Orbit CMM to Host Interface Specification

Money Controls

Contents

Tables

Figures

1. Diary of changes

Issue 1.0 ..... February 2002

Issue 1.1 ..... March 2002
- cs (checksum) reference changed to cc

Issue 1.2 ..... 6th Sept 2002
- Modification to disclaimer.

Issue 2.0 ..... 22nd Nov 2002
- *Appendix 4: Additional commands for units fitted with the C120S Coin Acceptor* added.
- Battery supply details added to

Issue 2.1 ..... 6th Jan 2004
- All references to 'Scorpion' replaced with 'C120S'.
- Maximum C120S coin data file size reduced from 5k bytes to 2417 bytes.

Issue 2.2 ..... 22nd Jan 2004
- Amendments made to section 9.1.

Issue 2.3 ..... 9th Jun 2004
- Amendments made to section 9.4
- Sections 9.5, 9.6, 9.7 & 9.8 added to the manual.

Issue 2.31 ..... 30th Jun 2004
- Interim release. Further details to be added soon.
- cs (checksum) reference changed to cc in section 9.5 & 9.6.
- Footers changed

Issue 2.4 ..... 30th Jun 2004
- Changed footers

Issue 2.5 ..... 24th Aug 2004
- Sections 10, 11 & 12 added to the manual.
- Amendments made to sections 8.7, 9.2, 9.3, 9.4 & 9.8

2. Message Format

This document defines the software requirements to implement the protocol between the Coin Management Module and the controlling application, e.g. Host Main Control Board. It describes the procedure for information exchange, the frame characteristics, and the fields within the message packets. The CMM version with C120P and C120S (Appendix 4) coin acceptors supplied by Money Controls is described in this document.

This document also describes the initialisation process for the CMM.

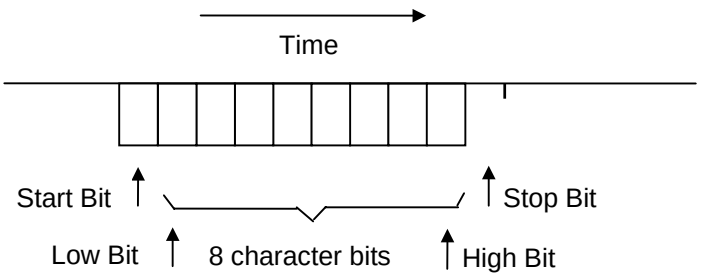
3. Message and Frame Characteristics

Transmission of data between nodes uses an asynchronous serial bus. Each message frame consists of several characters.

3.1 Character Format

The character format is shown below and consists of 1 start bit, 8 character bits (sent lowest bit first), and 1 stop bit. No parity bits are required.

Figure 1: Character Format

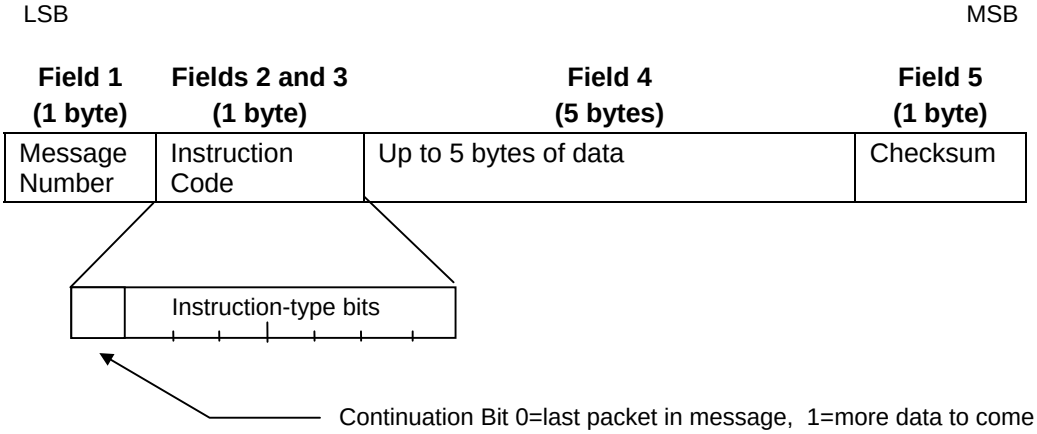


Characters are transmitted at a Baud Rate of 4800.

3.2 Message Frame Fields

Each frame consists of 5 fields contained within 8 bytes and is shown in the diagram below.

Figure 2: Message Frame Fields



The first byte is used to identify the message packet, so that in the event of a message being re-transmitted, it is not interpreted as a second occurrence of the initial message. Each time a message transfer is completed, this message number is incremented, and after 255 messages it rolls around to message 0.

The second byte is used to describe the instruction code for this message packet. The 2 fields in the instruction code are 1 (continuation bit) and 7 (instruction type bits).

The continuation bit is set if the data contained in the message packet is incomplete, and more data is to come in further messages. In the last message of a string of continued messages, the continuation bit will be reset to zero.

The instruction type bits define which instruction is being sent.

The third to seventh bytes contain the fourth field, which is up to 5 bytes of data relevant to the instruction. If more than 5 bytes are required then the continuation bit in the packet is set. Any unused bytes must be padded out.

The eighth byte contains the message packet checksum. This is evaluated by adding the values of all other bytes together, taking the two's compliment of this total and adding 1. Hence the sum of all of the bytes in the message (including the checksum) will be zero, ignoring any carry that occurs.
For transmission of a message to the CMM, the ‘CMM_Busy’ line from the CMM must be checked to ensure that the CMM is in a state where it can receive a message.

4. Message Handling Procedures

4.1 Pre-transmission Requirements

If the 'CMM_Busy' line is inactive, the following procedure should be followed:

- Raise the wake-up line to the CMM
- wait a minimum of 5ms
- send the message
- lower the wake-up line

This provides a frame for the message.

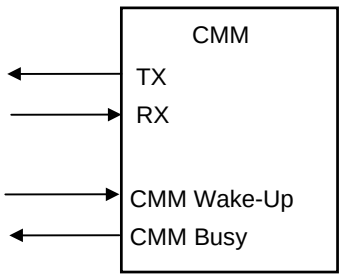
If the 'CMM_Busy' line is active, then the CMM is busy, and a message cannot be sent to the CMM at this time.

Note: The 'CMM_Busy' line is active HIGH. For transmission of messages from the CMM, the host is assumed to always be in a state to receive messages from the CMM and therefore no busy checking is required. The CMM will raise the 'CMM_Busy' and wait 20 ms before sending, if a message arrives from the host equipment in this time the CMM will back off for 50ms and handle the incoming message.

If the host equipment wishes to prevent messages being sent by the CMM, it can put the CMM in a mode where outgoing messages are inhibited. (See Appendix 1, section a) Operating Mode)

The diagram below shows the connections between the CMM and the host equipment.

Figure 3: CMM to Host Connections



4.2 Message Protocol

When a message is to be sent, the transmitting node will build the message frame, pass the message into a transmission buffer, and start a 'no reply' timer.
On receipt of a message packet the receiving node will check for framing errors. If any errors are found a 2-byte ‘NACK’ message is immediately returned to the transmitting node, with the reason code (listed below) set to indicate a hardware transmission error.
Providing there is no framing error, then the checksum is evaluated, and if incorrect, will cause a ‘NACK’ message to be sent with the reason code set to communication error. If no such errors are found, then the instruction code is checked against a list of valid instructions relevant to that node.
An invalid instruction will prompt the receiving node to return a ‘NACK’ message to the originating node, with the reason code set to unknown instruction, and an echo of that instruction. The transmitting node can then verify that the instruction sent was correct. A valid instruction will result in an 'acknowledge' message being sent.
The diagram below shows the configuration of the 2 byte return message to the originating node.

	1st Byte	2nd Byte
Acknowledge	1 1 0 0 X X X X	X X X X X X X X
	X = don’t care	
Not Acknowledge	0 0 1 1 r r r r	i i i i i i i i
(“NACK”)	Reason Code	Instruction Echo

Table 1: Transmission Error Codes

Reason Code	1st byte	Reason for Failure
00	30	Unknown Message
01	31	Comms Error (checksum fail)
02	32	Hardware Error

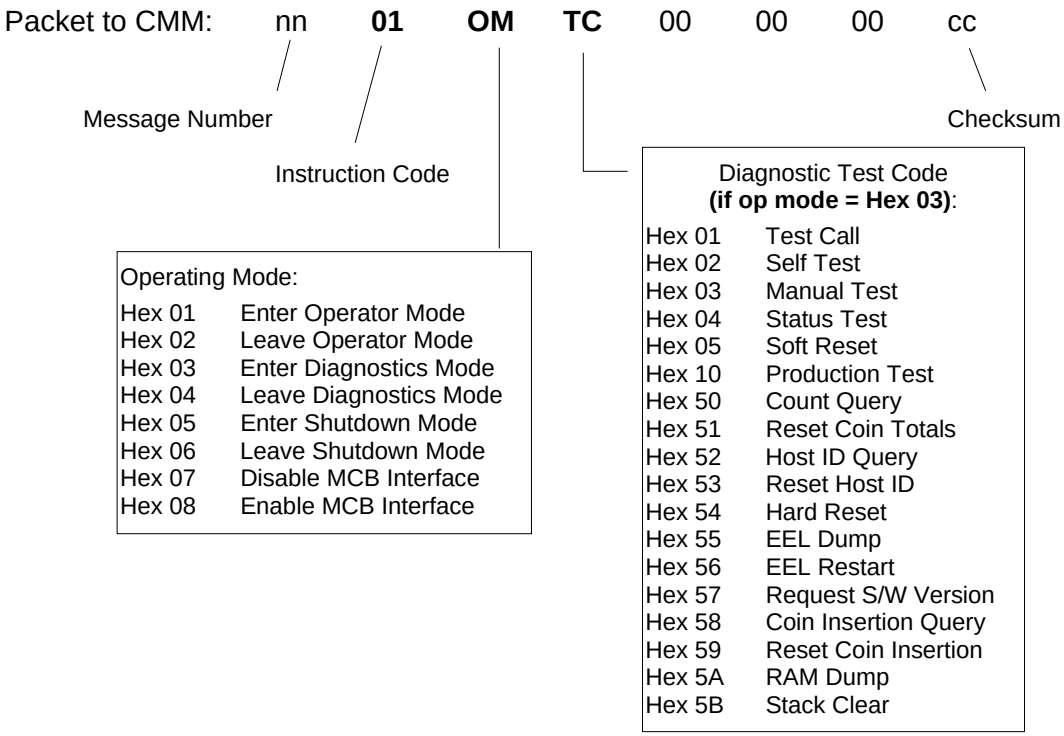
The message number byte can be used to determine if repeat messages are being received. When the transmitting node initiates the sending of a message, the message number is incremented. The receiving node can then compare the message number of the last message and the new incoming message, and similarly the message type, and decide whether the message is a repeat. (A repeat message may be caused by the transmitting node not receiving the acknowledge correctly.) The incoming message can then be either ignored if it is a repeat message, or actioned. (The message must still be acknowledged as valid, in the normal manner)
Note: The receiving node can only transfer the 'incoming message' number into the 'last message' number store when the incoming message has passed all tests and the instruction received actioned.
If the transmitting node does not receive a valid reply (either ‘NACK’ or 'acknowledge') before the 'no reply' timer expires then the transmitting node will attempt to re-send the message, up to a maximum of MAX. TRANSMISSION RETRY (recommended value 3) times. After the maximum number of re-sends, the node will stop sending that message and increment an interface failure counter. This failure counter can then be interrogated via diagnostics or self test routines.
If a message fails to be transmitted, it will remain on the message queue, and further attempts to send the message will be made when the transmitting node is reactivated. After the node has been reset, all un-transmitted messages are removed from the queue, except for fault type messages.

5. Interface Instruction Types

Figure 4, at the end of this section, shows all transactions available on the interface. Where a message contains data longer than a single byte, the data is passed with the least significant byte first.

5.1 Operating Mode. (Hex code 01)

This message is sent to the CMM to change the mode within the CMM.



The available CMM modes are: Operator mode, Diagnostics mode (see appendix 2), and Shutdown mode. Shutdown mode is the mode into which the CMM powers-up, and can be used to halt coin acceptance. Note: At the end of call, the CMM will need to be put into shutdown mode prior to the receipt of a final cost of call message. This command can also enable/disable the CMM interface.

The first data byte will contain the mode change for the CMM, which will stay in the selected mode until the next operating mode message is received, or the unit powers down, in which case the CMM returns to shutdown mode on power up. The Shutdown mode is the default mode, and mode changes available are as follows:

- In Shutdown mode and Call Mode all mode changes are available.
- In Diagnostic Mode only access to Shutdown mode is available.

The possible mode changes are listed above. The second byte contains the diagnostic test code (see appendix 2), which must be zero if diagnostics is not required.

5.2 Current Cost of Call. (Hex code 02)

This message is sent to the CMM at each charging period. The actual monetary cost of call (4 bytes) is passed within the message data field. This message will be ignored if the CMM is in a memory fail condition.

Packet to CMM:	nn	**02**	**cc**	**cc**	**cc**	**cc**	00	cc
			LSB			MSB		
			Monetary Cost of Call (Hex)					

5.3 Final Cost of Call. (Hex code 03)

This is sent to the CMM at the end of a call sequence. The actual monetary cost of the call (4 bytes) is passed within the data field of the message. On receipt of this message the CMM will perform best refund calculations, and cash/refund the appropriate coins. This will be ignored if the CMM is in a memory fail condition.

Packet to CMM:	nn	**03**	**cc**	**cc**	**cc**	**cc**	00	cc
			LSB			MSB		
			Monetary Cost of Call (Hex)					

5.4 Coin Entered. (Hex code 04)

This message is sent by the CMM when coins are accepted by the CMM. The data field contains the monetary value of the coin inserted, (3 bytes) and the coin Id of the coin inserted. (1 byte)

Packet from CMM:	nn	**04**	**cc**	**cc**	**cc**	00	**id**	cc
			LSB		MSB			
			Monetary Value of Coin (Hex)				Coin ID	

5.5 Cashing Complete. (Hex code 05)

This message is sent by the CMM in response to a 'final cost of call' message. It indicates that CMM activity is complete. The data field contains the new cash box totals for monetary value (5 bytes) and total displacement value (4 bytes), and number of the coins cashed since the 'final cost of call' message. (1 byte)

1st Packet:	nn	**85**	**cv**	**cv**	**cv**	**cv**	**cv**	cc
			LSB	Cash box Total Value			MSB	

2nd Packet:	nn	**05**	**cd**	**cd**	**cd**	**cd**	**nn**	cc
			LSB	Cash box Total Displacement		MSB	Number of coins Cashed	

5.6 Refund Amount. (Hex code 06)

This message is sent by the CMM after a 'final cost of call' message. The data field contains the monetary value of the refund (4 bytes) to be given.

Packet from CMM:	nn	**06**	**rv**	**rv**	**rv**	**rv**	00	cc

LSB MSB
Refund Value (Hex)

5.7 CMM Parameters. (Hex code 07)

This command message is sent to the CMM. This message will contain at least 2 packets, the first packet containing a 1-byte identifier in the first data byte, indicating the data type being sent. The second (and any following packets) will contain the relevant data, in the strict order described in following parameter descriptions.

1st Packet to CMM:	nn	**87**	**pp**	00	00	00	00	cc
nth Packet to CMM:	nn	**87**	**dd**	**dd**	**dd**	**dd**	**dd**	cc
					:			
Last packet to CMM:	nn	**07**	**dd**	**dd**	**dd**	**dd**	**dd**	cc

Parameter Identity (see below)

Parameter Data

Parameter Data

After this message has been received, the CMM must reply with a 'CMM parameter update status' message, which will indicate success or failure of the update, and the nature of any fault that may occur.
Parameters that can be updated are listed in Appendix 3.

5.8 CMM Parameter Update Status. (Hex Code 08)

This message is sent in response to a CMM parameter update, and indicates the result of a CMM parameter update message. The first data byte contains a return code, which indicates the success or failure of the parameter update and is a reason code should failure occur. The second byte is an echo of the parameter_id.

Packet from CMM: nn **08** **dd** **dd** 00 00 00 cc

Return Code | Parameter Identity Echo |

Table 2: Parameter Update Codes

Return Code	Money Controls Acceptor Reason	CMM In Service
00	Successful Data Transfer	Yes
01	Calibration Denied	No
02	Calibration Recharge Required	No
03	Product Mismatch	No
04	Database Version Number Mismatch	No
05	Non Programmable Acceptor	Yes
06	Data Transfer Time out	No
07	Data Oversize	No
08	Data Undersize	No
09	Invalid Parameter ID	Yes
250-255	Acceptor Calibration Error	No

If the return code 00 is received, then the CMM has now been correctly programmed.

If the return code 05 is received, then CMM service is not affected, but further coin set update should not be attempted. (Coin value & coin displacement updates are still available).

If return code 09 is received, then CMM service is not affected, and the CMM continues to use the old data.

Any other return code will indicate the appropriate failure, and will result in the CMM being out of service. The parameter-id echo enables the user to see during which update the failure occurred.

5.9 Cash box Status. (Hex code 09)

This event message is sent to the CMM when the Cash Box State is changed. The first data byte indicates the new state, and is set to 0F(hex) for cash box removed, and F0(hex) for cash box replaced. On Cash box removal, a cash box data reply message is sent by the CMM.

Packet to CMM: nn **09** **dd** 00 00 00 00 cc

Cash box Status
Cash box Removed: 0F (hex)
Cash box Replaced: F0 (hex)

5.10 Test Result. (Hex code 0A)

This message is sent by the CMM after completion of a diagnostic test. The data bytes contain the results of the test. (See appendix 2.)

Packet from CMM: nn **0A** **dd** **dd** **dd** **dd** **dd** cc

Test Results

5.11 Fault Message. (Hex code 0B)

This event message is sent by the CMM when a fault is detected by the CMM. The data space (bytes 3 to 7 in the message packet) in the fault message provides information about the fault, in the following format.

Packet from CMM: nn **0B** **ft** **ft** **ss** **aa** **aa** cc

LSB MSB — Fault Type (hex)
Additional Data
Severity & Fault Status

The first and second data byte describes the fault type. Each group is attributed an offset, these being 1100 and 1400 respectively, to which the fault code is added.
The third data byte has 2 functions, indicating the fault severity, and whether a fault is being raised or cleared. The high nibble describes the fault severity and the low nibble describes the fault status.

Table 3: Fault Messages

Data Byte		Meaning
Severity	0x	Warning Fault, coin call service is unaffected.
	1x	Fatal Fault, coin call service is disabled.
Status	x0	Incident, no raise or clear.
	x1	Raising a fault.
	x2	Clearing a fault.

The fourth and fifth byte may contain fault-specific additional data.

The fault conditions and fault codes (decimal, with hex value in brackets) sent by the CMM are as follows:

Table 4: Fatal Store Faults

	Store
Motor Current Fail	**1400** (578 hex)
Cashing Errors Exceeded	**1401** (579 hex)
Store Blocked	**1402** (57A hex)
Store Positioning Failed	**1403** (57B hex)
Store Sensor Fail	**1404** (57C hex)
Refund Fail	**1405** (57D hex)

Table 5: Warning Store Faults

	Store
Refund NOT Cash Error	**1410** (582 hex)
Cash NOT Refund Error	**1411** (583 hex)
Coin Stuck in Pocket	**1412** (584 hex)

Table 6: Fatal Faults

Flight Deck Open/Closed	**1120** (460 hex)
Escrow Entry Blocked	**1121** (461 hex)
Coin Exit Sensor Blocked	**1122** (462 hex)
CMM Memory Fail	**1123** (463 hex)

Table 7: Warning Faults

CMM/Acceptor Interface Fail	**1130** (46A hex)
Not used	**1132** (46C hex)
Coin Insertion Limit Exceeded	**1133** (46D hex)
Max. Coin Rejects Exceeded	**1134** (46E hex)
Coin Reject/Coin Accept ratio Exceeded	**1135** (46F hex)

5.12 Send Inhibit Status. (Hex code 0C)

This message is sent to the CMM to request the inhibit status of the coin set in the CMM. No data is passed with this message.

Packet to CMM:	nn	**0C**	00	00	00	00	00	cc

5.13 Inhibit Status Reply. (Hex code 0D)

This message is sent by the CMM in response to the 'send inhibit status' message. The data field contains the current inhibit map for the CMM. (2 bytes).

Packet from CMM:	nn	**0D**	**dd**	**dd**	00	00	00	cc

Inhibit Map, coins 8 to 15

MSB	Coin 15	Coin 14	Coin 13	Coin 12	Coin 11	Coin 10	Coin 9	Coin 8	LSB

Inhibit Map, coins 0 to 7

MSB	Coin 7	Coin 6	Coin 5	Coin 4	Coin 3	Coin 2	Coin 1	Coin 0	LSB

For each bit: Bit = 1, coin is inhibited, Bit = 0, coin is enabled.

5.14 Request Cash box Data. (Hex code 0E)

This command message is sent to the CMM, whenever an update of the cash box data is required. If, for example, a Host MCB is removed for repair, the CMM and the cash box need not be replaced, and the new MCB module uses this message to find the present cash box totals. No data is passed within this message.

Packet to CMM:	nn	**0E**	00	00	00	00	00	cc

5.15 Cash box Data Reply. (Hex code 0F)

This message is sent by the CMM in response to the 'request cash box data' message. The data field contains the cash box value total (5 bytes) and the cash box displacement total (4 bytes).

Packet 1 from CMM:	nn	**8F**	**dd**	**dd**	**dd**	**dd**	**dd**	cc

LSB MSB

Cash box Total Value

Packet 2:	nn	**0F**	**dd**	**dd**	**dd**	**dd**	00	cc

LSB MSB

Cash box Total Displacement

5.16 Request Acceptor Configuration Data. (Hex code 10)

This message is sent to the CMM to request the acceptor configuration data, and will normally be used during installation or re-initialisation of the host. No data is passed within this message.

Packet to CMM:	nn	**10**	00	00	00	00	00	cc

5.17 Acceptor Configuration Data Message. (Hex code 11)

This message is sent by the CMM in response to the 'request acceptor configuration data' message, or after a 'CMM Status' message where either one or both the CMM and/or acceptor has been changed. There are 13 bytes to be transmitted, therefore requiring three packets. The data field contains the calibration standard, acceptor reference number, Master Set Code, Acceptor Programming Mode, Signature File Id, and the Signature File Id Version in the order shown below. This data enables coin set tables to be chosen.

Packet 1 from CMM:	nn	**91**	**cs**	**cs**	**cs**	**cs**	00	cc
			|_____ Calibration Standard _____|					
Packet 2:	nn	**91**	**vr**	**vr**	**vr**	**vr**	00	cc
			|_____ Acceptor Reference Number _____|					
Packet 3:	nn	**11**	**ms**	**ms**	**pm**	**id**	**ve**	cc

ms ms — Master Set Code

pm — Acceptor Programming Mode

id — Signature File ID

ve — Signature File ID Version

The table below shows the settings for the Money Controls C120P

Table 8: Money Controls C120P Configuration Data Messages

	Money Controls Acceptor
Calibration Standard	1 byte (1st byte, & 3 zero filled)
Acceptor Reference No.	3 Bytes (1st 3 bytes & 1 zero filled)
Master Set Code	Not Applicable (set to zero)
Acceptor Prog. Mode	Not Applicable (set to zero)
Signature File ID	1 Byte
Signature File ID Version	1 Byte

5.18 Call Start Message. (Hex code 12)

This message is sent to the CMM at the start of every call sequence. The total number of bytes transmitted in this message is 7, therefore this message is required to be sent in two packets. The data field(s) contain the Host identity (6 bytes), and the acceptor program flag. The Host identity is used by the CMM to determine if it has been relocated since the last power-up. The acceptor program flag enables/disables the coin download facility in the CMM, providing the installed acceptor has coin download capabilities. This flag does not affect an acceptor without coin download capability. This message also causes a mode change to Call mode.

The Acceptor Programming Mode is defined as:

00	disables coin download
10 (Hex)	enables coin download

Packet 1 to CMM: nn **92** **id** **id** **id** **id** **id** cc

MSB Host ID

Packet 2: nn **12** **id** **vv** 00 00 00 cc

Host ID (LSB) Host Programming Mode

5.19 CMM Status. (Hex code 13)

This message is sent by the CMM in response to the 'Call Start' message. The data field contains the CMM site status byte, the remaining 4 bytes are blank. The CMM site status byte is used to decide whether the CMM installed requires a new download.

Packet from CMM: nn **13** **ss** 00 00 00 00 cc

Site Status

Table 9: CMM and Coin Acceptor Status Messages

Site Status Byte	CMM and Acceptor status
0000 0000	CMM unchanged.
0000 0001	CMM changed.
0000 0010	CMM unchanged, Acceptor changed, and requires a download.
0000 0100	CMM unchanged, Acceptor changed, does not require a download.
1xxx xxxx	Non-Programmable Acceptor in CMM.

5.20 Request Coin Counts (Hex code 14)

This message is sent to the CMM to request the current cashed coins count. No data is passed within this message.

Packet to CMM:	nn	**14**	00	00	00	00	00	cc

5.21 Coin Counts Reply (Hex code 15)

This message is sent by the CMM in reply to the request coin counts message. The data shows the number of coins of each type that have been successfully routed to the cash box. Messages are sent, with each message containing the data for 2 coins, until data for each coin slot in the acceptor has been sent.

Packet 1 from CMM:	nn	**95**	**cc**	**cc**	**cc**	**cc**	00	cc
			LSB	MSB	LSB	MSB		

No. of Coin type 1 cashed

No. of Coin type 2 cashed

:
:

Packet n/2:	nn	**15**	**cc**	**cc**	**cc**	**cc**	00	cc
			LSB	MSB	LSB	MSB		

No. of Coin type n-1 cashed

No. of Coin type n cashed

Messages are sent in the same format until data for all coins has been sent.

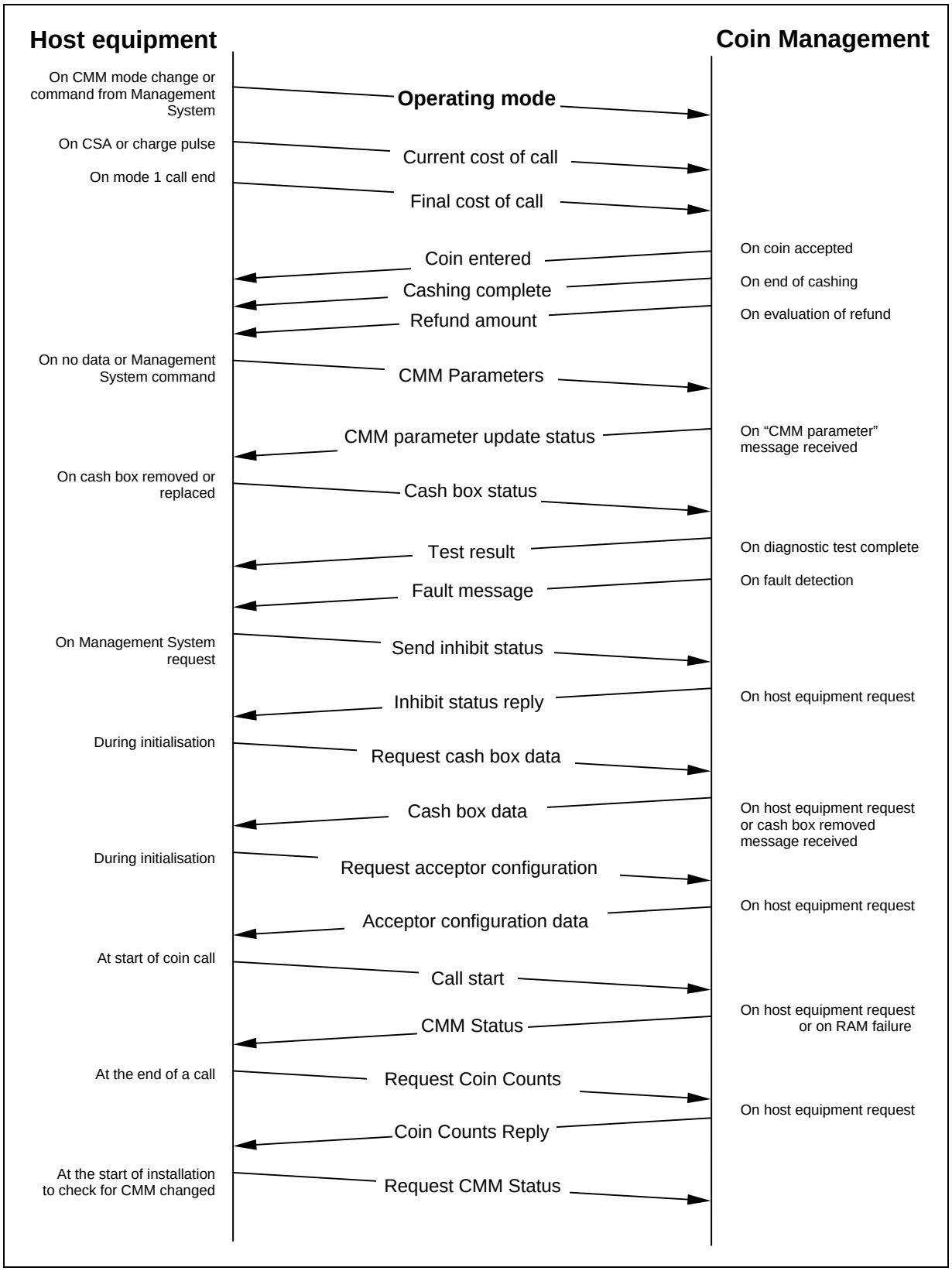
5.22 Request CMM Status Message. (Hex code 16)

This message is sent to the CMM to request the CMM status. It has the same format and affect as the 'Call Start' message, except that the CMM does not change mode to Call mode.

1st Packet:	nn	**96**	**id**	**id**	**id**	**id**	**id**	cc
			MSB		Host ID			

2nd Packet:	nn	**16**	**id**	**vv**	00	00	00	cc
			Host ID (LSB)		Acceptor Programming Mode			

Figure 4: Coin Management - Host Equipment data interchange



6. Appendix 1: Installation & Initialisation Routine

When the host equipment powers-up, the internal modules must be able to communicate, and follow a defined procedure to determine if any or all of the modules have been replaced. If it is the case, that new modules have been situated in the host equipment, then initialisation of these new modules is needed.

This appendix describes this procedure, and the required functionality of both the CMM and the host equipment. Figure 2 shows the message interchange for this procedure.

The procedure can be divided up into 2 phases, these being:

1. Installation
2. Initialisation

On every power-up condition, the initialisation procedure will begin at phase 1.

6.1 Phase 1: Installation

After power-up, the host equipment will begin the procedure by evaluating whether it needs to download new data from the Management System. If this is not required, then the procedure moves directly to phase 2, otherwise the host equipment should request Acceptor data from the Management System.

The host equipment receives all data up to the coin signature download service, and then decides from previous data if the coin signature service is required.

If the coin signature service is required, the host equipment must contact the CMM in order to obtain the acceptor configuration data, using the "request acceptor configuration data" message. The CMM having done a first stage internal initialisation, will interrogate the acceptor and send the information back to the host equipment using the 'acceptor configuration data' message. This data is then used in creating the correct coin signature database to be returned to the host equipment.

If the host equipment does not receive a reply from the CMM, then it will assume that no CMM is present, and a fault report is required to indicate that access to the CMM has failed.

The host equipment now sends a 'Request CMM Status' message to the CMM, containing the host identity, and the acceptor programming mode. The CMM returns a 'CMM Status' message, which describes to the host equipment which data is required, if any. The host equipment sends the data to the CMM using the 'CMM Parameter' message.

Once the download has been received, the procedure moves on to phase 2, but note that the host identities will match, and the host will be in full service, and able to accept coins.

6.2 Phase 2: Initialisation.

In order to initialise the CMM, the host equipment must ensure that the CMM currently installed has not been exchanged since the last power-up. The host equipment sends the 'Call Start' message to the CMM, which contains the host identity and the acceptor program mode for the host. The CMM will then compare the received host identity, with the host identity already held by the CMM. The CMM also ensures on power-up, that the acceptor has not been changed.

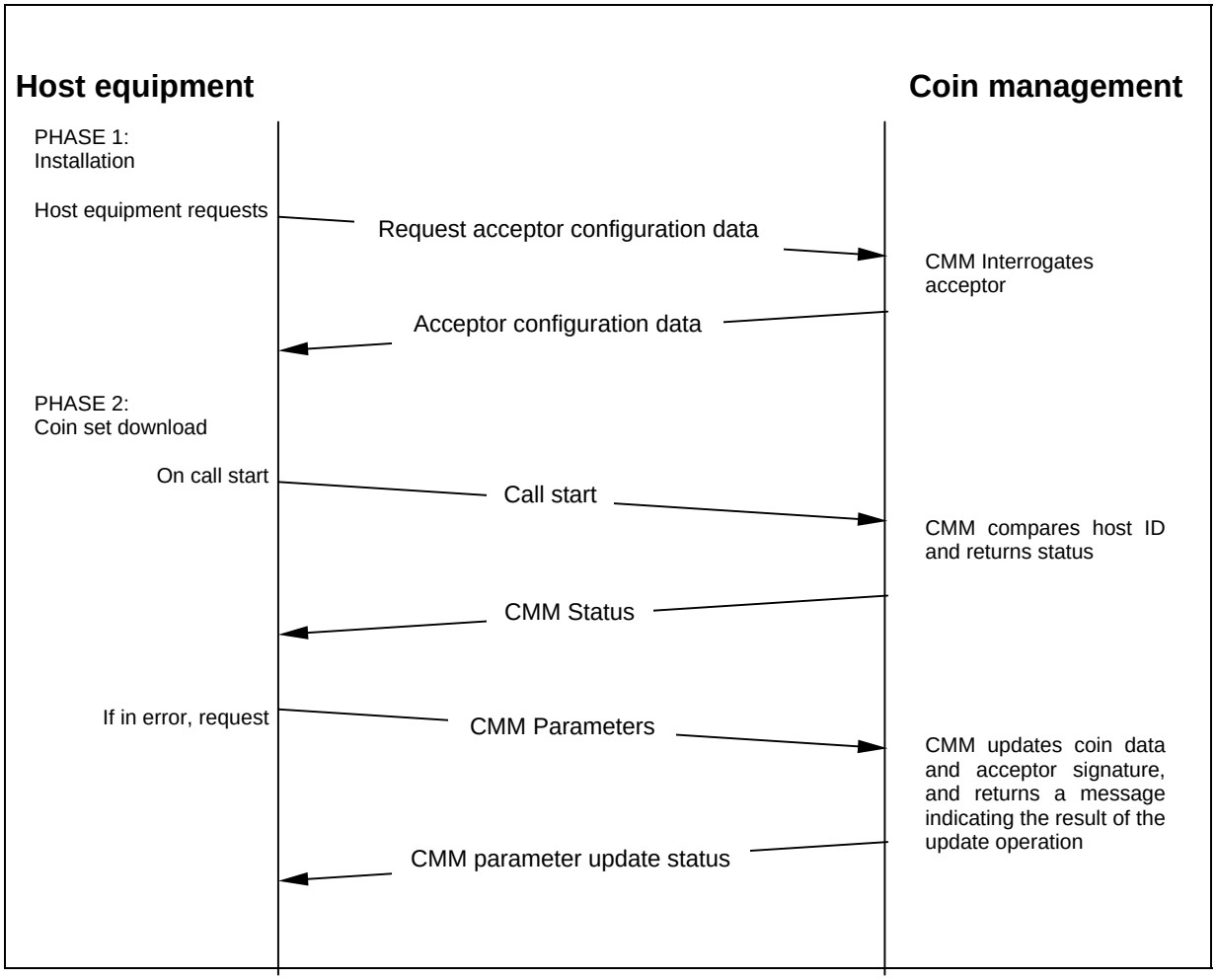
If the acceptor identities are different, and/or the acceptor has changed with the CMM unable to re-program it, then this CMM requires re-initialisation. A 'CMM status' message is sent indicating as such, followed immediately by a 'Acceptor Configuration Data' message. The host equipment now has the required data to obtain a coin set download, and if the acceptor config. data is different to that which it has already stored, then at the end of the call the host equipment will communicate with the Management System, receive the appropriate download, and transfer the data to the CMM (and hence the acceptor). When coin set download is complete, the host equipment must also convey the new host identity, and the latest cash box totals, to the CMM using the 'CMM parameter' message. Once this is done, initialisation is complete.

If the host identities and the acceptor configuration data match, then this CMM has not been exchanged, and the 'CMM status' message is sent indicating such. If the acceptor configuration data is different but the CMM was able to re-program the acceptor, then a 'acceptor configuration data' message will be sent following the 'CMM status' message. (The status will be set CMM unchanged, Acceptor changed, and does not require a download). This will allow the host equipment to update it's own configuration data.

The host is now ready to accept coins. (Note: If the scenario is such that the host equipment has been changed, but not the CMM, then at the end of this call sequence, the cash box totals in the host equipment will be updated as normal, in the 'cashing complete' message.)

These 2 phases of the installation/initialisation procedure can be seen in figure 2 which shows the typical message interchange for the case of a host being installed with a 'new' host equipment and a 'new' CMM.

Figure 5: Host Initialisation Procedure



7. Appendix 2: Diagnostic Commands

The CMM diagnostic mode is entered using the operating mode command (instruction 01, mode change 03), and followed with a test number. Any one from a suite of tests can be performed by sending the 'enter diagnostic mode' operating message with the test code byte (2nd data byte) set to one of those listed below.

7.1 Test Call. (Test Code 01)

This test puts the CMM into test mode, and behaves as if a standard call is in progress, except that any coins cashed will not be added to the coin totals. No data is expected from the CMM in response.

Packet to CMM:	nn	**01**	**03**	**01**	00	00	00	cc

7.2 Self-Test. (Test Code 02)

This test performs a series of tests without the need for any input from the test engineer, and will check the CMM/Acceptor interface link, all sensors, and the motor. A test result message will be returned to the host equipment, with the outcome of the test described by the data bytes in the following format. The first block will indicate the average time in milliseconds for the motor to turn a store through one pocket, (i.e. 1/9th of a revolution). The value is only valid if the motor test has passed as indicated in the second block, and is contained in 2 bytes, with the lowest significant byte first.

Packet to CMM:	nn	**01**	**03**	**02**	00	00	00	cc

	nn	**8A**	**ms**	**ms**	**ms**	**ms**	00	cc

Average Time for 1 pocket to rotate (ms).

	nn	**0A**	**tr**	**tr**	**tr**	**tr**	**tr**	cc

Spare

Acceptor Interface Test Result

Motor Test Result

Sensor Test Result

Acceptor Self Test Result

Table 10: Acceptor Self Test result byte for Money Controls C120P

Data	Meaning
0	OK (No Fault Detected)
1	EEPROM checksum corrupted
2	Fault on Inductive Coils
3	Fault on Credit Sensor

Table 11: Sensor test result byte:

0000	0000	Test Passed
xxxx	xx01	Test inconclusive, Store Jammed.
xxxx	xx10	Store sensor permanently OFF
xxxx	xx11	Store sensor permanently ON
xxxx	01xx	Not used.
xxxx	10xx	Reverse store sensor permanently OFF
xxxx	11xx	Reverse store sensor permanently ON
xx10	xxxx	Exit sensor permanently OFF
xx11	xxxx	Exit sensor permanently ON
10xx	xxxx	Gate sensor permanently OFF
11xx	xxxx	Gate sensor permanently ON

Table 12: Motor test result byte:

0000	0000	Test Passed
xxxx	0001	Motor stalled
xxxx	0010	Motor Disconnected
xxxx	0011	Blockage in store
xxxx	0100	Store Position Fail
xxxx	0101	Store Not Moving
xxxx	0110	Store Rotation Speed varies by ±12.5%
xxxx	0111	Unable To Perform Test
xxxx	1xxx	Not used
0001	xxxx	Not used
0010	xxxx	Not used
0011	xxxx	Not used
0100	xxxx	Not used
0101	xxxx	Not used
0110	xxxx	Not used
0111	xxxx	Not used
1xxx	xxxx	Not used

(x = does not matter.)

Table 13: Acceptor Interface test result byte:

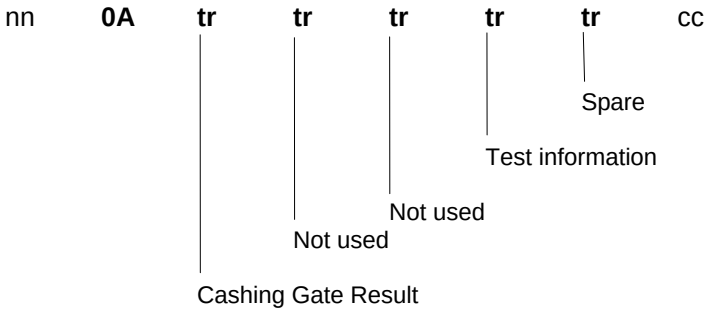
Data	Meaning
08	Transmit Message Fail
40	Receive Message Fail
80	No Response Time out

For a complete list of all possible acceptor fault codes see acceptor specification.

7.3 Manual Test (Test Code 03)

Packet to CMM: nn **01** **03** **03** 00 00 00 cc

This test checks the operation of the cashing gates. This test requires that the test engineer fill the CMM with 8 coins. On each coin entered, a coin accepted message is sent to the host equipment in the normal manner. When the escrow is full, all coins are then refunded and cashed alternately (refund first), and a test result message sent. The format for the test result will be as described below.



Cashing Gate Test Results:

High Nibble	Low Nibble
Number of Refund-not-Cash failures	Number of Cash-not-Refund failures

Test Information:

The low nibble indicates the number of coins fed into the CMM. If the high nibble is set to 1111, then the test has been aborted due to a time out.

7.4 Status Test (Test Code 04)

Packet to CMM: nn **01** **03** **04** 00 00 00 cc

This test polls all the fault flags within the CMM, and returns a test result message indicating each flag's state, in the following format.

Packet from CMM: nn **0A** **tr** **tr** **tr** **tr** **tr** cc

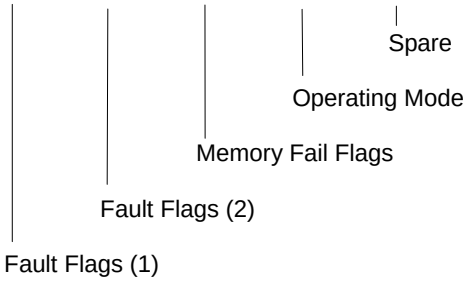


Table 14: Fault Flags (1):

0000	0000	No Faults
1xxx	xxxx	Store Disabled
x1xx	xxxx	Not used
xx1x	xxxx	CMM/MCB Interface Fail
xxx1	xxxx	Coin exit Sensor Failed
xxxx	1xxx	CMM Cashing Off
xxxx	x1xx	CMM/Acceptor Interface Fail
xxxx	xx1x	CMM Full
xxxx	xxx1	Coin Entered

Table 15: Fault Flags (2):

0000	0000	No Faults
1xxx	xxxx	Coin Store Moving
x1xx	xxxx	CMM/MCB Interface Operating
xx1x	xxxx	Updating CMM Parameters
xxx1	xxxx	Inhibit Coins
xxxx	1xxx	Cash box Out

Table 16: Memory Fail Flags:

0000	0000	Memory OK
xxxx	xxx1	Coin Signature Memory Fail
xxxx	xx1x	Coin Data Memory Fail
xxxx	x1xx	Cash box Memory Fail
xxxx	1xxx	Host Identity Memory Fail

Table 17: Operating Mode:

01	Call Mode.
03	Diagnostics Mode.
04	Shutdown Mode.

7.5 Soft Reset (Test Code 05)

Packet to CMM: nn **01** **03** **05** 00 00 00 cc

This test is used to reset the flags listed in test 3. All Fault flags will be reset on the next power-up cycle. When this message is received, the CMM will also cease to accept coins until instructed otherwise. No data is expected from the CMM in response.

7.6 Production Test (Test Code 10)

Packet to CMM: nn **01** **03** **10** 00 00 00 cc

This test is used to place the CMM in Production Test mode. In this mode the following CMM messages may be received from the CMM in the appropriate circumstances:

Full Escrow: nn **50** 00 00 00 00 00 cc

Payment Rejected: nn **51** **r1** **r2** 00 00 00 cc
|_____|

(Ref. C120P Serial Communication Protocol Interface & Command Specification Iss 3.2 Para 7.5)

Cashed Coin: nn **52** **dd** **dd** **dd** **dd** 00 cc
|_____|
LSB MSB
Coin Value

7.7 Count Query - inserted, rejected & cancelled coins (Test Code 50)

Packet to CMM:	nn	**01**	**03**	**50**	00	00	00	cc

This is used to request the number of inserted coins in each category since the last time the totals were reset. Data for two coins is contained in one packet; the number of packets depends on the number of coin types allowed in the acceptor.

1st Packet:	nn	**8A**	**dd**	**dd**	00	**dd**	**dd**	cc
			LSB	MSB		LSB	MSB	
			Coin 1 Count			Coin 2 Count		
					:			
					:			
(n/2)th:	nn	**8A**	**dd**	**dd**	00	**dd**	**dd**	cc
			LSB	MSB		LSB	MSB	
			Coin n-1 Count			Coin n Count		
1+(n/2)th:	nn	**0A**	**dd**	**dd**	00	**dd**	**dd**	cc
			LSB	MSB		LSB	MSB	
			Rejected Count			Cancelled Count		

7.8 Reset Coin Totals (Test Code 51)

Packet to CMM:	nn	**01**	**03**	**51**	00	00	00	cc

This is used to reset the inserted, rejected & cancelled coin totals. No data is expected from the CMM in response.

7.9 Host ID Query (Test Code 52)

Packet to CMM:	nn	**01**	**03**	**52**	00	00	00	cc

This is used to request the host ID currently stored in the CMM. Note: The host ID is in packed BCD, each unused nibble filled with Hex E (Decimal 14).

1st Packet:	nn	**8A**	**id**	**id**	**id**	**id**	**id**	cc
			MSB		Host ID			
2nd Packet:	nn	**0A**	**id**	00	00	00	00	cc
			Host ID (LSB)					

7.10 Reset Host ID (Test Code 53)

Packet to CMM:	nn	**01**	**03**	**53**	00	00	00	cc

This is use to set the Host ID in the CMM to blanks. No data is expected from the CMM in response.

7.11 Hard Reset (Test Code 54)

Packet to CMM:	nn	**01**	**03**	**54**	00	00	00	cc

This is used to cause the RAM in the CMM to be cleared on the next power-up cycle. No data is expected from the CMM in response.

7.12 EEL Dump (Test Code 55)

Packet to CMM:	nn	**01**	**03**	**55**	00	00	00	cc

This is used to download the EEL stored in the CMM. The EEL records are 6 byes long and are transmitted from the CMM in a continuous data stream within a sequence of messages.

1st Packet:	nn	**8A**	**d1**	**d1**	**d1**	**d1**	**d1**	cc
			Incident 1					
2nd Packet:	nn	**8A**	**d1**	**d2**	**d2**	**d2**	**d2**	cc
			Incident 1	Incident 2				
				:				
				:				
3600th:	nn	**0A**	**dd**	**dd**	**dd**	**dd**	**dd**	cc
				Incident 3000				

7.13 EEL Restart (Test Code 56)

This message is used to initalise the contents of the EEL in the CMM. No data is expected from the CMM in response.

Packet to CMM:	nn	**01**	**03**	**56**	00	00	00	cc

7.14 Request S/W Version (Test Code 57)

This message is used to request the Version of the CMM Software.

Packet to CMM:	nn	**01**	**03**	**57**	00	00	00	cc

The data is returned in the following format:.

Packet From CMM:	nn	**0A**	**sv**	**sv**	00	00	00	cc
			CMM Major (BCD)	CMM Minor (BCD)				

7.15 Coin Insertion Query (Test Code 58)

This message is used to request count of total coins inserted.

Packet to CMM:	nn	**01**	**03**	**58**	00	00	00	cc

The data is returned in the following format:

Packet from CMM:	nn	**0A**	**cc**	**cc**	00	00	00	cc
			LSB	MSB				
			Coin Insertions					

7.16 Reset Coin Insertions (Test Code 59)

This message is used to clear the coin insertion counter. No data is expected from the CMM in response.

Packet to CMM:	nn	**01**	**03**	**59**	00	00	00	cc

7.17 RAM Dump (Test Code 5A)

This message causes the CMM to send the contents of the RAM in a sequence of packets.

Packet to CMM:	nn	**01**	**03**	**5A**	00	00	00	cc

The data is returned in a total of 400 packets comprising 2,000 bytes of data, in the following format.

Packet 1 from CMM:	nn	**8A**	**dd**	**dd**	**dd**	**dd**	**dd**	cc
			RAM Data					
					:			
					:			
Packet 400:	nn	**0A**	**dd**	**dd**	**dd**	**dd**	**dd**	cc
			RAM Data					

7.18 Stack Clear (Test Code 5B)

This message is used to initalise the contents of the stack in the CMM. No data is expected in response from the CMM.

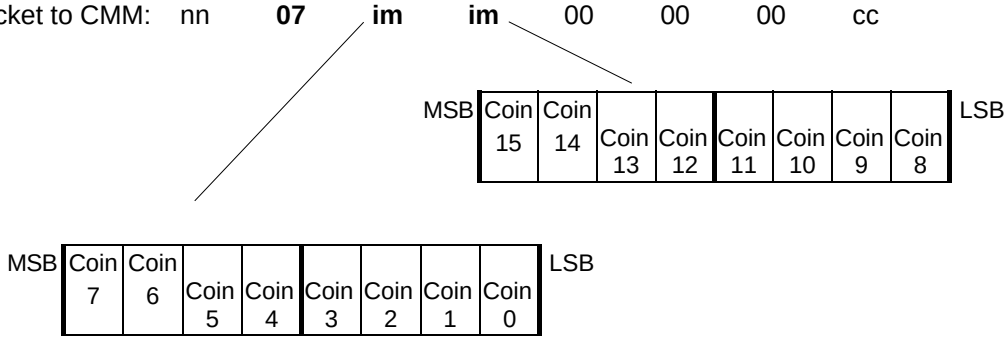
Packet to CMM:	nn	**01**	**03**	**5B**	00	00	00	cc

8. Appendix 3: Programmable Data Commands.

8.1 Coin Inhibit Map, (Hex code 01)

This data defines coins to be inhibited in the coin map.(2 bytes long).

1st Packet to CMM:	nn	**87**	**01**	00	00	00	00	cc
2nd Packet to CMM:	nn	**07**	**im**	**im**	00	00	00	cc



For each bit Bit = 1, coin is inhibited, Bit = 0, coin is enabled.

8.2 Coin Data Table, (Hex code 02)

1st Packet to CMM:	nn	**87**	**02**	00	00	00	00	cc
					:			
					:			
nth Packet to CMM:	nn	**07**	**dd**	**dd**	**dd**	**dd**	**dd**	cc
			|_____	_____	_____	_____	_____|	
					Coin Data			

The data, which is 16 bytes per coin in length, is split into blocks, each five bytes long, for transmission in the packet stream.The data structure (per coin) is shown below, therefore for n coins the structure must be repeated n times.

Table 18: Coin Data Table Download Data

Field	Length
Coin Code	1 byte
Coin ID	1 byte
Sig File Id	1 byte
Sig File Id Version	1 byte
Spare	3 bytes
Inhibit Status	1 byte
Coin Value	3 bytes
Coin Displacement	2 bytes
Coin ID Tones	1 byte
Coin Pulses	1 byte
Padding	1 byte
Total	**16 bytes**

8.3 Coin Signature Table, (Hex code 03)

1st Packet to CMM:	nn	**87**	**03**	00	00	00	00	cc
				⋮				
nth Packet to CMM:	nn	**07**	**dd**	**dd**	**dd**	**dd**	**dd**	cc

Coin Signature Data

This data, which is 136 bytes per coin long, is used for coin acceptor remote coin update. The data is split into blocks, each five bytes long, for transmission in the packet stream.

The data structure (per coin) is shown below, therefore for n coins the structure must be repeated n times. The internal data structure varies with the type of acceptor being used, although the data block is always the same size.

Table 19: Coin Signature Table Download Data

Money Controls C120P Download Data	
Field	**Length**
Coin ID	1 byte
Sig File Id	1 byte
Sig File Id Version	1 byte
Coin Signature:	
Programming Size	1 byte
Coin Type	1 byte
Coin Data	130 bytes
Padding	1 byte
Total	**136 bytes**

8.4 Cash box Totals, (Hex code 04)

The data, which is 9 bytes long, describes the cash box totals.

1st Packet to CMM:	nn	**87**	**04**	00	00	00	00	cc
2nd Packet to CMM:	nn	**87**	**cv**	**cv**	**cv**	**cv**	**cv**	cc

Cash box Total Value

3rd Packet to CMM:	nn	**07**	**cd**	**cd**	**cd**	**cd**	00	cc

Cash box Total Displacement

8.5 Host Identity, (Hex code 05)

The host ID, which is 6 bytes long, is in packed BCD, each unused nibble filled with Hex E (Decimal 14).

1st Packet to CMM:	nn	**87**	**05**	00	00	00	00	cc
2nd Packet to CMM:	nn	**87**	**id** (MSB)	**id**	**id** Host ID	**id**	**id**	cc
3rd Packet to CMM:	nn	**07**	**id** Host ID (LSB)	00	00	00	00	cc

8.6 Coin Count Totals, (Hex code 06)

The data, which is 2 bytes per coin long, describes the current coin counts for each coin.

1st Packet to CMM:	nn	**87**	**06**	00	00	00	00	cc
2nd Packet to CMM:	nn	**87**	**dd** LSB	**dd** MSB	**dd** LSB	**dd** MSB	00	cc
			Coin 1 Count		Coin 2 Count			
					:			
					:			
1+(n/2)th to CMM:	nn	**07**	**dd** LSB	**dd** MSB	**dd** LSB	**dd** MSB	00	cc
			Coin n-1 Count		Coin n Count (00s if n is odd)			

8.7 Report Thresholds, (Hex code 07)

1st Packet to CMM:	nn	**87**	**07**	00	00	00	00	cc
2nd Packet to CMM:	nn	**87**	**dd** Threshold for 1st Item	**dd**	**dd**	**dd**	**dd**	cc
					:			
					:			
8th Packet to CMM:	nn	**07**	**dd**	**dd** Threshold for 32nd Item	00	00	00	cc

The data, which is 29 bytes long, describes the value for each fault threshold. The thresholds specify the number of faults of each type which must occur before a fault message is sent across the interface.
A value of 0 disables the fault message for that fault type. The default values of the thresholds are show in the table below.

Table 20: Fault Report Thresholds

No.	Report Type	Default Threshold
1	Motor Current Fail	2
2	Not Used	0
3	Cashing Errors Exceeded	6
4	Not Used	0
5	Refund Errors Exceeded	6
6	Not Used	0
7	Store Blocked	0
8	Not Used	0
9	Store Position Fail	0
10	Not Used	0
11	Store Sensor Fail	2
12	Not Used	0
13	Refund Fail	0
14	Not Used	0
15	Refund Not Cash Fail	0
16	Not Used	0
17	Cash Not Refund Fail	0
18	Not Used	0
19	Flight Deck Status	1
20	Escrow Entry Blockage	2
21	Refund Sensor Blocked	4
22	CMM Memory Fail	2
23	CMM Acceptor Interface Fail	2
24	Acceptor Fraud Attempt	0
25	Not Used	0
26	Insertion Limit Exceeded	1
27	Maximum Rejects Exceeded	0
28	Accept/Reject Ratio Exceeded	0
29	No-Coin Calls Exceeded	0
30	Coin Acceptor Error 17 received	1
31	Coin Acceptor Error 19 received	1
32	Coin Acceptor Error 23 received	1

9. Appendix 4: Additional commands for units fitted with the C120S Coin Acceptor

This appendix details the modifications and extension to the CMM Interface Specification to permit the use of Money Controls C120S coin acceptors with the Orbit CMM. The main difference with the C120S coin acceptor is that individidual coin signature tables can be up to 2417 bytes long rather than the previous fixed 136 bytes for the C120P. This requires the use of an extended message frame format (Up to 251 bytes of data per frame rather than 5) and some additional standard commands. In addition a new format is used within the signature table download to permit multiple coin windows to be programmed with a single signature table.

9.1 Orbit Identification Code (Hex Code 11)

The existing coin Acceptor Configuration Data Message is changed to include an identification byte in packet 2. There are three packets sent by the CMM in response to a Request Acceptor Configuration Data Message (Hex Code 10). These are:

Packet 1:	nn	**91**	**ca**	00	00	00	00	cc
Packet 2:	nn	**91**	**vr**	**vr**	**vr**	**xx**	00	cc
Packet 3:	nn	**11**	**00**	**00**	**00**	**yy**	**zz**	cc

Where: ca = Calibration standard
vr = Coin Acceptor Reference No
yy = Signature File ID
zz = Signature File ID Version

Current xx is set to zero but in future 9 Coin CMM units will set xx to 00 and Orbit units with C120P coin acceptors will set xx to 80 and those with C120S coin acceptors will set xx to C0.

9.2 CMM Busy due to Signature File Download (Hex Code 17)

This message is sent by the Orbit CMM to signify that a signature file download to the C120S coin acceptor is in progress. It is intended to prevent the payphone or host system from switching off the power to the CMM during a critical phase. The command is sent by the CMM at the start and end of the download process and at least once per 20 seconds during the process. This then enables the payphone to instigate a 30 second timeout which should be re-triggered on receipt of the command. While in this mode the power to the CMM should be maintained and no commands issued by the payphone until the process is complete.

Packet from CMM:	nn	**17**	**ss**	00	00	00	00	cc

The byte ss will be set to AA for download in process or 00 for download complete. Note that this feature is supplemental to the CMM Parameter Update Status (Hex Code 08) which is issued by the CMM once the signature table has been loaded into the C120S coin acceptor. This is simply a means to ensure that power is maintained by the payphone until the programming of the coin acceptor's data is complete.

9.3 Extended Coin Signature Table Download

The normal packet structure for communications between the payphone and CMM can take a maximum of 5 data bytes. While this was acceptable for signature tables of 136 bytes as used by the C120P, the tables used by the C120S coin acceptor (up to 2417bytes in length) require a more efficient means of download. The following system switches the standard data packet size over to an extended packet size.

Packet 1 to CMM:	nn	**87**	**08**	00	00	00	00	cc
Packet n to CMM:	nn	**87**	**le**	**…….. data …….**				cc
Final Packet:	nn	**07**	**le**	**…….. data …….**				cc

Where: data is variable length data defined in section *9.4*
le is the length of that data up to a maximum of 251

In order to simplify the communications software, the length of the data in a packet should be fixed at 251 bytes although generically the structure can take anything up to 251 bytes. The remaining data in the final packet will normally be less than 251 bytes so this packet should be padded out with zeros to maintain the structure. The data sent by this command to be reconstructed (concatenated) by the CMM software such that it is processed as one complete file.

The successful receipt of the final packet (header of 07 as opposed to 87) will also automatically switch the CMM receiving software back to normal size data packets.

9.4 C120S Download Data Structure

The data file sent by the above command (section *9.3*) should be constructed as follows:

Header + Signature Table + Header + Signature Table ….. + Termination Header

The file will be contiguous and is not required to be synchronized to the packet size. The header is defined as 6 hexadecimal bytes as follows:

AA 55 bm1 bm2 le1 le2

Where bm is a bit map of the windows to which the following signature table refers. Bm1 is the low order (bits zero thru 7 referring to windows 1 thru 8) and bm2 is the high order (bits zero thru 7 referring to windows 9 thru 16 respectively).

le1 & 2 are the length of the following signature table, low byte first.

The signature table is to be concatenated exactly as supplied by Money Controls.

The Termination header follows similar rules but uses a length of 0xFFFF and signifies to the CMM that the download is complete. The reason for this is so that the bit map bytes can then be used for signature file version tracking information:

AA 55 yy zz FF FF

Where yy is the signature file ID and zz the version number. The main function of this information is to permit the CMM to report this data in its Coin Acceptor Configuration Data Message – See section *9.1*. There is, however, a secondary function of these two bytes. If they are set to zero then the information just sent will be stored in the CMM memory without initiating a download to the C120S coin acceptor. This then provides for the signature tables to be sent individually or as one big file. To initiate the download from the CMM memory to the C120S, all that is necessary is to ensure that at least one of these bytes is non-zero in the last termination header sent.

There is a special case of the header to delete a previously sent window or to initialise the CMM. This is:

AA 55 bm1 bm2 00 00

The zero length bytes will indicate to the CMM that the windows specified by the bit map should be set to unused. It would thus be good practice to send the following header at the start of a complete new download:

AA 55 FF FF 00 00

Note that this can be followed immediately by the first header and signature table or (if required to be left in an un-programmed state without signature tables) by the termination header.

Upon receipt of the termination header, the CMM will send a CMM Parameter Update Status packet. In the case of either of the signature file ID / Version no. bytes (see above) being non-zero then this packet will be delayed while the tables are sent on to the C120S coin acceptor. In either case the payphone may send the next signature table or resume normal operation. In the event of a bad download to the C120S coin acceptor, the unit will resume normal operation but with only the tables that were unaffected by the download. If the download contained several signature tables then the Request C120S Windows Command

will have to be used to detect which tables were valid and which failed. If the download contained just one file then the windows referred to in this download will remain in an un-programmed state.

It should also be noted that this system will set the windows according to the last bit map received. Thus if windows 1, 3 & 5 are specified in the first header, along with signature table 1 followed by windows 3, 6 & 7 in the second header and signature table 2, then the result will be that windows 1 & 5 use table 1 and windows 3, 6 & 7 use table 2.

9.5 Request C120S Windows Command (Hex Code 18)

This command is sent by the payphone to request information on which windows are programmed in the C120S coin acceptor.

Packet:	nn	18	00	00	00	00	00	cc

Note that this command will result in an unknown command reply for Orbit units fitted with non-C120S coin acceptors.

9.6 C120S Windows Command Reply (Hex Code 19)

This message is sent by the CMM in reply to the Request C120S Windows command.

Packet 1:	nn	99	aa	bb	cc	ee	ff	cc
Packet 2:	nn	99	gg	hh	ii	jj	kk	cc
Packet 3:	nn	99	ll	mm	nn	oo	pp	cc
Packet 4:	nn	19	qq	00	00	00	00	cc

Where: aa to qq are references to signature files nos. 01 to 0x10. Any unprogrammed windows will return a zero.

As an example if the only non-zero results were in packet 1 e.g.

Nn	98	01	02	01	03	04	cc

Then it would mean that window 1 and 3 refer to signature table 1, window 2 to signature table 2, window 4 to signature table 3 and window 5 to signature table 4. The signature table numbers will usually but not necessarily refer to the order in which the tables were sent.

9.7 Program C120S Windows Command (Hex Code 1A)

This command is sent by the payphone to request that the signature tables, previously sent to the CMM and held in battery backed RAM should be used to program the C120S coin acceptor. This coin acceptor is automatically programmed with the tables when they are downloaded from the payphone but they are also held in the CMM memory. This command allows a new coin acceptor to be fitted to the unit and to be programmed with the original coinspec without the requirement to download remotely. Note that this reprogramming of a

new coin acceptor will not happen automatically, as is the case with the C120P. The first time that the CMM Status (Hex code 13) is returned to the payphone after a new coin acceptor is fitted the CMM unit will indicate that fact to the payphone. The payphone can then make the decision to use this new command or carry out a complete remote download of the signature tables.

Note that while programming the coin acceptor the CMM will issue CMM busy due to Signature File Download messages (See section 9.2) and will signal completion of the process with a parameter update status message.

9.8 Signature Table Checksum

It should be noted that whether or not the signature table is sent as one big file or several individual ones the CMM unit will store these tables in RAM so that it can implement the Program C120S Windows Command. Even if this facility is not required it is possible that the RAM check on this area may at some stage fail, necessitating a signature file download to clear the checksum fail flag and thus remove the inhibit from the coin acceptor. It should be noted that unlike the C120P, the signature files for the C120S are only checked upon receipt of a Final Cost of Call message and before the Cashing Complete message is returned by the CMM. The reason for this is that the checksums for the larger tables required for the C120S take longer to calculate and this delay would be unacceptable at the off-hook stage. Any checksum failures will be flagged at the next off-hook – indicating that a download is required before normal operation of the payphone can be resumed. If the information in the CMM RAM is not relevant to the system then all that is required to clear the problem is to send a signature file download with the following data:

AA	55	00	00	00	00	AA	55	yy	zz	FF	FF

Note that the values of yy and zz are as given in section 9.4

10. Appendix 5: Security – Anti Tamper Feature

10.1 Detecting a Fraud

The Anti-Tamper feature makes it difficult for fraudulent users to obtain free calls. The usual attempts at fraud involve the use of coins with a thread attached – trying to pull the coin back through the coin acceptor once the coin has been credited by the CMM. Although the coin is identified by the coin acceptor it is not actually added to the available credit until it has been detected in the escrow pocket at which point the escrow will start to move forwards ready to accept the next coin. The first means of detecting a fraud is thus to detect a motor jam as the coin is removed from the pocket.

There is a further means of attempting fraud which is to hold the coin on its thread until the cashing gate has closed thus preventing it being routed to the cashbox. This action can be detected by ensuring that coins do not remain too long covering the cashing sensor and also to ensure that coins in pockets pass the exit sensor and empty pockets do not contain coins.

10.2 Anti-Tamper Action

When the CMM detects one of the above situations it sets a value in a special memory variable. This variable is checked upon completion of each pocket movement and if the tamper action is detected it immediately rotates the escrow for a complete revolution, routing all the coins to the cashbox. The cashing gate is held open and all further communications with the CMM are locked out until the phone goes back on hook and the power is removed. Once the power returns (i.e. the phone goes back off hook) the CMM executes an affective soft reset and the unit operates normally except for the fact that the cashing gate is held open until the CMM receives a final cost of call message. Note that then escrow must contain at least one coin at the time of the final cost of call message for the anti-tamper action to be terminated and the cash gate released. The difference between a soft reset due to anti-tamper and a normal soft-reset is that in the former case, store recovery and motor errors are not cleared because doing so may assist the fraudulent user.

10.3 Hard Reset & Anti-Tamper

If a hard reset is applied to the CMM, the state of each escrow pocket cannot be ascertained with any certainty. If a subsequent coin exits from the pocket when it was not expected then a spurious anti-tamper action can result. In order to eliminate this situation the anti-tamper feature is disabled for the first two cashing transactions after a hard reset command is received from the payphone.

11. Appendix 6: Example *.cdt and *.cst files for C120P & C120S

11.1 Example *.cdt file for both C120P and C120S coin acceptors

The following example coin data table file shows the format of a typical *.cdt file for use with both C120P (12 coin slots) and C120S (16 coin slots) coin acceptors. In the following example coin 2 is inhibited. Note that all unused slots must be filled with FF (see slot 11).

```
                              * Slot Number 0
                              * GREAT BRITAIN 10 PENCE 1992 -
01                            * coin code
00 01 02 00 00 00             * Coin Part Code
00                            * channel inhibit
0A 00 00                      * Coin Value
2B 00                         * Coin Volume
02                            * ID Tone
02                            * No of Pulses
00                            * Padding

                              * Slot Number 1
                              * GB 20 PENCE 1982 -
02                            * coin code
00 01 02 00 00 00             * Coin Part Code
01                            * channel inhibit
14 00 00                      * Coin Value
25 00                         * Coin Volume
02                            * ID Tone
02                            * No of Pulses
00                            * Padding

                              * Slot Number 2
                              * GREAT BRITAIN 50 PENCE 1997 -
03                            * coin code
00 01 02 00 00 00             * Coin Part Code
00                            * channel inhibit
32 00 00                      * Coin Value
1F 00                         * Coin Volume
02                            * ID Tone
02                            * No of Pulses
00                            * Padding

Slots 3 to 10 not shown

                              * Slot Number 11
                              * Blank Coin 12
0C                            * coin code
FF FF FF FF FF FF             * Coin Part Code
01                            * channel inhibit
FF FF FF                      * Coin Value
FF FF                         * Coin Volume
FF                            * ID Tone
FF                            * No of Pulses
00                            * Padding
```

11.2 Example *.cst file for C120P coin acceptors

The following example coin signature table file shows the format of a typical *.cst file for use with C120P coin acceptors. Refer to section 11.3 for example C120S *.cst file.

```
01 * Coin Number GB00010B      "10 pence New  " * Coin Identifier
01
08
6A * Block Length
01 * Coin Type

05 43 31 32 30 50 08 09 00 01 02 03 04 47 42 30
31 30 42 04 02 DC 01 06 36 87 CA 86 CF 87 62 88
06 89 C7 89 07 06 02 32 01 0D 91 22 2E 20 CB 1D
1D 1C 1C 1B A1 19 55 18 68 17 4A 17 56 15 10 16
D4 14 30 13 0A 14 03 90 01 0B 1E 9F AB 9F EE 9F
CD 9F CC A0 35 A7 F6 A5 07 A8 08 A9 68 A9 11 AB
10 0C 30 00 00 00 00 00 2B 3B FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF

FF

02 * Coin Number GB00020A      "20 pence  " * Coin Identifier
01
08
6A * Block Length
02 * Coin Type

05 43 31 32 30 50 08 09 00 01 02 03 04 47 42 30
32 30 41 04 02 DC 01 06 6B 9B 9D 9B C2 9B 6D 9C
B8 9C 4E 9D 09 06 02 32 01 0D 57 85 94 85 D1 85
76 86 8D 86 BD 86 C2 86 E1 86 84 86 87 86 DB 86
05 87 BA 86 05 06 03 90 01 0B 8B 99 07 9A E4 99
FE 99 F4 9A 16 A1 D9 A0 67 A2 D0 A2 B7 A3 CC A3
0D 09 30 00 00 00 00 00 A6 1A FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF

FF

03 * Coin Number GB00050B      "50 pence New  " * Coin Identifier
01
08
70 * Block Length
03 * Coin Type

05 43 31 32 30 50 08 08 00 01 02 03 04 47 42 30
35 30 42 04 02 DC 01 06 31 88 19 88 09 89 B1 89
78 8A 10 8B 06 06 02 32 01 0D 02 75 0E 74 1B 73
FD 71 AA 70 A0 6F 4A 6E 58 6D 6A 6C 31 6B CD 69
EB 68 9A 66 0D 0B 02 B4 01 0E 7D C0 C0 C0 AE C0
83 C2 38 C3 6B C4 DE C5 2A C7 A0 C9 31 CC 7D CA
92 CE 4F D1 67 D1 10 0B 30 00 00 00 00 00 84 F0
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF

FF
```

11.3 Example *.cst file for C120S coin acceptors

The following example coin signature table file shows the format of a typical *.cst file for use with C120s coin acceptors. Refer to section 11.2 for example C120P *.cst file.

```
nn 87 08 00 00 00 00 cs (Start Extended Sig. File Download)

nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
AA 55 FF FF 00 00       (Clear All Orbit Signature Tables command)
AA 55 10 00 8B 03       (Start Sig. Table data – 0x38B Bytes long,
                        example assigned to window 5 (0x0010))
43 20 53 63 6F 72 70 69 6F 6E 53 74 64 20 20 20 20 20 30 30 30 30 31 1A 00 02 EF 91 A7
B1 C5 17 49 43 BB A2 BF 0D 07 E0 80 B8 3B E9 5C 45 92 CF AB 05 EC 08 69 4E B6 ED 59 2C
8C 25 77 68 61 52 A8 17 CE 6E B6 2E AC 10 68 06 30 9A 96 B5 9B 1A 44 74 19 BA C7 FA 81
74 5D F5 ED 5A FB 63 96 BA C9 28 10 1A B4 28 38 E7 8C 63 CE F1 A8 D9 E2 2C 73 C5 C6 9F
1A C0 55 EE 85 F2 3B 29 C0 3E 75 08 0C BE D3 26 D5 D9 FF 95 FE 74 9C F8 C0 6F 3E 60 51
1B 2C 21 67 E9 07 6E 11 7B B0 10 2A 78 38 F9 3C D2 64 E2 01 21 CB 82 3F 54 8C 38 20 8D
DC 5F DD 69 C8 B9 41 C8 A8 8E 54 DB 40 AE EB 69 6B BE E1 16 35 5C 02 87 EF B4 48 C7 A9
03 8B CB 54 21 8D 9C C1 E6 A2 0B 15 DB 9F DE 23 33 79 A1 1D 82 B4 71 D9 B0 3C 00 70 80
CC CC 54 D0 E5 7C 21
cs                      (Packet Checksum)

nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
9C 88 A9 96 9A 94 7A 1B 99 B8 7C 3C 0F 81 F2 BF 77 9A 79 05 E8 6E BB 5C E6 72 A7 51 8D
72 98 A2 6B 6F 0D B6 5F FF 8E BC F6 18 A2 7B 31 EE E2 40 49 D0 E2 6B 40 73 60 2E A3 18
4C 2E DC 71 E1 91 E3 3B 6D 16 15 37 7B C7 EB B6 4A 14 0F 99 0B 61 02 AF B4 4C 00 48 B7
4E 70 FF 61 1E 9E AF 0F 13 A5 73 3D 20 C8 B0 7E F8 50 E4 35 A0 CE 84 5F D9 57 75 0E 58
83 13 C7 15 5C 04 54 10 9A 68 1A 99 87 66 ED 63 40 E9 29 03 3B 33 90 8F DB B2 D0 35 E2
1E 0F 5D 6B BE DB 9F 25 DE 18 32 4E 68 56 B8 78 C9 C5 83 F1 00 C5 ED 86 0B CC 9F 10 91
6C 87 54 7A D1 AB 70 CF DE A0 E5 4B 6F 0A 5B 7C 33 8F 82 11 D1 36 EF 1E 7E 0D 10 A0 AE
C3 60 6E D4 0A CC FB F4 E3 A9 60 08 69 42 40 B4 4B 90 21 34 49 6C 81 BE CA 39 CE D3 B8
4B 5A 4A 3D DA 23 58 B9 08 0D E6 DC C8 6B 4B 91 54 4F D4
cs                      (Packet Checksum)

nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
EA D5 D0 90 21 AA BD 25 D3 EC DA 47 8A ED 24 D3 29 16 BF B5 E9 FF AC 8E 8F 40 18 45 FF
EA 2D C0 55 75 AA B6 29 25 C3 A6 AA F1 FE C8 20 53 25 0B D4 CF D2 D9 FE 40 20 C4 61 C0
12 6F 33 68 C9 47 31 6B C6 C5 B3 23 AB 84 51 53 25 92 CA 5B 3C A3 7B 0B 6E 0C CC 3B 56
92 59 F6 21 8B 3E DA 72 CB 4C EC 7B 5C F9 87 92 CA 68 C7 A8 09 5A 28 02 80 42 FF 36 95
8A B0 F6 09 EB 1F 9D E6 F8 74 59 8E BD 4F 4D 5D 49 5E 7A DA 65 4D 8C 93 7F D6 C0 87 26
CF 2C 8F F8 FB 36 00 3B 1C BE E0 D0 DE 00 AE 89 B8 15 A9 D3 B0 1E 2C 53 0C 51 E3 E9 0B
29 AB B9 3A AD 58 56 A0 F9 8A 92 C8 4B 27 FF 1A 6F 91 B2 01 58 32 02 EA A3 92 0A C2 65
1A 03 3A DB 5B 35 AC FB F2 27 6E AC 88 AA 32 ED 25 9D C4 87 97 55 F9 6F EA 17 F2 39 98
5C 94 75 DB 56 38 E7 92 5E D1 50 55 93 1C 0B 2D E0 21 47
cs                      (Packet Checksum)

nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
3F F8 32 7D 09 6A 79 75 7A 8A 8C 62 86 1A 88 0E 1E E9 22 F0 28 DC 99 B4 75 83 DD 20 8D
33 26 AC 61 DE E2 3F 4F 63 FE 18 2F 85 1F 8F 63 85 0E 55 56 A5 E9 E4 9C B1 43 32 B6 DD
F2 D9 93 80 A4 B9 1B 24 CA 1D CE 25 5B 15 2B 7C 74 60 F6 B0 A5 9A F5 82 2D B7 F9 A2 A4
FF 65 8D C3 2C EB 51 36 E0 B9 C5 FB CE D5 B7 AA 90 93 79 50 D0 C0 55 06 C8 59 B6 34 AA
0C 77 54 08 CD EC 10 61 E7 62 D8 59 4E 05 63 70 4D 77 FB 3E 09 58 CC CA 0A 66 01 D4 41
F3 A7 12 6A BF 0D 51 D2 B0 A9 90 E4 33 25 6D 2C 28 26 34 CC 04 AA 55 02 00 8B 03 43 20
53 63 6F 72 70 69 6F 6E 53 74 64 20 20 20 20 20 30 30 30 30 31 1A 00 02 38 4E 64 A5 1F
30 F6 DE 2A 57 91 E4 F3 6A 01 BB 42 1B 6C 12 FB 15 0F F7 C9 09 DA BB EB 0C 05 29 9F AD
0B 2E 04 89 01 B4 A6 37 ED C8 06 B4 5D C7 EE 04 7E 72 5D
cs                      (Packet Checksum)

nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
2E 7D DF D6 CC 18 59 6D 38 0A 4B 19 2D 0A CE B9 62 9B B3 23 AA A9 38 CB 5C B1 31 B8 C6
6F 95 7B A1 FC 8E 15 70 50 1D 02 83 EF 74 56 B5 E9 7F A5 3F 1D E7 21 30 84 0F 58 D4 06
8B D7 99 E5 81 5B C4 1E EC 9B CA 01 4E 3E 0E 34 99 78 80 D7 B2 E0 0F 15 A2 0E 88 3C F7
57 77 1A B2 71 10 32 AA 23 D5 36 90 EB 7A DA 9A 6F B8 C5 8C 94 C0 E1 FB 32 D2 71 EA 40
63 31 95 57 E3 78 B0 B9 E8 42 05 E9 67 E8 C8 4C 5E 68 47 3F AD 43 36 2C 0F 00 DC 3F F3
2C 15 74 39 5F C8 38 A2 B5 A5 26 A5 DE 86 39 EE 00 CF 43 A0 FC 2F 82 A6 41 01 BA 5A 2E
97 B9 1C 76 54 A3 1C A6 69 DA 9B 9C ED D7 BF 23 6F 22 86 E8 5E 8C BB 38 50 12 B4 F8 1A
D0 1C 18 4A 2B 06 93 CB 68 5C 78 E4 78 CF 98 85 DA 0D C1 F1 BA AD 12 05 5C 26 5D 98 C0
2B 2D D3 1F 12 B2 13 DC 7F 98 47 F8 2B DC 44 E9 5C 0F C8
cs                      (Packet Checksum)
```

```
nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
2D 6C 00 3E 14 79 0D 68 2C 21 48 F6 55 9D 03 16 FF 04 6E 05 17 80 15 84 26 5A 59 E6 A1
59 FB 99 50 BA D5 C0 CE 0C E4 23 A9 2C 74 8D B8 C7 43 D7 9A EF F6 67 89 37 06 0D 10 9A
15 EE 41 72 F1 60 BC C1 72 7B FE C2 EC 27 44 24 DB E1 8A 9E EB 2A DC B3 37 A8 E6 12 84
C2 16 DF CC 63 7F 84 AB CD BB 87 5F C3 1B 68 F4 30 98 1B B5 C2 EC FE E2 78 2A 18 42 5A
0F D0 F2 12 64 89 4D 12 5B 60 B0 72 8E F3 34 6F 92 56 4D 8F F6 08 56 EE 5D 83 39 37 F2
DA 52 24 D5 13 B5 7A 35 34 E4 09 69 6F C9 F4 A4 F6 91 62 BB 48 77 4A DA AF 72 FA 3F AB
E3 E3 0F 86 CE BF 7A 0B 70 95 82 6A 72 4A C9 B0 A9 AA A3 FF 2D C8 54 C9 02 35 AC B4 A9
3F 10 08 71 D8 AF 04 C4 0C 5A F2 51 0C 85 B1 EE EC C8 AB 61 D4 54 49 BB F2 16 CF E8 5F
82 DB 1C F3 8E 95 64 56 62 C8 A2 8F B6 DC 32 13 FB 1E 57
cs                      (Packet Checksum)

nn 87 FB                (Start Extended Packet – 0xFB or 251 Bytes Long)
FE A1 A4 B2 5B 32 6A 82 D6 95 00 A0 D0 3F CC A6 E2 B3 BD DD FA 47 A7 39 FE 42 56 7C EF
53 53 57 52 88 5E 69 59 13 E3 E4 5F 2A 30 BE C9 33 CB 23 3E A1 CD DF 0E 87 2F 38 74 28
40 74 22 6E 45 EA F4 B3 C7 D4 14 43 55 78 FA 42 0A 75 AE 4E 2B 96 94 68 7E BD 24 F2 D2
C9 7F D9 5B A4 6C 00 96 8C 80 FA A8 A4 82 70 99 46 01 CD F6 56 CD F4 BE FC 89 9B AA A1
6A 53 BB 56 27 F1 E1 65 95 E1 5B E5 85 7C 0E 26 D7 7A 1D A9 15 CE B9 A4 97 2B 94 CA 24
C7 42 0B B8 D7 11 10 BE 4C 5F 82 9A 41 5D 81 22 A3 98 A1 C6 BD 43 58 BA 08 BD 0A 77 26
AD C2 60 37 48 E3 0D 09 AB 87 B7 F8 CF D3 58 20 C6 4B E1 C7 E3 58 A1 43 95 55 69 3A 39
E7 6D 93 8C 6D 77 63 F6 24 20 E9 6F 05 49 6D 47 6C C9 56 C5 0A DF 02 58 22 6C A9 66 BD
A0 D4 AD A1 28 E9 7F 59 23 2F 0B 65 94 0D 78 6E 31 A1 B3
cs                      (Packet Checksum)

nn 07 51                (Start Last Extended Packet – 0x51 or 81 Bytes Long – padded to
                        0xFB Bytes)
EA AD 2A E4 64 DE 0B F4 6F 95 01 72 96 ED 92 EA DC 1D 15 C3 33 3B 44 18 A6 18 76 96 53
40 18 9C CC DD BE 4F 02 D6 AC B2 82 59 00 75 D0 FC A9 84 DA 6F 43 94 34 EC EF 25 F9 CC
D3 39 0B 4A A9 45 33 F9 34 8C 84 79 3D 2C 92 AA 04
AA 55 03 02 FF FF       (Termination Header With Sig File ID (03) & Vers (02) See Note)
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
cs                      (Packet Checksum)

nn 17 AA 00 00 00 00 cs (Reply from Orbit – File being downloaded to validator – Sent
                        every 20 seconds)
nn 17 00 00 00 00 00 cs (Reply from Orbit – Download to validator Complete)
nn 08 00 08 00 00 00 cs (Reply from Orbit – Parameter Update Message Download OK)
```

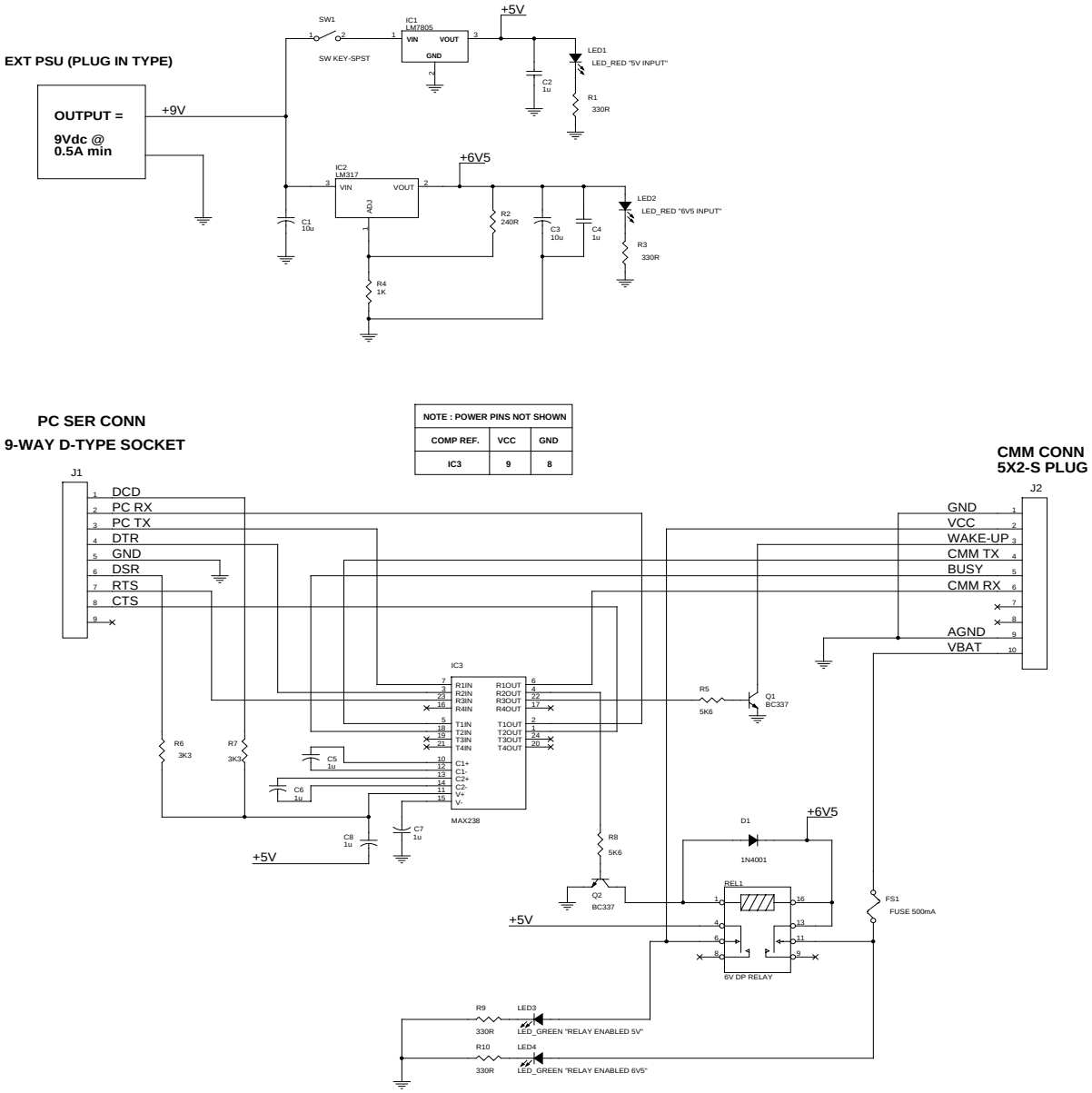
Note. Either the Signature File ID Byte or the Version no. must be non-zero in order to force a download of the file to the validator. If both the Signature File ID or Version Number byte are zero, the data will only be stored in the CMM RAM.

12. Appendix 7: Orbit CMM/PC Interface Module Schematic

12.1 Circuit Schematic

The circuit below shows how Orbit can be interfaced to the 9 pin serial port of a PC (COM1 or COM2). The circuit uses a Maxim MAX238 Quad RS232/CMOS Driver/Receiver which operates off a single 5V supply. The PC controlled relay switches separate 5V0 and 6V5 supplies to provide Orbit Vss and Vbat supply voltages.

Table 21: Schematic



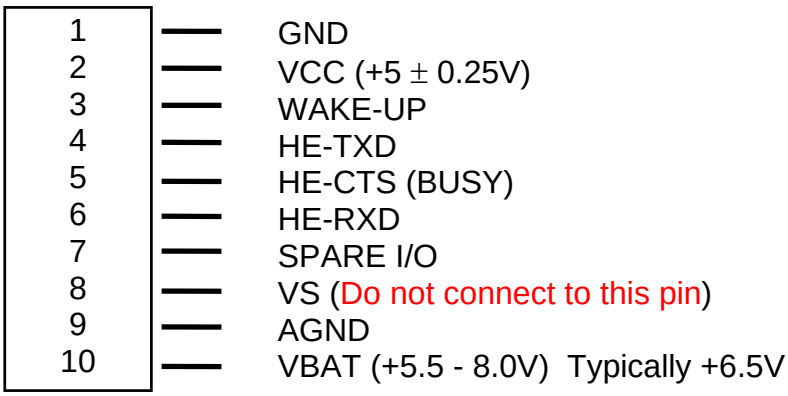
12.2 Bill of materials

Reference	Quantity	Part
C1, C3	2	Capacitor 10u 16V elect
C2, C4, C5, C6, C7, C8	6	Capacitor 1u0 16V elect
D1	1	Diode 1N4001
FS1	1	Fuse 500mA Quick Blow
IC1	1	Regulator LM7805
IC2	1	Regulator LM317
IC3	1	IC MAX238 Quad RS232 / CMOS driver
J1	1	Connector 9-way D-type socket
J2	1	Connector 10- way (5x2) 0.1” pitch header
LED1, LED2	2	RED LED
LED3, LED4	2	GREEN LED
Q1, Q2	2	Transistor BC337
REL1	1	Relay BT47W/6
R1, R3, R9, R10	4	Resistor 330R
R2	1	Resistor 240R
R4	1	Resistor 1k0
R5, R8	2	Resistor 5k6
R6, R7	2	Resistor 3k3
SW1	1	Toggle switch SPST

13. Appendix 8: Hardware Requirements.

Figure 6: CMM Pin Connections

CMM Pin Connections



13.1 Description of signal connections

Pin 1

Pin name: GND
Status: Power input
Signal name: Digital ground / Digital supply 0V
Signal: 0Vdc
Comments: Vcc (pin 2) ground / 0V connection. Must be commoned externally with AGND (pin 9).

Pin 2

Pin name: Vcc
Status: Power input
Signal name: Digital positive dc supply input
Signal: Switched +5V0 ±0.25Vdc regulated dc supply. 100mA max, 20mA typ. 3mA idle.
Comments: This regulated voltage supplies power to the Orbit controller and coin acceptor hardware.

Pin 3

Pin name: Wake-up
Status: Input
Signal name: CMM wake-up input
Signal: Open collector, switching to GND. Input has internal 470k pullup resistor.
Comments: Active HIGH

Pin 4

Pin name: HE-TxD
Status: Output
Signal name: Host Equipment – Transmit Data output
Signal: 5V0 CMOS

Pin 5	*Pin name:*	HE-CTS
	Status:	Output
	Signal name:	Host Equipment – Clear To Send output
	Signal:	5V0 CMOS
	Comments:	Active HIGH. Also referred to as CMM Busy.
Pin 6	*Pin name:*	HE-RxD
	Status:	Input
	Signal name:	Host Equipment – Receive Data input
	Signal:	Open collector, switching to GND. Input has internal 470k pullup resistor.
Pin 7	*Pin name:*	Spare I/O
	Status:	Spare input / output pin for future use.
	Signal name:	Spare I/O
	Signal:	Input has internal 470k pulldown resistor.
	Comments:	For future use. **Do not connect any external signal to this pin.**
Pin 8	*Pin name:*	Vs
	Status:	Power input
	Signal name:	RAM backup dc supply voltage
	Signal:	+2.5 – 5.0 Vdc
	Comments:	**Do not connect any external voltage to this pin.** This voltage (2.5-5Vdc) is only required if the Orbit internal SRAM is not battery backed. Note that all Orbit CMM's are supplied with battery backed RAM (internal 3V0 Lithium cell).
Pin 9	*Pin name:*	AGND
	Status:	Power input
	Signal name:	Analogue ground / battery supply 0Vdc
	Signal:	0Vdc
	Comments:	Vbat (pin 10) ground / 0V connection. Must be commoned externally with GND (pin 1)
Pin 10	*Pin name:*	Vbat
	Status:	Power input
	Signal name:	Analogue positive dc supply input / battery positive dc supply input
	Signal:	+5.5 – 8.0V dc battery supply. 200mA max, 60mA/200ms typ, 0mA idle.
	Comments:	Connection to 6Vdc re-chargeable lead acid battery providing power to the motor, solenoid and opto sensor circuits. For optimum performance, this supply voltage should be typically 6.5V.

This manual is intended only to assist the reader in the use of this product and therefore Money Controls shall not be liable for any loss or damage whatsoever arising form the use of any information or particulars in, or any incorrect use of the product. Money Controls reserve the right to change product specifications on any item without prior notice