



OnBoardRIP Software Integration Guide

Document No: IJM017



Document Revision Status:

REVISIONS			
LTR	DESCRIPTION	INIT	DATE
-	Release	ZR	7/30/09
A	Updated PEN_ID command structure. Added LOAD_XML_JOB (0xAB) command. Added Section 8.0 Replaced Appendix B Job Structure with Barcode App. Notes	ZR	6/22/10
B	Added Error Code 28 Update structure for SET_PEN_CONFIGURATION (0x8A)	ZR	8/04/10

Reference Documents (Latest Revisions Thereof):

REFERENCES			
LTR	DOC NO	TITLE	CONTROLLER/ORIGINATOR
<u>a</u>			inc.jet
<u>b</u>			inc.jet

Inc.jet Inc welcomes input from its customers. Any questions or comments can be emailed to jetengine@incjet.com.

inc.jet™ Proprietary

All information contained in this document is the property of inc.jet™ and is to be maintained as confidential. This document may not be forwarded, copied or otherwise transmitted in any form to others without prior approval of inc.jet™. No patent liability is assumed with respect to the use of the information contained herein. Moreover, inc.jet™ is constantly striving to improve its high quality products. Therefore, the information contained in this document is subject to change without notice. This document is provided as-is, assuming no liability for damages resulting from the information herein.

Table of Contents

1.0	INTRODUCTION	6
2.0	ONBOARD RIP THEORY OF OPERATION	6
3.0	COMMUNICATION OVERVIEW	7
3.1	SERIAL (RS-232) INTERFACE	7
3.2	ETHERNET INTERFACE	7
3.3	HANDSHAKING	8
3.4	UNSOLICITED MESSAGES	8
4.0	MESSAGE FORMAT	9
4.1	DATA SIZES AND BYTE ORDERING	9
4.2	BINARY PROTOCOL COMMAND STRUCTURE	10
4.2.1	Transmit Packet Format	10
4.2.2	Receive Packet Format	10
4.3	SIMPLE ASCII COMMAND STRUCTURE	11
4.3.1	Print Data	11
4.3.2	Control Commands (Sent from Host to Imager)	11
4.3.3	Status Messages (Sent from Imager to Host)	11
5.0	HOW TO	12
5.1	START A STORED JOB AND PROVIDE DYNAMIC DATA	12
5.2	CONFIGURE IMAGER TO PRINT	14
5.3	SETUP PEN OFFSETS OR STITCH PENS HORIZONTALLY	15
5.4	EXECUTE SERVICE STATION OPERATIONS	16
6.0	BINARY PROTOCOL	18
6.1	POWER_UP_MSG (0x01)	20
6.2	RASTER_BUFFER_READY_FOR_DATA (0x06)	20
6.3	ACKNOWLEDGE_MSG (0x09)	22
6.4	STOP_PRINTING (0x16)	23
6.5	CLEANING_STATION (0x18)	23
6.6	PURGE_DONE (0x19)	25
6.7	SET_OUTPUT_CHANNEL (0x21)	25
6.8	ENABLE_PRINTING (0x25)	26
6.9	READY_TO_PRINT (0x26)	26
6.10	KEEP_ALIVE (0x30)	26
6.11	IMAGER_ERROR (0x34)	27
6.12	SAVE_CONFIGURATION (0x40)	29
6.13	SENSOR_INPUT_ENABLE (0x41)	30
6.14	OUTPUT_CONTROL (0x42)	30
6.15	INPUT_CONTROL (0x43)	32
6.16	CONFIGURATION_SAVED (0x44)	32
6.17	INPUT_LEVEL_CHANGED (0x47)	33
6.18	IMAGER_LOGGING_REQUEST (0x48)	34
6.19	IMAGER_LOG_DATA (0x49)	35



6.20	PRINT_RECORD (0x55)	36
6.21	START_PRINT (0x56)	38
6.22	START_DEFAULT_PRINT_JOB (0x57)	39
6.23	REQUEST_SYSTEM_READY_STATUS (0x58)	39
6.24	SYSTEM_READY_TO_PRINT_STATUS (0x59)	39
6.25	SET_RTC (0x62)	40
6.26	GET_RTC (0x63)	40
6.27	CURRENT_RTC (0x64)	41
6.28	REPORT_IO_STATES (0x65)	41
6.29	IO_STATES (0x66)	41
6.30	TRIGGER_PULSE_GENERATOR (0x67)	42
6.31	IMAGER_WARNING (0x68)	42
6.32	DELETE_PRINT_JOB (0x73)	44
6.33	LIST_PRINT_JOBS (0x74)	44
6.34	PRINT_JOBS_LIST (0x75)	44
6.35	SET_DFLT_PRINT_JOB (0x76)	45
6.36	GET_PEN_IDS (0x82)	45
6.37	PEN_IDS (0x83)	46
6.38	INITIALIZE_STATIC_RECORD_DATA (0x85)	46
6.39	CLEAR_PRINTING_SYSTEM (0x86)	48
6.40	PAGE_LOADED_NOTIFICATION (0x87)	48
6.41	INK_LEVEL_ALERT (0x88)	49
6.42	SET_CONTRAST (0x89)	50
6.43	SET_PEN_CONFIGURATION (0x8A)	50
6.44	REQUEST_STATUS (0x8D)	54
6.45	IMAGER_STATUS (0x8E)	54
6.46	IMAGER_CONFIGURATION_TIPS (0x8F)	60
6.47	SHIFT_PEN_OFFSET (0x92)	66
6.48	STACK_LIGHT_CONTROL (0x93)	67
6.49	EXECUTE_PURGE (0x94)	67
6.50	SKIP_PAGE (0x95)	68
6.51	GENERATE_STITCH_TABLES (0x99)	69
6.52	LOAD_COMMUNICATIONS_CONFIG (0x9A)	70
6.53	LOAD_XML_JOB (0xAB)	74
7.0	SIMPLE ASCII PROTOCOL	75
7.1	PRINT_DATA	76
7.2	Go_To_PARK_POSITION (0x20)	76
7.3	Go_To_PRINT_POSITION (0x21)	77
7.4	EXECUTE_CLEAN (0x22)	77
7.5	EXECUTE_PURGE (0x23)	77
7.6	CLEAR_PRINTER_SYSTEM (0x24)	77
7.7	GET_SYSTEM_STATUS (0x25)	78
7.8	START_PRINT (0x26)	78
7.9	STOP_PRINT (0x27)	78
7.10	REDIRECT_IMAGER_MESSAGES (0x35)	78
7.11	ENABLE_PRINTING (0x37)	79
7.12	INITIALIZE_STATIC_RECORD_DATA (0x40)	79
7.13	EJECT_SERVICE_STATION_TRAY (0x41)	79
7.14	COMMAND_ACKNOWLEDGE (0x20)	80
7.15	DATA_BUFFER_AVAILABLE (0x28)	80



7.16	CLEANING_STATION_COMMAND_SUCCESS (0x29)	80
7.17	CLEANING_STATION_COMMAND_FAILED (0x30)	80
7.18	IMAGER_ERROR (0x31)	80
7.19	IMAGER_WARNING (0x32)	82
7.20	SYSTEM_NOT_READY (0x33)	82
7.21	SYSTEM_READY (0x34)	82
7.22	DATA_ACKNOWLEDGE (0x36)	82
7.23	PURGE_CYCLE_COMPLETED (0x38)	83
7.24	PAGE_PRINTED (0x39)	83
8.0	XML JOB STRUCTURE LOAD_XML_JOB (0xAB)	84
8.1	JOB CONFIGURATION	84
8.2	JOB FILE STRUCTURE	84
8.3	MAIN JOB TAGS	85
8.4	RASTER BUFFER CONFIGURATION SEGMENT	86
8.5	RIP CONFIGURATION BLOCKS	87
8.5.1	Field Block	88
8.5.2	Record Block	95
8.5.3	Bitmap Block	96
8.6	EMBEDDING FONTS AND BITMAPS	98
APPENDIX A: EXPLANATION OF STITCHING TABLE PARAMETERS		100
APPENDIX B: BARCODE PARAMETERS APPLICATION NOTES		104

1.0 Introduction

This document provides explanation of the Binary and Simple ASCII protocols used to communicate with jet.engine printers in OnBoardRIP mode. The sections below will provide:

- Software interface overview.
- Procedures to configure the printers.
- Procedures to properly start jobs and provide dynamic data.
- List of commands used in OnBoardRIP operation.
- Serial and Ethernet interface specification.
- XML job definition.

This document will reference printers using several names. Following is an explanation of the names.

- jet.engine – the processor board that is used in all inc.jet printers.
- Imager -- a general term for the printers.
- jet.engine AC – an Auto Capping 3-pen printer with a service station that is programmed to perform pen capping procedure automatically.
- jet.engine MC – a Manual Capping 3-pen printer with a capping tray that has to be installed and removed manually.
- Print Controller – a rugged, stainless steel industrial box that contains a jet.engine print controller board and a power supply.
- Satellite Controller – is a stainless steel box containing a jet.engine print controller board.

The electronic version of this document contains internal links to reference topics. To access them, hold down the CTRL key and click any topic on the table of contents. The reader will then skip ahead to the appropriate section of the document.

2.0 OnBoardRIP Theory of Operation

The OnBoardRIP functionality allows a user to compose and store job templates in an Imager and print them without using a GUI. These job templates may contain static data, dynamic data, or both static data and dynamic data. A job is composed and stored in the Imager's flash memory along with hardware configurations using the OnBoardRIP GUI or custom hosts using XML files. Once the jobs and configurations are stored in the Imager, the OBR GUI can be disconnected and the Imager can print in stand-alone mode or it could be controlled with a custom host application sending simple ASCII commands and Data.

Printing may begin immediately after starting a job. For jobs requiring dynamic data, the data is received by the Imager in ASCII string form, not raster data as required by current jet.engine Imagers. Each page image is created within the Imager based on parameters given by a job definition file and dynamic data supplied by the host.



3.0 Communication Overview

Host applications can communicate with the jet.engine printing units via a Serial (RS-232) Interface or an Ethernet (10/100 Base-T) Interface. Either ASCII or Binary protocol may be assigned to any communications port other than the Main TCP ports used for main connection (10000 and 10001).

The default configurations of these interfaces can be changed using the LOAD_COMMUNICATIONS_CONFIG (0x9A) command.

3.1 Serial (RS-232) Interface

Remote applications can communicate with the printing units via a 3-wire COM 1 or COM 2 (RS-232 Interface). Commands sent over the serial port must be completely received by the imager within the configured time-out period or the partial message will be cleared from the imager's command buffer. The serial port Baud Rates can be configured to 115K, 57.6K, 38.4K, 28.8K, 19.2K and 9.2K. Default serial port parameters are listed below.

Baud Rate:	115,200
Data Bit:	8
Parity:	None
Stop Bit:	1
Flow Control:	None

Note: All Imager Log, Warning and Error messages are defaulted to the Ethernet Port 10001. When using the Serial Port, the host application should redirect appropriate messages to the Serial Port using the LOAD_COMMUNICATIONS_CONFIG (0x9A) command.

3.2 Ethernet Interface

Host applications can connect to the printing systems using the Ethernet (10/100 Base-T) Interface. All communication is done using TCP/IP protocol. By default, all Imagers are shipped with IP address 192.168.0.2. The IP address can be changed using the OnBoardRIP GUI. Messages are sent to the Imager over Main Port 10000. The Imager will reply to all requested messages on Port 10000 but any unsolicited messages (like errors, warnings, etc.) will be sent back on Main Port 10001. When connecting on the main ports the host has to connect on both ports 10000 and 10001 to create a good connection.

Additional Ports 10002 and 10003 could also be used. By default, Port 10002 is configured to Binary Protocol and can be used as a second command port. Port 10003 by default is configured to Simple ASCII protocol.

Note: Avoid setting the last number of the IP address to "0" or "255". These numbers put the Imagers into broadcast mode, which causes the Imagers to stop communicating with any host application. Imagers that have been set to broadcast have to be sent back to inc.jet for recovery.

3.3 Handshaking

The firmware will acknowledge all commands received from an outside source, confirming that the command was received and is structured properly. In Binary Protocol, the firmware will send ACKNOWLEDGE_MSG 0x09 back with the ID of the command it is acknowledging. In Simple ASCII mode, the firmware will send Data_Acknowledge 0x36 to acknowledge data and Command_Acknowledge 0x20 to acknowledge commands.

3.4 Unsolicited Messages

The firmware will issue messages without a specific request from the host. These messages include, but are not limited to: notification of print buffer space being available, warnings, errors, and cleaning station operation complete notifications.

By default, the Imager directs these messages to TCP Port 10001. If an application requires notification of errors, warnings, buffers available, cleaning station operation complete, and all other unsolicited messages, a LOAD_COMMUNICATIONS_CONFIG (0x9A) command must be sent specifying any additional ports that should receive unsolicited messages.

4.0 Message Format

inc.jet's jet.engine line of printers use two custom command protocols to interface to the printers. One is called Binary Protocol and the other is Simple ASCII Protocol. All Serial and Ethernet messages have to be organized in one of the packet formats described below.

The Imager will reply back to all messages it receives with ACKNOWLEDGE_MSG, letting the host know that it received a valid message.

All messages must be "packed" to a 1 byte packing alignment (i.e. `#pragma pack(1)`), and that all message components longer than 1 byte such as unsigned long int must be in "Big-endian" (Motorola) format as opposed to "little-endian" (Intel) format.

4.1 Data Sizes and Byte Ordering

Data Sizes

Data Type	Size (Bytes)
char, unsigned char	1
bool	1
short, unsigned short	2
int, unsigned int	4
long, unsigned long	4

Byte Ordering

Byte ordering must be "Big Endian". The most significant byte is transmitted first, followed by bytes in descending order.

Example:

```
Data to transmit:
typedef struct{
    unsigned long X = 0xABCD1234;
    char    Y = 0x55;
    short   Z = 0x4321;
} example;
```

Data appears in serial stream as:
ABCD1234554321

4.2 Binary Protocol Command Structure

All binary commands and data have to be packetized in the following format.

4.2.1 Transmit Packet Format

<STX><NumBytes><Checksum><SeqNum><MsgID><TBuf><ETX>

Field	Description	Number of Bytes
STX	Start of transmission = 0x02	1
NumBytes	Number of bytes following this parameter, beginning with Checksum field and ending with the ETX.	4
Checksum	Checksum validation byte, unused	1
SeqNum	Message sequence number	1
MsgID	Message ID	1
TBuf	Message data	Variable (0 - 4294967286)
ETX	End of Transmission = 0x03	1

4.2.2 Receive Packet Format

Imager will reply with messages packetized in the following format.

<STX><NumBytes><Checksum><RecSeqNum>><MsgID><TBuf><ETX>

Field	Description	Number of Bytes
STX	Start of transmission = 0x02	1
NumBytes	Number of bytes following this parameter, beginning with Checksum field and ending with the ETX.	4
Checksum	Checksum validation byte, unused	1
SeqNum	Message sequence number, unused	1
MsgID	Message ID	1
TBuf	Message data	Variable (0 - 4294967286)
ETX	End of Transmission = 0x03	1

4.3 Simple ASCII Command Structure

4.3.1 Print Data

Print Data should be sent to the Imager as an ASCII string with no start and stop characters. The Print Data may be any character supported by the font in use, code range from 32 to 255 decimal. Unicode is not supported. Each line of data must be terminated by at least one terminating character, default being 0x0D, carriage return line feed (0xA). Each page of data must be terminated by a unique code, default being 0xC, form feed. A complete page of data must be received by the imager before a page becomes eligible for printing.

4.3.2 Control Commands (Sent from Host to Imager)

All commands must be wrapped with a preamble and postamble, STX (0x02) and ETX (0x03) by default.

<STX><Command Data><ETX>

The command code shall be the first byte immediately following the STX byte, encoded as a binary number ranging from 0x20 to 0xFF. Codes ranging from 0 to 0x1F are reserved for control codes. See Section 7.0 for command definition.

Command Data may consist of only a single byte if a command is being issued which requires no arguments. For commands which require a set of parameters be sent, an argument list, the arguments must be encoded as ASCII characters. Arguments may be separated by commas. Codes below decimal 32 are forbidden, reserved for control functions.

Example: To send an argument list (ie. 12, 5, 17), the ASCII encoded string would be represented as: 0x31 0x32 0x2c 0x35 0x2c 0x31 0x37.

4.3.3 Status Messages (Sent from Imager to Host)

All messages returned from the Imager will be wrapped with a preamble and postamble, STX (0x02) and ETX (0x03) by default.

<STX><Message Data><ETX>

See Section 7 for a complete list of Message Codes.

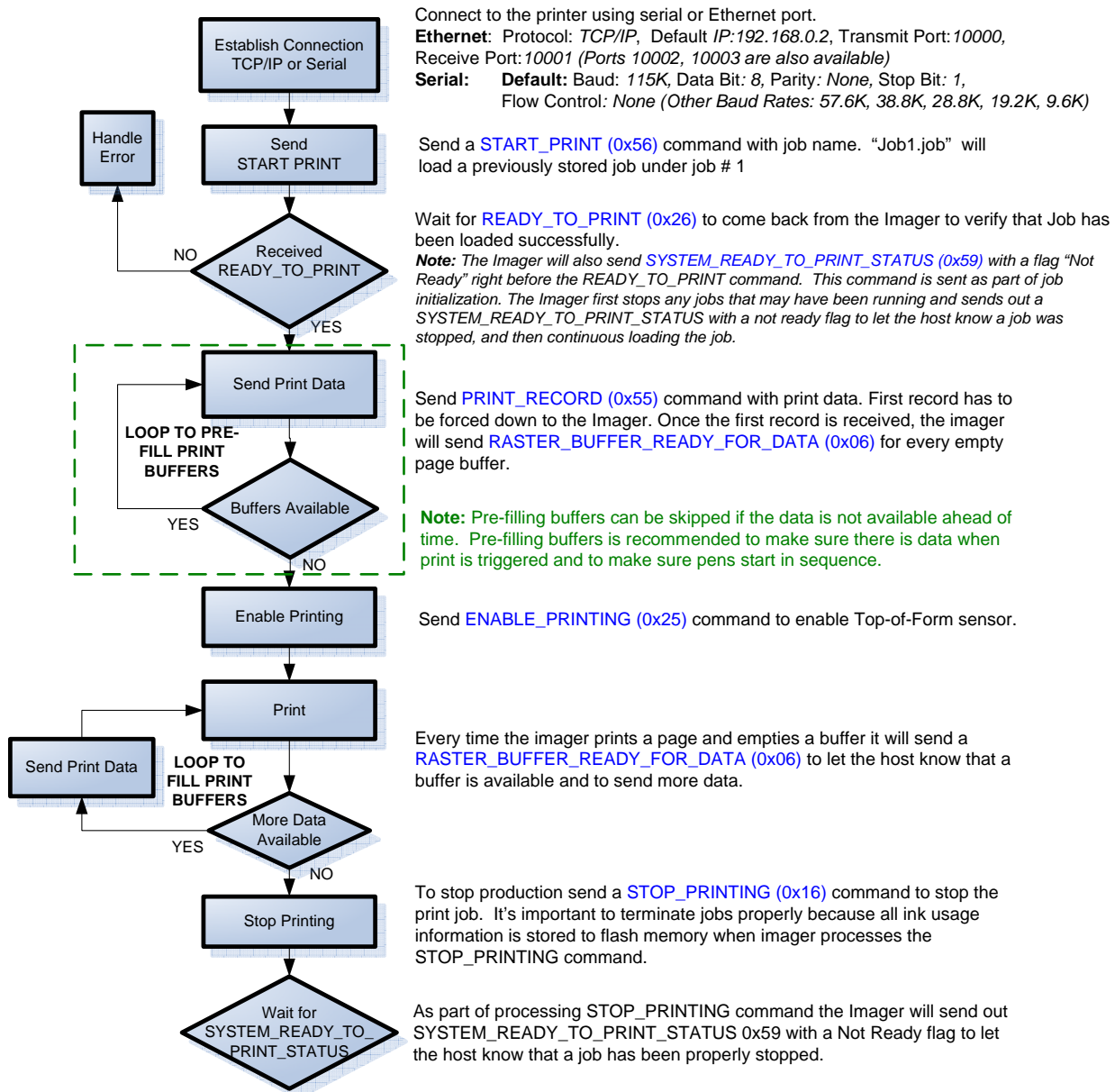
5.0 How To...

5.1 Start a Stored Job and Provide Dynamic Data

Binary Protocol

The flow chart below shows steps and commands that need to be sent to the Imager to load a previously stored job and provide dynamic data using binary protocol. Hosts should utilize this technique on an application that will be printing dynamic data that will change on every piece.

Note: The Imager replies with [ACKNOWLEDGE_MSG\(0x09\)](#) to the Host when a general command message is received confirming to the Host the command has been received without error. This command is not included in the flow chart. It assumes the Host will wait for this message every time it sends a command to the Imager.

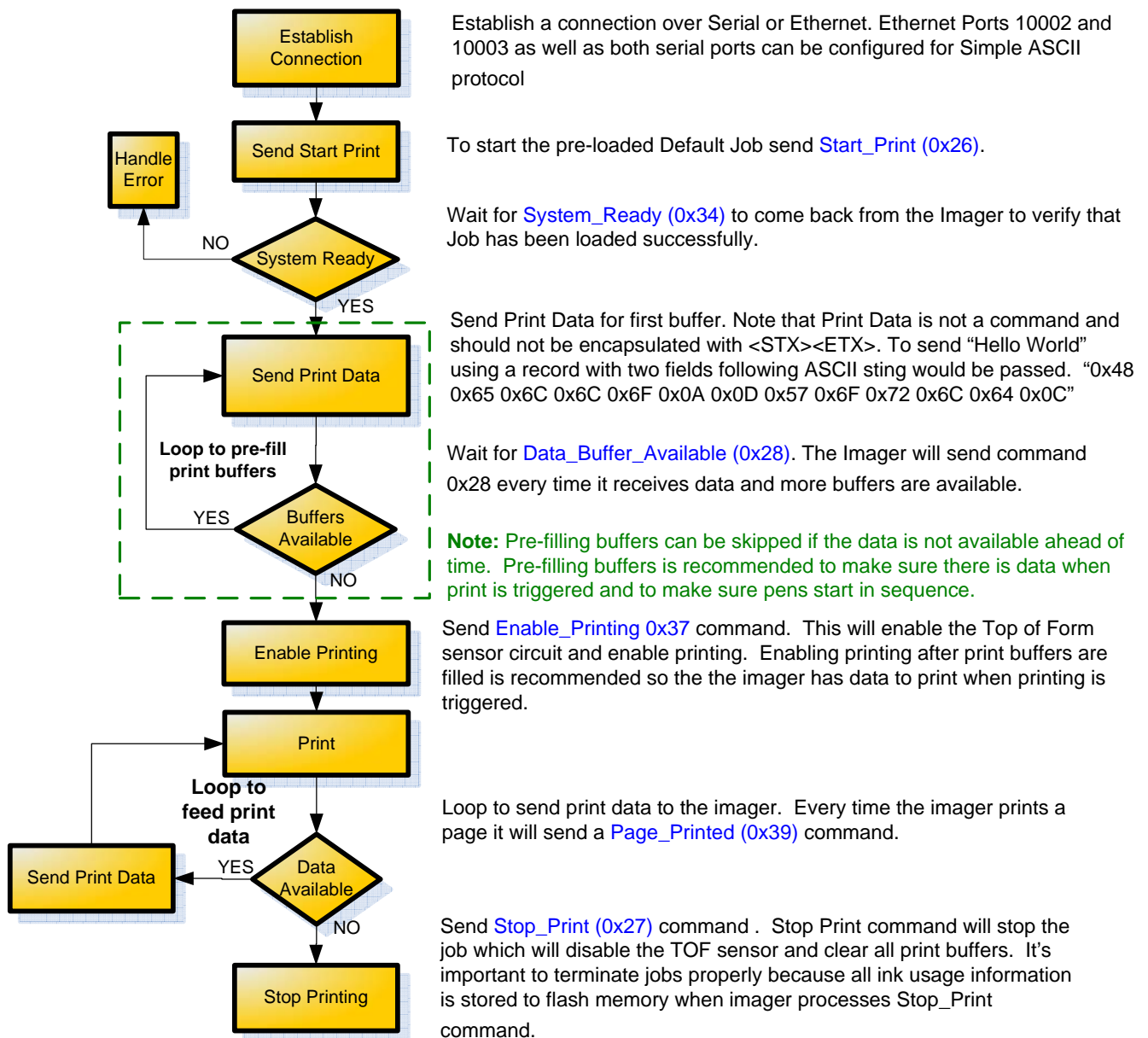


Simple ASCII Protocol

The flow chart below shows steps and commands that need to be sent to the Imager to load a default job and provide dynamic data using Simple ASCII protocol. Hosts should utilize this technique on applications that will be printing dynamic data that will change on every piece.

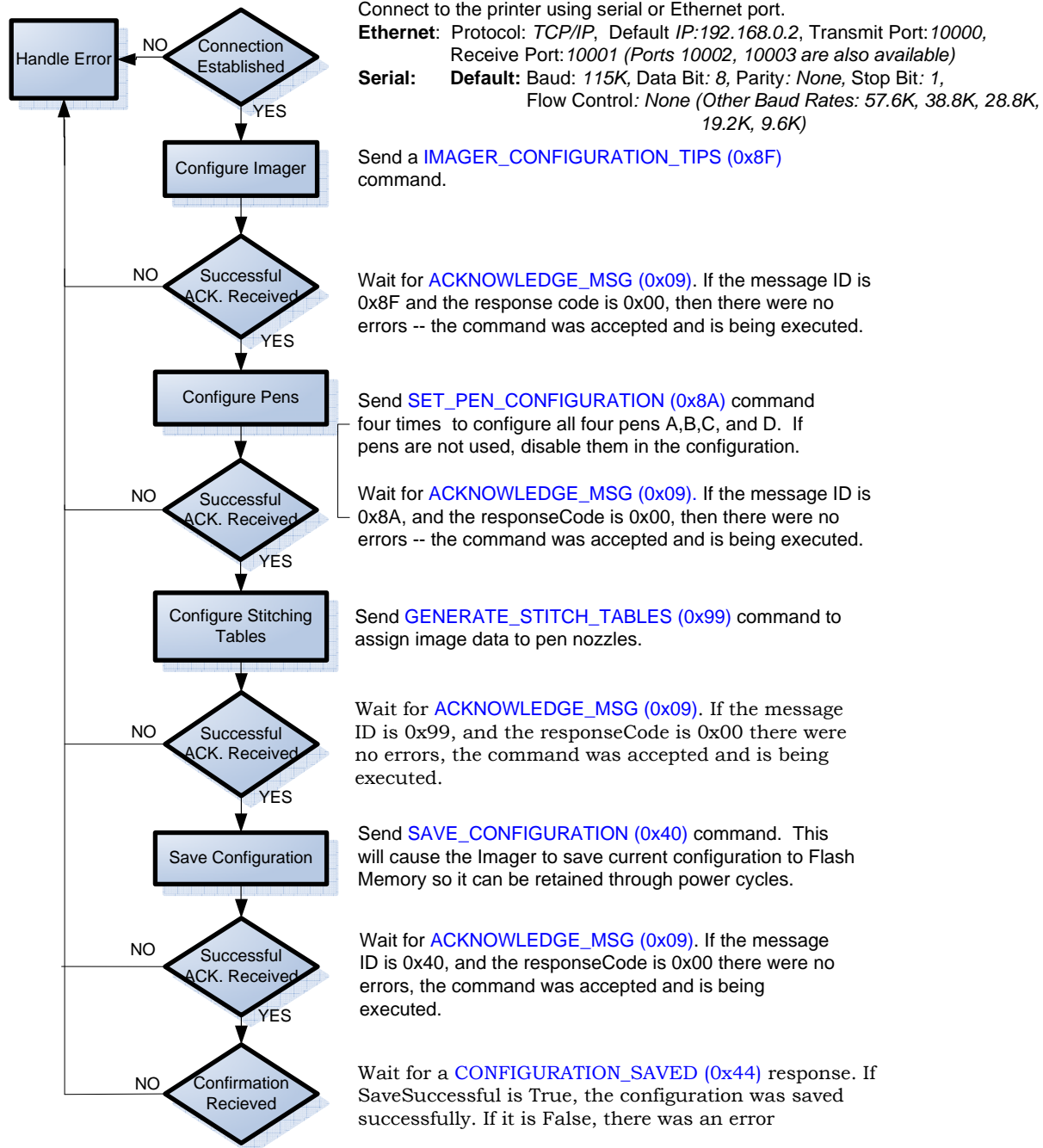
Note1: The Imager replies with [Command_Acknowledge\(0x20\)](#) to the Host when a general command message is received with no errors. This command is not included in the flow chart. It assumes the Host will wait for this message every time it sends a command to the Imager.

Note2: The communication setup has to be configured so that unsolicited messages which fall under Print Control and Status 1 are sent back on the connected port. Use OnBoardRIP GUI or [LOAD_COMMUNICATION_CONFIG 0x9A](#) command in binary protocol to configure the ports and unsolicited messages.



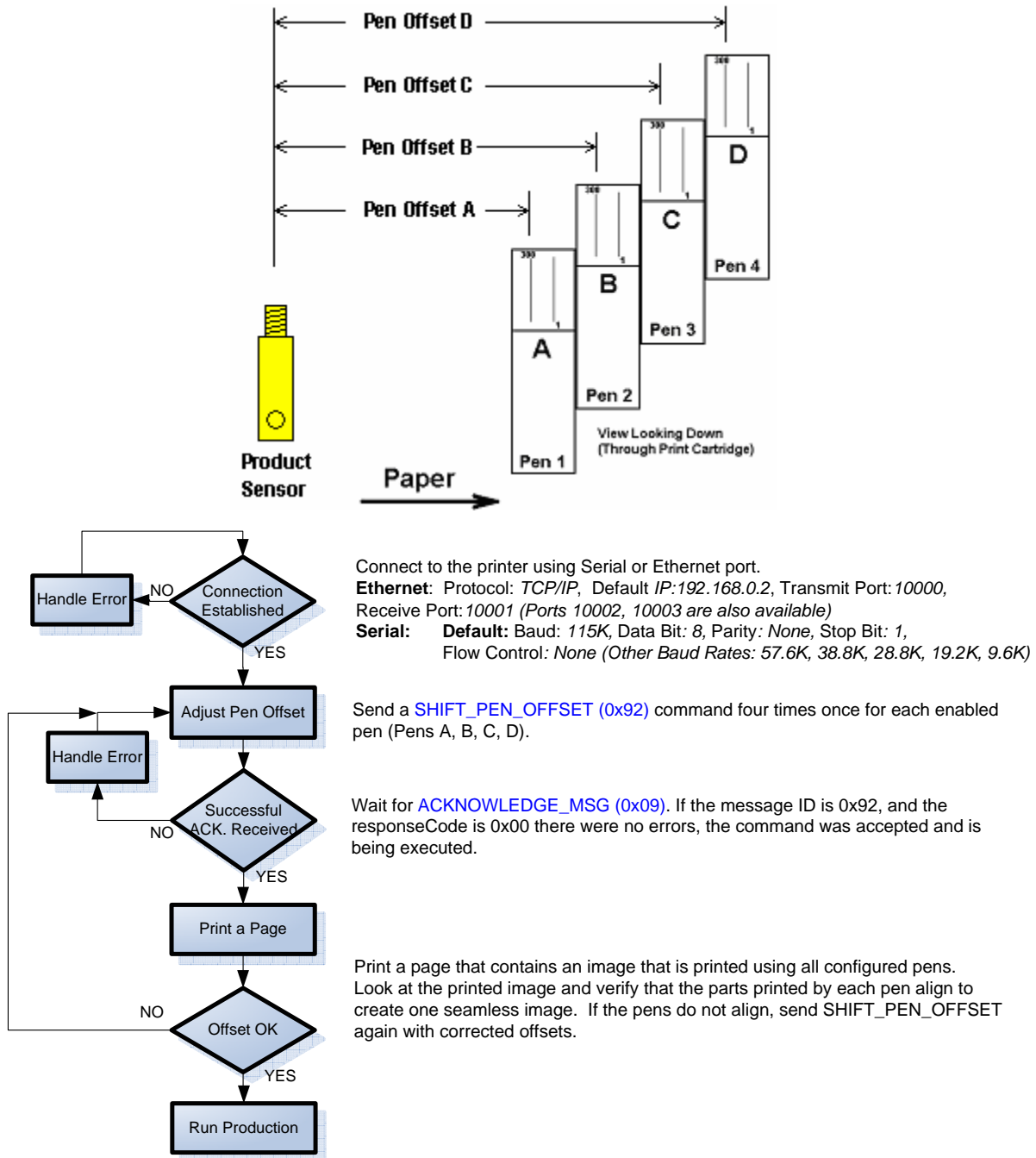
5.2 Configure Imager to Print

The Imager can only be configured using Binary Protocol. In order to fully configure the Imagers for printing, several commands have to be sent. The flow chart below lists all the required commands and the sequence in which they should be sent.



5.3 Setup Pen Offsets or Stitch Pens Horizontally

As part of normal setup for printing, the host has to adjust the print trigger delay to be able to stitch pens together and place the print image on the media correctly. Trigger delay or Pen Offset is the distance from the product sensor to the first nozzle column of the pen. The distance is measured in encoder ticks. The host has to set the distance to each pen individually. Below is a flow chart that explains the steps and commands needed to adjust pen offsets.

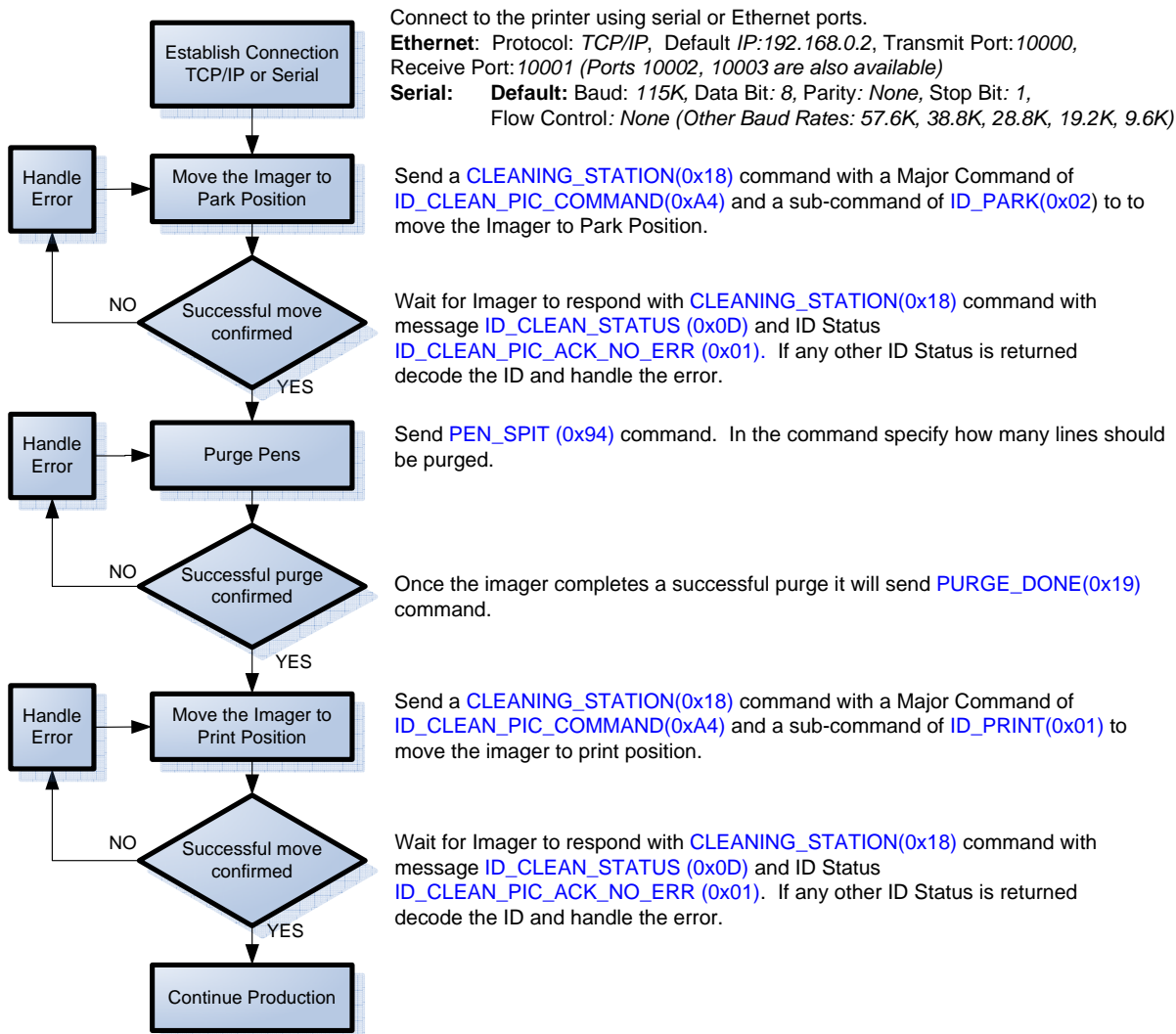


5.4 Execute Service Station Operations

Binary Protocol

The flow chart below lists the steps required by the host to execute a purge into the service station using Binary Protocol. The example will move the service station into park position, execute a spit, and move the service station back into print position. This procedure could be used to recover dry pen nozzles after they sit uncapped for a while during production.

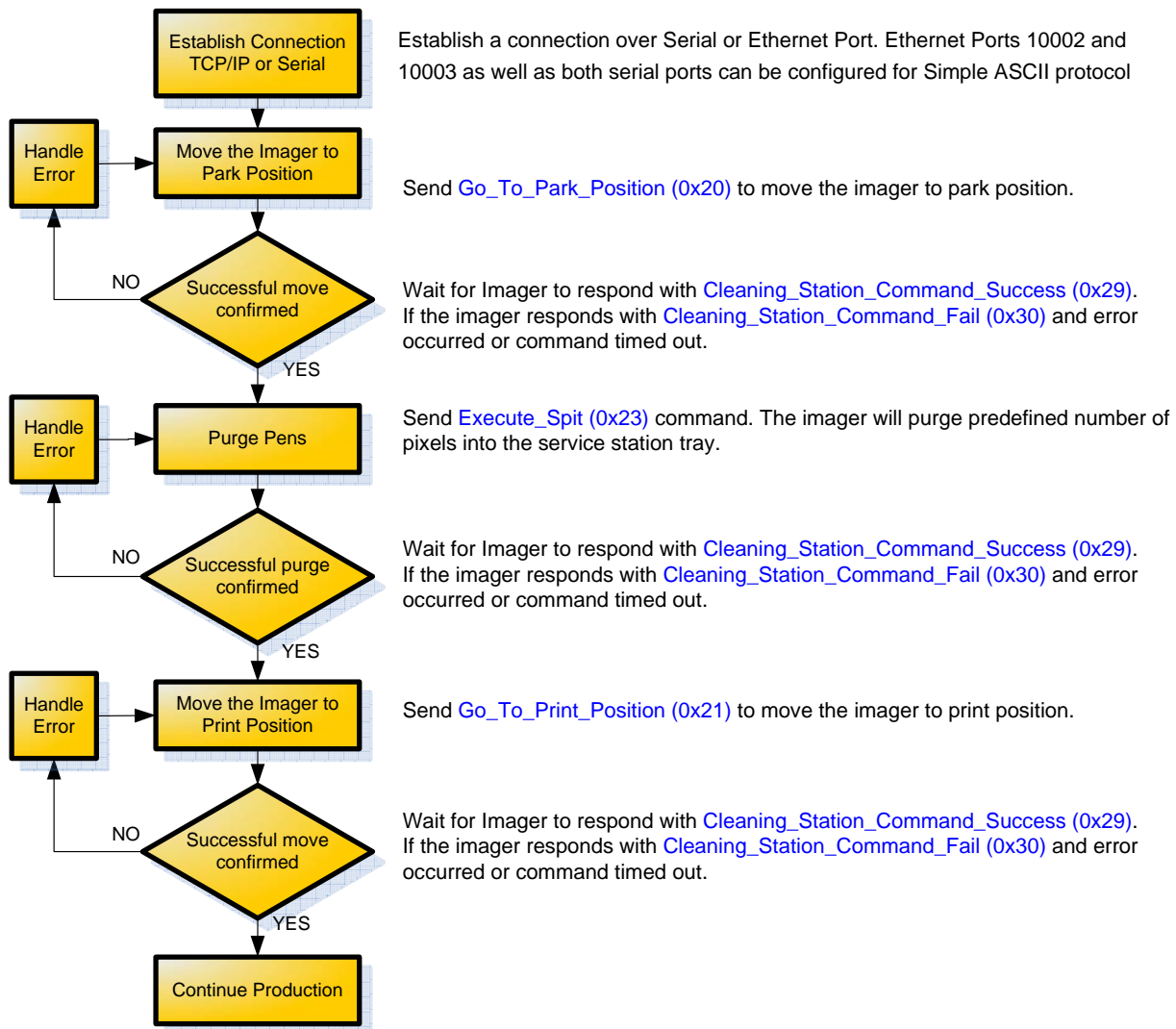
Note: The Imager replies with [ACKNOWLEDGE_MSG\(0x09\)](#) to the Host when a general command message is received confirming to the Host the command has been received without error. This command is not included in the flow chart. It assumes the Host will wait for this message every time it sends a command to the Imager.



Simple ASCII Protocol

The flow chart below lists the steps required by the host to execute a purge into the service station using Simple ASCII protocol. The example will move the service station into park position, execute a spit, and move the service station back into print position. This procedure could be used to recover dry pen nozzles after they sit uncapped for a while during production.

Note: The Imager replies with [Command_Acknowledge\(0x20\)](#) to the Host when a general command message is received confirming to the Host the command has been received without error. This command is not included in the flow chart. It assumes the Host will wait for this message every time it sends a command to the Imager.





6.0 Binary Protocol

Message	Code	Origin	Socket	Definition
COMMUNICATION				
POWER_UP_MSG	0x01	Imager	Cmd	Indicates a new connection has been made between imager and host.
ACKNOWLEDGE_MSG	0x09	Imager	Event	Imager acknowledge to a host command.
KEEP_ALIVE	0x30	Host	Cmd	Used by Host and Imager to verify active communication.
LOAD_COMMUNICATIONS_CONFIG	0x9A	Host	Cmd	Configure serial and TCP/IP ports and protocols.
STATUS/ERROR/WARNING				
REQUEST_STATUS	0x8D	Host	Cmd	Request for imager status.
IMAGER_STATUS	0x8E	Imager	Cmd	Response to REQUEST_STATUS.
IMAGER_ERROR	0x34	Imager	Event	Message transmitted from Imager to Host, indicates an error event has taken place.
IMAGER_WARNING	0x68	Imager	Event	Message transmitted from Imager to Host, indicates a warning event has taken place.
INK_LEVEL_ALERT	0x88	Imager	Event	Imager generated message signaling an ink level threshold has been reached or exceeded
REQUEST_SYSTEM_READY_STATUS	0x58	Host	Cmd	Request system ready to print status. Imager will return a SYSTEM_READY_TO_PRINT_STATUS
SYSTEM_READY_TO_PRINT_STATUS	0x59	Host	Cmd	System ready to print status. (Ready = 1; not Ready = 0).
PRINT CONTROL				
START_PRINT	0x56	Host	Cmd	Command used to initialize print.
PRINT_RECORD	0x55	Host	Cmd	Command used to send dynamic data to the Imager.
STOP_PRINTING	0x16	Host	Cmd	Host command to imager to stop printing
ENABLE_PRINTING	0x25	Host	Cmd	Enables the Top Of Form trigger. Could be used to synchronize all the Imagers.
READY_TO_PRINT	0x26	Imager	Event	Imager loaded a job successfully and is ready to print.
LOAD_XML_JOB	0xAB	Host	Cmd	Command loads XML based job configuration onto Imager and starts the job.
SENSOR_INPUT_ENABLE	0x41	Host	Cmd	Command to Enable/Disable Input sensor. Could be used to temporarily disable printing.
START_DEFAULT_JOB	0x57	Host	Cmd	Start default print job stored in imager flash.
CLEAR_PRINTING_SYSTEM	0x86	Host	Cmd	Command imager to clear all image buffers and any pieces that were queued in by the sensor.
PAGE_LOADED_NOTIFICATION	0x87	Imager	Event	Imager generated message signaling a page is loaded and is ready to be printed.
INITIALIZE_STATIC_RECORD_DATA	0x85	Host	Cmd	Send Static Data Record to Imager.
SKIP_PAGE	0x95	Host	Cmd	Request a page to be skipped.
RASTER_BUFFER_READY_FOR_DATA	0x06	Imager	Event	Raster buffer ready for new data.
SET_CONTRAST	0x89	Host	Cmd	Host command used to set vertical contrast.



LOGGING

IMAGER_LOGGING_REQUEST	0x48	Host	Cmd	Enable debug logging at a specified level of detail.
IMAGER_LOG_DATA	0x49	Imager	Event	This command transfers Imager logging messages to the host.

IMAGER AND PEN CONFIGURATION

GENERATE_STITCH_TABLES	0x99	Host	Cmd	Command to Generate stitch tables based on parameters given.
IMAGER_CONFIGURATION_TIPS	0x8F	Host	Cmd	Command to set Imager hardware configuration.
SET_PEN_CONFIGURATION	0x8A	Host	Cmd	Command used to configure pen firing parameters.
SAVE_CONFIGURATION	0x40	Host	Cmd	Instructs the Imager to save current configuration to flash memory.
CONFIGURATION_SAVED	0x44	Imager	Event	Notifies the host configuration saved correctly.
GET_PEN_IDS	0x82	Host	Cmd	Request IDs of all pens installed.
PEN_IDS	0x83	Imager	Cmd	Response to GET_PEN_IDS
SHIFT_PEN_OFFSET	0x92	Host	Cmd	Command to change Pen TOF on the fly while printing.

JOB MANAGEMENT

LIST_PRINT_JOBS	0x74	Host	Cmd	Request that the Imager return the number and the names of the current stored Print Jobs
PRINT_JOBS_LIST	0x75	Imager	Cmd	Response to a LIST_PRINT_JOBS.
DELETE_PRINT_JOB	0x73	Host	Cmd	Command to delete specific job.
SET_DFLT_PRINT_JOB	0x76	Host	Cmd	Command to set the Default Print Job.

I/O

SET_OUTPUT_CHANNEL	0x21	Host	Cmd	Set imager output channel.
OUTPUT_CONTROL	0x42	Host	Cmd	Configures the output channel specified to pulse when triggered using parameters specified
INPUT_CONTROL	0x43	Host	Cmd	Configures Imager to notify the host if the level of the specified Input changes.
INPUT_LEVEL_CHANGED	0x47	Imager	Event	Notifies the host of the Input Level Change.
REPORT_IO_STATES	0x65	Host	Cmd	Request state of inputs and outputs to host.
IO_STATES	0x66	Imager	Cmd	Input and Output state report for host.
TRIGGER_PULSE_GENERATOR	0x67	Host	Cmd	Trigger requested pulse generator (configured in OUTPUT_CONTROL command).
STACK_LIGHT_CONTROL	0x93	Host	Cmd	Configure Stack light control for 4 outputs to: FLASH, SOLID or OFF.

SERVICE STATION and PURGE COMMANDS

CLEANING_STATION	0x18	Imager/ Host	Cmd	Cleaning station command or response. May originate at either host or imager.
EXECUTE_PURGE	0x94	Host	Cmd	Command all pens to spit supplied number of pixels.



PURGE_DONE	0x19	Imager	Event	Response after completing EXECUTE_PURGE.
CLOCK				
SET_RTC	0x62	Host	Cmd	Set imager RTC to date/time specified by host.
GET_RTC	0x63	Host	Cmd	Request imager return instantaneous value of its RTC (UTC).
CURRENT_RTC	0x64	Imager	Cmd	Instantaneous value of imager RTC (UTC).

6.1 POWER_UP_MSG (0x01)

Description

Message transmitted from Imager to Host, indicates a new connection has been made between Imager and host.

Data

```
typedef struct {
    unsigned long int    uliData;
    unsigned char       protocolRev[];
} sSysPwrUp;
```

Data Description

[uliData](#) Firmware revision number
[protocolRev](#) Null terminated ASCII string containing the highest protocol revision that this Imager understands. (Decimal representation) IE: "216_017" where 16 is Major Firmware Revision and 017 is the protocol revision for this firmware.

Sample Message

The command received at the host with a firmware revision of 5162027
 And a protocol revision of 216_017 will look like the following:

STX	Num of Bytes					Check sum	Seq Num	Msg ID	uliData				protocolRev								ETX
02	00	00	00	10		00	00	01	00	4E	C4	2B	32	31	36	5F	30	31	37	00	03

6.2 RASTER_BUFFER_READY_FOR_DATA (0x06)

Description

This message is generated by the Imager every time it prints a page or a free buffer is available during initialization. **pagePrinted= True(0x01)** indicates the buffer was emptied by a printed page.

Data

```
typedef struct {
    unsigned char    ucBufferNum;
    unsigned char    ucNumFPGABuffsRemaining;
    unsigned long int uliPageNumber;
```

```

        unsigned long int PixelCount0;
        unsigned long int PixelCount1;
        unsigned long int PixelCount2;
        unsigned long int PixelCount3;
        unsigned long int EncoderCount;
        bool pagePrinted;
        unsigned char pen_A_SkipStatus;
        unsigned char pen_B_SkipStatus;
        unsigned char pen_C_SkipStatus;
        unsigned char pen_D_SkipStatus;
        unsigned char print_Status;
    }SRasterBufferReadyForData;

```

Data Description

ucBufferNum	- The zero-based raster buffer number that the Imager is telling the Host, is ready for data, due to the fact that it has already sent its data to the Gate Array to be printed, or because the buffer has just been allocated and is ready for its initial data.
ucNumFPGABuffsRemaining	- Number of FPGA Buffers Remaining to be Printed.
uliPageNumber	- Page number that this buffer printed the last time
PixelCount0	- Pixels printed, Pen 0, since start of job.
PixelCount1	- Pixels printed, Pen 1, since start of job.
PixelCount2	- Pixels printed, Pen 2, since start of job.
PixelCount3	- Pixels printed, Pen 3, since start of job.
EncoderCount	- Encoder count at time of data request.
pagePrinted	- False (0) = Data request to prefill available buffer. True (1) = Data request to refresh printed buffer.
pen_A_SkipStatus	- 0 = No print data inhibited. 1 = Page of print data inhibited 2 = Partial page of print data inhibited.
pen_B_SkipStatus	- 0 = No print data inhibited. 1 = Page of print data inhibited 2 = Partial page of print data inhibited.
pen_C_SkipStatus	- 0 = No print data inhibited. 1 = Page of print data inhibited 2 = Partial page of print data inhibited.
pen_D_SkipStatus	- 0 = No print data inhibited. 1 = Page of print data inhibited 2 = Partial page of print data inhibited.
print_Status	- 0 = No error 1 = partial page printed, some but not all pens showed an underrun status upon end of page. 2 = Pen sync errors detected

Sample Message

STX	Num of Bytes					Check sum	Seq Num	Msg ID	ucBufferNum	ucNumFPGABuffsRemaining					uliPageNumber				PixelCount0	
02	00	00	00	00	24	00	00	06	00	03					00	00	00	01	00	00

PixelCount0		PixelCount1				PixelCount2				PixelCount3				EncoderCount			
09	00	00	00	03	00	00	00	03	00	00	00	03	00	00	00	01	14

PagePrinted	Pen_A_SkipStatus	Pen_B_SkipStatus	Pen_C_SkipStatus	Pen_D_SkipStatus	Print_Status	ETX
01	00	00	00	00	00	03

6.3 ACKNOWLEDGE_MSG (0x09)

Description

The Imager will reply to any command sent by host with this message. This message indicates that the Imager is ready to execute the command.

Data

```
typedef struct{
    unsigned char    ucMessageID;
    unsigned char    ucAcknowledgeSequenceNum;
    unsigned int     encoderRate;
    unsigned int     responseCode;
    unsigned int     encoderCount;
}SAcknowledgeMsg;
```

Data Description

ucMessageID - ID of message being acknowledged.
ucAcknowledgeSequenceNum – Sequence number of message being acknowledged.
encoderRate - Current encoder rate (ticks per second)
responseCode - 0 = No error, command accepted.
 Other = Error, command rejected. The following error codes can be expected:
 MSG_LENGTH_EXCEEDED 10
 COMMUNICATIONS_ERROR 26
encoderCount - Encoder count at time of message acknowledge.

Sample Message

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ucMessageID	ucASN	EncoderRate			
02	00	00	00	12	00	00	09	16	03	00	03	00	00

ResponseCode				encoderCount				ETX
00	00	00	00	00	00	12	33	03

6.4 STOP_PRINTING (0x16)

Description

Command sent from the host to the printer to stop current print job and put the system in Idle mode.

Data

Unsigned char [Stop_Arg](#)

Data Description

[Stop_Arg](#)

- 0 = Stop printing upon emptying buffers.
- 1 = Stop printing at end of current page.
- 2 = Stop printing after 2 second delay (Legacy)

Sample Command:

The following binary sting will be sent to printer to stop printing after 2 second delay.
Binary string: 02 00 00 00 05 00 01 16 02 03.

STX	NumBytes				Checksum	SeqNum	Msg ID	Stop_Arg	ETX
02	00	00	00	05	00	01	16	02	03

6.5 CLEANING_STATION (0x18)

Description

Cleaning station command used to command the imager to execute service station cycles. This message may originate from either the host or the imager. Use with Imager Auto Capping (AC) units only.

Note: Sending this command to a print controller will cause it to crash.

Data originating from host

unsigned char [ucCommand](#);
unsigned char [ucSubCommand](#);

Data Description

[ucCommand](#) - Cleaning system major command.
[ucSubCommand](#) - Cleaning system sub command.

Major Command		Sub-Command	
ID_CLEAN_PIC_COMMAND	0xA4	ID_PRINT	0x01
		ID_PARK	0x02
		ID_CLEAN	0x03
		ID_REMOVE_TRAY	0x04
ID_CLEAN_PIC_CONFIG	0xA5		
ID_CLEAN_PIC_MAINT	0xA6	ID_JOG_UP	0x01
		ID_JOG_DOWN	0x02
		ID_JOG_IN	0x03

ID_JOG_OUT	0x04
ID_HOME_VERT	0x06
ID_HOME_HORZ	0x07
ID_SPIT	0x08
ID_DOWN_PRINT	0x09
ID_DOWN_CAP	0x0a
ID_DOWN_SPIT	0x0b
ID_DOWN_WICK	0x0c
ID_DOWN_WIPE	0x1e
ID_OUT_REM_TRAY	0x0d
ID_OUT_WICK_WIPE	0x0e
ID_OUT_PRINT	0x0f
ID_CALIBRATE_VERTICAL	0x10
ID_RETURN_CYCLE_COUNTS	0x11

Data originating from imager

```
typedef struct {
    unsigned char    ucStatusResponseMsgID;
    unsigned char    ucStatusID;
}SServiceStationMsg;
```

Data Description

ucStatusResponseMsgID		- Status message ID
ucStatusID		- Status code.
Status Message ID		Status Code
ID_CLEAN_STATUS	0xd0	ID_CLEAN_PIC_ACK_NO_ERR 0x01
		ID_CLEAN_PIC_ACK_HORZ_ERR 0x04
		ID_CLEAN_PIC_ACK_VERT_ERR 0x05
		ID_CLEAN_PIC_ACK_TRAY_ERR 0x06
		ID_CLEAN_PIC_NO_PENS_ERR 0x07
		ID_CLEAN_PIC_INSUFF_TRAVEL 0x08
		ID_CLEAN_PIC_EXCESS_TRAVEL 0x09
		ID_CLEAN_PIC_VERT_HOME_ERR 0x0a
		ID_CLEAN_PIC_EXCESS_CAL_RT 0x0b
		ID_CLEAN_PIC_OPEN_INTERLOCK 0x0c
ID_CLEAN_STATE	0xd1	ID_IN_MOTION 0x01
		ID_IN_PRINT_POSTION 0x02
		ID_IN_PARK_POSITION 0x03

Sample Command

From Host

STX	Num of Bytes				Checksum	SeqNum	MsgID	ucCommand	ucSubCommand	ETX
02	00	00	00	06	00	00	18	A4	02	03

From Imager

STX	Num of Bytes				Checksum	SeqNum	MsgID	ucCommand	ucSubCommand	ETX
02	00	00	00	06	00	00	18	D1	02	03

6.6 PURGE_DONE (0x19)

Description

Imager notifies Host that the purge cycle completed successfully.

Data

None

Data Description

N/A

Sample Message

STX	Num of Bytes				Checksum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	19	03

6.7 SET_OUTPUT_CHANNEL (0x21)

Description

Command used to trigger outputs relays to On/Off state.

Data

```
typedef struct {
    unsigned char    ucOutputChannelNum
    unsigned char    ucValue;
}SSetOutputChannel;
```

Data Description

ucOutputChannelNum

- 0 = Relay 1
- 1 = Relay 2
- 2 – 17 = External Output. Note: An error message will be generated if no external I/O module connected or not responding.

ucValue

- 0 = Disable,
- 1 = Enable

Sample Command

STX	Num of Bytes				Check sum	Seq Num	Msg ID	ucOutputChannelNum	ucValue	ETX
02	00	00	00	06	00	00	21	00	01	03

6.8 ENABLE_PRINTING (0x25)

Description

This command enables the print trigger which will enable printing. In multiple imager system wired in daisy chain it's recommend all secondary imagers are enable first before main unit is enabled. Such sequence will synchronize all the Imagers to start printing at the same time.

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	25	03

6.9 READY_TO_PRINT (0x26)

Description

This message is sent from the Imager to the Host after a job is successfully loaded.

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	26	03

6.10 KEEP_ALIVE (0x30)

Description

Host to imager command used to test for non-responding imagers. May be used as a heart beat to test connection to imager. Imager will reply with ACKNOWLEDGE_MSG (0x09)

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	30	03

6.11 IMAGER_ERROR (0x34)

Description

Message transmitted from Imager to Host, indicates an error event has taken place. List of error codes is shown below.

Data

unsigned int [ErrorCode](#)
 unsigned char [Error Data\[\]](#)

Data Description

[ErrorCode](#) Error number, see error code table for detailed description.
[ErrorData](#) Supplemental error information, dependent on error.

ERROR CODES

Message structure:

<Error Code><Supplemental Data>
 Error Code – <unsigned int>
 Supplemental Data – <unsigned int>
 <unsigned int>
 <unsigned int>
 <unsigned int>
 < Null terminated descriptive string>

Error Code	Description	Supplemental Data	Data Description
1	Pen Data Underrun. Attempted to print but no print data was available.	<lastpagePrintedID <PrintDataIndex> <FPGAStatusReg> <PenStatusReg>	lastpagePrintedID - Id of the last page successfully printed May be 0 if no pages printed prior to underrun error.
2	Illegal command The command does not exist is not formatted correctly, is not allowed at this time(Imager is in the wrong mode), or may have failed for unknown reasons.	< CommandId> <unsigned int reserved> <unsigned int reserved> <unsigned int reserved> < descriptive string>	Command code, Optional text message
3	External I/O device failure. Cannot configure or use external IO device.	None	Optional text message
4	Pen Buffer Write Error Had a hardware failure trying to load the Image Buffer.	<unsigned int PageId> <unsigned int reserved> <unsigned int reserved> <unsigned int reserved> < descriptive string>	Id of last page printed May be 0 if no pages printed prior to Pen Buffer Write Error. Optional text message
5	Pen Buffer Out of Sync If the Imager ends a print page using a different buffer from	<unsigned int PageId> <unsigned int Pen Id> <unsigned int reserved>	Id of last page printed May be 0 if no pages printed prior to Pen Buffer Out of Sync error.

	the current buffer.	<unsigned int reserved> < descriptive string>	Id of pen which triggered error. Optional text message.
6	Raster Buffer Load Error Error loading buffer or trying to load a non-existent buffer.	none	Optional text message
7	Pen Power Fault Error This error happens if the user powered up the Imager and then disconnected the pen driver.	<unsigned int PageId> <unsigned int Pen Id> <unsigned int reserved> <unsigned int reserved> < descriptive string>	Id of last page printed May be 0 if no pages printed prior to Pen Power Fault Error error. Id of pen which triggered error. Optional text message.
8	Unused		
9	Command Execution Error	< CommandId> <unsigned int reserved> <unsigned int reserved> <unsigned int reserved> < descriptive string>	
10	Communications Buffer Overflow message to long.	None	Optional text message
11	System Overload Main Task Queue is full, messages have been dropped.	None	Optional text message
12	Print Cycle Start Failure Start Job command failed.	None	
13	Illegal Cartridge Removal	None	Not implemented
14	R10X Out of Range	None	Not implemented
15	Unassigned		
16	Illegal Cartridge Change	None	Not Implemented
17	Illegal Ink Type	None	Not Implemented
18	A/D Read Error	None	Not Implemented
19	Pen Control Failure	None	Not Implemented
20	Double TOF Detected	None	Not Implemented
21	Partial Page Skip Underrun occurred but not on all pens. Data printed on the next page will be out of synch between pens.	< lastPageID> < reserved> < FPGAStatusReg> < encoder cntr> < descriptive string>	Last page printed ID Unused FPGA Status Register Encoder Count
22	Unused.		
23	Out of Buffers Error Synchronization error. Imager received print data but all print buffers were full.	< reserved> < reserved> < buffRmn> < swTrackBfr> < descriptive string>	Number of buffer remaining Software buffer count
24	Debug Data	None	Engineering debug data. This error is generated with error code 5 and provides additional information about the error.
25	Unused.		
26	Communications Error Message header was received but the body of the message	None	

	could not be received, or got a bad ETX character,		
27	OBR Rendering Error Either a character was not rendered at all, or was only partially rendered.	None	
28	XML Translation Error An error occurred while parsing an XML file. Descriptive string will show area of the XML file with the error.	<unsigned int >, <unsigned int >, <unsigned int >, <unsigned int >, <Null terminated descriptive string (optional)>	Optional descriptive string is provided if failure occurred during parsing.

Sample Message

Below is a sample sting the imager will return for when it detects an underrun error.

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	Error Code				PageID			
02	00	00	00	19	00	00	34	00	00	00	01	00	00	00	01

Reserved				reserved				reserved				String	ETX
00	00	00	00	00	00	00	00	00	00	00	00	00	03

6.12 SAVE_CONFIGURATION (0x40)

Description

Commands imager to save current setup parameters to flash.

Parameters saved will be used for initialization upon next power-up.

Option flag allows host to specify saving image data and configuration data independently. An option to enable/disable auto run after power up is also provided.

Parameters

```

struct SaveConfig{
    bool    saveConfigData;
    bool    saveRasterBuffers;
    bool    autoRun;
}

```

Parameter Descriptions

saveConfigData - If true, all configuration initialization data will be stored to flash.

saveRasterBuffers - If true, all data currently stored in the first raster buffer will be stored in flash. Upon power-up, if more than one buffer is allocated, all buffers will be filled with the same data.

autoRun If true, the imager will read initialization parameters from flash to restore current configuration and attempt to enter print mode. If errors are detected, the imager will perform a stop print and return to idle.

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	saveConfigData	saveRasterBuffers	autoRun	ETX
02	00	00	00	07	00	00	40	01	00	00	03

6.13 SENSOR_INPUT_ENABLE (0x41)

Description

Print trigger (Top-of-Form) input enable/disable command. Could be used to temporarily disable printing without actually stopping the print job.

Data

Bool **SensorInputEnable**;

Parameters Descriptions

SensorInputEnable If true, sensor input is enabled, if false, disabled. Default value = enabled.

Sample Command

STX	Num of Bytes				Check sum	Seq Num	Msg ID	SensorInputEnable	ETX
02	00	00	00	05	00	00	41	01	03

6.14 OUTPUT_CONTROL (0x42)

Description

This message configures the output channel specified to pulse when triggered using parameters specified. The following parameters must be set at least once before output pulse utility may be used.

Data

```
typedef struct {
    unsigned char    ucChannelNumber;
    unsigned char    ucDelayMode;
    unsigned char    ucPulseMode;
    unsigned long    uliDelayDuration;
    unsigned long    uliPulseDuration;
    unsigned char    ucDelayMultiplier;
    unsigned char    ucPulseMultiplier;
    unsigned char    ucOutputAssign;
}SPulseOutput;
```

Data Description

- ucChannelNumber** - Channel number which parameters are applied to.
0 = Channel 1.
1 = Channel 2.
- ucDelayMode** - 0 = Delay parameter is in units of time, 0.01 seconds per step.
1 = Delay parameter is in units of distance. Number of sensor ticks per Sensor bin accuracy.
- ucPulseMode** - 0 = Delay parameter is in units of time, 0.01 seconds per step.
1 = Delay parameter is in units of distance. Number of sensor ticks per Sensor bin accuracy.
- uliDelayDuration** - The amount to delay after receiving command, before starting the pulse. (Valid range: 0 to 1024).
- uliPulseDuration** - The amount in time or distance to hold the pulse active. (Valid range: 0 to 1024).
- ucDelayMultiplier** - Multiplier to be applied to uliDelayDuration parameter. Valid range 1 to 16.
- ucPulseMultiplier** - Multiplier to be applied to uliPulseDuration parameter. Valid range 1 to 16.
- ucOutputAssign** - Output to be pulsed by selected timer channel. 0 and 1 are relay outputs supported by the Imager. 2 – 17 are assumed to be external outputs supported by external digital I/O unit.
Default: Timer channel 0 pulses output relay 1. Timer channel 1 pulses output relay 2. External output pulsing not supported yet.

Sample Command

STX	Num of Bytes				Check sum	Seq Num	Msg ID	ucChannelNumber	ucDelayMode	ucPulseMode
02	00	00	00	12	00	00	42	00	00	00

ucDelayDuration				ucPulseDuration				ucDelayMultiplier	ucPulseMultiplier	ucOutputAssign	ETX
00	00	00	01	00	02	00	00	01	01	00	03

6.15 INPUT_CONTROL (0x43)

Description

This message requests that the Imager notify the host if the level of the specified Input changes in the specified manner.

Note: Input 6, '0' based, is cleaning station interlock switch.

Data

```
typedef struct {
    unsigned char    ucInputNumber;
    unsigned char    ucChangeDirection;
}SMonitorInput;
```

Data Description

ucInputNumber- The 0-based Input number, ranging from 0 to 13. 0 – 5 are imager inputs. 6 is cleaning station interlock, 7 – 14 are assumed to be external inputs which may be monitored.

ucChangeDirection- 0 = Cancel notification.. (default)
 1 = Notify host if Input level changes from "High" to "Low".
 2 = Notify HOST if Input level changes from "Low" to "High".
 3 = Notify HOST if Input level has any "Change" in state.

Sample Command

Notify Host if input 0 changes from high to low.

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ucInputNumber	ucChangeDirection	ETX
02	00	00	00	06	00	00	43	00	01	03

6.16 CONFIGURATION_SAVED (0x44)

Description

Imager to Host message notifying the host that configuration data saved successfully to flash memory.

Parameters

Bool **SaveSuccessful**

Parameter Descriptions

SaveSuccessful – True, configuration data saved successfully.
 False, configuration not saved, error.

Sample Message

STX	Num of Bytes				Checksum	Seq Num	Msg ID	SaveSuccessful	ETX
02	00	00	00	05	00	00	44	01	03

6.17 INPUT_LEVEL_CHANGED (0x47)

Description

This message is the reply to the "INPUT_CONTROL" message and notifies the Host of the Input Level Change.

Data

```
typedef struct {
    unsigned          encoderTicks;
    unsigned          UTCTime;
    unsigned          Usecs;
    unsigned char     numInputsReported;
    IOStatus          IOStates[];
} SInputLevelChanged;
typedef struct {
    bool              input_Changed;
    unsigned char     input_state;
} IOStatus;
```

Data Description

encoderTicks - Encoder tick count at time of IO event.

UTCTime - UTC (Universal Time Coordinate or Greenwich Mean Time) at occurrence of IO event.

Usecs - Sub-second timer, units of 4 usec per increment. This time may be appended to UTCTime to reach sub-second accuracy.

numInputsReported - Number of inputs being reported. Dependent on type of extended I/O is installed if any.

IOStates - Array of all inputs reported with flags indicating whether a tracked state change has occurred and current state information.

Input_Changed - True – Input has changed, False – Input has not changed.

Input_state - Current input state.

Sample Command

Sample message reporting low-to-high state change for one input.

STX	Num of Bytes				Check sum	Seq Num	Msg ID	encoderTicks				UTCTime			
02	00	00	00	13	00	01	47	00	1a	25	37	00	1b	E3	12

Usecs				numInputsReported	Input_Changed	Input_state	ETX
00	00	01	12	01	01	01	03

6.18 IMAGER_LOGGING_REQUEST (0x48)

Description

The Host may request debugging trace messages be sent by the Imager to the host for logging. The Imager upon power-up will not send debug trace messages. The host must request a non-zero debug level to receive debug messages. Recommended filters for debugging are Error and Command. REGISTER_OP, REG_OP_LOW, PEN OPS, PRINT OPS, FLASH OPS should not be enabled unless instructed by inc.jet support staff.

Note: this command works with the Load Communications Config Command to define which debug messages are sent (Imager Logging Request) and where those messages go to (Load Communications Config).

Data

```
typedef struct {
    unsigned      enet_filter;
    unsigned      stdio_filter;
}DebugLogParameters;
```

Data Description

[enet_filter](#) - Filter mask applied to log messages being sent to HOST over Ethernet.
[stdio_filter](#) - Filter mask applied to log messages being sent to stdio.
 Filter parameters may be generated by bitwise 'or'ing the following mask bits. A set bit allows messages in that group to be passed.

Filter Parameter Chart

Filter Parameter	Description	Hexadecimal Code
NONE	Disables all trace messages. Note: ERROR trace messages cannot be disabled from std io port (Serial).	0x0000
COMMAND	Enables trace messages related to command and responses.	0x0001
REGISTER_OP	Enables register operation trace messages.	0x0002
REG_OP_LOW	Enables low level register trace messages. (Read/Modify/Write)	0x0004
PRINT OPS	Enables print operation trace messages.	0x0008
PEN OPS	Enables pen operation trace messages.	0x0010
FLASH OPS	Enables flash memory trace messages.	0x0020

Filter Parameter	Description	Hexadecimal Code
CLN_ST_OPS	Enables tracing cleaning station operations.	0x0040
MISC	Enables miscellaneous trace messages. Typically initialization messages or other set up operations.	0x0080
ERROR	Enables trace of error messages generated. Note: This message cannot be masked off from stdio device. (Serial)	0x8000
ALL	Enable all trace messages. Note: This setting will generate a high volume of message traffic. If messages are generated faster than they may be transmitted, some messages may be lost.	0xFFFF

Sample Command

Send Error and Command messages over Ethernet. Send just Error messages over stdio.

STX	Num of Bytes				Check sum	Seq Num	Msg ID	Enet_filter				Stdio_filter				ETX
02	00	00	00	0C	00	00	48	00	00	80	01	00	00	00	01	03

6.19 IMAGER_LOG_DATA (0x49)

Description

This command transfers Imager logging messages to the HOST.

Data

```
typedef struct {
    unsigned msgType
    unsigned stringLength;
    unsigned char stringData[];
}LogMessage;
```

Data Description

- msgType** - Filter level of message.
- stringLength** - Length of remaining data in string.
- stringData** - non-zero terminated string data(max of 256 characters).

Filter Level is one of the following:

NONE	= 0x0000
COMMAND	= 0x0001
REGISTER_OP	= 0x0002
REG_OP_LOW	= 0x0004
PRINT_OPS	= 0x0008
PEN_OPS	= 0x0010
FLASH_OPS	= 0x0020
CLN_ST_OPS	= 0x0040
MISC	= 0x0080
ERROR_F	= 0x8000
ALL	= 0xFFFF

6.20 PRINT_RECORD (0x55)

Description

This command is used to send dynamic data to be printed. The data is passed as an ASCII string and/or binary image data. Each record must end with a Form-Feed character (ASCII 12). Each field/line of the record to be printed must be separated with a Carriage-Return Line-Feed pair (ASCII 0x0D, ASCII 0x0A) and the whole record has to be terminated with a Form Feed (ASCII 0x1C).

Binary image data may be inserted into the data stream using a special flag 0xF8 identifying the data as being binary image data. Selection of image data is determined by the line number it appears in as with text data. The Host specifies which line number in the incoming data stream to expect image data. The data stream may consist of only ASCII text data, only binary image data, or text and image data mixed. The data structure defined as sPrintImage must be used to encapsulate binary image data passed in the data stream.

Data

```
typedef struct {
    unsigned char ucBufferNum;
    unsigned long int uliPageNumber
    unsigned char ucGeneratePulse_1,
    unsigned char ucGeneratePulse_2,
    unsigned long int uliByteCount;
    unsigned char[] recordData;
}SPrintRecord;

typedef struct{
    unsigned char ucImageDataFlag (0xF8)
    unsigned uiBitsWidth
    unsigned uiBytesWidth;
    unsigned uiBitsHeight
    unsigned char invertPixels;
```

```

        unsigned char    rotateImage;
        unsigned char    flipVerticalAxis;
        unsigned         uiDataLength
        unsigned char[]  ucImageData;
    }sPrintImage;

```

Data Description

ucBufferNum	- Number of buffer to be loaded (Loading buffer 0x00 will load the next available buffer)
uliPageNumber	- Reference page number. When the data will be printed the imager will send back a RASTER_BUFFER_READY_FOR_DATA with the reference page number.
ucGeneratePulse_1	- Generate a pulse on channel 1 when this page has been printed. Pulse parameters must be configured prior to this request.
ucGeneratePulse_2	- Generate a pulse on channel 2 when this page has been printed. Pulse parameters must be configured prior to this request.
uliByteCount	- Total length of following record field including carriage-returns, line-feeds and form-feed.
recordData	- Record data to be printed including carriage-returns, line-feeds and form-feed. May contain binary image data encapsulated within sPrintImage data structure.
sPrintImage	Data structure used to encapsulate binary image data.
ucImageDataFlag	Single byte flag used to identify data following to be interpreted as binary image data. (0xF8)
uiBitsWidth	Width of image data in pixels, 8 pixels per byte. Horizontal image data, pixels, are passed 8 pixels per byte. Some bits may be unused if pixel width of image is not a multiple of 8.
uiBytesWidth	Bytes per scan, may be greater than number of bytes required to accommodate uiBitsWidth if 16 or 32 bit alignment is used.
uiBitsHeight	Height of image in pixels.
invertPixels	Optional, default = 0 (<i>Not implemented</i>) 0 = No pixel inversion. (most efficient) 1 = Invert pixels, 0 – black, 1 – white.
rotateImage	Optional, default = 0 (<i>Not implemented</i>) 0 = No image rotation (most efficient) 1 = rotate image 90 degrees counter clockwise. 2 = rotate image 180 degrees 3 = rotate image 270 degrees counter clockwise
flipVerticalAxis	Optional, default = 0 (<i>Not implemented</i>) 0 = no flip (most efficient) 1 = Flip image about its vertical axis.
uiDataLength	Length of data to follow given in bytes.
ucImageData	Binary image data, horizontal pixels passed 8 pixels



per byte, last byte for each row may not be filled completely.

Sample Command

Blew is a sample print record command that with two fields containing "Hello World" string as print data.

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ucBufferNum	uliPageNumber				ucGeneratePulse_1
02	00	00	00	1C	00	01	55	00	00	00	00	01	00

ucGeneratePulse_2	uliByteCount				recordData[]													ETX
00	00	00	00	0D	48	65	6C	6C	6F	0A	0D	57	6F	72	6C	64	0C	03

6.21 START_PRINT (0x56)

Description

This command is used to instruct the Imager to start a print job that has been saved in flash memory. The job structure can also be passed down with this command by setting the filename to NULL and specify a new job to be executed immediately. Job structure is defined in Appendix B.

Data

char[14] [filename](#);
sJobRecord [Job](#);

Data Description

[Filename](#) Filename of stored job to be started. The jobs are stored using a 8.3 naming where the extension is "*.job".
If a NULL string is provided for filename, job data must follow.
Allows loading a job for testing.

[Job](#) If filename is NULL, this block will provide job configuration information. See Appendix B for structure definition.

Sample Command

This is an example of starting a print job stored flash memory as "Job1.job".

STX	Num of Bytes					Check Sum	Seq Num	Msg ID	Filename[]												ETX	
02	00	00	00	12	00	00	56	4A	6F	62	31	2E	6A	6F	62	00	00	00	00	00	00	03

6.22 START_DEFAULT_PRINT_JOB (0x57)

Description

Start default print job stored in imager flash. The default job can also be started automatically.

Data

None

Data Description

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	57	03

6.23 REQUEST_SYSTEM_READY_STATUS (0x58)

Description

Request system ready to print status. Imager will return a SYSTEM_READY_TO_PRINT_STATUS with a status indicating system readiness to print.

Data

None

Data Description

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	58	03

6.24 SYSTEM_READY_TO_PRINT_STATUS (0x59)

Description

System ready to print status. The system will indicate ready to print state when the Imager has successfully loaded a job and the TOF Sensor is enabled. This message will be generated automatically with a Not Ready State every time the Imager processes a Stop Printing command.

Note: When a job is started the imager first executes a Stop Printing sequence to stop any previous jobs that may have been running. So this message will be generated when the Imager receives STOP_PRINTING 0x16 command and when it starts a job.

Data

unsigned

readyToPrint

Data Description

readyToPrint -

0 = System not ready to print

1 = System ready to print

Sample Message

STX	Num of Bytes				Check sum	Seq Num	Msg ID	readyToPrint				ETX
02	00	00	00	08	00	00	59	00	00	00	01	03

6.25 SET_RTC (0x62)

Description

Set imager RTC to date/time specified by host.

Data

unsigned [UTCTime](#);

Data Description

[UTCTime](#) - Universal Time Coordinate or Greenwich Mean Time setting used to initialize Imager RTC. Number of seconds from midnight Jan 1 1970.

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	UTCTime				ETX
02	00	00	00	08	00	00	62	00	15	A3	5D	03

6.26 GET_RTC (0x63)

Description

Request imager to return instantaneous value of its RTC (UTC).

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	63	03

6.27 CURRENT_RTC (0x64)

Description

Instantaneous value of imager RTC (UTC).

Data

unsigned [UTCTime](#);

Data Description

[UTCTime](#) - Universal Time Coordinate or Greenwich Mean Time. Number of seconds from midnight Jan 1 1970.

Sample Message

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	UTCTime				ETX
02	00	00	00	08	00	00	64	00	1C	5A	4E	03

6.28 REPORT_IO_STATES (0x65)

Description

Host to Imager command requesting current states of inputs and outputs. Imager will return IO_STATES 0x66 in reply.

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	65	03

6.29 IO_STATES (0x66)

Description

Input and Output state report for host.

Data

unsigned char [numInputsReported](#);
unsigned char [numOutputsReported](#);
unsigned char [Inputs\[\]](#);
unsigned char [Outputs\[\]](#);

Data Description

[numInputsReported](#) - Number of inputs being reported.
[numOutputsReported](#) - Number of outputs being reported.
[Inputs](#) - Array of input states.
0,1 – State of input.
0xff – Error reading input
[Outputs](#) - Array of output states.

0,1 – State of output.
0xff – Error reading output.

Sample Message

STX	Num of Bytes				Check sum	Seq Num	Msg ID	numInputsReported	numOutputsReported	Inputs[]	Outputs[]	ETX
02	00	00	00	08	00	00	66	04	04	0A	05	03

6.30 TRIGGER_PULSE_GENERATOR (0x67)

Description

Trigger a pulse on one of the output relays. The output relay will be pulsed as configured by an OUTPUT_CONTROL command.

Data

unsigned char [generatorSelect](#)

Data Description

[generatorSelect](#) 0 - Generator 0
 1 - Generator 1

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	generatorSelect	ETX
02	00	00	00	05	00	00	67	00	03

6.31 IMAGER_WARNING (0x68)

Description

Imager to Host message notifying the host a warning condition happened. Warnings do not cause the Imager to stop printing.

Data

unsigned int [WarningCode](#)
unsigned char [Warning Data\[\]](#)

Data Description

[WarningCode](#) = See warning code table
[Warning Data](#) = Supplemental warning information, dependent on warning.



WARNING CODES

Warning Code	Description	Description
1	No Response from Pen Driver	Pen may be configured but no pen driver board is installed.
2	Illegal Cartridge ID	Can't read cartridge ID. Cartridge may not be installed.
3	Pen Out of Sync Check Disabled	This warning will be issued when a page size is smaller than the span of the pens, because multiple pages could be printed by the some pens but not others and Pen Sequence checking is no longer valid.
4	Pen failed to power up	A pen has failed to power up.
5	Pen early page termination	Indicates that the pen was still printing when end of page was detected. It must be enabled in system config.
6	Invalid Command	Command cannot be executed while a job is running.
24	Debug Data	Supplementary data following a Pen Data Under Run Error. This warning provides additional debug information about the error.

Sample Error Message

STX	Num of Bytes				Check sum	Seq Num	Msg ID	WarningCode				WarningData			
02	00	00	00	43	00	00	68	00	00	00	01	57	61	72	6E

WarningData															
69	6E	67	2C	20	50	65	6E	20	41	20	49	6E	76	61	6C

WarningData															
69	64	20	49	44	2C	20	43	61	72	74	72	69	64	67	65

WarningData															
20	6D	61	79	20	6E	6F	74	20	62	65	20	69	6E	73	74

WarningData								ETX
61	6C	6C	65	2E	0A	00	03	

6.32 DELETE_PRINT_JOB (0x73)

Description

This message is issued from the Host to the Imager. It is used to delete a print Jobs stored in flash memory.

Data

unsigned char [ucfileName\[14\]](#);

Data Description

[ucfileName\[14\]](#) The filename of the Job to delete. Filename should not contain drive letter designation. "*" is allowed when all files are to be cleared.

Sample Command

Below is a sample command to delete "Job2.job"

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ucfileName[]														ETX
02	00	00	00	12	00	01	73	4A	6F	62	32	2E	6A	6F	62	00	00	00	00	00	00	03

6.33 LIST_PRINT_JOBS (0x74)

Description

This message is issued from the Host to the Imager requesting that the Imager return the number and the names of the current stored Print Jobs. The Imager will issue a PRINT_JOBS_LIST 0x75 command in response to this request.

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	74	03

6.34 PRINT_JOBS_LIST (0x75)

Description

This message is issued from the Imager to the Host with a list of jobs that are currently stored in flash memory. It is the response to a LIST_PRINT_JOBS 0x74 request from the Host.

Data

unsigned char [ucNumPrintJobs](#)
SPrintJobID [printJobs\[\]](#)

Typedef struct {

```

    unsigned char    ucfileName[14]
    unsigned         uifileSize
    unsigned char    ucPrintJobName[64]
} SPrintJobID

```

Data Description

ucNumPrintJobs The number of current stored Internal RIP Print Jobs.
 ucfileName[14] Filename job is stored under in imager.
 uifileSize Size of file in bytes. Used for managing Imager storage.
 ucPrintJobName[64] Name of print job.

6.35 SET_DFLT_PRINT_JOB (0x76)

Description

This command is issued from the Host to the Imager and is used to set the Default Print Job, which will load automatically at power up.

Parameters

unsigned char [ucfileName \[14\]](#)

Parameter Descriptions

[ucfileName \[14\]](#) Filename of file containing the Print Job to be loaded upon power-up if imager is conditioned to autoRun. Filename should not contain drive letter designation.

Sample Command

Sample command below sets "Job1.job" as the default job.

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ucfileName														ETX
02	00	00	00	12	00	01	76	4A	6F	62	31	2E	6A	6F	62	00	00	00	00	00	00	03

6.36 GET_PEN_IDS (0x82)

Description

Host request all installed pen IDs. Imager will respond with PEN_IDS 0x83.

Data

None

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	01	82	03

6.37 PEN_IDS (0x83)

Description

Pen ID data in response to GET_PEN_IDS.

Data

```
typedef struct {
    unsigned          NumberCartridges;.
    sPenCartridgeID   Pen_ID_Data[ ];
    unsigned          good_nozzles[NumberCartridges];
    unsigned          shortedNozzles[NumberCartridges];
    unsigned          openNozzles[NumberCartridges];

} sPenCartridgeIDSet;

typedef struct {
    unsigned          Pen_ID;
    unsigned          Pen_R10X;
} sPenCartridgeID;
```

Data Description

NumberCartridges	Number of sPenCartridgeID records to follow.
Pen_ID_Data[]	One or more sPenCartridgeID records.
Pen_ID	22 bit Pen ID read from cartridge. 0x3FFFFFF indicates no cartridge.
Pen_R10X	Value of R10X resistor read from cartridge. Used to supplement Pen_ID for identification. Only valid when Pen_ID is valid.
good_nozzles	Number of nozzles which passed electrical test and are fully functional.

6.38 INITIALIZE_STATIC_RECORD_DATA (0x85)

Description

This command provides static record data to the imager, data which is necessary to initialize static record fields contained in the job. It may also contain static bitmaps enclosed in an sPrintImage structure within recordData of SPrintRecord.

Static data is data that can be initialized and maintained for a number of pieces and then changed. This can go on as many times as necessary over the course of the job.

Data

```
typedef struct {
    unsigned char      ucBufferNum
    unsigned long int  uliPageNumber
    unsigned char      ucGeneratePulse_1
```

```

        unsigned char      ucGeneratePulse_2
        unsigned long int  uliByteCount
        unsigned char[]    recordData
    }SPrintRecord;

    typedef struct{
        unsigned char      ucImageDataFlag      (0xF8)
        unsigned           uiBitsWidth
        unsigned           uiBytesWidth
        unsigned           uiBitsHeight
        unsigned char      invertPixels
        unsigned char      rotateImage
        unsigned char      flipVerticalAxis
        unsigned           uiDataLength
        unsigned char[]    ucImageData
    }sPrintImage;

```

Data Description

ucBufferNum	Number of buffer to be loaded.
uliPageNumber	Page number that this data goes to.
ucGeneratePulse_1	Generate a pulse on channel 1 when this page has been printed. Pulse parameters must be configured prior to this request.
ucGeneratePulse_2	Generate a pulse on channel 2 when this page has been printed. Pulse parameters must be configured prior to this request.
uliByteCount	Total length of following record field including carriage-returns, line-feeds and form-feed.
recordData	Record data to be printed including carriage-returns, line-feeds and form-feed. May contain binary image data encapsulated within sPrintImage data structure.
sPrintImage	Data structure used to encapsulate binary image data.
ucImageDataFlag	Single byte flag used to identify data following to be interpreted as binary image data. (0xF8)
uiBitsWidth	Width of image data in pixels, 8 pixels per byte. Horizontal image data, pixels, are passed 8 pixels per byte. Some bits may be unused if pixel width of image is not a multiple of 8.
uiBytesWidth	Bytes per scan, may be greater then number of bytes required to accommodate uiBitsWidth if 16 or 32 bit alignment is used.
uiBitsHeight	Height of image in pixels.
invertPixels	Optional, default = 0 (Not implemented) 0 = No pixel inversion. (most efficient) 1 = Invert pixels, 0 – black, 1 – white.
rotateImage	Optional, default = 0 (Not implemented) 0 = No image rotation (most efficient) 1 = rotate image 90 degrees counter clockwise. 2 = rotate image 180 degrees

flipVerticalAxis

3 = rotate image 270 degrees counter clockwise

Optional, default = 0 (Not implemented)

0 = no flip (most efficient)

1 = Flip image about its vertical axis.

uiDataLength

Length of data to follow given in bytes.

ucImageData

Binary image data, horizontal pixels passed 8 pixels per byte, last byte for each row may not be filled completely.

Sample Command

STX	Num of Bytes				Check sum	Seq Num	Msg ID	ucBufferNum	uliPageNumber			
02	00	00	00	14	00	00	85	01	00	00	00	01

ucGeneratorpulse_1	ucGeneratorPulse_2	uliByteCount				recordData[]				ETX
00	00	00	00	00	04	41	42	43	0C	03

6.39 CLEAR_PRINTING_SYSTEM (0x86)

Description

Message transmitted from Host to Imager. It commands the imager to clear all image buffers and any pieces that were queued in by the sensor to be printed. More print data must be transferred to the imager to continue printing after this command.

Data

None

Sample Command

STX	Num of Bytes				Check sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	86	03

6.40 PAGE_LOADED_NOTIFICATION (0x87)

Description

Imager generated message signaling a page load complete cycle to the host. This means that the imager received data, built a page, loaded it into the print buffer and is ready to print that page.

Data

```
typedef struct {
    unsigned encoderCount;.
    unsigned ucNumBuffersRemaining;.
    unsigned uliPageNumber;
} sPageLoaded;
```




Sample Message

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	encoderCount				ucNumBuffersRemaining				uliPageNumber				ETX
02	00	00	00	10	00	01	87	00	01	23	45	00	00	00	03	00	00	00	01	03

6.41 INK_LEVEL_ALERT (0x88)

Description

Imager generated message signaling an ink level threshold has been reached or exceeded. This message will be issued only once per job for any given pen.

Data

```
typedef struct {
    unsigned          penID
    sPenCartridgeID  PenSerialNumber
    unsigned          StatusCode
} sInkLevelAlert;

typedef struct {
    unsigned          Pen_ID
    unsigned          Pen_R10X
} sPenCartridgeID
```

Data Description

penID Pen designation, A, B, C, or D. A is fourth pen, not part of base imager

PenSerialNumber Pen identification data.

Pen_ID 22 bit Pen ID read from cartridge. 0x3FFFFFF indicates no cartridge.

Pen_R10X Value of R10X resistor read from cartridge. Used to supplement Pen_ID for identification. Only valid when Pen_ID is valid

StatusCode 0 – Low ink level warning.
1 – Ink level exhausted (estimated)

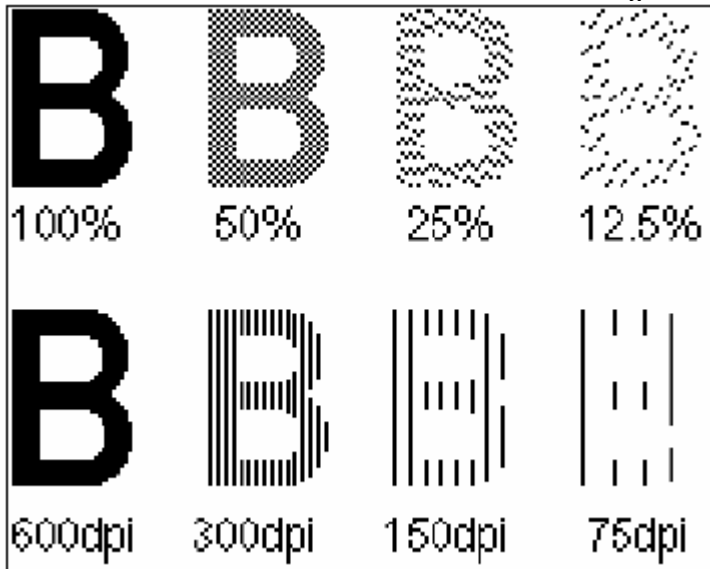
Sample Message

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	penID				Pen_ID				Pen_R10X				StatusCode	ETX
02	00	00	00	14	00	00	88	00	00	00	41	00	00	03	21	00	01	34	12	00	03

6.42 SET_CONTRAST (0x89)

Description

Host command used to set vertical contrast. The Contrast setting is used to “lighten” the print of the image. There are sixteen different density settings available. It’s recommended that Contrast is used to reduce ink usage by lightening the image instead of Print Resolution. By reducing print resolution image edge acuity is reduced as well. At lower resolution image stripping is visible. The Contrast algorithm randomizes disabled pixels creating a cleaner image with much better edge acuity. Below is an example of both methods. The top row of Bs was lightened using Contrast and the bottom row used Print Resolution. The same amount of dots are placed in both top and bottom rows. Contrast also has more level settings to enable the print to be lightened.



Data

unsigned char

[Contrast](#)

Data Description

[Contrast](#)

Range 0 to 0xF, 6.25% increments.

0 = 100% contrast (default)

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	Contrast	ETX
02	00	00	00	05	00	00	89	01	03

6.43 SET_PEN_CONFIGURATION (0x8A)

Description

Host command used to configure the pens for printing. Host needs to send one command per installed pen.

Data

```

unsigned char      PenID; // A,B,C, or D
sPenConfigData    PenConfigData;

typedef struct {
    bool            trackInk;
    unsigned        WarningThreshold_H;
    unsigned        WarningThreshold_L;
    unsigned        ErrorThreshold_H;
    unsigned        ErrorThreshold_L;
    bool            autoReset;
    bool            Autocal;
    bool            PowerEnable;
    bool            PurgeEnable;
    bool            SparseSpitEnable;
    bool            WarmingPulseEnable;
    bool            PixelCountEnable;
    unsigned        RegulatorTrim;
    unsigned        TemperatureA2DThreshold;
    unsigned        FirePulseLength;
    unsigned        WarmingPulseLength;
    unsigned        SubFireInterval;
    unsigned        EncoderSubTick;
    unsigned        SensorTriggerOffset;
    unsigned        PrintMode;
    unsigned long    columnOffset;
    bool            PrintDirection;
    unsigned        PowerUpDelay;
    bool            PenVoltageReprogram;
} sPenConfigData;

```

Data Description

PenID	ID of physical pen to configure. Must be sent as ASCII 'A', 'B', 'C' or 'D' (0x41, 0x42, 0x43, 0x44).
trackInk	Ink tracking flag. False = no Ink tracking. True = Track ink usage.
WarningThreshold_H	Upper 32 bits of 64 bit ink usage warning threshold. Number of pixels the pen will print per cartridge until it generates an ink low warning message. 42ml cartridge contains 1.4 billion pixels based on 30 pl pixel volume.
WarningThreshold_L ErrorThreshold_H	Lower 32 bits of 64 bit ink usage warning threshold. Upper 32 bits of 64 bit ink usage out of ink threshold. Number of pixels the pen will print per cartridge until it generates an ink empty error message. 42ml cartridge contains 1.4 billion pixels based on 30 pl pixel volume.

ErrorThreshold_L autoReset	Lower 32 bits of 64 bit ink usage out of ink threshold. Automatically reset ink usage and serial number settings upon detection of a new pen. False = Do not reset data upon detecting new pen serial number. True = Automatically reset ink usage and serial number data upon detecting a new pen serial number.
Autocal	Automatically optimize pen firing voltage upon detecting new pen serial number. NOT IMPLEMENTED.
PowerEnable	Pen Enable False = Pen not installed or unused. True = Pen installed, power up for printing.
PurgeEnable	Pen purge enable. False = Pen does not purge. True = Pen purges during purge cycles.
SparseSpitEnable	Pen sparse spit enable. False = Sparse spit disabled. True = Sparse spit enabled.
WarmingPulseEnable	Pen warming pulse enable. False = Warming pulse disabled. True = Warming pulse enabled.
PixelCountEnable	Enable hardware pixel counting. False = Disable; True = Enable.
RegulatorTrim	Pen cartridge regulator trim setting or Pen Firing Voltage set in 0.2V step increment. Range: 0 to 0x23. (5.0VDC to +12.0VDC) Recommended default setting of (0x00 0x00 0x00 0x1C) or 10.6V
TemperatureA2DThreshold	Pulse warming temperature expressed in °C. The value is set 0.2°C per increment and should be a multiple of 5. Range: 0 to 0x1FF. Recommended default setting is 255 or (0x00 0x00 0x00 0xE1).
FirePulseLength	Firing pulse length set in 46.875 nsec per increment Range: 0 to 0x7F. Recommended default setting is (0x00 0x00 0x00 0x29) or 2.0µs.
WarmingPulseLength	Warming pulse length set in 46.875 nsec per increment Range: 0 to 0x1F. Recommended default setting is (0x00 0x00 0x00 0x10) or 780 nsec.
SubFireInterval	Data load delay set in 46.875 nsec per increment. Must be equal or greater than (FirePulseLength – 29). Range: 0 to 0xFF. Recommended default setting is 11 or (0x00 0x00 0x00 0x0B)
EncoderSubTick	Legacy, not used.

SensorTriggerOffset

Delay from sensor trigger to initial firing of pen. Set as number of encoder ticks. Accuracy is dictated by

[ResolutionSelect](#) set in

IMAGER_CONFIGURATION_TIPS(0x8F)

Range: 0 to 0xFFFF

Example: If the pen was 1.5" from the sensor and sensor accuracy was set to 600. The offset would be 1.5" x 600 = 900 or (0x00 0x00 0x03 0x84)

PrintDirection

Determines the leading and trailing columns for printing. Leading column is determined based on pen orientation.

0 – Forward.

1 – Reverse.

columnOffset

Offset added to column addresses to index into the image for this pen.

ImageDirection

A Boolean flag used to indicate which way to process image buffer. Reverse is set when pen is to be configured to print in reverse direction so the image will be printed back to front.

0 = Normal

1 = Reverse

PowerUpDelay

Delay time from time pen power is applied to time of first command issued to pen. Allows for user power supply to stabilize after powering pen. Should be set to 1.

Range: 0 to 20.

Units: 50 milliseconds per increment.

PenVoltageReprogram

Reprogram pen voltage at end of each pass through the stitching table. Used to recover pens that drop print in the middle of production due to intermittent connection.

0 = No reprogramming. (Default)

1 = Reprogram.

Sample Command

STX	Num of Bytes				Checksum	Seq Num	Msg ID	PenID	trackInk	WarningThreshold_H				WarningThreshold_L			
02	00	00	00	47	00	00	8A	43	00	00	00	00	00	10	00	00	00

ErrorThreshold_H				ErrorThreshold_L				autoReset	Autocal	PowerEnable	PurgeEnable	SparseSpitEnable			
00	00	00	00	00	00	10	00	01	00	01	00	00			

WarmingPulseEnable				PixelCountEnable				RegulatorTrim				TemperatureA2DThreshold				FirePulseLength			
01				00				00	00	00	14	00	00	00	BE	00	00	00	60

WarmingPulseLength				SubFireInterval				EncoderSubTick				SensorTriggerOffset			
00	00	00	30	00	00	00	37	00	00	00	00	00	00	25	00

PrintMode				columnOffset				PrintDirection	PowerUpDelay				PenVoltageReprogram	ETX
00	00	00	00	00	00	00	00	00	00	00	00	01	00	03

6.44 REQUEST_STATUS (0x8D)

Description

Request for imager status. Imager returns IMAGER_STATUS (0x8E).

Data

None

Sample Command

STX	Num of Bytes				Check sum	Seq Num	Msg ID	ETX
02	00	00	00	04	00	00	8D	03

6.45 IMAGER_STATUS (0x8E)

Description

Current Imager Status;

Data

```
typedef struct {
    unsigned long          schemaRevisionNumber;
    unsigned char[64]      Job Name;
    unsigned long          JobStartTime;
    unsigned long          CurrentTime;
    unsigned char          ImagerState;
    unsigned               PagesPrinted;
    unsigned               EncoderRate;
    unsigned               ElapsedTimeSincePurge;
    unsigned               Imager_Warning_Code;
    unsigned               Imager_Error_Code;
    unsigned               Imager_Spare_1;
    unsigned               Imager_Spare_2;
    unsigned               Imager_Spare_3;
    unsigned               Imager_Spare_4;

    unsigned char          Pen_0_EnableStatus;
    unsigned char          Pen_0_Ink_Level_Low;
    unsigned char          Pen_0_Ink_Level_Error;
    unsigned char          Pen_0_Test_Status;
    unsigned               Pen_0_PixelsPrinted_L;
    unsigned               Pen_0_PixelsPrinted_H;
    unsigned               Pen_0_Good_Nozzles;
```

unsigned char	Pen_0_Driver_Not_Responding;
unsigned	Pen_0_Spare_1;
unsigned	Pen_0_Spare_2;
unsigned	Pen_0_Spare_3;
unsigned	Pen_0_Spare_4;
unsigned char	Pen_1_EnableStatus;
unsigned char	Pen_1_Ink_Level_Low;
unsigned char	Pen_1_Ink_Level_Error;
unsigned char	Pen_1_Test_Status;
unsigned	Pen_1_PixelsPrinted_L;
unsigned	Pen_1_PixelsPrinted_H;
unsigned	Pen_1_Good_Nozzles;
unsigned char	Pen_1_Driver_Not_Responding;
unsigned	Pen_1_Spare_1;
unsigned	Pen_1_Spare_2;
unsigned	Pen_1_Spare_3;
unsigned	Pen_1_Spare_4;
unsigned char	Pen_2_EnableStatus;
unsigned char	Pen_2_Ink_Level_Low;
unsigned char	Pen_2_Ink_Level_Error;
unsigned char	Pen_2_Test_Status;
unsigned	Pen_2_PixelsPrinted_L;
unsigned	Pen_2_PixelsPrinted_H;
unsigned	Pen_2_Good_Nozzles;
unsigned char	Pen_2_Driver_Not_Responding;
unsigned	Pen_2_Spare_1;
unsigned	Pen_2_Spare_2;
unsigned	Pen_2_Spare_3;
unsigned	Pen_2_Spare_4;
unsigned char	Pen_3_EnableStatus;
unsigned char	Pen_3_Ink_Level_Low;
unsigned char	Pen_3_Ink_Level_Error;
unsigned char	Pen_3_Test_Status;
unsigned	Pen_3_PixelsPrinted_L;
unsigned	Pen_3_PixelsPrinted_H;
unsigned	Pen_3_Good_Nozzles;
unsigned char	Pen_3_Driver_Not_Responding;
unsigned	Pen_3_Spare_1;
unsigned	Pen_3_Spare_2;
unsigned	Pen_3_Spare_3;
unsigned	Pen_3_Spare_4;
unsigned char	numberInputsReported;
unsigned	inputs;

```

        unsigned char
        unsigned
        outState
        numberOutputsReported;
        outputs;
        stackLightOuts[4];

    } sImagerStatus;

```

Data Description

schemaRevisionNumber	Revision number of this data structure. = 1;
Job Name	Job name of job currently running, taken from job name field supplied with job.
JobStartTime	Job start time. Expressed in UTC format, seconds since January 1, 1970.
CurrentTime	Current time. Expressed in UTC format, seconds since January 1, 1970.
ImagerState	Current imager state. 0 = Idle 1 = Ready to print 2 = Printing 3 = Error
PagesPrinted	Number of pages printed for current job.
EncoderRate	Current encoder rate (ticks per second).
ElapsedTimeSincePurge	Number of seconds since last purge. 0 if no purge since power up of imager.
Imager_Warning_Code	Warning code
Imager_Error_Code	Error code
Pen_0_EnableStatus	Pen enable status. 0 = disabled, 1 = enabled
Pen_0_Ink_Level_Low	0 = Ink level not low. 1 = Ink Level Low
Pen_0_Ink_Level_Error	0 = No ink level error. 1 = Estimated ink level is empty
Pen_0_Test_Status	0 = No error 1 = Pen failed opens/shorts test.
Pen_0_PixelsPrinted_L	- 64 bit pixels printed to date - Low bytes
Pen_0_PixelsPrinted_H	- Upper bytes
Pen_0_Good_Nozzles	- Number of useable nozzles.
Pen_0_Driver_Not_Responding	Pen Driver Status 1 = Driver not responding, 0 = Driver OK.
Pen_0_Spare_1	- Unused variable.
Pen_0_Spare_2	- Unused variable.
Pen_0_Spare_3	- Unused variable.
Pen_0_Spare_4	- Unused variable.

Pen_1_EnableStatus	- Pen enable status. 0 = disabled, 1 = enabled
Pen_1_Ink_Level_Low	0 = Ink level not low. 1 = Ink Level Low
Pen_1_Ink_Level_Error	0 = No ink level error. 1 = Estimated ink level is empty
Pen_1_Test_Status	0 = No error 1 = Pen failed opens/shorts test.
Pen_1_PixelsPrinted_L	- 64 bit pixels printed to date - Low bytes
Pen_1_PixelsPrinted_H	- Upper bytes
Pen_1_Good_Nozzles	- Number of useable nozzles.
Pen_1_Driver_Not_Responding	Pen Driver Status 1 = Driver not responding, 0 = Driver OK.
Pen_1_Spare_1	- Unused variable.
Pen_1_Spare_2	- Unused variable.
Pen_1_Spare_3	- Unused variable.
Pen_1_Spare_4	- Unused variable.
Pen_2_EnableStatus	- Pen enable status. 0 = disabled, 1 = enabled
Pen_2_Ink_Level_Low	0 = Ink level not low. 1 = Ink Level Low
Pen_2_Ink_Level_Error	0 = No ink level error. 1 = Estimated ink level = 0
Pen_2_Test_Status	0 = No error 1 = Pen failed opens/shorts test.
Pen_2_PixelsPrinted_L	- 64 bit pixels printed to date - Low bytes
Pen_2_PixelsPrinted_H	- Upper bytes
Pen_2_Good_Nozzles	- Number of useable nozzles.
Pen_2_Driver_Not_Responding	Pen Driver Status 1 = Driver not responding, 0 = Driver OK.
Pen_2_Spare_1	- Unused variable.
Pen_2_Spare_2	- Unused variable.
Pen_2_Spare_3	- Unused variable.
Pen_2_Spare_4	- Unused variable.
Pen_3_EnableStatus	- Pen enable status. 0 = disabled, 1 = enabled
Pen_3_Ink_Level_Low	0 = Ink level not low.

Pen_3_Ink_Level_Error	1 = Ink Level Low 0 = No ink level error.
Pen_3_Test_Status	1 = Estimated ink level = 0 0 = No error
Pen_3_PixelsPrinted_L	1 = Pen failed opens/shorts test.
Pen_3_PixelsPrinted_H	- 64 bit pixels printed to date
Pen_3_Good_Nozzles	- Low bytes
Pen_3_Driver_Not_Responding	- Upper bytes
	- Number of useable nozzles.
	Pen Driver Status
	1 = Driver not responding,
	0 = Driver OK.
Pen_3_Spare_1	- Unused variable.
Pen_3_Spare_2	- Unused variable.
Pen_3_Spare_3	- Unused variable.
Pen_3_Spare_4	- Unused variable.
numberInputsReported	- Number of input states reported.
Inputs	- Input states represented by bits starting with bit 0. So 3 states, all high, would be 0xE0000000.
numberOutputsReported	- Number of output states reported
Outputs	- Output states represented by bits starting with bit 0. So 5 states, all high, would be 0xF8000000.
stackLightOuts[4]	Current state of 3 stacklight lights and alarm. stackLightOuts[0] = Green light stackLightOuts[1] = Yellow light stackLightOuts[2] = Red light stackLightOuts[3] = Alarm States 0 = Off 1 = On 2 = Flash

Sample Message

STX	Num of Bytes				Check sum	Seq Num	Msg ID	SchemaRevisionNumber				JobName[]			
02	00	00	01	07	00	00	8E	00	00	00	01	45	78	61	6D
JobName[]															
70	6C	65	20	4A	6F	62	00	00	00	00	00	00	00	00	00
JobName[]															
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
JobName[]												JobStartTime			
00	00	00	00	00	00	00	00	00	00	00	00	00	00	10	00



CurrentTime				ImagerState		PagesPrinted				EncoderRate				ElapsedTimeSincePurge		
00	00	30	00	02		00	00	10	00	00	00	01	00	00	00	00

ElapsedTimeSincePurge				Imager_Warning_Code				Imager_Error_Code				Imager_Spare_1				Imager_Spare_2			
00				00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Imager_Spare_2				Imager_Spare_3				Imager_Spare_4				Pen_0_Enable_Status				Pen_0_Ink_Level_Low			
00				00	00	00	00	00	00	00	00	01				00			

Pen_0_Ink_Level_Error				Pen_0_Test_Status				Pen_0_Pixels_Printed_L			
00				00				00	10	20	00

Pen_0_Pixels_Printed_H				Pen_0_Good_Nozzles				Pen_0_Driver_Not_Responding				Pen_0_Spare_1				Pen_0_Spare_2			
00	20	28	12	00	00	01	2C	00				00	00	00	00	00	00	00	00

Pen_0_Spare_3				Pen_0_Spare_4				Pen_1_Enable_Status				Pen_1_Ink_Level_Low				Pen_1_Ink_Level_Error			
00	00	00	00	00	00	00	00	01				00				00			

Pen_1_Test_Status				Pen_1_Pixels_Printed_L				Pen_1_Pixels_Printed_H				Pen_1_Good_Nozzles				Pen_1_Driver_Not_Responding			
00				00	10	20	00	00	20	29	34	00	00	01	2C	00			

Pen_1_Spare_1				Pen_1_Spare_2				Pen_1_Spare_3				Pen_1_Spare_4			
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Pen_2_Enable_Status				Pen_2_Ink_Level_Low				Pen_2_Ink_Level_Error				Pen_2_Test_Status				Pen_2_Pixels_Printed_L			
01				00				00				00				00			

Pen_2_Pixels_Printed_L				Pen_2_Pixels_Printed_H				Pen_2_Good_Nozzles				Pen_2_Driver_Not_Responding				Pen_2_Spare_1			
00	02	28	56	00	01	22	45	00	00	01	2C	00				00	00	00	00

Pen_2_Spare_2				Pen_2_Spare_3				Pen_2_Spare_4				Pen_3_Enable_Status				Pen_3_Ink_Level_Low				Pen_3_Ink_Level_Error			
00	00	00	00	00	00	00	00	00	00	00	00	01				00				00			

Pen_3_Test_Status				Pen_3_Pixels_Printed_L				Pen_3_Pixels_Printed_H				Pen_3_Good_Nozzles				Pen_3_Driver_Not_Responding			
00				00	04	18	29	00	00	40	89	00	00	01	2C	00			

Pen_3_Spare_1				Pen_3_Spare_2				Pen_3_Spare_3				Pen_3_Spare_4				numberInputsReported				Inputs	
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	02				00	00

Inputs		numberOutputsReported				outputs				stackLightOuts[4]				ETX	
00	01	03				00	00	00	06	01	00	00	00	03	

6.46 IMAGER_CONFIGURATION_TIPS (0x8F)

Description

Configuration data transmitted from Host to Imager when Host configures an imager for printing. May be updated by Host at any time.

Configuration data transmitted from Imager to Host when Host requests current imager configuration.

Data

```
struct imagerConfig_TIPS{
    EncoderConfig    EncoderCfg;
    SensorConfig     SensorCfg;
    IOConfig         IOCfg;
    bool             EarlyPageTermination;
    bool             synchClockToHost;
    unsigned char    systemType;
    unsigned char    timeOffset;
    unsigned char    encoderEnableInput;
    unsigned char    sensorEnableInput;
    unsigned char    encoderResampleModeSelect;
    unsigned char    encoderResampleMode;
    InputEventConfig InputEvtCfg[15];
    OutputEventConfig OutputEvtCfg[18];
    SPulseOutputCfg  OutPulseCfg[2];
}
```

Sub-struct defines

```
struct EncoderConfig{
    unsigned char    modeSelect;
    unsigned short   EncoderDPI;
    unsigned char    Encoder_A_InputSelect;
    unsigned char    Encoder_B_InputSelect;
    unsigned         EncoderFreeRunDivider;
}

struct SensorConfig{
    bool             InputInvert;
    unsigned char    ResolutionSelect;
    bool             leadingEdgeSelect;
    unsigned char    SensorInputSelect;
    unsigned         sensorFreeRunLoopCount;
}

struct IOConfig{
    unsigned char    Input_0_DebounceSelect;
    unsigned char    Input_1_DebounceSelect;
    unsigned char    Input_2_DebounceSelect;
    unsigned char    Input_3_DebounceSelect;
    unsigned char    Input_4_DebounceSelect;
    unsigned char    Input_5_DebounceSelect;
```

```

bool      Output_1_RelayEnable;
bool      Output_2_RelayEnable;
unsigned char sparseSpitIncrement;
unsigned char lockout;
unsigned char hardwareDebugBits;
unsigned char externalIO;
}
typedef struct {
    unsigned char low_high_action[4];
    unsigned char high_low_action[4];
} InputEventConfig;

typedef struct {
    unsigned char output_event;
    unsigned char polarity;
    unsigned char pulse;
} OutputEventConfig;

typedef struct {
    unsigned char ucDelayMode;
    unsigned char ucPulseMode;
    unsigned long uliDelayDuration;
    unsigned long uliPulseDuration;
    unsigned char ucDelayMultiplier;
    unsigned char ucPulseMultiplier;
    unsigned char ucOutputAssign;
} SPulseOutputCfg;

```

Data Description

modeSelect

The encoder can be configured in 1 or 2 channel mode. In 2 channel mode Quadrature will be used to determine direction.

0 – 1 channel mode

1 – 2 channel mode.

EncoderDPI

Output Resolution. This variable is used to calculate Distance, in pixels, from one pen column to the other. This should always be set to 300.

Encoder_A_InputSelect

Input Channel A of the encoder is tied to. Setting this to input 7 sets the encoder to internally generated encoder or free run.

0 – Input 0

1 – Input 1

2 – Input 2

3 – Input 3

4 – Input 4

5 – Input 5

6 – Manual

Encoder_B_InputSelect EncoderFreeRunDivider	<p>7 – Free Run</p> <p>Same as above, but for B input.</p> <p>In Free Run Mode Counter Mode, the entry * 500 nsec (4MHz clock) sets the auto generated encoder rate.</p> <p>In Encoder Resample Mode, the entry is used as a Multiplier applied to the encoder rising edge time period to create a new encoder rate</p> <p>3 MSB bits are used as whole numbers(Range = 0 - 7). Maximum Multiplier value = 7.99988</p> <p>13 LSB bits are used as decimal numbers. (Value / 8192) Minimum Multiplier value = 0.000122</p> <p>In Sensor/Sensor Auto Encoder Generation Mode, the entry is used as a Multiplier applied to the sensor/sensor time / 256 to create the encoder rate.</p>
InputInvert	<p>0 – Normal</p> <p>1 – Inverts the Sensor Signal.</p>
ResolutionSelect	<p>Sensor Accuracy, number of encoder tick per sensor bin. Should always be set to 600.</p> <p>1 = 600 dpi accuracy 2 = 300 dpi accuracy 3 = 200 dpi accuracy 4 = 150 dpi accuracy 5 = 120 dpi accuracy 6 = 100 dpi accuracy 7 = 75 dpi accuracy</p>
leadingEdgeSelect	<p>0 – Normal (Light On)</p> <p>1 – Inverted (Dark On)</p>
SensorInputSelect	<p>Specifies the input the sensor is connected to. Setting this to input 7 puts the sensor in fixed repeat mode. Distance of the repeat is adjusted by sensorFreeRunLoopCount.</p> <p>0 – Input 0 1 – Input 1 2 – Input 2 3 – Input 3 4 – Input 4 5 – Input 5 6 – Manual 7 – Loop Mode</p>

<code>sensorFreeRunLoopCount</code>	Used to adjust the repeat length of the sensor when the sensor is set to fixed repeat mode. The value is in Encoder Ticks. The units depend on the sensor accuracy (ResolutionSelect). Resolution of 300DPI yields 300 ticks/inch, a 600DPI setting yields 600 ticks/inch.
<code>Input_0_DebounceSelect</code>	0 = Passthrough 1 = 32 uS Debounce (31.25 KHz) 2 = 1 mS Debounce (1 KHz) 3 = 64 mS Debounce (15 Hz)
<code>Input_1_DebounceSelect</code>	same as above
<code>Input_2_DebounceSelect</code>	same as above
<code>Input_3_DebounceSelect</code>	same as above
<code>Input_4_DebounceSelect</code>	same as above
<code>Input_5_DebounceSelect</code>	same as above
<code>Output_1_RelayEnable</code>	0 – Disable 1 - Enable
<code>Output_2_RelayEnable</code>	0 – Disable 1 - Enable
<code>sparseSpitIncrement</code>	Sparse Spit fires each nozzle in turn. This register defines the distance, in 16 inch increments, between firings of a particular nozzle.
<code>lockout</code>	Unused.
<code>hardwareDebugBits</code>	Engineering use only. (Should always be set to 0)
<code>externalIO</code>	Defines if any external I/O boards are connected to the second serial port. 0 = No external I/O module. 1 = ONTRAK ADR2205 I/O 2 = Pencom 2 Output Only
<code>EarlyPageTermination</code>	If enabled the hardware will keep track of page size using the sensor state and encoder counts. If it detects a page that is shorter than predefined length it will terminate the printing of that page at the recorded length. 0 = Disabled (Normal Operation) 1 = Enabled.
<code>synchClockToHost</code>	0 = No autosynchronization of time clock. 1 = Use NTP protocol to synch time clock to host system.(when autosynching make sure the imager is connected to a device that supports NTP protocol imager will continually ping the network to update it's clock.)
<code>systemType</code>	Defines the system type the jet.engine board is set to. (Make sure not to set jet.engine AC to print controller is it may damage the motor controls.)

- 0 = Undefined
- 1 = Print Controller
- 2 = jet.engine MC
- 3 = jet.engine AC

timeOffset	Time offset from GMT in hours (0 – 23)
encoderEnableInput	Encoder can be enable/disable using another input. <ul style="list-style-type: none"> 0 – Always Enabled 1 – Enabled by input 0 2 – Enabled by input 1 3 – Enabled by input 2 4 – Enabled by input 3 5 – Enabled by input 4 6 – Enabled by input 5 7 – Disabled;
sensorEnableInput	Input used to enable sensor. <ul style="list-style-type: none"> 0 – Always Enabled 1 – Enabled by input 0 2 – Enabled by input 1 3 – Enabled by input 2 4 – Enabled by input 3 5 – Enabled by input 4 6 – Enabled by input 5 7 – Disabled;
encoderResampleModeSelect	The Imager requires an 300 tick per inch encoder rate in order to print correctly. If the encoder used on the line is at different resolution the internal hardware can resample it to 300 ticks per inch. A multiplier to adjust the encoder rate is defined by EncoderFreeRunDivider when resample mode is defined. <ul style="list-style-type: none"> 0 - Disable encoder resample mode. 1 - Enable encoder resample mode.
encoderResampleMode	There are two resampling modes. One resamples the encoder rate the other uses two sensors to determine the speed of conveyor <ul style="list-style-type: none"> 0 - Encoder Resampling. 1 - Sensor/Sensor to encoder resampling.
low_high_action[4]	Array of actions associated with input event, input going from low to high. Supports up to 4 actions. Actions will be executed in the order they are listed in the array. <ul style="list-style-type: none"> 0 – no action. 1 – Trigger on-board rip. Used for OnBoard RIP

	only, pages which require no external data but contain either Date/Time fields or a counter. A page will be processed upon detecting a trigger from the selected input.
	2 – Send cleaning station to park.
	3 – Send cleaning station to print.
	4 – Start cleaning cycle.
	5 – Start default job
	6 – Stop job.
	7 – Spit
	8 – Report Input change event
	9 – Clear Error and Warning Flags
high_low_action[4]	Array of actions associated with input event, input going from high to low. Actions are the same as defined for low_high_action[4].
output_event	0 – No action. 1 – Active on error 2 – Active on ready to print + TOF sensor enabled. 3 – Active when cleaning station is in print position. 4 – Active when cleaning station is in park. 5 – Active on ink level warning. 6 – Active on ink level error. 7 – General warning condition from imager. 8 – Page available for printing and printing has been enabled. Must be configured to Pulse. 9 - Active on ready to print + TOF sensor enabled. + cleaning station is in print position. 10 – Activate upon End of Page Print event. 11 - Activate upon Start of Page Print event
polarity	Not implemented, for future use
pulse	0 – Static level 1 – Pulse output
ucDelayMode	0 - Pulse delay in units of time(seconds).
	1 - Pulse delay in units of distance (encoder ticks).
ucPulseMode	0 - Pulse duration in units of time(seconds)
	1 - Pulse duration in units of distance(encoder ticks).
uliDelayDuration	The amount to delay after receiving command, before starting the pulse. (Valid range: 0 to 1024). In time delay 0.01 sec per step.
	In distance delay 1 encode tick per step.
uliPulseDuration	The amount to time or distance to hold pulse active. (Valid range: 0 to 1024).

	In time delay 0.01 sec per step.
	In distance delay 1 encode tick per step.
ucDelayMultiplier	Multiplier to be applied to uliDelayDuration parameter. (Valid range 1 to 16).
ucPulseMultiplier	Multiplier to be applied to uliPulseDuration parameter. (Valid range 1 to 16).
ucOutputAssign	Output to be pulsed by selected timer channel. 0 and 1 are relay outputs supported by imager. 2 – 17 are assumed to be external outputs supported by external digital I/O unit. Default: Timer channel 0 pulses output relay 1. Timer channel 1 pulses output relay 2. External output pulsing not supported yet.

6.47 SHIFT_PEN_OFFSET (0x92)

Description

This command changes the distance from product detect sensor to the pen. It's distance the pen will wait to fire from the position where the product is detected and where the pen starts printing. Distance adjustments are made in encoder "ticks" which could be configured to 600, 300, 200, 150, 120, 100, 75 dpi accuracy using IMAGER_CONFIGURATION_TIPS 0x8F(encoder signal has to always be 300dpi). All printer system can support up to 4 pens that are usually staggered. In order for all the pens to print one image seamlessly pen offsets need to be configured properly. This command needs to be sent once for every enabled pen.

Data

```

Typedef struct {
    unsigned char    PenStallID;
    long int         PenSensorOffset\_Shift;
} sPenOffset;

```

Data Description

PenStallID	Pen Stall ID, 0, 1, 2, or 3. (0=A, 1=B, 2=C, 3=D)
PenSensorOffset_Shift	Value to be applied to current 16 bit pen sensor offset. Distance adjustment is made in encoder ticks. <i>Example:</i> If encoder resolution accuracy is set to 600dpi and the pen is 2.5" away from the product sensor PenSensorOffset_Shift = 1500 (0x00 0x00 0x05 0xDC).

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	PenStallID	PenSensorOffset_Shift				ETX
02	00	00	00	09	00	00	92	02	00	00	05	DC	03

6.48 STACK_LIGHT_CONTROL (0x93)

Description

Stack light control command. Host may request one of three states be set for any of 4 outputs. Allowable states are FLASH, SOLID or OFF. Output control is shared with firmware.

Data

```
Typedef struct {
    unsigned char    outputSelect;
    unsigned char    requestedState;
} sStackControl;
```

Data Description

outputSelect Stacklight output select.
 0 = GREEN
 1 = YELLOW
 2 = RED
 3 = ALARM.

requestedState Requested output state.
 0 = OFF
 1 = SOLID
 2 = FLASH

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	outputSelect	requestedState	ETX
02	00	00	00	06	00	00	93	00	01	03

6.49 EXECUTE_PURGE (0x94)

Description

Pen spit command allows host to purge all installed pens upon request. User must wait for purge completion status message, PURGE_DONE (0x19), before issuing new commands.

Data

```
Typedef struct {
    unsigned long int    blackLinesToSpit;
    unsigned long int    whiteLinesToSpit;
    unsigned long int    spitRepeatCount;
} sPenSpit;
```

Data Description

blackLinesToSpit The number of black lines to spit when performing a spit operation.

whiteLinesToSpit The number of white lines to spit when performing a spit operation.

spitRepeatCount

Number of times the purge sequence should be repeated during a spit operation.

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	blackLinesToSpit				whiteLinesToSpit				spitRepeatCount				ETX
02	00	00	00	10	00	01	94	00	00	00	10	00	00	00	10	00	00	00	1A	03

6.50 SKIP_PAGE (0x95)

Description

Host may request a page to be skipped during printing. The request must contain a valid page number and be received before the first pen begins to print the requested page, else an error will be returned and the page will not be skipped.

Data

```
typedef struct {
    unsigned char    type;
    unsigned int     pageNumber;
    unsigned char    pen_A;
    unsigned char    pen_B;
    unsigned char    pen_C;
    unsigned char    pen_D;
} sPageSkip
```

Data Description

Type

0 = pageNumber interpreted to be page number.
1 = pageNumber interpreted to be buffer number, 1's based.

pageNumber

Page number assigned by the host to be skipped.

Pen_A

0 = print, 1 = skip

Pen_B

0 = print, 1 = skip

Pen_C

0 = print, 1 = skip

Pen_D

0 = print, 1 = skip

Sample Command

STX	Num of Bytes				Check Sum	Seq Num	Msg ID	type	pageNumber				Pen_A	Pen_B	Pen_C	Pen_D	ETX
02	00	00	00	0D	00	01	95	00	00	00	10	00	01	01	01	01	03

6.51 GENERATE_STITCH_TABLES (0x99)

Description

Generate stitch tables based on the parameters given. Stitching tables assigns image pixels to pen nozzles with the goal of printing seamlessly across all pens. Reference Appendix A for detailed explanation of the command.

Data

```
typedef struct StitchTableParams {
    sStitchTableVariables PenA;
    sStitchTableVariables PenB;
    sStitchTableVariables PenC;
    sStitchTableVariables PenD;
};
typedef struct sStitchTableVariables {
    unsigned char    ucPD;
    unsigned int     uiLJTD;
    unsigned int     uiTJTD;
    unsigned int     uiRES;
    unsigned int     uiPFIP;
    unsigned char    ucPFISP;
    unsigned char    ucCOL;
    unsigned char    ucCS;
    unsigned char    ucPO;
    unsigned char    ucFLAGS;
};
```

Data Description

For a more detailed description of the data, see Appendix A.

ucPD	Print Direction (Binary 0 = Forward, 1 = Reverse)
uiLJTD	Leading Jets To Disable (Integer 0 – 300)
uiTJTD	Trailing Jets To Disable (Integer 0 – 300)
uiRES	Vertical Image Resolution (Integer 600, 300)
uiPFIP	Pen First Image Pixel (Integer 0 - 65535)
ucPFISP	Pen First Image Sub Pixel (Integer 0-7)
ucCOL	Number of Columns to use (1 = Disable Odd Jets, 2)
ucCS	Column Select (1 = Normal, 0 = Reverse) – should always be set to Normal.
ucPO	Pixel Ordering (0 = Normal, 1 = Reverse)
ucFLAGS	Bit 0 – Double Speed Enable (0 = Disabled, 1 = Enabled) Bits 1 – 7 Unused

6.52 LOAD_COMMUNICATIONS_CONFIG (0x9A)

Description

Command used to configure the communication protocol on the serial and Ethernet ports. This command is used to specify settings like communication buffer size, protocol assignments, and unsolicited message destinations.

Parameters

```

unsigned char      saveToFlash;
portCfg           configData;

=====

typedef struct{
    unsigned          schema;  ← MUST BE SET TO 1
    _portCfg          config;
    msgDirectCfg      msgRedirect;
    unsigned char     disableUDPMsgs;
} portCfg;

typedef struct{
    serialConfig      SerialAConfig;
    serialConfig      SerialBConfig;
    unsigned          MasterEthernetBufferSize;
    protocolCfg       AuxTcpPort1;
    protocolCfg       AuxTcpPort2;
    protocolCfg       SerialA;
    protocolCfg       SerialB;
} _portCfg;

typedef struct {
    unsigned char     port;
    unsigned          baud;
    unsigned char     stopBits;
    unsigned char     bits;
    parity_mode       parity;
} serialConfig;

typedef enum {eParityNone, eParityOdd, eParityEven, eParityMulti}parity_mode;

typedef struct {
    V4ProtocolEnum    type;
    union {
        ASCII_Protocol_Cfg  ASCIICfg;
        Binary_Protocol_Cfg BinaryCfg;
    }protocolDefUnion;
}

```

```

} protocolCfg;

enum V4ProtocolEnum {0 = RAW, 1 = BINARY, 2 = ASCII};

typedef struct {
    unsigned          schema;
    unsigned          bufferSize;
    unsigned          timeout;
    unsigned char     PreambleDef;
    unsigned char     PostambleDef;
    unsigned char     lineTerm1Def;
    unsigned char     lineTerm2Def;
    unsigned char     endOfPageDef;
    unsigned char     verbose;
    unsigned char     matchString[4][80];
    unsigned          matchEvent[4];
} ASCII_Protocol_Cfg;

typedef struct {
    unsigned          schema;
    unsigned          bufferSize;
    unsigned          timeout;
    unsigned char     commandMode;
} Binary_Protocol_Cfg;

typedef struct {
    unsigned          PrintControl;
    unsigned          Status1;
    unsigned          Debug;
    unsigned          Status2;
    unsigned          I/OEvents;
    unsigned          PenStatus;
} msgDirectCfg;

```

Parameter Descriptions

saveToFlash	Save configuration to flash.
portCfg	Port configuration structure.
schema -	Revision number for top level structure,
config	portCfg. MUST BE SET TO 1
SerialAConfig -	_portCfg data structure describing port
SerialBConfig -	configuration parameters.
	Serial port A configuration..
	Serial port B configuration..

MasterEthernetBufferSize	Buffer size (bytes) assigned to root Ethernet port 10000. Must be >= 1MB.
AuxTcpPort1	Protocol configuration assigned to TCP port 10002.
AuxTcpPort2	Protocol configuration assigned to TCP port 10003.
SerialA	Protocol configuration assigned to Serial port A.
SerialB	Protocol configuration assigned to Serial port B.
serialConfig	Serial configuration structure.
port	Serial port being configured. 0 = Serial port A, 1 = Serial port B.
baud	Baud rate. Options are: 115200, 57600, 38400, 28800, 19200, or 9600.
stopBits	Stop bits, 1 or 2
bits	Data bits, 5,6,7,or 8.
parity	Parity 0 = None 1 = Odd 2 = Even
MasterEthernetBufferSize	Size of buffer allocated for root TCP/IP port 10000. Must be greater than or equal to 1MB.
protocolCfg	Protocol configuration data structure.
type	Protocol assignment if any. Type will be followed by protocolDefUnion. 0 = None. (No protocol definition will follow) 1 = Binary. (Binary_Protocol_Cfg will follow) 2 = ASCII. (ASCII_Protocol_Cfg will follow)
protocolDefUnion	Protocol configuration data structure union.
ASCII_Protocol_Cfg	ASCII protocol configuration data structure.
Schema	Revision number for ASCII protocol data structure.
bufferSize	Buffer size (bytes) to be assigned to this instance of protocol.
timeout	Maximum timeout in seconds between receipt of bytes of data or command bytes.
PreambleDef	User defineable code, single byte, to be used for Preamble. Default is 0x2, STX.
PostambleDef	User defineable code, single byte, to be used for Postamble. Default is 0x3, ETX.
lineTerm1Def	Line terminator, default is 0XD, carriage return.
lineTerm2Def	Optional 2 nd line terminator, default is line feed, 0XA.
endOfPageDef;	User definable end of page marker, single byte code. Default is Form Feed, 0XC.
verbose	Verbosity control/ 0 = minimal length messages, typically just codes. 1 = Enhanced response messages.

matchString[4][80]	Four 80 character match strings to be used for triggering events when incoming data segments match at least one string. This string data is to be NULL terminated.
matchEvent[4]	Event codes, one for each match string, used to trigger events on the imager when a matching string is detected. Codes TBD.
Binary_Protocol_Cfg	Data structure defining Binary protocol configuration.
Schema	Revision number of data structure. 1
bufferSize	Buffer size, bytes, committed to this instance of protocol.
timeout	Maximum timeout in seconds between receipt of bytes of data or command bytes.
commandMode	Command permission level. 0 = Partial command set, monitor commands only. 1 = Full command set.
msgDirectCfg	Data structure defining Binary protocol configuration.

Configuration data shall be passed to the imager using the msgDirectCfg data structure described below. For each message class, a group of bits will be set or cleared to indicate where messages will be sent. This provides up to 32 bits of routing data for each message group. The bits shall be assigned as follows.

TCP Port 10000	0x1	Bit 0
TCP Port 10001	0x2	Bit 1
TCP Port 10002	0x4	Bit 2
TCP Port 10003	0x8	Bit 3
Serial Port 1	0x10	Bit 4
Serial Port 2	0x20	Bit 5

Any group configured with no bits set, 0, indicates there is no destination port(s) assigned for this group, do not transmit.

PrintControl -	Print control class of messages. Print control messages consist of unsolicited messages sent from the imager to the host, signaling events related to printing ie requests for data and ready to print.
Status1 -	Status 1 class of messages. Status 1 messages are a group of messages made up of error messages, warning messages, and ink level alerts, messages which may require immediate attention from the host.
Debug -	Debug class of messages. The debug messages group currently consists of logging messages used for debug.
Status2 -	Status 2 class of messages. Status 2 messages are event messages related to printing, messages which are not crucial to control print operation. Currently the Page Loaded message falls into this group.

I/OEvents -	I/O Event Reporting class of messages. I/O Event messages consist of Input state change reporting, cleaning station status for operations which were triggered by an input change, and Purge Done messages.
PenStatus -	Pen Status reporting class of messages. This class of messages contains pen status report

Example Configuration

Assume a PLC driving communications over Serial port A, debugging software connected to port 10002. The PLC will be supplying text data, minimal messaging returned from imager, only what is required to maintain synchronization. Assume a host system is acting as supervisor on port 10000.

PrintControl	= 0x10;	Serial port A for all print control messages.
Status1	= 0x06;	Send status and warning messages to monitoring host and debugging system.
Debug	= 0x04;	Send log messages to TCP port 10002.
Status2	= 0x06;	status and warning messages to monitoring host and debugging system.
I/OEvents	= 0;	I/O events not monitored.
PenStatus	= 0;	Pen Status not monitored.

disableUDPMsgs	Flag enabling/disabling UDP debug messages. 0 – Disable transmission of UDP debug messages. 1 – Enable transmission of UDP debug messages. (default)
--------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------

6.53 LOAD_XML_JOB (0xAB)

Description

Command loads text based job configuration onto imager and starts job.

Data

unsigned char [jobConfig\[\]](#);

Data Descriptions

- [jobConfig](#)
- Job configuration data.
 - See Chapter 8 for details of XML job configuration.



7.0 Simple ASCII Protocol

Supported Host Commands

Command	Code	Description
Print_Dynamic_Data	NONE	Send print data to the Imager
Go_To_Park_Position	0x20	Move the Service Station to Park Position
Go_To_Print_Position	0x21	Move the Service Station to Print Position
Execute_Clean	0x22	Command Service Station to execute a cleaning cycle
Execute_Purge	0x23	Commands pens to purge ink in order to recover any dried nozzles.
Clear_Print_System	0x24	Command imager to clear all image buffers and any pieces that were queued in by the sensor.
Get_System_Status	0x25	Request system ready to print status. Imager will reply with System_Ready or System_Not_Ready.
Start_Print	0x26	Command the imager to start a job stored in the imager.
Stop_Print	0x27	Command the Imager to stop printing.
Redirect_Imager_Messages	0x35	Redirect unsolicited messages to port this command originated from.
Enable_Printing	0x37	Once printing has been started this command enables the Top Of Form Sensor.
Initialize_Static_Record_Data	0x40	Send initialization data to the printer that will not change throughout production.
Eject_Service_Station_Tray	0x41	Command the imager to eject the service station tray for cleaning/replacement.

Imager Responses

Response	Code	Description
Command_Acknowledge	0x20	Acknowledge the Imager received a command from host
Data_Buffer_Available	0x28	Message letting host know more print buffers are available
Cleaning_Station_Command_Success	0x29	Message sent when cleaning station command executes successfully
Cleaning_Station_Command_Failed	0x30	Message sent when cleaning station command fails to execute properly
Imager_Error	0x31	Notifies the host an error occurred
Imager_Warning	0x32	Notifies the host a warning occurred
System_Not_Ready	0x33	Response to Get_System_Status indicating system is not ready to print
System_Ready	0x34	Response to Get_System_Status indicating system is ready to print
Data_Acknowledge	0x36	Acknowledge the Imager received print data from host.
Purge_Cycle_Completed	0x38	Message notifying the host Purge Cycle completed without error.
Page_Printed (Data Buffer Available)	0x39	Message notifying the host a page has been printed and data buffer is available to accept print data.

Host Commands

7.1 Print_Data

The Print Data may be any character supported by the font in use, code range from 32 to 255 decimal. Unicode is not acceptable. Each line of data must be terminated by at least one terminating character, default being 0xD, carriage return. A line feed (0xA) in addition to or in place of carriage return is acceptable as well. Each page of data must be terminated by a unique code, default being 0xC, form feed. A complete page of data must be received by the imager before a page may become eligible for printing.

The imager will signal availability of buffers for print data by sending a BUFFER_AVAILABLE command. This command will be sent to the host whenever the Imager detects an empty buffer, either pre-filling or upon printing a page, freeing up a buffer.

Page data will be accepted at any time after a job has been loaded and successfully started, providing buffer space is available to hold user print data. The imager will issue buffer available notification messages upon detecting an available buffer after the host transmits the first page of data. One BUFFER_AVAILABLE command message will be transmitted for each buffer as it becomes available. Host systems may use this notification to synchronize data transfers to the imager or may transmit data as it becomes available so long as a buffer is available.

The host must provide enough print data to keep at least one buffer filled prior to triggering a TOF on the imager. An error message will be returned if the host attempts to send pages of data when there are no empty buffers. We recommend the host system synchronize print data commands being sent to the imager with BUFFER_AVAILABLE commands being received from the imager though this is not a strict requirement.

7.2 Go_To_Park_Position (0x20)

This is a Cleaning Station command (AC model of jet.engine Imagers) used to instruct the Service Station to move it to Park Position and "cap" the pens.

This command will cause a response Status message of one of the following two types:

- Cleaning Station Command Failed / Timed Out/ Command Error
- Cleaning Station Command Success / Completed/ Command Success

Example: If stopping the conveyor for several minutes without printing, send the Park command to keep the pens from drying out.

Sample Command Hex String: 02 20 03

7.3 Go_To_Print_Position (0x21)

This is a Cleaning Station command (AC model of jet.engine Imagers) used to instruct the Service Station to move it to Print Position or “uncap” and position the pens over the product in preparation for printing.

This command will cause a response Status message of one of the following two types:

- Cleaning Station Command Failed / Timed Out/ Command Error
- Cleaning Station Command Success / Completed/ Command Success

Example: If stopping the conveyor for several minutes without printing, send the Park command to keep the pens from drying out. Then send a Print command before starting the conveyor again.

Sample Command Hex String: 02 21 03

7.4 Execute_Clean (0x22)

This is a Cleaning Station command (AC model of jet.engine Imagers) used to instruct the Service Station to execute a pen cleaning cycle.

This command will cause a response Status message of one of the following two types:

- Cleaning Station Command Failed / Timed Out/ Command Error
- Cleaning Station Command Success / Completed/ Command Success

Example: If you desire to clean the pens after several hundred prints, stop your conveyor and send this command, and then start the conveyor again.

Sample Command Hex Sting: 02 22 03

7.5 Execute_Purge (0x23)

This command executes a purge cycle on all installed pens that have been configured for purging. A single purge cycle will eject 25 pixels on every nozzle. Purge should be used to recover pens after prolonged decap time. If more then 25 pixels are required to recover print quality multiple purge commands need to be sent to the printer.

Sample Command Hex String: 02 23 03

7.6 Clear_Printer_System (0x24)

This command will clear all image buffers and any pieces that were queued in by the sensor to be printed, effectively resetting print on the system.

Sample Command Hex String: 02 24 03

7.7 Get_System_Status (0x25)

This command is used to query the Imager if it is ready to print or not. Imager will reply back to Host if it's ready to print with System_Ready or System_Not_Ready if it's not ready to print.

Sample Command Hex String: 02 25 03

7.8 Start_Print (0x26)

Start job stored on imager. If no job name is provided the default job, if loaded and configured on imager, will be executed. Job names must be no longer than 8 characters followed by a period and no more than three characters for the extension. 8.3 DOS filenames convention. OnBoardRIP GUI saves jobs as "Job#.job" where the # is the number selected by the user.

After sending Start_Print the host should issue Get System Ready to determine when the imager is ready to start printing. If the job requires dynamic data, the imager will issue a Data Buffer Request command to the host.

Example:

Start default job, no job name specified.

<STX><26><ETX>

Sample Command Hex String: 02 26 03

Start job with specified job name.

<STX><26><Job1.job><ETX>

Sample Command Hex String: 02 26 4A 6F 62 31 2E 6A 6F 62 03

7.9 Stop_Print (0x27)

Stop printing, terminate job currently running and return the imager to idle. A part of stop print the system will clear any unprinted buffers.

Sample Command Hex String: 02 27 03

7.10 Redirect_Imager_Messages (0x35)

This command is sent from the Host to the Imager to instruct the imager to send all unsolicited messages i.e. Data_Buffer_Available, Page_Printed to the port over which this command was received.

Sample Command Hex String: 02 35 03

7.11 Enable_Printing (0x37)

This command enables the Top of Form sensor for printing. The imager automatically enables the Top of Form sensor for jobs that do not require external data. Hosts have to enable the sensor for any jobs that require external data. The host may use this command to synchronize multiple imager systems or make sure Imager has data before it starts printing.

Sample Command Hex String: 02 37 03

7.12 Initialize_Static_Record_Data (0x40)

This command provides the imager with data used to initialize static data fields. Static data fields are fields that could be initialized once at beginning of production and the imager will print the same data until it is changed or the job is stopped. Data may contain more than one line, blank lines allowed, and must be terminated with an End Of Page marker, Form Feed (0x0C) by default.

Example:

The following example illustrates a static data initialization command covering 4 lines. The 3rd line is blank. Default end of line and end of page markers, Carriage Return(CR), Line Feed(LF) and Form Feed(FF) are used for illustration.

<STX><0x40><line 1><CR><LF><line 2><CR><LF><CR><LF><line 4><FF><ETX>

Sample Command Hex String:

02 40 6C 69 6E 65 20 31 0D 0A 6C 69 6E 65 20 32 0D 0A 0D 0A 6C 69 6E 65 20 31 0C 03

7.13 Eject_Service_Station_Tray (0x41)

This command can only be used for Auto Capping (AC) units that have a service station tray. This command eject the service station tray for cleaning or replacement.

Sample Message Hex String: 02 41 03

Imager Responses

7.14 Command_Acknowledge (0x20)

This message is sent from the Imager to the Host when an ASCII Protocol command message is received confirming to the Host the command has been received without error. The third byte following of the message will contain the ID of the command the Imager is acknowledging.

Sample Message Hex String: 02 20 26 03

7.15 Data_Buffer_Available (0x28)

The host may start to send print data right after a successful job load. During buffer prefill the imager will reply to every data command with this command if there are more buffers available that were not yet filled. The host can use this message to make sure not to send more data than there are print buffers.

Sample Message Hex String: 02 28 03

7.16 Cleaning_Station_Command_Success (0x29)

This is a Cleaning Station status command that responds to any of the Cleaning Station Commands, indicating the command was executed successfully.

Sample Message Hex String: 02 29 03

7.17 Cleaning_Station_Command_Failed (0x30)

This is a Cleaning Station status command that responds to any of the Cleaning Station Commands, indicating the command failed to execute.

Note: If the user ignores the error, normal operation will resume. This means that the Imager will stay in the position it failed in and can start printing even though it may not be in the proper position. In this situation, it is possible to print over the Ink Tray instead of on the product.

Sample Message Hex String: 02 30 03

7.18 Imager_Error (0x31)

This message will be sent to the Host if any error condition is detected with an Imager. Error codes are listed below. Error codes are reported in ASCII therefore error code 11 will be reported as "0x31 0x31"



Error Name	Error Code	Error Description
PEN_BUFFER_UNDERFLOW	1	Attempted to print but no data was available
ILLEGAL_COMMAND	2	The command sent by host does not exist, is not formatted correctly, is not allowed at this time(Imager is in the wrong mode), or may have failed for unknown reasons.
EXTERNAL_IO_DEVICE_FAIL	3	Cannot configure or use external IO device.
PEN_BUFFER_WRITE_ERROR	4	Had a hardware failure trying to load the Image Buffer
PEN_BUFFER_OUT_OF_SYNC	5	Data from different pages was printed on the same piece.
BUFFER_LOAD_ERROR	6	Error loading buffer or trying to load a non-existent buffer.
PEN_POWER_FAULT_ERROR	7	This error happens if the user powered up the Imager and then disconnected the pen driver.
COMMAND_EXECUTION_ERROR	9	Imager failed to execute a command.
Communications Buffer Overflow	10	Communication Buffer Overflowed.
SYSTEM_OVERLOAD	11	Main Message Queue is full, messages have been dropped
PRINT_CYCLE_FAILURE	12	Job failed to execute.
PARTIAL_PAGE_SKIP	21	Underrun occurred but not on all pens. Data printed on the next page will be out of synch between pens.
Unused	22	
OUT_OF_BUFFERS_FAILURE	23	Imager received print data but all print buffers were full.
COMMUNICATIONS_ERROR	26	Did not receive a complete command.
OBR_RENDERING_ERROR	27	Either a character was not rendered at all, or was only partially rendered

Sample Error Messages

STX	IMAGER_ERROR_ASCII	ILLEGAL_COMMAND	ETX
02	31	32	03

STX	IMAGER_ERROR_ASCII	SYSTEM_OVERLOAD	ETX
03	31	31	03

7.19 Imager_Warning (0x32)

This message will be sent to the host if any warning condition is detected in the Imager. Unlike errors warning are not critical failures and may not need immediate intervention.

Warning codes are listed below. Warning codes are reported in ASCII therefore warning code 6 would be reported as "0x36"

Warning Name	Warning Code	Description
PEN_DRIVER_NOT_RESPONDING	1	No pen driver present.
ILLEGAL_CARTRIDGE_ID	2	Set if no pen id was read.
PEN_SYNC_CHECK_DISABLED	3	Pen Synchronization check is disabled.
PEN_POWER_UP_FAILED	4	Pen did not power up correctly.
PEN_EARLY_TERMINATION	5	Pen stopped printing early.
INVALID_COMMAND	6	Command cannot be executed while a job is running.

Sample Warning Message

STX	IMAGER_WARNING	INVALID_COMMAND	ETX
02	32	36	03

7.20 System_Not_Ready (0x33)

This message is a response to Get_System_Status command indicating the Imager is not ready to print. If Imager is not ready to print it will not receive any print data.

Sample Message Hex String: 02 33 03

7.21 System_Ready (0x34)

This message is a response to Get_System_Status command indicating the Imager is ready to print and will accept print data.

Sample Message Hex Sting: 02 34 03

7.22 Data_Acknowledge (0x36)

This message is sent from the Imager to the Host acknowledging print data has been received.

Sample Message String:

STX	ACKNOWLEDGE_ASCII	DATA_ACK	ETX
02	20	36	03

7.23 Purge_Cycle_Completed (0x38)

The requested purge cycle (SPIT) has completed without errors.

Sample Message Hex String: 02 38 03

7.24 Page_Printed (0x39)

This message is sent from the Imager to the Host, indicating a page has been printed and a buffer is available for filling. This message could be used as request for data.

Sample Message Hex String: 02 39 03

8.0 XML Job Structure LOAD_XML_JOB (0xAB)

8.1 Job Configuration

Imager firmware 218.x.xxx and 518.x.xxx supports text based job configuration allowing users to edit a job layout using a text editor. An XML like syntax is used to describe all parameters required. The XML based job configuration allows users to create job layouts using a text based syntax which may be produced and edited using a text or XML editor. All data contained within the configuration file are from the ASCII character set. Binary data is indirectly supported by encoding binary data using BASE64 encoding prior to insertion into the configuration file. A job configuration file must contain a minimal set of configuration data, the firmware calculates parameters when possible, eliminating need for the user to specify units such as number of print objects or size of page buffers. There are many parameters which are optional, defaulted by firmware if not specified by the user, simplifying job configuration for jobs which are simple. The full capacity of OnBoardRip Job is supported by this syntax should a job require more advanced features. XML job configuration file is passed down to the imager using LOAD_XML_JOB (0xAB) command.

8.2 Job File Structure

The structure of the configuration file is modeled after XML, each parameter specified by an opening tag ie '`<JOB_RECORD>`' and closed by a matching closing tag ie '`</JOB_RECORD >`'. Major blocks, print objects, may be described with an opening tag for the print object, followed by another opening tag for a parameter of the object. This may be nested several levels if needed, the only rule being any closing tag must match the most recent opening tag encountered while parsing.

Whitespace (spaces, cr, lf, tabs, etc) which appear outside of parameters are ignored, allowing configuration text to be wrapped around, enhancing human readability.

Provisions have been made to allow the user to place text comments or descriptive text within the file. If the comments are inserted *before* the opening `<JOB_RECORD>` tag for the job, the parser will ignore the comments while looking for the first opening tag in the job. Limitation being no text may be inserted which matches any tags supported by this syntax. For example the string "`<JOB_RECORD >`" must not appear in comment text preceding the actual start of the job description. Comments may appear within the job description as well but they must be enclosed with the `<COMMENT></COMMENT>` tag.

Four major blocks may appear directly below the level of `JOB_RECORD`, describing global job configuration parameters, raster buffer configuration, print object configuration, user supplied font data and user supplied bitmap data. Job configurations must be opened with the "`<JOB_RECORD>`" tag, typically followed by global configuration data ie job name, job file name, and measurement units (optional). Raster buffer configuration data may follow, `<RASTER_BUFFER_CFG>`, describing the page length and height, number of buffers, and other parameters related to the page buffers. Once raster buffer configuration has been fully described, print object configuration follows, `<RIP_CONFIG>`, with specification of all print objects required. Binary font data, `<FONTFILE>`, and bitmap data, `<BITMAPFILE>` may follow the OBR data if the fonts required or bitmaps are not pre-stored on the imager. Binary data has to be converted to Base64 encoding to be able to copy it into the XML file.

8.3 Main Job Tags

Below is a simple job definition that will print Hello World in the upper left corner of the page. Explanation of all job tags follows.

```
<JOB_RECORD>
  <JOB_FILENAME>Job1.job</JOB_FILENAME>
  <JOB_NAME>Hello World Test Job</JOB_NAME>
  <RASTER_BUFFER_CFG>
    <NUM_BUFFS>4</NUM_BUFFS>
    <PAGE_HEIGHT>2.0</PAGE_HEIGHT>
    <PAGE_LENGTH>10.0</PAGE_LENGTH>
    <HORIZONTAL_DPI>300</HORIZONTAL_DPI>
  </RASTER_BUFFER_CFG>
  <COMMENT>This job will print "Hello Word" in the upper left corner</COMMENT>
  <RIP_CONFIG>
    <FIELDBLOCK>
      <XPOS>0.0</XPOS>
      <YPOS>0.0</YPOS>
      <WIDTH>1.8</WIDTH>
      <HEIGHT>1.8</HEIGHT>
      <FONT_FILENAME>Font_5.fnt</FONT_FILENAME>
      <FONT_NAME>Times New Roman_12_WR100_TBS0</FONT_NAME>
      <LINE_DEF>
        <FIXED_TEXT>Hello World</FIXED_TEXT>
      </LINE_DEF>
    </FIELDBLOCK>
  </RIP_CONFIG>
</JOB_RECORD>
```

<JOB_RECORD>	Opening tag for job configuration data. All data which appears before this tag is assumed to be comment data. This tag must appear first in the configuration segment. Not data should appear after the </JOB_RECORD> closing tag.
<JOB_FILENAME>	Filename assigned to this job. Filename must follow the 8.3 DOS file naming standard with "*.job" extension.
<JOB_NAME>	Descriptive job name may be up to 64 characters in length.
<RASTER_BUFFER_CFG>	Start of raster buffer configuration segment.
<RIP_CONFIG>	Start of OBR configuration data. All layout information is defined with this tag.
<UNITS> (Optional)	Units parameter. Specifies which units inches or millimeters will be used to define the job. <i>Values: INCHES, MILLIMETERS</i> . If units are not defined inches will be used as default.
<COMMENT> (Optional)	Comments, may contain any data except '<' and be placed anywhere within the job configuration file.

8.4 Raster Buffer Configuration Segment

The following example illustrates a full specification of raster buffer configuration with all possible parameters. Only the top four tags are mandatory specifications for which there is no default. All other items are optional, may be left out if defaults are suitable

```
<RASTER_BUFFER_CFG>
  <NUM_BUFFS>2</NUM_BUFFS>
  <PAGE_HEIGHT>2.0</PAGE_HEIGHT>
  <PAGE_LENGTH>10.0</PAGE_LENGTH>
  <HORIZONTAL_DPI>300</HORIZONTAL_DPI>
  <GLOBAL_CONTRAST>0</GLOBAL_CONTRAST>
  <RASTER_BUFFER_PULSE_CONFIG>
    <RASTER_BUFFER_PULSE_GEN1_TRIGGER>1</RASTER_BUFFER_PULSE_GEN1_TRIGGER>
    <RASTER_BUFFER_PULSE_GEN2_TRIGGER>1</RASTER_BUFFER_PULSE_GEN2_TRIGGER>
  </RASTER_BUFFER_PULSE_CONFIG>
  <RASTER_BUFFER_PULSE_CONFIG>
    <RASTER_BUFFER_PULSE_GEN1_TRIGGER>0</RASTER_BUFFER_PULSE_GEN1_TRIGGER>
    <RASTER_BUFFER_PULSE_GEN2_TRIGGER>1</RASTER_BUFFER_PULSE_GEN2_TRIGGER>
  </RASTER_BUFFER_PULSE_CONFIG>
</RASTER_BUFFER_CFG>
```

- <NUM_BUFFS>** Number of page buffers to allocate for this job.
Minimum = 1;
Maximum = 32
- <PAGE_HEIGHT>** Height of page. May be expressed in inches or millimeters depending on the value of the **<UNITS>** tag.
Minimum = 1.5 inches.
Maximum = 3 inches.
- <PAGE_LENGTH>** Length of page. May be expressed in inches or millimeters depending on the value of the **<UNITS>** tag.
Minimum = 0.5 inches.
Maximum = Resolution dependent.

Resolution	600	300	200	150	120	100	75
Max Page Length (in)	54	108	162	216	270	324	432
Max Page Length (mm)	1372	2743	4115	5486	6858	8230	10973

- <HORIZONTAL_DPI>** Horizontal print resolution. Units: 600, 300, 200, 150, 120, 100, 75
- <GLOBAL_CONTRAST>** (Optional) Print density adjustment. This parameter could be used to reduce the density of the imager reducing the amount of ink that will be used to print the image.
Range 0 to 15. (6.25% decrease per increment).
Default = 0; (100%)

<RASTER_BUFFER_PULSE_CONFIG> (optional) Opening tag for segment describing buffer pulse configuration. The outputs can be configured to pulse every time a buffer is printed. Specifications for both output 1 pulse and output 2 pulsing may be specified for each page buffer allocated. If pulse configurations are not defined for each buffer allocated, the last specification will be carried forward for all remaining buffers. Pulse duration has to be setup using OUTPUT_CONTROL 0x42 command.

<RASTER_BUFFER_PULSE_GEN1_TRIGGER> Pulse output 1 upon end of page event
 0 = No pulse (Default)
 1 = Pulse output at end of page.
 2 = Pulse output at start of page

<RASTER_BUFFER_PULSE_GEN2_TRIGGER> Pulse output 2 upon end of page event
 0 = No pulse (Default)
 2 = Pulse output at start of page
 1 = Pulse output at end of page.

8.5 Rip Configuration Blocks

The rip configuration block defines print objects which may be placed on a page, field block, record block, and bitmaps.

```
<RIP_CONFIG>
  <FIELDBLOCK>
    <XPOS>0.0</XPOS>
    <YPOS>0.0</YPOS>
    <WIDTH>1.8</WIDTH>
    <HEIGHT>1.8</HEIGHT>
    <FONT_FILENAME>Font_5.fnt</FONT_FILENAME>
    <FONT_NAME>Times New Roman_12_WR100_TBS0</FONT_NAME>
    <LINE_DEF>
      <FIXED_TEXT>Hello World</FIXED_TEXT>
    </LINE_DEF>
  </FIELDBLOCK>
  <RECORD_BLOCK>
    <XPOS>0.5</XPOS>
    <YPOS>0.0</YPOS>
    <WIDTH>1</WIDTH>
    <HEIGHT>1</HEIGHT>
    <FONT_FILENAME>Font_6.fnt</FONT_FILENAME>
    <FONT_NAME>Verdana_20_WR75_TBS0</FONT_NAME>
    <STARTING_FIELD>1</STARTING_FIELD>
    <RECORD_BLOCK_LINES>4</RECORD_BLOCK_LINES>
  </RECORD_BLOCK>
  <BITMAP_BLOCK>
    <XPOS>0.0</XPOS>
    <YPOS>0.0</YPOS>
    <WIDTH>3</WIDTH>
    <HEIGHT>1.5</HEIGHT>
    <BITMAP_DEF>
      <BITMAP_DEF_STATIC>0</BITMAP_DEF_STATIC>
      <BITMAP_DEF_FILENAME>J01_000.bmp</BITMAP_DEF_FILENAME>
      <BITMAP_DEF_NAME>J01_000_{6913F562}</BITMAP_DEF_NAME>
```

```
</BITMAP_DEF>  
</BITMAP_BLOCK>  
</RIP_CONFIG>
```

8.5.1 Field Block

- <FIELDBLOCK>** (optional) Beginning for Field block configuration data. Number of field blocks is limited only by imager resources ie memory.
- <XPOS>** Object top left corner X Position on the page defined in inches or millimeters depending on the **<UNITS>** tag. Object must be within bounds of page definition. Allow for object width when setting.
- <YPOS>** Object top left corner Y Position on the page defined in inches or millimeters depending on the **<UNITS>** tag. Object must be within bounds of page definition. Allow for object height when setting.
- <WIDTH>** Object width defined in inches or millimeters depending on the **<UNITS>** tag. Whole object must fit within bounds of page definition.
- <HEIGHT>** Object height defined in inches or millimeters depending on the **<UNITS>** tag. Whole object must fit within bounds of page definition.
- <FONT_FILENAME>** Filename of font used for this print object. Must be included whether font is embedded in configuration file or prestored in the imager. Filename adheres to DOS 8.3 standard with "*.fnt" as the font file extension. Not needed if print object does not contain human readable data.
- <FONT_NAME>** Up to 64 characters of name data. Must match name data found in font file exactly. Case sensitive.
- <ROTATION>** (Optional) Specifies rotation of print object.
Valid parameters: 0 , 90, 180, 270. Default: 0
- <PRINT_DENSITY>** (Optional) Specifies print density for individual print objects.
Range 0 to 15. (6.25% decrease per increment).
Default = 0; (100%)
- <REMOVE_LEADING_WHITE_SPACE_LINES>** (Optional) Boolean flag, when set, removes leading whitespace from beginning of line. Default = 0;
- <REMOVE_LEADING_WHITE_SPACE_FIELDS>** (Optional) Boolean flag, when set, removes leading whitespace from beginning of field. Default = 0;
- <REMOVE_TRAILING_WHITE_SPACE_LINES>** (Optional) Boolean flag, when set, removes trailing whitespace from end of line. Default = 0;
- <REMOVE_TRAILING_WHITE_SPACE_FIELDS>** (Optional) Boolean flag, when set, removes trailing whitespace from end of field. Default = 0;
- <REMOVE_BLANK_LINES>** (Optional) Boolean flag, when set, removes blanks lines from field block. Lines are shifted up to replace the blank line. Default = 0.

<BARCODE_TYPE>

Barcode type. Below is a list of supported barcodes.

Note: When creating a barcode at minimum the host has to specify the type of barcode, object size and number of characters that will be encoded into the barcode. Firmware will calculate magnification factor or bar size needed to fill out window as closely as possible for given resolution. If magnification factor is specified and object width is not wide enough a rendering error will be generated.

BARCODE_DATAMATRIX_10
BARCODE_DATAMATRIX_12
BARCODE_DATAMATRIX_14
BARCODE_DATAMATRIX_16
BARCODE_DATAMATRIX_18
BARCODE_DATAMATRIX_20
BARCODE_DATAMATRIX_22
BARCODE_DATAMATRIX_24
BARCODE_DATAMATRIX_26
BARCODE_DATAMATRIX_32
BARCODE_DATAMATRIX_36
BARCODE_DATAMATRIX_40
BARCODE_DATAMATRIX_44
BARCODE_DATAMATRIX_48
BARCODE_DATAMATRIX_8_18
BARCODE_DATAMATRIX_8_32
BARCODE_DATAMATRIX_12_26
BARCODE_DATAMATRIX_12_36
BARCODE_DATAMATRIX_16_36
BARCODE_DATAMATRIX_16_48
BARCODE_UPC_A
BARCODE_CODE128
BARCODE_CODE128_CODESET_A
BARCODE_CODE128_CODESET_B
BARCODE_CODE128_CODESET_C
BARCODE_EAN128
BARCODE_EAN13
BARCODE_CODE39
BARCODE_CODE93
BARCODE_INT2OF5
BARCODE_ITF14
BARCODE_POSTNET
BARCODE_IMB

<BARCODE_MAG_FACTOR>

(Optional) Barcode magnification factor specified in percents.

Range: 20-140

Range for UPC-A/EAN13: 80-200

Default = 100 (100%)

<BARCODE_HUMAN_READABLE>

(Optional) Boolean flag to enable barcode human readable.

0 = No Human readable. (Default)

	1 = Human readable characters included.
<BARCODE_TEXT_JUSTIFICATION>	(Optional) Human readable text alignment. 0 = Left justify 1 = Center (Default)
<BARCODE_HEIGHT>	(Optional) Barcode bar height specified in pixels. Used for 1-D barcodes. Will be calculated automatically if omitted to fit the field block height. When defining a specific bar height users need to specify field block height to fit the bars and human readable.
<BARCODE_CHECK_CHARACTER>	(Optional) Boolean, include check character for barcodes where check character is optional. 0 = No check character (default) 1 = Include check character
<BARCODE_SHAVING>	(Optional) Barcode shaving. Number of pixels that will be shaved from black bar widths to compensate for ink bleed. Units = Pixels. Default = 0
<BARCODE_BEARER_BAR>	(Optional) Boolean, include bearer bar for barcodes which support bearer bars. 0 = No Bearer Bar (default) 1 = Horizontal bars only 2 = Vertical bars only 3 = Both horizontal and vertical bars
<BARCODE_BEARER_BAR_WIDTH>	(Optional) Barcode bearer bar width. When not specified firmware will calculate bearer bar width based on magnification factor. Units = pixels, Default = 0
<BARCODE_PARAMETER_1>	(Optional) Barcode supplemental parameter. Usage dependent on specific barcode. May be unused for some barcodes. See Appendix B for details.
<BARCODE_PARAMETER_2>	(Optional) Barcode supplemental parameter. Usage dependent on specific barcode. May be unused for some barcodes. See Appendix B for details.
<BARCODE_PARAMETER_3>	(Optional) Barcode supplemental parameter. Usage dependent on specific barcode. May be unused for some barcodes. See Appendix B for details.
<BARCODE_PARAMETER_4>	(Optional) Barcode supplemental parameter. Usage dependent on specific barcode. May be unused for some barcodes. See Appendix B for details.
<BARCODE_FONT_FILENAME_1>	(Optional) Font file name to be used for human readable. Name of file must be in DOS 8.3 format with "*.fnt" file extension. Not needed if no human readable characters are rendered.

<BARCODE_FONT_NAME_1>	(Optional) Font name of the human readable font that should be used when printing the barcode. This name must match the name appearing in the font file exactly, case sensitive.
<BARCODE_FONT_FILENAME_2>	(Optional) Font file name for barcodes requiring a secondary smaller font (UPC_A). Name of file must be in DOS 8.3 format with "*.fnt" file extension.
<BARCODE_FONT_NAME_2>	(Optional) Secondary font name. This name must match the name appearing in the font file exactly, case sensitive.
<LINE_DEF>	Start of line definition, define print item appearing on this line within field block. Multiple items could be configured on one line. Each <LINE_DEF> specifies a new line within the Field Block.
<FIXED_TEXT>	(Optional) Line item, fixed text. Data specifies fixed string to be printed. Example: <FIXED_TEXT>Hello World</FIXED_TEXT>
<DYNAMIC_TEXT>	(Optional) Line item dynamic text. 1 based parameter specifies field number within a record which specifies the data to be printed. Example: <DYNAMIC_TEXT>3</DYNAMIC_TEXT>
<FIELD_BLOCK_STATIC_DATA>	Line item static text. Static data is initialized using INITIALIZE_STATIC_RECORD_DATA 0x85 once after job starts and is used until updated or when a job is stopped. Field number within the print record needs to be provided which will be used to initialize the data.
<FIELD_BLOCK_DATE_TIME>	(Optional) Line item Date/Time field. Date time parameters are defined within this tag. Example: <FIELD_BLOCK_DATE_TIME> <DATE_TIME_MONTH_OFFSET>3</DATE_TIME_MONTH_OFFSET> <DATE_TIME_FORMAT_STRING>Today IS: %c</DATE_TIME_FORMAT_STRING> </FIELD_BLOCK_DATE_TIME>

NOTE: ONLY ONE DATE OFFSET CAN BE DEFINED PER DATE TIME TAG

<DATE_TIME_MONTH_OFFSET>	(Optional) Month offset from current time. Range = 0 – 99 0 = default Cannot be combined with hours or days offset.
<DATE_TIME_HOURS_OFFSET>	(Optional) Hours offset from current time. Range = 0 to 99 0 = default Cannot be combined with months or days offset.

<DATE_TIME_DAYS_OFFSET> (Optional) Days offset from current time.
 Range = 0 to 999,
 0 = default
 Cannot be combined with months or hours offset.

<DATE_TIME_FORMAT STRING> (Optional) Date/Time format string. ASCII format string. String may consist of any arbitrary text with the following sequences reserved to define date/time strings.

Format String	Description
%a	An abbreviation for the day of the week.
%A	The full name for the day of the week.
%b	An abbreviation for the month name.
%B	The full name of the month.
%c	A string representing the complete date and time, in the form: Mon 4/1/2010 1:13:13 PM 2010
%d	The day of the month, formatted with two digits.
%H	The hour (on a 24-hour clock), formatted with two digits.
%I	The hour (on a 12-hour clock), formatted with two digits.
%j	The count of days in the year, formatted with three digits (from `001' to `366').
%m	The month number, formatted with two digits.
%M	The minute, formatted with two digits.
%p	Either `AM' or `PM' as appropriate.
%S	The second, formatted with two digits.
%U	The week number, formatted with two digits (from `00' to `53'; week number 1 is taken as beginning with the first Sunday in a year). See also %W.
%w	A single digit representing the day of the week: Sunday is day 0.
%W	Another version of the week number: like `%U', but counting week 1 as beginning with the first Monday in a year.
%x	A string representing the complete date, in a format like: Mon Apr 01 1992
%X	A string representing the full time of day(hours, mins, and secs), in a format like: 13:13:13
%y	The last two digits of the year.
%Y	The full year, formatted with four digits to include the century.
%Z	Defined by ANSI C as eliciting the time zone if available; it is not available in this implementation (which accepts `%Z' but generates no output for it).
%%	A single character, `%'.

<DATE_TIME_MONTH_OFFSET_SOURCE_LINE> (Optional) Date/Time month offset set by external static data. Field number, 1 based from within record from which initialization data is taken.

<DATE_TIME_HOURS_OFFSET_SOURCE_LINE> (Optional) Date/Time hours offset set by external static data. Field number, 1 based from within record from which initialization data is taken.

<DATE_TIME_DAYS_OFFSET_SOURCE_LINE> (Optional) Date/Time days offset set by external static data. Field number, 1 based from within record from which initialization data is taken.

<FIELD_BLOCK_COUNTER> (Optional) Line item counter definition. Multiple counters could be defined in one field block.

Example:

```
<FIELD_BLOCK_COUNTER>
  <COUNTER_MIN_FIELD_WIDTH>3</COUNTER_MIN_FIELD_WIDTH>
  <COUNTER_PAD_ZEROES>1</COUNTER_PAD_ZEROES>
  <COUNTER_START_COUNT>5</COUNTER_START_COUNT>
  <COUNTER_INCREMENT>5</COUNTER_INCREMENT>
  <COUNTER_WRAP_LIMIT>200</COUNTER_WRAP_LIMIT>
  <COUNTER_DISPLAY_SIGN>0</COUNTER_DISPLAY_SIGN>
  <COUNTER_TRIGGER_INTERVAL>2</COUNTER_TRIGGER_INTERVAL>
  <COUNTER_WRAP_RESTART_COUNT>10</COUNTER_WRAP_RESTART_COUNT>
</FIELD_BLOCK_COUNTER>
```

<COUNTER_MIN_FIELD_WIDTH> (Optional) When included this tag enables padding.
0 - no padding (Default)
1 to N – Pad with N spaces or '0's.

<COUNTER_PAD_ZEROS> (Optional) If padding is enabled this tag defines padding type.
0 - Pad with spaces.
1 - Pad with '0's.

<COUNTER_START_COUNT> (Optional) Counter starting value. Default = 0

<COUNTER_INCREMENT> (Optional) Counter increment value, may be positive or negative. Default = 1.

<COUNTER_WRAP_LIMIT> (Optional) Wrap Limit. Maximum value the counter will count to before it restarts.

<COUNTER_DISPLAY_SIGN> (Optional) Display Sign.
0 – Do not display sign. (Default)
1 – Display sign.

<COUNTER_TRIGGER_INTERVAL> (Optional) Skip count, update counter every N pages.
Range 1 – 2,000,000

Default = 1.

<COUNTER_WRAP_RESTART_COUNT> (Optional) Number the counter will reset to after a wrap limit is reached.

<COUNTER_INIT_LINE> (Optional) If counter is to be initialized by the host this parameter defines the Field number within the initialization record the starting value will come from. 1 based field number.

<FIELD_BLOCK_ALPHA_DATE_TIME> Line item Alpha Date/Time parameters are defined within this tag.

Example:

```
<FIELD_BLOCK_ALPHA_DATE_TIME>
  <DATE_TIME_MONTH_OFFSET>10</DATE_TIME_MONTH_OFFSET>
  <DATE_TIME_FORMAT_STRING>MT YR</DATE_TIME_FORMAT_STRING>
</FIELD_BLOCK_ALPHA_DATE_TIME>
```

<DATE_TIME_FORMAT_STRING> ASCII Date/Time format string. String may consist of any arbitrary string composed the following substrings.

Format String	Description
"HR"	Insert alpha code for hour.
"MN"	Insert alpha code for minute.
"DA"	Insert alpha code for day.
"DY"	Insert alpha code for day of year
"DT"	Insert alpha code for day of month.
"WK"	Insert alpha code for week of year
"MT"	Insert alpha code for month
"YR"	Insert alpha code for year
' '	Insert space
':'	Insert colon
'.'	Insert period
','	Insert comma
'-'	Insert dash
';'	Insert semicolon
'_'	Insert underscore

<DATE_TIME_MONTH_OFFSET> (Optional) Month offset from current time.
Range = 0 – 99
Default = 0
Cannot be combined with hours or days offset.

<DATE_TIME_HOURS_OFFSET> (Optional) Hours offset from current time.
Range = 0 to 99

Default = 0

Cannot be combined with months or days offset.

<DATE_TIME_DAYS_OFFSET> (Optional) Days offset from current time.

Range = 0 to 999,

Default = 0

Cannot be combined with months or hours offset.

<DATE_TIME_MONTH_OFFSET_SOURCE_LINE> (Optional) Date/Time month offset set by external static data. Field number, 1 based from within the initialization record.

<DATE_TIME_HOURS_OFFSET_SOURCE_LINE> (Optional) Date/Time hour offset set by external static data. Field number, 1 based from within the initialization record.

<DATE_TIME_DAYS_OFFSET_SOURCE_LINE> (Optional) Date/Time days offset set by external static data. Field number, 1 based from within the initialization record.

<FIELD_BLOCK_SHIFT_CODE> (Optional) Line item shift code. No parameters are required. Shift codes have to be pre-defined.

Example: **<FIELD_BLOCK_SHIFT_CODE></FIELD_BLOCK_SHIFT_CODE>**

8.5.2 Record Block

<RECORD_BLOCK> (Optional) Beginning for Record Block configuration data. Number of Record Blocks is limited only by imager resources ie memory.

Example:

```
<RECORD_BLOCK>
<XPOS>0.5</XPOS>
<YPOS>0.0</YPOS>
<WIDTH>1</WIDTH>
<HEIGHT>1</HEIGHT>
<FONT_FILENAME>Font_6.fnt</FONT_FILENAME>
<FONT_NAME>Verdana_20_WR75_TBS0</FONT_NAME>
<STARTING_FIELD>1</STARTING_FIELD>
<RECORD_BLOCK_LINES>4</RECORD_BLOCK_LINES>
</RECORD_BLOCK>
```

<XPOS> Object top left corner X Position on the page defined in inches or millimeters depending on the **<UNITS>** tag. Object must be within bounds of page definition. Allow for object width when setting.

<YPOS> Object top left corner Y Position on the page defined in inches or millimeters depending on the **<UNITS>** tag. Object must be within bounds of page definition. Allow for object width when setting.

<WIDTH>	Object width defined in inches or millimeters depending on the <UNITS> tag. Whole object must fit within bounds of page definition.
<HEIGHT>	Object height defined in inches or millimeters depending on the <UNITS> tag. Whole object must fit within bounds of page definition.
<FONT_FILENAME>	Filename of font used for this print object. Must be included whether font is embedded in configuration file or prestored in the imager. Filename adheres to DOS 8.3 standard with "*.fnt" as the font file extension. Not needed if print object does not contain human readable data.
<FONT_NAME>	Up to 64 characters of name data. Must match name data found in font file exactly. Case sensitive.
<ROTATION>	(Optional) Specifies rotation of print object. Valid parameters: 0 , 90, 180, 270. Default: 0
<PRINT_DENSITY>	(Optional) Specifies print density for individual print objects. Range 0 to 15. (6.25% decrease per increment). Default = 0; (100%)
<STARTING_FIELD>	Starting field number. 1 based field number within a print record which will be used as the first line.
<RECORD_BLOCK_LINES>	Number of lines in record block.

8.5.3 Bitmap Block

<BITMAP>	(Optional)	Beginning of Bitmap Block configuration data. Object may define a single, fixed bitmap or multiple bitmaps selected by matching incoming data strings.
----------	------------	--------------------------------------------------------------------------------------------------------------------------------------------------------

Example:

```
<BITMAP_BLOCK>
  <XPOS>0.0</XPOS>
  <YPOS>0.0</YPOS>
  <WIDTH>1.5</WIDTH>
  <HEIGHT>1.5</HEIGHT>
  <ROTATION>0</ROTATION>
  <BITMAP_DEF>
    <BITMAP_DEF_STATIC>0</BITMAP_DEF_STATIC>
    <BITMAP_DEF_FILENAME>J01_000.bmp</BITMAP_DEF_FILENAME>
    <BITMAP_DEF_NAME>J01_000_{02A86577}</BITMAP_DEF_NAME>
    <BITMAP_DEF_SRC_LINE>1</BITMAP_DEF_SRC_LINE>
    <BITMAP_DEF_MATCH_STRING>1</BITMAP_DEF_MATCH_STRING>
  </BITMAP_DEF>
  <BITMAP_DEF>
    <BITMAP_DEF_STATIC>0</BITMAP_DEF_STATIC>
    <BITMAP_DEF_FILENAME>J01_001.bmp</BITMAP_DEF_FILENAME>
    <BITMAP_DEF_NAME>J01_001_{8F34F94C}</BITMAP_DEF_NAME>
    <BITMAP_DEF_SRC_LINE>2</BITMAP_DEF_SRC_LINE>
    <BITMAP_DEF_MATCH_STRING>2</BITMAP_DEF_MATCH_STRING>
  </BITMAP_DEF>
</BITMAP_BLOCK>
```


<XPOS>	Object top left corner X Position on the page defined in inches or millimeters depending on the <UNITS> tag. Object must be within bounds of page definition. Allow for object width when setting.
<YPOS>	Object top left corner Y Position on the page defined in inches or millimeters depending on the <UNITS> tag. Object must be within bounds of page definition. Allow for object width when setting.
<WIDTH>	Object width defined in inches or millimeters depending on the <UNITS> tag. Whole object must fit within bounds of page definition.
<HEIGHT>	Object height defined in inches or millimeters depending on the <UNITS> tag. Whole object must fit within bounds of page definition.
<FONT_FILENAME>	Filename of font used for this print object. Must be included whether font is embedded in configuration file or prestored in the imager. Filename adheres to DOS 8.3 standard with "*.fnt" as the font file extension. Not needed if print object does not contain human readable data.
<FONT_NAME>	Up to 64 characters of name data. Must match name data found in font file exactly. Case sensitive.
<ROTATION>	(Optional) Specifies rotation of print object. Valid parameters: 0 , 90, 180, 270. Default: 0
<PRINT_DENSITY>	(Optional) Specifies print density for individual print objects. Range 0 to 15. (6.25% decrease per increment). Default = 0; (100%)
<BITMAP_DEF_STATIC>	(Optional) Boolean flag, when set, indicates selection of bitmap is made using static data. Default = false
<BITMAP_DEF_SRC_LINE>	(Optional) 1 based field number to search for matching data. If bitmap data is supplied dynamically, external, this parameter identifies the line number in the incoming record from which the binary image data may be found. <i>Not used if only one bitmap is supplied.</i>
<BITMAP_DEF>	Definition of a single bitmap contained in bitmap block. Multiple bitmap can be defined within one bitmap object using multiple <BITMAP_DEF> tags. All BITMAP_DEF parameters should be enclosed with this tag.
<BITMAP_DEF_FILENAME>	(Optional) Bitmap filename. Must be in DOS 8.3 format. If bitmap data is supplied dynamically, external, leave this parameter blank. Default = blank
<BITMAP_DEF_NAME>	(Optional) Bitmap name. Can be up to 64 characters in length. Must match name found in bitmap file exactly, case sensitive.
<BITMAP_DEF_MATCH_STRING>	(Optional) Match string, when data from incoming field matches this string, print this bitmap.

8.6 Embedding Fonts and Bitmaps

Fonts and bitmaps can be embedded into the XML job definition file using <FONTFILE> tag for font data and <BITMAPFILE> tag for bitmap data. The tags have to be embedded after the <RIP_CONFIG> and before the closing </JOB_CONFIG> tag. Font and bitmap files are created using OnBoardRIP GUI. The GUI can load the files directly to the imager or save them to a text file using Base 64 encoding. Refer to *IJM019A OnBoardRIP GUI Operations Manual.pdf* for detail on how to create and load the files. The contents of the text file created by the OBR GUI can then be copied and pasted into the XML job definition file. The name of the text file generated by the GUI includes the 8.3 and the user friendly font or bitmap file names.

Example: In "Arial_12_WR100_TBS0_Font_1.fnt.txt" text file name "Font_1.fnt" is the 8.3 font file name which has to be referenced in <FONT_FILENAME> and "Arial_12_WR100_TBS0" is the user friendly job name that has to be referenced in <FONT_NAME> tag. The user friendly name includes Font Name (Arial), Point Size (12), Width Ratio (100%) and Top-Bottom-Spacing (0%)

Below is a sample XML job definition that shows how to embed and reference font and bitmap files.

```
<JOB_RECORD>
  <JOB_FILENAME>Sample.job</JOB_FILENAME>
  <JOB_NAME>Example with embedded font and bitmap</JOB_NAME>
  <RASTER_BUFFER_CFG>
    <NUM_BUFFS>4</NUM_BUFFS>
    <PAGE_HEIGHT>2</PAGE_HEIGHT>
    <PAGE_LENGTH>4.0</PAGE_LENGTH>
    <HORIZONTAL_DPI>300</HORIZONTAL_DPI>
    <RASTER_BUFFER_PREFILL>0</RASTER_BUFFER_PREFILL>
  </RASTER_BUFFER_CFG>
  <RIP_CONFIG>
    <FIELDBLOCK>
      <XPOS>0.0</XPOS>
      <YPOS>0.0</YPOS>
      <WIDTH>4</WIDTH>
      <HEIGHT>2</HEIGHT>
      <FONT_FILENAME>Font_1.fnt</FONT_FILENAME>
      <FONT_NAME>Arial_WR100_TBS0</FONT_NAME>
      <LINE_DEF>
        <FIXED_TEXT>Text Sample</FIXED_TEXT>
      </LINE_DEF>
    </FIELDBLOCK>

    <BITMAP_BLOCK>
      <XPOS>0.5</XPOS>
      <YPOS>0.0</YPOS>
      <WIDTH>3</WIDTH>
      <HEIGHT>1</HEIGHT>
      <ROTATION>0</ROTATION>
      <BITMAP_DEF>
        <BITMAP_DEF_FILENAME>logo.bmp</BITMAP_DEF_FILENAME>
        <BITMAP_DEF_NAME>inc.jet Logo</BITMAP_DEF_NAME>
      </BITMAP_DEF>
    </BITMAP_BLOCK>
  </RIP_CONFIG>
</JOB_RECORD>
```



```
</BITMAP_BLOCK>
</RIP_CONFIG>

<COMMENT> Bitmap_FileName: logo.bmp Font_Name: inc.jet Logo </COMMENT>
<BITMAPFILE> AAAAAWxvZ28uYm1wAAAAAaW5jLmpldCBMb2dvAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABQAAAFZAAAA9AAAAAAAAAAAAAAAA
AAAAAWxvZ28uYm1wAAAAAaW5jLm... (Partial Bitmap Data Shown)
</BITMAPFILE>

<COMMENT>Font_FileName: Font_1.fnt Font_Name: Arial_WR100_TBS0</COMMENT>
<FONTFILE> AAAAUZvbnRfNi5mbnQAAAAVmVyZGFuYV8yMF9XUjc1X1RCUzAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUZvbnRfNi5mbnQAAAAVmVyZGFuYV8yMF9XUjc1X1RCUzAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA... (Partial Font Data Shown)
</FONTFILE>

</JOB_RECORD>
```

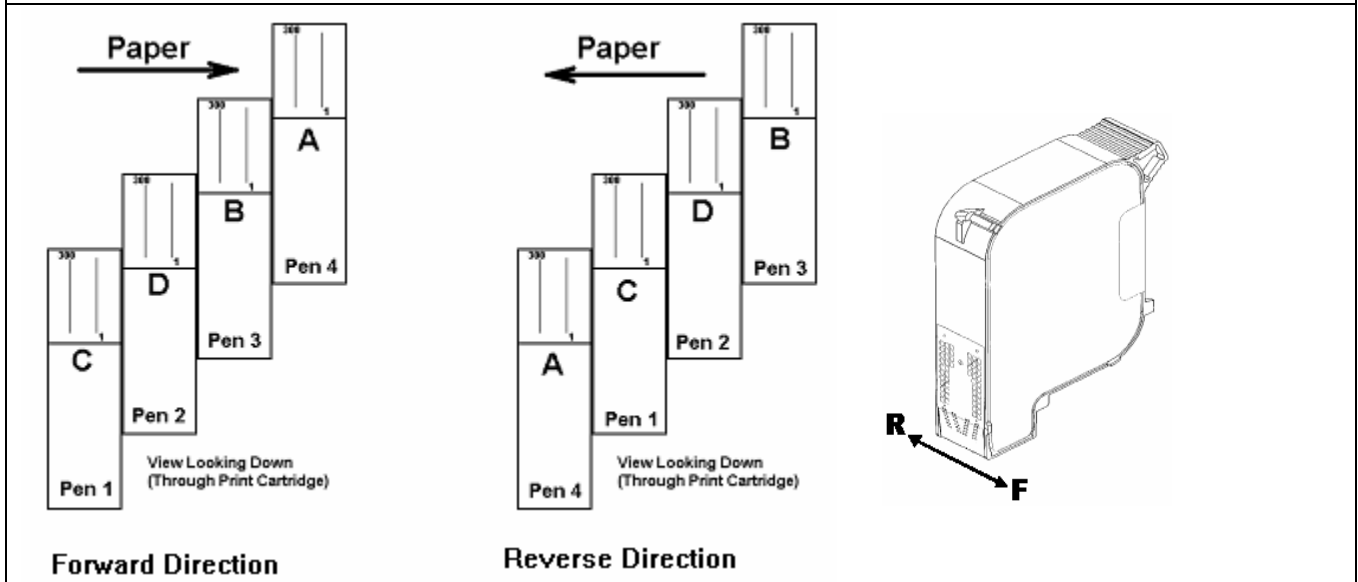
Appendix A: Explanation of Stitching Table Parameters

The Generate Stitching Table Command is a Binary Protocol command that is defined in Section 5.0 of this document. To create a set of stitching tables you can either create a Generate Stitching Tables(0x99) command and send it to the Imager over the Standard(port 10000) or Auxilliary(port 10002) Ethernet ports or use the OnBoardRIP GUI to set the fields and create the command for you. The OnBoardRIP GUI page that creates the fields for this command is shown below. It can be found by clicking Options->Advanced Configurations... and clicking on the Generate Stitching Table tab. The Use Parameterized Stitching Table checkbox must be checked so that when the hardware configuration is downloaded, it will also send a Generate Stitching Table command with the settings made on this window. If you wish to save the settings across uses of the GUI, press Save to File button. To generate the command and send it to the Imager first close the Advanced Configuration window. Then, click Options->Imager Configuration and select the Stand Alone Operation tab. Press the Store Hardware Configuration button in the lower left corner of the screen.

Pen A: (L->R Pen4, R->L Pen4)	Pen B: (L->R Pen3, R->L Pen1)	Pen C: (L->R Pen1, R->L Pen3)	Pen D: (L->R Pen2, R->L Pen2)
Print Direction <input checked="" type="radio"/> Forward <input type="radio"/> Reverse	Print Direction <input checked="" type="radio"/> Forward <input type="radio"/> Reverse	Print Direction <input checked="" type="radio"/> Forward <input type="radio"/> Reverse	Print Direction <input checked="" type="radio"/> Forward <input type="radio"/> Reverse
Leading Jets To Disable: 0	Leading Jets To Disable: 0	Leading Jets To Disable: 0	Leading Jets To Disable: 0
Trailing Jets To Disable: 0	Trailing Jets To Disable: 0	Trailing Jets To Disable: 0	Trailing Jets To Disable: 0
Vertical Image Resolution: 300	Vertical Image Resolution: 300	Vertical Image Resolution: 300	Vertical Image Resolution: 600
First Image Pixel: 0	First Image Pixel: 150	First Image Pixel: 300	First Image Pixel: 450
First Image Sub Pixel: 0	First Image Sub Pixel: 0	First Image Sub Pixel: 0	First Image Sub Pixel: 0
Number of Columns to use <input type="radio"/> One <input checked="" type="radio"/> Two	Number of Columns to use <input type="radio"/> One <input checked="" type="radio"/> Two	Number of Columns to use <input type="radio"/> One <input checked="" type="radio"/> Two	Number of Columns to use <input type="radio"/> One <input checked="" type="radio"/> Two
Column Select <input type="radio"/> Reverse <input checked="" type="radio"/> Normal	Column Select <input type="radio"/> Reverse <input checked="" type="radio"/> Normal	Column Select <input type="radio"/> Reverse <input checked="" type="radio"/> Normal	Column Select <input type="radio"/> Reverse <input checked="" type="radio"/> Normal
Pixel Ordering <input checked="" type="radio"/> Normal <input type="radio"/> Reverse	Pixel Ordering <input checked="" type="radio"/> Normal <input type="radio"/> Reverse	Pixel Ordering <input checked="" type="radio"/> Normal <input type="radio"/> Reverse	Pixel Ordering <input checked="" type="radio"/> Normal <input type="radio"/> Reverse
<input type="checkbox"/> Enable Double Speed Printing	<input type="checkbox"/> Enable Double Speed Printing	<input type="checkbox"/> Enable Double Speed Printing	<input type="checkbox"/> Enable Double Speed Printing

The following is a description of each of the fields required for the command.

Print Direction - Defines the direction of paper travel across the pen. F (Forward) is from Column 1 to 2, R (Reversed) is from Column 2 to 1. Diagram below shows column ordering as well as jet.engine AC/MC pen ordering. In Print Controller system Pens can be mapped in any order.



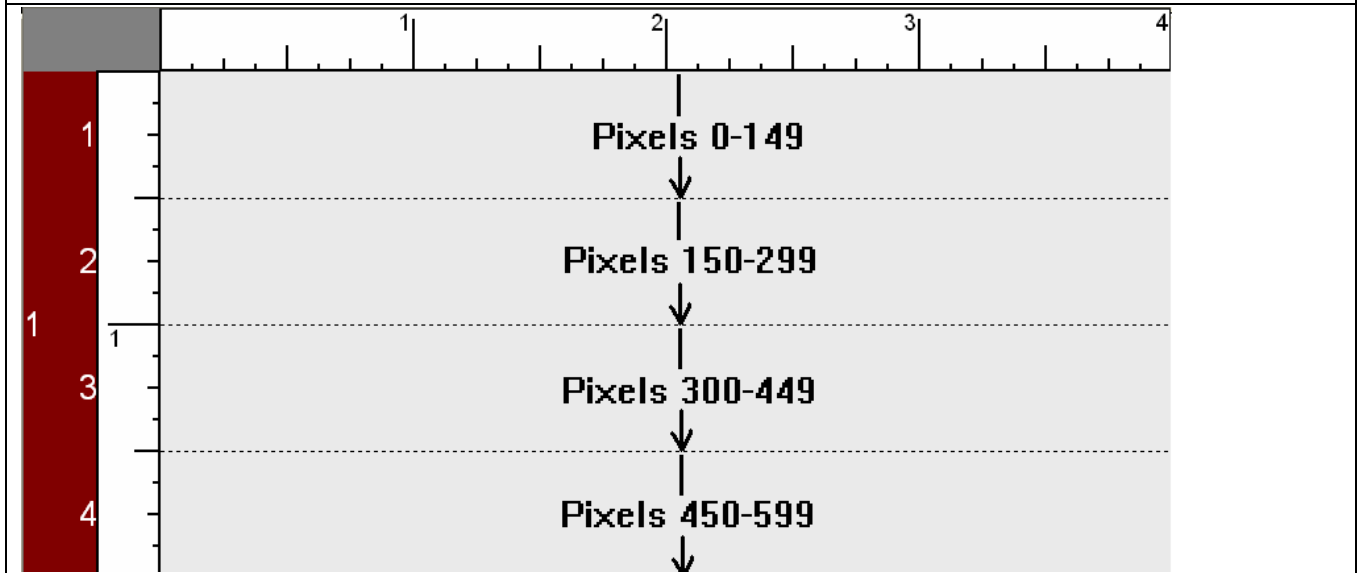
Leading Jets to Disable - Defines the number of nozzles to be disabled on the Leading side of the pen. The first Pen Pixel will be printed on the first nozzle that is not disabled. Note: Disabling jets reduces total print area of the pen. (1 jet = 1/600")

Trailing Jets to Disable - Defines the number of nozzles to be disabled on the Trailing side of the pen. The last Pen Pixel will be printed on the last nozzle that is not disabled.

Note: Disabling jets reduces total print area of the pen. (1 jet = 1/600")

Vertical Image Resolution - Defines the vertical resolution of the input image. Any value is allowed and the image will be "sampled" at the specified resolution. The best printing is obtained if the resolution is kept at increments divisible by 600 (dpi). Recommended settings for OnBoardRIP operation is 300 dpi)

First Image Pixel - Defines the first image pixel to print. This Image Pixel will be assigned to the first nozzle that is enabled to print. In OnBoardRIP operation each pen is assigned 150 pixels. If no jets are disabled Pen 1 = 0, Pen 2 = 150, Pen 3 = 300, Pen 4 = 450. If overlap jets are disabled First Image Pixel has to be adjusted accordingly not to lose any sections of the image.

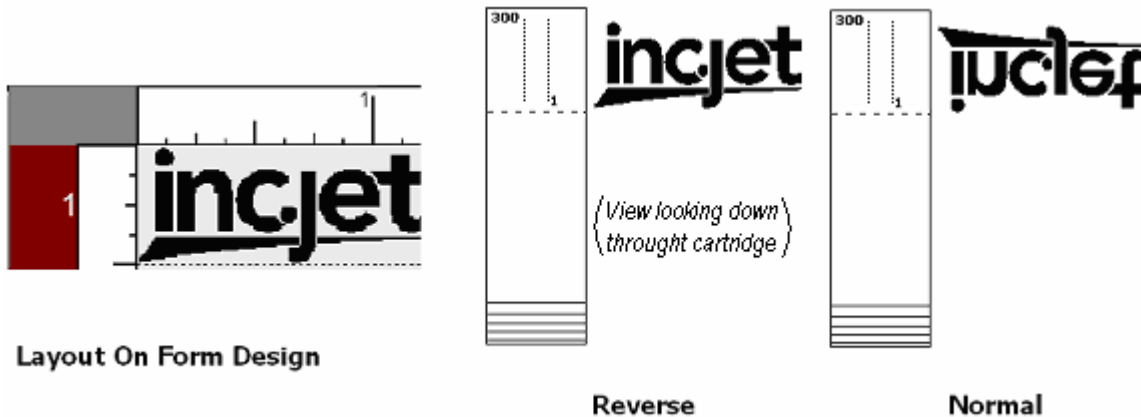


First Image Sub Pixel - Defines Sub Pixel to start with on the first nozzle. When printing at resolutions less than 600 dpi, the same pixel is printed multiple times. This setting allows the definition of the starting Sub Pixel to allow stitching between pens.

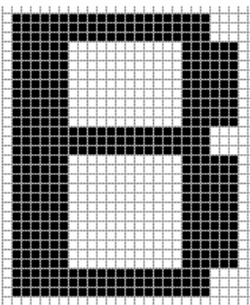
Number of Columns to Use – Defines the number of nozzle columns to be used. In OnBoardRIP using two columns is equivalent to printing at 600 dpi vertical resolution.

Column Select – Swaps data on the two columns. This parameter is used for advanced configuration and should always be set to "Normal" for standard operation.

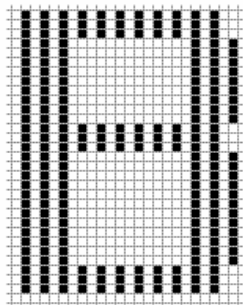
Pixel Ordering - Defines the assignment order of cartridge nozzles to pixels in each vertical column of the image. Normal assigns the first pixel in an image column to the first available nozzle. Reverse does an image swap, assigning the first pixel in each image column to the last available nozzle.



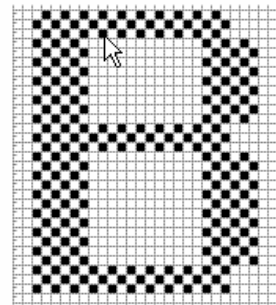
Enable Double Speed Printing - Enables the hardware to print at double the maximum speed allowed by each horizontal resolution. When printing with double speed enabled the hardware skips every other image pixel allowing it to print at double the speed and stay within its maximum 12 KHz firing frequency. Final image placed on the paper is 150 dpi but with edge acuity of a 300 DPI image.



300 DPI Print



150 DPI Print



**300 DPI Print with
Double Speed Enabled**

Appendix B: Barcode Parameters Application Notes

Code 39 Barcode Parameters

- `<BARCODE_PARAMETER_1>` – Wide to narrow bar width multiplier. Default = 3.0. Range from 2.0 to 3.0 when magnification factor is greater than 50%. If magnification factor is less than 50%, range for this parameter is 2.0 to 2.2. When entering the parameter into the XML file the decimal point is dropped. So a multiplier of 2.6 is entered as
`<BARCODE_PARAMETER_1>26</BARCODE_PARAMETER_1>`
- `<BARCODE_PARAMETER_2>` – Maximum number of data characters that will be encoded into the barcode. Add +1 character when including a checksum. This parameter may be used to fine tune calculated magnification factor or calculated window width by adjusting the maximum number of bar characters anticipated. If a data string arrives which requires a greater number of bar characters than specified by this parameter, a rendering error will be reported.
Default = 64
- `<BARCODE_PARAMETER_3>` – Unused.
- `<BARCODE_PARAMETER_4>` – Quiet Zone Disable. Default = 0 (Enabled) If this parameter is non zero, the quiet zone will be dropped from the barcode. Quiet Zone will be 0.1 inches or 10 * Module Width, whichever is greater.

Code 93 Barcode Parameters

- `<BARCODE_PARAMETER_1>` – Unused.
- `<BARCODE_PARAMETER_2>` – Maximum number of data characters that will be encoded into the barcode. Add +1 character when including a checksum. This parameter may be used to fine tune calculated magnification factor or calculated window width by adjusting the maximum number of bar characters anticipated. If a data string arrives which requires a greater number of bar characters than specified by this parameter, a rendering error will be reported.
Default = 64
- `<BARCODE_PARAMETER_3>` – Unused.
- `<BARCODE_PARAMETER_4>` – Quiet Zone Disable. Default = 0 (Enabled) If this parameter is non zero, the quiet zone will be dropped from the barcode. Quiet Zone will be 0.1 inches or 10 * Module Width, whichever is greater.

Code 2 of 5 Interleaved Barcode Parameters

- <BARCODE_PARAMETER_1>** – Wide to narrow bar width multiplier. Default = 3.0. Range from 2.0 to 3.0 when magnification factor is greater than 50%. If magnification factor is less than 50%, range for this parameter is 2.0 to 2.2. When entering the parameter into the XML file the decimal point is dropped. So a multiplier of 2.6 is entered as
<BARCODE_PARAMETER_1>26</BARCODE_PARAMETER_1>
- <BARCODE_PARAMETER_2>** – Maximum number of data characters that will be encoded into the barcode. Add +1 character when including a checksum. Number of characters must be an even number. Firmware will add 1 to make this number even. This parameter may be used to fine tune calculated magnification factor or calculated window width by adjusting the maximum number of bar characters anticipated. If a data string arrives which requires a greater number of bar characters than specified by this parameter, a rendering error will be reported.
Default = 64
- <BARCODE_PARAMETER_3>** – Unused.
- <BARCODE_PARAMETER_4>** – Quiet Zone Disable. Default = 0 (Enabled) If this parameter is non zero, the quiet zone will be dropped from the barcode. Quiet zone will be 0.25 inches or 12 * Module Width, whichever is greater.

Code/EAN 128 (A,B,C) Barcode Parameters

Note: Code 128 barcode uses compression to reduce the number of bar to represent the data. Number of characters refers to number of barcode characters as firmware can not anticipate the level of compression that will be applied to the barcode. If a data string arrives which requires a greater number of bar characters than specified by this parameter, a rendering error will be reported.

- <BARCODE_PARAMETER_1>** – Unused.
- <BARCODE_PARAMETER_2>** – Maximum number of data characters that will be encoded into the barcode. Add +1 character when including a checksum. This parameter may be used to fine tune calculated magnification factor or calculated window width by adjusting the maximum number of bar characters anticipated. When Code 128C is specified host must provide even number of characters.
Default = 150.
- <BARCODE_PARAMETER_3>** – Unused.
- <BARCODE_PARAMETER_4>** – Quiet Zone Disable. Default = 0 (Enabled) If this parameter is non zero, the quiet zone will be dropped from the barcode. Quiet zone will be 0.25 inches or 12 * Module Width, whichever is greater.



IMB Barcode Parameters

Note: Specifications require 0.040 inches minimal clear space above and below barcode. This clear space is built into the firmware generated bars. Window height calculated by firmware includes this white space. Number of bars is hardcoded to 65.

- <BARCODE_PARAMETER_1> – Gap specification between bars.
 - 0 - nominal (7 pixels @ 300 dpi) 0.023 inches
 - 1 - narrow (6 pixels @ 300 dpi) 0.02 inches
 - 2 - wide (8 pixels @ 300 dpi) 0.026 inches
- <BARCODE_PARAMETER_2> – Encoding Select
 - 0 - Host supplies bars data. Host has to specify which bars should be printed using following letters.
 - F – Full Bar
 - T – Tracking Bar
 - A – Ascending Bar
 - D – Descending Bar
 - 1 - Host supplies ASCII data (not implemented)
- <BARCODE_PARAMETER_3> – Bar Height
 - 0 - short (long bar = 42 pixels @ 300 dpi) 0.140 inches
 - 1 - nominal (long bar = 56 pixels @ 300 dpi) 0.186 inches
 - 2 - tall (long bar = 66 pixels @ 300 dpi) 0.220 inches
- <BARCODE_PARAMETER_4> – Unused

Postnet Barcode Parameters

Note: Specifications require 0.040 inches minimal clear space above and below barcode. This clear space is built into the firmware generated font. Window height calculated by firmware includes this white space.

- <BARCODE_PARAMETER_1> – Gap specification between bars.
 - 0 - nominal (7 pixels @ 300 dpi) 0.023 inches
 - 1 - narrow (6 pixels @ 300 dpi) 0.02 inches
 - 2 - wide (8 pixels @ 300 dpi) 0.026 inches
- <BARCODE_PARAMETER_2> – Encoding Select
 - 0 - Host supplies bars. Host has to specify which bars should be printed using following letters.
 - F – Full Bar
 - D – Descending Bar
 - 1 - Host supplies ASCII data (not implemented)
- <BARCODE_PARAMETER_3> – Number of bars that will be used in the barcodes. Used to calculate barcode size.
- <BARCODE_PARAMETER_4> – Unused



UPC-A/EAN 13 Barcode Parameters

<BARCODE_PARAMETER_1> – Unused
<BARCODE_PARAMETER_2> – Unused
<BARCODE_PARAMETER_3> – Unused
<BARCODE_PARAMETER_4> – Unused

ITF-14 Barcode Parameters

<BARCODE_PARAMETER_1> – Unused
<BARCODE_PARAMETER_2> – Unused
<BARCODE_PARAMETER_3> – Unused
<BARCODE_PARAMETER_4> – Unused