CashCode®

corporate headquarters
553 Basaltic Road Concord, Ontario Canada L4K 4W8
phone (905) 303-8874 fax (905) 303-8875
web www.cashcode.com

# BI-DIRECTIONAL

## Bill Validator Protocol

## Interface Description

PT# MAN_BDP Rev.B 11/99

# 1. Introduction.

CashCode serial validators are high security validators and can accept bills in all four directions i.e. face up, face down regardless of its positioning in the entry slot. It takes only 2 seconds to transport, identify and stack each bill from start to finish. The acceptance rate is 95% or greater for street banknotes. Features include security to protect against acceptance of fraudulent bills.The Bill Validator is equipped with DIP switches allowing the User select the operating mode and bill combination of the denominations to be accepted.

The electronic Interface of CashCode Bill Validators supports Isolated Pulse InterFace (IP), Non-
Isolated Pulse Interface (NIP), and Non-Isolated Serial Interface (NISR), and Non-Isolated Low Level Pulse (NILLP) modified. These Interfaces establish communication of varying degrees with an external control system. Features included in the Interfaces are: Escrow, lockout, Credit pulse Accumulation, and an Out-of-Service line.

The Escrow features allows the control system to decide whether or not that the valid last bill inserted will be credited or returned to the user. Normally, a valid bill will be accepted and held mid-transport. At this point, the validator will transmit the credit value of the bill. The control system will then make a decision on whether to keep or return the bill. Afterwards, if the bill was kept, the validator completes transportation of the bill and re-transmits the credit value for conformation. The Escrow feature is available using the NISR and NILLP Interfaces.

The Lockout features allows the control system the option of disabling the validator. This function is controlled via the *Accept Enable* line ( see Connector Information, Section 5 ). Normally, the *Accept Enable line* is held low allowing the validator to accept valid bills, but when *Accept Enable* is brought high, the validator is disabled; it will not even attempt to take a bill. The Lockout features is available on the NIP, NISR and NILLP Interfaces.

This Document describes the operation of the bi-directional serial Interface used to communicate between some “Cashcode” Bill Validators and a “ Controller” . In the typical application, the Controller is a processor which is telling the bill validator when to accept currency, which denominations are to be accepted, when to return a bill to the customer, etc.

The functionality of the bill validator (which denominations to accept, whether to hold a bill in escrow,... etc.) can be changed in real-time by the controller while the system is operational. It is not required to power down the system and/or manipulate dip switches on the bill validator’s control board.

Before going into a detailed bit level description of the bi-directional Interface, a set of terms representing “bill validator lingo” is presented here to help understand what the bits and bytes represent. Throughout this document, an important distinction is made between “states” and “events”. A Bill Validator remains in a certain “state” for some period of time (e.g. *IDLING* ) the validator continually reports the state in which it currently resides. In contrast, an “event” is a one-shot deal which is reported to the controller untill the validator believes the message was understood (e.g. *Stacked* a $20 bill).

Note that whenever a description involves an actual field in a bi-directional serial message, the location on that field within the message is given in parenthesis. This will be helpful to refer back to while we are describing the message formats in later sections of the document.

## 1.1. Description of Bill Validator States.

The first list contains definitions of the STATES that the bill validator can pass through during the act of processing a bill. Only one of the bits in this list can be set at any time in any single message. However, the bill validator will usually have a particular state bit set across multiple message.

1. *Idling (Byte 0, Bit 0)* - In this state, the bill validator is not processing a bill and is waiting for a bill to be inserted into the validator. The bill validator is typically performing background checks, communicating as needed with the controller and monitoring its sensors for a bill to be inserted into its front bezel. This state indicates that there are currently no problems detected within the bill validator.

2. *Accepting (Byte 0, Bit 1 )* **-** In this state, a bill is normally being pulled into the bill validator and has not reached the " Escrow" position yet (described next). the bill validator has "seen" a bill being inserted into its bezel and the bill validator was currently enable to accept currency. If the bill has to be returned because of a misfeed or other condition that would not permit the bill to reach the Escrow position or if a bill falls the validation tests and is being rejected, the validator will still report this state until the bill is out of the validator or a jam occurs.

3. *Escrowed (byte 0, Bit 2)* - Typically, the bill validator will run an inserted bill into the validator until the bill reaches the "escrow" position. This is a position where enough information has been retrieved from the bill for the bill validator to make a decision on the validity and denomination of the bill. The entire bill is inside the validator, out of the customer's hands, but is still in a position where it can be returned to the customer if needed. When this state is reported to the controller, the bill has passed the validator tests and the validator is waiting for instructions from the controller to either accept or return the bill. This state will only appear if the Escrow Mode is enabled. When this state bit is reported, the Bill value field will indicate the denomination of the bill just validated. if a bill appears to be in the

Escrow position when power is applied, this state will be reported with the Bill Value field indicating zero for unknown value.

4. *Stacking (Byte 0, Bit 3 )* - This is a Post-escrow state. The validator remains in this state while it is moving a bill from the escrow position toward a fully-secured position past all of the bill validator's internal sensors. This state is initiated per command from the controller (with Escrow Mode enable) or automatically after validation (with Escrow Mode disabled). The validator will remain in this state until the bill is successfully stacked or a jam occurs. This state bit will also be reported during the initialization cycle (with the Power-Up bit set) if the bill is not in the Escrow position.

5. *Returning (Byte 0, Bit 5)* - In this state, the bill is being returned to the customer per order of the controller in response to an Escrow message. A distinction is made here between "rejected " a bill and "returning" a bill. A rejected bill is returned to the customer based on the validator's verification process. A returned bill is one which passed the validator's checks but the controller told the bill validator to return the bill to the customer.

6. *Bill Jammed (Byte 1, Bit 2 )* - In this state, it has been determined that the validator cannot successfully finish an operation such as stacking or returning the bill. The validator will try to resolve the condition and, if successful, will automatically exit this state and indicate what the final disposition was.

7. *Stacker Full (Byte 1, Bit 3)* - In this state, the validator had determined that it is not able to press the current bill into the stacker magazine. Typically, this is an indication that the cassette or magazine attached to the validator is full. It will continue to try at reduced power on the stacker motor and, if finally successful, will automatically exit this state.

8. Failure (Byte 2, Bit 2) - In this state, the validator has discovered some condition that does not permit it to continue accepting bills other than those conditions listed above. Typically, the bill validator has determined after some attempts at error recovery that it is no longer able to accept currency. For example, the validator may determine that one its optical sensors has failed.

9. Nopush (Byte 3, Bit 0) - A bill validator will operate in either a **Push** or a **Nopush** mode. There are times when the bill validator has a bill for which it is not going to issue credit, yet the bill validator can not return the bill to the customer. As an example, a bill may jam in the validator while in the process of trying to return the bill to the customer. After multiple attempts to return the bill fail, the validator will run the bill into the unit even though it is not going to issue credit. Some applications require that the bill validator just STACK the bill and remain in service. This is called PUSH mode because the bill is "pushed" into the stacker assembly. Other applications require that the validator "freeze" or go out of service without moving the bill into the cassette or magazine. When a customer complains that they have not received credit for a bill, this leaves the bill as evidence at an accessible point within the bill validator for an on site service tech to verify the customer complaint. This is NOPUSH mode. A bill validator which is "frozen" in this manner will report that it is in the NOPUSH state bit set. PUSH mode is used to keep the unit in service where the customer can be trusted or their complaints can be verified at a latter time. NOPUSH mode is used where customers' complaints need to be verified "on the spot". If the controller has enable the NOPUSH mode and a nocredit situation like this occurs, this bit will be set and the bill validator will freeze until power is reset or the mode is changed to PUSH by the controller.

The next list contains definitions of the events that occur within the bill validator and are only reported to the controller one time. These generally indicate the completion of some operation. It is possible for more than one of these bits to be set in the same message. These bits get cleared after a message is received from the controller with a new Msg/Ask# (to be described later) indicating the message containing these bits was received correctly by the controller.

1. Stacked (Byte 0, Bit 4) - This event is reported after the bill has been fully moved to a secure position within the bill validator. This is the point at which credit should be issued by the controller for an inserted bill. Even though the bill validator tells the controller the

denomination of the bill at the escrow position, the controller should only give the amount of credit as indicated in the stacked message. There are some instances where the credit amount told to the controller at escrow will be wiped out in the stacked message (e.g. a CHEAT attempt is detected).

2. Returned (Byte 0, Bit 6) - This event is reported after a bill has been successfully returned to the customer per the order from the controller following an Escrowed message. This bill had been validated by the validator but the validator was told by the controller to return the bill to the customer.

3. Cheated (Byte 1, Bit 0) - This event is reported if the validator detects some condition that causes it to believe the bill is being manipulated. This can result in the bill either being returned to the customer or, if the bill is too far into the validator to return, being stacked with no credit being given.

4. Bill Rejected (Byte 1, Bit 1) - This event is reported after the bill has been to the customer because it failed the validation tests.

5. Power-Up (Byte 2, Bit 0) - This event indicates that the validator has experienced a power reset condition. It will continue to be reported until the validator has finished its initialization operation. While this condition is being reported, the controller is not able to affect the operation of the validator. This event is guaranteed to be reported at least once even if the initialization is complete before the first request for status is received. This bit is sort of a hybrid between a state and an event in that it can be reported more than once but must be reported at least once. During the validator's power up process, the validator will encure that it has a clear bill path by running its validator and stacker motors ( unless a bill is detected in escrow).

6. Invalid Command (Byte 2, Bit 1) - This event is reported whenever a message containing a command code that is not recognized is received from the controller. This condition represents an error in the format of the message rather than in its contents.

This list represents those terms that do not fit into either of categories above.

1. Lockable Removable Cassette present (Byte 1, Bit 4) - This bit is controlled completely independently of the other bits in the system. If this unit is NOT fitted with a lockable removable cassette, this bit is always set (The Lockable Removable Cassette is a variant of the normal magazine in that when removed from the validator a key is required to gain access to the collected bills). If fitted with a Lockable Removable Cassette, this bit is set if the Lockable Removable Cassette is properly attached to the validator. When this bit is clear (indicating the LRC is not attached), the validator will not take any bills regardless of any commands from the controller.

2. Bill Value (Byte 2, Bits 3-5) - These three bits are used to encode the denomination of the bill in process within the bill validator. They are meaningful in combination with: (1) the Escrowed state bit to indicate what the bill is at the escrow position, (2) the Stacked event bit to indicate the credit amount, and (3) the Returned event bit to indicate the bill value that was returned to the customer. The value of zero indicates unknown denomination. There are multiple cases where the bill validator will issue a denomination field set to zero. Here are a few examples: (1) This is significant at power-up if a bill is in escrow. After a power reset the bill validator will not know the denomination of a bill in

escrow. It tells the controller this by setting the denomination field to zero in the ESCROWED message. (2) In some instances the bill validator senses that someone is tampering with the unit, possibly attempting to fraud the validator. A bill value that may have been set to some denomination escrow may be wiped out in a subsequent message, telling the controller not to give credit. (3) A Bookmark (described later) has a denomination value of zero. Once these bits are set to a nonzero denomination, they remain set until a transition into the Accepting or Failure state or until the report of a Cheated or Bill Rejected event.

3. Paused (Byte 1, Bit 5) - While this bit is set , the validator has detected something in its front bezel while it is preparing to stack a bill. In order to prevent what is in front bezel from being run into the validator, the validator will stop bill motion until what is in front bezel has been removed. In other words, a customer has quickly inserted a second bill while the first bill is still in motion within the validator. This is known as a “fast feed”. In applications where the controller can display messages to the customer, this PAUSE message can prompt a “please remove your bill from the validator” message. The state is exited automatically upon detection of a clear bezel area and processing of the original bill being stacked is resumed. This bit will always be set along with one of the “State” bits described earlier.

The following is a description of the “bookmark” feature implemented in certain CashCode Bill Validators. A bookmark is a piece of paper that is placed into the Validator’s bill stacker to mark the location of special event. For example, a customer may complain that the bill validator has taken some currency and has not given credit. In certain applications (especially those which use Lockable Removable Cassettes) it may be deemed not feasible to immediately gain access to the cassette or magazine to verify a customer complaint. The bookmark can be run into the validator at the time of the event and the customer’s complaint can be verified at some later time, when the cassette or magazine is normally removed from the system.

1. The processing of bookmarks must be explicitly enabled by the Controller to control their acceptance. This will also prevent unexpected status information from showing up in applications that do not make use of this features.

2. While the acceptance of bookmarks is enabled, processing of valid bills will continue as normal. This allows the Controller to decide whether to accept or return the bills while in this special mode (assuming escrow mode is enabled). Any regular bills for which the enable bit is set in Byte 0 of controller messages will be processed normally. While Bookmarks are enabled, the validator will take in bills even with no normal denominations enabled in Byte 0 of controller messages. This allows only bookmarks to be accepted if desired.

3. Bookmark acceptance will be enabled by changing the :Msg Type” value of Controller-to Validator messages from 1 to 3 (bits 4 - 6 of the third message byte). As long as the Msg Type is a 3, bookmarks will be accepted. The Controller must indicate when this mode is to be terminated by reverting to a Msg Type of 1. It is recommended that the Controller terminate the mode automatically upon receipt of the first bookmark. See section 1, MESSAGE FORMAT, in this specification.

4. The acceptance of a bookmark will be indicated to the Controller in the same manner as a normal bill except that the denomination field will be set to zero (none/unknown). If

escrow mode is enabled, the Controller must still indicate whether to accept or return the object.

5. A bookmark will consist of a piece of paper that is the same width as a normal bill but shorter in length. This bookmark should be inserted into the Validator with any printing on the bottom.

## 2. Format For Data Fields

In later sections we will define the line- level protocol and the message bytes involved with message transfer. the “meat” or real information of each message between the bill validator and the controller (used the application layer) is contained in the data fields. these data fields are described in this Section.

There is always a maximum of 7 bits of information in each byte. The line-level protocol uses the bit 7 as a parity bit.

### 2.1. Data Fields For Messages Sent By The Controller.

The messages sent by the Controller contain 14 bits of information:

Byte 0
Bit 0: $1 Accept Enable ( = 1 to enable the denomination for acceptance)
Bit 1: $2 Accept Enable
Bit 2: $5 Accept Enable
Bit 3: $10 Accept Enable
Bit 4: $20 Accept Enable
Bit 5: $50 Accept Enable
Bit 6: $100 Accept Enable
Byte 1
Bit 0: Interrupt Mode (= 1 enable the interrupt mode) (see note 1)
Bit 1: Security ( = 1 for high security) (see note 2)
Bit 2: Orientation ( = 1 for acceptance in both orientations)
Bit 3: Orientation ( = 0 reserved for future options)
Bit 4: Escrow Enable ( = 1 for Escrow enabled) (see note 3)
Bit 5: Stack ( = 1 causes bill to be stacked) (see note 4)
Bit 6: Return ( = 1 causes bill to be returned) (see note 4)

In the current Bill Validator, if none of the bill enable bits for $1 through $20 are set in Byte 0 and the bookmark feature is not enabled, the validator will not accept currency. The bill validator will not turn on its motor when a bill is inserted into the bezel. If some denomination are enabled and a non-enabled denomination is inserted, the bill validator will run the bill into validator, see that the bill’s value is not enabled then reject the bill. The bill validator only issues a REJECTED message if it could not recognize the bill.

Byte 2
Bit 0: NoPush Mode ( = 1 enables the NoPush mode)
Bit 1-6: (reserved for future use)

Notes:

(1) When this bit is set, the Validator will send an “ENQ” message whenever there is a state change (e.g. the Validator would send a message when it changed from the “Stacking” state to the “Stacked” state).

(2) The detailed functionality of the security bit within the bill validator will change as software revisions are upgraded. It is anticipated that some algorithms for fraud detection within the bill validator may be optionally enabled (bit =1) and disabled (bit=0) depending on the application. For example, in a highly-supervised environment, some anti-fraud algorithms may not be needed and could be disabled in an effort to obtain higher acceptance rates.

(3) Disabling the escrow function (Byte 1 Bit 4 = 0) causes all accepted bills to be stacked without waiting in the escrow position for a STACK command from the Controller.

(4) If the escrow enable bit was set, the stack and return bits are used to direct the bill from the escrow position. The “Stack” or “Return” bits should be set only when a bill is in Escrow. These two bits sent from the controller are ignored by the bill validator unless the bill validator is in the Escrow state. If a bill is in Escrow, a message with either bit set will cause the Validator to immediately begin the process of stacking or returning. Once a bill is stacked or returned, the Validator resets the bit internally (the Controller does not have to send an additional message with the bits reset). Therefore, extra “Stack” or “Return” commands from the Controller are ignored until the bill validator is again in the escrow state.

**2.2. Data Fields For Messages Sent By The Validator.**

The bytes in messages sent by Validator contain both bit fields and byte fields.

Byte 0
Bit 0: Idling ( = 1 If the Validator is in the idle state) (see note 1)
Bit 1: Accepting ( = 1 If accepting a bill) (see note 1)
Bit 2: Escrowed ( = 1 If a bill is in Escrow)
Bit 3: Stacking ( = 1 If a bill is being stacked) (see note 1)
Bit 4: Stacked ( = 1 If a bill was stacked) (see note 2)
Bit 5: Returning ( = 1 If a bill is being returned) (see note 1)
Bit 6: Returned ( = 1 If the bill has been returned) (see note 2)

Byte 1
Bit 0: Cheated ( = 1 If the Validator was cheated) (see note 2)
Bit 1: Bill rejected ( = 1 If a bill was rejected) (see note 2)
Bit 2: Bill jammed ( = 1 If a bill is jammed)
Bit 3: Stacker full ( = 1 If the stacker is full)
Bit 4: Lockable Removable Cassette Present ( = 1 If the Lockable Removable Cassette is present)
Bit 5: Paused ( = 1 If Validator in PAUSE mode)
Bit 6: = 0 (reserved for future use)

Byte 2
Bit 0: Power-up ( = 1 If the Validator is initializing) (see note 3)

Bit 1: Invalid Command ( = 1 If invalid command received)
Bit 2: Failure ( = 1 If the Validator is in a fault mode)
Bit 3-5: Bill Value (see note 4)
  000 = None / Unknown / Bookmark
  001 = $1
  010 = $2
  011 = $5
  100 = $10
  101 = $20
  110 = $50
  111 = $100
Bit 6: = 0 (reserved for future use)

Byte 3
Bit 0: NOPUSH mode ( = 1 If the Validator is stalled in NOPUSH)
Bit 1-6: ( = 0) Reserved for future use

Byte 4: Model # (00- 07FH)

Byte 5: Revision of Code (00, 01, 02....07FH)

Notes:

(1) During the “Interrupt Mode”, the Bill Validator will not generate an interrupt (“ENQ” sent) when there is a state transition into the following 4 states: Idling, Accepting, Stacking, Returning. An interrupt will be generated (‘ENQ” sent) for the following state transitions (when the corresponding bit changes from either a 0 to a 1 or from a 1 to a 0):

STATE STATE ENTERED STATE EXITED
(bit gets set) (bit gets cleared)

Escrowed ENQ Sent
Stacked ENQ Sent
Returned ENQ Sent
Cheated ENQ Sent
Rejected ENQ Sent
No Push ENQ Sent
Jammed ENQ Sent ENQ Sent
Stacker Full ENQ Sent ENQ Sent
LRC Pres. ENQ Sent ENQ Sent
Failure ENQ Sent ENQ Sent

(2) These bits indicate that a state has been entered or that there has been an occurrence: Bill Stacked, Bill Returned, Unit Cheated, Bill Rejected. Once the Bill Validator is assured that the Controller has received these bits correctly (the Controller sends a new message, rather than retransmitting its last message), the Validator resets the bits. As an example, for each bill that is rejected, the Controller will only receive one message with the “Bill Rejected” bit set.

(3) The Power-up bit indicates that the Validator has been reset or is in the process of being reset. The “Power-up” bit will be set (for multiple messages) as long as the Validator is in the process of initialization (e.g. clearing the bill path).

Note that this bit may be set along with other bits (e.g. the “Stacking” or “Idling” bits may be set at the same time that the “Power-up” bit is set). It is guaranteed that a reset will cause at least one message to be sent with the “Power-up” bit set. The “Power-up” bit must be clear before the Controller can control the operation of the Validator.

(4) The “Bill Value” field indicates the value of the last bill received. It is valid when one of the following are true:

| | |
|---|---|
| (5) | Validator is in IDLE |
| (6) | Bill is in Escrow |
| (7) | Bill has just been stacked |

The value is reset when the Validator begins accepting the next bill.
When the Validator powers up, it may have a bill in escrow. Under these conditions, the Validator will send a message indicating:

Bill in Escrow (byte. bit 0.2 = 1)
Validator Reset (byte. bit 2.0 = 1)
Bill Value is Unknown (byte. bits 2.3 - 2.5 = 000)

Once this message is received by the controller, the controller can decide to stack or return the bill. It is anticipated that although the Bill Validator does not know the value of the bill (due to being reset), the Controller may know the value as the result of previous communications. It is recommended that normally the bill be returned.

## 3. Detailed Interface Description

### 3.1. Message Format

The format for each transmitted is as follows:

/ STX / Length / Msg Type and Ack # / Data Field /...../ / ETX / Checksum /

STX = 02H
Indicates start of message

Length = The number of bytes in the message (in binary), including the STX, ETX, and the Checksum. For ease of future enhancements, this should be treated as a variable and therefore messages treated as variable length messages.

Msg Type and Ack #

Msg Type (bit 4, 5, & 6 of 3rd byte of messages)
= 1 for standard Controller to Validator messages
= 2 for standard Validator to Controller messages
= 3 Controller to Validator Bookmark enabled
= 4 - 07H for future optional messages

Ack # = 0 or 1 (lower 4 bits of 3rd byte)

In messages sent by the Controller, the number is used to identify the message. As messages are sent to the bill validator, the number alternates between 00 and 01. If the Validator receives two consecutive messages with the same number, the second message is treated as the retransmission of the first.

In messages sent by the Bill Validator to the Controller, the number is used to acknowledge specific messages sent by the Controller. When the Bill Validator receives a Controller's message correctly, the "Ack #" of the Validator's response is set to the "Msg #" of the Controller's message. If the Bill Validator receives a message incorrectly, it will respond with the previous "Ack #".

Data = The data portion of the message consists of the multiple data fields as described in Section 2; "application layer" information.

ETX = 03H

Checksum = The checksum is calculated on all bytes between the STX and the ETX ( excluding the STX and the ETX). The calculation is performed by xor'ing the bytes.

## 3.1. Message Transfer

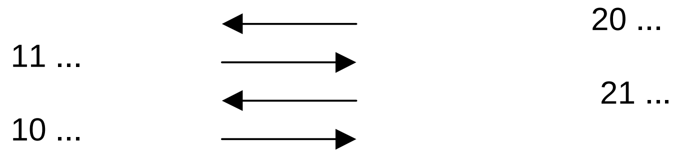
The Bi-directional Serial Interface can be operated in one of two basic modes: standard polling mode or special interrupt mode. In standard polling mode, the controller typically maintains constant communication with the validator at some constant polling rate. For example, the Controller may poll the Validator once every 100 milliseconds. In special interrupt mode, there is much less communication between the Validator and the Controller while the Validator is not busy. In special interrupt mode, most communication occurs while something "important" is occuring at the Validator (e.g. a bill is being fed into the unit).

The Controller is the Master. The Bill Validator is always responding to messages that are initiated by the Controller. In special interrupt mode, the Bill Validator sends an ENQ to the Controller To get the Controller to send a message. For each message sent by the Controller, the Bill Validator responds with a message ( the Validator's message contains both data information and a protocol link response).

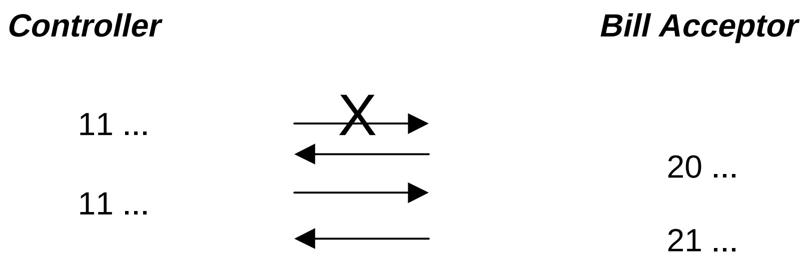
The Message number alternates for each message (retransmissions do not alter the number). When the bill validator receives a message successfully , it responds with a message containing an "Ack #" equal to the "Msg #" of the received message. The byte representing the message type and ack number during normal transaction appear as follows:

***Controller*** ***Bill Acceptor***

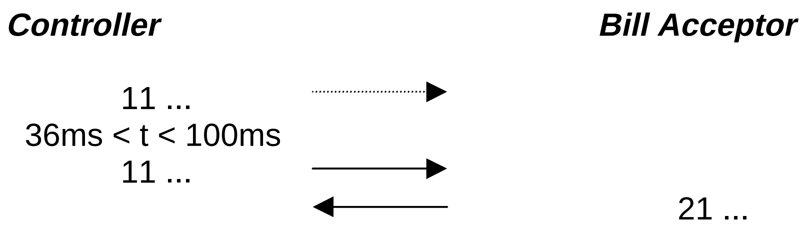
10 ... $\longrightarrow$



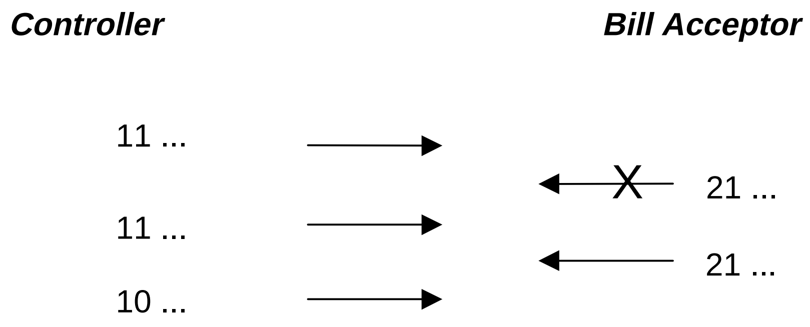
If the Bill Acceptor receives a message unsuccessfully (the correct number of bytes is received but an error is detected through parity or the CRC), the Bill Acceptor will respond with the previous “Ack #”. This should prompt the Controller to retransmit its last message with its “Ack #” unmodified.



If the Bill Acceptor received a partial message, it does not respond.
After the Controller times out (time > “Max Line Turn Around Time”), the message is retransmitted.



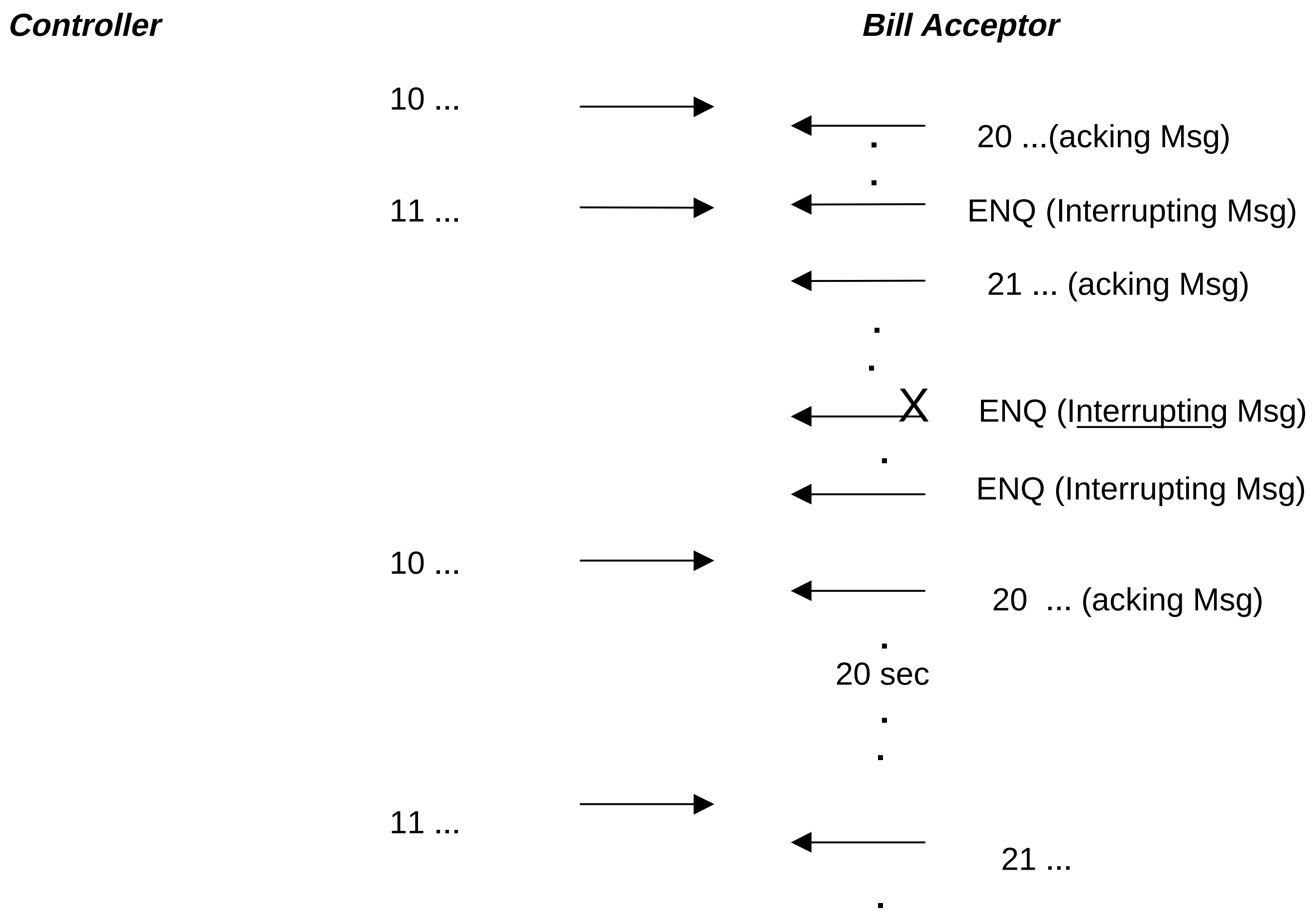
If the Bill Acceptor’s response is lost or damaged, the Controller will retransmit the previous message.



## 3.2. Special Interrupt Mode

By setting a bit, the Controller can initiate a special interrupt mode. In this mode the Validator will send an “ENQ” (05H) message whenever it has a state change (e.g. going from “Stacking” to “Stacked” is considered a state change). When the Controller receives the “ENQ”,it begins the normal polling sequence outlined in section 2.1. If the Validator does not receive a poll within 100ms after sending an “ENQ”, it will send another “ENQ” (multiple “ENQ’s” should not cause problems within the Controller).

When using the interrupt mode, the Controller should still send unrequested polling messages every 10 - 60 seconds. This should be done as a precautionary measure to recover from Validator power fall resets, which will put the Validator into the polling mode (the Validator will never send an “ENQ’



## 3.3. Timing and Character Format

Inter Character Timing: $0 < t < 20$ms

Line Turn Around Time:

The Bill Validator will respond within 20ms after receiving the last byte of a message or not respond at all. If the Controller has not starting receiving a response within 35ms, it assumes that a response is not coming and will retransmit its initial polling message.

The Bill Validator will assume that its message has been received successfully if:

- It receives an acknowledging message from the Controller with a new Msg/Ack #

- 100ms has expired without a further polling message from the Controller

Consequently: If the Controller does not receive a response from its polling message (or the response is bad), it should retransmit its message within 100ms.

Inactive Timing: 3 seconds/ 30 seconds

If a device has been unsuccessfully receiving messages or responses for 3 seconds, it assumes that the other device has been reset or permanently failed. Initially the device will be reset the protocol parameters (e.g. Msg / Ack #) and try to reestablish communications. If communications cannot be reestablished, the device will assume that the distant device has failed permanently or lost power (reestablishment may be tried periodically).

Note that the 3 second time out also applies to the Bill Validator's transmission of ENQ's. If the Validator does not receive a polling messages after 3 seconds of sending ENQ's, It will assume that the communications link is lost.

The Validator will inhibit the acceptance of bills when it is powered up or when the communications link is lost:

3 seconds of unsuccessful communications
30 seconds without a poll from the Controller (assumes the non-interrupt mode is active)

If a bill is in escrow when the communications link is lost, it will be kept in escrow. Consequently, every time the link is reestablished the Controller should first verify that a bill is in escrow before sending a "Stack" or "Return" command (if a bill is in escrow, byte.bit 0.2 of the Validator's messages will = 1; see 3.2.). The Validator will stay in the disabled state until the link is established and a message enabling the acceptance of bills (see 3.1.) is received.
Note that it is not uncommon for the Bill Validator to miss messages (during bill processing, the communication channel is frequently turned off for approximately 1 second).

Character format:
1 start & 1 stop bit
7 data bits (bit 0 = LSB, bit 0 sent first)
1 partly bit (bit 7) (even partly)
NRZ format

Baud rate:
Standard: 9600
Optional: 1200 *

Duplex: half duplex

* The Validator will be capable of determining the incoming baud rate (1200 or 9600). at power-up, the Validator will try to determine the incoming baud rate. This will typically be done by processing the incoming "STX" character.

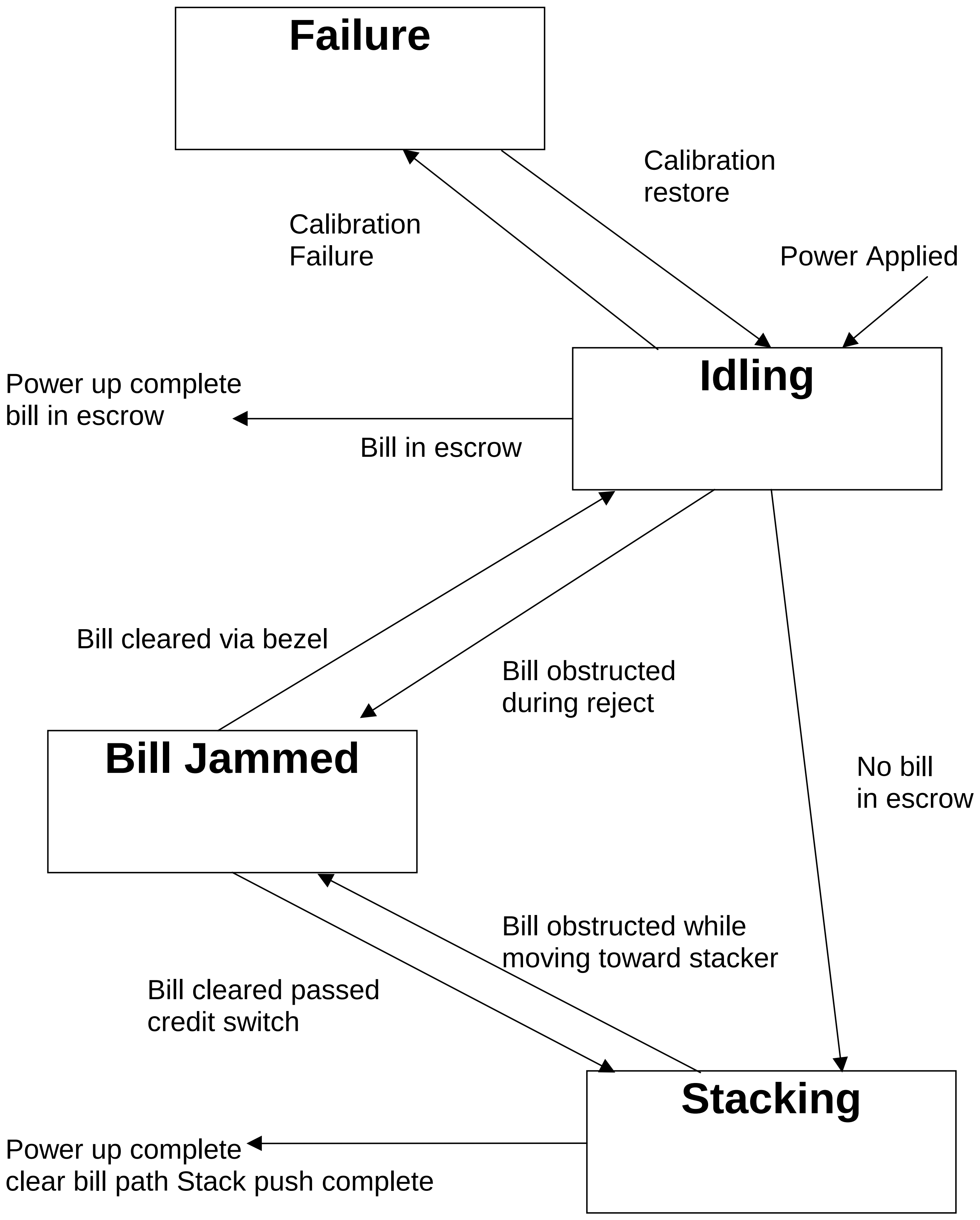
## 7.9. State Diagrams.

The following pages illustrate the typical state transitions of the Bill Validator as reported to the Controller.

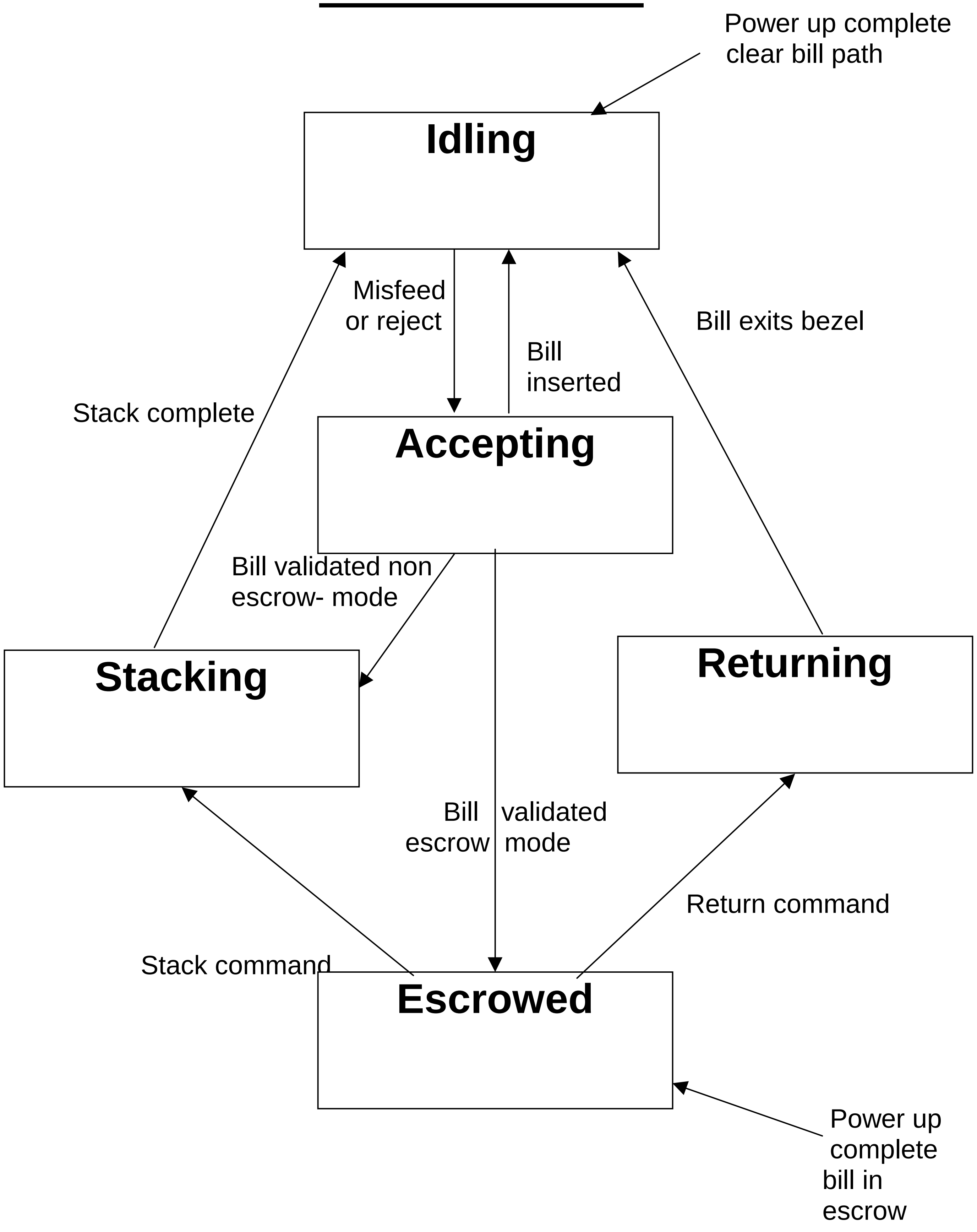
Each state on these diagrams represent an interface bit described in this document.

- POWER UP STATES
- NORMAL BILL STATES
- EXCEPTION STATES
- NO PUSH MODE STATES

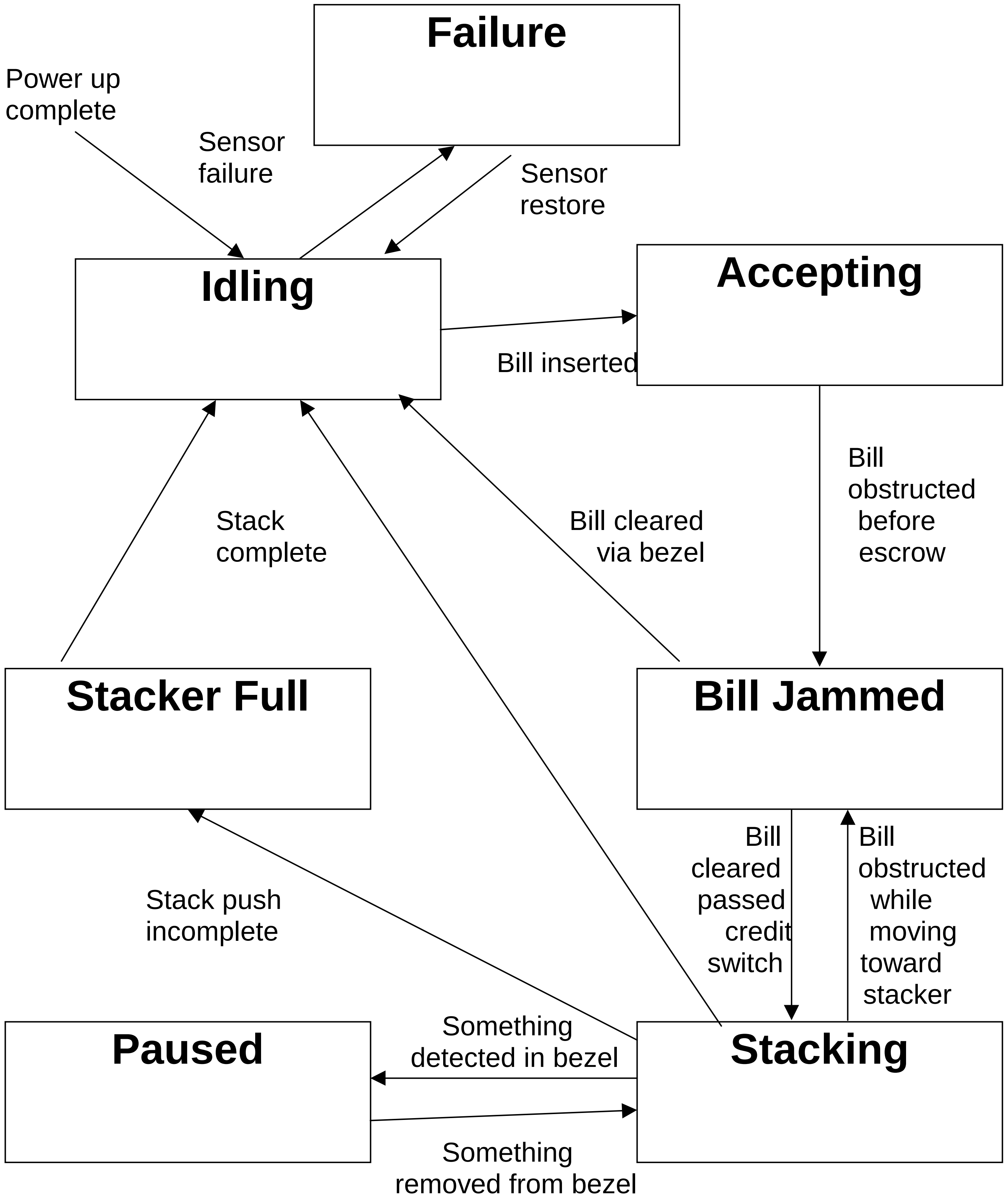
## Power Up States

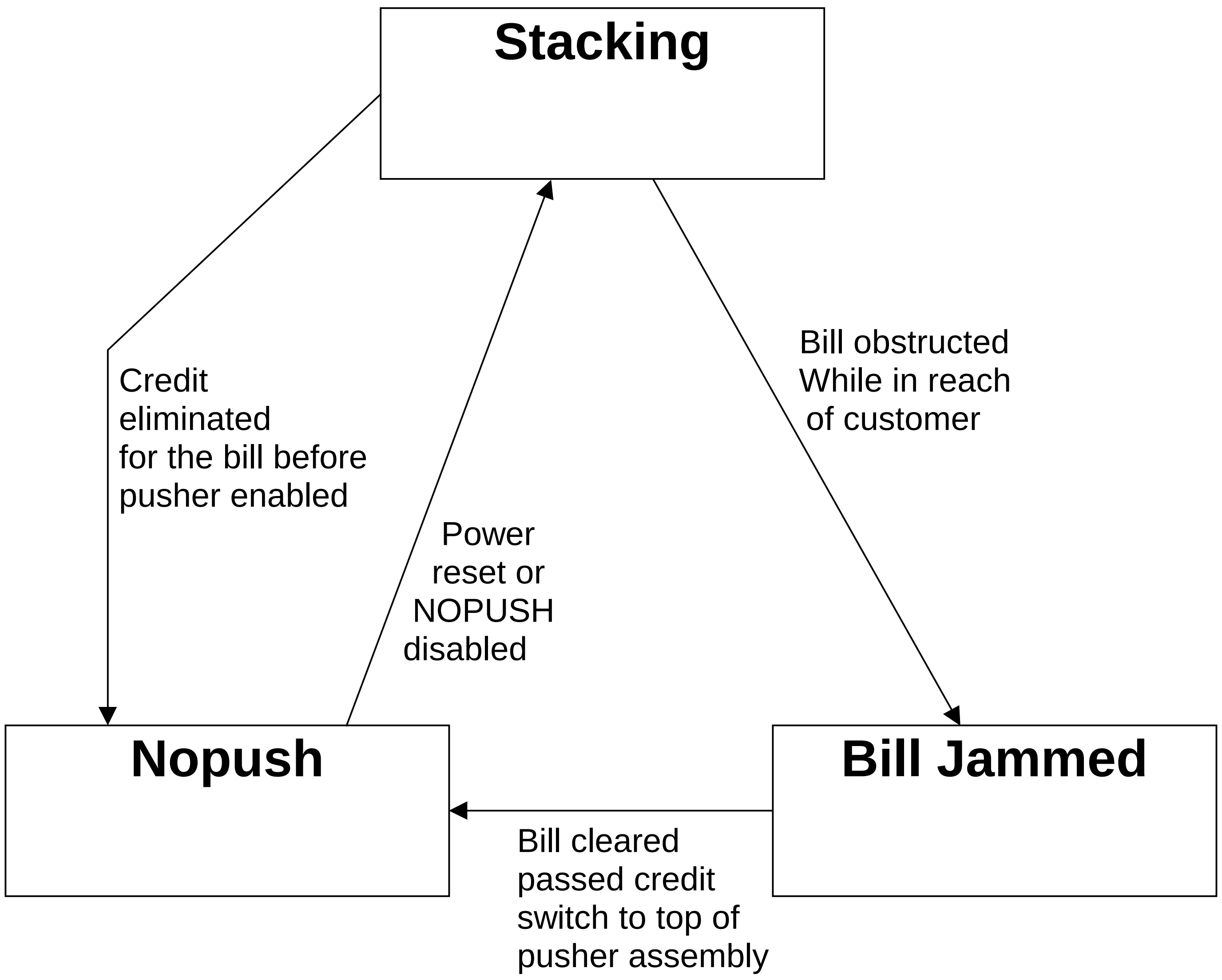

Failure
Calibration restore
Calibration Failure
Power Applied
Idling
Power up complete bill in escrow
Bill in escrow
Bill cleared via bezel
Bill obstructed during reject
Bill Jammed
No bill in escrow
Bill obstructed while moving toward stacker
Bill cleared passed credit switch
Stacking
Power up complete clear bill path Stack push complete

# Normal Bill States



## Exception States




Stacking
Credit eliminated for the bill before pusher enabled
Power reset or NOPUSH disabled
Bill obstructed While in reach of customer
Nopush
Bill Jammed
Bill cleared passed credit switch to top of pusher assembly