PROPRIETARY

February 24, 1999

International Coin Validator
Serial Link Interface Specification

ECN #: 102-13087
IF5U05BE
Stream: 00, Issue: 01

Author(s): Brian Pardoe
Date: February 24, 1999

Approved by: Ron Field
Date: February 24, 1999

Keywords: Millennium, Interface, Validator, International

Abstract: This document describes the serial link protocol for the Millennium international coin validator, firmware version 3.

Security classification downgrading

Refer to BNR Corporate Procedure 5030.01 appendix A paragraph 5.10

Reclassify On:
to:

Declassify Never

Funder:

Prime: ENC

Other Reference: Case:

Change Control Record

Issue 1.0: NT issue of International Coin Validator Serial Link Protocol Specification – IF5U05BE.

Symbols, Abbreviations and Acronyms

ASIC	Application Specific Integrated Circuit
ASCII	American Standard Code for Information Interchange
BPS	Bits Per Second
CMOS	Complimentary Metal Oxide Semiconductor
EEPROM	Electronically Erasable Programmable ROM
RAM	Random Access Memory
ROM	Read Only Memory
SRAM	Static RAM
TTL	Transistor-Transistor Logic

References

1. Terminal / Validator Interface Specification
   IF5U01AJ, Issue 1
   R.G. Field, Dept. 2800
   May 1, 1991

15539136. International Coin Validator Best-Fit Firmware
   NT5U09QR, Version 3.0
   B. Pardoe, Dept. M008
   February 24, 1999

Introduction

The international coin validator uses a full-duplex, asynchronous serial link for communication. All control and status information is transported via this communication path. A separate discrete control line is used for hardware reset. This specification if aplicable to version 3 of the coin validator firmware.

Serial Link Characteristics

The serial link has the following characteristics:

Separate lines are used for transmitting and receiving data. For the purposes of this document, a control message is data that is received by the validator, and a status message is data that is transmitted by the validator.

Both transmit and receive data are framed using one start bit, eight data bits, and one stop bit.

The data bits are transmitted and received least significant bit first.

The data rate is either 600 BPS or 2400 BPS, selectable via an interface command. The data rate will default to 600 BPS following a reset or restart.

The electrical interface is the Motorola MC68HC705B16 microcontroller. The transmit output level is TTL/CMOS compatible. The receive input level is CMOS compatible (0.2 / 0.7 Vdd).

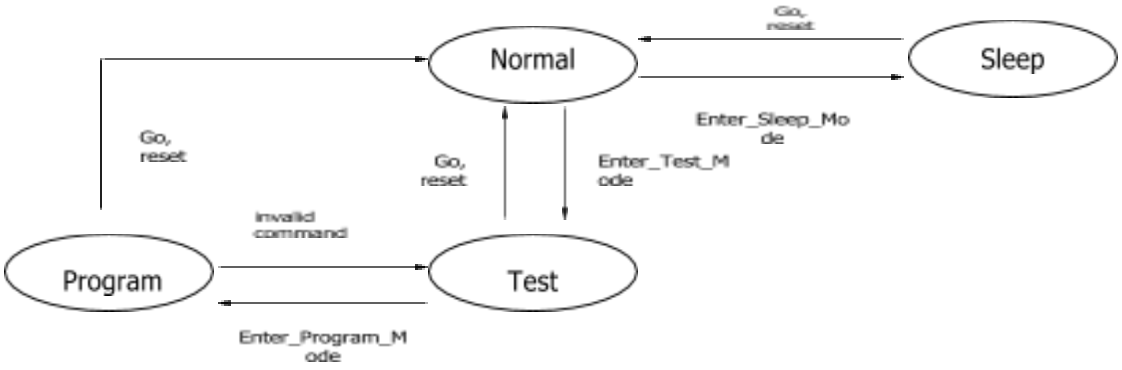
All messages start with an ASCII-format data byte, except for the query memory reply (this is binary data, and is retained for backward compatibility.

Some messages also have associated data bytes, which are either binary-format or ASCII-format (depending on message type and validator mode).

Modes Of Operation

The international validator has four modes of operation: normal, test, program, and sleep. Figure 1 gives a state transition diagram for these modes.

Figure 1: Operating Mode State Diagram



Normal Mode

The default operating mode for the validator is normal mode. This operating mode is entered after a hardware reset or a “Go” control message is received. All messages used in normal mode are backward compatible with the set of commands for the North American coin validator (IF5U01AJ).

In normal mode, only normal mode control messages are accepted; all other messages result in a “Communication_Error” status message response.

Test Mode

The primary use of test mode is calibration of the validator. This operating mode is entered after an “Enter_Test_Mode” control message is received.

In test mode, both normal mode and test mode control messages are accepted; all other messages result in a “Communication_Error” status message response.

When in test mode, all message content is in ASCII; this allows a terminal to be used for interrogation or debugging of the validator. As a result, the messages from normal mode that use binary data have the data converted to hexadecimal-ASCII for test mode operation (the binary-format data was retained for backward compatibility).

Program Mode

Program mode allows data to be written into EEPROM in the validator. This operating mode can be entered only from test mode, after an “Enable_Programming” control message is received.

In program mode, all valid control messages are accepted. If an invalid or unknown command message is received, the program mode is terminated, a “Communication_Error” status message response is sent, and the validator reverts to test mode.

When in program mode, all message content is in ASCII.

Sleep Mode

When in this mode, the validator consumes minimum power; only the serial link hardware remains active. This operating mode is entered after an “Enter_Sleep_Mode” control message is received.

All messages that are received when the validator is in sleep mode will be ignored, except for the “Go” command.

Messages

When a coin identifier component of a message is required, it must be one of the following twelve lowercase ASCII characters:

- “0” through “9”
- “a” through “c”.

Control Messages

All messages start with a lowercase, ASCII-format byte. Control messages are messages received by the validator. All data is ASCII encoded unless otherwise specified.

Normal Mode Control Messages

The control messages that the validator can receive when the validator is operating in normal mode are specified in Table 4.1/1.

Test Mode Control Messages

The additional control messages that the validator can receive when the validator is operating in test mode are specified in Table 4.1/2. The format of the “Query_Memory” message is changed from the format that is used in normal mode (the data portion of the message is ASCII instead of binary); this message is repeated in table 4.1/2 for clarity.

Program Mode Control Messages

The additional control messages available when the validator is operating in test mode are specified in Table 4.1/3.

Table 4.1/1 Normal Mode Control Messages Received by Validator

Message	Description
a	Accept_Coins. Accept all coins that have a signature matching an enabled coin. When in test mode this is interpreted as unconditionally accept all coins.
c	Cash_Escrow. Actuate escrow to empty bucket into the cash box.
d*n*	Disable_Coin. Disable the acceptance of coin *n*; where *n* is a valid coin identifier.
e*n*	Enable_Coin. Enable the acceptance of coin *n*; where *n* is a valid coin identifier.
f	Refund_Escrow. Actuate escrow to empty bucket into the refund chute.

g	Go. Wake up, reset, perform built-in-test, Null_Escrow, and give status response. Also forces validator to normal mode of operation, sets serial link rate to 600 BPS, and disables EEPROM programming.
hsc	High_Speed_Communication. Set serial link rate to 2400 BPS.
k	Calculate_Checksum. Calculate_Checksum of validator EEPROM, then update checksum value stored in EEPROM.
l	Learn_Escrow. Calibrate the escrow sensors by measuring the nominal times sensors C, F, G. This takes approximately two seconds, during which no other commands should be issued.
n	Null_Escrow. Center the escrow bucket. Normally only used to recover from a jam during a cash or refund operation.
q*aa*	Query_Memory. Read and return contents of a single byte in the validator's memory space. "*aa*" is the binary address (two bytes) of the location being queried.
r	Request_Hardware_Status. Request for status of validator hardware.
s	Enter_Sleep_Mode. Causes validator to Enter_Sleep_Mode of operation.
tst	Enter_Test_Mode. Causes validator to Enter_Test_Mode of operation.
v	Request_Validator_Version. Request for version validator firmware and options.
x	Exercise_Solenoid. Energize the accept solenoid briefly to allow it to latch the reject mechanism in the open (accept coin) position.
z	Reject_Coins. Reject all coins until reset or commanded otherwise.

Table 4.1/2 Test Mode Control Messages Received by Validator

Message	Description
j	Output_Calibration_Factors. Request for stored set of calibration factors.
m	Minimal_Diagnostics. Suppresses output of data measured for a coin; output best fit metrics.
o*n*	Output_Coin_Signature_Data. Request for stored set of encoded (nominal/window) coin parameters for coin "*n*"; where *n* is a valid coin identifier.
pro	Enable_Programming. Causes validator to enter program mode of operation; enables writing of data into EEPROM.
q*aaaa*	Query_Memory. Read and return contents of a single byte in the validator's memory space. "*aaaa*" is the hexadecimal address (four bytes) of the location being queried.

Table 4.1/3 Program Mode Control Messages Received by Validator

Message	Description
b	Apply_Calibration_Factors. Performs final step of calibration by applying the calibration factors to the reference signature data for all coins.
i*ndddd…*	Input_Coin_Signature_Data. Write set of encoded (nominal/window) coin parameters "*dddd…*" for coin "*n*" into the coin table segment of EEPROM; where *n* is a valid coin identifier.
u*dddd…*	Update_Constants. Write constants "*dddd…*" to the constants segment (36 bytes of data) of EEPROM.
w*aadd*	Write_Byte. Write a single byte of data "*dd*" to address "*aa*" in the validator EEPROM (EEPROM memory is 256 bytes long).
y*dddd…*	Input_Calibration_Factors. Write calibration data "*dddd…*" to the calibration factors segment of EEPROM.

Status Messages

Status messages are messages transmitted by the validator. All data is ASCII encoded unless otherwise specified.

Normal Mode Status Messages

The status messages transmitted when the validator is operating in normal mode are specified in Table 4.2/1.

Test Mode Status Messages

The additional status messages transmitted when the validator is operating in test mode are specified in Table 4.2/2. The format of the "Hardware_Status" message is changed from the format that is used in normal mode (the data portion of the message is ASCII instead of binary); this message is repeated in table 4.2/2 for clarity.

Message Usage

The validator communication protocol has three classes of messages.

Control Messages With No Status Response

The following control messages have no associated response:

- Accept_Coins
- Cash_Escrow
- Refund_Escrow
- Null_Escrow
- Minimal_Diagnostics
- Enter_Sleep_Mode
- Exercise_Solenoid
- Reject_Coins.

When operating in normal mode, the following control messages also have no associated response:

- Disable_Coin
- Enable_Coin
- Calculate_Checksum
- Learn_Escrow.

Table 4.2/1 Normal Mode Status Messages Transmitted by Validator

Message	Description
A	Status_Okay. Hardware built-in test completed successfully.

B*hs*	Hardware_Status. "B" is the message annunciator, "*h*" is the bit-mapped status of the hardware (binary), and "*s*" is the bit-mapped status of the optical sensors (binary). Bit definitions for Hardware_Status are as follows: bit 0 = ASIC or oscillator fault. Cleared only by reset. bit 1 = Timeout on escrow movement. Cleared by successful escrow command. bit 2 = EEPROM checksum failure. Cleared only by reset. bit 3 = RAM failure. Cleared only by reset. bit 4 = Communication failure (invalid command). Cleared after "B*hhss*" sent. bit 5 = EEPROM write failure. Cleared upon successful write. bit 6 = Validator jam or sensor failure. Cleared when jam disappears. bit 7 = Coin entry fraud attempt. Cleared when coin passes through system. Bit definitions for sensor status are as follows: bit 0 = Sensor A blocked. Indicates coin entry jam. bit 1 = Sensor B blocked. Indicates coin exit jam (solenoid/activator problem). bit 2 = Sensor C blocked. Indicates escrow full. bit 3 = Sensor D state. Part of escrow bucket position indication. bit 4 = Sensor E state. Part of escrow bucket position indication. bit 5 = Sensor F blocked. Indicates refund chute blockage. bit 6 = Sensor G blocked. Indicates cash box full. bit 7 = not used.
C	ASIC_or_Oscillator_Fault. At least one of the oscillators is out of tolerance.
D	Escrow_Fault. Sent if any of the following sensors become blocked: C (escrow entry), D (bucket position), or E (bucket position).
E	EEPROM_Checksum_Failure. Sent if EEPROM fails startup built-in test.
F	RAM_Failure. Sent if RAM fails startup built-in test.
G	Communication_Error. Sent if unknown/invalid command received.
H	EEPROM_Write_Failure. Sent if a write to EEPROM fails.
I	Validator_Jam. Sent if any of the following sensors become blocked: B (coin exit), F (refund chute), or G (cash box).
J	Coin_Entry_Fraud. Sent if sensor A (coin entry) becomes blocked.
V*vvrroo…*	Version_Data. "V" is the message annunciator, "*vv*" is the firmware version byte, "*rr*" is the firmware revision byte, "*oo…*" is the contents of the option bytes (four bytes).
n	Coin_Identification (where "*n*" is a valid coin identifier; 0 = coin rejected).
d	Binary data. Response to Query_Memory control message.

Table 4.22 Test Mode Status Messages Transmitted by Validator

Message	Description
B*hhss*	Hardware_Status. "B" is the message annunciator, "*hh*" is the bit-mapped Hardware_Status (hexadecimal), and "*ss*" is the bit-mapped sensor status (hexadecimal). Bit definitions are the same as for normal mode.
L	Coin_Lost. A coin was identified as being valid, but was not detected entering either the escrow.
M ss qq	Minimal_Coin_Data. "M" is the message annunciator, "*ss*" is the best score for the coin, "*qq*" is the quartile mask for the coin.
N*dddd…*	Measured_Coin_Data. "N" is the message annunciator, "*dddd…*" is the set of coin parameters.
P*dd*	Byte_Programmed. "P" is the message annunciator, "*dd"* is the data that was written.
Q*dd*	Query_Data. "Q" is the message annunciator, "*dd"* is the data that was read.
R	No_Coin_Match. Coin rejected due to not matching any known coins.

S*ndddd…*	Stored_Coin_Signature. “S” is the message annunciator, “*n*” is the coin identifier, “*dddd…*” is the set of encoded (nominal/window) coin parameters.
T*dddd…*	Oscillator_No-coin_Data. “T” is the message annunciator, “*dddd…*” is the set of oscillator calibration readings.
Y*dddd…*	Calibration_Data. “Y” is the message annunciator, “*dddd…*” is the set of calibration factors.
Z	ASIC_Timeout. ASIC did not complete readings for a coin.

Unsolicited Status Responses

The following status messages are transmitted in response to events detected by the validator:

- Escrow_Fault
- Communication_Error
- Validator_Jam
- Coin_Entry_Fraud
- Enter_Sleep_Mode
- Exercise_Solenoid
- Coin_Identification.

When operating in test or program mode, the following status messages may also be transmitted in response to events detected by the validator:

- Coin_Lost
- Measured_Coin_Data
- No_Coin_Match
- ASIC_Timeout

The “Measured_Coin_Data” message is followed by either the “Coin_Identification” or “No_Coin_Match” message, and then either the “Coin_Identification” or “Coin_Lost” message. If “Minimal_Diagnostics” has been sent, the “Measured_Coin_Data” message is suppressed (this prevents data loss during testing of rapid coin insertion due to the data link being too slow).

Paired Control/Status Message Sequences

When operating in normal mode, the paired message sequences given in Table 4.3/1 apply. When operating in test or program mode, the paired message sequences given in Table 4.3/2 apply. When operating in program mode, the paired message sequences given in Table 4.3/3 apply.

Table 4.3/1 Normal Mode Message Sequences

Control message received	Status message transmitted
Go	Either Status_Okay or One or more of the following: ASIC_or_Oscillator_Fault, EEPROM_Checksum_Failure, RAM_Failure
Query_Memory	Single byte of binary data.
Request_Hardware_Status	Hardware_Status
Enter_Test_Mode	Oscillator_No-coin_Data
Request_Validator_Version	Version_Data

Table 4.3/2 Test & Program Mode Message Sequences

Control message received	Status message transmitted
Disable_Coin	Byte_Programmed or EEPROM_Write_Failure
Enable_Coin	Byte_Programmed or EEPROM_Write_Failure
Output_Coin_Signature_Data	Stored_Coin_Signature
Calculate_Checksum	Byte_Programmed message for each byte of checksum (2), or EEPROM_Write_Failure.
Learn_Escrow	Byte_Programmed message for each byte of escrow data (12), or EEPROM_Write_Failure.

Table 4.3/3 Program Mode Message Sequences

Control message received	Status message transmitted
Apply_Calibration_Factors	Byte_Programmed message for each byte of data in coin table, or EEPROM_Write_Failure.
Input_Coin_Signature_Data	Byte_Programmed message for each byte of coin signature data, or EEPROM_Write_Failure.
Update_Constants	Byte_Programmed message for each byte of constants data, or EEPROM_Write_Failure.
Write_Byte	Byte_Programmed or EEPROM_Write_Failure
Input_Calibration_Factors	Byte_Programmed message for each byte of calibration factor data, or EEPROM_Write_Failure.