# **INNOVATIVE TECHNOLOGY LTD**

**Protocol Manual** 

SSP

NV9USB, NV10USB, NV11, SMART HOPPER, NV200, SMART PAYOUT, BV20, BV50, BV100, SMART SYSTEM, SMART TICKET, COUPON PRINTER, TEBS, NV150, FLATBED PRINTER, NV12

version GA138\_2\_2\_649A

# **Contents**

Introduction General Description Hardware layer Transport Layer Encryption Layer Encryption Keys Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Table NV10USB Command Table NV10USB Command Table NV11 Command Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Levent Table SMART HOPPER Levent Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Command Table BV20 Command Table BV50 Event Table SWART SYSTEM Command Table SMART SYSTEM Command Table SMART SYSTEM Command Table SMART TICKET Command Table SMART TICKET Command Table SMART TICKET Event Table TEBS Event Table SMART TICKET Event Table SMART TICKET Event Table SMART TICKET Event Table TEBS Event Table SMART TICKET Event Table TEBS Event Table SMART TICKET Event Table TEBS Event Table	Descriptions	
General Description Hardware layer Transport Layer Encryption Layer Encryption Keys Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Command Table SMART HOPPER Command Table SMART HOPPER Command Table SMART HOPPER Table NV200 Command Table SMART PAYOUT Command Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Command Table SW20 Event Table SW30 Event Table SW40 Event Table SW50 Event Table SW50 Event Table SW50 Command Table SW50 Event Table SW50 Event Table SW50 Command Table SW50 Event Table EV50 Command Table EV50 Command Table EV50 Command Table SW50 Event Table SW50 Event Table EV50 Command Table EV50 Command Table EV50 Command Table EV50 Command Table EV50 Event Table EV50 Command Table EV50 Command Table EV50 Command Table EV50 Event Table EV50 Command Table EV50 Event Table EV50 Command Table EV50 Command Table EV50 Event Table EV50 Command Table EV50 Event Table EV50 Command Table EV50 Event Table	-	
Hardware layer Transport Layer Encryption Layer Encryption Keys Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART PAYOUT Command Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Command Table BV20 Command Table SW20 Event Table SW30 Command Table SW40 Command Table SMART PAYOUT Event Table SW50 Event Table SW50 Event Table SW50 Command Table SW50 Event Table SW50 Command Table SW50 Command Table SW50 Feent Table SW50 Command Table SW50 Feent Table SW50 Command Table SW50 Feent Table SW50 Feent Table SW50 Command Table SW50 Feent Table SW60 Feent Table SW		
Transport Layer Encryption Layer Encryption Keys Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Command Table SWART PAYOUT Event Table BV20 Command Table BV20 Command Table SW50 Event Table SWART PAYOUT Event Table SWART PAYOUT Event Table SWART PAYOUT Event Table SW50 Event Table SW60 Event Table SW700 Event Table SW700 Event Table SW700 Event Table EV50 Event Table SW700 Event Table EV50 Event Table SW60 Event Table EV60 Event Table EV700		
Encryption Layer Encryption Keys Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV11 Command Table NV11 Command Table NV11 Command Table SMART HOPPER Command Table SMART HOPPER Event Table SMART PAYOUT Command Table SMART PAYOUT Command Table SMART PAYOUT Tommand Table BV20 Command Table BV20 Command Table BV20 Command Table SMART PAYOUT Event Table SMART PAYOUT Event Table BV20 Command Table BV20 Event Table SMART SYSTEM Event Table SMART SYSTEM Command Table SMART SYSTEM Command Table SMART TICKET Command Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table TEBS Command Table FLATBED PRINTER Event Table	·	
Encryption Keys Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Command Table SW20 Command Table BV20 Command Table BV20 Command Table SMART PAYOUT Event Table BV30 Command Table SMART PAYOUT Event Table BV50 Event Table SMART SYSTEM Event Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Command Table SMART TICKET Command Table SMART TICKET Event Table EVAD COUPON PRINTER Event Table EVSD Command Table SMART TICKET Event Table EVSD Event Table SMART TICKET Event Table EVSD COMMAND Table SMART TICKET Event Table EVSD COMMAND Table EVSD COMMAND Table SMART TICKET Event Table EVSD COMMAND Table EVSD Event Table EVSD COMMAND Table EVSD Event		
Generic Commands and Responses Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table SMART PAYOUT Event Table SMART PAYOUT Event Table BV20 Command Table BV20 Command Table BV20 Event Table SWART PAYOUT Event Table SWART PAYOUT Event Table SWART PAYOUT Event Table SWART Table SWART Table SWART Table SWART Table SWART Table SWART PAYOUT Event Table SWART PAYOUT Event Table BV20 Event Table SWART Table SWART Table SWART SYSTEM Command Table SWART SYSTEM Command Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Event Table TEBS Command Table FLATBED PRINTER Event Table VV12 Event Table Commands Commands		
Protocol Versions Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table NV200 Event Table BV20 Event Table BV20 Command Table BV20 Command Table BV20 Event Table SMART PAYOUT Event Table BV20 Event Table BV20 Event Table BV30 Event Table SWART PAYOUT Event Table BV40 Event Table BV50 Command Table BV50 Event Table BV100 Event Table BV100 Event Table BV100 Event Table BV100 Event Table BV50 Event Table BV100 Event Table EVART SYSTEM Event Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Event Table TEBS Event Table FLATBED PRINTER Event Table FLATBED PRINTER Event Table FLATBED PRINTER Event Table FLATBED PRINTER Event Table VV12 Event Table FLATBED PRINTER Event Table VV12 Event Table VC00mands		
Banknote Validator Reject Codes SMART Ticket SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV50 Event Table BV50 Command Table BV50 Event Table SW50 Event Table SWART PAYOUT Event Table SWART PAYOUT Event Table BV50 Command Table SW50 Event Table SW50 Event Table BV50 Event Event Table BV50 Event	•	
SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table NV200 Command Table NV200 Event Table BV20 Event Table BV20 Event Table BV30 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Event Table SMART TABLE SMART HOPPER Event Table SMART PAYOUT Event Table BV30 Event Table BV40 Event Table BV50 Command Table BV50 Event Table BV50 Event Table BV50 Event Table BV50 Event Table BV100 Fernt Table BV100 Fernt Table BV50 Event Table BV50 Event Table BV50 Event Table BV50 Event Table BV100 Fernt Table		
SMART Hopper Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV9USB Event Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Command Table BV20 Event Table BV50 Command Table BV50 Command Table BV50 Event Table BV50 Command Table BV50 Event Table BV100 Fernt Table BV50 Event Table BV60 Event Table EV60 Event Table EV710 Event Table	Reject Codes	
Coupon Printer SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV9USB Event Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Event Table BV20 Event Table BV50 Command Table BV50 Event Table BV100 Fernt Table BV50 Event Table EVAMART SYSTEM Event Table EVAMART SYSTEM Event Table EVAMART TICKET Event Table COUPON PRINTER Event Table TEBS Command Table TEBS Event Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Event Table Commands	SMART Ticket	
SMART Payout Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV9USB Event Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Event Table BV20 Command Table BV20 Event Table BV20 Event Table SMART PAYOUT Event Table BV20 Event Table BV20 Event Table BV30 Event Table BV50 Command Table BV50 Event Table BV50 Event Table BV50 Event Table BV100 Command Table BV100 Event Table EV100 Event Table BV100 Event Table BV100 Event Table EV100 Event Table BV100 Event Table EV100 Event Table EV100 Event Table SMART SYSTEM Event Table SMART SYSTEM Event Table EV100 E	SMART Hopper	
Smart System Note Float (NV11) TEBS NVR-280 (NV12) Flatbed Printer (FBF-166) Command/Event Tables NV9USB Command Table NV9USB Event Table NV10USB Event Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Event Table BV20 Event Table BV50 Command Table BV50 Command Table BV750 Command Table SMART PAYOUT Event Table BV760 Event Table BV770 Event Table EMART SYSTEM Command Table SMART SYSTEM Event Table SMART SYSTEM Event Table EMART TICKET Event Table COUPON PRINTER Command Table TEBS Command Table TEBS Event Table	Coupon Printer	
Note Float (NV11) TEBS  NVR-280 (NV12) Flatbed Printer (FBF-166)  Command/Event Tables  NV9USB Command Table  NV9USB Event Table  NV10USB Event Table  NV10USB Event Table  NV11 Command Table  NV11 Event Table  SMART HOPPER Command Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Event Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV50 Event Table  BV100 Command Table  BV50 Event Table  BV100 Ferent Table  BV100 Ferent Table  BV100 Event Table  BV100 Ferent Table  BV100 Ferent Table  BV100 Ferent Table  BV100 Ferent Table  SMART PYSTEM Command Table  SMART SYSTEM Command Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  TEBS Command Table  TEBS Command Table  TEBS Ferent Table  TEBS Command Table  NV150 Command Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Commands  Commands	SMART Payout	
TEBS  NVR-280 (NV12)  Flatbed Printer (FBF-166)  Command / Event Tables  NV9USB Command Table  NV9USB Event Table  NV10USB Event Table  NV10USB Event Table  NV11 Command Table  NV11 Event Table  SMART HOPPER Command Table  SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Event Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART IICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  TEBS Command Table  TEBS Command Table  TEBS Peent Table  NV150 Command Table  RV150 Command Table  FLATBED PRINTER Event Table  RV12 Command Table  NV12 Command Table  COCOMMAND  NV12 Event Table  COCOMMAND  COCOMMA	Smart System	
NVR-280 (NV12) Flatbed Printer (FBF-166)  Command/Event Tables  NV9USB Command Table  NV9USB Event Table  NV10USB Event Table  NV10USB Event Table  NV11 Command Table  NV11 Event Table  SMART HOPPER Command Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV20 Event Table  BV50 Command Table  BV50 Command Table  BV50 Event Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Event Table  SMART TICKET Event Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Command Table  TEBS Command Table  NV150 Event Table  FLATBED PRINTER Event Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands  Commands	Note Float (NV11)	
Flatbed Printer (FBF-166)  Command / Event Tables  NY9USB Command Table  NY10USB Event Table  NV10USB Event Table  NV11 Command Table  NV11 Event Table  NV11 Event Table  SMART HOPPER Command Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  SV20 Event Table  BV20 Event Table  BV50 Command Table  BV50 Command Table  BV50 Event Table  BV50 Event Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Command Table  TEBS Event Table  FLATBED PRINTER Event Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands  Commands	TEBS	
NY9USB Command Table NY9USB Event Table NV10USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table NV11 Event Table SMART HOPPER Command Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Event Table BV20 Event Table BV20 Event Table BV50 Command Table BV50 Command Table BV50 Event Table BV50 Event Table BV50 Event Table BV100 Event Table BV100 First Event Table BV100 First Event Table BV100 Event Table BV100 Event Table BV100 Event Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table COUPON PRINTER Command Table SMART TICKET Event Table COUPON PRINTER Event Table TEBS Event Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV15 Command Table FLATBED PRINTER Event Table NV15 Command Table NV15 Command Table FLATBED PRINTER Event Table NV15 Command Table NV15 Command Table FLATBED PRINTER Event Table NV15 Command Table NV15 Event Table FLATBED PRINTER Event Table NV15 Event Table NV15 Event Table FLATBED PRINTER Event Table NV15 Event Table NV15 Event Table NV15 Event Table NV15 Event Table ENV15 Event Table NV15 Event Table	NVR-280 (NV12)	
NV9USB Command Table NV9USB Event Table NV10USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Command Table BV20 Event Table BV20 Event Table BV50 Command Table BV50 Event Table BV50 Event Table BV100 Command Table BV100 Event Table BV100 Fevent Table BV100 Event Table BV100 Fevent Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table TEBS Command Table TEBS Command Table TEBS Event Table TEBS Event Table TEBS Event Table NV150 Event Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Event Table NV150 Event Table FLATBED PRINTER Event Table NV150 Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Event Table NV150 Event Table NV150 Event Table FLATBED PRINTER Event Table NV150 Event Table	Flatbed Printer (FBF-166)	
NV9USB Event Table NV10USB Command Table NV10USB Event Table NV11 Command Table NV11 Event Table SMART HOPPER Command Table SMART HOPPER Event Table NV200 Command Table NV200 Event Table SMART PAYOUT Command Table SMART PAYOUT Event Table BV20 Command Table BV20 Event Table BV50 Command Table BV50 Event Table BV50 Event Table BV100 Event Table BV100 Event Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table COUPON PRINTER Command Table TEBS Command Table TEBS Command Table TEBS Event Table TEBS Event Table TEBS Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Command Table NV12 Event Table TEATBED PRINTER Event Table TEATBED PRINTER Event Table NV12 Event Table TEBS Event Table TEATBED PRINTER Event Table	Command/Event Tables	
NV10USB Command Table  NV10USB Event Table  NV11 Command Table  NV11 Event Table  SMART HOPPER Command Table  SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Command Table  BV50 Command Table  BV50 Command Table  BV100 Command Table  BV100 Event Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  TEBS Command Table  NV150 Command Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	NV9USB Command Table	
NV10USB Event Table  NV11 Command Table  NV11 Event Table  SMART HOPPER Command Table  SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV50 Command Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Command Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Event Table  TEBS Command Table  NV150 Command Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	NV9USB Event Table	
NV11 Command Table  NV11 Event Table  SMART HOPPER Command Table  SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV100 Command Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Command Table  SMART TICKET Command Table  SMART TICKET Command Table  COUPON PRINTER Event Table  COUPON PRINTER Event Table  TEBS Command Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	NV10USB Command Table	
NV11 Event Table  SMART HOPPER Command Table  SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Event Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Event Table  TEBS Command Table  NV150 Command Table  NV150 Command Table  FLATBED PRINTER Command Table  NV12 Command Table  NV12 Command Table  NV12 Command Table  Commands  Commands	NV10USB Event Table	
SMART HOPPER Command Table  SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Command Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	NV11 Command Table	
SMART HOPPER Event Table  NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	NV11 Event Table	
NV200 Command Table  NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Command Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Command Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	SMART HOPPER Command Table	
NV200 Event Table  SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  TEBS Command Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	SMART HOPPER Event Table	
SMART PAYOUT Command Table  SMART PAYOUT Event Table  BV20 Command Table  BV20 Event Table  BV50 Command Table  BV50 Event Table  BV100 Command Table  BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  TEBS Event Table  TEBS Event Table  NV150 Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands  Commands	NV200 Command Table	
SMART PAYOUT Event Table BV20 Command Table BV20 Event Table BV50 Command Table BV50 Event Table BV100 Command Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table COUPON PRINTER Event Table TEBS Command Table TEBS Command Table TEBS Event Table NV150 Command Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Event Table	NV200 Event Table	
BV20 Command Table BV20 Event Table BV50 Command Table BV50 Event Table BV100 Command Table BV100 Event Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table COUPON PRINTER Event Table TEBS Command Table TEBS Command Table TEBS Event Table NV150 Command Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Command Table NV12 Event Table	SMART PAYOUT Command Table	
BV20 Event Table BV50 Command Table BV50 Event Table BV100 Command Table BV100 Event Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table COUPON PRINTER Event Table TEBS Command Table TEBS Event Table NV150 Command Table RV150 Event Table FLATBED PRINTER Event Table FLATBED PRINTER Event Table NV12 Command Table NV12 Command Table NV12 Event Table	SMART PAYOUT Event Table	
BV50 Command Table BV50 Event Table BV100 Command Table BV100 Event Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table COUPON PRINTER Event Table TEBS Command Table TEBS Command Table TEBS Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Command Table NV12 Event Table	BV20 Command Table	
BV50 Event Table BV100 Command Table BV100 Event Table SWART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table COUPON PRINTER Event Table TEBS Command Table TEBS Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Command Table NV12 Command Table NV12 Command Table NV12 Event Table NV12 Event Table Commands	BV20 Event Table	
BV100 Command Table BV100 Event Table SMART SYSTEM Command Table SMART SYSTEM Event Table SMART TICKET Command Table SMART TICKET Event Table COUPON PRINTER Command Table COUPON PRINTER Event Table TEBS Command Table TEBS Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Command Table FLATBED PRINTER Command Table FLATBED PRINTER Command Table NV12 Command Table NV12 Command Table NV12 Event Table	BV50 Command Table	
BV100 Event Table  SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Command Table  NV12 Command Table  NV12 Event Table	BV50 Event Table	
SMART SYSTEM Command Table  SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Command Table  FLATBED PRINTER Command Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands	BV100 Command Table	
SMART SYSTEM Event Table  SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Command Table  NV12 Event Table  Commands	BV100 Event Table	
SMART TICKET Command Table  SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  NV12 Event Table  Commands	SMART SYSTEM Command Table	
SMART TICKET Event Table  COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands	SMART SYSTEM Event Table	
COUPON PRINTER Command Table  COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands		
COUPON PRINTER Event Table  TEBS Command Table  TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  NV12 Event Table  Commands	SMART TICKET Event Table	
TEBS Command Table TEBS Event Table NV150 Command Table NV150 Event Table FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Event Table NV12 Event Table SOURCE STATES	COUPON PRINTER Command Table	
TEBS Event Table  NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  NV12 Event Table  Commands	COUPON PRINTER Event Table	
NV150 Command Table  NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands		
NV150 Event Table  FLATBED PRINTER Command Table  FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands		
FLATBED PRINTER Command Table FLATBED PRINTER Event Table NV12 Command Table NV12 Event Table Commands  Signal		
FLATBED PRINTER Event Table  NV12 Command Table  NV12 Event Table  Commands		
NV12 Command Table  NV12 Event Table  Commands		
NV12 Event Table  Commands		
Commands		
Comp		
Sync		
Pocot	•	

......

Reset

Host Protocol Version	
Poll	
Get Serial Number	
Disable	
Enable	
Get Firmware Version	
Get Dataset Version	
Set Inhibits	
Display On	
Display Off	
Setup Request	
Reject	
Uint Data	
Channel Value Data	
Channel Security Data	
Last Reject Code	
Hold	
Get Barcode Reader Configuration	
Set Barcode Reader Configuration	
Get Barcode Inhibit	
Set Barcode Inhibit	
Get Barcode Data	
Configure Bezel	
Poll With Ack	
Event Ack	
Set Denomination Route	
Get Denomination Route	
Payout Amount	
Get Denomination Level	
Set Denomination Level	
Halt Payout	
Float Amount	
Get Min Payout	
Set Coin Mech Inhibits	
Payout By Denomination	
Float By Denomination	
Empty All	
Set Options	
Get Options	
Coin Mech Global Inhibit	
Smart Empty	
Cashbox Payout Operation Data	
Get All Levels	
Get Counters	
Reset Counters	
Set Refill Mode	
Get Note Positions	
Payout Note	
Stack Note	
Set Value Report Type	
Set Generator	
Set Modulus	
Request Key Exchange	
Coin Mech Options	
Get Build Revision	
Enable Payout Device	
Disable Payout Device	
Comms Pass Through	
Set Baud Rate	
Ssp Set Encryption Key	
Ssp Encryption Reset To Default	
Get Real Time Clock Configuration	
Set Real Time Clock	

Get Real Time Clock	
Set Cashbox Payout Limit	
Get Tebs Barcode	
Enable Tito Events	
Coin Stir	
Ticket Print	
Add Static Text	
Add Place Holder Text	
Add Static Barcode	
Get Image Size	
Get Barcode Size	
Get Ticket Resolution	
Get Font Information	
Get Qr Code Dimensions	
Print Ticket	
Print Blank Ticket	
Get Text Size	
Set Qr Placeholder	
Add Qr Code	
Add Qr Placeholder	
Clear On The Fly Buffer	
Set Placeholder	
Clear Template	
Add Placeholder Barcode	
Add Image	
Get Ticket Size	
Get Free Storage	
Check For Template	
Get Present Templates	
Get Present Fonts	
Get Present Images	
Get Template Info	
Get Template Item Info	
Get Image File Checksum	
Get Ticket Bounds	
Get Pixel Density	
Printer Configuration	
Set Ticket Width	
Set Ticket Height	
Enable Reverse Validation	
Disable Reverse Validation	
Delete File	
Delete File Group	
Set Paper Saving Mode	
Set Bezel Type	
Set Printing Quality	
Get Tebs Log	
Cashbox Unlock Enable	
Cashbox Lock Enable	
Reset Tebs Logs	
Cancel Escrow Transaction	
Commit Escrow Transaction	
Read Escrow Value	
Get Escrow Size	
Set Escrow Size	
Payout Amount By Denomination	
Coin Escrow	
Open Count	
Open Value	
Start	
Reject	
Close	

Slave Reset	
Read	
Note Credit	
Rejecting	
Rejected	
Stacking	
Stacked	
Unsafe Jam	
Disabled	
Fraud Attempt Stacker Full	
Note Cleared From Front	
Note Cleared Into Cashbox	
Cashbox Removed	
Cashbox Replaced	
Barcode Ticket Validated	
Barcode Ticket Validated  Barcode Ticket Ack	
Note Path Open Channel Disable	
Initialising	
Dispensing	
Dispensed Coins Low	
Hopper Jammed Halted	
Floating Floated	
Timeout	
Incomplete Payout	
Incomplete Float  Cashbox Paid	
Coin Credit	
Coin Mech Jammed	
Coin Mech Return Active	
Emptying	
Emptied	
Smart Emptying	
Smart Emptied	
Calibration Failed	
Note Stored In Payout	
Payout Out Of Service	
Jam Recovery	
Error During Payout	
Note Transfered To Stacker	
Note Held In Bezel	
Note Into Store At Reset	
Note Into Stacker At Reset	
Note Dispensed At Reset	
Note Float Removed	
Note Float Attached	
Device Full	
Coin Mech Error	
Coin Rejected	
Attached Coin Mech Disabled	
Attached Coin Mech Enabled	
Value Added	
Tickets Low	
Tickets Replaced	
Printer Head Removed	
Ticket Path Open	
Ticket Jam	
Ticket Printing	
Ticket Printed	

Ticket Printing Error	
Printer Head Replaced	
Ticket Path Closed	
No Paper	
Print Halted	
Ticket In Bezel	
Paper Replaced	
Cashbox Out Of Service	
Printed To Cashbox	
Pay-in Active	
Cashbox Back In Service	
Cashbox Unlock Enabled	
Ticket In Bezel At Startup	
Maintenance Required	
Escrow Active	

#### Introduction

This manual describes the operation of the Smiley ® Secure Protocol SSP.

ITL recommend that you study this manual as there are many new features permitting new uses and more secure applications.

If you do not understand any part of this manual please contact the ITL for assistance. In this way we may continue to improve our product.

Alternatively visit our web site at www.innovative-technology.co.uk

Enhancements of SSP can be requested by contacting: <a href="mailto:support@innovative-technology.co.uk">support@innovative-technology.co.uk</a>

#### MAIN HEADQUARTERS

Innovative Technology Ltd Derker Street, Oldham, England. OL1 4EQ

Tel: +44 161 626 9999 Fax: +44 161 620 2090 E-mail: <a href="mailto:support@innovative-technology.co.uk">support@innovative-technology.co.uk</a>

Web site: www.innovative-technology.co.uk

**Smiley** ® and the **ITL Logo** are international registered trademarks and they are the property of **Innovative Technology Limited**.

Innovative Technology has a number of European and International Patents and Patents Pending protecting this product. If you require further details please contact ITL ®.

Innovative Technology is not responsible for any loss, harm, or damage caused by the installation and use of this product.

This does not affect your local statutory rights.

If in doubt please contact innovative technology for details of any changes.

#### **General Description**

Smiley ® Secure Protocol (SSP) is a secure interface specifically designed by ITL ® to address the problems experienced by cash handling systems in gaming machines. Problems such as acceptor swapping, reprogramming acceptors and line tapping areall addressed.

The interface uses a master-slave model, the host machine is the master and the peripherals (note acceptor, coin acceptor or coin hopper) are the slaves.

Data transfer is over a multi-drop bus using clock asynchronous serial transmissionwith simple open collector drivers. The integrity of data transfers is ensured through the use of 16 bit CRC checksums on all packets.

Each SSP device of a particular type has a unique serial number; this number is used to validate each device in the direction of credit transfer before transactions can takeplace. It is recommended that the encryption system be used to prevent fraud through busmonitoring and tapping. This is compulsory for all payout devices

Commands are currently provided for coin acceptors, note acceptors and coinhoppers. All current features of these devices are supported.

#### **FEATURES:**

- Serial control of Note / Coin Validators and Hoppers
- 4 wire (Tx, Rx, +V, Gnd) system
- Open collector driver, similar to RS232
- High Speed 9600 Baud Rate
- 16 bit CRC error checking
- Data Transfer Mode
- Encryption key negotiation
- 128 Bit AES Encrypted Mode

#### **BENEFITS:**

- Proven in the field
- Simple and low cost interfacing of transaction peripherals.
- High security control of payout peripherals.
- Defence against surrogate validator fraud.
- Straightforward integration into host machines.
- Remote programming of transaction peripherals
- Open standard for universal use.

To help in the software implementation of the SSP, ITL can provide, C/C++ Code, C#.Net Code, DLL controls available on request. Please contact: <a href="mailto:support@innovative-technology.co.uk">support@innovative-technology.co.uk</a>

#### **Hardware layer**

Communication is by character transmission based on standard 8-bit asynchronous data transfer.

Only four wires are required TxD, RxD, +V and ground. The transmit line of the host is open collector, the receive line of each peripheral has a 10Kohm pull-up to 5 volts. The transmit output of each slave is open collector, the receive input of the host has a single 3k3 ohm pull-up to 5 volts.

The data format is as follows:

Encoding	NRZ
Baud Rate	9600
Duplex	Full
Start bits	1
Data Bits	8
Parity	none
Stop bits	2

Caution: Power to peripheral devices would normally be via the serial bus. However devices that require a high current supply in excess of 1.5 Amps, e.g. hoppers, would be expected to be supplied via a separate connector.

#### **Transport Layer**

Data and commands are transported between the host and the slave(s) using a packet format as shown helow:

STX   SEQ/SLAVE ID   LENGTH   DATA   CRCL   CRCH
--------------------------------------------------

STX	Single byte indicating the start of a message - 0x7F hex
SEQ/ Slave ID	Bit 7 is the sequence flag of the packet, bits 6-0 represent the address of the slave the packet is intended for, the highest allowable slave ID is 0x7D
LENGTH	The length of the data included in the packet - this does not include STX, the CRC or the slave $\ensuremath{ID}$
DATA	Commands and data to be transferred
CRCL, CRCH	Low and high byte of a forward CRC-16 algorithm using the Polynomial (X16 $\pm$ X15 $\pm$ X2 $\pm$ 1) calculated on all bytes, except STX. It is initialised using the seed 0xFFFF. The CRC is calculated before byte stuffing.

#### **PACKET SEQUENCING**

Byte stuffing is used to encode any STX bytes that are included in the data to be transmitted. If 0x7F (STX) appears in the data to be transmitted then it should be replaced by 0x7F, 0x7F.

Byte stuffing is done after the CRC is calculated, the CRC its self can be byte stuffed. The maximum length of data is 0xFF bytes.

The sequence flag is used to allow the slave to determine whether a packet is a re-transmission due to its last reply being lost. Each time the master sends a new packet to a slave it alternates the sequence flag. If a slave receives a packet with the same sequence flag as the last one, it does not execute the command but simply repeats it's last reply. In a reply packet the address and sequence flag match the command packet.

This ensures that no other slaves interpret the reply as a command and informs the master that the correct slave replied. After the master has sent a command to one of the slaves, it will wait for 1 second fora reply. After that, it will assume the slave did not receive the command intact so it will re-transmit it with the same sequence flag. The host should also record the fact that a gap in transmission has occurred and prepare to poll the slave for its serial number identity following the current message. In this way, the replacement of the hosts validator by a fraudulent unit can be detected.

The frequency of polling should be selected to minimise the possibility of swapping a validator between polls. If the slave has not received the original transmission, it will see the re-transmission as a new command so it will execute it and reply. If the slave had seen the original command but its reply had been corrupted then the slave will ignore the command but repeat its reply. After twenty retries, the master will assume that the slave has crashed. A slave has no time-out or retry limit. If it receives a lone sync byte part way through receiving a packet it will discard the packet received so far and treat the next byte as an address byte.

#### **Encryption Layer**

#### **PACKET FORMAT**

Encryption is mandatory for all payout devices and optional for pay in devices. Encrypted data and commands are transported between the host and the slave(s) using the transport mechanism described above, the encrypted information is stored in the data field in the format shown below:

STX		SEQ/SL	AVE ID		LEN	GTH	DATA	CRCL	CRCH
				D	ATA				
ST	EX			Encr	ypted Da	ta			
Encrypted Data									
eLENGT	ТН	eCOUNT	eDATA	ePA	CKING	eCRCL	eCRCH		

STEX	Single byte indicating the start of an encrypted data block - 0x7E
eLENGTH	The length of the data included in the packet - this does not include STEX, COUNT, the packing or the CRC
eCOUNT	A four byte unsigned integer. This is a sequence count of encrypted packets, it is incremented each time a packet is encrypted and sent, and each time an encrypted packet is received and decrypted.
eDATA	Commands or data to be transferred
	Random data to make the length of the length +count + data + packing + CRCL + CRCH to be a multiple of 16 bytes
eCRCL/eCRCH	Low and high byte of a forward CRC-16 algorithm using the polynomial (X16 + X15 + X2 +1) calculated on all bytes except STEX. It is initialised using the seed $0xFFFF$

After power up and reset the slave will stay disabled and will respond to all commands with the generic response KEY\_NOT\_SET (0xFA), without executing the command, until the key has been negotiated. There are two classes of command and response, general commands and commands involved in credit transfer.

General commands may be sent with or without using the encryption layer. The slave will reply using the same method, unless the response contains credit information, in this case the reply will always be encrypted. Credit

transfer commands, a hopper payout for example, will only be accepted by the slave if received encrypted. Commands that must be encrypted on an encryption-enabled product are indicated on the command descriptions for each command. The STEX byte is used to determine the packet type. Ideally all communications will be encrypted.

After the data has been decrypted the CRC algorithm is performed on all bytes including the CRC. The result of this calculation will be zero if the data has been decrypted with the correct key. If the result of this calculation is non-zero then the peripheral should assume that the host did not encrypt the data (transmission errors are detected by the transport layer). The slave should go out of service until it is reset.

The packets are sequenced using the sequence count; this is reset to 0 after a power cycle and each time the encryption keys are successfully negotiated. The count is incremented by the host and slave each time they successfully encrypt and transmit a

packet. After a packet is successfully decrypted the COUNT in the packet should be compared with the internal COUNT, if they do not match then the packet is discarded.

#### **Encryption Keys**

The encryption key length is 128 bits. However this is divided into two parts. The lower 64 bits are fixed and specified by the machine manufacturer, this allows the manufacturer control which devices are used in their machines.

The higher 64 bits are securely negotiated by the slave and host at power up, this ensures each machine and each session are using different keys. The key is negotiated by the Diffie-Hellman key exchange method. See: <a href="mailto:en.wikipedia.org/wiki/Diffie-Hellman">en.wikipedia.org/wiki/Diffie-Hellman</a>

The exchange method is summarised in the table below. C code for the exchange algorithm is available from ITI

Step	Host	Slave
1	Generate prime number GENERATOR	
2	Use command Set Generator to send to slave Check GENERATOR is prime and store	Check GENERATOR is prime and store
3	Generate prime number MODULUS	
4	Use command Set Modulus to send to slave Check MODULUS is prime and store	Check MODULUS is prime and store
5	Generate Random Number HOST_RND	
6	Calculate HostInterKey: = GENERATOR ^ HOST_RND mod MODULUS	
7	Use command Request Key Exchange to send to slave.	Generate Random Number SLAVE_RND
8		Calculate SlaveInterKey: = GENERATOR ^ SLAVE_RND mod MODULUS
9		Send to host as reply to Request Key Exchange
10	Calculate Key: = SlaveInterKey ^ HOST_RND mod MODULUS	Calculate Key: = HostInterKey ^ SLAVE_RND mod MODULUS

Note:  $\ ^{\ }$  represents to the power of

# **Generic Commands and Responses**

All devices must respond to a list of so-called Generic Commands as show in the table below.

Command	Code
Reset	0x01
Host Protocol Version	0x06
Get Serial Number	0x0C
Sync	0x11
Disable	0x09
Enable	0x0A
Get Firmware Version	0x20
Get Dataset Version	0x21

A device will respond to all commands with the first data byte as one of the Generic responses list below.

Generic Response	Code	Description
ОК	0xF0	Returned when a command from the host is understood and
COMMAND NOT KNOWN	0xF2	has been, or is in the process of, being executed.  Returned when an invalid command is received by a peripheral.
WRONG No PARAMETERS	0xF3	A command was received by a peripheral, but an incorrect number of parameters were received.
PARAMETERS	0xF4	One of the parameters sent with a command is out of range.
COMMAND CANNOT BE PROCESSED	0xF5	A command sent could not be processed at that time. E.g. sending a dispense command before the last dispense operation has completed.
SOFTWARE ERROR	0xF6	Reported for errors in the execution of software e.g. Divide by zero. This may also be reported if there is a problem resulting from a failed remote firmware upgrade, in this case the firmware upgrade should be redone.
FAIL	0xF8	Command failure
KEY NOT SET	0xFA	The slave is in encrypted communication mode but the encryption keys have not been negotiated.

#### **Protocol Versions**

An SSP Poll command returns a list of events and data that have occurred in the device since the last poll.

The host machine then reads this event list taking note of the data length (if any) of each event.

On order to introduce new events, SSP uses a system of **Protocol Version** levels to identify the event types and sizes a machine can expect to see in reponse to a poll. If this were not done, new unknown events with unknown datasize to a machine not set-up for these would cause the event reading to fail.

A host system should take note of the protocol version of the device connected and ensure that it is not set for a higer version that the one it is expecting to use.

The host can also check that the device can also be set to the higher protocol level, ensuring that expected events will be seen.

The listed events in this manual show the protocol version level of each event.

As part of the start-up procedure, the host should read the current protocol level of the device (using the <u>set-up reques</u>t command).

# **Banknote Validator**

A Banknote Validator is a device which will scan, validate and stack a banknote it detects as valid or reject it from the front if not valid. Some banknote validators can be transformed into payout devices by the addition of a pay-out unit. All  $ITL^{TM}$  Banknote validators support the SSP protocol described here.

# The Banknote Validators have a default SSP Address of 0.

The <u>setup request</u> reponse table for banknote validator types:

#### **Protocol versions less than 6:**

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x00 = Banknote validator
Firmware version	1	4	ASCII data of device firmware version (e.g. $^{\prime}0110^{\prime}=1.10)$
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device

# Protocol versions greater than or equal to 6:

Data	byte offset	size (bytes)	notes
Unit type	0	1	0 = Banknote validator
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device
Expanded channel country code	16 + (n * 2)	n * 3	Three byte ascii code for each channel. This allows multi currency datasets to be used on SSP devices. These bytes are given only on protocol versions >= 6.
Expanded channel value	16 + (n * 5)	n * 4	4 bytes for each channel value. These bytes are given only on protocol versions >= 6.

# **Reject Codes**

The banknote validator specification includes a command <u>Last Reject Code</u>.

Use this command after a note has been rejected to return a one-byte code to determine the cause of the note reject.

Table showing reject codes:

		<b>5</b> • <b>5</b> • • • • • • • • • • • • • • • • • • •	
0×00	0	NOTE ACCEPTED	The banknote has been accepted. No reject has occured.
0×01	1	LENGTH FAIL	A validation fail: The banknote has been read but it's length registers over the max length parameter.
0x02	2	AVERAGE FAIL	Internal validation failure - banknote not recognised.
0x03	3	COASTLINE FAIL	Internal validation failure - banknote not recognised.
0x04	4	GRAPH FAIL	Internal validation failure - banknote not recognised.
0x05	5	BURIED FAIL	Internal validation failure - banknote not recognised.
0x06	6	CHANNEL INHIBIT	This banknote has been inhibited for acceptance in the dataset configuration.
0×07	7	SECOND NOTE DETECTED	A second banknote was inserted into the validator while the first one was still being transported through the banknote path.
0x08	8	REJECT BY HOST	The host system issues a Reject command when this banknote was held in escrow.
0x09	9	CROSS CHANNEL DETECTED	This bank note was identified as exisiting in two or more seperate channel definitions in the dataset.
0x0A	10	REAR SENSOR ERROR	An inconsistency in a position sensor detection was seen
0x0B	11	NOTE TOO LONG	The banknote failed dataset length checks.
0x0C	12	DISABLED BY HOST	The bank note was validated on a channel that has been inhibited for acceptance by the host system.
0x0D	13	SLOW MECH	The internal mechanism was detected as moving too slowly for correct validation.
0x0E	14	STRIM ATTEMPT	An attempt to fraud the system was detected.
0x0F	15	FRAUD CHANNEL	Obselete response.
0x10	16	NO NOTES DETECTED	A banknote detection was initiated but no banknotes were seen at the validation section.
0x11	17	PEAK DETECT FAIL	Internal validation fail. Banknote not recognised.
0x12	18	TWISTED NOTE REJECT	Internal validation fail. Banknote not recognised.
0x13	19	ESCROW TIME-OUT	A banknote held in escrow was rejected due to the host not communicating within the time-out period. The default timeout period is the same as the poll timeout i.e. $10$ seconds.
0x14	20	BAR CODE SCAN FAIL	Internal validation fail. Banknote not recognised.
0x15	21	NO CAM ACTIVATE	A banknote did not reach the internal note path for validation during transport.
0x16	22	SLOT FAIL 1	Internal validation fail. Banknote not recognised.
0×17	23	SLOT FAIL 2	Internal validation fail. Banknote not recognised.
0x18	24	LENS OVERSAMPLE	The banknote was transported faster than the system could sample the note.
0×19	25	WIDTH DETECTION FAIL	The banknote failed a measurement test.
		SHORT NOTE	

0x1A	26	DETECT	The banknote measured length fell outside of the validation parameter for minimum length.
0x1B	27	PAYOUT NOTE	The reject code cammand was issued after a note was payed out using a note payout device.
0x1C	28	DOUBLE NOTE DETECTED	Mote than one banknote was detected as overlayed during note entry.
0x1D	29	UNABLE TO STACK	The bill was unable to reach it's correct stacking position during transport.

#### **SMART Ticket**

The SMART Ticket device is an add on unit to the NV200 to enable printing and payout via the NV200 bezel of paper tickets of configurabale designs. A range of SSP commands may be implemented to configure, modify and maintain print designs from the host on-the-fly or by pre-configured templates.

The SMART Ticket device is addressed seperately from the NV200, the NV200 setup request command will return 0x08 for the Uint type if a SMART Ticket device is attached.

When communicating with the NV200 attached to the printer, optional additional poll events may be enabled. These are enabled by sending an SSP packet with the command header 0x72 to the NV200. Polls will the respond with the same printing (0xA5) and printed (0xA6) poll responses as the printer.

#### The SMART Ticket has a default SSP Address of 64 dec 0x40 hex

The <u>setup request</u> reponse table for SMART Ticket types:

Data	byte offset	size (bytes)	notes
Unit type	0	1	8 = Addon Printer
Firmware version	1	4	ASCII data of device firmware version (e.g. $'0110' = 1.10$ )
Cutter enabled status	5	1	(0 for disabled)
Tab enabled status	6	1	(0 for disabled)
Reverse validation enabled status	7	1	(0 for disabled)
Font pack code (ASCII)	8	3	e.g. 'FP1'
Printer type	11	1	Printer Type: 0x0 for Fan Fold, 0x1 Paper Roll (Cutter fitted)
SD card fitted status	12	1	1 for detected
Printer darkness/quality setting	13	1	The current protocol version set for this device

#### **SMART Hopper**

SMART Hopper is a coin payout device capable of discriminating and paying out multi-denominations of stored coins from its internal storage hopper.

Coins added to the hopper can be designated to be routed to an external cashbox on detection or recycled and stored in the hopper unit to be available for a requested payout.

SMART Hopper also supports the addition of a connected  $cctalk^{\text{TM}}$  or  $essp^{\text{TM}}$  coin mechanism which will automatically add its validated coins to the SMART Hopper system levels.

Note that payout values are in terms of the of the penny value of that currency. So for 5.00, the value sent and returned by the hopper would be 500. All transactions with a SMART hopper must be encrypted to prevent dispense commands being recorded and replayed by an external device.

#### Addressing

#### The SMART Hopper has a default SSP Address of 16 dec 0x10 hex.

The <u>setup request</u> reponse table for coin hopper types:

#### **Protocol version less than 6:**

Data	byte offset	size (bytes)	notes
Unit type	0	1	3 = SMART Hopper
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Protocol Version	8	1	The current protocol version set for this device
Number of coin values	9	1	The number of coin denominations in this device dataset. [n]
Coin values	10	n * 2	2 byte each value for the coin denominations (e.g. $0.05$ coin = $0x05,0x00$ )

#### Protocol version greater or equal to 6:

Data	byte offset	size (bytes)	notes
Unit type	0	1	3 = SMART Hopper
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Protocol Version	8	1	The current protocol version set for this device
Number of coin values	9	1	The number of coin denominations in this device dataset. [n]
Coin values	10	n * 2	2 byte each value for the coin denominations (e.g. $0.05$ coin = $0x05,0x00$ )
Country codes	10 + (n * 2)		An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.

# **Coupon Printer**

The Coupon Printer device is a stand alone thermal printer designed for printing coupons/recepits/tickets using roll media with a width of 58mm. A range of SSP command may be implemented to configure, modify and maintain print designs from the host on-the-fly or by pre-configured templates.

The commands rely on per-existing resources of images, fonts and templates that are programmed into the Coupon Printer device.

# The Coupon Printer has a default SSP Address of 65 dec 0x41 hex

The <u>setup request</u> reponse table for Coupon Printer types:

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x0B = Stand Alone Printer
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = $1.10$ )
Cutter enabled status	5	1	(0 for disabled)
Tab enabled status	6	1	(0 for disabled)
Reverse validation enabled status	7	1	(0 for disabled)
Font pack code (ASCII)	8	3	e.g. 'FP1'
Printer type	11	1	Printer Type: 0x0 for Fan Fold, 0x1 Paper Roll (Cutter fitted)
SD card fitted status	12	1	1 for detected
Printer darkness/quality setting	13	1	The current protocol version set for this device

#### **SMART Payout**

The Smart Payout is an extension of a banknote validator, all commands are sent to the validator using its address (0x00). Information on the types of note that can be handled is obtained from the standard note validator commands.

Note that payout values are in terms of the penny value of that currency. So for 5.00, the value sent and returned by the payout would be 500.

The host simply has to tell the unit the value it wishes to dispense. The unit will manage which notes are stored to be used for payout and their location to minimise the payout time, and which notes, of the type enable for storage, are sent to the stacker. This is the recommended mode of operation.

#### The SMART Payout has a default SSP Address of 0.

The <u>setup request</u> reponse table for SMART Payout types:

#### **Protocol versions less than 6:**

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x06 = SMART Payout
Firmware version	1	4	ASCII data of device firmware version (e.g. $'0110' = 1.10)$
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is $> 0$ otherwise 0.
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device

#### Protocol versions greater than or equal to 6:

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x06 = SMART Payout
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device
Expanded channel			Three byte ascii code for each channel. This allows multi currency datasets to be used on SSP devices. These bytes are

country code	16 + (n * 2)	n * 3	given only on protocol versions >= 6.
Expanded channel value	16 + (n * 5)	n * 4	4 bytes for each channel value. These bytes are given only on protocol versions >= 6.

#### **Smart System**

The Smart System device is a multi-coin pay-in, pay-out system with detachable fast coin pay-in feeder.

Coins fed into the pay-in head will be validated and counted and recognised coins are routed to the attached hopper while rejected coins are fed out of the front of the system.

Coin hopper levels are adjusted internally.

The system can function as a stand-alone hopper payout system if the pay-in feeder head is removed.

# The SMART Systemhas a default SSP Address of 16 dec 0x10 hex

The <u>setup request</u> reponse table for coin hopper types:

# **Protocol version less than 6:**

Data	byte offset	size (bytes)	notes
Unit type	0	1	3 = SMART Hopper 9 = Smart Coin System
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Protocol Version	8	1	The current protocol version set for this device
Number of coin values	9	1	The number of coin denominations in this device dataset. [n]
Coin values	10	n * 2	2 byte each value for the coin denominations (e.g. $0.05$ coin = $0x05,0x00$ )

#### Protocol version greater or equal to 6:

Data	byte offset	size (bytes)	notes	
Unit type	0	1	3 = SMART Hopper 9 = Smart Coin System	
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)	
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')	
Protocol Version	8	1	The current protocol version set for this device	
Number of coin values	coin values  9 1 The number of coin denoming  10  10  10  10  10  10  10  10  10  1		The number of coin denominations in this device dataset. [n]	
Coin values			2 byte each value for the coin denominations (e.g. $0.05$ coin = $0x05,0x00$ )	
Country codes	10 + (n * 2)		An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.	

# Note Float (NV11)

The Note Float is an extension of a banknote validator, all commands are sent to the validator using its address (0x00). Information on the types of note that can be handled is obtained from the standard note validator commands.

# The NV11 (Note Float) has a default SSP Address of 0.

The <u>setup request</u> reponse table for Note Float types:

#### **Protocol versions less than 6:**

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x07 = Note Float (NV11)
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device

# Protocol versions greater than or equal to 6:

Data	byte offset	size (bytes)	notes	
Unit type	0	1	0x07 = Note Float (NV11)	
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)	
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')	
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.	
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)	
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.	
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.	
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents	
Protocol version	15 + (n * 2)	1	The current protocol version set for this device	
Expanded channel country code	16 + (n * 2)	n * 3	Three byte ascii code for each channel. This allows multi currency datasets to be used on SSP devices. These bytes are given only on protocol versions >= 6.	
Expanded channel value	16 + (n * 5)	n * 4	4 bytes for each channel value. These bytes are given only on protocol versions >= 6.	

# **TEBS**

TEBS or Tamper Evident Bag System is a version of the NV200 banknote validator with a special cashbox attachedment which operates as device to store bank notes into a special bag which will then be sealed when the cashbox is extracted.

Each of the bags has a unique barcode which is registered by the TEBS system enableing the host system to register cash amounts in each bag.

The <u>setup request</u> reponse table for TEBS types:

# **Protocol versions less than 6:**

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x0D = TEBS, 0x0E = TEBS with SMART Payout, 0x0F = TEBS with SMART Ticket
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.
Number of channels	11	1	The highest channel used in this device dataset [n] (1-16)
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents
Protocol version	15 + (n * 2)	1	The current protocol version set for this device

# Protocol versions greater than or equal to 6:

Data	byte offset	size (bytes)	notes	
Unit type	0	1	0x0D = TEBS, 0x0E = TEBS with SMART Payout, 0x0F = TEBS with SMART Ticket	
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)	
Country code	5	3	ASCII code of the device dataset (e.g. 'EUR')	
Value Multiplier	8	3	3 The value to multiply the individual channels by to get the full value. If this value is 0 then it indicates that this is a protocol version 6 or greater compatible dataset where the values are given in the expanded segment of the return data.	
Number of channels	11	1	1 The highest channel used in this device dataset [n] (1	
Channel Values	12	n	A variable size array of byes, 1 for each channel with a value from 1 to 255 which when multiplied by the value multiplier gives the full value of the note. If the value multiplier is zero then these values are zero.	
Channel Security	12 + n	n	An obsolete value showing security level. This is set to 2 if the value multiplier is > 0 otherwise 0.	
Real value Multiplier	12 +( n * 2)	3	The value by which the channel values can be multiplied to show their full value e.g. 5.00 EUR = 500 EUR cents	
Protocol version	15 + (n * 2)	1	The current protocol version set for this device	
Expanded channel country code	16 + (n * 2)	n * 3	Three byte ascii code for each channel. This allows multi currency datasets to be used on SSP devices. These bytes are given only on protocol versions >= 6.	

Expanded channel	16 + (n * 5)	n * 4	4 bytes for each channel value. These bytes are given only on protocol versions >= 6.
value			

#### NVR-280 (NV12)

The NVR-280 is an addon printer for the NV9 USB Plus. Combined with the NV9 USB Plus, the device is known as an NV12. This devices allows the printing of tickets with will exit through the NV9's note path and out of it's bezel. It also allows the NV9 to read barcodes on these printed tickets.

A range of SSP commands may be implemented to configure, modify and maintain print designs from the host on-the-fly or by pre-configured templates.

The NVR-280 device is addressed seperately from the NV9, the NV9 setup request command will return 0x08 for the Unit type if an NVR-280 device is attached.

When communicating with the NV9 attached to the printer, optional additional poll events may be enabled. These are enabled by sending an SSP packet with the command header 0x72 to the NV9. Polls will the respond with the same printing (0xA5) and printed (0xA6) poll responses as the printer.

#### The NVR-280 has a default SSP Address of 64 dec 0x40 hex

The setup request reponse table for NVR-280 types:

Data	byte offset	size (bytes)	notes
Unit type	0	1	8 = Addon Printer
Firmware version	1	4	ASCII data of device firmware version (e.g. '0110' = 1.10)
Cutter enabled status	5	1	(0 for disabled, always 1 on this printer)
Tab enabled status	6	1	(0 for disabled, always 0 on this printer)
Reverse validation enabled status	7	1	(0 for disabled)
Font pack code (ASCII)	8	3	e.g. 'FP1'
Printer type	11	1	Printer Type: 0x0 for Fan Fold, 0x1 Paper Roll (Cutter fitted)(Always 0x1 on this printer)
SD card fitted status	12	1	1 for detected
Printer darkness/quality setting	13	1	The current protocol version set for this device

# Flatbed Printer (FBF-166)

The Flatbed Printer device is a stand alone thermal printer designed for printing tickets using fanfold media with a width of 65mm. A range of SSP command may be implemented to configure, modify and maintain print designs from the host on-the-fly or by pre-configured templates.

The commands rely on per-existing resources of images, fonts and templates that are programmed into the Flatbed Printer device.

# The Flatbed Printer has a default SSP Address of 65 dec 0x41 hex

The <u>setup request</u> reponse table for Coupon Printer types:

Data	byte offset	size (bytes)	notes
Unit type	0	1	0x0B = Stand Alone Printer
Firmware version	1	4	ASCII data of device firmware version (e.g. $'0110' = 1.10$ )
Cutter enabled status	5	1	(0 for disabled)
Tab enabled status	6	1	(0 for disabled)
Reverse validation enabled status	7	1	(0 for disabled)
Font pack code (ASCII)	code 8 3	e.g. 'FP1'	
Printer type	11	1	Printer Type: 0x0 for Fan Fold, 0x1 Paper Roll (Cutter fitted)
SD card fitted status	12	1	1 for detected
Printer darkness/quality setting	13	1	The current protocol version set for this device

# **NV9USB Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Poll With Ack	0x56	86
Event Ack	0x57	87
Get Counters	0x58	88
Reset Counters	0x59	89
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
	0x00	

# **NV9USB Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xEF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Channel Disable	0xB5	181
Initialising	0xB6	182
Ticket Printing	0xA5	165
Ticket Printed	0xA6	166
Ticket Printing Error	0xA8	168
Print Halted	0xAE	174
Ticket In Bezel	0xAD	173
Printed To Cashbox	0xAF	175
Ticket In Bezel At Startup	0xA7	167

# **NV10USB Command Table**

	Header code (hex)	dec
Sync	0×11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97

# **NV10USB Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xEF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0xE7	231
Note Cleared Into Cashbox	0xE2	226
Channel Disable	0xB5	181

# **NV11 Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Denomination Route	0x3B	59
Get Denomination Route	0x3C	60
Empty All	0x3F	63
Smart Empty	0x52	82
Get Counters	0x58	88
Reset Counters	0x59	89
Get Note Positions	0x41	65
Payout Note	0x42	66
Stack Note	0x43	67
Set Value Report Type	0x45	69
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Get Build Revision	0x4F	79
Enable Payout Device	0x5C	92
Disable Payout Device	0x5B	91
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
	0x00	

# **NV11 Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xEF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Channel Disable	0xB5	181
Dispensing	0xDA	218
Halted	0xD6	214
Emptying	0xC2	194
Emptied	0xC3	195
Smart Emptying	0xB3	179
Smart Emptied	0xB4	180
Note Stored In Payout	0xDB	219
Payout Out Of Service	0xC6	198
Note Transfered To Stacker	0xC9	201
Note Held In Bezel	0xCE	206
Note Into Store At Reset	0xCB	203
Note Into Stacker At Reset	0xCA	202
Note Dispensed At Reset	0xCD	205
Note Float Removed	0xC7	199
Note Float Attached	0xC8	200
Device Full	0xCF	207

# **SMART HOPPER Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Setup Request	0x05	5
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Denomination Route	0x3B	59
Get Denomination Route	0x3C	60
Payout Amount	0x33	51
Get Denomination Level	0x35	53
Set Denomination Level	0x34	52
Halt Payout	0x38	56
Float Amount	0x3D	61
Get Min Payout	0x3E	62
Set Coin Mech Inhibits	0x40	64
Payout By Denomination	0x46	70
Float By Denomination	0x44	68
Empty All	0x3F	63
Set Options	0x50	80
Get Options	0x51	81
Coin Mech Global Inhibit	0x49	73
Smart Empty	0x52	82
Cashbox Payout Operation Data	0x53	83
Get All Levels	0x22	34
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Coin Mech Options	0x5A	90
Get Build Revision	0x4F	79
Comms Pass Through	0x37	55
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Set Cashbox Payout Limit	0x4E	78

# **SMART HOPPER Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Initialising	0xB6	182
Dispensing	0xDA	218
Coins Low	0xD3	211
Hopper Jammed	0xD5	213
Halted	0xD6	214
Floating	0xD7	215
Floated	0xD8	216
Timeout	0xD9	217
Incomplete Payout	0xDC	220
Incomplete Float	0xDD	221
Cashbox Paid	0xDE	222
Coin Credit	0xDF	223
Coin Mech Jammed	0xC4	196
Coin Mech Return Active	0xC5	197
Emptying	0xC2	194
Emptied	0xC3	195
Smart Emptying	0xB3	179
Smart Emptied	0xB4	180
Calibration Failed	0x83	131
Coin Mech Error	0xB7	183
Attached Coin Mech Disabled	0xBD	189
Attached Coin Mech Enabled	0xBE	190

### **NV200 Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Get Barcode Reader Configuration	0x23	35
Set Barcode Reader Configuration	0x24	36
Get Barcode Inhibit	0x25	37
Set Barcode Inhibit	0x26	38
Get Barcode Data	0x27	39
Configure Bezel	0x54	84
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Get Build Revision	0x4F	79
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Enable Tito Events	0x72	114

# **NV200 Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xEF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Cashbox Removed	0xE3	227
Cashbox Replaced	0xE4	228
Barcode Ticket Validated	0xE5	229
Barcode Ticket Ack	0xD1	209
Note Path Open	0xE0	224
Channel Disable	0xB5	181
Initialising	0xB6	182
Ticket Printing	0xA5	165
Ticket Printed	0xA6	166
Ticket Printing Error	0xA8	168
Print Halted	0xAE	174
Ticket In Bezel	0xAD	173
Printed To Cashbox	0xAF	175

### **SMART PAYOUT Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Get Barcode Reader Configuration	0x23	35
Set Barcode Reader Configuration	0x24	36
Get Barcode Inhibit	0x25	37
Set Barcode Inhibit	0x26	38
Get Barcode Data	0x27	39
Configure Bezel	0x54	84
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Denomination Route	0x3B	59
Get Denomination Route	0x3C	60
Payout Amount	0x33	51
Get Denomination Level	0x35	53
Halt Payout	0x38	56
Float Amount	0x3D	61
Get Min Payout	0x3E	62
Payout By Denomination	0x46	70
Float By Denomination	0x44	68
Empty All	0x3F	63
Smart Empty	0x52	82
Cashbox Payout Operation Data	0x53	83
Get All Levels	0x22	34
Get Counters	0x58	88
Reset Counters	0x59	89
Set Refill Mode	0x30	48
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Get Build Revision	0x4F	79
Enable Payout Device Disable Payout Device	0x5C	92
•	0x5B	91
Set Baud Rate	0x4D 0x60	77 06
Ssp Set Encryption Key Ssp Encryption Reset To Default		96 97
Cancel Escrow Transaction	0x61	
Commit Escrow Transaction	0x76 0x77	118 119
Read Escrow Value		
Get Escrow Value	0x78 0x79	120 121
Set Escrow Size	0x79 0x7A	121
SEE ESCHOW SIZE	UA/A	122

# **SMART PAYOUT Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xFF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Cashbox Removed	0xE3	227
Cashbox Replaced	0xE4	228
Barcode Ticket Validated	0xE5	229
Barcode Ticket Ack	0xD1	209
Note Path Open	0xE0	224
Channel Disable	0xB5	181
Initialising	0xB6	182
Dispensing	0xDA	218
Dispensed	0xD2	210
Hopper Jammed	0xD5	213
Halted	0xD6	214
Floating	0xD7	215
Floated	0xD8	216
Incomplete Payout	0xDC	220
Incomplete Float	0xDD	221
Emptying	0xC2	194
Emptied	0xC3	195
Smart Emptying	0xB3	179
Smart Emptied	0xB4	180
Note Stored In Payout	0xDB	219
Jam Recovery	0xB0	176
Error During Payout	0xB1	177
Note Transfered To Stacker	0xC9	201
Note Held In Bezel	0xCE	206
Note Into Store At Reset	0xCB	203
Note Into Stacker At Reset	0xCA	202

### **BV20 Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97

### **BV20 Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0×EF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0×E7	231
Channel Disable	0xB5	181
Initialising	0xB6	182

### **BV50 Command Table**

	Header code (hex)	dec
Sync	0×11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97

### **BV50 Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Disabled	0xE8	232
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Cashbox Removed	0xE3	227
Cashbox Replaced	0xE4	228
Channel Disable	0xB5	181
Initialising	0xB6	182

### **BV100 Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97

### **BV100 Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xEF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Cashbox Removed	0xE3	227
Cashbox Replaced	0xE4	228
Channel Disable	0xB5	181
Initialising	0xB6	182

### **SMART SYSTEM Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Setup Request	0x05	5
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Denomination Route	0x3B	59
Get Denomination Route	0x3C	60
Payout Amount	0x33	51
Get Denomination Level	0x35	53
Set Denomination Level	0x34	52
Halt Payout	0x38	56
Float Amount	0x3D	61
Get Min Payout	0x3E	62
Set Coin Mech Inhibits	0x40	64
Payout By Denomination	0x46	70
Float By Denomination	0x44	68
Empty All	0x3F	63
Set Options	0x50	80
Get Options	0x51	81
Coin Mech Global Inhibit	0x49	73
Smart Empty	0x52	82
Cashbox Payout Operation Data	0x53	83
Get All Levels	0x22	34
Get Counters	0x58	88
Reset Counters	0x59	89
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Coin Mech Options	0x5A	90
Get Build Revision	0x4F	79
Comms Pass Through	0x37	55
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Get Real Time Clock Configuration	0x62	98
Set Real Time Clock	0x64	100
Get Real Time Clock	0x63	99
Set Cashbox Payout Limit	0x4E	78
Coin Stir	0x5D	93
Payout Amount By Denomination	0x39	57
Coin Escrow	0x3A	58

### **SMART SYSTEM Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Initialising	0xB6	182
Dispensing	0xDA	218
Dispensed	0xD2	210
Hopper Jammed	0xD5	213
Halted	0xD6	214
Floating	0xD7	215
Floated	0xD8	216
Timeout	0xD9	217
Incomplete Payout	0xDC	220
Incomplete Float	0xDD	221
Cashbox Paid	0xDE	222
Coin Mech Jammed	0xC4	196
Coin Mech Return Active	0xC5	197
Emptying	0xC2	194
Emptied	0xC3	195
Smart Emptying	0xB3	179
Smart Emptied	0xB4	180
Calibration Failed	0x83	131
Device Full	0xCF	207
Coin Mech Error	0xB7	183
Coin Rejected	0xBA	186
Attached Coin Mech Disabled	0xBD	189
Attached Coin Mech Enabled	0xBE	190
Value Added	0xBF	191
Pay-in Active	0xC1	193
Maintenance Required	0xC0	192
Escrow Active	0x8B	139

### **SMART TICKET Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Setup Request	0x05	5
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Get Real Time Clock Configuration	0x62	98
Set Real Time Clock	0x64	100
Get Real Time Clock	0x63	99
Ticket Print	0x70	112
Printer Configuration	0x71	113

# **SMART TICKET Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Disabled	0xE8	232
Tickets Low	0xA0	160
Tickets Replaced	0xA1	161
Printer Head Removed	0xA2	162
Ticket Path Open	0xA3	163
Ticket Jam	0xA4	164
Ticket Printing	0xA5	165
Ticket Printed	0xA6	166
Ticket Printing Error	0xA8	168
Printer Head Replaced	0xA9	169
Ticket Path Closed	0xAA	170
No Paper	0xAB	171
Print Halted	0xAE	174
Ticket In Bezel	0xAD	173
Paper Replaced	0xAC	172
Printed To Cashbox	0×AF	175

# **COUPON PRINTER Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Get Real Time Clock Configuration	0x62	98
Set Real Time Clock	0x64	100
Get Real Time Clock	0x63	99
Ticket Print	0x70	112
Printer Configuration	0x71	113

# **COUPON PRINTER Event Table**

	Header code (hex)	dec
Slave Reset	0×F1	241
Disabled	0xE8	232
Tickets Low	0xA0	160
Tickets Replaced	0xA1	161
Printer Head Removed	0xA2	162
Ticket Jam	0xA4	164
Ticket Printing	0xA5	165
Ticket Printed	0xA6	166
Ticket Printing Error	0xA8	168
Printer Head Replaced	0xA9	169
No Paper	0×AB	171
Paper Replaced	0xAC	172

### **TEBS Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Get Dataset Version	0x21	33
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0x05	5
Reject	0x08	8
Uint Data	0x0D	13
Channel Value Data	0x0E	14
Channel Security Data	0x0F	15
	0x00	
Last Reject Code	0x17	23
Hold	0x18	24
Get Barcode Reader Configuration	0x23	35
Set Barcode Reader Configuration	0x24	36
Get Barcode Inhibit	0x25	37
Set Barcode Inhibit	0x26	38
Configure Bezel	0x54	84
Poll With Ack	0x56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Set Baud Rate	0x4D	77
Ssp Set Encryption Key	0x60	96
Ssp Encryption Reset To Default	0x61	97
Get Tebs Barcode	0x65	101
Get Tebs Log	0x66	102
Cashbox Unlock Enable	0x67	103
Cashbox Lock Enable	0x68	104
Reset Tebs Logs	0x69	105

### **TEBS Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0xFF	239
Note Credit	0xEE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0xE6	230
Stacker Full	0xE7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Cashbox Removed	0xE3	227
Cashbox Replaced	0xE4	228
Barcode Ticket Validated	0xE5	229
Barcode Ticket Ack	0xD1	209
Note Path Open	0xE0	224
Channel Disable	0xB5	181
Initialising	0xB6	182
Cashbox Out Of Service	0x90	144
Cashbox Back In Service	0x92	146
Cashbox Unlock Enabled	0x93	147

### **NV150 Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0×01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0×09	9
Enable	0x0A	10
Set Inhibits	0x02	2
Display On	0x03	3
Display Off	0x04	4
Setup Request	0×05	5
Reject	0×08	8
Uint Data	0x0D	13
Channel Value Data	0×0E	14
Channel Security Data	0×0F	15
	0×00	
Last Reject Code	0×17	23
Hold	0x18	24
Get Barcode Reader Configuration	0x23	35
Set Barcode Reader Configuration	0x24	36
Get Barcode Inhibit	0x25	37
Set Barcode Inhibit	0x26	38
Get Barcode Data	0x27	39
Poll With Ack	0×56	86
Event Ack	0x57	87
Set Generator	0x4A	74
Set Modulus	0x4B	75
Request Key Exchange	0x4C	76
Set Baud Rate	0x4D	77

### **NV150 Event Table**

	Header code (hex)	dec
Slave Reset	0xF1	241
Read	0×EF	239
Note Credit	0×EE	238
Rejecting	0xED	237
Rejected	0xEC	236
Stacking	0xCC	204
Stacked	0xEB	235
Unsafe Jam	0xE9	233
Disabled	0xE8	232
Fraud Attempt	0×E6	230
Stacker Full	0×E7	231
Note Cleared From Front	0xE1	225
Note Cleared Into Cashbox	0xE2	226
Barcode Ticket Validated	0xE5	229
Barcode Ticket Ack	0xD1	209
Note Path Open	0×E0	224

# **FLATBED PRINTER Command Table**

	Header code (hex)	dec
Sync	0x11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Setup Request	0x05	5
Ssp Set Encryption Key	0x60	96
Get Real Time Clock Configuration	0x62	98
Set Real Time Clock	0x64	100
Get Real Time Clock	0x63	99
Ticket Print	0x70	112
Printer Configuration	0x71	113

# **FLATBED PRINTER Event Table**

	Header code (hex)	dec
Tickets Low	0xA0	160
Tickets Replaced	0xA1	161
Printer Head Removed	0xA2	162
Ticket Path Open	0xA3	163
Ticket Jam	0xA4	164
Ticket Printing	0xA5	165
Ticket Printed	0xA6	166
Ticket Printing Error	0xA8	168
Printer Head Replaced	0xA9	169
Ticket Path Closed	0xAA	170
No Paper	0xAB	171
Paper Replaced	0xAC	172
Ticket In Bezel At Startup	0xA7	167

### **NV12 Command Table**

	Header code (hex)	dec
Sync	0×11	17
Reset	0x01	1
Host Protocol Version	0x06	6
Poll	0x07	7
Get Serial Number	0x0C	12
Disable	0x09	9
Enable	0x0A	10
Get Firmware Version	0x20	32
Setup Request	0x05	5
Get Barcode Reader Configuration	0x23	35
Set Barcode Reader Configuration	0x24	36
Get Barcode Inhibit	0x25	37
Set Barcode Inhibit	0x26	38
Get Barcode Data	0x27	39
Ssp Set Encryption Key	0x60	96
Get Real Time Clock Configuration	0x62	98
Set Real Time Clock	0x64	100
Get Real Time Clock	0x63	99
Enable Tito Events	0x72	114
Ticket Print	0x70	112
Printer Configuration	0x71	113
	0x00	

### **NV12 Event Table**

	Header code (hex)	dec
Tickets Low	0xA0	160
Tickets Replaced	0xA1	161
Printer Head Removed	0xA2	162
Ticket Jam	0xA4	164
Ticket Printing	0xA5	165
Ticket Printed	0xA6	166
Ticket Printing Error	0xA8	168
Printer Head Replaced	0xA9	169
No Paper	0xAB	171
Print Halted	0xAE	174
Ticket In Bezel	0xAD	173
Paper Replaced	0xAC	172
Printed To Cashbox	0xAF	175
Ticket In Bezel At Startup	0xA7	167

Command	Code hex	Code decimal
Sync	0x11	17

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

SSP uses a system of sequence bits to ensure that packets have been received by the slave and the reply received by the host. If the slave receives the same sequence bit as the previous command packet then this is signal to re-transmit the last reply.

A mechanism is required to initially set the host and slave to the same sequence bits and this is done by the use of the SYNC command.

A Sync command resets the seq bit of the packet so that the slave device expects the next seq bit to be 0. The host then sets its next seq bit to 0 and the seq sequence is synchronised.

The SYNC command should be the first command sent to the slave during a session.

# Packet examples

#### Set seq bit to 1

Host transmit: **7F 80 01 11 65 82**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Reset	0x01	1

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Performs a software and hardware reset of the device.

After this command has been acknowledged with OK(0xF0), any encryption, baud rate changes, etc will be reset to default settings.

# Packet examples

No data parameters, sequence bit set and address 0

Host transmit: **7F 80 01 01 06 02**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Host Protocol Version	0x06	6

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

ITL SSP devices use a system of protocol levels to control the event responses to polls to ensure that changes would not affect systems with finite state machines unable to test for new events with non-defined data lengths.

If the device supports the requested protocol OK (OxFO) will be returned. If not then FAIL (OxF8) will be returned

#### Packet examples

The slave supports the protocol version 8

Host transmit: **7F 80 02 06 08 03 94**Slave Reply: **7F 80 01 F0 23 80** 

Host protocol version 9 not supported

Host transmit: **7F 80 02 06 09 06 14**Slave Reply: **7F 80 01 F8 10 00** 

Command	Code hex	Code decimal
Poll	0x07	7

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

This command returns a list of events occured in the device since the last poll was sent.

The SSP devices share some common events and have some unique events of their own. See event tables for details for a specific device.

### Packet examples

Poll command returning device reset and disabled response

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 F1 F8 DC 0C** 

Event response note credit channel 1 and note stacked

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 04 F0 EE 01 EB B9 48** 

Command	Code hex	Code decimal
Get Serial Number	0x0C	12

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

This command returns a 4-byte big endian array representing the unique factory programmed serial number of the device.

### Packet examples

The device responds with 4 bytes of serial number data. In this case, the serial number is 01873452 = 0x1c962c. The return array is formatted as big endian (MSB first).

Host transmit: **7F 80 01 0C 2B 82** 

Slave Reply: **7F 80 05 F0 00 1C 96 2C D4 97** 

Command	Code hex	Code decimal
Disable	0x09	9

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Disabled the slave device from operation.

For example, this command would block a banknote validator from allowing any more banknotes to be entered.

For most SSP devices, the default state is to be disabled after reset.

#### Packet examples

Single byte command with no parameters

Host transmit: **7F 80 01 09 35 82**Slave Reply: **7F 80 01 F0 23 80** 

NV11 when note float is jammed/disconnected responds COMMAND\_CANNOT\_BE\_PROCESSED

Host transmit: **7F 80 01 09 35 82**Slave Reply: **7F 80 01 F5 3D 80** 

Command	Code hex	Code decimal
Enable	0x0A	10

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

This command will enable the SSP device for normal operation. For example, it will allow a banknote validator to commence validating banknotes entered into it's bezel.

### Packet examples

Single byte command with no parameters

Host transmit: **7F 80 01 0A 3F 82**Slave Reply: **7F 80 01 F0 23 80** 

NV11 when note float is jammed/disconnected responds COMMAND\_CANNOT\_BE\_PROCESSED

Host transmit: **7F 80 01 0A 3F 82**Slave Reply: **7F 80 01 F5 3D 80** 

Command	Code hex	Code decimal
Get Firmware Version	0x20	32

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Returns a variable length ASCII array containg the full firmware version of the attached device.

# Packet examples

In this example, the firmware version of the device is: NV02004141498000

Host transmit: **7F 80 01 20 C0 02** 

Slave Reply: **7F 80 11 F0 4E 56 30 32 30 30 34 31 34 31 34 39 38 30 30 30 DE 55** 

ascii: . N V 0 2 0 0 4 1 4 1 4 9 8 0 0 0

Command	Code hex	Code decimal
Get Dataset Version	0x21	33

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, TEBS	optional

Returns a varibale length ASCII array giving the installed dataset version of the device.

Packet examples

This example shows a device with dataset version EUR01610.

Host transmit: **7F 80 01 21 C5 82** 

Slave Reply: **7F 80 09 F0 45 55 52 30 31 36 31 30 B8 2A** ascii: **. E U R 0 1 6 1 0** 

Command	Code hex	Code decimal
Set Inhibits	0x02	2

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, SMART SYSTEM, TEBS	optional

Sets the channel inhibit level for the device. each byte sent represents 8 bits (channels of inhibit).

Nv200 has the option to send 2,3,or 4 bytes to represent 16,24, or 64 channels, the other BNV devices have the option of sending 1 or 2 bytes for 8 or 16 channel operation.

Set the bit low to inhibit all note acceptance on that channel, high to allow note acceptance.

#### Packet examples

Set channels 1-3 enabled, 4-16 inhibited

Host transmit: **7F 80 03 02 07 00 2B B6**Slave Reply: **7F 80 01 F0 23 80** 

All channels enabled

Host transmit: **7F 80 03 02 FF FF 25 A4**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Display On	0x03	3

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

Allows the host to control the illumination of the bezel. Send this command to show bezel illumination when the device is enabled for banknote validation. (This is the default condition at reset).

Note that the validator will still override the illumination of the bezel, i.e. the bezel will **not** be illuminated if the device is **not enabled** even if this command is sent.

### Packet examples

Single byte command with no parameters.

Host transmit: **7F 80 01 03 09 82**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Display Off	0×04	4

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

Allows the host to control banknote validator bezel illumination. Use this command to disable illumination whne the validator is enabled for note entry.

# Packet examples

Single byte command with no parameters

Host transmit: **7F 80 01 04 18 02**Slave Reply: **7F 80 01 F0 23 80** 



Command	Code hex	Code decimal
Setup Request	0×05	5

Implemented on	Encryption Required
BV100, BV20, BV50, FLATBED PRINTER, NV10USB, NV11, NV12, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Request the set-up configuration of the device. Gives details about versions, channel assignments, country codes and values.

Each device type has a different return data format. Please refer to the table information for each individual device.

SMART Ticket/Coupon Printer Response

Smart Ticket Data	Response Offset	Size	Notes
Unit Type	0	1	0x08 = SMART Ticket, 0x0B = Coupon Printer
Firmware Version	1	4	Ascii data of device firmware (eg 0123)
Cutter Enabled	5	1	(0 for disabled)
Tab enabled status	6	1	(0 for disabled)
Reverse validation enabled status	7	1	(0 for disabled)
Font pack code (ASCII)	8	3	e.g. FP1
Printer type	11	1	Printer Type: 0x0 for Fan Fold, 0x1 Paper Roll (Cutter fitted)
SD card fitted status	12	1	(1 for detected)
Printer darkness/quality setting	13	1	value between 0 - 3
SSP Protocol Version	14	1	

SMART Hopper/SMART System

Smart Hopper Data	Response Offset	Size	Notes
Unit Type	0	1	0x03 = SMART Hopper, 0x09 = SMART System.
Firmware Version	1	4	Ascii data of device firmware (e.g. 0123).
Country Code	5	3	ASCII country code (e.g. EUR).
Protocol Version	8	1	The current protocol version set for this device.
Number of Coin Channels	9	1	The number of coin denominations in this device dataset n.
Coin Values	10	n * 2	2 byte each value for the coin denominations (e.g. $0.05$ coin = $0x05$ , $0x00$ ).
Country Codes for Values	10 + (n * 2)	n * 3	3 byte ASCII country code for each coin denomination. This is given only when protocol version >= 6.

#### Packet examples

This example shows the data returned for a BNV with GBP dataset, firmware version  $1.00,\,3$  channels GBP 5, GBP 10, GBP 20

Host transmit: **7F 80 01 05 1D 82** 

Slave Reply: 7F 80 17 F0 00 30 31 30 30 47 42 50 00 00 01 03 05 0A 14 02 02 02 40 00

00 05 61 81

. .

Command	Code hex	Code decimal
Reject	0x08	8

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

After a banknote validator device reports a valid note is held in escrow, this command may be sent to cause the banknote to be rejected back to the user.

Packet examples

Single byte command with no parameters

Host transmit: **7F 80 01 08 30 02**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Uint Data	0x0D	13

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

A command to return version information about the connected device to the format described in the table below:

byte offset	function	size
0	Generic OK Response (OxF0)	1
1	Unit type: see Uint Type Table for codes	1
2	Firmware version (4 byte ASCII)	4
6	Dataset country (3 byte ASCII)	3
9	Value multiplier	3
12	Protocol version	1

# Packet examples

This is a response example for a banknote validator EUR 5,10,20 version 3.00 protocol version 7

Host transmit: **7F 80 01 0D 2E 02** 

Slave Reply: **7F 80 0D F0 00 30 33 30 30 45 55 52 01 00 00 07 01 85** 

ascii: . . 0 3 0 0 E U R . . . .

Command	Code hex	Code decimal
Channel Value Data	0x0E	14

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

Returns channel value data for a banknote validator. Note that this will differ depending on the protocl version used/supported.

#### For protocol versions less than 6:

byte offset	function	size
0	Generic Ok (0xF0)	1
1	Highest channel in set 1-16 (n)	1
2 : (2 + n)	A byte value for each channel - gives the total channel value when nultiplied by the value multiplier. Zero value shows that the channel is not used	n

# For protocol versions greater or equal to 6:

byte offset	function	size
0	Generic Ok (0xF0)	1
1	Highest channel in set 1-16 (n)	1
2	A byte value for each channel - gives the total channel value when nultiplied by the value multiplier. Zero value shows that the channel is not used	n
2 + n	3 byte for each ASCII country code in set	3 * n
(2 + n) + (3*n)	4 byte value for each denomination	4 * n

#### Packet examples

This example shows a response for notes in channels 1,2,4,6,7 when in protocol version 5

Host transmit: **7F 80 01 0E 24 02** 

Slave Reply: 7F 80 09 F0 07 05 0A 00 14 00 32 64 BC DA

This example shows a response for notes in channels 1,2,4,6,7 when in protocol version 6

Host transmit: **7F 80 01 0E 24 02** 

Slave Reply: 7F 80 3C F0 07 00 00 00 00 00 00 45 55 52 45 55 52 45 55 52 00 45 55

52 45 55 52 00 45 55 52 45 55 52 05 00 00 00 0A 00 00 00 00 00 00 14

00 00 00 00 00 00 00 32 00 00 00 64 00 00 00 D0 DF

ascii:

Command	Code hex	Code decimal
Channel Security Data	0x0F	15

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

Command which returns a number of channels byte (the highest channel used) and then 1 to n bytes which give the security of each channel up to the highest one, a zero indicates that the channel is not implemented.

(1 = low, 2 = std, 3 = high, 4 = inhibited).

# Packet examples

In this example a validator has notes in channels 1,2,4,6,7 all at standard security.

Host transmit: **7F 80 01 0F 21 82** 

Slave Reply: 7F 80 09 F0 07 02 02 00 02 00 02 94 84

Command	Code hex	Code decimal
Last Reject Code	0x17	23

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	optional

Returns a one byte code representing the reason the BNV rejected the last note. See  $\frac{\text{Reject}}{\text{Code}}$  table for details.

Packet examples

Note rejected due to a request by the host

Host transmit: **7F 80 01 17 71 82**Slave Reply: **7F 80 02 F0 08 0C 20** 

Command	Code hex	Code decimal
Hold	0x18	24

Implemented on	Encryption Required
BV100, BV20, BV50, NV10USB, NV150, NV200, NV9USB, TEBS	optional

SSP banknote validators include a poll-time-out of 10 seconds. If a new poll is not received withinh this time, then a note held in escrow will be rejected.

The host may require that the note is continued to be held, but a new poll would accept the note.

Send this command (or any other command but not a simple poll) to reset the timeout and continue to hold the note in escrow until such time as either a reject or poll command is sent.

#### Packet examples

Returns OK if note is in escrow

Host transmit: **7F 80 01 18 53 82**Slave Reply: **7F 80 01 F0 23 80** 

Returns FAIL if no note in escrow

Host transmit: **7F 80 01 18 53 82**Slave Reply: **7F 80 01 F8 10 00** 

Command	Code hex	Code decimal
Get Barcode Reader Configuration	0x23	35

Implemented on	Encryption Required
NV12, NV150, NV200, SMART PAYOUT, TEBS	optional

Returns the set-up data for the device bar code readers.

Responds (if supported) with five bytes of data formatted as:

byte	function	size
0	Generic OK	1
1	Bar code hardware status (0x00 = none, 0x01 = Top reader fitted, 0x02 = Bottom reader fitted, 0x03 = both fitted)	1
2	Readers enabled (0x00 = none, 0x01 = top, 0x02 = bottom, 0x03 = both)	1
3	Bar code format $(0x01 = Interleaved 2 of 5)$	1
4	Number of characters (Min 6 max 24)	1

# Packet examples

Response for device with top and bottom readers fitted, both enabled, interleaved 2 of 5 with  $18\ chars$ 

Host transmit: **7F 80 01 23 CA 02** 

Slave Reply: **7F 80 05 F0 03 03 01 12 D5 58** 

Command	Code hex	Code decimal
Set Barcode Reader Configuration	0x24	36

Implemented on	Encryption Required
NV12, NV150, NV200, SMART PAYOUT, TEBS	optional

This command allows the host to set-up the bar code reader(s) configuration on the device.

Three bytes of data define the configuarion:

byte	function	size
0	0x00 Enable none, $0x01$ enable top, $0x02$ = enable bottom, $0x03$ = enable both	1
1	Bar code format $(0x01 = Interleaved 2 of 5)$	1
2	Number of characters (Min 6 Max 24)	1

Packet examples

Enable both readers with format interleaved 1 of 5 for 18 characters.

Host transmit: **7F 80 04 24 03 01 12 EC D7** 

Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Get Barcode Inhibit	0x25	37

Implemented on	Encryption Required
NV12, NV150, NV200, SMART PAYOUT, TEBS	optional

Command to return the current bar code/currency inhibit status.

If supported, responds with 1 byte bit regsiter data:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
not used 1	not used 1	not used 1	not used 1	not used 1	not used 1	barcode read enable (0 = enabled)	currency read enable (0 = enabled)

# Packet examples

A response from a device with bar code disabled, currency enabled

Host transmit: **7F 80 01 25 DE 02** Slave Reply: **7F 80 02 FE 38 22** 

Command	Code hex	Code decimal
Set Barcode Inhibit	0x26	38

Implemented on	Encryption Required
NV12, NV150, NV200, SMART PAYOUT, TEBS	optional

Sets up the bar code inhibit status register.

Send a single data bit register byte formatted as:

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
not used 1	not used 1	not used 1	not used 1	not used 1	not used 1	barcode read enable (0 = enabled)	currency read enable (0 = enabled)

# Packet examples

Shows a request to enabled bar code, disable currency on the device

Host transmit: **7F 80 02 26 FD 3E D6**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Get Barcode Data	0x27	39

Implemented on	Encryption Required
NV12, NV150, NV200, SMART PAYOUT	optional

Command to obtain last valid bar code ticket data, send in response to a <u>bar code ticket validated</u> event. This command will return a variable length data steam, a generic response (OK) followed by a status byte, a bar code data length byte, then a stream of bytes of the ticket data in ASCII.

Response is formatted as:

byte	function	size
0	Generic OK	1
1	Status (0=no valid data, 1=ticket in escrow, 2=ticket stacked, 3=ticket rejected)	1
2	data length	1
3	variable length ASCII array of bar code data	v

### Packet examples

shows ticket is in escrow with data length 6 and data 123456.

Host transmit: 7F 80 01 27 D1 82

Command	Code hex	Code decimal
Configure Bezel	0x54	84

Implemented on	Encryption Required
NV200, SMART PAYOUT, TEBS	optional

This command allows the host to configure a supported BNV bezel.

In NV200 firmware 4.28 an extra optional byte was added to specify the bezel type.

# Command format:

byte	function	size
0	red pwm (0-255)	1
1	green pwm (0-255)	1
2	blue pwm (0-255)	1
3	Config 0 for volatile,1 - for non-volatile.	1
4	Optional Bezel Type (0 - Enable Solid Colour, 1 - Enable Flashing Colour, 2 - Disable Colour)	1

# Packet examples

In this example, we want a red bezel fixed to EEPROM.

Host transmit: **7F 80 05 54 FF 00 00 01 48 DC** 

Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Poll With Ack	0x56	86

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	yes

A command that behaves in the same way as the Poll command but with this command, the specified events will need to be acknowledged by the host using the EVENT ACK command (0x56).

The events will repeat until the EVENT ACK command is sent and the BNV will not allow any further note actions until the event has been cleared by the EVENT ACK command. If this command is not supported by the slave device, then generic response 0xF2 will be returned and standard poll command (0x07) will have to be used.

#### Packet examples

Command	Code hex	Code decimal
Event Ack	0x57	87

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	a yes

This command will clear a repeating Poll ACK response and allow further note operations.

Packet examples

Host transmit: **7F 80 01 57 F2 03**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Set Denomination Route	0x3B	59

Implemented on	Encryption Required
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

This command will configure the denomination to be either routed to the cashbox on detection or stored to be made available for later possible payout.

Note on protocol versions: For protocol versions less than 6 a value only data array is sent. For protocol version greater or equal to 6, a 3 byte country code is also sent to allow mulit-currency functionality to the payout.

Please note that there exists a difference in the data format between SMART Payout and SMART Hopper for protocol versions less than 6. In these protocol versions the value was determined by a 2 byte array rather than 4 byte array for SMART Hopper.

For NV11 devices the host must send the required note value in the same form that the device is set to report by (see Set Value Reporting Type command).

Protocol version less than 6 command format:

byte	function	
0	requested route (0 = payout, 1= cashbox)	1
1	value (2 bytes for hopper, 4 bytes for others)	2 or 4

Protocol version greater of equal to 6 format:

byte	function	size
0	requested route (0 = payout, 1= cashbox)	1
1	value of requested denomination to route (4 byte integer)	4
5	ASCII country code of requested denomination	3

With note payouts, the device responds with COMMAND CANNOT BE PROCESSED and an error byte for request failure:

Error	code
No payout connected	1
Invalid currency detected	2
Payout device failure	3

# Packet examples

An example of a request to route a  $10c\ EUR$  coin to be stored for payout using protocol version 6

Host transmit: 7F 80 09 3B 00 0A 00 00 45 55 52 08 43

Slave Reply: **7F 80 01 F0 23 80** 

Example command with error response Invalid currency detected

Host transmit: **7F 80 09 3B 00 0A 00 00 00 45 55 52 08 43** 

Slave Reply: **7F 80 02 F5 02 30 3E** 

Command	Code hex	Code decimal
Get Denomination Route	0x3C	60

Implemented on	Encryption Required
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	<b>△</b> yes

This command allows the host to determine the route of a denomination.

#### Note protocol versions:

For protocol versions less than 6 a value only data array is sent. For protocol version greater or equal to 6, a 3 byte country code is also sent to allow multi-currency functionality to the payout.

# Please note that there exists a difference in the data format between SMART Payout and SMART

# Hopper for protocol versions less than 6. In these protocol versions the value was determined by a 2 byte array rather than 4 byte array

For NV11 devices the host must send the required note value in the same form that the device is set to report by (see Set Value Reporting Type command).

Protocol version less than 6 command format:

byte	function	
0	value (2 bytes for hopper, 4 bytes for others)	2 or 4

Protocol version greater of equal to 6 format:

byte	function	
0	value of requested denomination to route (4 byte integer)	4
4	ASCII country code of requested denomination	3

The device responds with a data byte representing the current route of the denomination.

byte	function	
0	Generic OK	1
1	Route (0 = recycle for payout,1 = system cashbox)	1

With note payouts, the device responds with COMMAND CANNOT BE PROCESSED and an error byte for request failure:

Error	code
No payout connected	1
Invalid currency detected	2
Payout device failure	3

# Packet examples

This example shows a request to obtain the route of EUR 5.00 note in protocol version 6. Returns 0 for payout.

Host transmit: 7F 80 08 3C F4 01 00 00 45 55 52 2F 0E

Slave Reply: **7F 80 02 F0 00 3F A0** 

Command	Code hex	Code decimal
Payout Amount	0x33	51

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

A command to set the monetary value to be paid by the payout unit.

This command was expanded after and including protocol version 6 to include country codes and payout test option.

Command format protocol version less than 6:

byte	byte function	
0	payout value (4 byte integer of the full penny amount)	4

Command format protocol greater than or equal to 6:

byte	function	
0	payout value (4 byte integer of the full penny amount)	4
4	ASCII country code of currency to pay	3
7	Option byte (TEST_PAYOUT_AMOUT 0x19, PAYOUT_AMOUNT 0x58),	1

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the error code.

Error	Code
Not enough value in device	1
Cannot pay exact amount	2
Device busy	3
Device disabled	4

# Packet examples

Shows a request to payout EUR 5.00 using protocol version 4

Host transmit: **7F 80 05 33 F4 01 00 00 32 50** 

Slave Reply: **7F 80 01 F0 23 80** 

Shows an example is a request to payout EUR 5.00 in protocol version 6 with commit option.

Host transmit: 7F 80 09 33 F4 01 00 00 45 55 52 58 C3 EE

Slave Reply: **7F 80 01 F0 23 80** 

Shows an example is a request to payout EUR 5.00 in protocol version 6 failed due to cannot pay exact amount

Host transmit: 7F 80 09 33 F4 01 00 00 45 55 52 58 C3 EE

Slave Reply: **7F 80 02 F5 02 30 3E** 

Command	Code hex	Code decimal
Get Denomination Level	0x35	53

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	optional

This command returns the level of a denomination stored in a payout device as a 2 byte value.

This command was expanded in protocol version 6 to include country codes for multicurrency functionality.

Protocol version 5 command format:

byte	function	size
0	4 byte value of denomination requested	4

Protocol version 6 and greater command format:

byte	function	size
0	4 byte value of denomination requested	4
4	ASCII country code of denomination required	3

#### Packet examples

Example shows a request to find the amount of 0.10c coins in protocol version 5. Returns a level of 100

Host transmit: **7F 80 05 35 0A 00 00 00 1E 49**Slave Reply: **7F 80 03 F0 64 00 C5 F0** 

Shows a request to find the level of EUR 5.00 notes using protocol version 6. Returns 12.

Host transmit: **7F 80 08 35 F4 01 00 00 45 55 52 19 9E** 

Slave Reply: **7F 80 03 F0 0C 00 C3 80** 

If the denomination is not in the device, it will respond with COMMAND CANNOT BE  $\ensuremath{\mathsf{PROCESSED}}$ 

Host transmit: **7F 80 08 35 F4 01 00 00 45 55 52 19 9E** 

Slave Reply: **7F 80 01 F5 3D 80** 

Command	Code hex	Code decimal
Set Denomination Level	0x34	52

Implemented on	Encryption Required
SMART HOPPER, SMART SYSTEM	<b>△</b> yes

A command to increment the level of coins of a denomination stored in the hopper. The command is formatted with the command byte first, amount of coins to add as a 2-byte little endian, the value of coin as 2-byte little endian and (if using protocol version 6) the country code of the coin as 3 byte ASCII. The level of coins for a denomination can be set to zero by sending a zero level for that value.

This command was updated when using version 6 and greater to allow for larger 4 byte coin values and country codes.

Protocol version less than 6:

byte	function	size
0	number of coins to add to level (0 will clear the level)	2
2	value fo denimonation to set	2

Protocol version great or equal to 6:

byte	function	size
0	number of coins to add to level (0 will clear the level)	2
2	value of denomination to set	4
6	ASCII country code of denomination	3

Packet examples

Example to increase the level of .50c coin by 20 using protocol version 5

Host transmit: 7F 80 05 34 14 00 32 00 63 FD

Slave Reply: **7F 80 01 F0 23 80** 

Example to increase the level of EUR 1.00 coins by 12 on a device set with protocol version 6

Host transmit: 7F 80 0A 34 0C 00 64 00 00 00 45 55 52 C7 28

Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Halt Payout	0x38	56

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

A command to stop the execution of an existing payout. The device will stop payout at the earliest convenient place and generate a Halted event giving the value paid up to that point.

Packet examples

Ok response for halt command accepted.

Host transmit: **7F 80 01 38 90 02**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Float Amount	0x3D	61

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	<b>△</b> yes

A command to float the payout unit to leave a requested value of money, with a requested minimum possible payout level. All monies not required to meet float value are routed to cashbox. Using protocol version 6, the host also sends a pre-test option byte (TEST\_FLOAT\_AMOUT 0x19, FLOAT\_AMOUNT 0x58), which will determine if the command amount is tested or floated. This is useful for multi-payout systems so that the ability to pay a split down amount can be tested before committing to actual float.

This command differs between Smart Hopper and Smart Payout. The minimum payout amount is a different size.

This command was expanded after and including protocol version 6 to include country codes and payout test option.

Command format protocol version less than 6:

#### Smart Hopper:

byte	function	size
0	value of minimum payout to remain	2
2	float value (4 byte integer of the full penny amount)	4

#### Smart Payout:

byte	function	
0	value of minimum payout to remain	
4	float value (4 byte integer of the full penny amount)	

Command format protocol greater than or equal to 6:

Smart Hopper:

byte	function	
0	value of minimum payout to remain	
2	payout value (4 byte integer of the full penny amount)	
6	ASCII country code of currency to pay	
9	Option byte (TEST_FLOAT_AMOUT 0x19, FLOAT_AMOUNT 0x58),	

# Smart Payout:

byte	function	
0	value of minimum payout to remain	
4	payout value (4 byte integer of the full penny amount)	
8	ASCII country code of currency to pay	
11	Option byte (TEST_FLOAT_AMOUT 0x19, FLOAT_AMOUNT 0x58),	

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the error code.

Error	Code
Not enough value in device	1
Cannot pay exact amount	2
Device busy	3
Device disabled	4

# Packet examples

Example to request to float to a value of 100.00 leaving a min possible payout of 0.50c for protocol version 5

Host transmit: 7F 80 07 3D 32 00 10 27 00 00 1D 1C

Slave Reply: **7F 80 01 F0 23 80** 

In protocol version greater than 6, we add a 3 byte ascii country code and a test or commit data byte. In this example a request to float to a value of EUR 100.00 leaving a min possible payout of 0.50c

Host transmit: 7F 80 0B 3D 32 00 27 10 00 00 45 55 52 58 A7 DA

Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Get Min Payout	0x3E	62

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	optional

A command to request the minimum possible payout amount that this device can provide.

For protocol versions less than 6, no parameters are sent.

For protocol version 6 or greater, we add the 3 byte country code of the country we are requesting.

# Packet examples

Example for protocol version 5 returning min payout of 200

Host transmit: **7F 80 01 3E 84 02** 

Slave Reply: **7F 80 05 F0 C8 00 00 00 A7 C2** 

Protocol version 6 example returning a min payout value of 5.00 EUR

 Host transmit:
 7F
 80
 04
 3E
 45
 55
 52
 14
 E3

 ascii:
 .
 .
 .
 .
 .
 E
 U
 R
 .
 .
 .

 Slave Reply:
 7F
 80
 75
 F0
 F4
 70
 00
 00
 BA
 72

ascii:

Command	Code hex	Code decimal
Set Coin Mech Inhibits	0x40	64

Implemented on	Encryption Required	
SMART HOPPER, SMART SYSTEM	<b>△</b> yes	

This command is used to enable or disable acceptance of individual coin values from a coin acceptor connected to the hopper.

Protocol versions less than 6:

byte	function	
0	Requested inhibit state (0 =inhibit,1=enable)	
1	coin value (2 byte integer)	

Protocol versions greater or equal to 6:.

byte	function	
0	Requested inhibit state (0 =inhibit,1=enable)	
1	coin value (2 byte integer)	
3	ASCII country code of value	

# Packet examples

Example we want to enable acceptance of EUR  $0.50c\ coins\ in\ protocol\ version\ 6.$ 

Command	Code hex	Code decimal
Payout By Denomination	0x46	70

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

A command to payout the requested quantity of individual denominations.

# Requires Protocol Version 6 or above. Attempting to use the command with an earlier protocol version will generate a response 0xF4 (parameter out of range).

The quantities of denominations to pay are sent as a 2 byte little endian array; the money values as 4-byte little endian array and the country code as a 3-byte ASCII array.

The host also adds an option byte to the end of the command array (TEST\_PAYOUT\_AMOUT 0x19 or PAYOUT\_AMOUNT 0x58). This will allow a pre-test of the ability to payout the requested levels before actual payout executes.

#### Command format:

byte	function	
0	the number of individual requests in this command (max 20)	1
1	the number to pay	2
3	the denomination value	4
7	the denomination ASCII country code	3
10	repeat block for each required denomination	
	The option byte (TEST_FLOAT_AMOUT 0x19 or FLOAT_AMOUNT 0x58).	1

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the error code.

Error	Code
Not enough value in device	1
Cannot pay exact amount	2
Device busy	3
Device disabled	4

#### Packet examples

Example - A hopper unit has stored 100 x 0.10 EUR, 50 x 0.20 EUR, 30 x 1.00 EUR, 10 x 1.00 GBP, 50 x 0.50 GBP and the host wishes to payout to 5 x 1.00 EUR, 5 x 0.10 EUR, 3 x 1.00 GBP and 2 x 0.50 GBP.

Host transmit: **7F 80 27 46 04 04 00 64 00 00 00 45 55 52 05 00 0A 00 00 45 55 52 03** 

00 64 00 00 00 47 42 50 02 00 32 00 00 00 47 42 50 58 94 B7

ascii: . ' F . . . d . . . E U R . . . . . . E U R .

. d . . . G B P . . 2 . . . G B P X . .

Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Float By Denomination	0x44	68

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

A command to float (leave in device) the requested quantity of individual denominations.

Requires Protocol Version 6 or above. Attempting to use the command with an earlier protocol version will generate a response 0xF4 (parameter out of range).

The quantities of denominations to leave are sent as a 2 byte little endian array; the money values as 4-byte little endian array and the country code as a 3-byte ASCII array. The host also adds an option byte to the end of the command array (TEST\_PAYOUT\_AMOUT 0x19 or PAYOUT\_AMOUNT 0x58). This will allow a pre-test of the ability to float to the requested levels before actual float executes.

#### Command format:

byte	function	
0	the number of individual requests in this command (max 20)	1
1	the number required to leave in device (little endian array)	2
3	the denomination value (little endian array)	4
7	the denomination ASCII country code	3
10	repeat block for each required denomination	
last	The option byte (TEST_FLOAT_AMOUT 0x19 or FLOAT_AMOUNT 0x58).	1

For request failure, the device responds with COMMAND CANNOT BE PROCESSED and a data byte showing the error code.

Error	Code
Not enough value in device	1
Cannot pay exact amount	2
Device busy	3
Device disabled	4

Events used to indicate progress:

While floating is being carried out, the Floating and Floated events are used to keep the host informed.

Packet examples

Command	Code hex	Code decimal
Empty All	0x3F	63

Implemented on	Encryption Required
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

This command will direct all stored monies to the cash box without reporting any value and reset all the stored counters to zero. See Smart Empty command to record the value emptied.

A poll command during this process will respond with Emptying and Empty events

Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 3F 81 82**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Set Options	0×50	80

Implemented on	Encryption Required	
SMART HOPPER, SMART SYSTEM	<u></u> yes	

Description
-------------

The host can set the following options for the Smart Hopper. These options do not persist in memory and after a reset they will This command is valid only when using protocol version 6 or greater.

Table below shows the available options for the SMART Hopper. The command data is formatted as a 2 byte register REG\_0 and REG\_1  $\,$ 

# Reg\_0 bits and their meaning

Bit	parameter		
0	Pay Mode	Default: Split by highest value (0x00) The device will attempt to payout a requested value by starting from the highest to the lowest coins available. This mode will payout the minimum number of coins possible.  Free pay (0x01). The device will payout a coin as it passes its discriminator system if it fits into the current payout value and will leave enough of other coins to payout the rest of the value. This may give a faster payout but could result in a large number of coins of small denominations paid out.	
1	Level Check	Disabled (0x00). The device will not refer to the level counters when calculating if a payout value can be made. Enabled (0x01) (Default state after reset). The device will check the level counters and accept or refuse a payout request based on levels and/or split of available levels.	
2	Motor Speed	Low speed (0x00). Payouts run at a lower motor speed. High Speed (Default state after reset) (0x01). The motors run at max speed for payouts.	
3	Cashbox Pay Active	' Idescribed in hit (). It Bit 3 is set then coins routed to the cashbox will be use	
4	Route 0 level coins to cashbox Set to 1 means that any coins detected with a level setting of 0 will be p to the cashbox, even if it is routed to the payout.		
5	High Efficiency Split Default: Set to 0x01 to enable a more efficient, smarter coin payout algorithm which will tend to use coins which have higher level counts - thus speeding up the payout process.		
6	Unknown to Set to 1 means any unknown coins will be paid out during Smart Empty (otherwise they will be routed to cashbox).		
7	Value Added Set to 0 for coin added event set to 1 for value added event.		

Reg\_1 bits and their meaning

Bit	parameter	
Cod	0 Reject Events Set to 1 gives reject event 0xBA Coin Rejected. Meaning	
1	Reject Events Full	Set to 1 gives reject event 0xBA with coin value if known.
2	Smart Empty Route	Set to 1 will route coins to payout when a Smart Empty command is received.
3	N/A	Set to 0.
4	N/A	Set to 0.
5	N/A	Set to 0.
6	N/A	Set to 0.
7	N/A	Set to 0.

# Response

When responding to this command, the Smart Hopper returns a byte which indicates the current operational mode as follows:

# **Set Options: Response Codes**

Code	Meaning
0xFC	Highest split, use coins routed to cashbox in the split
0xFD	Free pay, use coins routed to cashbox in the split
0xFE	Highest split
0xFF	Free pay

# Packet examples

The example shows a request to turn off level check, run at high speed and split by highest value.

Host transmit: **7F 80 03 50 04 00 40 38**Slave Reply: **7F 80 02 F0 FE 38 22** 

Command	Code hex	Code decimal
Get Options	0x51	81

Implemented on	Encryption Required	
SMART HOPPER, SMART SYSTEM	a yes	

This command returns 2 option register bytes described in <u>Set Options</u> command.

Packet examples

Command	Code hex	Code decimal
Coin Mech Global Inhibit	0x49	73

Implemented on	Encryption Required	
SMART HOPPER, SMART SYSTEM	<b>△</b> yes	

This command allows the host to enable/disable the attached coin mech in one command rather than by each individual value with previous firmware versions. Send this command and one Mode data byte: Data byte =0x00 - mech disabled. Date byte =0x01 - mech enabled.

# Packet examples

In this example we are sending a command to enable the coin mech.

Host transmit: **7F 80 02 49 01 33 36**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Smart Empty	0x52	82

Implemented on	Encryption Required
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	🖺 yes

Empties payout device of contents, maintaining a count of value emptied. The current total value emptied is given is response to a poll command. All coin counters will be set to 0 after running this command. Use <u>Cashbox Payout Operation Data</u> command to retrieve a breakdown of the denominations routed to the cashbox through this operation.

# Packet examples

Host transmit: **7F 80 01 52 EC 03**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Cashbox Payout Operation Data	0x53	83

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	<b>△</b> yes

Can be sent at the end of a SMART Empty, float or dispense operation. Returns the amount emptied to cashbox from the payout in the last dispense, float or empty command.

# Response format:

byte	function	size
0	generic OK	
1	number of denominations in report	1
2	qty of denomination	2
4	denomination value	4
8	denomination country (ASCII)	3
	repeated above block for each denomination	
	qauntity of unknown	4

Packet examples

Command	Code hex	Code decimal
Get All Levels	0x22	34

Implemented on	Encryption Required
SMART HOPPER, SMART PAYOUT, SMART SYSTEM	optional

Use this command to return all the stored levels of denominations in the device (including those at zero level).

This gives a faster response than sending each individual denomination level request.

Response data consists of blocks of nine bytes data for each denimonation in the device:

byte	function	
0	Generic OK	
1	number of denominations in the device	
2	level of denomination stored	
4	denomination value (4 byte little endian integer)	4
7	denomination code (3 Byte ASCII)	
10	Repeat for each denomination	

# Packet examples

In this example, we have a device coin dataset of EURO s with 20c,50c,1 EUR and 2 EUR. It currently has 100 x 20c, 65 x 50x, 0 x 1 EUR and 12 x 2 EUR.

Host transmit: **7F 80 01 22 CF 82** 

Slave Reply: 7F 80 26 F0 04 64 00 14 00 00 00 45 55 52 41 00 32 00 00 00 45 55 52 00

 $00\ 64\ 00\ 00\ 00\ 45\ 55\ 52\ 0C\ 00\ C8\ 00\ 00\ 00\ 45\ 55\ 52\ 84\ D0$ 

Command	Code hex	Code decimal
Get Counters	0x58	88

Implemented on	Encryption Required
NV11, NV9USB, SMART PAYOUT, SMART SYSTEM	optional

A command to return a global note activity counter set for the slave device. The response is formatted as in the table below and the counter values are persistent in memory after a power down- power up cycle.

These counters are note set independent and will wrap to zero and begin again if their maximum value is reached. Each counter is made up of 4 bytes of data giving a max value of 4294967295.

# Response format:

byte	function	
0	Generic OK	1
1	Number of counters in set	1
2	Stacked	4
6	Stored	4
10	Dispensed	4
14	Transferred to stack	
18	Rejected	4

#### Packet examples

Command	Code hex	Code decimal
Reset Counters	0x59	89

Implemented on	Encryption Required
NV11, NV9USB, SMART PAYOUT, SMART SYSTEM	optional

Resets the note activity counters described in Get Counters command to all zero values.

Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 59 D5 83**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Set Refill Mode	0x30	48

Implemented on	Encryption Required
SMART PAYOUT	optional

A command sequence to set or reset the facility for the payout to reject notes that are routed to the payout store but the firmware determines that they are un-suitable for storage. In default mode, they would be re-routed to the stacker. In refill mode they will be rejected from the front of the NV200.

#### Packet examples

This example show the sequence of command bytes to set the mode.

Host transmit: **7F 80 06 30 05 81 10 11 01 52 F5** 

Slave Reply: **7F 80 01 F0 23 80** 

This sequence will un-set the mode for normal operation.

Host transmit: **7F 80 06 30 05 81 10 11 00 57 75** 

Slave Reply: **7F 80 01 F0 23 80** 

To read the current refill mode send this sequence: Returns 1 byte: 0x00 the option is not set, 0x01 the option is set. This shows a return with option set.

Host transmit: 7F 80 05 30 05 81 10 01 94 EE

Slave Reply: **7F 80 02 F0 01 3A 20** 

Command	Code hex	Code decimal
Get Note Positions	0x41	65

Implemented on	Encryption Required
NV11	<u></u> yes

This command will return the number of notes in the Note Float and the value in each position. The way the value is reported is specified by the Set Reporting Type command. The value can be reported by its value or by the channel number of the bill validator. The first note in the table is the first note that was paid into the Note Float.

The Note Float is a LIFO system, so the note that is last in the table is the only one that is available to be paid out or moved into the stacker.

Data response format when Report by value is set:

byte	function	size
0	Generic OK	1
1	Number of notes stored	1
2	Value of note in slot 1	4
6	Value of note in slot 2	4
10	Value of note in slot 3	4
	continues for how many notes stored	

Data response format when Report by channel is set:

byte	function	size
0	Generic OK	1
1	Number of notes stored	1
2	Channel of note in slot 1	1
3	Channel of note in slot 2	1
4	Channel of note in slot 3	1
	continues for how many notes stored	

If the currency in the validator does not match the country of the notes stored, then this command will respond with COMMAND CANNOT BE PROCESSED and error byte 2 (Invalid currency)

#### Packet examples

Response example for 2 notes store value 5 and 10  $\,$ 

Host transmit: **7F 80 01 41 85 83** 

Slave Reply: **7F 80 09 02 F4 01 00 00 E8 03 00 00 7D CF** 

Response given to command when BNV currency does not match stored note currency.

Host transmit: **7F 80 01 41 85 83**Slave Reply: **7F 80 02 F5 02 30 3E** 

Command	Code hex	Code decimal
Payout Note	0x42	66

Implemented on	Encryption Required
NV11	🖺 yes

The Note Float will payout the last note that was stored. This is the note that is in the highest position in the table returned by the Get Note Positions Command. If the payout is possible the Note Float will reply with generic response OK.

If the payout is not possible the reply will be generic response COMMAND CANNOT BE PROCESSED, followed by an error code shown in the table below.

Error	Code
not connected	1
empty	2
busy	3
disabled	4

#### Packet examples

Command acknowledged to payout first note in queue.

Host transmit: **7F 80 01 42 8F 83**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Stack Note	0x43	67

Implemented on	Encryption Required
NV11	<u></u> yes

The Note Float will stack the last note that was stored. This is the note that is in the highest position in the table returned by the Get Note Positions Command. If the stack operation is possible the Note Float will reply with generic response OK.

If the stack operation is not possible the reply will be generic response COMMAND CANNOT BE PROCESSED, followed by an error code shown in the table below.

Error	Code
not connected	1
empty	2
busy	3
disabled	4

## Packet examples

Command acknowledged to stack first note in queue.

Host transmit: **7F 80 01 43 8A 03**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Set Value Report Type	0x45	69

Implemented on	Encryption Required
NV11	🖺 yes

This will set the method of reporting values of notes. There are two options, by a four-byte value of the note or by the channel number of the value from the banknote validator. If the channel number is used then the actual value must be determined using the data from the Validator command Unit Data. The default operation is by 4-byte value. Send 0x00 to set Report by value, 0x01 to set Report By Channel.

If the setting is not possible the reply will be generic response COMMAND CANNOT BE PROCESSED, followed by an error code shown in the table below.

Error	Code
not connected	1
empty	2
busy	3
disabled	4

#### Packet examples

example to set report by value

Host transmit: **7F 80 02 45 00 36 9E**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Set Generator	0x4A	74

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Part of the eSSP encryption negotiation sequence.

Eight data bytes are sent. This is a 64 bit number representing the Generator and must be a prime number. The slave will reply with OK or PARAMETER\_OUT\_OF\_RANGE if the number is not prime.

Packet examples

In this example we are sending the prime number 982451653. This = 3A8F05C5 hex

Host transmit: 7F 80 09 4A C5 05 8F 3A 00 00 00 00 B2 73

Command	Code hex	Code decimal
Set Modulus	0x4B	75

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Part of the eSSP encryption negotiation sequence.

Eight data bytes are sent. This is a 64 bit number representing the Moduls and must be a prime number. The slave will reply with OK or PARAMETER\_OUT\_OF\_RANGE if the number is not prime.

Packet examples

In this example we are sending the prime number 1287821. This = 13A68D hex

Host transmit: 7F 80 09 4B 8D A6 13 00 00 00 00 6C F6

Command	Code hex	Code decimal
Request Key Exchange	0x4C	76

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

The eight data bytes are a 64 bit number representing the Host intermediate key. If the Generator and Modulus have been set the slave will calculate the reply with the generic response and eight data bytes representing the slave intermediate key. The host and slave will then calculate the key.

If Generator and Modulus are not set then the slave will reply FAIL.

#### Packet examples

An example of Host intermediate key of 7554354432121 = 6DEE29CC879 hex

Host transmit: 7F 80 09 4C 79 C8 9C E2 DE 06 00 00 9D 52

Command	Code hex	Code decimal
Coin Mech Options	0x5A	90

Implemented on	Encryption Required
SMART HOPPER, SMART SYSTEM	<b>△</b> yes

The host can set the following options for the Smart Hopper. These options do not persist in memory and after a reset they will go to their default values.

#### Bit function

0 Coin Mech error events 1 = ccTalk format, 0 = Coin mech jam and Coin return mech open only

1:7 Unused set to 0

If coin mech error events are set to ccTalk format, then event Coin Mech Error 0xB7 is given with 1 byte ccTalk

coin mech error reason directly from coin mech ccTalk event queue. Otherwise only error events Coin Mech

Jam 0xC4 and Coin Mech Return 0xC5 are given.

#### Packet examples

In this example we send register byte configured to return cctalk style events.

Host transmit: **7F 80 02 5A 01 30 DC**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Get Build Revision	0x4F	79

Implemented on	Encryption Required
NV11, NV200, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	optional

A command to return the build revision information of a device. The command returns 3 bytes of information representing the build of the product.

Byte 0 is the product type, next two bytes make up the revision number(0-65536). For NV200 and Nv9usb, the type byte is 0, for Note Float, byte is 3 and for SMART Payout the byte is 6.

#### Packet examples

This example is from an NV200 (issue 20) with payout attached (issue 21).

Host transmit: **7F 80 01 4F A2 03** 

Slave Reply: **7F 80 07 F0 00 14 00 06 15 00 0F 97** 

Command	Code hex	Code decimal
Enable Payout Device	0x5C	92

Implemented on	Encryption Required
NV11, SMART PAYOUT	<b>△</b> yes

A command to enable the attached payout device for storing/paying out notes. A successful enable will return OK, If there is a problem the reply will be generic response COMMAND\_CANNOT\_BE\_PROCESSED, followed by an error code.

For NV11 devices, this command uses an addition data byte, a bit register allows some options to be set.

bit	function	
0	GIVE_VALUE_ON_STORED. Set to 1 to enable the value of the note stored to be given with the Note Stored event	
1	NO_HOLD_NOTE_ON_PAYOUT. Set to 1 to enable the function of fully rejecting the dispensed banknote rather then holding it in the bezel.	
2:7	7 Unused- set to 0	

For SMART Payout devices with firmware greater or equal to 4.16, this command uses an addition data byte. A bit register allows some options to be set.

bit	function	
0	REQUIRE_FULL_STARTUP. If set to 1, the Smart Payout will return busy until it has fully completed the startup procedure	
1	OPTIMISE_FOR_PAYIN_SPEED. If set to 1 The Smart Payout will always move towards an empty slot when idle to try and ensure the shortest pay in speed possible.	
2:7	Unused- set to 0	

The device responds with COMMAND CANNOT BE PROCESSED and an error byte for failure to enable.

error	code
No device connected	1
Invalid currency detected	2
Busy	3
Empty only (Note float only)	4
Device error	5

Packet examples

Command	Code hex	Code decimal
Disable Payout Device	0x5B	91

Implemented on	Encryption Required
NV11, SMART PAYOUT	<u></u> yes

Description

All accepted notes will be routed to the stacker and payout commands will not be accepted.

# Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 5B DA 03**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Comms Pass Through	0x37	55

Implemented on	Encryption Required
SMART HOPPER, SMART SYSTEM	optional

The SMART Hopper includes two serial connections and this command enables the user to convert either of these into a USB to serial convertor so that the host can communicate directly with periferla connected to these ports.

This may be usful for updating or special configurations outside of the scope of the usual SMART Hopper to periferal protocols.

#### Command data format:

byte	function	size
0	UART select (0 - SSP Uart, 1 - cctalk UART)	1

Once this command is sent the device will respond with OK (0xF0) and from then all serial data via the USB will be routed to the periferal port directly.

To exit this mode, the host waits for at least 500ms since the last communication then sends byte array 0x55,0xAA,0x5A waits for 500ms and then sends the array again. The device will then reset and communications will restore to normal.

## Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 37 B2 02** Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Set Baud Rate	0x4D	77

Implemented on	Encryption Required
BV100, BV20, BV50, NV11, NV150, NV200, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, TEBS	optional

This command has two data bytes to allow communication speed to be set on a device.

byte	function	size
0	Required rate (0= 9600, 1=38400, 2= 115200)	1
1	Change persist (1=change will remain over reset, 0=rate sets to default after reset)	1

The device will respond with 0xF0 at the old baud rate before changing. Please allow a minimum of 100 millseconds before attempting to communicate at the new baud rate.

# Packet examples

In this example, we want to set the speed to 38400 bd with but to reset to default (9600) on reset.

Host transmit: **7F 80 03 4D 01 00 E4 27**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Ssp Set Encryption Key	0x60	96

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, FLATBED PRINTER, NV10USB, NV11, NV12, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	<u></u> yes

A command to allow the host to change the fixed part of the eSSP key. The eight data bytes are a 64 bit number representing the fixed part of the key. This command must be encrypted.

byte	function	size
0	new fixed key 64 bit, 8 byte	8

# Packet examples

Example to set new fixed key to 0x0123456701234567

Host transmit: **7F 80 09 60 67 45 23 01 67 45 23 01 BF 6F** 

Command	Code hex	Code decimal
Ssp Encryption Reset To Default	0x61	97

Implemented on	Encryption Required
BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS	optional

Resets the fixed encryption key to the device default. The device may have extra security requirements before it will accept this command (e.g. The Hopper must be empty) if these requirements are not met, the device will reply with Command Cannot be Processed. If successful, the device will reply OK, then reset. When it starts up the fixed key will be the default.

#### Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 61 46 03**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Get Real Time Clock Configuration	0x62	98

Implemented on	Encryption Required
COUPON PRINTER, FLATBED PRINTER, NV12, SMART SYSTEM, SMART TICKET	optional

Returns the configuration of the device Real Time Clock.

#### Response

The device responds with 1 data byte giving the configuration of the RTC. Data = 0, the RTC resets on power up and the date/time will need to be setup. Data = 1, the date/time is persistant after a power cycle.

#### Packet examples

In this example the device responds that the RTC does not hold it\'s settings after a power cycle.

Host transmit: **7F 80 01 62 4C 03**Slave Reply: **7F 80 02 F0 00 3F A0** 

Command	Code hex	Code decimal
Set Real Time Clock	0x64	100

Implemented on	Encryption Required
COUPON PRINTER, FLATBED PRINTER, NV12, SMART SYSTEM, SMART TICKET	optional

Send six bytes of parameter data to set the system time and date.

# Command data format:

byte	function	size
0	Generic OK	
1	Day of month (1-31)	1
2	Month of year (1-12)	
3	Year (0-99)	
4	Hour of day (0-23)	
5	Minute of hour (0-59)	
6	Second of minute (0-59)	1

Packet examples

Packet example for setting system time to 21st December 2012 10:22:30

Host transmit: 7F 80 07 64 15 0C 0C 0A 16 1E AF EC

Command	Code hex	Code decimal
Get Real Time Clock	0x63	99

Implemented on	Encryption Required
COUPON PRINTER, FLATBED PRINTER, NV12, SMART SYSTEM, SMART TICKET	optional

Gets the current system RTC date and time. Responds with 6 bytes of data.

# Response format:

byte	function	
0	Generic OK	
1	Day of month (1-31)	
2	Month of year (1-12)	
3	Year (0-99)	
4	Hour of day (0-23)	
5	Minute of hour (0-59)	
6	Second of minute (0-59)	1

Packet examples

In this example the system time is 21st December 2012 10:22:30

Host transmit: **7F 80 01 63 49 83** 

Slave Reply: **7F 80 07 F0 15 0C 0C 0A 16 1E EC F1** 

Command	Code hex	Code decimal
Set Cashbox Payout Limit	0x4E	78

Implemented on	Encryption Required
SMART HOPPER, SMART SYSTEM	<u></u> yes

Allow the host to specify a maximum level of coins, by denomination, to be left in the hopper.

During any payout operation, if there are coins in the hopper in excess of the set levels, when they are encountered on the conveyor belt they will be sent to the cashbox (beneath the hopper).

This means that over time (and multiple payout operations) any excess coins will be sent to the cashbox and the desired level will be achieved.

It effectively allows the hopper to do the 'floating' for the host machine i.e. it is an auto float mechanism.

NB: If a coin route is changed from cashbox to payout and then back to cashbox then the level for this coin will be reset to 0 (any of the coins will then be sent to cashbox).

# Command format.

byte	function	size
0	The number of individual requests	1
1	The level limit to set	
3	The denomination value	
7	7 The denomination country code (3 byte ASCII)	
	Repeat above block for each denomination required	

# Packet examples

Command	Code hex	Code decimal
Get Tebs Barcode	0x65	101

Implemented on	Encryption Required
TEBS	optional

This command is sent to the device to retrieve the barcode of the tamper evident cash bag. A successful request will return OK (0xF0), followed by the barcode data. If however there is a problem the device will reply COMMAND\_CANNOT\_BE\_PROCESSED (0xF5), followed by an error code.

#### Response format:

byte	function	size
0	Generic OK	1
1	Barcode data. Formated as plain number data. (see example)	variable

# Error code table:

Error	Code
No device connected	1
Unable to read barcode	2
Cashbox out of position	3
Cashbox removed	4
Cashbox unlocked	5
Currency mismatch	6

#### Packet examples

Example showing ok return with bar code number read 3421833

Host transmit: **7F 80 01 65 5D 83** 

Slave Reply: **7F 80 05 F0 03 2A 12 21 4B 30** 

Response showing cashbox out of position error

Host transmit: **7F 80 01 65 5D 83** Slave Reply: **7F 80 02 F5 03 35 BE** 

Command	Code hex	Code decimal
Enable Tito Events	0x72	114

Implemented on	Encryption Required
NV12, NV200	optional

When communicating with the NV200 attached to the printer, optional additional poll events may be enabled. These are enabled by sending an SSP packet with the command header 0x72 to the NV200. Polls will the respond with the same printing (0xA5) and printed (0xA6) poll responses as the printer.

#### Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 40 01 72 2F 8C**Slave Reply: **7F 40 01 F0 23 8F** 

Command	Code hex	Code decimal
Coin Stir	0x5D	93

Implemented on	Encryption Required
SMART SYSTEM	<b>△</b> yes

Description

Mixes the coins by performs a rotation of the Coin Hopper Motor for a specifed time.

Command has 1 parameter, a byte value (1-255) giving the time in seconds for which to stir the coins.

Packet examples

Stir the coins for 5 seconds

Host transmit: **7F 80 02 5D 05 28 CE**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Ticket Print	0×70	112

Implemented on	
COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET	

## Description

The *Ticket Print* command uses a system of sub commands to allow the host to send printer commands to the device.

See the sub command list for details.



Add Static Text (01 01 )

Encryption required
yes

### Description

Adds a fixed text item to a ticket template, or to the on the fly ticket buffer.

The font to use, position, rotation, and text to display are sent with this command. The template number to add this text to is also sent, and if 0 is used for this, the item is added to the on the fly buffer.

If added to a numbered template, the text item will be stored in the selected template file and printed every time that template is printed out.

If it is added to the on the fly buffer, it will be printed when the the print ticket command is called with a template parameter of 0. If a template file is printed, it will overwrite the contents of the on the fly buffer and the text will be lost.

The maximum number of charaters to print is 50. The text to display is sent with UTF-16 encoding.

The following table shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	01	1	Add Text Sub Command
3	٧	1	Template to Add to (0 for on the fly, 1-255 for other templates)
4	٧	1	Font index ID (0-255)
5	٧	1	Orientation (0-3, multiples of 90°)
6	٧	2	16-bit x Position of Text
8	٧	2	16-bit y Position of Text
10	V	٧	The Text to Display, Encoded as a UTF-16 String. Maximum of 50 Charaters (100 bytes)

Packet examples

Host transmit: 7F 40 14 70 01 03 01 01 00 14 00 19 00 48 00 65 00 6C 00 6C 00 6F 00 1D E2

Byte Value (hex) Size	3_2_2_649A Function

<< back to index

### Sub command of TICKET PRINT

Add Place Holder Text (01 02)

Encryption required

yes

## Description

Adds a place-holder for text to a template.

This text can then be filled in with the Set Placeholder Variable command, allowing for things such as dynamic counters on a ticket which changes every time while printing from the same template. The maximum number of characters to print is limited to 50 (100 bytes UTF-16).

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Set-up Sub Command
2	02	1	Add Placeholder Text Sub Command
3	V	1	Template to Add to (1-255 only, not allowed to add to on the fly)
4	V	1	Font Index (0-255)
5	V	1	Orientation, (0-3, multiples of 90°)
6	V	2	16-bit x Position of Text
8	V	2	16-bit y Position of Text
10	V	1	Maximum Character Count (max 50)
11	V	1	The Place Holder Reference ID (0-19)

# Packet examples

Example to add the text with max 10 characters to template 1 using font 2 with 0 x 90 degrees of rotation at position 20,25 using place holder index 1:

Host transmit: 7F 80 0C 70 01 02 01 02 00 14 00 19 00 0A 01 F7 9B

Protoco Value al Byte (hex)	GA138_2_2_649A Size	Function
--------------------------------	------------------------	----------

Add Static Barcode (01 03)

Encryption required
yes

## Description

Add a barcode to the ticket.

This is done in the same way as text, and the numbers are passed as UTF-16 characters rather than straight

values. The passed in height will be the height of the bar code on the ticket. The width represents the width of

a single thin bar in the bar code. The maximum number of characters to print is limited to 50 (100 bytes UTF-16).

The table below shows the command format:

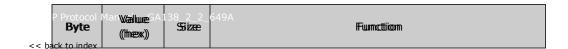
Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	03	1	Add Static Barcode Sub Command
3	V	1	Template to Add to (0 for on the fly, 1-255 for other templates)
4	V	1	Type of Barcode: 0x00 = Interleaved 2 of 5 (only currently supported format)
5	٧	1	Orientation (0-3, multiples of 90°)
6	٧	2	16-bit x Position of Barcode
8	٧	2	16-bit y Position of Barcode
10	٧	2	16-bit Width of Bars
12	V	2	16-bit Height of Barcode
14	V	v	The Barcode Number (UTF-16), Maximum of 50 Characters (100 bytes)

Packet examples

Host transmit: **7F 40 16 70 01 03 01 00 00 14 00 19 00 05 00 78 00 31 00 32 00 33 00 34 00 28 C1** 

Slave **7F 40 01 F0 23 8F** 

Reply:



Get Image Size (05 02 )

Encryption required
optional

## Description

Gets the area, in pixels, that an image will take up on a ticket. The width and height of the image are returned as 16-bit unsigned integers. The command assumes no rotation, and the image is to be rotated, the returned height should be used as width, and the width as height in any layout calculations.

The follwing table shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	02	1	Get Image Size Sub Command
3	٧	1	Image Index (0-255)

### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	2	16-Bit Width of Image (in Dots)
3	V	2	16-Bit Height of Image (in Dots)

Packet examples

Gets the size of image at index 5, and returns the size 30 x 40  $\,$ 

Host transmit: **7F 40 04 70 05 02 05 32 CD**Slave Reply: **7F 40 04 1E 00 28 00 79 E9** 



Get Barcode Size (05 03 )

Encryption required optional

## Description

Calculates and returns the width, in pixels, that a given barcode will take up on the ticket.

The width of the barcode is returned as a 16-bit unsigned integer. The height is not calculated or returned, as that is set directly by the command to add a barcode.

The follwing table shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	03	1	Get Barcode Size Sub Command
3	V	1	Type of Barcode: 0x00 = Interleaved 2 of 5 (only currently supported format)
4	٧	1	The Width of an individual bar
5	V	v	The Barcode Number (UTF-16), Maximum of 50 Characters (100 bytes)

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	2	16-Bit Width of Resulting Barcode

Packet examples

Host transmit: **7F 40 0D 70 05 03 00 04 31 00 32 00 33 00 34 00 57 65** 

Slave Reply: **7F 40 03 F0 00 01 C6 0A** 

Get Ticket Resolution (05 04)

Encryption required	
optional	

## Description

Gets the height and width that the ticket image printed can be, in pixels, for use in setting the coordinates of printed elements. Responds with a 16-bit width and 16-bit height.

The following table shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	04	1	Get Ticket Resolution Command

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	2	16-Bit Width (x) of Ticket
2	v	2	16-Bit Height (y) of Ticket

Packet examples

Gets the 16-bit x and y resolution of the ticket at 1096x520

Host transmit: **7F 40 03 70 05 04 DB 9E** 

Slave Reply: **7F 40 05 F0 48 04 08 02 C7 3E** 

Get Font Information (05 05 )

Encryption required	
optional	

## Description

Gets information about a font. Returns the 16-bit max character width, 16-bit max character height, 16-bit font size, 1 byte bold, 1 byte itallic and variable length font name string.

The following table shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	05	1	Get Font Info Sub Command
3	V	1	The Font Index ID (0-255)

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	٧	2	16-Bit Maximum Character Width in Pixels (in Dots)
4	٧	2	16-Bit Maximum Character Height in Pixels (in Dots)
6	٧	2	16-Bit Font Size
8	٧	1	Bold Flag
9	٧	1	Italic Flag
10	٧	٧	ASCII Windows Font Filename

## Packet examples

Gets the font information for font 2. Returns info on a font with size 10, maximum character width 15, maximum character height 28, and filename consola

Host transmit: **7F 40 04 70 05 05 02 20 DF** 

Slave Reply: 7F 40 10 F0 0F 00 1C 00 0A 00 00 63 6F 6E 73 6F 6C 61 78 71

Get Qr Code Dimensions (05 0C)

Encryption required optional

## Description

Find the height and width in dots of a QR code.

The get QR code dimensions command can be used to find the height and width in dots of a QR code with a particular set of data (the height and width will always be the same as the QR Code is square.) This can be multiplied by the dot size you intend to use to find out how much room the QR code will take up on the ticket.

The follwing table shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Setup Sub Command
2	0C	1	Get QR Code Dimensions Sub Command
3	٧	1	The Length of the ASCII Data to be Used (1-120)

### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	2	16-Bit Width and Height of QR Code (in Dots)

Packet examples

Get the size of a QR code with data of length 21, returning a size of 25

Host transmit: **7F 80 04 70 05 12 15 9E AD** 

Slave Reply: **7F 80 02 F0 19 6A 20** 

P Protocol Byte	(hex)	138_2_2 Size	649A Function
<< back to index			

Print Ticket (02)

Encryption required

yes

## Description

Prints a ticket from a template or on the fly data.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	02	1	Print Ticket Sub Command
2	V	1	Template to Print (1-255) or 0 for On-the-fly Buffer

Packet examples

Tell the device to print template 7

Host transmit: **7F 40 03 70 02 07 D2 0C** 

P Protocol	Ma <b>Val</b> ue G/	138_2_2	649A Function
Byte	(hex)	Size	
<< hack to index			

Print Blank Ticket (03 )

Encryption required

yes

## Description

Causes a blank (no print) ticket to be dispensed.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	03	1	Print Blank Ticket Sub Command

Packet examples

Host transmit: **7F 40 02 70 03 1E 20**Slave Reply: **7F 40 01 F0 23 8F** 



Get Text Size (05 01)

Encryption required optional

## Description

Finds the amount of space a text string will take up on the ticket. Returns the width and height of the text as 16-bit unsigned integers. Assumes no rotation.

The table below shows the command format:

Byte	Value (hex)	Size	Function	
0	70	1	Print Command	
1	05	1	Get Info Sub Command	
2	01	1	Get Text Size Sub Command	
3	٧	1	Font index ID (0-255)	
4	V	>	The UFT-16 text sring array that will be used (Max 50 characters (100 bytes))	

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	٧	2	16-Bit Width of Text (in Dots)
3	v	2	16-Bit Height of Text (in Dots)

Packet examples

Gets the size of the text \"WIN\" using font 2, and reports back a width of 45, and height of 28  $\,$ 

Host transmit: 7F 40 0A 70 05 01 02 57 00 49 00 4E 00 02 4A

Slave Reply: **7F 40 05 F0 2D 00 1C 00 8A 02** 

P Protocol MValue	GA138_	2_2_649A
Byte (hex)	Size	Function
< < halck to index		

Set Qr Placeholder (01 0B)

Encryption required
yes

## Description

Load the designated QR placeholder with the supplied ASCII data.

QR placeholder values are set with a different command to standard ones, as the data for QR codes is in ASCII format and not UTF-16. There are three QR placeholder buffers available.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	0B	1	Set QR Code Placeholder Sub Command
3	V	1	Placeholder Index to Use (0-2)
4	V	v	The ASCII Data to Place in the Placeholder

### Packet examples

Set QR placeholder index 0 to "test"

Host transmit: 7F 80 08 70 01 0B 00 74 65 73 74 85 43

ascii: . . p . . . t e s t . C

Add Qr Code (01 09 )

Encryption required

yes

## Description

Adds a QR code image to the ticket.

The size (height and width, which are always the same as each other) of the dots is sent in the command. Unlike other ticket data, the info within the QR code is send as ASCII text, as oppose to UTF-16. The maximum number of ASCII characters the QR code can store is 120.

The table below shows the command format:

Byte	Value (hex)	Size	Function	
0	70	1	Print Command	
1	01	1	Setup Sub Command	
2	09	1	Add Static QR Code Sub Command	
3	V	1	Template to Add to (0 for on the fly, 1-255 for other templates)	
4	V	1	Dot Size (>=1)	
5	v	1	Orientation (0-3, multiples of 90°)	
6	V	2	16-bit x Position of QR Code	
8	V	2	16-bit y Position of QR Code	
10	v	٧	ASCII Data (1-120 characters)	

### Packet examples

Add a QR code to template 2, with a dot size of 4, no rotation, at coordinates 50, 50 with the data "hello"

Host transmit: **7F 80 0F 70 01 09 02 04 00 32 00 32 00 68 65 6C 6C 6F 57 2F** ascii: . . p . . . . . **2** . **2** . **h e I I o W** /

Add Qr Placeholder (01 0A )

Encryption required

yes

## Description

Adds a placeholder QR code to the ticket.

Placeholder QR codes do not use the same placeholder buffers as other placeholder items, and their placeholders are set with a different command (detailed separately.) The maximum data size for the QR code is sent with the command.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	0A	1	Add Placeholder QR Code Sub Command
3	v	1	Template to Add to (1-255 only, not allowed to add to on the fly)
4	v	1	Dot Size (>=1)
5	v	1	Orientation (0-3, multiples of 90°)
6	v	2	16-bit x Position of QR Code
8	v	2	16-bit y Position of QR Code
10	v	1	Maximum Data Length (1-120 characters)
11	v	1	Placeholder to Use (0-2)

## Packet examples

 $\mbox{Add QR Placeholder: to template 2, dot size 4, no rotation, at location 320,116, with max data size of 120, using placeholder 0 \\$ 

Host transmit: **7F 80 0C 70 01 0A 02 04 00 40 01 74 00 78 00 D0 59** 

Clear On The Fly Buffer (01 07 )

Encryption required
yes

## Description

Clears all stored information in the on the fly ticket buffer. Send this command before sending a new set of on the fly information.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	07	1	Clear On-the-fly Buffer Sub Command

## Packet examples

Host transmit: **7F 40 03 70 01 07 D2 06** 

Set Placeholder (01 08 )

Encryption required

yes

## Description

Sets the value of a place holder variable at a given index for the next print. The maximum number of characters to print is limited to 50 (100 bytes UTF-16).

The table below shows the command format:

Byte	Value (hex)	Size	Function	
0	70	1	Print Command	
1	01	1	Setup Sub Command	
2	08	1	Set Placeholder Variable Sub Command	
3	V	1	Placeholder Index (0-19)	
4	V	v	The Text to Display, Encoded as a UTF-16 String. Maximum of 50 characters (100 bytes)	

## Packet examples

Sets the placeholder string 6 to contain "\$3.00"

Host transmit: 7F 40 0E 70 01 08 06 24 00 33 00 2E 00 30 00 30 00 58 03

Clear Template (01 06 )

Encryption required
yes

## Description

Clears all stored information for a given template.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	06	1	Clear Template Sub Command
3	V	1	Template to Clear (1-255)

Packet examples

Clears template 13

Host transmit: **7F 40 04 70 01 06 0D 51 55** 

Protoco Value al Byte (hex)	GA138_2_2_649A Size	Function
--------------------------------	------------------------	----------

Add Placeholder Barcode (01 04 )

Encryption required
yes

## Description

Adds a place holder barcode to allow dynamic updating of ticket codes. The maximum number of characters to print is limited to 50 (100 bytes UTF-16).

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	01	1	Setup Sub Command
2	04	1	Add Placeholder Barcode Sub Command
3	V	1	Template to Add to (0 for on the fly, 1-255 for other templates)
4	V	1	Type of Barcode: 0x00 = Interleaved 2 of 5 (only currently supported format)
5	٧	1	Orientation (0-3, multiples of 90°)
6	٧	2	16-bit x Position of Barcode
8	٧	2	16-bit y Position of Barcode
10	V	2	16-bit Width of Bars
12	V	2	16-bit Height of Barcode
14	V	1	Maximum Chracter Count (Max 50 characters, 100 bytes)
15	V	1	The Place Holder Reference ID (0-19)

Packet examples

Adds a placeholder to template 9, at position 60, 60, with a bar width of 4, a height of 100, a maximum of 20 characters, using palceholder 3

Host transmit: 7F 40 10 70 01 04 09 00 00 3C 00 3C 00 04 00 64 00 20 03 48 7E

Add Image (01 05)

Encryption required

yes

## Description

Allows the host to specify the resource index and placement variables of the image to add to the ticket or template.

The table below shows the command format:

Byte	Value (hex)	Size	Function	
0	70	1	Print Command	
1	01	1	Setup Sub Command	
2	05	1	Add Image Sub Command	
3	٧	1	Template to Add to (0 for on the fly, 1-255 for other templates)	
4	٧	1	Image Index ID (0-255)	
5	٧	1	Orientation (0-3, multiples of 90°)	
6	V	2	16-bit x Position of Text	
8	V	2	16-bit y Position of Text	

Packet examples

Adds image 18 to template 7, at position 300, 50, with 90 degrees rotation

Host transmit: 7F 40 0A 70 01 05 07 12 01 2C 01 00 32 5C EE



Get Ticket Size (05 06 )

Encryption required optional

## Description

Gets the size of the ticket in mm that the printer is set to use. Returns 16-bit length and 16-bit height.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	06	1	Get Ticket Size Sub Command

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	2	16-Bit Width of Text (mm)
3	٧	2	16-Bit Height of Text (mm)

Packet examples

Gets the ticket size in mm of 155 x 65mm

Host transmit: **7F 40 03 70 05 06 D4 1E** 

Slave Reply: **7F 40 05 F0 9B 00 41 00 B9 F4** 

Get Free Storage (05 07 )

Encryption required	
optional	

## Description

Gets the amount of free storage, in KB, on either the printer internal memory, or an inserted sd card as 32 bit little endian number.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	07	1	Get Free Storage Sub Command
3	٧	1	Location of Memory to Check (0 for Internal Memory, 1 for SD Card.)

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	v	4	32-Bit Amount of Free Space in KB

Packet examples

Get the free storage on the internal flash, returning 1964 KB

Host transmit: **7F 40 04 70 05 07 00 2C D3**Slave Reply: **7F 40 05 F0 AC 07 00 00 DA 5E** 

Protocol Byte (hex)	138_2_2 Size	649A Function
---------------------	-----------------	---------------

Check For Template (05 08)

Encryption required optional

## Description

Check if a template with a given index exists on the device.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	08	1	Check for Template Sub Command
3	V	1	Template Index to Check

### Response

If a template with the requested index exists, the command will return a generic SSP OK (0xF0) and will return a Parameter Out of Range (0xF4) if it does not.

Packet examples

Checks to see if template 18 exists

Host transmit: **7F 40 05 70 70 05 08 12 B9 62** 

Get Present Templates (05 09 )

Encryption required

optional

## Description

Returns a 32 byte bit-mask to indicate which template indexes are present on the device (index 0 to 255). The bytes are sent little endian, with bit 0 of the array representing index 0.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	09	1	Get Present Templates Sub Command

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	32	256-Bit Resource Mask

Packet examples

Returns a bit mask where templates 1 to 11 are present

Host transmit: **7F 40 03 70 05 09 F6 1E** 

Get Present Fonts (05 0A)

Encryption required	
optional	

## Description

Returns a 32 byte bit-mask to indicate which font indexes (0-255) are present on the device. The bytes are sent little endian, with bit 0 of the array representing index 0.

Index 0 cant be overwritten and a font will always be present.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	08	1	Get Info Sub Command
2	0A	1	Get Present Fonts Sub Command

### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	32	256-Bit Resource Mask

Packet examples

Returns a bit mask where templates 1 to 5 are present

Host transmit: **7F 40 03 70 08 0A FF B0** 

Get Present Images (05 0B)

Encryption required optional

## Description

Returns a 32 byte bit-mask to indicate which image indexes (0-255) are present on the device. The bytes are sent little endian, with bit 0 of the array representing index 0.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	08	1	Get Info Sub Command
2	0B	1	Get Present Images Sub Command

### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	v	32	256-Bit Resource Mask

Packet examples

Returns a bit mask where templates 1 to 11 are present

Host transmit: **7F 40 03 70 05 0B F9 9E** 

Get Template Info (05 0D )

Encryption required optional

## Description

Returns the information about the make-up of a particular stored template index.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	0D	1	Get Template Info Sub Command
3	V	1	Template Index

## Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	٧	1	Total Number of Items in Template
2	٧	1	Number of Static Text Items
3	٧	1	Number of Placeholder Text Items
4	٧	1	Number of Static Barcode Items
5	٧	1	Number of Placeholder Barcode Items
6	٧	1	Number of Image Items
7	٧	1	Number of Static QR Code Items
8	V	1	Number of Placeholder QR Code Items

## Packet examples

Gets information about template 2, which has a total of 8 items: 4 static texts, 1 placeholder barcode, and 3 images

Host transmit: **7F 40 04 70 05 0D 02 23 6F** 

Slave Reply: 7F 40 09 F0 08 04 00 00 01 03 00 00 C7 C2

Get Template Item Info (05 0E)

Encryption required optional

## Description

Returns the information about the make-up of a particular stored template index.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	0E	1	Get Template Item Info Sub Command
3	٧	1	Template Index
4	٧	1	Item index *

<sup>\*</sup> This index is obtained using the Get Template Info command. If this returns 7 items on a template then the indexes of the items will be (0-6).

### Response

The returned data varies based on the item type. The start of the data is generic and is formatted as follows:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	1	Type of Item (1 = Static Text, 2 = Placeholder Text, 3 = Static Barcode, 4 = Placeholder Barcode, 5 = Image, 8 = Static QR Code, 9 = Placeholder QR Code)
2	٧	2	16-Bit x Position of Item
4	V	2	16-Bit y Position of Item
6	٧	1	Orientation (0-3, multiples of 90°)

<b>Bythy</b> te	V <b>alake</b> lue ( <b>((bæ)/)</b> ex) Value		
Byte	(hex)	Size	Function
7	V	1	Text Font ID
8	V	٧	UTF-16 Item Text

### **Placeholder Text Item Information**

Byte	Value (hex)	Size	Function
7	V	1	Text Font ID
8	V	1	Placeholder Index
9	V	1	Maximum Length

### Static Barcode Item Information

Byte	Value (hex)	Size	Function
7	V	1	Barcode Type
8	V	2	Individual Bar Width
10	V	2	Barcode Height
12	V	٧	UTF-16 Item Code

## Placeholder Barcode Item Information

Byte	Value (hex)	Size	Function
7	٧	1	Barcode Type
8	٧	2	Individual Bar Width
10	V	2	Barcode Height
12	٧	1	Placeholder Index
13	V	1	Maximum Length

## Image Item Information

Byte	Value (hex)	Size	Function
7	V	1	Text Font ID
8	V	V	Image Index

Static QR Code Bytte	e Item Inform Waaluee ((hess))	nation Sæi∉e	Flunction
Byte	Value (hex)	Size	Function
7	V	1	Dot Size
8	V	V	ASCII QR Code Data

#### Placeholder QR Code Item Information

Byte	Value (hex)	Size	Function
7	٧	1	Dot Size
8	V	1	Placeholder Index
9	V	1	Maximum Data Length

Packet examples

Gets information about template item 6 in template 2, which is a static text item at position 534, 406, with no rotation, using font 1, with the text "SMART Ticket""

Host transmit: **7F 40 05 70 05 0E 02 06 49 DA** 

Slave Reply: 7F 40 20 F0 01 16 02 96 01 00 01 53 00 4D 00 41 00 52 00 54 00 20 00 54 00 69 00 63 00 6B 00 6



#### Sub command of TICKET PRINT

Get Image File Checksum (05 0F)

Encryption required optional

#### Description

Returns the CRC check sum for an image stored on the SMART Ticket file system. This may be useful for checking which images are present on a system. (Seed = 0xFFFF, same function as the packet check sum for SSP).

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	0F	1	Get Image Checksum Sub Command
3	V	1	Image Index (0-255)

#### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	٧	2	16-Bit CRC Checksum of the File on the Printer

Packet examples

Gets a checksum of image 3, which has a checksum of E5AA (hex)

Host transmit: **7F 40 04 70 05 0F 03 25 63**Slave Reply: **7F 40 03 F0 AA E5 94 F4** 



#### Sub command of TICKET PRINT

Get Ticket Bounds (05 10 )

Encryption required optional

#### Description

A ticket printer command to get information about the printable area of the ticket (pixel offsets).

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	10	1	Get Pixel Bounds Sub Command

#### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	٧	2	16-Bit Top-most Printable Pixel Coordinate
3	٧	2	16-Bit Bottom-most Printable Pixel Coordinate
5	٧	2	16-Bit Left-most Printable Pixel Coordinate
7	٧	2	16-Bit Right-most Printable Pixel Coordinate

Packet examples

Returns the ticket bounds of 28, 224, 80, 1176

Host transmit: **7F 40 05 70 70 05 10 03 DF 32** 

Slave Reply: 7F 40 09 F0 1C 00 24 02 50 00 98 04 1B 62

#### Sub command of TICKET PRINT

Get Pixel Density (05 11 )

Encryption required optional

#### Description

Returns the DPI or DPmm of the device printer.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	70	1	Print Command
1	05	1	Get Info Sub Command
2	11	1	Get Pixel Density Sub Command
3	V	1	Return Type. 0 = Dots Per mm, 1 = Dots Per Inch

#### Response

The following table shows the structure of the response data:

Byte	Value (hex)	Size	Function
0	F0	1	Generic OK
1	V	1	Pixel Density in the Selected Unit

Packet examples

Gets back the pixels per mm of 8

Host transmit: **7F 40 04 70 05 11 00 2F 27** 

Slave Reply: **7F 40 02 F0 08 2E 20** 

Command	Code hex	Code decimal
Printer Configuration	0x71	113

Implemented on	
COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET	

The **Printer Configuration** command uses a system of sub commands to allow the host to send printer configuration commands to the device.

See the sub command list for details.



Set Ticket Width (02)

Encryption required optional

#### Description

Sets the width (size in the direction of print) of the ticket (x direction, and direction of ticket travel) in mm using a 16-bit integer.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	71	1	Printer Config Command
1	02	1	Set Ticket Width Sub Command
2	٧	2	Ticket Width (mm)

Packet examples

Sets the ticket width to 130mm

Host transmit: **7F 40 03 71 02 82 D8 0F** 



Set Ticket Height (03)

Encryption required optional

#### Description

Sets the height (size perpendicular to the direction of print) of the ticket (y direction) in mm using a 16-bit integer.

The table below shows the command format:

Byte	Value (hex)	Size	Function
0	71	1	Printer Config Command
1	03	1	Set Ticket Height Sub Command
2	٧	2	Ticket Height (mm)

Packet examples

Sets the ticket height to 50mm

Host transmit: **7F 40 03 71 03 32 78 0A** 



Enable Reverse Validation (07)

Encryption required optional

Description

Enables reverse validation on printers which attach to a validator.

The table below shows the command format:

Byte	Value (hex)	Size	Function Printer Config Command	
0	71	1		
1	07	1	Enable Reverse Validator Sub Command	

Packet examples

Host transmit: **7F 40 02 71 07 06 26**Slave Reply: **7F 40 01 00 03 8D** 

Disable Reverse Validation (08)

Encryption required optional

#### Description

Disables reverse validation on printers which attach to a validator.

The table below shows the command format:

Byte	Value (hex)	Size	Function Printer Config Command	
0	71	1		
1	08	1	Disable Reverse Validator Sub Command	

Packet examples

Host transmit: **7F 40 02 71 08 24 26** 

Slave Reply: **7F 40 01 00 03 8D** 

Delete File (0A )

Encryption required optional

#### Description

Deletes a selected resource file of the selected type, on the selected drive.

The table below shows the command format:

Byte	Value (hex)	Size	Function	
0	71	1	Printer Config Command	
1	0A	1	Delete File Sub Command	
2	V	1	The type of file to be deleted. 0x01 for templates, 0x02 for fonts, 0x03 for images.  The location to delete the file from. 0x01 for internal flash, 0x02 for SD card, 0x03 for both.	
3	V	1		
4	V	1	The index of the file to delete (0-255)	

Packet examples

Delete font 6 from the internal flash

Host transmit: 7F 40 05 71 0A 02 01 06 C2 9C

Delete File Group (0B)

Encryption required optional

#### Description

Removes all instances of a selected type of resource from a selected drive.

The table below shows the command format:

Byte	Value (hex)	Size	Function  Printer Config Command	
0	71	1		
1	0B	1	Delete File Group Sub Command	
2	V	1	The type of file to be deleted. $0x01$ for templates, $0x02$ for fonts, $0x03$ for images.	
3	V	1	The location to delete the file from. 0x01 for internal flash, 0x02 SD card, 0x03 for both.	

Packet examples

Delete all templates from the SD card

Host transmit: 7F 40 04 71 0B 01 02 FB 53

Set Paper Saving Mode (0D)

Encryption required
optional

#### Description

Sets the paper saving mode on printers which support variable length tickets. With paper saving mode enabled, if the contents of the ticket doens't take up the entire ticket length, a shorter ticket will be printed.

Byte	Value (hex)	Size	Function	
0	71	1	Printer Config Command	
1	0D	1	Set Paper Saving Mode Sub Command	
2	V	1	The Paper Saving Setting. 0x00 for Disabled, 0x01 for Enabled	

#### Packet examples

Turn on paper saving mode

Host transmit: **7F 40 03 71 0D 01 D1 AE** 

Set Bezel Type (0E)

Encryption required

optional

#### Description

Sets the bezel type on printers which support bezels of different length. This will effect the minimum ticket length, as the length of the ticket must be able to exit the bezel.

Byte	Value (hex)	Size	Function
0	71	1	Printer Config Command
1	0E	1	Set Bezel Type Sub Command
2	V	1	The Bezel Type Setting

#### Packet examples

Host transmit: **7F 40 02 71 E001 12 26** 

Set Printing Quality (06)

Encryption required

optional

#### Description

Sets the quality setting for printed tickets. Higher values will produce a better quality print, but print times will be increased.

Byte	Value (hex)	Size	Function	
0	71	1	Printer Config Command	
1	06	1	Set Print Quality Sub Command	
2	V	1	The Quality Setting. $0x00 = High Speed$ , $0x01 = Standard$ , $0x02 = High Quality$	

#### Packet examples

Set the print quality setting to high quality

Host transmit: **7F 40 03 71 06 02 D8 14** 

Command	Code hex	Code decimal	
Get Tebs Log	0x66	102	

Implemented on	Encryption Required	
TEBS	optional	

This command is sent to the device to retrieve historical transaction data for the last **FIVE** cash bags. The index of the log is sent as a parameter with the command where 0 is the most recent log and 4 is the oldest. You should expect a response COMMAND\_CANNOT\_BE\_PROCESSED (0xF5) with Data byte 5 (GETTING DATA) while the data is gathered from the TEBS unit at first. This will change to OK (0xF0) followed by the data when on successful completion of the request

#### Command format:

byte	function	size
0	data history index (0- 4)	1

OK Response format:

Pre 4.28/5.28 log format

byte	function	size
0	Generic OK	1
1	Data checksum (XOR of byte 2 to end, 0 seed)	1
2	Log index	1
3	Barcode data (eg 01,23,45,67 for 1234567)	4
7	Country code (ASCII)	3
10	Max Bill Capacity	2
12	Spare	3
15	Other count (2 byte integer)	2
17	Channel counts (8 x 2 byte integers)	16

byte	function	Size
0	Generic OK	1
1	Log Index (0-4)	1
2	CRC - 16 using polynomial   X16 + X15 + X2+1 initialised with 0xFFFF and calculated on bytes 4 - 65	2
4	Spare	4
8	Country code (ASCII eg [E],[U],[R])	3
11	Maximum bill capacity	2
13	Barcode length	1
14	Number of channels	1
15	Spare	1
16	Other count	2
18	Channel counts	16(byte[14]) * 2
50	Barcode Data (eg 01,23,45,67 for 1234567)	16

#### Packet examples

Response for history index 0, bar code 276745 with GBP 120 notes channel1, 180, channel2 and 56 in channel3

Host transmit: **7F 80 02 66 00 35 D4** ascii: **. . f . 5 .** 

Slave Reply: 7F 80 21 F0 16 00 00 1B 43 2D 47 42 50 00 00 00 00 00 00 78 00 B4 00

38 00 00 00 00 00 00 00 00 00 00 79 A2

ascii: . . . . . C - G B P . . . . . . . . . . . . . . . .

8 . . . . . . . . . . . .

Command	Code hex	Code decimal
Cashbox Unlock Enable	0x67	103

Implemented on	Encryption Required
TEBS	🖺 yes

This command allows the TEBS device to be unlocked using the physical key. Please note that the NV200 will still not allow the cashbox to be opened when a note is in transit or during setup.

If the host does not want to control this functionality on the fly then sending this command as part of the start-up routine is recommended.

#### Packet examples

Command format (no parameters) for acknowledged request.

Host transmit: **7F 80 01 67 52 03**Slave Reply: **7F 80 01 F0 23 80** 

Failed command gives COMMAND\_CANNOT\_BE\_PROCESSED response:

Host transmit: **7F 80 01 67 52 03**Slave Reply: **7F 80 01 F5 3D 80** 

Command	Code hex	Code decimal
Cashbox Lock Enable	0x68	104

Implemented on	Encryption Required
TEBS	<b>△</b> yes

This command stops the TEBS device from being unlocked using the physical key.

Packet examples

Fail generates COMMAND-CANNOT\_BE\_PROCESSED response:

Host transmit: **7F 80 01 68 70 03**Slave Reply: **7F 80 01 F5 3D 80** 

**Example of Accepted Command** 

Host transmit: **7F 80 01 68 70 03**Slave Reply: **7F 80 01 F0 23 80** 

Command	Code hex	Code decimal
Reset Tebs Logs	0x69	105

Implemented on	Encryption Required
TEBS	optional

Reset the TEBS logs as described in the  $\underline{\text{Get TEBS Log}}$  command . All fields are reset to zero values.

Packet examples

Failed command gives COMMAND\_CANNOT\_BE\_PROCESSED response:

Host transmit: **7F 80 01 69 75 83**Slave Reply: **7F 80 01 F5 3D 80** 

Command	Code hex	Code decimal
Cancel Escrow Transaction	0x76	118

Implemented on	Encryption Required
SMART PAYOUT	optional

#### Description:

All notes In Escrow are Returned. As each note is Returned its value is deducted from the Escrow Value.

 $<sup>\</sup>ensuremath{^{**}}$  Please Note: A special version of firmware is required to enable Escrow on the Smart Payout  $\ensuremath{^{**}}$ 

Command	Code hex	Code decimal
Commit Escrow Transaction	0x77	119

Implemented on	Encryption Required
SMART PAYOUT	optional

#### Description:

Notes in Escrow are Committed. If the free spaces in the escrow are less than Escrow Spaces, notes are Stacked to increase the free slots until they equal Escrow Spaces

 $<sup>\</sup>ensuremath{^{**}}$  Please Note: A special version of firmware is required to enable Escrow on the Smart Payout  $\ensuremath{^{**}}$ 

Command	Code hex	Code decimal
Read Escrow Value	0x78	120

Implemented on	Encryption Required
SMART PAYOUT	optional

Description:

Returns current Escrow Value.

 $<sup>\</sup>ensuremath{^{**}}$  Please Note: A special version of firmware is required to enable Escrow on the Smart Payout  $\ensuremath{^{**}}$ 

Command	Code hex	Code decimal
Get Escrow Size	0x79	121

Implemented on	Encryption Required
SMART PAYOUT	optional

Description:

Returns Escrow Spaces (from the Set Escrow Size command).

 $<sup>\</sup>ensuremath{^{**}}$  Please Note: A special version of firmware is required to enable Escrow on the Smart Payout  $\ensuremath{^{**}}$ 

Command	Code hex	Code decimal
Set Escrow Size	0×7A	122

Implemented on	Encryption Required
SMART PAYOUT	optional

#### Description:

Sets Escrow Spaces. Value is sent as a 1 byte parameter. This command can only be used when the Validator is disabled and the escrow is empty.

 $<sup>\</sup>ensuremath{^{**}}$  Please Note: A special version of firmware is required to enable Escrow on the Smart Payout  $\ensuremath{^{**}}$ 

Command	Code hex	Code decimal
Payout Amount By Denomination	0x39	57

Implemented on	Encryption Required
SMART SYSTEM	<b>△</b> yes

This command is similar to 'Payout Amount' but has two values in the payout which you can select the denominations for each.

Command	Code hex	Code decimal
Coin Escrow	0x3A	58

Implemented on
SMART SYSTEM

Description
-------------

Command to hold coins in the feeder without accepting into hopper.

#### Open Count ()

Encryption required optional

#### Description

Open feeder escrow in count mode (all coins give value of 1 regardless of denomination giving a coin count).

Starts 1 minute timeout after which all coins will be rejected.

#### Open Value ()

Encryption required optional

#### Description

Open feeder escrow in value mode (all coins are added to give their actual value).

Starts 1 minute timeout after which all coins will be rejected.

#### Start ()

Encryption required optional

Description

Start accepting coins in escrow into the hopper.

#### Reject ()

Encryption required
optional

Description

Reject all coins in escrow when start command received.

#### Close ()

Encryption required optional

Description

Close feeder escrow entry.

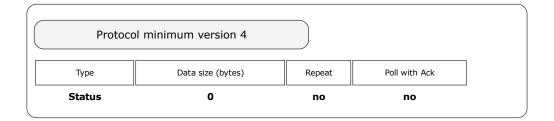
Event	Code hex	Code decimal
Slave Reset	0xF1	241

#### Implemented on

BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS

#### Description

An event gven when the device has been powered up or power cycled and has run through its reset process.



#### Packet examples

Poll returns slave reset event

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 01 F1 26 00** 

Event	Code hex	Code decimal
Read	0xEF	239

#### Implemented on

BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

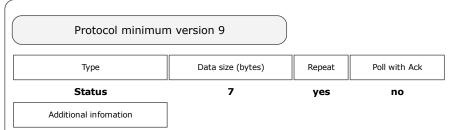
#### Description

An event given when the BNV is reading a banknote.

# Protocol minimum version 4 Type Data size (bytes) Repeat Poll with Ack Status 1 yes no Additional infomation

If the event data byte is zero, then the note is in the process of being scanned and validated.

If the data byte value changes from zero to a vaule greater then zero, this indicates a valid banknote is now held in the escrow position. The byte value shows the channel of the banknote that has been validated. A poll command after this value has been given will cause the banknote to be accepted from the escrow position. The host can also issue a reject command at this point to reject the banknote back to the user.



**For the SMART Currency device only** - 7 data bytes are given. If all bytes are zero then a banknote is in the process of being scanned and validated. Non zero show the country code and value of a validated banknote held in escrow.

data byte	function	size
0	3 byte ASCII code for country validated	3
3	4 byte code for banknote value	4

#### Packet examples

Poll response showing a biil being read but not yet validated.

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 EF 00 CF CA** 

Poll response showing channel 3 bill held in escrow

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 EF 03 C5 CA** 

Event	Code hex	Code decimal
Note Credit	0xEE	238

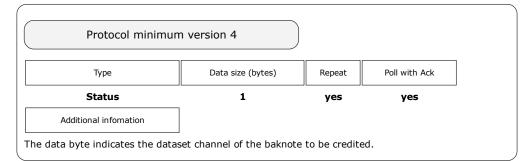
#### Implemented on

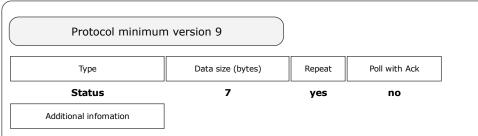
BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

This event is generated when the banknote has been moved from the escrow position to a safe postion within the validator system where the baknote cannot be retreived by the user.

At this point, it is safe for the host to use this event as it's 'Credt' point.





For the SMART Currency device only - 7 data bytes are given showing the country code and value of a Credited banknote.

data byte	function	size
0	3 byte ASCII code for country validated	3
3	4 byte code for banknote value	4

#### Packet examples

Poll response showing bill credit channel 4

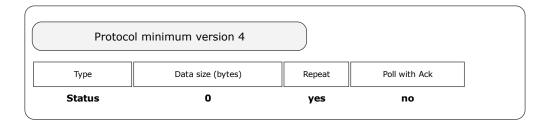
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 EE 04 D7 CC** 

Event	Code hex	Code decimal
Rejecting	0xED	237

## Implemented on BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

A bill is in the process of being rejected back to the user by the Banknte Validator.



#### Packet examples

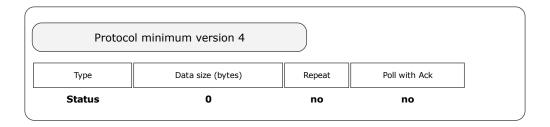
Poll response showing bill rejecting

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 ED 51 A2** 

Event	Code hex	Code decimal
Rejected	0xEC	236

Implemented on	
BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS	

A bill has been rejected back to the user by the Banknote Validator.



### Packet examples

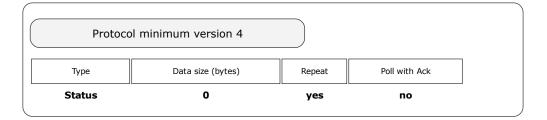
Poll response showing bill rejected by the validator.

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 EC 54 22** 

Event	Code hex	Code decimal
Stacking	0xCC	204

Implemented on
BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, TEBS

The bill is currently being transported to and through the device stacker.



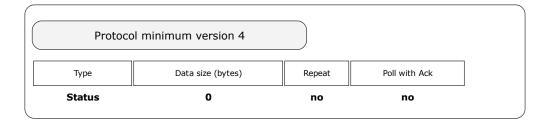
### Packet examples

Event	Code hex	Code decimal
Stacked	0xEB	235

# Implemented on BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

A bill has been transported trough the banknote validator and is in it's stacked position.



### Packet examples

Poll response showing stacked bill seen

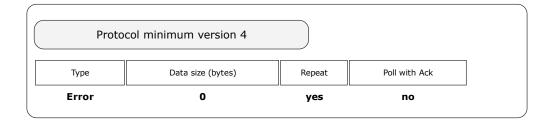
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 EB 45 A2** 

Event	Code hex	Code decimal
Unsafe Jam	0xE9	233

BV100, BV20, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

A bill has been detected as jammed during it's transport through the validator. An unsafe jam indicates that this bill may be in a position when the user could retrieve it from the validator bezel.



#### Packet examples

Poll response showing unsafe bill jam detected

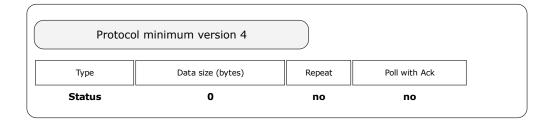
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E9 4A 22** 

Event	Code hex	Code decimal
Disabled	0xE8	232

BV100, BV20, BV50, COUPON PRINTER, NV10USB, NV11, NV150, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, SMART TICKET, TEBS

#### Description

A disabled event is given in response to a poll command when a device has been disabled by the host or by some other internal function of the device.



#### Packet examples

Response to poll showing disabled event

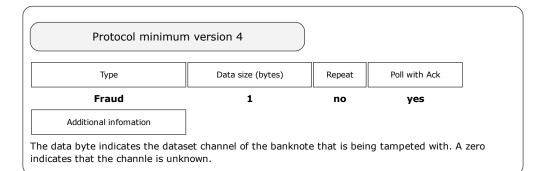
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E8 4F A2** 

Event	Code hex	Code decimal
Fraud Attempt	0xE6	230

BV100, BV20, NV10USB, NV150, NV200, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, TEBS

#### Description

The validator system has detected an attempt to mauipulate the coin/banknote in order to fool the system to register credits with no monies added.



#### Packet examples

Poll response showing fraud attempt seen on channel 2

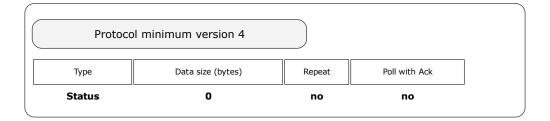
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 E6 02 C0 7C** 

Event	Code hex	Code decimal
Stacker Full	0xE7	231

BV100, BV20, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

Event in response to poll given when the device has detected that the stacker unit has stacked it's full limit of banknotes.



#### Packet examples

Poll response showing stacker full

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E7 6D A2** 

Event	Code hex	Code decimal
Note Cleared From Front	0xE1	225

# Implemented on BV100, BV50, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

During the device power-up sequence a bill was detected as being in the note path. This bill is then rejected from the device via the bezel and this event is issued. If the bill value is known then the channel number is given in the data byte, otherwise the data byte will be zero value.

#### Packet examples

Poll response showing unknown bill rejected from the front at power-up

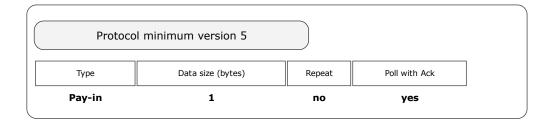
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 E1 00 CC 6E** 

Event	Code hex	Code decimal
Note Cleared Into Cashbox	0xE2	226

BV100, BV50, NV10USB, NV11, NV150, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

During the device power-up sequence a bill was detected as being in the stack path. This bill is then moved into the device cashbox and this event is issued. If the bill value is known then the channel number is given in the data byte, otherwise the data byte will be zero value.



#### Packet examples

Poll response showing a channel 2 bill moved to the cashbox at power-up

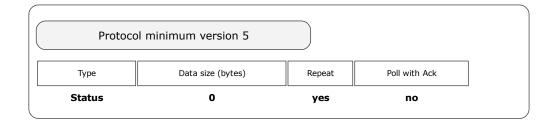
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 E2 02 C3 E4** 

Event	Code hex	Code decimal
Cashbox Removed	0xE3	227

Implemented on
BV100, BV50, NV200, SMART PAYOUT, TEBS

The system has detected that the cashbox unit has been removed from it's working position.

The system will remain disabled for bill entry until the cashbox unit is replaced into it's working position.



#### Packet examples

Poll response showing cashbox removed

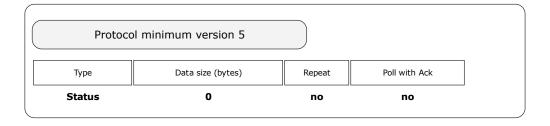
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E3 76 22** 

Event	Code hex	Code decimal
Cashbox Replaced	0xE4	228

Implemented on
BV100, BV50, NV200, SMART PAYOUT, TEBS

The device cashbox box unit has been detected as replaced into it's working position.

The validator will re-enable if it has not alreday been disabled by the host system.



#### Packet examples

Poll response showing cashbox replaced

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E4 67 A2** 

Event	Code hex	Code decimal
Barcode Ticket Validated	0xE5	229

Implemented on	
NV150, NV200, SMART PAYOUT, TEBS	

A barcode ticket has been scanned and identified by the system and is currently held in the escrow position.

The host can send the <u>Get Barcode Data</u> command to retrive the number of the ticket scanned. The host can then sedn a Reject or Poll command to reject or accept the ticket as required.

#### Packet examples

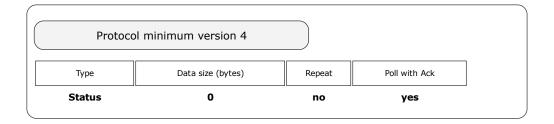
Poll response showing bar code held in escrow

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E5 62 22** 

Event	Code hex	Code decimal
Barcode Ticket Ack	0xD1	209

Implemented on	
NV150, NV200, SMART PAYOUT, TEBS	

The device has moved the barcode ticket to a safe stack position.



### Packet examples

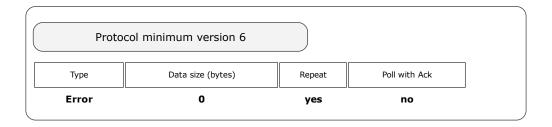
Poll response showing bar code ticket ack

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 D1 D9 A2** 

Event	Code hex	Code decimal
Note Path Open	0×E0	224

Implemented on
NV150, NV200, SMART PAYOUT, TEBS

The device has detected that it's note path has been opened. The device will be disabled for bill entry until the note path is re-closed.



#### Packet examples

Poll response showing note path open

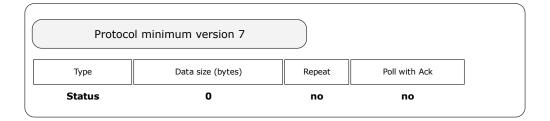
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 E0 7C 22** 

Event	Code hex	Code decimal
Channel Disable	0xB5	181

BV100, BV20, BV50, NV10USB, NV11, NV200, NV9USB, SMART PAYOUT, TEBS

#### Description

The device has had all its note channels inhibited and has become disabled for note insertion.



#### Packet examples

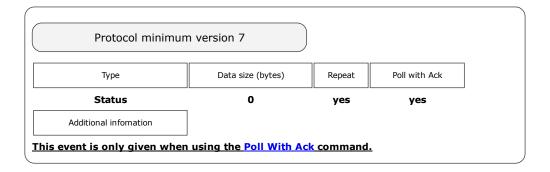
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 B5 82 23** 

Event	Code hex	Code decimal
Initialising	0xB6	182

BV100, BV20, BV50, NV200, NV9USB, SMART HOPPER, SMART PAYOUT, SMART SYSTEM, TEBS

#### Description

This event is given only when using the Poll with ACK command. It is given when the BNV is powered up and setting its sensors and mechanisms to be ready for Note acceptance. When the event response does not contain this event, the BNV is ready to be enabled and used.



#### Packet examples

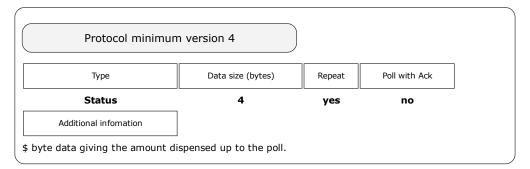
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 B6 88 23** 

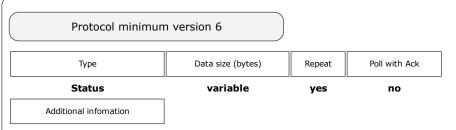
Event	Code hex	Code decimal
Dispensing	0xDA	218

# Implemented on NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

The device is in the process of paying out a requested value. The value paid at the poll is given in the event data.





An array of data giving the dispensed at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value dispensed up to this point	4
5	country code	3
	repeat above block for each country in set	

#### Packet examples

Protocol version 5 poll response showing 12.50 dispensed at this point

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 05 F0 E2 04 00 00 F8 4A** 

Protocol version 6 poll response showing 23.00 EUR and 12.00 GBP dispensed to this point

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 10 F0 02 FC 08 00 00 45 55 52 B0 04 00 00 47 42 50 04 B3** 

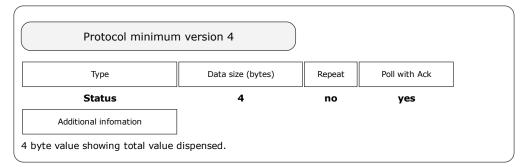
ascii: . . . . . E U R . . . G B P

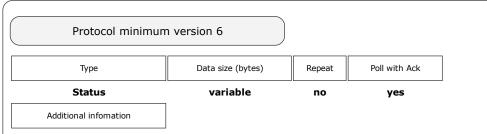
Event	Code hex	Code decimal
Dispensed	0xD2	210

# Implemented on SMART PAYOUT, SMART SYSTEM

#### Description

Show the total value the device has dispensed in repsonse to a <u>Dispense</u> command.





An array of data giving the total dispensed for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value dispensed	4
5	country code	3
	repeat above block for each country in set	

### Packet examples

#### << back to index

Event	Code hex	Code decimal
Coins Low	0xD3	211

Implemented on	
SMART HOPPER	

Description	
-------------	--

## Packet examples

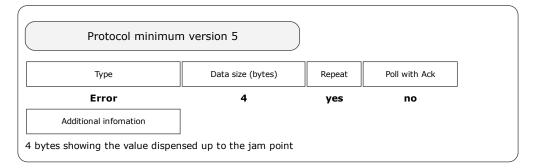
Event	Code hex	Code decimal
Hopper Jammed	0xD5	213

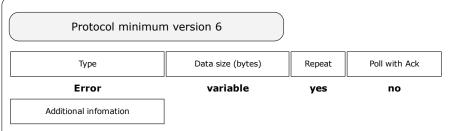
# Implemented on SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

An event showing the hopper unit has jammed and giving the value paid/floated upto that jam.

On the smart payout this event is used when a jam occurs during a payout / float / empty operation.





An array of data giving the dispensed/floated at the jammed point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value dispensed/floated up to this point	4
5	country code	3
	repeat above block for each country in set	

#### Packet examples

Protocol version 5 poll response showing 2.30 paid up to the jam point

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 06 F0 D5 E6 00 00 00 49 DB** 

Event	Code hex	Code decimal
Halted	0xD6	214

Implemented on	
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	

Triggered when payout is interrupted for some reason.

#### **Protocol Version 6 and earlier**

This event is given when:

- the host has requested a halt to the device.
- the payout is automatically cancelled (due to a jam/reverse validation fail/cashbox error etc.)

The value paid at the point of halting is given in the event data.

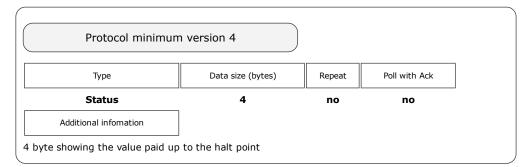
#### **Protocol Version 7 and later**

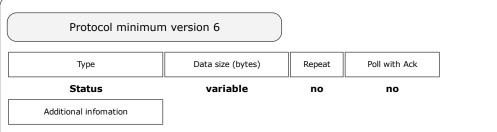
This event is given when:

• the host has requested a halt to the device.

The value paid at the point of halting is given in the event data.

Note: a different event 'Error During Payout' is generated when errors occur





An array of data giving the dispensed/floated at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value dispensed/floated up to this point	4
5	country code	3
	repeat above block for each country in set	

#### Packet examples

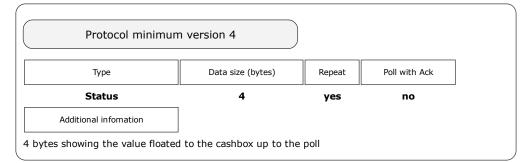
Host transmit: **7F 80 01 07 12 02** 

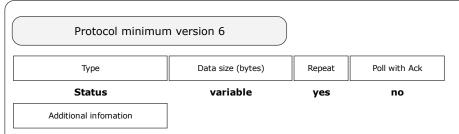
Event	Code hex	Code decimal
Floating	0xD7	215

# Implemented on SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

Event showing the amount of cash floated up to the poll point





An array of data giving the floated value at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value floated to this point	4
5	country code	3
	repeat above block for each country in set	

#### Packet examples

Protocol version 5 poll response showing 45.00 floated

Host transmit: 7F 80 01 07 12 02

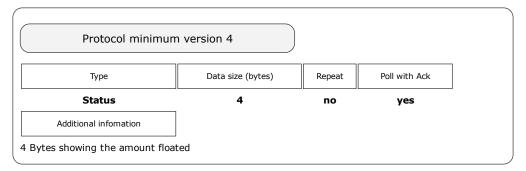
Slave Reply: **7F 80 05 F0 94 11 00 00 E8 F3** 

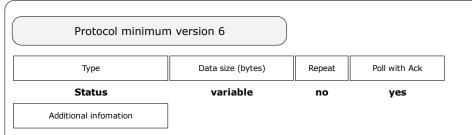
Event	Code hex	Code decimal
Floated	0xD8	216

# Implemented on SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

Event given at the end of the floating process which will display the amount actually floated.





An array of data giving the floated value at the end of the process for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value floated	4
5	country code	3
	repeat above block for each country in set	

#### Packet examples

Protocol version 6 poll response showing a floated value of 20.50 EUR

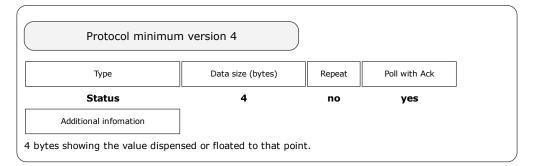
Host transmit: **7F 80 01 07 12 02** 

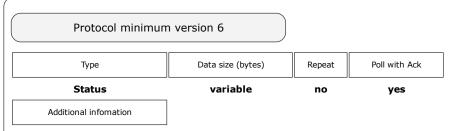
Slave Reply: **7F 80 0A F0 D8 01 02 08 00 00 45 55 52 81 C0** ascii: . . . . . . . . . **E U R** 

Event	Code hex	Code decimal
Timeout	0xD9	217



The device has been unable to complete a request. The value paid up until the time-out point is given in the event data.





An array of data giving the dispensed/floated at the poll point for each of the countries supported in the dataset. The first byte gives the number of countries in the set the a block of data for each of the countries.

byte	function	size
0	number of countries in set	1
1	value dispensed/floated up to this point	4
5	country code	3
	repeat above block for each country in set	

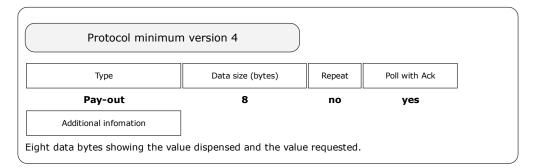
#### Packet examples

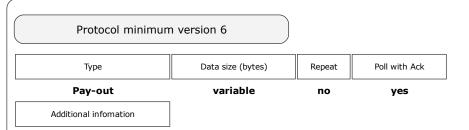
Event	Code hex	Code decimal
Incomplete Payout	0xDC	220

# Implemented on SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

The device has detected a discrepancy on power-up that the last payout request was interrupted (possibly due to a power failure). The amounts of the value paid and requested are given in the event data.





An array of data giving the value dispensed and the original value requested before the power down for each of the countries supported in the dataset. The first byte gives the number of countries in the set then a block of data for each of the countries (see table below).

byte	function	size
0	number of countries in set	1
1	value dispensed	4
5	value requested	4
9	country code (ASCII)	3
	repeat above block for each country in set	

#### Packet examples

Protocol version 5 poll response showing 25.20 paid out of request for 50.00

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 09 F0 D8 09 00 00 58 0D 00 00 3B C9** 

Host transmit: **7F 80 01 07 12 02** 

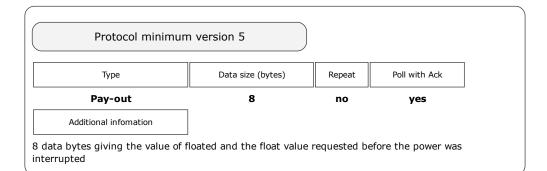
Slave Reply: 7F 80 0D F0 01 FC 08 00 00 88 13 00 00 45 55 52 C3 E5

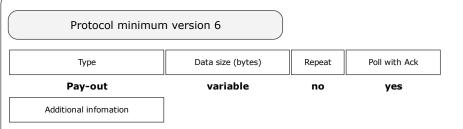
Event	Code hex	Code decimal
Incomplete Float	0xDD	221

# Implemented on SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

The device has detected a discrepancy on power-up that the last float request was interrupted (possibly due to a power failure). The amounts of the value paid and requested are given in the event data.





An array of data giving the value floated and the original value requested before the power down for each of the countries supported in the dataset. The first byte gives the number of countries in the set then a block of data for each of the countries (see table below).

byte	function	size
0	number of countries in set	1
1	value floated	4
5	value requested	4
9	country code (ASCII)	3
	repeat above block for each country in set	

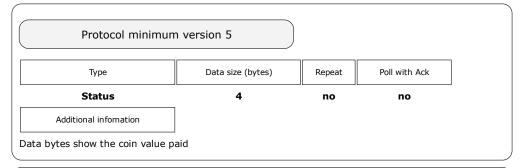
### Packet examples

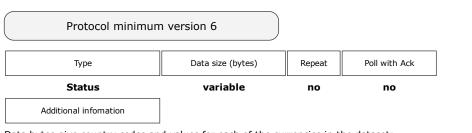
Event	Code hex	Code decimal
Cashbox Paid	0xDE	222

# Implemented on SMART HOPPER, SMART SYSTEM

#### Description

Coin values have been detected and paid to the cashbox since the last poll.





Data bytes give country codes and values for each of the currencies in the dataset:

function	size
number of countries in set	1
value dispensed	4
country code	3
repeat above block for each country in set	
	number of countries in set  value dispensed  country code

#### Packet examples

Protocol version 5 poll response showing 2.00 (200 c) coin paid to cashbox

Host transmit: **7F 90 01 07 51 83** 

Slave Reply: **7F 90 06 F0 DE C8 00 00 00 68 00** 

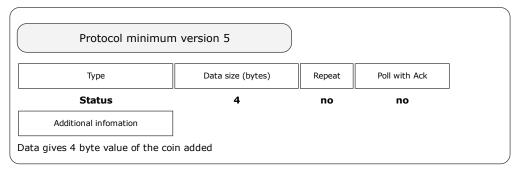
Protocol version 6 poll response showing 5.30 GBP adn 0.20 EUR paid to cashbox

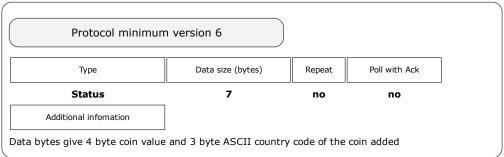
Host transmit: **7F 90 01 07 51 83** 

Event	Code hex	Code decimal
Coin Credit	0xDF	223

Implemented on
SMART HOPPER

A coin has been detected as added to the system. This would be usually via the seperate coin mech attached to the system port.





#### Packet examples

Protocol version 5 poll response showing 1.00 (100 c) coin added

Host transmit: **7F 90 01 07 51 83** 

Slave Reply: **7F 90 05 F0 64 00 00 00 97 A3** 

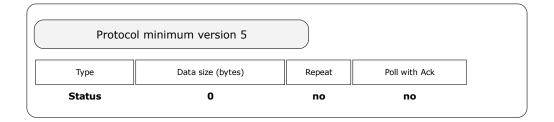
Protocol version 6 poll response showing 5.00 GBP coin added

Host transmit: 7F 90 01 07 51 83

Event	Code hex	Code decimal
Coin Mech Jammed	0xC4	196

Implemented on
SMART HOPPER, SMART SYSTEM

The attached coin mechanism has been detected as having a jam.



### Packet examples

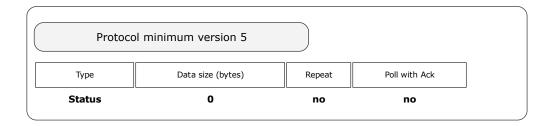
Poll response showing coin mech jam

Host transmit: **7F 90 01 07 51 83**Slave Reply: **7F 90 02 F0 C4 A2 62** 

Event	Code hex	Code decimal
Coin Mech Return Active	0xC5	197



The attached coin mechanism has been detected as having it's reject or return button pressed.

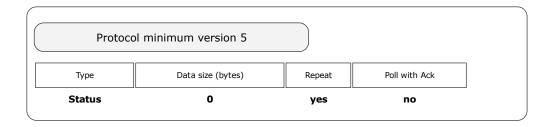


#### Packet examples

Event	Code hex	Code decimal
Emptying	0xC2	194

Implemented on	
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM	

The device is currently performing is empty operation following an <a href="Empty"><u>Empty</u></a> command request.



### Packet examples

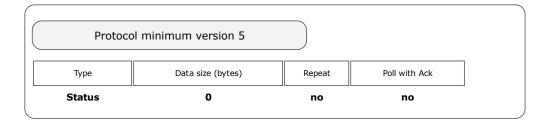
Poll response showing device emptying

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 C2 B0 22** 

Event	Code hex	Code decimal
Emptied	0xC3	195

Implemented on
NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM

The device has completed it's empty operation in response to the **Empty** command.



### Packet examples

Poll response showing device emptied

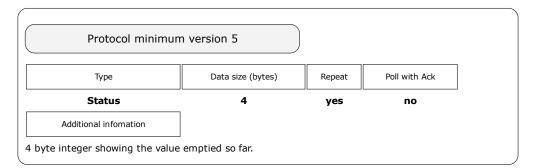
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 C3 B5 A2** 

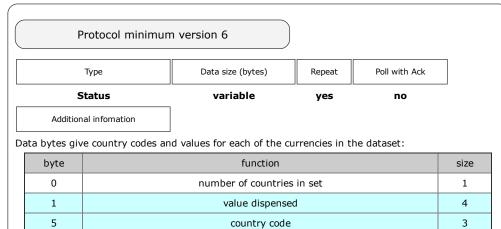
Event	Code hex	Code decimal
Smart Emptying	0xB3	179

# Implemented on NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

The device is in the process of carrying out its Smart Empty command from the host. The value emptied at the poll point is given in the event data





#### Packet examples

repeat above block for each country in set

A device has emptied 22.60 EUR up to this poll with protocol version 5

Host transmit:  $\mathbf{7F}$  80 01 07 12 02

...

Slave Reply: 7F 80 07 F0 B3 01 D4 08 00 00 53 F7

A device has emptied 22.60 EUR up to this poll with protocol version 6

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 0A F0 B3 01 D4 08 00 00 45 55 52 44 F6** 

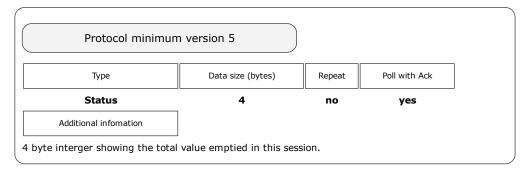
ascii: . . . . . . E U R

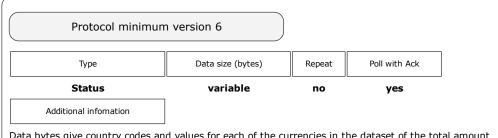
Event	Code hex	Code decimal
Smart Emptied	0xB4	180

# Implemented on NV11, SMART HOPPER, SMART PAYOUT, SMART SYSTEM

#### Description

The device has completed its Smart Empty command. The total amount emptied is given in the event data.





Data bytes give country codes and values for each of the currencies in the dataset of the total amount emptied.

byte	function	size
0	number of countries in set	1
1	value dispensed	4
5	country code	3
	repeat above block for each country in set	

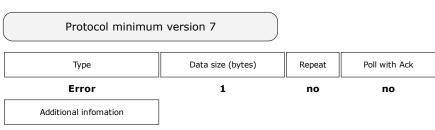
# Packet examples

Event	Code hex	Code decimal
Calibration Failed	0x83	131

# Implemented on SMART HOPPER, SMART SYSTEM

#### Description

During the devices normal re-calibration process, an error has been detected which indicates a sensor failure or out-of-range issue. This usually indicate a hardware failure and the device should be taken out of service until the cause is found.



A data byte error reason is given detailed in the table below.

Error	Code
	0
Payout flap sensor	1
Exit sensor	2
Coil 1 sensor	3
Coil 2 sensor	4
Unit not initialised	5
Checksum error	6
Recalibration by command required (obsolete)	7
Motor opto slot error	8,9
Exit sensor error 2	10

#### Packet examples

The example below shows a calibration fail due to an issue with coil 1.

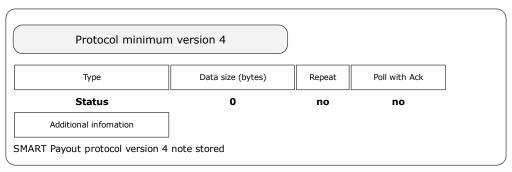
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 83 03 C0 22** 

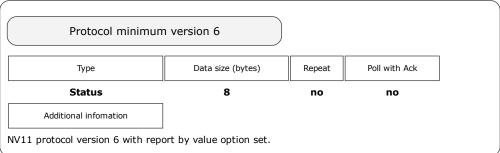
Event	Code hex	Code decimal
Note Stored In Payout	0xDB	219

Implemented on	
NV11, SMART PAYOUT	

The note has been passed into the note store of the payout unit.

# Note that NV11 devices report a value of note stored if Report By Value option has been set.





# Packet examples

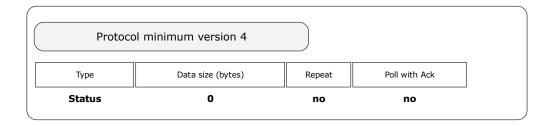
Poll response showing note stored in payout for SMART Payout

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 DB E5 A2** 

Event	Code hex	Code decimal
Payout Out Of Service	0xC6	198

Implemented on
NV11

This event is given if the payout goes out of service during operation. If this event is detected after a poll, the host can send the ENABLE PAYOUT DEVICE command to determine if the payout unit comes back into service.



#### Packet examples

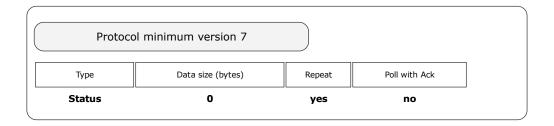
Poll response showing payout out of service

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 C6 AB A2** 

Event	Code hex	Code decimal
Jam Recovery	0xB0	176

Implemented on	
SMART PAYOUT	

The SMART Payout unit is in the process of recovering from a detected jam. This process will typically move five notes to the cash box; this is done to minimise the possibility the unit will go out of service.



#### Packet examples

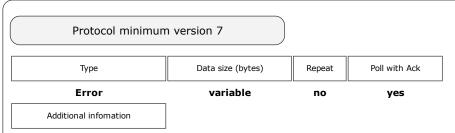
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 B0 9C 23** 

Event	Code hex	Code decimal
Error During Payout	0xB1	177

Implemented on	
SMART PAYOUT	

Returned if an error is detected whilst moving a note inside the SMART Payout unit. The cause of error (1 byte) indicates the source of the condition - see table below for error causes.

In the case of the incorrect detection, the response to Cashbox Payout Operation Data request would report the note expected to be paid out.



The data with this event has variable length depending on the number of dataset denominations in the device:

byte	function	size
0	number of countries in set	1
1	value dispensed	4
5	country code	3
	repeat above block for each country in set	
last	Final byte is an error code (see table below)	1

#### Error Code (final byte from above):

Value	Meaning
0x00	note not correctly detected as it is routed (reverse validation fail)
0x01	note jammed in transport*
0x02	cashbox error e.g. stacker full. removed, jammed**
0x03	payout stalled e.g. unable to seek note in payout
0x04	payout cancelled due to poll timeout

<sup>\*</sup> this error can be reported for different fault types - such as a note missing from the cashbox - as the unit only knows that the note does not arrive at payout exit

<sup>\*\*</sup> stacker may be required during payout (for recovery or stacking poor condition notes)

# Packet examples

Payout error due to jam after GBP 50.00 and EUR 20.00 have been paid

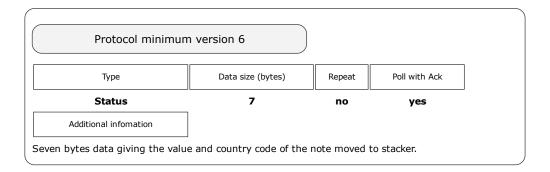
Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 0F F0 B1 02 88 13 00 00 47 42 50 D0 07 00 00 01 34 B3** 

Event	Code hex	Code decimal
Note Transfered To Stacker	0xC9	201



Reported when a note has been successfully moved from the payout store into the stacker cashbox.



#### Packet examples

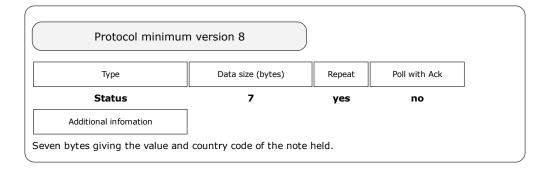
Poll response showing  $5.00\ EUR$  note moved from payout to stacker

Host transmit: **7F 80 01 07 12 02** 

Event	Code hex	Code decimal
Note Held In Bezel	0xCE	206

Implemented on
NV11, SMART PAYOUT

Reported when a dispensing note is held in the bezel of the payout device.



Packet examples

Poll response showing 10.00 EUR bill held in bezel

Host transmit: **7F 80 01 07 12 02** 

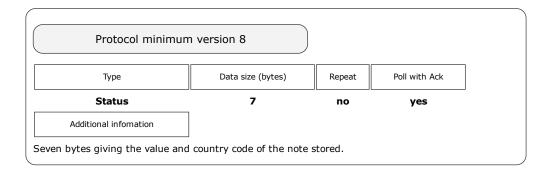
Slave Reply: 7F 80 09 F0 CE E8 03 00 00 45 55 52 08 54

ascii: . . . . . E U R

Event	Code hex	Code decimal
Note Into Store At Reset	0xCB	203

Implemented on	
NV11, SMART PAYOUT	

An event showing that a bill was moved into the paout storage as part of the power-up proceedure.

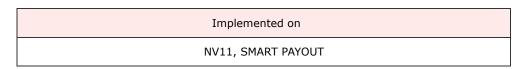


#### Packet examples

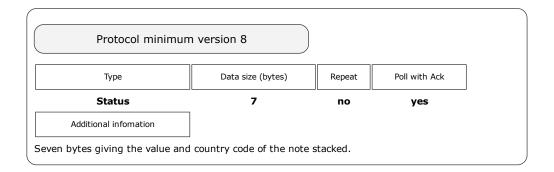
Poll response showing a 20.00 GBP note move to payout store during power-up

Host transmit: **7F 80 01 07 12 02** 

Event	Code hex	Code decimal
Note Into Stacker At Reset	0xCA	202



Reported when a note has been detected as paid into the cashbox stacker as part of the power-up procedure.



#### Packet examples

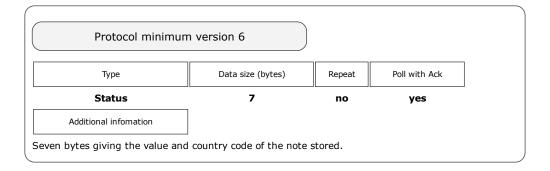
Poll response showing 5.00 EUR note stacked at power up

Host transmit: **7F 80 01 07 12 02** 

Event	Code hex	Code decimal
Note Dispensed At Reset	0xCD	205

Implemented on	
NV11	

Reported when a note has been dispensed as part of the power-up procedure.



#### Packet examples

Poll response showing 10.00 EUR note stored at power up

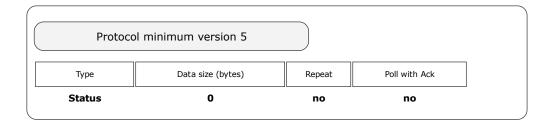
Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 09 F0 CD E8 03 00 00 45 55 52 02 64** ascii: **. . . . . . . E U R** 

Event	Code hex	Code decimal
Note Float Removed	0xC7	199

Implemented on	
NV11	

Reported when a note float unit has been detected as removed from its validator.



# Packet examples

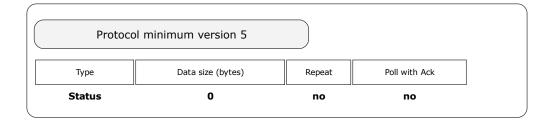
Poll response showing note float unit removed

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 C7 AE 22** 

Event	Code hex	Code decimal
Note Float Attached	0xC8	200

Implemented on	
NV11	

Reported when a note float unit has been attached to its validator.



Packet examples

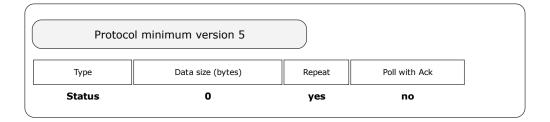
Poll response showing note float attached

Host transmit: **7F 80 01 07 12 02** Slave Reply: **7F 80 02 F0 C8 8C 22** 

Event	Code hex	Code decimal
Device Full	0xCF	207

Implemented on	
NV11, SMART SYSTEM	

The device has detected that it is full of coins/banknotes and no more can be added.



# Packet examples

Event	Code hex	Code decimal
Coin Mech Error	0xB7	183

Implemented on	
SMART HOPPER, SMART SYSTEM	

This event will only be gererated if the <u>Coin Mech Options</u> command has been sent to the device with data bit set to enable error events.

The data byte given with this event indicates the error type.

Code

Frror

Description

1

Reject coin

A coin was inserted which did not match any of the programmed types. The coin is returned to the customer and no credit is given.

2

Inhibited coin

A coin was inserted which did match a programmed window type but was prevented from accepting by the inhibit register. The inhibit register can be controlled serially but may also be linked to external DIL switches.

3

Multiple window

A coin was inserted which matched more than one enabled window type. This coin was rejected as the credit code was indeterminate.

4

Wake-up timeout

A coin acceptor fitted with a wake-up sensor picked up a coin entering the acceptor but it was not seen subsequently in the validation area. Possible coin jam.

5

Validation timeout

A coin was detected entering the validation area but failed to leave it. Possible coin jam.

6

Credit sensor timeout

A coin was validated as true but never made it to the post-gate credit sensor. Possible coin jam.

7

Sorter opto timeout

A coin was sent into the sorter / diverter but was not seen coming out. Possible coin jam.

8

2nd close coin error

A coin was inserted too close to the one in front. One or both coins will have rejected.

9

Accept gate not ready

A coin was inserted while the accept gate for the coin in front was still operating. Coins have been inserted too quickly.

10

Credit sensor not ready

A coin was still over the credit sensor when another coin was ready to accept. Coins have been inserted too quickly.

11

Sorter not ready

A coin was inserted while the sorter flaps for the coin in front were still operating. Coins have been inserted too quickly.

12

Reject coin not cleared

A coin was inserted before a previously rejected coin had time to clear the coin acceptor. Coins have been inserted too quickly.

13

Validation sensor not ready

The validator inductive sensors were not ready for coin validation. Possible fault developing.

14

Credit sensor blocked

There is a permanent blockage at the credit sensor. The coin acceptor will not accept any more coins.

15

Sorter opto blocked

There is a permanent blockage at the sorter exit sensor. The coin acceptor will not accept any more coins.

16

Credit sequence error

A coin or object was detected going backwards through a directional credit sensor. Possible fraud attempt.

Coin going backwards

A coin was detected going backwards through the coin acceptor. Possible fraud attempt.

18

Coin too fast ( over credit sensor )

A coin was timed going through the credit sensor and was too fast. Possible fraud attempt.

19

Coin too slow (over credit sensor)

20

C.O.S. mechanism activated

(coin-on-string)

A specific sensor for detecting a 'coin on string' was activated. Possible fraud attempt.

21

DCE opto timeout

A coin acceptor fitted with a Dual Coin Entry chute saw a coin or token which was not seen subsequently in the validation area. Possible coin jam.

22

DCE opto not seen

A coin acceptor fitted with a Dual Coin Entry chute saw a coin which was not seen previously by the chute sensor. Possible fraud attempt.

23

Credit sensor reached too early

A coin was timed from the end of the validation area to the post-gate credit sensor. It arrived too early. Possible fraud attempt.

24

Reject coin ( repeated sequential trip )

A coin was rejected N times in succession with no intervening true coins. Statistically unlikely if N greater than or equal to 5. Possible fraud attempt.

25

Reject slug

A coin was rejected but was identified as a known slug type - this may be a preprogrammed fraud coin or a known fraud material.

26

Reject sensor blocked

There is a permanent blockage at the reject sensor. The coin acceptor will not accept any more coins. Not all coin acceptors have a reject sensor.

27

Games overload

Totaliser mode : A game value was set too low - possibly zero. This is a product configuration error.

28

Max. coin meter pulses exceeded

Totaliser mode: A meter value was set too low - possibly zero. This is a product configuration error.

29

Accept gate open not closed

The accept gate was forced open when it should have been closed.

30

Accept gate closed not open

The accept gate did not open when the solenoid was driven.

31

Manifold opto timeout

A coin was sent into the manifold module ( coin diverter ) but was not seen coming out. Possible coin jam.

32

Manifold opto blocked

There is a permanent blockage at the manifold module sensor (coin diverter). The coin acceptor will not accept any more coins.

128

Inhibited coin (Type 1)

A true coin ( type 1, coin in position 1 ) was inserted but was prevented from accepting by the inhibit register.

Inhibited coin (Type n)

A true coin ( type n, coin in position n ) was inserted but was prevented from accepting by the inhibit register.

159

Inhibited coin (Type 32)

A true coin ( type 32, coin in position 32 ) was inserted but was prevented from accepting by the inhibit register.

253

Data block request (note a)

A 'not yet used' mechanism for a coin acceptor to request attention from the host machine. Perhaps it needs some data from the host machine or another peripheral.

254

Coin return mechanism activated

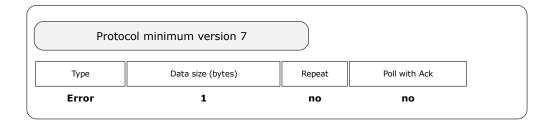
(Flight deck open)

An attempt to clear a coin jam by opening the flight deck was detected. The coin acceptor cannot operate until the flight deck is closed.

255

Unspecified alarm code

Any alarm code which does not fit into the above categories.



#### Packet examples

A coin error: too slow detected

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 B7 14 B1 1A** 

Event	Code hex	Code decimal
Coin Rejected	0xBA	186

Implemented on	
SMART SYSTEM	

This event will be returned if the 'Reject Events' bit is set in the 'Set Options' command. A 4 byte count of the number of coins rejected followed by a 3 byte country code.

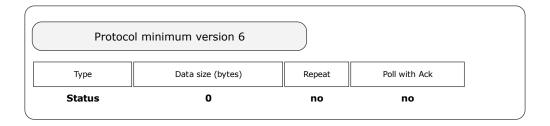
Or if also the 'Reject Full' bit is set in the 'Set Options' command then for each coin rejected a 4 byte value (if known) and 3 byte country code will be returned.

Packet examples

Event	Code hex	Code decimal
Attached Coin Mech Disabled	0xBD	189

Implemented on
SMART HOPPER, SMART SYSTEM

The device seperate coin mechanism attached to this device has been disabled.



# Packet examples

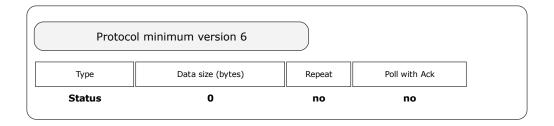
Poll response showing coin mech disabled

Host transmit: **7F 90 01 07 51 83**Slave Reply: **7F 90 02 F0 BD B7 E3** 

Event	Code hex	Code decimal
Attached Coin Mech Enabled	0xBE	190

Implemented on	
SMART HOPPER, SMART SYSTEM	

The seperate coin mechanism attached to this device has been enabled.



# Packet examples

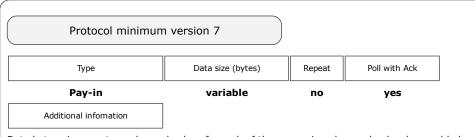
Poll response showing coin mech enabled

Host transmit: **7F 90 01 07 51 83**Slave Reply: **7F 90 02 F0 BE BD E3** 

Event	Code hex	Code decimal
Value Added	0xBF	191

Implemented on
SMART SYSTEM

An event giving the cumulative value of currency detected as added to the system since the last poll.



Data bytes give country codes and values for each of the currencies where value has been added

byte	function	size
0	Generic OK	1
1	Event code	1
2	number of countries in data	1
3	value added (4 byte integer)	4
7	country code (3 Byte ASCII)	3
	repeat above block for each country data	

#### Packet examples

5.50 EUR has been added since the last poll

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 0A F0 BF 01 26 02 00 00 45 55 52 ED 91** 

2.20 EUR and 3.60 GBP have been added since the last poll

Host transmit: **7F 80 01 07 12 02** 

Slave Reply: 7F 80 11 F0 BF 02 DC 00 00 00 45 55 52 68 01 00 00 47 42 50 D1 05

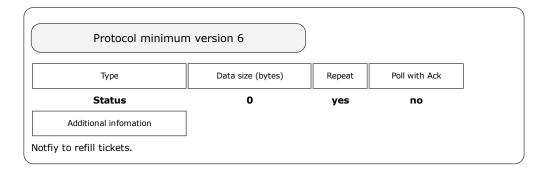
Event	Code hex	Code decimal
Tickets Low	0xA0	160

Implemented on

COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET

#### Description

This event is reported when the level of tickets in the device are detected as being at a low level on the device's ticket level sensor.



Notice: Undefined offset: 0 in /var/www/vhosts/software/phpitlcommon.php on line 44

Packet examples

Host transmit:

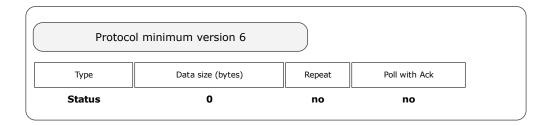
Slave Reply: **7F 80 02 F0 A0 FF A3** 

Event	Code hex	Code decimal
Tickets Replaced	0xA1	161

# Implemented on COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET

#### Description

This event is reported when the level of tickets has been detected as going over the low level again.



# Packet examples

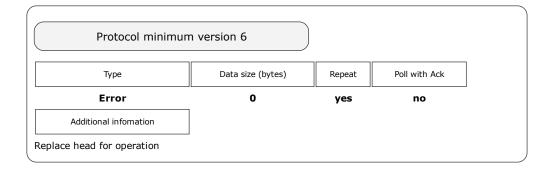
Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 A1 FA 23** 

Event	Code hex	Code decimal
Printer Head Removed	0xA2	162

# Implemented on COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET

#### Description

The head for the printer has been taken out and tickets cannot be printed.



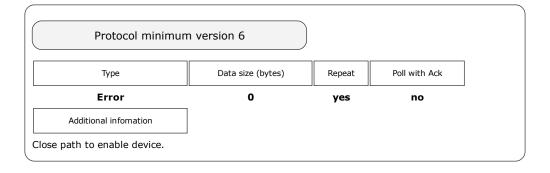
#### Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 A2 F0 23** 

Event	Code hex	Code decimal
Ticket Path Open	0xA3	163

Implemented on	
FLATBED PRINTER, SMART TICKET	

The printer's path has been opened, and tickets cannot be printed.

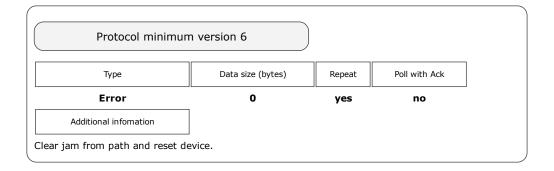


# Packet examples

Event	Code hex	Code decimal
Ticket Jam	0xA4	164

Implemented on
COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET

A jam occured when attempting to print a ticket.



# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 A4 E4 23** 

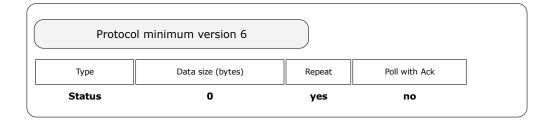
Event	Code hex	Code decimal
Ticket Printing	0×A5	165

#### Implemented on

COUPON PRINTER, FLATBED PRINTER, NV12, NV200, NV9USB, SMART TICKET

#### Description

A ticket is currently being printed. On an NV200 this event will only be reported if there is a SMART Ticket attached, and ticket events have been enabled.



# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 A5 E1 A3** 

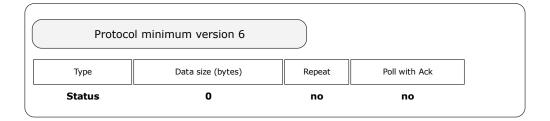
Event	Code hex	Code decimal
Ticket Printed	0xA6	166

#### Implemented on

COUPON PRINTER, FLATBED PRINTER, NV12, NV200, NV9USB, SMART TICKET

#### Description

A ticket has successfully been printed and dispensed. On an NV200 this event will only be reported if there is a SMART Ticket attached, and ticket events have been enabled.



# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 A6 EB A3** 

Event	Event Code hex	
Ticket Printing Error	0xA8	168

#### Implemented on

COUPON PRINTER, FLATBED PRINTER, NV12, NV200, NV9USB, SMART TICKET

#### Description

Unable to print the requested ticket. The event includes a data byte indicating the reason for failure:

Error	Code	Devices
No paper	0	SMART Ticket, Coupon Printer
Load fail	1	SMART Ticket, Coupon Printer
No head	2	SMART Ticket, Coupon Printer
Diverter did not open	3	SMART Ticket
Diverter did not close	4	SMART Ticket
Burst fail	5	SMART Ticket
Cut fail	6	SMART Ticket, Coupon Printer
Reverse validate fail	7	SMART Ticket, NV200
Jam	8	SMART Ticket, NV200
NV200 fail	9	SMART Ticket
NV200 Timeout	10	SMART Ticket
NV200 Cashbox Error	17	NV200
SMART Ticket Timeout	19	NV200

On an NV200 this event will only be reported if there is a SMART Ticket attached, and ticket events have been enabled.

The SMART Ticket will report reasons 0 to 10 as an error. If the error is with the NV200, it will report NV200 Fail or NV200 Timeout. The NV200 will report reason 7, 8, 17 or 19. The two devices will generally report different errors. Jam from a SMART Ticket refers to a specific jam in transit from the SMART Ticket to the NV200 when reported from the SMART Ticket. From the NV200, a jam could be any of the jam conditions the SMART Ticket may encounter, and the event data from the SMART Ticket should be defered to.

Protocol minimum version 6

Туре	Data size (bytes)	Repeat	Poll with Ack
Error	1	no	no
Additional infomation			
Examine error byte cause and remedy.			

# Packet examples

Show print fail response due to jam

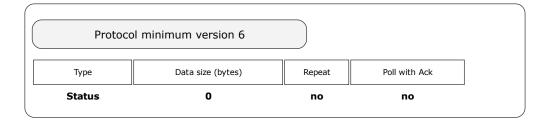
Host transmit: **7F 80 01 07 12 02** 

Slave Reply: **7F 80 03 F0 A8 08 F9 58** 

Event	Code hex	Code decimal
Printer Head Replaced	0xA9	169

Implemented on
COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET

The printer head was replaced after being removed.



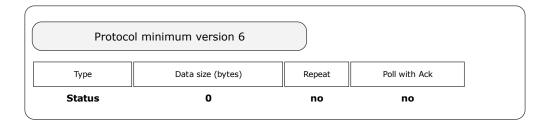
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 A9 C9 A3** 

Event	Code hex Code decimal	
Ticket Path Closed	0xAA	170

Implemented on
FLATBED PRINTER, SMART TICKET

The ticket path was closed after being opened.



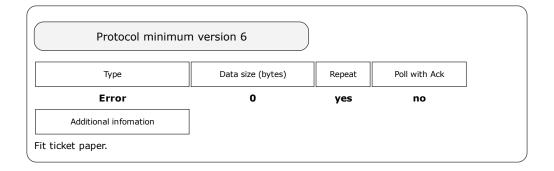
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 AA C3 A3** 

Event	Code hex	Code decimal
No Paper	0xAB	171

Implemented on	
COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET	

There is no paper currently fed into the device.



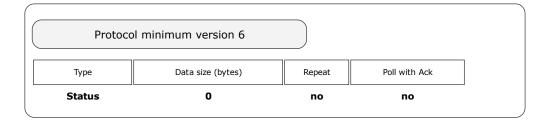
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 AB C6 23** 

Event	Code hex	Code decimal
Print Halted	0×AE	174

Implemented on
NV12, NV200, NV9USB, SMART TICKET

The ticket printing process stopped.



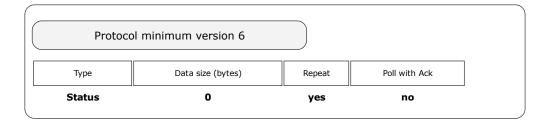
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 AE D8 23** 

Event	Code hex	Code decimal
Ticket In Bezel	0xAD	173

Implemented on
NV12, NV200, NV9USB, SMART TICKET

Printed ticket is held in bezel.



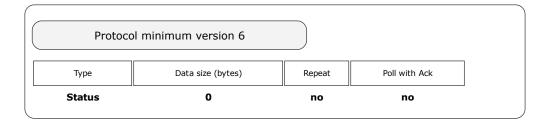
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 AD D2 23** 

Event	Code hex	Code decimal
Paper Replaced	0xAC	172

Implemented on
COUPON PRINTER, FLATBED PRINTER, NV12, SMART TICKET

Ticket paper was replaced in the device.



# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 AC D7 A3** 

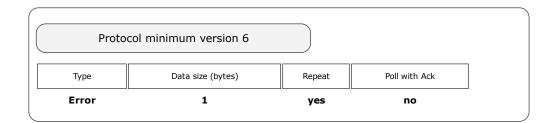
Event	Code hex	Code decimal
Cashbox Out Of Service	0x90	144

Implemented on	
TEBS	

An error has been detected pertaining to the TEBS cashbox. This event has an associated error code byte.

#### Error code:

Error	
No device connected	1
Unable to read barcode	2
Cashbox out of position	3
Cashbox removed	4
Cashbox unlocked	5
Currency Mismatch (Bag currency does not match the NV200 head and must be replaced)	6
Firmware Error	7
Tebs comms error	8



#### Packet examples

Show error cashbox removed

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 03 F0 90 04 D2 48** 

Event	Code hex	Code decimal
Printed To Cashbox	0xAF	175

Implemented on
NV12, NV200, NV9USB, SMART TICKET

A printed ticket has beed stored in the device cashbox.

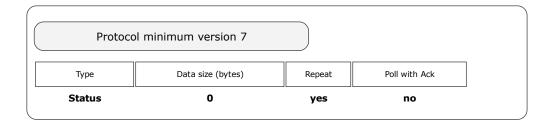
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 AF DD A3** 

Event	Code hex	Code decimal
Pay-in Active	0xC1	193

Implemented on	
SMART SYSTEM	

The pay-in function of the system is active.



Packet examples

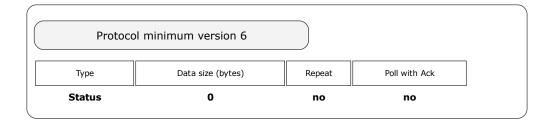
Poll response showing pay-in function is active

Host transmit: **7F 90 01 07 51 83**Slave Reply: **7F 90 02 F0 C1 BC 62** 

Event	Code hex	Code decimal
Cashbox Back In Service	0x92	146

Implemented on	
TEBS	

The device cashbox is now back in service.



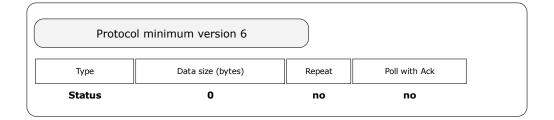
# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 92 50 23** 

Event	Code hex	Code decimal
Cashbox Unlock Enabled	0x93	147

Implemented on	
TEBS	

This event will be reported whenever the cashbox is allowed to be unlocked.



# Packet examples

Host transmit: **7F 80 01 07 12 02**Slave Reply: **7F 80 02 F0 93 55 A3** 

Event	Code hex	Code decimal
Ticket In Bezel At Startup	0xA7	167

Implemented on
FLATBED PRINTER, NV12, NV9USB

A ticket was dispensed out of the front of the device at startup due to power loss during a print. It's possible this ticket print was incomplete, and so any data printed on the ticket should be invalidated.

## Packet examples

Event	Code hex	Code decimal
Maintenance Required	0xC0	192

Implemented on
SMART SYSTEM

The device has run for a large number of cycles and a routine maintenance check should be performed.

On SMART System this event may be given if it has been configured to trigger when no coins have been detected after the coin entry sensor has been activated. This could be due foreign objects in the feeder disc stopping a correct coin feed.

SMART System can be configered to trigger this event after a number of 'no coin pay in'.

## Packet examples

Event	Code hex	Code decimal
Escrow Active	0x8B	139

Implemented on	
SMART SYSTEM	

Description
-------------

This event is repeated while the coin escrow is open.

Packet examples