xD8 | CMD_ACK_ACTION |
| 0xE4 | START_SESSION |
| 0xE5 | STOP_SESSION |
| 0xE6 | BEEP |

Table 2-2 SSI Commands by Opcode (Continued)

| Opcode | Name |
|---------------|--------------|
| 0xE7 | LED_ON |
| 0xE8 | LED_OFF |
| 0xE9 | SCAN_ENABLE |
| 0xEA | SCAN_DISABLE |
| 0xEB | SLEEP |
| 0xF3 | DECODE_DATA |
| 0xF6 | EVENT |
| 0xF7 | IMAGER_MODE |
| N/A | WAKEUP |

ABORT_MACRO_PDF

Description

Terminates MacroPDF sequence and discards all captured segments.

Table 2-3 Packet Format - ABORT_MACRO_PDF

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | 11h | 04h | | |

Table 2-4 Field Descriptions - ABORT_MACRO_PDF

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | 11h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

None.

Decoder Requirements

The decoder terminates the current MacroPDF sequence and discards all captured MacroPDF segments.

AIM_OFF

Description

Turns off aiming pattern.

Table 2-5 Packet Format - AIM_OFF

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | C4h | 04h | | |

Table 2-6 Field Descriptions - AIM_OFF

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C4h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

This command applies only to decoders that support an aim pattern.

Decoder Requirements

The decoder turns off the aim pattern, and responds with a CMD_ACK (if ACK/NAK handshaking is enabled).

If the aim pattern is not supported, the decoder responds with NAK_DENIED (if ACK/NAK handshaking is enabled).

AIM_ON

Description

Turns on aiming pattern.

Table 2-7 Packet Format - AIM_ON

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | C5h | 04h | | |

Table 2-8 Field Descriptions - AIM_ON

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C5h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

This command applies only to decoders which support an aim pattern.

Decoder Requirements

The decoder turns on the aim pattern, and responds with a CMD_ACK (if ACK/NAK handshaking is enabled).

If the aim pattern is not supported, the decoder responds with NAK_DENIED (if ACK/NAK handshaking is enabled).

The Aim Duration parameter controls the amount of time the aiming pattern stays on during a trigger pull. The valid values for this parameter are 0 - 99, which equal 0.1 to 9.9 seconds in 100 msec increments. [Table 2-9](#) lists Aim mode behavior in various situations.

Table 2-9 *Aim Mode*

| Command Sequence | Action Performed | Aim Duration Parameters |
|---|--|--------------------------------|
| AIM_ON | Turns on the aiming pattern indefinitely. | aim duration = 0 |
| AIM_OFF | Turns off the aiming pattern. | aim duration = 0 |
| AIM_ON, START_DECODE | Turns on the aiming pattern, when START_DECODE received turns on scan pattern and begins decoding. | aim duration = 0 |
| AIM_ON,AIM_OFF, START_DECODE | Turns on aiming pattern, turns off aiming pattern, turns on scan pattern and begins decoding. | aim duration = 0 |
| START_DECODE | Turns on aiming pattern for aim duration time, turns on scan pattern and begins decoding. | aim duration > 0 |

BEEP

Description

Sounds the beeper.

Table 2-10 Packet Format - BEEP

| Length | Opcode | Message Source | Status | Beep Code | Checksum |
|--------|--------|----------------|--------|-----------|----------|
| 05h | E6h | 04h | | | |

Table 2-11 Field Descriptions - BEEP

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | E6h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Beep Code | See Table 2-12 . | 1 Byte | Number that identifies a beep sequence. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This Opcode instructs the receiver to sound the beep sequence indicated by the *Beep Code* field.

For [Table 2-12](#), *Duration* is the length of a sound, *Pitch* is the pitch of the sound, and *Number of Beeps* indicates the number of times a beep pitch is repeated at the specified duration.

Table 2-12 *Beep Code Definitions*

| Beep Code | Duration | Pitch | Number of Beeps |
|-----------|-------------|-------------------|-----------------|
| 00h | Short | High | 1 |
| 01h | Short | High | 2 |
| 02h | Short | High | 3 |
| 03h | Short | High | 4 |
| 04h | Short | High | 5 |
| 05h | Short | Low | 1 |
| 06h | Short | Low | 2 |
| 07h | Short | Low | 3 |
| 08h | Short | Low | 4 |
| 09h | Short | Low | 5 |
| 0Ah | Long | High | 1 |
| 0Bh | Long | High | 2 |
| 0Ch | Long | High | 3 |
| 0Dh | Long | High | 4 |
| 0Eh | Long | High | 5 |
| 0Fh | Long | Low | 1 |
| 10h | Long | Low | 2 |
| 11h | Long | Low | 3 |
| 12h | Long | Low | 4 |
| 13h | Long | Low | 5 |
| 14h | Fast Warble | High-Low-High-Low | 4 |
| 15h | Slow Warble | High-Low-High-Low | 4 |
| 16h | Mix 1 | High-Low | 2 |
| 17h | Mix 2 | Low-High | 2 |
| 18h | Mix 3 | High-Low-High | 3 |
| 19h | Mix 4 | Low-High-Low | 3 |
| 1Ah | Long | High-High-Low-Low | 4 |

Table 2-12 *Beep Code Definitions (Continued)*

| Beep Code | Duration | Pitch | Number of Beeps |
|-----------|------------|----------------|-----------------|
| 1Bh | Short | High-High-High | 13 |
| 1Ch | High Click | High | 1 |
| 1Dh | Low Click | Low Click | 1 |

Host Requirements

The host sends this command to cause the decoder to beep. The host may also send these beep codes as part of the PARAM_SEND directive.

Decoder Requirements

When the decoder receives this command, it beeps the sequence provided in the BEEP directive. If ACK/NAK handshaking is enabled, the decoder ACKs if a valid beep code is requested. Otherwise it sends CMD_NAK, host directive denied.

CAPABILITIES_REQUEST

Description

Requests the decoder's serial capabilities.

Table 2-13 Packet Format - CAPABILITIES_REQUEST

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | D3h | | | |

Table 2-14 Field Descriptions - CAPABILITIES_REQUEST

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D3h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

The host transmits this message to request the serial capabilities of the decoder system.

Decoder Requirements

Upon receipt of this command, the decoder responds with the CAPABILITIES_REPLY message.

CAPABILITIES_REPLY

Description

Decoder details the serial capabilities.

Table 2-15 Packet Format - CAPABILITIES_REPLY

| Length | Opcode | Message Source | Status | Data | Checksum |
|--------|--------|----------------|--------|------|----------|
| 04h | D4h | | | | |

Table 2-16 Field Descriptions - CAPABILITIES_REPLY

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D4h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Data | | | Table 2-17 on page 2-15. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

The host must not CMD_ACK or CMD_NAK this message, as this is a natural response to the CAPABILITIES_REQUEST message.

Decoder Requirements

The decoder sends this message upon receipt of the CAPABILITIES_REQUEST message.

Table 2-17 Data Fields

| Field | Size | Description | | Supported |
|------------------------|-----------------------|-------------|---------------------|------------------------------------|
| Baud Rates Supported | 2 Bytes Bit mapped | | | 1 = Supported 0 = Not Supported |
| | | 0 | 300 Baud | |
| | | 1 | 600 Baud | |
| | | 2 | 1200 Baud | |
| | | 3 | 2400 Baud | |
| | | 4 | 4800 Baud | |
| | | 5 | 9600 Baud | |
| | | 6 | 19200 Baud | |
| | | 7 | 28800 Baud | |
| | | 8 | 38400 Baud | |
| | | 9 | 57600 Baud | |
| | | 10 | 115200 Baud | |
| | | 11 | 230400 Baud | |
| | | 12 | 460800 Baud | |
| | | 13 | 921600 Baud | |
| 14 | Reserved | | | |
| 15 | Reserved | | | |
| Misc Serial Parameters | 1 Byte Bit Mapped | | | 1 = Supported 0 = Not Supported |
| | | 0 | Odd Parity | |
| | | 1 | Even Parity | |
| | | 2 | Parity None | |
| | | 3 | Check Parity | |
| | | 4 | Do Not Check Parity | |
| | | 5 | One Stop Bit | |
| 6 | Two Stop Bits | | | |

Table 2-17 *Data Fields (Continued)*

| Field | Size | Description | Supported | |
|-----------------------------|----------------------|--|---------------|-------------------|
| Multi Packet Options | 1 Byte Bit Mapped | | 1 = Supported | |
| | | 0 | Option 1 | 0 = Not Supported |
| | | 1 | Option 2 | |
| | | 2 | Option 3 | |
| Command List | 1 Byte per Command | In this sequential list, the decoder details the commands it supports. For example, imagers support video commands, while laser-based decoders do not. Commands associated with video mode will not appear in the list for laser-based decoders, but will for imagers. | | |

BATCH_DATA

Description

Transmits stored decode data as a reply to the BATCH_REQUEST command. Scanners that can not store scans send a NAK DENIED or NAK BAD CONTEXT response.

Table 2-18 Packet Format - BATCH_DATA

| Length | Opcode | Message Source | Status | Bar Code String(s) | Checksum |
|--------|--------|----------------|--------|--------------------|----------|
| | D6h | 00h | | | |

Table 2-19 Field Descriptions - BATCH_DATA

| Field Name | Format | Size | Description |
|---------------------------|--|----------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D6h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Bar Code String(s) | | Variable | Data from a bar code scan in the bar code string format. Multiple instances of this field may be repeated in one message. For multipacket messages, a partial string may be sent, continued in the next packet. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Bar Code String

Each string is stored in this message in three components: Size, Type, and Scan Data. To specify a bar code string these components are combined in the order specified.

- Size: One byte value that contains the length of the Scan Data component
- Type: One byte value that indicates the bar code type of the data scanned:
 - A = UPC/EAN
 - B = Code 39
 - D = EAN 128
 - F = Interleaved 2 of 5
 - G = Discrete 2 of 5
 - K = Code 128
 - N = Coupon code
 - W = Web Code
- Scan Data: One or more bytes of the scanner bar code data in ASCII.

BATCH_REQUEST

Description

Requests stored decode data from the scanner. The scanner responds with the BATCH_DATA command. Scanners that can not store scans respond with a NAK DENIED or NAK BAD CONTEXT.

Table 2-20 Packet Format - BATCH_REQUEST

| Length | Opcode | Message Source | Status | Bar Code String(s) | Checksum |
|--------|--------|----------------|--------|--------------------|----------|
| | D5h | 04h | | | |

Table 2-21 Field Descriptions - BATCH REQUEST

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D5h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

CHANGE_ALL_CODE_TYPES

Description

This command enables and disables all code types.

Table 2-22 Packet Format - BATCH_REQUEST

| Length | Opcode | Message Source | Status | Change Value | Bar Code String(s) | Checksum |
|--------|--------|----------------|--------|--------------|--------------------|----------|
| 05h | C9h | 04h | | | | |

Table 2-23 Field Descriptions - BATCH_REQUEST

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C9h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Change Value | | 1 Byte | 0 = Disable All Code Types 1 = Enable All Code Types |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

CMD_ACK

Description

Positive acknowledgment of received packet.

Table 2-24 Packet Format - CMD_ACK

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | D0h | | | |

Table 2-25 Field Descriptions - CMD_ACK

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D0h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This message is sent to the SSI packet transmitter when the received packet passes the checksum check and no negative acknowledgment conditions apply (see [CMD_NAK on page 2-24](#)). If the data is in response to a command (e.g., PARAM_REQUEST, REQUEST_REVISION, etc.), no ACK is sent.

✓ **NOTE** ACK/NAK handshaking can be disabled, although we recommend it remain enabled.

✓ **NOTE** DO NOT respond to a valid ACK or NAK message.

Host Requirements

A CMD_ACK or response data must be sent by the decoder within the programmable Serial Response Time-out to acknowledge receipt of all messages, unless noted otherwise in the message description section. If the host sends data and does not receive a response within the programmable serial response time-out, it should resend the message (with the retransmit status bit set) before declaring a failure. The host should limit the number of retries.

Decoder Requirements

A CMD_ACK or response data must be sent by the decoder within the programmable Serial Response Time-out to acknowledge receipt of all messages, unless noted otherwise in the message description section. If the decoder does not receive an ACK within this time period, it sends the previous message again (retry). The decoder retries two more times (with the retransmit status bit set) before declaring a transmit error.

CMD_ACK_ACTION

Description

This is the positive acknowledgment of a received packet. This command can be used in place of the standard SSI CMD_ACK to control the beeper, pager motor and LEDs.

✓ **NOTE** This command is not supported by all scanners.

Table 2-26 Packet Format - CMD_ACK_ACTION

| Length | Opcode | Message Source | Status | Beep Command | Pager Motor | LED On | LED Duration | Checksum |
|--------|--------|----------------|--------|--------------|-------------|--------|--------------|----------|
| 08h | D8h | 04h | | | | | | |

Table 2-27 Field Descriptions - CMD_ACK_ACTION

| Field Name | Format | Size | Description |
|---|---|--------|--|
| Length | Length of message not including the checksum. | 1 Byte | Length of field. |
| Opcode | D8h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Beep Command | 0xFF = Do nothing | 1 Byte | Beep code. See Beep Code Definitions on page 2-11 . |
| Pager Motor (PAGER_MOTOR_ACTIVATION) | 0 = Do nothing | 1 Byte | <p>Pager Motor. See PAGER_MOTOR_ACTIVATION on page 2-51.</p> <p>Integer number from 0 to FEh (i.e., 0 to 254 decimal) of 10 ms increments to vibrate the pager motor. For example, 01h = motor vibrates for 10 ms, 02h motor vibrates for 20 ms, etc.</p> |

Table 2-27 Field Descriptions - *CMD_ACK_ACTION* (Continued)

| Field Name | Format | Size | Description |
|-------------------------|--|---------|--|
| LED_ON Selection | 0 = Do nothing | 1 Byte | <p>Bits 0 - 7 correspond to different LEDs on the product. Set each bit to '1' to turn on the corresponding LED; set <i>LED_ON Duration</i> to the amount of time LEDs should remain on.</p> <p>Example of LEDs on the product:</p> <ul style="list-style-type: none"> • Bit 0 = Decode LED. • Bit 1 = Red LED. <p>See LED_ON on page 2-50 for more information.</p> <p>Also refer to the Product Reference Guide (PRG) for further information about the LEDs supported via SSI on the device.</p> |
| LED_ON Duration | | 1 Byte | <p>This byte field is an integer number (0 - 254 decimal, 00h to FEh) used in conjunction with the <i>LED_ON Selection</i> byte to control the LED On duration. The duration is controlled in increments of 10 ms (i.e., 1 - 10 ms, 2 - 20 ms etc.).</p> <p>Note: If this field is 0, and any of the LED bits are set to '1' in the <i>LED_ON Selection</i> field, then the LED's remain on until an <i>LED_OFF</i> command is sent.</p> |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This message is sent to the SSI packet transmitter when the received packet passes the checksum check, and no negative acknowledgment conditions apply (see [CMD_NAK on page 2-24](#)). If the data is in response to a command (e.g., *PARAM_REQUEST*, *REQUEST_REVISION*, etc.), no ACK is sent.



NOTES 1. ACK/NAK handshaking can be disabled, although it is recommended it remain enabled.

2. DO NOT respond to a valid ACK or NAK message.

CMD_NAK**Description**

Negative acknowledgment of received packet.

Table 2-28 *Packet Format - CMD_NAK*

| Length | Opcode | Message Source | Status | Cause | Checksum |
|--------|--------|----------------|--------|-------|----------|
| 05 | D1h | | | | |

Table 2-29 *Field Descriptions - CMD_NAK*

| Field Name | Format | Size | Description |
|-----------------------|---|--------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D1h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder 4 = Host | 1 Byte | Identifies where the message is coming from. |

Table 2-29 Field Descriptions - CMD_NAK (Continued)

| Field Name | Format | Size | Description |
|-----------------|--|---------|---|
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Cause | Reason code | 1 Byte | Identifies the reason the NAK occurred: 0 = Reserved 1 = (RESEND) Checksum failure 2 = (BAD_CONTEXT) Unexpected or Unknown message 3 = Reserved 4 = Reserved 5 = Reserved 6 = (DENIED) Host Directive Denied 7 = Reserved 8 = Reserved 9 = Reserved 10 = (CANCEL) Undesired Message |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This message is sent when the received packet fails the checksum verification or some error occurred while handling the message.

✓ **NOTE** ACK/NAK handshaking can be disabled, although we recommend it remain enabled.

✓ **NOTE** DO NOT respond to a valid ACK or NAK message.

NAK types supported by the decoder are listed in [Table 2-30](#).

Table 2-30 *Decoder-Supported NAK Types*

| NAK Type | Meaning | Receiver Action |
|--------------------|--|--|
| BAD_CONTEXT | Host does not recognize the command. | |
| CANCEL | Host does not want the message in progress. | Decoder discards the current message. |
| DENIED | Host is unable to comply with the requested message (e.g., beep code is out of range). | Do not send data with this message again. Developer should check values with specified values. Developer should ensure the proper character is sent, if using wake-up character. |
| RESEND | Checksum incorrect. | Ensure checksum is correct. Limit number of resends. Send packet again with resend bit set. |

The decoder only resends a message twice. If the message has not been sent successfully at that time, the decoder declares a transmit error, and issues transmit error beeps (LOW-LOW-LOW-LOW).

CMD_NAK, cancel is a special message used when the decoder is sending a message the host does not want, for example a very large image message. The message is discarded by the decoder upon receipt of the CMD_NAK, cancel. This only affects the first queued message. Subsequent messages are not touched. If the host wants the decoder to discard all messages, the host must send a FLUSH_QUEUE message.

CUSTOM_DEFAULTS

Description

Writes or restores parameters to custom defaults.

Table 2-31 Packet Format - CUSTOM_DEFAULTS

| Length | Opcode | Message Source | Status | Action | Decode Data | Checksum |
|--------|--------|----------------|--------|--------|-------------|----------|
| 05h | 12h | 04h | | | | |

Table 2-32 Field Descriptions - CUSTOM_DEFAULTS

| Field Name | Format | Size | Description |
|-----------------------|--|----------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | 12h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Action | | 1 Byte | 0 = Write to Custom Defaults 1 = Restore Custom Defaults |
| Decode Data | <data> | Variable | Data is decoded data including prefix and suffix sent in ASCII format. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This command writes or restores parameters to their custom default settings.

Host Requirements

The host sends this command to program or restore the product's custom default values.

Decoder Requirements

If supported by the scanner, upon receiving this command, the scanner will write the current parameter settings to the custom defaults buffer. If the restore action is requested, then the parameters are restored to their previously stored custom defaults. CMD_ACK / CMD_NAK is transmitted if handshaking is enabled.

DECODE_DATA

Description

Decode data in SSI packet format.

Table 2-33 Packet Format - DECODE_DATA

| Length | Opcode | Message Source | Status | Bar code Type | Decode Data | Checksum |
|--------|--------|----------------|--------|---------------|-------------|----------|
| | F3h | 00h | | | | |

Table 2-34 Field Descriptions - DECODE_DATA

| Field Name | Format | Size | Description |
|-----------------------|--|----------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | F3h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Bar Code Type | See Table 2-37 | 1 Byte | Identifies the scanned data code type. 0 = Not Applicable |
| Decode Data | <data> | Variable | Data is decoded data including prefix and suffix sent in ASCII format. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This Opcode is used by the decoder when packeted data is selected to send decoded bar code data to the host. The decoded message is contained in the *Decode Data* field.

If the decoded data contains more structure than can be presented in the standard format, the Bar Code Type field is set to 0x99 to indicate the Decode Data message contains multiple packets. The format of the Decode Data field

contains the actual Bar Code Type and a packeted form of decode data. For example, a packeted Decode Data message for Micro PDF417 would look like:

Table 2-35 *Packeted Decode Data Message for Micro PDF417*

| Length | Opcode | Message Source | Status | Bar code Type | Decode Data | Checksum |
|--------|--------|----------------|--------|---------------|-------------|----------|
| 12 | F3h | 00h | 0 | 99 | see below | |

where the Decode Data field is broken out as follows:

Table 2-36 *Decode Data*

| Actual Bar Code Type | # of Packets | Spare Byte | Byte Length of Packet #1 | Data | Spare Byte | Byte Length of Packet #2 | Data |
|----------------------|--------------|------------|--------------------------|------|------------|--------------------------|------|
| 1A | 2 | 0 | 00 03 | ABC | 0 | 00 04 | DEFG |

Note that the *Packet Length* subfields consist of two bytes, where the first byte represents the high value of length x 256.

Structured Append

Structured append data for PDF417 and Micro PDF417 can be transmitted in either an unstructured format which adheres to the PDF417 specification, or a structured "smart format" using the multipacketed format above. The *Bar Code Type* field contains 0x99 and data is sent from a single structured append symbol in two Decode Data packets. The first packet contains the main bar code data, and the second contains any bar code identification enabled for transmission (e.g., the control block, optional fields, symbol terminator). Each field begins with its identifying marker codeword (e.g., \928 for control blocks, \923 for optional fields, and \922 for the symbol terminator).

Table 2-37 and Table 2-38 lists all supported code types, by code name and hex value (SSI ID). The associated hex value for each code (as required) is entered in the *Code Type* field.

✓ **NOTE** For multipacketed data, this code type appears in every packet.

Table 2-37 *Code Types and Identifiers*

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|------------------------|--------|---------|---------------|-----------------|
| Aztec Code | 0x2D | z | z | 0 |
| Aztec Rune Code | 0x2E | z | z | C |
| Bookland | 0x16 | L | X | 0 |
| C 2 of 5 | 0x72 | U | X | 0 |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-37 Code Types and Identifiers (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|--|--------|---------|---------------|---|
| Codabar | 0x02 | C | F | 0 (1) - standard (ABC) |
| Code 11 | 0x0C | H | H | 0 (1) [2] - 1 (2) [0] check digits included |
| Code 128 | 0x03 | D | C | 0 (also see GS1-128) |
| Code 16K | 0x12 | X | X | 0 |
| Code 32 | 0x20 | B | A | Same rules as for Code 39 |
| Code 39 | 0x01 | B | A | 0 - no check digit 1 (3) - check digit included (excluded) |
| Code 39 Full ASCII | 0x13 | B | A | 4 - no check digit 5 (7) - check digit included (excluded) |
| Code 49 | 0x0D | X | X | 0 |
| Code 93 | 0x07 | E | G | 0 |
| Composite (CC-A + GS1-128) | 0x51 | T | | See Table 2-39 |
| Composite (CC-A + EAN-13) | 0x52 | T | | See Table 2-39 |
| Composite (CC-A + EAN-8) | 0x53 | T | | See Table 2-39 |
| Composite (CC-A + GS1 DataBar Expanded) | 0x54 | T | | See Table 2-39 |
| Composite (CC-A + GS1 DataBar Limited) | 0x55 | T | | See Table 2-39 |
| Composite (CC-A + GS1 DataBar-14) | 0x56 | T | | See Table 2-39 |
| Composite (CC-A + UPC-A) | 0x57 | T | | See Table 2-39 |
| Composite (CC-A + UPC-E) | 0x58 | T | | See Table 2-39 |
| Composite (CC-B + GS1-128) | 0x61 | T | | See Table 2-39 |
| Composite (CC-B + EAN-13) | 0x62 | T | | See Table 2-39 |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-37 Code Types and Identifiers (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|---|--------|---------|--------------------|--|
| Composite (CC-B + EAN-8) | 0x63 | T | See Table 2-39 | |
| Composite (CC-B + GS1 DataBar Expanded) | 0x64 | T | See Table 2-39 | |
| Composite (CC-B + GS1 DataBar Limited) | 0x65 | T | See Table 2-39 | |
| Composite (CC-B + GS1 DataBar-14) | 0x66 | T | See Table 2-39 | |
| Composite (CC-B + UPC-A) | 0x67 | T | See Table 2-39 | |
| Composite (CC-B + UPC-E) | 0x68 | T | See Table 2-39 | |
| Composite (CC-C + GS1-128) | 0x59 | T | See Table 2-39 | |
| Coupon Code | 0x17 | N | E+C ¹ | 0+1 |
| Cue CAT Code | 0x38 | Q | X | 0 |
| D25 | 0x04 | G | S | 0 |
| Data Matrix | 0x1B | P00 | d | 4 (1) - ECC 200 with (w/o) ECI |
| GS1-128 | 0x0F | K | C | 1 (2) - character 1 (2) is Function 1 (F1) |
| GS1 QR | 0xC2 | P0Q | Q | 3 |
| EAN-13 | 0x0B | A | E | 0 |
| EAN-13 + 2 | 0x4B | A | E + E ² | 0 for main block; 1 for supplemental |
| EAN-13 + 5 | 0x8B | A | E + E ² | 0 for main block; 2 for supplemental |
| EAN-8 | 0x0A | A | E | 4 |
| EAN-8 + 2 | 0x4A | A | E + E ² | 4 for main block; 1 for supplemental |
| EAN-8 + 5 | 0x8A | A | E + E ² | 4 for main block; 2 for supplemental |
| French Lottery | 0x2F | X | X | 0 |
| GS1 DataBar Expanded | 0x32 | R | | |
| GS1 DataBar Limited | 0x31 | R | | |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-37 Code Types and Identifiers (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|---------------------------|--------|---------|---------------|--|
| GS1 DataBar-14 | 0x30 | R | | |
| GS1 Datamatrix | 0xC1 | P0G | d | 2 |
| Han Xin | 0xB7 | P0H | X | 0 |
| IATA | 0x05 | G | S | 0 |
| ISBT-128 | 0x19 | D | C | 0 |
| ISBT-128 Concat. | 0x21 | D | C | 4 |
| ISSN | 0x36 | X | X | 0 |
| ITF | 0x06 | F | I | Same rules as for Code 39 |
| Korean 2 of 5 | 0x73 | V | X | 0 |
| Macro Micro PDF | 0x9A | X | L | Same rules as for Micro PDF-417 |
| Macro PDF-417 | 0x28 | X | L | Same rules as for PDF-417 |
| Macro QR Code | 0x29 | X | X | 0 |
| Matrix 2 of 5 | 0x39 | S | X | 0 |
| Maxicode | 0x25 | P02 | U | 1 - Mode 0, 2 or 3, without ECI 3 (1) - Extended EC with (w/o) ECI |
| Micro PDF | 0x1A | X | L | 3 - Code 128 emul: implied F1 in 1st position 4 - Code 128 emul: F1 after 1st letter/digits 5 - Code 128 emul: no implied F1 |
| Micro PDF CCA | 0x1d | X | X | 0 |
| Micro QR Code | 0x2C | P01 | Q | 1 |
| MSI | 0x0E | J | M | 0 - Modulo 10 symbol check character validated and transmitted 1 - Modulo 10 symbol check character validated but not transmitted |
| Multipacket Format | 0x99 | N/A | N/A | Data is packeted; SSI ID is embedded in decode data. |
| NW7 | 0x18 | X | X | 0 |
| OCRB | 0xA0 | X | X | 0 |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-37 Code Types and Identifiers (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|-----------------------------------|--------|---------|--------------------------------|---|
| Parameter (FNC3) | 0x33 | N/A | N/A | |
| PDF-417 | 0x11 | X | L | 0 - Conforms with 1994 PDF-417 spec 1 - Backslash characters doubled 2 - Backslash characters not doubled |
| Planet (US) | 0x1F | P04 | X | |
| Postal (Australia) | 0x23 | P09 | X | 0 |
| Postal (Dutch) | 0x24 | P08 | X | 0 |
| Postal (Japan) | 0x22 | P05 | X | 0 |
| Postal (UK) | 0x27 | P06 | X | 0 |
| Postbar (CA) | 0x26 | P07 | X | 0 |
| Postnet (US) | 0x1E | P03 | X | 0 |
| QR Code | 0x1C | P01 | Q | 0 |
| RFID Raw | 0xE0 | X | X | 0 |
| RFID URI | 0xE1 | X | X | 0 |
| RSS (GS1 Databar) Expanded Coupon | 0xB4 | R | X | 0 |
| Scanlet Webcode | 0x37 | W | X | 0 |
| Signature | 0x69 | P0X | X | 0 |
| TLC-39 | 0x5A | T | See Table 2-39 | |
| Trioptic | 0x15 | M | X | 0 |
| UPCA | 0x08 | A | E | 0 |
| UPCA + 2 | 0x48 | A | $E + E^2$ | 0 for main block; 1 for supplemental |
| UPCA + 5 | 0x88 | A | $E + E^2$ | 0 for main block; 2 for supplemental |
| UPCD | 0x14 | X | X | 0 |
| UPCE ³ | 0x09 | A | E | 0 |
| UPCE + 2 | 0x49 | A | $E + E^2$ | 0 for main block; 1 for supplemental |
| UPCE + 5 | 0x89 | A | $E + E^2$ | 0 for main block; 2 for supplemental |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-37 Code Types and Identifiers (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|------------|--------|---------|--------------------|--------------------------------------|
| UPCE1 | 0x10 | A | E | 0 |
| UPCE1 + 2 | 0x50 | A | E + E ² | 0 for main block; 1 for supplemental |
| UPCE1 + 5 | 0x90 | A | E + E ² | 0 for main block; 2 for supplemental |
| 4State US | 0x34 | P0A | X | 0 |
| 4State US4 | 0x35 | P0B | X | 0 |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-38 Code Types by SSI ID

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|-------------------|--------|---------|---------------|---|
| Code 39 | 0x01 | B | A | 0 - no check digit 1 (3) - check digit included (excluded) |
| Codabar | 0x02 | C | F | 0 (1) - standard (ABC) |
| Code 128 | 0x03 | D | C | 0 (also see GS1-128) |
| D25 | 0x04 | G | S | 0 |
| IATA | 0x05 | G | S | 0 |
| ITF | 0x06 | F | I | Same rules as for Code 39 |
| Code 93 | 0x07 | E | G | 0 |
| UPCA | 0x08 | A | E | 0 |
| UPCE ³ | 0x09 | A | E | 0 |
| EAN-8 | 0x0A | A | E | 4 |
| EAN-13 | 0x0B | A | E | 0 |
| Code 11 | 0x0C | H | H | 0 (1) [2] - 1 (2) [0] check digits included |
| Code 49 | 0x0D | X | X | 0 |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-38 Code Types by SSI ID (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|---------------------------|--------|---------|------------------|--|
| MSI | 0x0E | J | M | 0 - Modulo 10 symbol check character validated and transmitted 1 - Modulo 10 symbol check character validated but not transmitted |
| GS1-128 | 0x0F | K | C | 1 (2) - character 1 (2) is Function 1 (F1) |
| UPCE1 | 0x10 | A | E | 0 |
| PDF-417 | 0x11 | X | L | 0 - Conforms with 1994 PDF-417 spec 1 - Backslash characters doubled 2 - Backslash characters not doubled |
| Code 16K | 0x12 | | | |
| Code 39 Full ASCII | 0x13 | B | A | 4 - no check digit 5 (7) - check digit included (excluded) |
| UPCD | 0x14 | X | X | 0 |
| Trioptic | 0x15 | M | X | 0 |
| Bookland | 0x16 | L | X | 0 |
| Coupon Code | 0x17 | N | E+C ¹ | 0+1 |
| NW7 | 0x18 | X | X | 0 |
| ISBT-128 | 0x19 | D | C | 0 |
| Micro PDF | 0x1A | X | L | 3 - Code 128 emul: implied F1 in 1st position 4 - Code 128 emul: F1 after 1st letter/digits 5 - Code 128 emul: no implied F1 |
| Data Matrix | 0x1B | P00 | d | 4 (1) - ECC 200 with (w/o) ECI |
| QR Code | 0x1C | P01 | Q | 0 |
| Micro PDF CCA | 0x1d | X | X | 0 |
| Postnet (US) | 0x1E | P03 | X | 0 |
| Planet (US) | 0x1F | P04 | X | |
| Code 32 | 0x20 | B | A | Same rules as for Code 39 |
| ISBT-128 Concat. | 0x21 | D | C | 4 |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-38 Code Types by SSI ID (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|----------------------|--------|---------|---------------|--|
| Postal (Japan) | 0x22 | P05 | X | 0 |
| Postal (Australia) | 0x23 | P09 | X | 0 |
| Postal (Dutch) | 0x24 | P08 | X | 0 |
| Maxicode | 0x25 | P02 | U | 1 - Mode 0, 2 or 3, without ECI 3 (1) - Extended EC with (w/o) ECI |
| Postbar (CA) | 0x26 | P07 | X | 0 |
| Postal (UK) | 0x27 | P06 | X | 0 |
| Macro PDF-417 | 0x28 | X | L | Same rules as for PDF-417 |
| Macro QR Code | 0x29 | X | X | 0 |
| Micro QR Code | 0x2C | P01 | Q | 1 |
| Aztec Code | 0x2D | z | z | 0 |
| Aztec Rune Code | 0x2E | z | z | C |
| French Lottery | 0x2F | X | X | 0 |
| GS1 DataBar-14 | 0x30 | R | | |
| GS1 DataBar Limited | 0x31 | R | | |
| GS1 DataBar Expanded | 0x32 | R | | |
| Parameter (FNC3) | 0x33 | N/A | N/A | |
| 4State US | 0x34 | P0A | X | 0 |
| 4State US4 | 0x35 | P0B | X | 0 |
| Scanlet Webcode | 0x37 | W | X | 0 |
| Cue CAT Code | 0x38 | Q | X | 0 |
| UPCA + 2 | 0x48 | A | $E + E^2$ | 0 for main block; 1 for supplemental |
| UPCE + 2 | 0x49 | A | $E + E^2$ | 0 for main block; 1 for supplemental |
| EAN-8 + 2 | 0x4A | A | $E + E^2$ | 4 for main block; 1 for supplemental |
| EAN-13 + 2 | 0x4B | A | $E + E^2$ | 0 for main block; 1 for supplemental |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-38 Code Types by SSI ID (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|---|--------|---------|--------------------------------|--------------------------------------|
| UPCE1 + 2 | 0x50 | A | E + E ² | 0 for main block; 1 for supplemental |
| Composite (CC-A + GS1-128) | 0x51 | T | See Table 2-39 | |
| Composite (CC-A + EAN-13) | 0x52 | T | See Table 2-39 | |
| Composite (CC-A + EAN-8) | 0x53 | T | See Table 2-39 | |
| Composite (CC-A + GS1 DataBar Expanded) | 0x54 | T | See Table 2-39 | |
| Composite (CC-A + GS1 DataBar Limited) | 0x55 | T | See Table 2-39 | |
| Composite (CC-A + GS1 DataBar-14) | 0x56 | T | See Table 2-39 | |
| Composite (CC-A + UPC-A) | 0x57 | T | See Table 2-39 | |
| Composite (CC-A + UPC-E) | 0x58 | T | See Table 2-39 | |
| Composite (CC-C + GS1-128) | 0x59 | T | See Table 2-39 | |
| TLC-39 | 0x5A | T | See Table 2-39 | |
| Composite (CC-B + GS1-128) | 0x61 | T | See Table 2-39 | |
| Composite (CC-B + EAN-13) | 0x62 | T | See Table 2-39 | |
| Composite (CC-B + EAN-8) | 0x63 | T | See Table 2-39 | |
| Composite (CC-B + GS1 DataBar Expanded) | 0x64 | T | See Table 2-39 | |
| Composite (CC-B + GS1 DataBar Limited) | 0x65 | T | See Table 2-39 | |

Notes:

1. E+C denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
2. E+E denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
3. UPCE, UPCE1, and UPCA are converted to EAN-13 for AIM ID.

Table 2-38 Code Types by SSI ID (Continued)

| Symbology | SSI ID | Code ID | AIM ID Letter | AIM ID Modifier |
|--|--------|---------|--------------------------------|--|
| Composite (CC-B + GS1 DataBar-14) | 0x66 | T | See Table 2-39 | |
| Composite (CC-B + UPC-A) | 0x67 | T | See Table 2-39 | |
| Composite (CC-B + UPC-E) | 0x68 | T | See Table 2-39 | |
| Signature | 0x69 | P0X | X | 0 |
| Matrix 2 of 5 | 0x71 | S | X | 0 |
| C 2 of 5 | 0x72 | U | X | 0 |
| Korean 3 of 5 | 0x73 | V | X | 0 |
| UPCA + 5 | 0x88 | A | E + E ² | 0 for main block; 2 for supplemental |
| UPCE + 5 | 0x89 | A | E + E ² | 0 for main block; 2 for supplemental |
| EAN-8 + 5 | 0x8A | A | E + E ² | 4 for main block; 2 for supplemental |
| EAN-13 + 5 | 0x8B | A | E + E ² | 0 for main block; 2 for supplemental |
| UPCE1 + 5 | 0x90 | A | E + E ² | 0 for main block; 2 for supplemental |
| Multipacket Format | 0x99 | N/A | N/A | Data is packeted; SSI ID is embedded in decode data. |
| Macro Micro PDF | 0x9A | X | L | Same rules as for Micro PDF-417 |
| OCRB | 0xA0 | | | |
| RSS (GS1 Databar) Expanded Coupon | 0xB4 | R | X | 0 |
| Han Xin | 0xB7 | P0H | X | 0 |
| GS1 Datamatrix | 0xC1 | P0G | d | 2 |
| RFID Raw | 0xE0 | X | 0 | 0 |
| RFID URI | 0xE1 | X | 0 | 0 |

Notes:

- E+C** denotes 2 AIM IDs are transmitted: one for the UPC/EAN block; the second prefixes the extended GS1-128 data.
- E+E** denotes 2 AIM IDs are transmitted: the first prefixes the main UPC/EAN block; the second prefixes the supplemental block.
- UPCE, UPCE1, and UPCA** are converted to **EAN-13** for AIM ID.

Table 2-39 Composite Code Data Formats

| 1D Component | Data Format | |
|---|--|--|
| | Standard Mode | GS1-128 Emulation Mode |
| EAN-13, UPC-A, UPC-E | 1D:]E0 2D:]e0 See note 5 below. | 1D:]E0 2D:]C1 before each GS1-128 split transmission See notes 3 -5 below. |
| EAN-8 | 1D:]E4 2D:]e0 See note 5 below. | 1D:]E4 2D:]C1 before each GS1-128 split transmission See notes 3 -5 below. |
| GS1 DataBar-14 GS1 DataBar Limited | 1D:]e0 2D:]e1 See note 2 below. |]C1 before each GS1-128 split transmission See notes 3 -5 below. |
| Code 39 (TLC39) | ANSI MH10.8.3M syntax: 06 Format:]> ^R _S 06 ^G _S 6P 1D ^G _S S 2D ^R _S EOT 05 Format:]> ^R _S 05 ^G _S 906P 1D ^G _S 8004 2D ^R _S EOT See note 6 below. | |
| GS1-128 GS1 DataBar Expanded | If the last AI in the GS1-128 is a predefined, fixed length:]e0 Otherwise,]e0 GS See note 2 below. |]C1 before each GS1-128 split transmission See notes 3 and 4 below. |

Notes:

- All Function 1 characters in the 1D and 2D are sent as ^G_S (29₁₀); the first Function 1 in the GS1-128 is not transmitted.
- In standard mode, the data following symbol separator begins with AIM ID "]e1". The data following the composite component escape mechanism begins with AIM ID "]e2" if ECI interpretation is enabled, "]e3" if ECI interpretation is not enabled.
- In GS1-128 emulation mode, each packet is split on an AI boundary and limited to less than 48 characters.
- In GS1-128 emulation mode, data is discarded after the first symbol separator or escape mechanism.
- If the UPC/EAN component has a supplemental ,]E1 precedes a 2-digit supplemental and]E2 precedes the 5-digit supplemental
- RS is character 30₁₀ and EOT is character 04. The transmitted format (05 or 06) is data dependent.

Host Requirements

If DECODE_EVENT reporting is enabled, the decode event message is received before the DECODE_DATA message. If ACK/NAK handshaking is enabled, the host responds to each of these messages.

Decoder Requirements

Decode data is sent in this format if packeted decode data is selected via parameter. The host responds to this message with a CMD_ACK, if ACK/NAK handshaking is enabled.

EVENT

Description

Indicates selected events occurred.

Table 2-40 Packet Format - EVENT

| Length | Opcode | Message Source | Status | Event Code | Checksum |
|--------|--------|----------------|--------|------------|----------|
| 05h | F6h | 00h | | | |

Table 2-41 Field Descriptions - EVENT

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | F6h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Event Code | Type of Event Code. | 1 Byte | See Table 2-42 on page 2-41 . |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This message is sent by the decoder when an enabled event occurs. Use [Table 2-42](#) and parameters F0h 00h through F0h 07h to determine which events you would like to be reported.

Host Requirements

The host receives this message when a selected event occurs.

Decoder Requirements

Generate this message when a selected event occurs. Events may vary by decoder type.

Table 2-42 *Event Codes*

| Event | Code |
|-----------------------------------|-------------|
| Boot Event | 03h |
| Decode Event | 01h |
| Parameter Defaults | 0Ah |
| Parameter Entry Error | 07h |
| Parameter Num Expected | 0Fh |
| Parameter Stored | 08h |
| Trigger Pull Event | 02h |
| Parameter Entry Cancel | 09h |
| MPDF Incorrect Symbol | 11h |
| MPDF File ID Error | 12h |
| MPDF Out of Memory Error | 13h |
| MPDF Bad Symbology Error | 14h |
| MPDF Flush Buffer | 15h |
| MPDF Data Xmitted | 17h |
| MPDF Flush No Data | 18h |
| MPDF Abort | 19h |
| Buffer Code 39 Add | 1Ah |
| Buffer Code 39 Empty | 1Bh |
| Buffer Code 39 Full | 1Ch |
| Buffer Code 39 Clear | 1Dh |
| Buffer Code 39 Xmit | 1Eh |
| System Fault: Laser Safety | 2h |

FLUSH_MACRO_PDF

Description

Terminates MacroPDF sequence and sends all captured segments.

Table 2-43 Packet Format - FLUSH_MACRO_PDF

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | 10h | 04h | | |

Table 2-44 Field Descriptions - FLUSH_MACRO_PDF

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | 10h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

None.

Decoder Requirements

The decoder terminates the current MacroPDF sequence and transmits the captured MacroPDF segments.

FLUSH_QUEUE

Description

Eliminates content of decoder's transmission queue.

Table 2-45 Packet Format - FLUSH_QUEUE

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | D2h | 04h | | |

Table 2-46 Field Descriptions - FLUSH_QUEUE

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | D2h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This message is sent by the host to instruct the decoder to discard or flush the transmission queue. This is useful when the decoder is attempting to send a lengthy multipacket message. If the host does not want the message, the host can interrupt the decoder (by asserting RTS) and send a FLUSH_QUEUE message.

The decoder ACK/NAKs the FLUSH_QUEUE message. No further packets in the transmission queue are sent. Note that this does not abort decoder actions that cause packets to be added to the transmission queue.

We recommend issuing a SCAN_DISABLE prior to issuing a FLUSH_QUEUE so that new elements are not added to the queue just after it is emptied. Also, paradoxical cases may arise if a SCAN_DISABLE is not issued first.

ILLUMINATION_OFF

Description

Turns off Illumination pattern.

Table 2-47 Packet Format - ILLUMINATION_OFF

| Length | Opcode | Message Source | Status | Data | Checksum |
|--------|--------|----------------|--------|------|----------|
| 04h | C0h | 04h | | | |

Table 2-48 Field Descriptions - ILLUMINATION_OFF

| Field Name | Format | Size | Description |
|-----------------------|--|-----------------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C0h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission |
| Data Content | | Up to 251 Bytes | Image Data records. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Decoder Requirements

The decoder turns off the Illumination, and responds with a CMD_ACK (if ACK/NAK handshaking is enabled).

ILLUMINATION_ON

Description

Turns off Illumination pattern.

Table 2-49 Packet Format - ILLUMINATION_ON

| Length | Opcode | Message Source | Status | Data | Checksum |
|--------|--------|----------------|--------|------|----------|
| 04h | C1h | 04h | | | |

Table 2-50 Field Descriptions - ILLUMINATION_ON

| Field Name | Format | Size | Description |
|-----------------------|--|-----------------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C0h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission |
| Data Content | | Up to 251 Bytes | Image Data records. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Decoder Requirements

The decoder turns on the Illumination, and responds with a CMD_ACK (if ACK/NAK handshaking is enabled).

Table 2-51 Aim Mode

| Command Sequence | Action Performed |
|------------------|-----------------------------|
| ILLUMINATION_ON | Turns on the Illumination. |
| ILLUMINATION_OFF | Turns off the Illumination. |

IMAGE_DATA

Description

A JPEG, BMP, or TIFF image.

Table 2-52 Packet Format - IMAGE_DATA

| Length | Opcode | Message Source | Status | Data | Checksum |
|--------|--------|----------------|--------|------|----------|
| | B1h | 01h | | | |

Table 2-53 Field Descriptions - IMAGE_DATA

| Field Name | Format | Size | Description |
|-----------------------|--|-----------------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | B1h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Data Content | | Up to 251 Bytes | Image Data records. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This packet contains image information. Images sent from the decoder to the host are described by the image preamble contained in the first 10 bytes of the first packet of the image. The details of the image preamble follow. Due to the small packet size of SSI, multiple packets of image data should be received by the host and re-assembled in the order given by the decoder. The packets describe, for example, a JPEG image when re-assembled.

The image preamble consists of the following fields:

Table 2-54 *Image Preamble Fields*

| Field | Field Size | Description |
|-----------------------|--------------|---|
| File size | 4 byte field | Number of bytes in the overall image. |
| Image Width | 2 byte field | Image width in pixels |
| Image Height | 2 byte field | Image height in pixels |
| Image Type | 1 byte field | 0x31 = JPEG Image File 0x33 = BMP Windows Bit Map File 0x34 = TIFF File Note: These values are ASCII. |
| Bits per Pixel | 1 byte field | Number of bits per pixel in image 0 = 1 bit/pixel Black White Image 1 = 4 bit/pixel 16 Gray Scale Image 2 = 8 bit/pixel 256 Gray Scale Image |

Note: The preamble only appears in the first packet of a multipacket message.

In a multipacketed environment, one image frame is spread over several packets in the following format:

Packet 1

| | | | |
|--------|----------|--------------------|----------|
| Header | Preamble | Image Data, Part 1 | Checksum |
|--------|----------|--------------------|----------|

Packet 2

| | | |
|--------|--------------------|----------|
| Header | Image Data, Part 2 | Checksum |
|--------|--------------------|----------|

.

.

.

Packet N

| | | |
|--------|--------------------|----------|
| Header | Last of Image Data | Checksum |
|--------|--------------------|----------|

This is re-assembled by the host into:

| | |
|----------|-------------|
| Preamble | Image Frame |
|----------|-------------|

IMAGER_MODE

Description

Commands Imager into Operational Modes.

- 0 = Decode Mode
- 1 = Image Capture Mode
- 2 = Video Mode.

Table 2-55 Packet Format - IMAGER_MODE

| Length | Opcode | Message Source | Status | Data | Checksum |
|--------|--------|----------------|--------|------|----------|
| 05h | F7h | 00h | | | |

Table 2-56 Field Descriptions - IMAGER_MODE

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | F7h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Data Content | | 1 Byte | Value 0, 1, or 2 decimal |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

This command is supported by the imager only. The host sends this command with the data field set to 0 for decode mode, 1 for image capture mode, and 2 for video mode.

Decoder Requirements

The decoder (imager) sends a CMD_ACK if the mode is valid, and CMD_NAK if not.

LED_OFF

Description

De-activates LED output.

Table 2-57 Packet Format - LED_OFF

| Length | Opcode | Message Source | Status | LED Selection | Checksum |
|--------|--------|----------------|--------|---------------|----------|
| 05h | E8h | 04h | | | |

Table 2-58 Field Descriptions - LED_OFF

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | E8h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| LED Selection | Bit 0 - 7: LED bit numbers to turn off. | 1 Byte | Bit 0 = Decode LED See your product's Product Reference Guide for further bit information. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

The host sends this message to turn off the specified decoder LEDs.

Host Requirements

None.

Decoder Requirements

The decode LED is turned off by the decoder.

LED_ON

Description

Activates LED output.

Table 2-59 Packet Format - LED_ON

| Length | Opcode | Message Source | Status | LED Selection | Checksum |
|--------|--------|----------------|--------|---------------|----------|
| 05h | E7h | 04h | | | |

Table 2-60 Field Descriptions - LED_ON

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | E7h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| LED Selection | Bit 0 - 7: LED bit numbers to turn on. | 1 Byte | Bit 0 = Decode LED See your product's Product Reference Guide for further bit information. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

The host sends this message to turn on the specified decoder LEDs.

Host Requirements

None.

Decoder Requirements

The decode LED is turned on by the decoder.

PAGER_MOTOR_ACTIVATION

Description

Actuates the vibration feedback device in the target device (e.g., the pager motor). Example: A value of 15 causes the scanner to vibrate for 150 ms.

Table 2-61 Packet Format - PAGER_MOTOR_ACTIVATION

| Length | Opcode | Message Source | Status | Vibration Feedback Duration | Checksum |
|--------|--------|----------------|--------|-----------------------------|----------|
| 05h | CAh | 04h | | | |

Table 2-62 Field Descriptions - PAGER_MOTOR_ACTIVATION

| Field Name | Format | Size | Description |
|---------------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | CAh | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission |
| Vibration Duration | | 1 Byte | Number of 10 ms increments to vibrate. 0 = Use the system parameter vibration duration. Example: A value of 15 causes the scanner to vibrate for 150 ms. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This Opcode instructs the receiver to actuate the vibration feedback device (e.g., the pager motor) for the amount of 10 ms increments specified in the Vibration Duration field. If the Vibrations Duration field is set to 0, then the vibration feedback duration will be that which is defined in the system parameter for vibration duration.

Host Requirements

The host sends this command to cause the decoder to actuate its vibration feedback mechanism (e.g., its pager motor) for the specified amount of time.

Decoder Requirements

If the decoder has a Pager Motor and handshaking is enabled, it sends a CMD_ACK and activates the PAGER_MOTOR for the appropriate duration. If the decoder does not have a Pager Motor, it sends a CMD_NAK with type NAK_DENIED.

PARAM_DEFAULTS

Description

Sets the parameters to their default values.

Table 2-63 Packet Format - PARAM_DEFAULTS

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | C8h | 04h | | |

Table 2-64 Field Descriptions - PARAM_DEFAULTS

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C8h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This command returns all parameters to their default settings.

Host Requirements

The host sends this command to reset the decoder's parameter settings to the default values.

Decoder Requirements

Upon receiving this command, the decoder resets all its parameters to the default values. This is equivalent to scanning a SET DEFAULTS bar code.

PARAM_REQUEST

Description

Requests values of selected parameters.

Table 2-65 Packet Format - PARAM_REQUEST

| Length | Opcode | Message Source | Status | Request Data | Checksum |
|--------|--------|----------------|--------|--------------|----------|
| | C7h | 04h | | | |

Table 2-66 Field Descriptions - PARAM_REQUEST

| Field Name | Format | Size | Description |
|-----------------------|--|----------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C7h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Request Data | <Param_num><Param_num> <Param_num>... | Variable | |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

The host uses this message to request selected parameters from the decoder.

Host Requirements

The host requests the decoder's current values for specific parameters by listing the parameter numbers in the Request_Data field. If the host asks for a parameter value not supported by the decoder, the decoder does not send a value for this unsupported param_num. If none of the requested values is supported, an empty

PARAM_SEND message is transmitted. If the host requests the value of all the parameters, it sends a special param_num called ALL_PARAMS (FEh) in the first position of the Request_Data field.

- ✓ **NOTE** The decoder's response to this command is PARAM_SEND, not ACK. Depending on the time-out set, and the number of parameters requested, this reply may fall outside the programmable Serial Response Timeout. It should not be considered an error if the time-out is exceeded. To compensate, increase the time-out.

Decoder Requirements

When the decoder receives this message, it processes the information by formatting a PARAM_SEND message containing all requested parameters that are supported, and their values. The programmable Serial Response Time-out may be exceeded when processing this message, depending on the time-out set, and the number of parameters requested.

Hints for Requesting Parameter Values

Before forming a PARAM_REQUEST, be sure you are requesting parameters supported by the decoder ([Table 2-67](#)). To find out what parameters are supported, send an FEh (request all parameters). The decoder responds with a PARAM_SEND which contains all the supported parameters and their values. This response may be multipacketed; ACK responses are not necessary.

Table 2-67 Parameter Numbers Format

| Parameter Number | Encoding |
|---|---|
| 0 to 239 | <param_num> |
| 256 to 495 | F0<param_num - 256> |
| 512 to 751 | F1<param_num - 512> |
| 768 to 1007 | F2<param_num - 768> |
| 1024 or higher | F8<param_num_high_byte><param_num_low_byte> |
| Additionally, the following special codes are provided: | |
| All Parameters | FE |
| All Defaults | FD |

When using the FEh, it must be in the first position of the Request_Data field, or it is treated as an unsupported parameter.

Unsupported parameters are not listed in the PARAM_SEND response. Requesting unsupported parameters has no effect, but can cause delays in responding to requests for valid parameters. See [Table 2-68](#) for example requests and responses.

Table 2-68 Example Requests and Replies

| | PARAM_REQUEST Message | Response PARAM_SEND Message |
|--------------------|----------------------------|--|
| #ALL | 05 C7 04 00 FE FE 32 | 0D C6 00 00 FF 01 00 02 01 9C 07 E6 63 FC 3E |
| #1, 9C | 06 C7 04 00 01 9C FE 92 | 09 C6 00 00 FF 01 00 9C 07 FD 8E |
| #All, 1, 9C | 07 C7 04 00 FE 01 9C FD 93 | 0D C6 00 00 FF 01 00 02 01 9C 07 E6 63 FC 3E |

Table 2-68 Example Requests and Replies (Continued)

| PARAM_REQUEST Message | | Response PARAM_SEND Message |
|---|----------------------------|--|
| #1, 9C, ALL | 07 C7 04 00 01 9C FE FD 93 | 09 C6 00 00 FF 01 00 9C 07 FD 8E |
| #4 | 05 C7 04 00 04 FF 2C | 05 C6 00 00 FF FE 36 |
| #ALL - 3 times | 07 C7 04 00 FE FE FE FC 34 | 0D C6 00 00 FF 01 00 02 01 9C 07 E6 63 FC 3E |
| #1 -3 times | 07 C7 04 00 01 01 01 FF 2B | 0B C6 00 00 FF 01 00 01 00 01 00 FE 2D |
| 533 (F1 15) Example of buffer parameter above 512. | 06 C7 04 80 F1 15 FD | 1D C6 00 00 FF F7 F1 15 12 00 00 44 53 34 33 30 38 2D 53 52 30 30 30 30 37 5A 5A 57 57 F7 7E Where: F7 = Multipacket array F1h 15h / 533 =SSI number / parameter number Value= "DS4308-SR00007ZZ" |
| 318 (F0, 3E) Example of Word parameter above 256. | 06 C7 04 80 F0 3E FD 81 | 0A C6 00 00 FF F4 F0 3E 04 FF FB 0C Where: F4 = Word parameter F0 3E / 318 =SSI number / parameter number 04 FF = Value 1279 |
| 1118 (F8 04 5E) Example of a Word parameter with a parameter number above 1024. | 07 C7 04 80 F8 04 5E FD 54 | 0B C6 00 00 FF F4 F8 04 5E 00 00 FB E2 Where: F4 = Word Parameter F8 04 5E / 1118=SSI number / parameter number 00 00= Value |

PARAM_SEND

Description

Responds to a PARAM_REQUEST, changes particular parameter values.

Table 2-69 Packet Format - PARAM_SEND

| Length | Opcode | Message Source | Status | Beep Code | Param Data | Checksum |
|--------|--------|----------------|--------|-----------|------------|----------|
| | C6h | | | | | |

Table 2-70 Field Descriptions - PARAM_SEND

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | C6h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Beep code | See Table 2-12 on page 2-11 . | 1 Byte | If no beep is required, set this field to FF. |
| Param_Data | See Table 2-71 on page 2-57 . | | The parameter numbers and data to be sent to the requester. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This message is sent by the decoder in response to the PARAM_REQUEST message, or by the host to change the decoder's parameter values.

Parameter numbers F0h (+256), F1h (+512), F2h (+768) access parameters whose numbers are 256 and higher. For example, to access the first parameter in the 256-511 range, use F0h and 00h.

The PARAM_SEND message encodes parameter plus data as shown in [Table 2-71](#).

Table 2-71 Param Data Format

| Parameter Number | Encoding |
|------------------|---|
| 0 to 239 | <param_num> |
| 256 to 495 | F0<param_num - 256> |
| 512 to 751 | F1<param_num - 512> |
| 768 to 1007 | F2<param_num - 768> |
| 1024 or higher | F8<param_num_high_byte><param_num_low_byte> |

Additionally, there are modifiers to allow data other than byte values.

Table 2-72 Data Types

| Data Type | Format |
|--------------|---|
| String | F3 <param num><len of data><val1><val2>... |
| Word | F4<param num><high byte><low byte> |
| Array | F6<param num><len of array><byte0><byte1>... |
| Multi-Packet | F7<param num><packet len><2 byte offset><byte0><byte1>... |

Host Requirements

- ✓ **NOTE** Due to the processing time of interpreting and storing parameters contained in the message, it may not be possible for the decoder to send an ACK within the programmable Serial Response Timeout. It should not be considered an error if the time-out is exceeded. To compensate, increase the time-out.

The host transmits this message to change the decoder's parameters. Be sure the Change Type bit in the Status byte is set as desired. If no beep is required, the beep code must be set to FFh, or the decoder beeps as defined in [Table 2-12](#).

Decoder Requirements

When the decoder receives a PARAM_SEND, it interprets and stores the parameters, then ACKs the command (if ACK/NAK handshaking is enabled). These parameters are stored permanently only if the Change Type (bit 3 of the Status byte) is set to 1. If bit 3 is set to 0 the changes are temporary, and are lost when the decoder is powered down.

If the PARAM_SEND sent by the host contains a valid beep code, the decoder issues the requested beep sequence, and changes the requested parameter values.

The decoder issues a PARAM_SEND in response to a PARAM_REQUEST from the host. It sends the values for all the supported parameter values requested in the PARAM_REQUEST message. No value is sent for any unsupported param_num. If none of the requested values is supported, the PARAM_SEND message is transmitted with no parameters. When sending this command, the Change Type bit (bit 3 of Status byte) can be ignored.

- ✓ **NOTE** For multipacketed PARAM_SEND, the beep code appears in every packet.

REPLY_REVISION

Description

Replies to REQUEST_REVISION command with software revision string.

Table 2-73 Packet Format - REPLY_REVISION

| Length | Opcode | Message Source | Status | Revision | Checksum |
|--------|--------|----------------|--------|---|----------|
| | A4h | 00h | | S/W_REVISION <space> BOARD_TYPE <space> ENGINE_CODE <space> | |

Table 2-74 Field Descriptions - REPLY_REVISION

| Field Name | Format | Size | Description |
|-----------------------|--|----------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | A4h | 1 Byte | Identifies this Opcode type. |
| Message Source | 0 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Revision | ASCII data | variable | <p>Revision String fields indicate:</p> <p>S/W_REVISION is the release name of the software</p> <p>BOARD_TYPE is N for non-flash decoder board, F for flash</p> <p>ENGINE_CODE indicates the type of scan engine paired with the decoder (see the scan engine's Integration Guide for the engine code value)</p> |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

None.

Decoder Requirements

The decoder sends its revision string to the host. The revision string is decoder-dependent.

REQUEST_REVISION

Description

Requests the software revision string from the decoder.

Table 2-75 Packet Format - REQUEST_REVISION

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | A3h | 04h | | |

Table 2-76 Field Descriptions - REQUEST_REVISION

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | A3h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

The host sends this message to request revision information from the decoder. The decoder responds with `REPLY_REVISION`.

Decoder Requirements

The decoder sends its revision string to the host. See [REPLY_REVISION on page 2-58](#) for format.

SCAN_DISABLE

Description

Prevents the decoder from scanning bar codes.

Table 2-77 Packet Format - SCAN_DISABLE

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | EAh | 04h | | |

Table 2-78 Field Descriptions - SCAN_DISABLE

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | EAh | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

All scan attempts are disabled by this command until either a SCAN_ENABLE is sent, or the decoder is reset.

Decoder Requirements

When the decoder receives this command, it ignores all trigger/START_SESSION requests until a SCAN_ENABLE command is received.

SCAN_ENABLE

Description

Permits the decoder to scan bar codes.

Table 2-79 Packet Format - SCAN_ENABLE

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | E9h | 04h | | |

Table 2-80 Field Descriptions - SCAN_ENABLE

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | E9h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

The host sends the SCAN_ENABLE command to tell the decoder to allow scanning. Scanning is enabled upon power-up, so this command need only be send if a prior SCAN_DISABLE command has been sent.

Decoder Requirements

The decoder allows scanning and decoding upon receipt of this command.

✓ **NOTE** At initial power-up, the decoder should assume SCAN_ENABLED.

SLEEP

Description

Requests to place the decoder into low power mode.

Table 2-81 Packet Format - SLEEP

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | EBh | 04h | | |

Table 2-82 Field Descriptions - SLEEP

| Field Name | Format | Size | Description |
|-----------------------|--|---------|--|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | EBh | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | <p>Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission</p> <p>Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet</p> <p>Bit 2: Reserved Always 0</p> <p>Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change</p> |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

Host Requirements

The host sends this command to place the decoder into low power mode. If the low power mode parameter is enabled, the scanner goes into low power mode automatically, and the SLEEP command is not necessary.

✓ **NOTE** The decoder may not sleep immediately upon acknowledging the command, as it may be busy processing data at the time.

Decoder Requirements

None.

SSI_MGMT_COMMAND

Description

The SSI protocol allows the host to send a command that is variable in length up to 255 bytes. Although there is a provision in the protocol to multi-packet commands from the host, it is not supported in the scanner. It is required that the host fragment packets using the provisions supplied in the Remote Scanner Management (RSM) protocol (ATTRIBUTE_SET_OFFSET, ATTRIBUTE_GET_OFFSET).

RSM command is encapsulated in SSI packet as follows.

Table 2-83 Packet Format - SSI_MGMT_COMMAND

| Length | Opcode | Message Source | Status | Management Payload | Checksum |
|--------|--------|----------------|--------|--------------------|----------|
| | 80h | 04h | | | |

Table 2-84 Field Descriptions - SSI_MGMT_COMMAND

| Field Name | Format | Size | Description |
|---------------------------|--|----------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | 80h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Management Payload | Data | Variable | RSM command. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

The expected response in the positive case is SSI_MGMT_COMMAND that may be a multi-packet response. For devices that do not support the SSI_MGMT_COMMAND, the response will be the standard SSI_NAK (NAK_BADCONTEXT).

Host Requirements

None.

Decoder Requirements

Decoder reply the RSM command in the format below.

Table 2-85 Response Packet Format - SSI_MGMT_COMMAND

| Length | Opcode | Message Source | Status | Management Payload | Checksum |
|--------|--------|----------------|--------|--------------------|----------|
| | 80h | 04h | | | |

START_SESSION

Description

Tells decoder to attempt to obtain the requested data.

Table 2-86 Packet Format - START_SESSION

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | E4h | 04h | | |

Table 2-87 Field Descriptions - START_SESSION

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | E4h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This command tells the decoder to start a scan session. See [Table 2-88](#) for the decoder's reactions to this command in each operational mode.

Table 2-88 START_SESSION Actions

| Operational Mode | Actions Upon Receipt of Command | End Result of Session |
|----------------------|--|--|
| Decode Mode | The decoder attempts to decode a bar code | Successful decode, or STOP_SESSION command |
| Image Capture | The decoder clicks the shutter | An image is captured |
| Video Mode | The decoder continuously produces a video stream | STOP_SESSION command |

Decoder Requirements

Trigger Mode must be set to Host or a CMD_NAK DENIED response is issued.

STOP_SESSION

Description

Tells decoder to abort a decode attempt or video transmission.

Table 2-89 Packet Format - STOP_SESSION

| Length | Opcode | Message Source | Status | Checksum |
|--------|--------|----------------|--------|----------|
| 04h | E5h | 04h | | |

Table 2-90 Field Descriptions - STOP_SESSION

| Field Name | Format | Size | Description |
|-----------------------|--|---------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | E5h | 1 Byte | Identifies this Opcode type. |
| Message Source | 4 = Host | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

This command tells the decoder to stop a scan and decode attempt.

Host Requirements

None.

Decoder Requirements

None.

VIDEO_DATA

Description

Imager transmission of a video frame in JPEG format.

Table 2-91 Packet Format - VIDEO_DATA

| Length | Opcode | Message Source | Status | Data | Checksum |
|--------|--------|----------------|--------|------|----------|
| | B4h | 00h | | | |

Table 2-92 Field Descriptions - VIDEO_DATA

| Field Name | Format | Size | Description |
|-----------------------|--|-----------------|---|
| Length | Length of message (not including checksum). | 1 Byte | Length Field |
| Opcode | B4h | 1 Byte | Identifies this Opcode type. |
| Message Source | 00 = Decoder | 1 Byte | Identifies where the message is coming from. |
| Status | | 1 Byte | Bit 0: Retransmit 0 = First transmission 1 = Subsequent transmission Bit 1: Continuation 0 = Last packet of a multipacket message 1 = Intermediate packet Bit 2: Reserved Always 0 Bit 3: Parameter Change Type (for parameters) 0 = Temporary change 1 = Permanent change |
| Data | | Up to 251 Bytes | Image data. |
| Checksum | 2's complement sum of message contents excluding checksum. | 2 Bytes | Checksum of message. |

The first packet of a video frame contains the video preamble, described below. The first packet also contains the JPEG data comprising the video frame. Multipacketing is expected in video mode.

The video preamble consists of the following fields:

Table 2-93 Video Preamble Fields

| Field | Field Size | Description |
|-----------------------|--------------|---|
| File size | 4 byte field | Number of bytes in the overall image. |
| Image Width | 2 byte field | Image width in pixels |
| Image Height | 2 byte field | Image height in pixels |
| Image Type | 1 byte field | 0x31 = JPEG Image File 0x33 = BMP Windows Bit Map File 0x34 = TIFF File Note: These values are ASCII. |
| Bits per Pixel | 1 byte field | Number of bits per pixel in image 0 = 1 bit/pixel Black White Image 1 = 4 bit/pixel 16 Gray Scale Image 2 = 8 bit/pixel 256 Gray Scale Image |

In a multipacketed environment, one video frame is spread over several packets in the following format:

Packet 1

| | | | |
|--------|----------|--------------------|----------|
| Header | Preamble | Video Data, Part 1 | Checksum |
|--------|----------|--------------------|----------|

Packet 2

| | | |
|--------|--------------------|----------|
| Header | Video Data, Part 2 | Checksum |
|--------|--------------------|----------|

.

.

.

Packet N

| | | |
|--------|--------------------|----------|
| Header | Last of Video Data | Checksum |
|--------|--------------------|----------|

This is re-assembled by the host into:

| | |
|----------|-------------|
| Preamble | Video Frame |
|----------|-------------|

WAKEUP

Description

Wakes up decoder after it was put into low power operation.

If the decoder is in low power mode, sending the single character **NULL** (00) wakes up the decoder. This character is only needed when hardware handshaking is not being used or is bypassed.

Host Requirements

Once the WAKEUP character is sent, the host must wait a period of time to permit the decoder to wake up. The decoder remains awake for a fixed period of time after wake up. These time periods vary by decoder.

Decoder Requirements

The decoder must not go back into low power mode for a decoder-dependent time period after waking up.

- ✓ **NOTE** The mechanism to wake up a decoder in this manner also works if characters other than WAKEUP are sent to the decoder. There is, however, no guarantee that these commands are interpreted correctly upon power-up. Therefore, it is not recommended that characters other than WAKEUP be used to awaken the decoder.

The WAKEUP character has no effect if sent when the scanner is awake. If the host is unsure of the scanner state, it should send the wakeup character when it wants to communicate with the scanner.

Appendix A Transaction Examples

Various transaction examples are shown in [Figure A-1](#) through [Figure A-8](#).

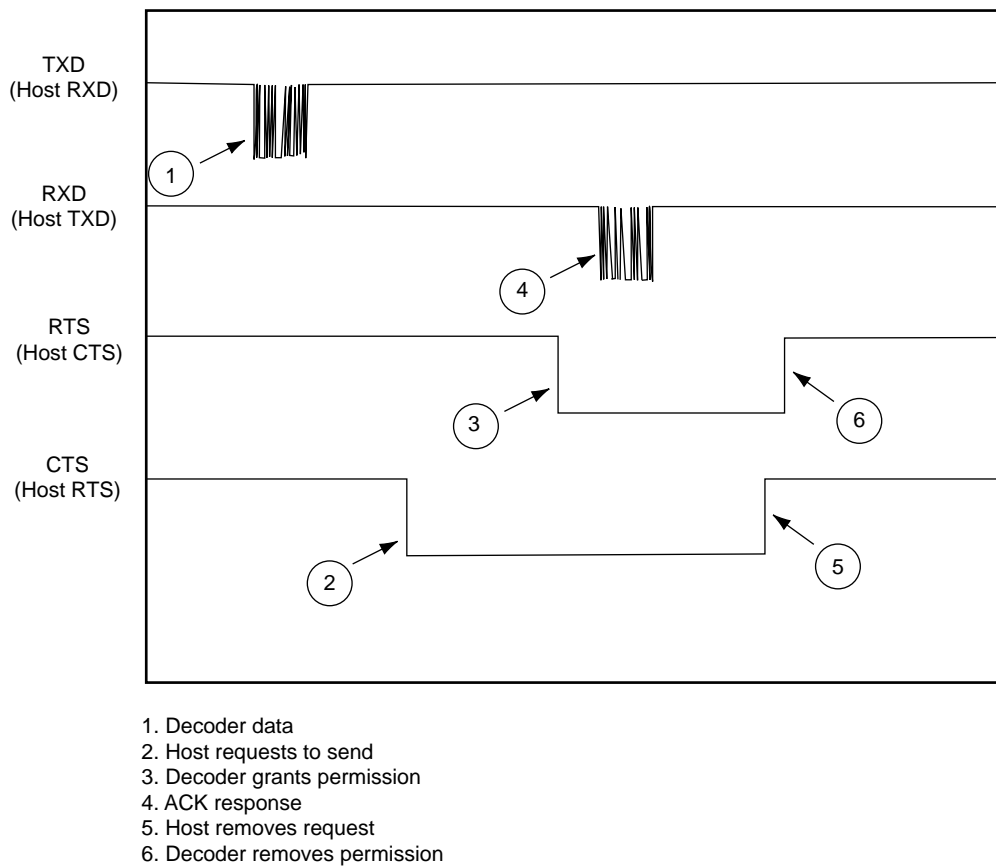
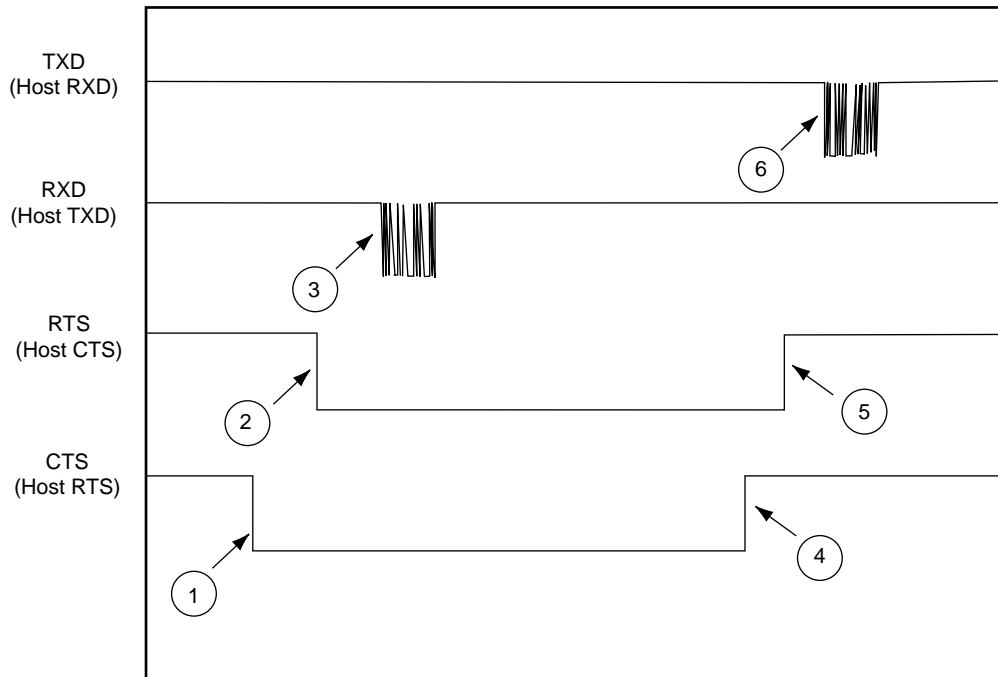
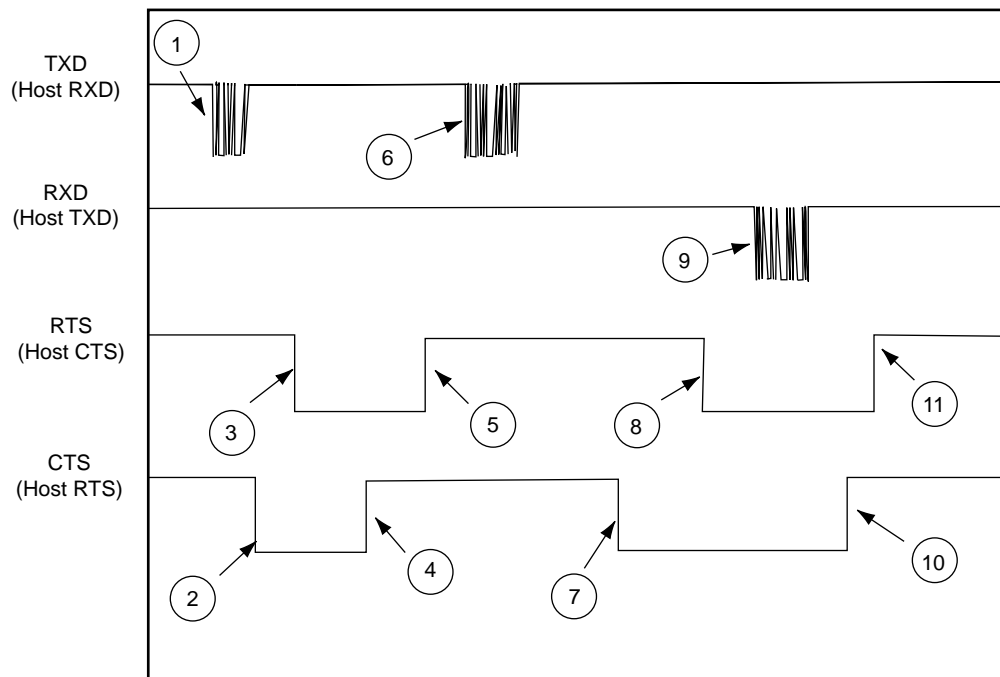


Figure A-1 *Basic Decoder Initiated Transaction*



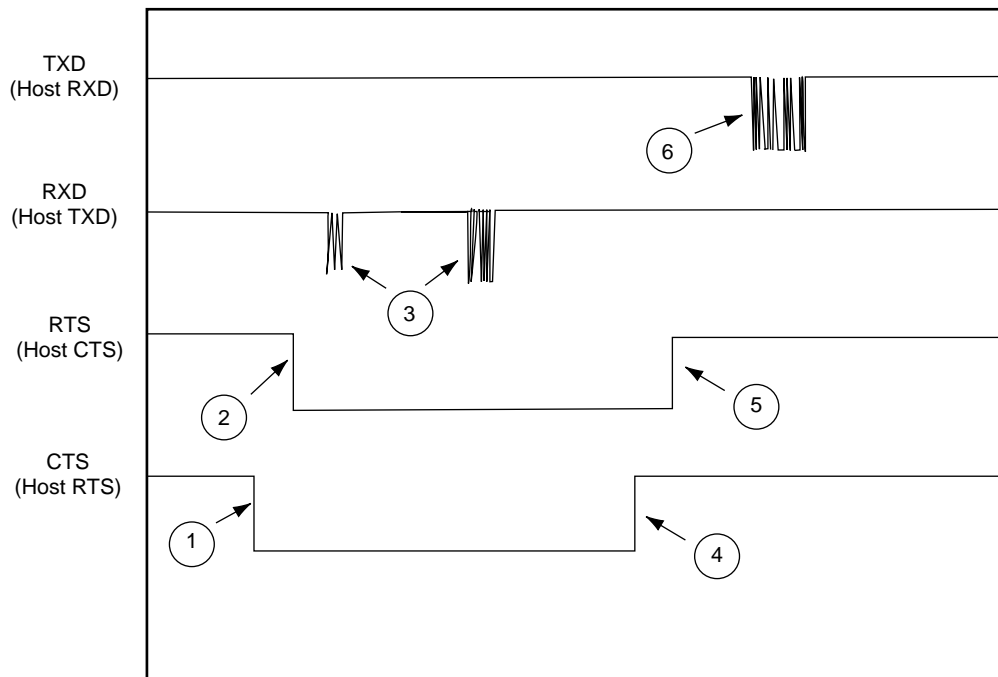
1. Host requests to send
2. Decoder grants permission
3. BEEP command sent
4. Host removes request
5. Decoder removes permission
6. Decoder ACKs

Figure A-2 Basic Host Initiated Transaction



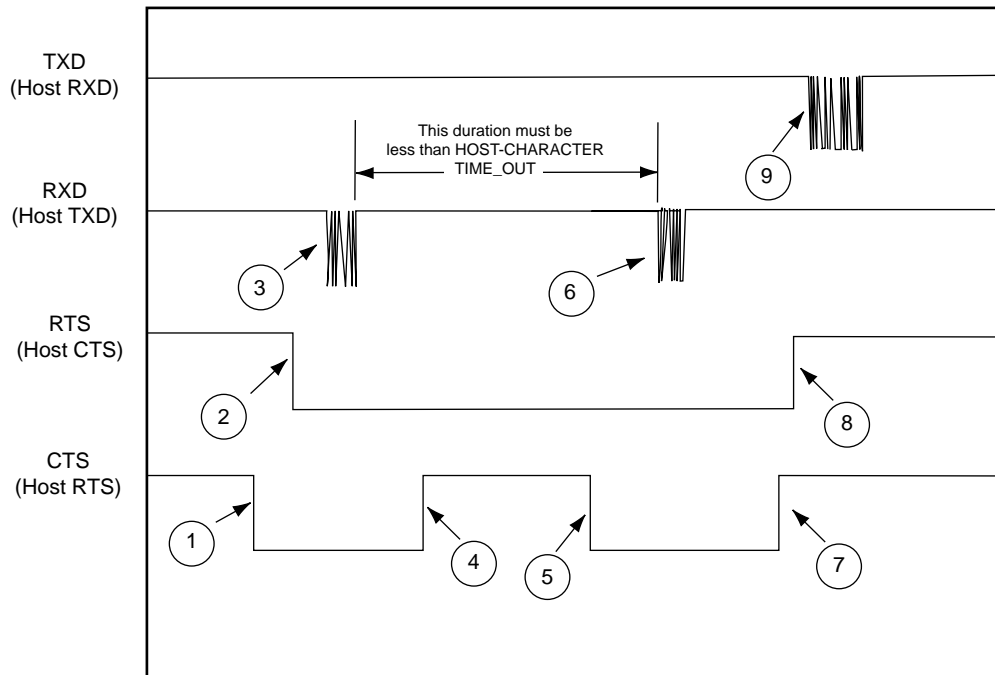
1. Decoder starts to transmit
2. Host asserts RTS causing transmission pause
3. Decoder grants permission for host to send
4. Host removes request without sending
5. Decoder removes permission
6. Decoder resumes transmission
7. Host requests permission to send ACK
8. Decoder grants permission
9. Host sends ACK
10. Host removes request when finished sending
11. Decoder removes permission

Figure A-3 *Host Interrupting Decoder's Transmission*



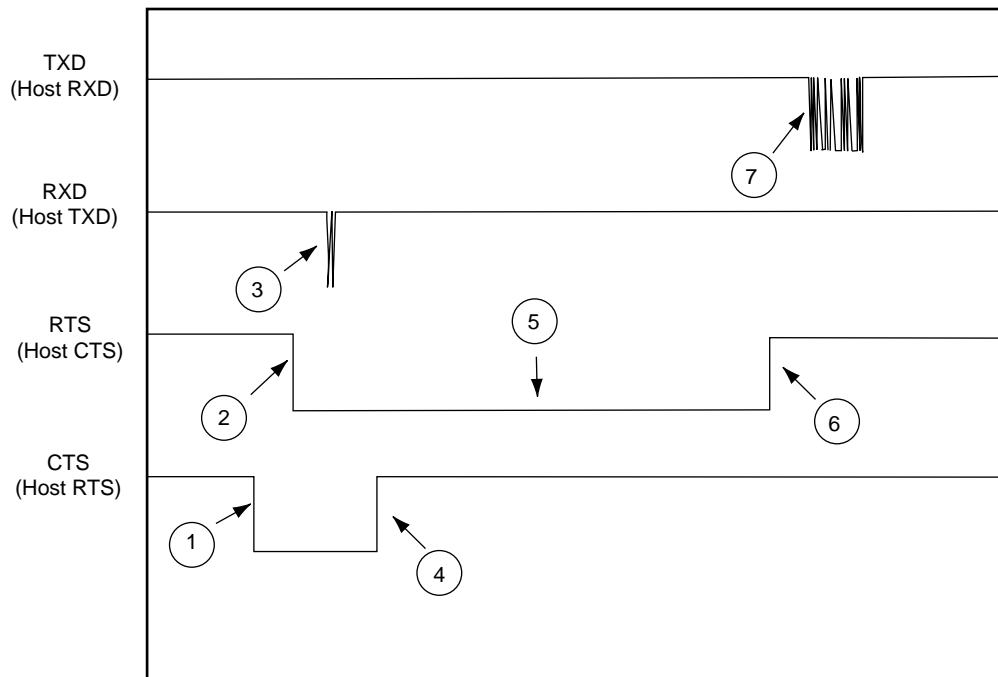
1. Host requests permission to send
2. Decoder grants permission
3. Host sends 3 nulls, then BEEP command
4. Host removes request when finished sending
5. Decoder removes permission
6. Decoder ACKs

Figure A-4 Host Initiated Transmission with Leading Nulls (Decoder in Continuous Power Mode)



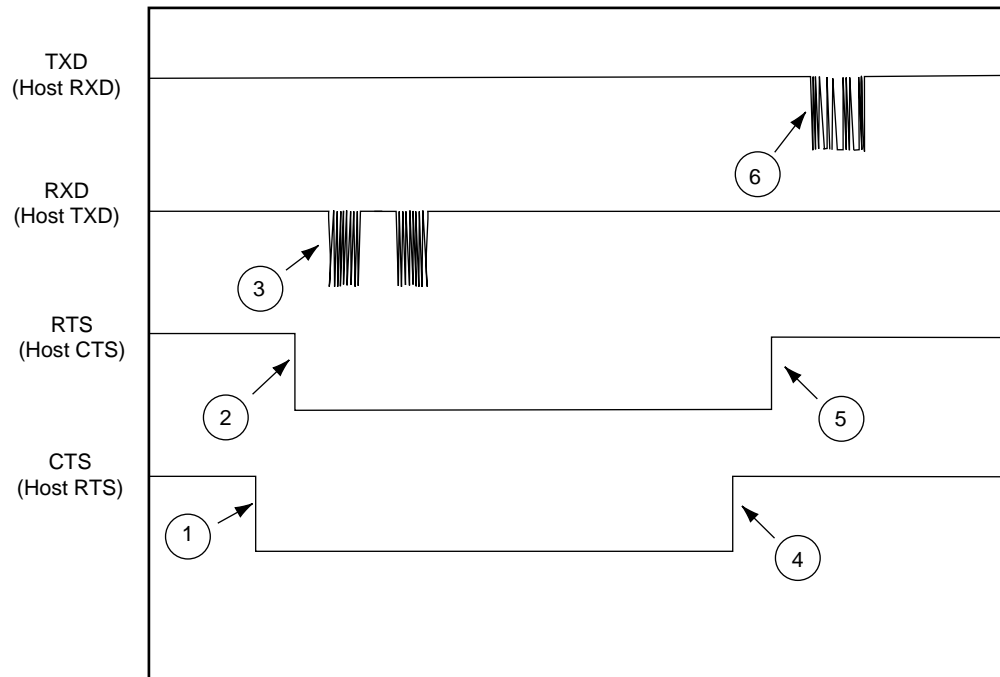
1. Host requests permission to send
2. Decoder grants permission
3. Host sends 1/2 BEEP command
4. Host removes request (ignored by decoder until transmit complete or timed out)
5. Host requests again (ignored by decoder until transmit complete or timed out)
6. Host sends remainder of BEEP command
7. Host removes request
8. Decoder removes permission
9. Decoder ACKs

Figure A-5 *Host Initiated Transaction with Host Pausing and Releasing RTS During Transmission*



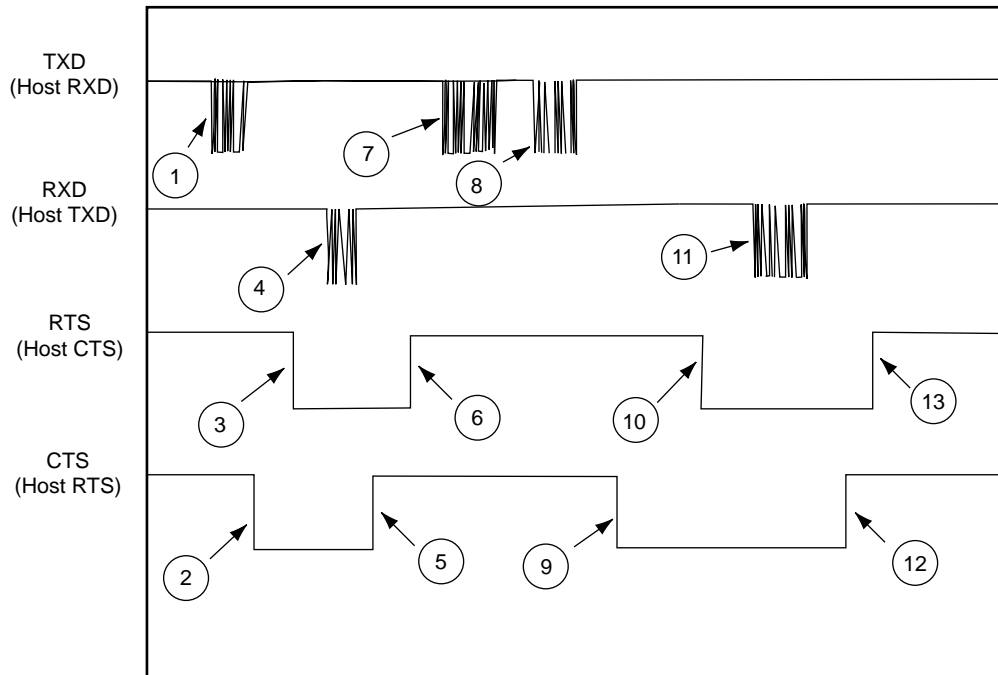
1. Host requests permission to send
2. Decoder grants permission
3. Host sends 2 characters of message
4. Host removes request
5. RTS remains low because decoder is still expecting data
6. Decoder times out waiting for a character and removes permission
7. Decoder sends a NAK resend

Figure A-6 Error Transmission: Host Sends Only First 2 Characters of 6 Character Message



1. Host requests permission to send
2. Decoder grants permission
3. Host sends 2 BEEP commands instead of 1
4. Host removes request
5. Decoder removes permission
6. Decoder ACKs first BEEP command

Figure A-7 Error Condition: Host Sends 2 Valid BEEP Commands Back to Back



1. Decoder starts to transmit
2. Host requests permission
3. Decoder grants permission
4. Host causes abort by sending BEEP
5. Host removes request
6. Decoder removes permission
7. Decoder ACKs
8. Decoder resends data
9. Host requests permission
10. Decoder grants permission
11. Host ACKs
12. Host removes request
13. Decoder removes permission

Figure A-8 *Host Causes Decoder to Abort Transmission*

Appendix B Mandatory Parameter

The Parameter Scanning option is required for each product using SSI. This parameter enables and disables parameter bar code scanning. The default value for this parameter must be 1, Enable.

When SSI hosts establish communication, they typically set this option to 0 (disable), which disables parameter scanning. The hosts should query this parameter periodically. If the host detects the value is 1, it infers the user scanned either the Set Defaults bar code or Enable Parameter Scanning bar code. The host may then take remedial action, which may include determining and resetting all parameters via the SSI interface.

Index

A

ACK/NAK handshaking 1-7
aim mode 2-9

B

beep code definitions 2-11

C

commands

ABORT_MACRO_PDF 2-6
AIM_OFF 2-7
AIM_ON 2-8
BATCH_DATA 2-17
BATCH_REQUEST 2-18
BEEP 2-10
CAPABILITIES_REPLY 2-14
CAPABILITIES_REQUEST 2-13
CHANGE_ALL_CODE_TYPES 2-19
CMD_ACK 2-20
CMD_ACK_ACTION 2-22
CMD_NAK 2-24
CUSTOM_DEFAULTS 2-27
DECODE_DATA 2-28
EVENT 2-40
FLUSH_MACRO_PDF 2-42
FLUSH_QUEUE 2-43
ILLUMINATION_OFF 2-44
ILLUMINATION_ON 2-45
IMAGE_DATA 2-46
IMAGER_MODE 2-48
LED_OFF 2-49
LED_ON 2-50
list by opcode 2-4
list, alphabetical 2-1

PAGER_MOTOR_ACTIVATION 2-51
PARAM_DEFAULTS 2-52
PARAM_REQUEST 2-53
PARAM_SEND 2-56
REPLY_REVISION 2-58
REQUEST_REVISION 2-59
SCAN_DISABLE 2-60
SCAN_ENABLE 2-61
SLEEP 2-62
SSI_MGMT_COMMAND 2-63
START_SESSION 2-64
STOP_SESSION 2-65
VIDEO_DATA 2-66
WAKEUP 2-68

conventions

notational xii

E

event codes 2-41

H

hardware handshaking 1-3
 decoder transmission 1-4
 host reception 1-6
 host transmission 1-3, 1-4
hardware signals 1-2

I

image preamble fields 2-47

M

message packets 1-9
 multipacketing 1-9

packet format 1-10
 multipackaging 1-9

N

NAK types 2-26
 notational conventions xii

P

packet format 1-10
 packeted data 1-7
 parameter scanning option B-1
 parameter values
 requesting 2-54

S

sample code
 decoder reception 1-4
 decoder transmission 1-5
 host reception 1-6
 host transmission 1-3
 serial settings 1-2
 service information xiii
 software handshaking 1-7
 ACK/NAK 1-7
 responses 1-8, 1-9
 transfer data 1-7
 support xiii

T

transaction examples A-1
 abort transmission A-8
 decoder initiated A-1
 error condition A-7
 error transmission A-6
 host initiated A-2, A-4, A-5
 host interruption A-3
 transmission
 decoder to host 1-4, 1-6
 host to decoder 1-3, 1-4
 transmission responses 1-8, 1-9

U

unpacked data 1-7

V

video preamble fields 2-67

Z

Zebra Technologies support xiii



Zebra Technologies Corporation
Lincolnshire, IL U.S.A.
<http://www.zebra.com>

Zebra and the stylized Zebra head are trademarks of ZIH Corp., registered in many jurisdictions worldwide. All other trademarks are the property of their respective owners.

©2015 ZIH Corp and/or its affiliates. All rights reserved.



72E-40451-03 Revision A - May 2015

