V.L.C.

Gaming Machine
COMMUNICATION PROTOCOL
for
South Australia
Gaming Machine Monitoring System

TECHNICAL SPECIFICATIONS

Protocol VERSION E2

2311 South Seventh Ave.
Bozeman, MT 59715
(406) 585-6600

CONFIDENTIAL & PROPRIETARY

This document contains specifications on a version of a Gaming Machine Communications Protocol that is proprietary to Video Lottery Consultants, Inc (VLC).

It describes the protocol necessary for Gaming Machines to communicate with a central Monitoring System supplied to South Australia by VLC.

License for its use is limited to the above South Australia Gaming Machine Monitoring System by companies approved by South Australia.

The protocol is confidential and strictly limited to those companies with a need to know and who have entered into VLC's copyright and patent nondisclosure agreement for adapting the above companies' Gaming Machines for compatibility with the South Australia Gaming Machine Monitoring System.

TABLE OF CONTENTS

1 INTRODUCTION

This document defines the communications protocol by which the South Australia Gaming Machine Monitoring System (GMMS or host) communicates with Gaming Machines (GMs or terminals). The host communicates to each location (site or venue) using the public telephone network in dial-up mode; however, the protocol alternatively supports communication through an on-line dedicated telephone network. To facilitate a single local modem servicing multiple Gaming Machines, an enhanced EIA RS-422 communication bus is utilized to link all Gaming Machines within each location.

1.1 COMMUNICATION OVERVIEW

In the South Australia system, telecommunication data transfer between the host and the modem within each site will take place at the fixed rate of 2400 baud. On the other hand, Gaming Machines are expected to be capable of communicating on the GM Communication Bus at a rate of 2400 or 9600 baud (with 9600 baud the recommended default rate), and to include a means of making that rate selection in a terminal set-up feature. Every GM has a unique communication address and each must listen to the enhanced RS422 communications bus at all times and process only packets addressed to it. When the host sends out a request, it expects a reply only from the terminal addressed and only when permitted by protocol standards.

The GMMS uses addressed Poll commands to permit (or request) the GM to transmit information. Three classes of information may be sent to or from the GMs: GM status reports, data read/write packets, and transactions. This information transfer provides the GMMS with the capability to monitor and control the status of each Gaming Machine in the Gaming Machine Monitoring System. Where appropriate, ACK/NAK commands are used to determine transmission success.

Each Gaming Machine must be capable of being a modem master and each must have a local means of establishing whether or not it is the modem master. The modem is typically housed within the master terminal and only one modem master is established in each location. All other terminals at the location are considered slaves and share the dial-up modem via the communication bus. The modem master must provide all necessary control of the modem and no other terminals on the communication bus should attempt to send any modem setup or control commands. A private dial-up telephone line will be installed to service each location, so the modem master is expected to answer all incoming calls.

After establishing the connection, the GMMS will send request packets. These packets will be seen by all Gaming Machines connected to the communication bus. The GM that recognizes its terminal address (or terminal ID) in the request packet is expected, when permitted by protocol, to activate its data transmit driver, send its reply packet via the bus, and then deactivate its transmit driver. When the GMMS has finished communicating, it will disconnect. When this occurs, the modem will return its Data Carrier Detect (DCD) line to the inactive state. The modem master is expected to send setup and control commands to the modem only while DCD is inactive.

Each approved Gaming Machine manufacturer is expected to provide the communication bus interface (as specified in Appendix B) in all Gaming Machines and is also expected to supply a system-compatible

modem (as specified in Appendix C) when one of its GMs is established as the modem master within a location.

1.2 COMMUNICATION TIMING SPECIFICATIONS

The packetized communication is asynchronous. However, if the host or GM communications handler is in the middle of receiving a packet and it encounters a pause in the data greater than 50 milliseconds, it will abort that packet and begin checking for a new one.

If the GM has any packets to send in response to a Poll command from the GMMS, the GM is expected to begin transmission within eight (8) milliseconds of the last bit of the Poll command. Also, after a terminal receives a request (other than a Poll), the GMMS expects to receive (in response to a subsequent Poll) a reply within 5 seconds; otherwise, the GMMS assumes a timeout error. If the GMMS does not receive a response within the above times or if a negative acknowledge response is received from the terminal, it may retry one or more times before taking other error-handling actions.

When responding to a Poll command, GMs must assert the transmit control line to enable the transmit driver on the communication bus. This control signal must be negated and the driver tri-stated within 8 milliseconds of the last stop bit of the last character transmitted.

1.3 STANDARD DATA FORMATS

Throughout the protocol document, a '$' preceding a value indicates that the value is being expressed in hexadecimal. Following is a description of the data formats used in this document:

Date(U) Universal Date: Julian number of days since March 1, 1900. March 1, 1900 is day number one (1). Date(U) is a two byte binary field.

Time(U) Universal Time: Seconds into the current day divided by two (2). Time(U) is a two byte binary field.

Note: Whenever both the date and time for a specific event are exchanged within the protocol, the Universal date and time are used. When just the time is transferred, it is Relative time.

Time(R) Relative Time is defined as the local time at the GM in seconds divided by two (2). Universal Time is adjusted to local time by using the Time Zone Offset code and the Daylight Savings Time code maintained for each GM.

BinaryX Binary data field where X indicates the length in bytes for the binary field.

HexX Binary field whose value is expressed in hexadecimal. X indicates the length in bytes of the Hex field.

BCDX Binary field whose value is expressed in Binary Coded Decimal. X indicates the length in bytes of the BCD field.

FlagsX — Bit fields used as flags. Bit zero (0) is the least significant bit. When a bit value is zero (0), it indicates OFF; when its value is one (1), it indicates ON. X indicates the length in bytes of the flag field.

CodeX — A Code field contains a combination of multi-bit fields and bit flags. X indicates the length in bytes of the Code field.

ASCII(X) — Standard ASCII text characters with no parity (bit 7 = 0). X indicates the length of the ASCII field in bytes.

Reserved — Indicates that the field is not currently being used; however, the specified number of bytes must be maintained in the packet. GMs should set all Reserved bytes to zero (0) before transmission and ignore values received in Reserved fields.

BYTE ORDERING

All packets are structured using the Motorola byte ordering scheme, often referred to as the "Big Endian" byte order. Multi-byte fields are transmitted with the most significant byte first. The following figure shows Motorola byte ordering for a 4-byte field:

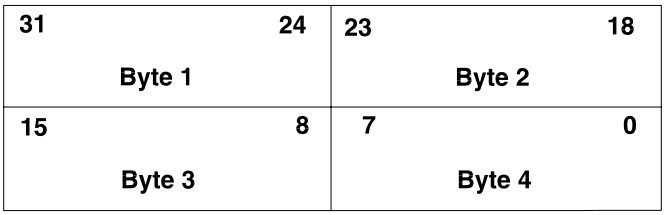


Figure 1

The order of byte transmission is Byte1, Byte2, Byte3, and Byte4, respectively.

2 BASIC PACKET STRUCTURE

All communications are packetized and the basic packet is eight (8) bytes in length. Each packet contains several segments, including a Header code, a Terminal ID (address), a Command, a Cyclic Redundancy Check (CRC) code, and a Trailer code. Following is a figure that shows the segments of a basic communication packet:

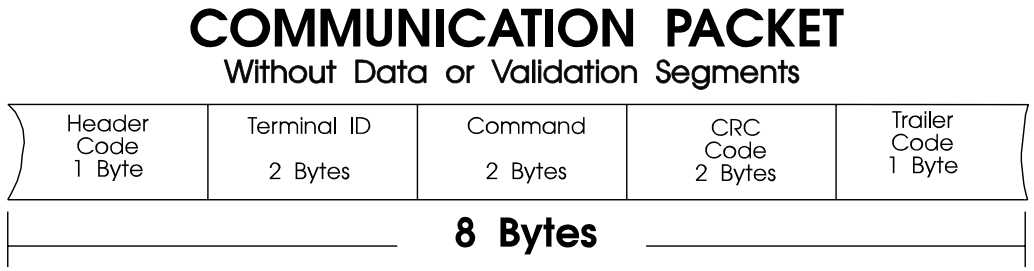


Figure 2

If a packet requires data to be transferred, preceding the CRC will be a Data Segment and a Validation Segment. The Data Segment has up to 256 bytes of data in multiples of eight (8) bytes. The Validation Segment contains eight (8) bytes of packet validation information calculated by the data source.

The following figure shows the communication packet structure when the data and validation segments are included:

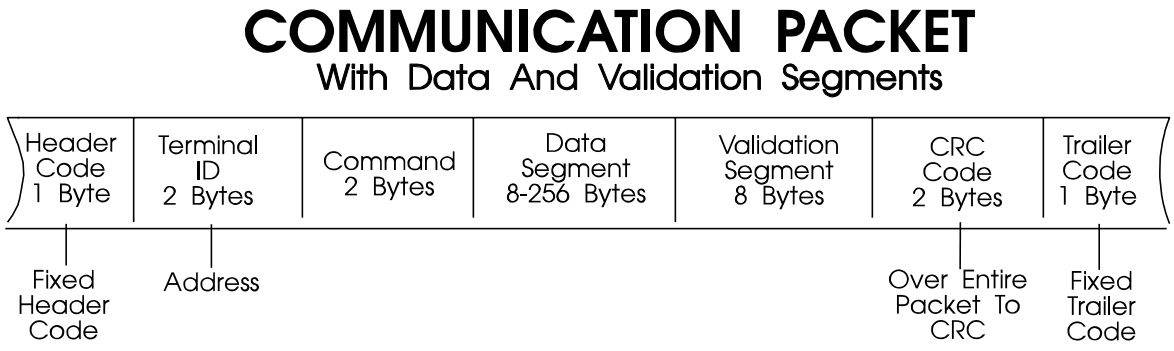


Figure 3

The maximum size of an individual packet will therefore be 272 bytes. Each of the communication packet segment types is described in the following subsections.

2.1 HEADER CODE

The fixed Header Code value is $05 (0000 0101) and the segment is one byte in length.

2.2 TERMINAL ID

The two-byte Terminal ID is expressed as a 4 digit hexadecimal number. Thus the Terminal ID $13A5 will be 0001 0011 1010 0101 in base 2. Administrators of the South Australia system will assign a block of Terminal IDs to each approved terminal manufacturer.

Terminal IDs provide the unique system address of each terminal. Terminal IDs must appear on any tickets printed by the terminal. The Terminal ID must be set in the hardware (non-volatile memory) of the main logic board at the time of manufacture and must be implemented so that they cannot be readily altered in the field.

In addition to the unique address for each GM, all terminals are expected to consider the address $0000 as addressed to them. This will serve as the "broadcast" address.

2.3 COMMAND SEGMENT

The command segment of a packet is two bytes (16 bits) in length and contains several bit fields used to define the specific details of the commands. Invariably, the upper 3 bits of the command segment form a command type field (ccc). There are currently seven "Command Types" defined in the communication protocol:

Command Type (ccc)
000 - Transaction Request or Transaction Reply
001 - Data Write Request or Data Read Reply
011 - Site Info Reply
100 - ACK/NAK
101 - Data Read Request
110 - Poll or Status Response
111 - Site Info Request

Command types in which the upper bit is a one (1) have no data segments associated with them and utilize the basic 8-byte packet structure. Command types in which the upper bit is a zero (0) have at least one 8-byte data segment transmitted with them and utilize the packet structure with data and validation segments.

The definition of all bits in the 16-bit command segment for each command type is provided in Section 4, COMMAND TYPES.

2.4 DATA SEGMENT

Communication packets utilize the contents of data segments to upload and download information to and from the host. Three command types, Transaction Request/Transaction Reply, Data Write Request/Data Read Reply, and Site Info Reply include data segments. The "Data Types" defined within each of these command types are as follows:

Data Types for TRANSACTION REQUEST/TRANSACTION REPLY commands:
- CASHOUT REQUEST
- CASHOUT REPLY

Data Types for DATA WRITE REQUEST, DATA READ REPLY commands:
- CONTROL
- CONFIGURATION
- DATE AND TIME / MEMORY SIGNATURE
- BANNER / REPORT
- MONITOR (read only)
- STATISTICS (read only)
- EVENTS LOG (read only)

Data Types for SITE INFO REPLY commands:
- CONTROL (read only)
- CONFIGURATION (read only)
- DATE AND TIME / MEMORY SIGNATURE (read only)
- BANNER / REPORT (read only)
- MONITOR (read only)
- STATISTICS (read only)
- EVENTS LOG (read only)

The data segment will always include a multiple of eight (8) bytes of data with up to a maximum of 256 data bytes in a single packet. If the actual data is not a multiple of eight (8) bytes, the end of the data segment must be padded with zeros to fulfill this requirement. The length of the included data segment is specified within the command segment of the packet.

Section 5, DATA TYPES, contains detailed formats and descriptions of each of the data types.

2.5 VALIDATION SEGMENT

The Validation Segment is provided by the source of the packet whenever the data segment is present. The validation segment serves to assure and verify integrity of the data and is made up of the following 8 bytes:

Validation Segment Expanded

Date	Time	Random Validation Seed	Validation Code
2 Bytes	2 Bytes	2 Bytes	2 Bytes

Figure 4

The following method is used to generate Validation Codes for packets containing data:

A random number from 0 to 65,520 (that is, a random number modulo 65521) is generated by the source of the packet. That random number becomes the starting Validation Seed.

To calculate the resulting validation code, the random validation seed is multiplied by 65,536 (shifted left 16 bits) and logically or'd with the second and third bytes of the packet (always the Terminal ID), which results in a 32-bit number. Note that the header code is not included when calculating the validation code. The 32 bit number is then divided by 65521 (a prime number) and the quotient is discarded.

The remainder is then shifted left 16-bits and or'd with the next two bytes of the packet and divided again by 65521. This process is continued over all bytes in the packet through the end of the data segment. The resulting 16-bit remainder is the validation code. All math is unsigned. The validation segment of the packet is then composed of the date, time, validation seed, and validation code.

When the terminal receives a packet with data, it first calculates the validation code based on the validation seed. The result is then compared with the Validation code received in the Validation segment. If they do not match, the validation code fails and a validation code NAK is returned to the Host. If the validation code passes, the validation date and time stamp is checked. If the validation date and time differ from the current date and time by more than plus/minus 10 seconds, a validation time stamp NAK is returned to the Host. If the validation fails for any reason, the terminal does not process or accept the data segment of the packet.

2.6 CRC (CYCLIC REDUNDANCY CHECK)

The CRC segment is provided for detection of communication errors. The CRC polynomial used is CRC-CCITT (16,12,5,0). Consistent with all other multi-byte fields in the protocol, the CRC has the most significant byte first. To facilitate this, the data must be processed with the most significant bit first. The CRC shift register is initialized to hexadecimal FFFF. The CRC is computed from the start of the packet (including the header byte), after any encryption, through all bytes up to the CRC field. For example, the CRC over the bytes $05 B1 6A A0 C1 is $3D87.
As with all standard CRC processing, received data is checked by processing through the CRC field, with a non-zero result indicating an error. In this example, the CRC of $05 B1 6A A0 C1 3D 87 is 0 so it is to be assumed that no communication errors occurred.

2.7 TRAILER CODE

The one-byte fixed Trailer Code value is $0D (0000 1101).

3 COMMUNICATION FLOW - GENERAL

COMMANDS - General

Irrespective of the site configuration, the GMMS uses Poll commands to control bus transmissions from all Gaming Machines. Following a request from the GMMS, GMs are only allowed to transmit a response immediately following an addressed Poll command and only within a defined response window. The GM is not allowed to transmit packets at any other time. Specific site configurations will conduct a continuous poll of all GMs, while other configurations will only have polls conducted to solicit a response to a request. In either case, the action taken by the GM in response to a Poll is identical.

Flags within the Poll command provide control of the communication flow by both limiting the types of packets the terminal may transmit and, in some cases, by forcing a response from the GM. The Poll command also indicates that a transaction from the GM is recognized as in progress.

GMs transmit a Status Response to report the occurrence of an exception condition detected by the GM. When the Poll command forces a response and no other Status Response is pending, the Status Response provides a "no change" report.

The host uses Data Read Request commands to read data from terminals. The terminal is expected to reply with a Data Read Reply command to upload the data. If multiple packets of data are required in the Data Read Reply to answer a request, the host indicates acceptance of all but the last packet by sending ACKnowledge commands.

Site Info Requests are also used to read data from the GMs and, in almost exactly the same way as the Data Read Reply, the Site Info Reply uploads the data (the encryption key is different). In multiple-packet transfers, the GMMS acknowledges all but the last packet.

The host uses Data Write Request commands to send non-transaction data to the terminals. The terminal is expected to reply with an ACKnowledge command indicating acceptance of each packet.

Transactions provide the means for a GM to initiate the exchange of data between the terminal and the GMMS. The terminal starts a transaction by sending a Transaction Request command and the GMMS responds with data in a Transaction Reply command. The terminal does not acknowledge a Transaction Reply, but is informed of its transaction in progress through a bit in the Poll command.

Of the commands sent by the terminal, only the Data Read Reply, the Site Info Reply, and the Transaction Request commands contain "data". The other commands sent by the terminal (Status Response and ACK/NAK) follow the basic 8-byte packet structure with no data or validation segments included.

OPERATION - General

A GM is expected to transmit packets only immediately following a Poll command addressed specifically to it, but full-duplex communication will take place on the RS-422 bus. That is, the GMMS will at times send "non-poll" packets (Data Read/Write Requests, Site Info Requests, ACK/NAKs, and/or Transaction Replies) to another (different) terminal while the GM currently being polled is responding. For efficient communication flow, the GMMS may have multiple data-type transfer activities in process to a single terminal, but this will be limited by GMMS design to no more than one of any particular Command Type in process at any one time. The GM must therefore be capable of "simultaneously" (before all aspects of the data transfer is complete) processing a Data Read (or Write) Request, a Site Info Request, and a Transaction Reply. (For example, the GM may receive a Data Read Request and, before the Reply is transmitted, also receive a Site Info Request, and/or a Transaction Reply.)

In a similar manner, the GM is expected to have no more than one Transaction Request outstanding at any time (transactions are the only Command Type initiated by the GM).

The GM is also expected to queue up all enabled, unreported exception conditions. Exceptions are reported to the GMMS in the form of Status Responses and the terminal should be capable of storing a minimum of 32 exceptions to reasonably cover situations when communication has been interrupted and several exceptions accumulate; if full, excess exceptions may be discarded (see the Command Details of the Status Response Buffer Full exception for additional information on handling the queue overflow condition). The exceptions which must be queued and reported are determined by a flag field in the Configuration packet; this flag field should be defaulted to zero (0) in all GMs.

The GMMS is designed to prevent transmission of more than one data request of a particular command type (Data Read/Write Request, or Site Info Request) to a terminal until it receives some appropriate response from the GM or until it times out due to no response. Therefore, the GM need not and should not buffer up multiple requests of the same command type. That is, if the terminal is generating (or transmitting) a data response (Data Read Reply, Site Info Reply or ACK/NAK) of a particular command type (defined by ccc bits in Section 4) when it recognizes a new data request of the same command type addressed to it, the GM is expected to abort the ongoing response and begin processing a response based on the new request.

Also, the GMMS communication structure is such that the GMMS will not poll any terminal while a GM is transmitting a legitimate response. Therefore, if a poll to any terminal is received while a GM is transmitting, that terminal and all terminals are expected to terminate transmitting and allow communication to proceed based on the current polling sequence.

The Poll command includes two bits which determine the type of response that a Gaming Machine can or must send.

When the GMMS sends a "forced" Poll command (f bit set), the addressed terminal is required to send a Status Response; any other response will be considered invalid and ignored by the GMMS.

When the GMMS sends an "unforced" Poll command, the permission bit establishes whether or not the terminal is allowed to send packets that contain "data".

When the permission bit in the Poll command is not set by the GMMS, the terminal is only allowed to send packets that do not contain data. That is: Status Response (reporting change in condition) or ACK/NAK.

When the Poll command permits packets which include data to be sent, the GM is allowed to send any pending response, with data packets having priority. This includes:

Priority 1: Transaction Request, Data Read Reply, or Site Info Reply.

Priority 2: ACK/NAK, or Status Response (condition change).

Within each of these two priority categories, there is to be no particular priority for a specific command type. **Rather, the Gaming Machine should transmit responses in a chronological order (first-processed, first-transmitted).**

In response to "unforced" Polls, the Gaming Machine is expected to send no response when there are no data-related packets pending and no change in exception status has occurred.

IMPLEMENTATION - General

The communications handler within a Gaming Machine can be implemented in a number of different ways. One possible means of implementation is to create separate response (transmission) queues, one for data commands and another for non-data commands which are to be transmitted to the GMMS.

The data command queue could include one Data Read Reply, one Site Info Reply, and one Transaction Request, each containing up to 256 bytes of data (272 total bytes in the packet).

The non-data command queue could include two ACK/NAKs (none for transactions), and a minimum of thirty-two exception Status Responses. (Note, since ACK/NAKs identify the command type within the packet, no ambiguities result.)

SYSTEM CHARACTERISTICS - General

Data transferred through the use of the Data Write Request, Data Read Reply, and Site Info Reply commands serves to configure and control the GM as well as monitor the accounting, security, and performance activity.

In some cases, particularly in GMs with multiple games, the data involved to fulfill a particular request or reply (Data Write Request, Data Read Reply, and Site Info Reply only) may exceed 256 data bytes. In these cases, the data will be divided into packets sent in multiple transmissions, termed continuation packets. Provisions are supplied in the protocol to identify and track continuation packets in these multi-packet transmissions.

GMs may be required to perform a transaction (Cashout) with the GMMS before printing a cash ticket or executing a cancel credit operation (manual pay). As a result of these transactions, ticket validation numbers, which must be printed on each cash ticket, will be provided by the GMMS.

In certain instances to confirm that qualified programs are installed, the GMMS will request the GM to perform a Memory Signature and report the result of the memory signature calculation to the GMMS. In some circumstances, the GM is allowed and expected to execute the memory signature

calculation with game play enabled, while in other cases the GM may be forced by the GMMS to remain disabled until the memory signature is verified.

In special error situations, the GMMS may issue a broadcast Poll command (Terminal ID of zero) on the local RS-422 bus to clear all GM transmissions. A GM must be capable of recognizing a broadcast Poll command whether transmitting or not and immediately stop any transmission in progress and release its bus transmit control signal. Note, terminals do not respond to a broadcast Poll with any type of transmission. If a packetization error is detected in a broadcast Poll or any other broadcast packet (such as a CRC error, validation error, or invalid command), the GM is to simply ignore the packet as if it were never received (no NAK is to be sent).

The following diagrams depict the general flow of error-free communications in the GMMS system for the command types used in the protocol:

GMMS		**Gaming Machine**
POLL	-------->	
	<-------	Status Response (if any)
POLL (forced)	-------->	
	<-------	Status Response
POLL	-------->	
	<-------	Transaction Request
POLL(s) (with i bit set)	-------->	
	<-------	Status Response(s) (if any)
Transaction Reply	-------->	
POLL	-------->	
Data Read Request	-------->	
POLL(s)	-------->	
		(no Status Response and Reply not ready)
POLL	-------->	
	<-------	Data Read Reply
POLL(s)	-------->	
	<-------	Status Response(s) (if any)
ACK (if more packets)	-------->	
POLL	-------->	
	<-------	last Data Read Reply continuation packet
(no ACK)		
Data Write Request	-------->	
POLL	-------->	
	<-------	ACK
Site Info Request	-------->	
POLL	-------->	
	<-------	Site Info Reply

This communications flow diagram may serve as a reference during the explanation of the command structure, data structure, and function of these command types in the following sections.

Note, a GM is permitted to transmit communication packets only in response to a Poll command from the GMMS, and a GM is allowed to send no more than one packet in each poll cycle. Notice also that more than one Poll may occur between data packet transfers.

Also note, the GM is expected to ACK every Data Write Request packet that it receives (all packets containing data except Transaction Replies). Conversely, the GMMS will not ACK any packets it receives except Data Read Replies and Site Info Replies, and of those, only the ones which are identified as having continuation packets to follow.

4 COMMAND TYPES

The two byte Command segment defines the command details of each packet. The format for all commands is provided in this section by defining the fields and bits within the segment. Where practical, letters used to identify the bits have been selected to aid recall. The first three bits, the Command Type field (**ccc**), identify the command type.

(ccc) Command Type:
110 - Poll / Status Response
101 - Data Read Request
001 - Data Write Request / Data Read Reply
000 - Transaction Request / Transaction Reply
011 - Site Info Reply
111 - Site Info Request
100 - ACK/NAK

The details and bit assignments of certain Command segment fields, specifically the data length field (nnnnn), the data type codes (rrrr, www, and tttt), the last packet flag (q), the odd/even flag (o), the source indicator (ss), the ACK/NAK code (aaaaa), and the command type being ACK/NAK'd (kkk), are presented in Command Summary, Section 4.10. Additional explanation of some Command segment fields is provided in Section 4.11. The general command segment formats and the functions of all command types are described in the following subsections. In all cases, bits identified by the letter "u" are unused and will always be zero (0).

4.1 POLL COMMAND - Command Type 110

Gaming Machines are allowed to transmit a packet only in response to a Poll command. No other transmission to the GM will cause it to transmit packets. Through the use of flags in the command, the Poll command has the ability to control the flow of communications by limiting the types of commands that the GM may transmit in response to that Poll and also the ability to control the game play enable status. The Poll command also serves to acknowledge the "in-progress" status of a transaction initiated by the GM. This command type does not contain data or validation segments.

command format: cccv dfpi uuuu uuss

The functions of the Specific Control bits (v,d,f,p,i) in the Poll command are as follows:

v bit: Toggles state to acknowledge the last Status Response. (Verify bit)
The GM must track the v bit to determine if the last Status Response sent to the GMMS was properly received. When sending a Status Response to the GMMS, the GM must note the state of the v bit in the current Poll command. If the state of the v bit in the next Poll command has changed, the previous Status Response was received properly and the GM may send any new status conditions. If the state of the v bit has not changed in the next Poll command, the GM must send the previous Status Response, when priorities permit, until verified by the state of the v bit. This includes Status Responses which reported "no change".

d bit: Controls the game play state of the GM. (Disable bit)

If the d bit is set to 0, the GM may enable normal game play. (Note, other conditions and communications parameters may still prevent game play from actually being enabled.)

If the d bit is set to 1, the GM must disable all game play and display a suitable message on the GM screen. The terminal must comply with any South Australia Gaming Machine regulations. Typical regulations usually address the player situation when disabling game play, considering the credit balance and the game win status. For example:

a) If a non-zero credit balance exists when the GM is instructed to disable, the player may be allowed to complete the game in progress (and any bonus feature) before game play is disabled. Subsequently, the terminal may be expected to offer the "cash out" option along with a descriptive explanation to the player.

b) If a zero credit balance exists at the time this bit is set, the terminal is usually expected to immediately disable game play and display a message which discourages attempts to play the terminal.

Note: If the terminal receives a Poll with the d bit set, it must observe and maintain this state through power-fails and resets and not re-enable game play until a Poll with the d bit cleared is received.

f bit: Forces the GM to send a Status Response. (Forced-status Poll bit)

If the f bit is set to 1, the GM must send a Status Response to the current Poll command. That is, if no Status Response reporting a change in condition is pending, a Status Response indicating "no change" must be sent. This allows the GMMS to establish or confirm communications with the GM.

If the f bit is set to 0, the GM is not expected to respond to the Poll command unless it has something ready to send and the Permission bit (p) in the Poll command permits transmission of that command type. When the f bit is set to 0, the GM need only send a Status Response if a status exception has occurred since the last Poll command or if it recognizes that a Status Response from the last Poll has gone unacknowledged, as indicated by the state of the v bit. Thus, the GM may often send no response at all to a Poll.

p bit: Controls the command types allowed in response to a Poll command. (Permission bit)

When the Permission bit is set to 1 (with f bit 0), the Poll command permits a packet that includes data to be sent. In this case, the GM is expected to determine the response based on the following priorities:

Priority 1: Transaction Request, Data Read Reply, or Site Info Reply.
Priority 2: ACK/NAK, or Status Response (status condition change).
Priority 3: No response.

Within each of the first two priority categories, there is to be no particular priority for a specific command type. Rather, the Gaming Machine should transmit responses in a chronological order (first-processed, first-transmitted).

When the Permission bit is set to 0 (with f bit 0), the GM is limited to sending only a command without data. In other words, the GM is expected to send only a Status Response (reporting

change in status) or an ACK/NAK, or no response at all. These responses are also to be treated in a chronological order.

Summarizing the response priorities based on the Permission (p) and Forced-status (f) bits in the Poll command:

f p	Priority
1 x	1) Status Response (reporting change in condition) 2) "No Change" Status Response
0 0	1) Status Response (reporting change in condition) or ACK/NAK (to any Request or Reply) 2) no response
0 1	1) Transaction Request, Data Read Reply, or Site Info Reply 2) ACK/NAK or Status Response (reporting change in condition) 3) no response

x = either state (don't care)

i bit: Indicates Transaction initiated by GM is active. (Transaction In progress bit)
When the GMMS receives a Transaction Request from the GM, the i bit will be set to 1 in the next Poll command to that terminal unless the GMMS has already transmitted the Transaction Reply by the time of the next poll cycle. As long as the i bit is set in subsequent Polls to the GM, the GM should wait for the Transaction Reply. If the i bit is not set or if the i bit becomes reset in a subsequent Poll without having received a Transaction Reply, the GM should resend (without incrementing the Transaction Number) the same (last) Transaction Request. After the Transaction Reply is sent by the GMMS, the i bit will be reset in subsequent polls to the GM.

Note, the GM may only make Transaction Requests when the i bit in the Poll command is not set (0); that is, each GM is allowed to have only one active transaction at a time. If the GMMS receives a Transaction Request while already processing a transaction from the same GM, a Transaction Request NAK will be sent to the GM. (The NAK will not be sent if the transaction number of the new request matches the number of the transaction currently being processed; the new request will be ignored and processing will continue on the current request.)

4.2 STATUS RESPONSE - Command Type 110

GMs transmit a Status Response to report the occurrence of an exception condition detected by the GM, or to respond to a forced Poll command (f bit = 1 in a Poll).

command format: cccu deee eeee ubss

The Exception code field (eee eeee) identifies the detected exception condition which is being reported to the GMMS. If the exception is enabled by the Exception Report Enable flags in the Configuration command, terminals are expected to report the change-of-state to each applicable exception condition as soon as possible after it occurs. When the GM encounters a forced Poll, an all-zero exception code field may be used to indicate the terminal has no new exception conditions to report. The identity and bit values of the all exception codes are presented in the Command Segment Fields Summary, Section 4.10, and further explained in Section 4.11.

In general, the GM is expected to report exception conditions in the order detected and only once while they persist. Some exception conditions are singular events, while others have associated resolution events that must be reported when the condition is concluded or corrected. Some resolution events are not considered complete until all exceptions in that category have returned to normal.

The functions of the Specific Information bits (d,b) are as follows:

d bit: Indicates the current GM game play state at the time of transmission. (Disable bit)
If the d bit is a 0, the GM is in the normal game play state. If the d bit is a 1, the GM is in the game play disabled state. This bit should be a real-time reflection of the game play status, such as, if play is disabled due to a door being open, the d bit in that response should be set.

b bit: Indicates that a packet containing data is ready to send. (Buffered bit)
If the b bit is set to 1, it indicates to the GMMS that the GM has a packet containing data (Data Read Reply, Transaction Request, or Site Info Reply) buffered to send. This flag may be used by the GMMS to efficiently schedule permission for data packet transfer.

DATA-RELATED COMMANDS

The host uses Data Read Request commands to read data from the terminals. The terminal is expected to reply with Data Read Reply commands to upload the data. If multiple packets of data are required to answer a request, the host indicates acceptance of all but the last packet by sending ACKnowledge commands.

In almost precisely the same way (except for the encryption scheme) Site Info Requests and Site Info Replies are used to request and receive data from the GM. The request is typically initiated by the (optional) local GMMS Site Controller.

The host uses Data Write Request commands to send configuration and control data to the terminals. The terminal is expected to reply with ACKnowledge commands indicating acceptance of each packet.

Transactions provide the means for a GM to initiate the exchange of data between the terminal and the GMMS. The terminal initiates transactions by sending Transaction Request commands and the GMMS responds by providing Transaction Reply commands. The i bit in the Poll command provides indication that a transaction is considered in progress.

The last packet flag (q bit) assignment is retained in all data-related commands, but it is fixed to one (1) in all commands that do not actually contain data (Data Read Request, Site Info Request, and ACK/NAK) and in data-related commands that are restricted to one data packet of at most 256 data bytes (Transaction Request and Transaction Reply). In data-related commands where continuation packets may be necessary (Data Read Reply, Data Write Request, and Site Info Reply), the last packet flag (q) along with the odd/even flag (o) serve to track continuation packets.

The data length field (nnnnn) is included in the command segment of all commands that contain data and validation segments. Data is always in eight-byte increments, with the value of nnnnn determined by: (bytes of data divided by eight) minus 1.

See Sections 4.10 and 4.11 for explanations that are not included in the following subsections.

4.3 DATA READ REQUEST - Command Type 101

The GMMS sends the Data Read Request command to request a specific data type from a GM. A Data Read Request does not contain data or validation.

command format: cccu uuuu rrrr uqss

The data type requested is indicated in the data type code (rrrr).

The GM is expected to respond to a Data Read Request on a subsequent Poll cycle with a Data Read Reply, but only when allowed to do so by the Poll command Permission flag.

4.4 DATA READ REPLY - Command Type 001

The GM sends a Data Read Reply in response to a Data Read Request from the GMMS. A Data Read Reply contains data and validation segments.

command format: cccn nnnn rrrr uqss

The data type code (rrrr) in the first Data Read Reply packet will echo the data type code of the corresponding Data Read Request. Continuation packets must have rrrr = 1111.

To solicit continuations, the GMMS will respond to the first packets of a multi-packet Data Read Reply with an ACK, but the GMMS will not send an ACK to the last packet of a Data Read Reply. The GM is expected to track the flow of multi-packet replies to confirm that the Odd/even flag in the ACK from the GMMS corresponds to the proper packet.

The GM is expected to prepare a Data Read Reply packet for transmission following a subsequent Poll command from the GMMS. The Data Read Reply is transmitted only when permitted by the Poll command Permission flag.

4.5 SITE INFO REQUEST - Command Type - 111

The GMMS Site Controller sends the Site Info Request command to request a specific data type from a GM. A Site Info Request does not contain data or validation.

command format: cccu uuuu rrrr uqss

The data type requested is indicated in the data type code (rrrr) and corresponds to the data types associated with the Data Read Request.

The GM is expected to respond to a Site Info Request on a subsequent Poll cycle with a Site Info Reply, but only when allowed to do so by the Poll command Permission flag.

4.6 SITE INFO REPLY - Command Type - 011

The GM sends a Site Info Reply in response to a Site Info Request from the GMMS Site Controller. A Site Info Reply contains data and validation segments.

command format: cccn nnnn rrrr uqss

The data type code (rrrr) in the first Site Info Reply packet will echo the data type code of the corresponding Site Info Request. Continuation packets must have rrrr = 1111.

To solicit continuations, the GMMS Site Controller will respond to the first packets of a multi-packet Site Info Reply with an ACK, but the Site Controller will not send an ACK to the last packet of a Site Info Reply. The GM is expected to track the flow of multi-packet replies to confirm that the Odd/even flag in the ACK from the Site Controller corresponds to the proper packet.

The GM is expected to buffer a single Site Info Reply packet in preparation for response to a subsequent Poll command from the GMMS. The Site Info Read Reply is transmitted only when permitted by the Poll command Permission flag.

Note, the Site Info Reply is very similar to the Data Read Reply with the differences being the contents of one data type (Events Log) and the encryption keys used.

4.7 DATA WRITE REQUEST - Command Type 001

The host uses this command to send control and configuration (non-transaction type) data to a GM. A Data Write Request contains data and validation segments.

command format: cccn nnnn wwwo uqss

The data type code (www) in the first packet of a Data Write Request will reflect the data type being sent. The data type code will equal 111 in continuation packets.

The GM is expected to respond to each packet of a Data Write Request with either an ACK or NAK command, but only in response to a subsequent Poll command from the GMMS.

4.8 TRANSACTION REQUEST/REPLY - Command Type 000

The GM uses the Transaction Request command to start a transaction with the GMMS. The GMMS responds with a Transaction Reply command. Transaction Request and Transaction Reply commands contain data and validation segments.

command format: cccn nnnn tttt uqss

The transaction data type in the request or reply is indicated in the data type code (tttt).

The GM initiates a transaction by sending a Transaction Request in response to a Poll command, but only when allowed to do so by the Poll command Permission flag. The GMMS will either respond with a Transaction Reply or a transaction NAK, or acknowledge that a transaction is in progress by setting the i bit in the next Poll command to that terminal. If the one of these cases is not met, the GM should re-initiate the same transaction. A Transaction number, sequentially assigned by the GM and included in the data segment, is used to confirm the related Transaction Reply from the GMMS.

The Transaction Reply to the GM provides data in response to the Transaction Request. The GM is only expected to respond (with a NAK) to a Transaction Reply if the packet is detected as invalid and then only in response to a subsequent Poll command from the GMMS.

4.9 ACK/NAK - Command Type 100

The ACK/NAK command serves to acknowledge (ACK) acceptable receipt of a packet, or to reply with an error code when a recognizable error is detected (NAK or negative acknowledge). This command type does not contain data or validation segments.

command format: ccca aaaa kkko uqss

The ACK/NAK code (aaaaa) provides insight into the reason that a command is being NAK'd. The (kkk) field corresponds to the command type (ccc) being ACK'd/NAK'd. See section 4.11 for the important role of the odd/even flag (o) in multi-packet transfers.

The GM is expected to send an ACK to the GMMS only when a Data Write Request packet is received. If errors are detected in any received packet (other than a Poll command), the GM is expected to respond to a subsequent poll with the appropriate NAK (see Section 4.11 for NAK expectations). Using logical discretion, the host will retransmit packets when it receives a NAK from the GM. As with all other commands, an ACK/NAK may be sent by the GM only in response to a Poll command.

To solicit continuations, the GMMS will send an ACK/NAK to the leading packets of a multi-packet Data Read Reply or a Site Info Reply received from the GM, but the GMMS will not send an ACK to the last packet. Likewise, the GMMS will not send an ACK to a Transaction Request or any data command type which does not require continuation packets. The GMMS will, however, send a NAK if necessary.

Note, in a multi-packet Data Read Reply, the terminal is expected to retransmit the previous packet (in response to the next permitted Poll command) when it receives an ACK in which the odd/even (o) flag does not reflect the proper state.

4.10 COMMAND SEGMENT FIELDS AND FLAGS SUMMARY

COMMAND TYPES:

(Command Types with no data or validation):

POLL (from GMMS host or Site Controller to GM)
command format: cccv dfpi uuuu uuss
(110v dfpi 0000 00ss)

STATUS RESPONSE (from GM)
command format: cccu deee eeee ubss
(1100 deee eeee 0b00)

ACK/NAK
command format: ccca aaaa kkko uqss
(100a aaaa kkko 01ss)

DATA READ REQUEST (to GM)
command format: cccu uuuu rrrr uqss
(1010 0000 rrrr 01ss)

SITE INFO REQUEST (from GMMS Site Controller to GM)
command format: cccu uuuu rrrr uqss
(1110 0000 rrrr 0111)

(Command Types which include data and validation):

DATA READ REPLY (from GM)
command format: cccn nnnn rrrr uqss
(001n nnnn rrrr 0q00)

SITE INFO REPLY (from GM to GMMS Site Controller)
command format: cccn nnnn rrrr uqss
(011n nnnn rrrr 0q00)

DATA WRITE REQUEST (to GM)
command format: cccn nnnn wwwo uqss
(001n nnnn wwwo 0qss)

TRANSACTION REQUEST/REPLY
command format: cccn nnnn tttt uqss
(000n nnnn tttt 01ss)

COMMAND FIELDS:

ccc = Command type code

ccc
000 = Transaction Request or Transaction Reply
001 = Data Read Reply or Data Write Request
011 = Site Info Reply
100 = ACK/NAK
101 = Data Read Request
110 = Poll or Status Response
111 = Site Info Request

aaaaa = ACK/NAK code

aaaaa
00000 = ACK
NAK codes:
00010 = Communication CRC error
00011 = Validation segment Time stamp out of window
10010 = Invalid Command
10100 = Validation Code error
10101 = Transaction error
10110 = Configuration error
10111 = Non-zero credit balance error

eee eeee = Exception code (* = not disabled by Exception Report Enable flags)

	eee eeee	**Significant Events**:
*	$00	000 0000 = No Change since last verified Status Response (for Forced Poll)
*	$02	000 0010 = Memory Signature Calculation Completed (only if initiated by non-zero divisor in Write Time and Date command)
	$03	000 0011 = Gaming Machine Entered Playable State
	$04	000 0100 = Entered Non-playable State (caused by GM)
	$05	000 0101 = Entered Non-playable State (caused by GMMS)
*	$OB	000 1011 = Status Response Buffer (exception queue) was Filled
	$0C	000 1100 = Events Log Full
	$0D	000 1101 = Daily Poll Communication Timeout

	eee eeee	**Special Conditions**:
	$10	001 0000 = Returned to All Special Conditions Cleared (applies to codes $11-1D)
	$11	001 0001 = Coin Acceptor Tilt (stringing, timing, jam, etc.)
	$13	001 0011 = Bill Acceptor Tilt (jam, stacker full, etc.)
	$15	001 0101 = Printer Paper Low
	$16	001 0110 = Printer Paper Out
	$19	001 1001 = Hopper Empty
	$1B	001 1011 = Hopper Jam
	$1D	001 1101 = Reel Tilt (mechanical reels)
	$1E	001 1110 = Coin Diverter Error (coin to wrong location) (NO CLEAR)
	$1F	001 1111 = Hopper Excess Coin (one) Dispensed (NO CLEAR)

eee eeee **Door Accesses**:

$2x 010 addd = Door accesses: (only closure reported is "all doors closed")
ddd = 000 = All doors Closed
ddd = 001 = Main Door Opened
ddd = 010 = Cash Door Opened
ddd = 011 = Secondary Door Opened
ddd = 100 = Logic Door Opened
a = State of Audit Key when doors are opened or all-closed

Door Examples:

$28 010 1000 = Change of state to All Doors Closed (while Audit Key on)
$22 010 0010 = Cash Door Opened (while Audit Key off)
$2C 010 1100 = Logic Door Opened (while Audit Key on)

eee eeee **Administrative/Diagnostics Access (Audit Key)**

$27 010 0111 = Administrative/Diagnostics Exit (Audit key turned Off)
$2F 010 1111 = Administrative/Diagnostics Access (Audit key turned On)

eee eeee **System Failure Faults**:

$30 011 0000 = Returned to "no Fault" condition (Applies to $31 - $47)
$34 011 0100 = EPROM Checksum Failure
$35 011 0101 = Fatal RAM Failure
$36 011 0110 = Battery Low
$37 011 0111 = Main Logic Failure
$38 011 1000 = Power Supply Failure
$39 011 1001 = Coin Acceptor Failure
$3B 011 1011 = Hopper Failure / Runaway (2 or more excess coins)
$3D 011 1101 = Printer Failure
$3E 011 1110 = Bill Acceptor Failure
$3F 011 1111 = Touchscreen Failure
$40 100 0000 = Over-temperature
$41 100 0001 = Electromechanical Meter Failure
$45 100 0101 = Miscellaneous Firmware Failure
$46 100 0110 = Miscellaneous Electronics Failure
$47 100 0111 = Miscellaneous Mechanical Failure

eee eeee **Local Site Items**:

$48 100 1000 = Request Answered
$49 100 1001 = Beverage Request by player
$4A 100 1010 = Change Request by player
$4B 100 1011 = Mechanic Request by player

eee eeee **Substantial Categories**:

$51 101 0001 = Substantial Win Awarded (based on Configuration field)
$52 101 0010 = Substantial Cashout Performed (based on Configuration field)

kkk = command type being ACK/NAK'd
Values correspond to command type (ccc).

nnnnn = data length field, in eight byte increments
value of nnnnn = (bytes of data / 8) - 1

rrrr = data Read type code
rrrr
0000 = Control
0001 = Monitor @ last Day End Snapshot
0010 = Configure
0011 = Statistics @ last Week End Snapshot
0100 = Banner / Reports
0101 = Monitor, Current
0111 = Statistics, Current
1001 = Events Log
1100 = Date and Time / Memory Signature
1111 = continuation packet

ss = Source indicator
ss
00 = Gaming Machine
01 = Audit device
10 = GMMS Host
11 = GMMS Site Controller

tttt = Transaction type code
tttt
0000 = Cashout

uuuuu = Unused field or bits (always 0's)

www = data Write type code
www
000 = Control
001 = Configure
010 = Banner / Reports
110 = Date and Time / Memory Signature
111 = continuation packet

COMMAND FLAGS (single bit):

b = Buffered flag

Indicates that the GM has a packet which contains data buffered to send (b = 1).

d = Disable flag

Controls or indicates the GM game play state, disabled or enabled (1 = disabled).

f = Forced-status flag

Forces GM to send a Status Response (f = 1). If no new exceptions have occurred, a Status Response reporting "no change" will be sent.

i = transaction In progress

Indicates to GM that the GMMS recognizes a transaction as currently active.

o = Odd/even flag

Initial data packet is odd (o = 0), flag value toggles on consecutive continuation packets.

p = Permission flag

Permits (p = 1) GM to transmit packets including data. If set to 0, GM transmission is limited to command types which do not contain data.

q = last packet flag

Indicates last packet (q = 1) of data related packets. Set to 0 to indicate continuation packets to follow.

u = unused bits (always 0)

v = Verify flag

Toggles to acknowledge receipt of the last Status Response.

GM RESPONSE PRIORITIES:

(Based on Permission (p) and Forced-response (f) bits in Poll Command)

f	p	Priority
1	x	1) Status Response (reporting change in condition) 2) "No Change" Status Response
0	0	1) Status Response (reporting change in condition) or ACK/NAK 2) no response
0	1	1) Transaction Request, Data Read Reply, or Site Info Reply 2) Status Response (reporting change in condition) or ACK/NAK 3) no response

4.11 COMMAND FIELDS AND FLAGS SUPPLEMENTARY DETAIL

ACK/NAK Codes

NAK Codes sent by the GM:

When the GM detects any of the following conditions, it is expected to transmit the NAK with the appropriate error code (in response to a Poll) and abort processing of the received packet. Any further action will be based on subsequent commands from the GMMS.
A **Communications CRC error** (00010) is to be generated in response to a received packet which appears correct (that is, header, terminal ID, length, and trailer are proper) except that the CRC check performed on the packet fails. (Applicable in response to Data Read Request, Data Write Request, Transaction Reply, and ACK only. This is **not** to be sent to signify CRC errors in Poll commands; the Poll should be ignored.)

A **Validation Segment Time Stamp Out of Window error** (00011) is to be generated in response to a received data packet which appears correct (that is , header, terminal ID, length, trailer, CRC check, and validation code check are all proper) except that the date and time stamp in the validation segment differs from the current time at the GM by more than the allowed window (10 seconds). (Applicable in response to Data Write Request, and Transaction Reply.)

An **Invalid Command** (10010) code is to be generated in response to a received packet which appears correct (that is, header, terminal ID, length, trailer, CRC check, and validation code check and validation time stamp, if applicable, are all proper) except that the command segment of the packet refers to a data read or write type (rrrr, www, or kkk in ACK) or transaction type (tttt) which is undefined or not supported by the GM. (Applicable in response to Data Read Request, Data Write Request, Transaction Reply, and ACK.)

A **Validation Code error** (10100) is to be generated in response to a received data packet which appears correct (that is header, terminal ID, length, trailer, and CRC check are all proper) except that the validation code check over the data segment of the packet does not pass. (Applicable in response to Data Write Request, and Transaction Reply.)

A **Transaction error** (10101) is to be generated in response to a received data packet which appears correct (that is, header, terminal ID, length, trailer, CRC check, validation code check, and validation time stamp are all proper) and is a Transaction Reply packet that the GM is expecting except that any data echoed from the request, such as the transaction number, does not match the data in the GM. (Applicable only in response to Transaction Reply.)

Note, if the GM has a transaction request outstanding after aborting the received packet, it should resend that transaction request at the next opportunity.

A **Configuration error** (10110) is to be generated in response to a received data packet which appears correct (that is, header, terminal ID, length, trailer, CRC check, validation code check, and validation time stamp are all proper) and the packet is a Configuration Data Write Request that is not valid for the GM, such as a Game Type Number, or any other field that it is unable to support, such as a Game Credit Size that is fixed at a different value in the terminal. Another example would be if game type configurations are attempted when the GM is not in Full Configure Enable Mode. This NAK is also expected in response to a Control Data Write Request command attempting to set the GM into Normal mode when the game configuration on the GM does not allow Normal mode, such as without having ever received a valid Configuration command. (Applicable only in response to Configuration or Control Data Write Request.)

A **Non-zero credit balance error** (10111) is to be generated in response to a received data packet which appears correct (that is, header, terminal ID, length, trailer, CRC check, validation code check, and validation time stamp are all proper) but the command would be inappropriate to accept with credit on the terminal. For example, a Control command to Full Configure Enable mode is improper since reconfiguration should not occur while the terminal is being played. (Applicable only in response to Data Write Request.)

NAK Codes received by the GM:

A **Transaction Error** (10101) will be sent by the GMMS in response to a Transaction Request whenever a Transaction Reply is already pending to a different transaction from that GM, or under any other regulatory conditions which do not allow the GMMS to accept Transaction Requests from the GM. Examples may include when the GM is not properly enrolled on the system, when the Site Controller enclosure is accessed, and Cashout Requests when a Memory Signature has been initiated by the Site Controller. In all cases, the GM is expected to abort the NAK'd transaction and proceed as if the NAK'd transaction had never occurred (do not automatically retry the transaction and do not increment the transaction number).

Note, this NAK is not generated for the case when the transaction number in a Transaction Request matches the number of the transaction currently in progress (considered a retry).

EXCEPTION Codes

The exception conditions which are reported through Status Responses (and similarly for Events Log Entries) are grouped into several categories. In general, exceptions within each group have common traits or require similar attention. Some have a generic resolution event that is not reported until all exceptions in that group have been resolved or corrected.

The determination as to whether the GM is expected to actually report specific exception events is enabled/disabled through the Exception Report Enable flags field in the Configuration command. The terminal is expected to default (until Configured) this flag field to zero; this enables only the reporting (and queuing) of Exceptions $00, $02, and $0B.

The reporting of most exception conditions is only useful when collected in real time and when provided directly to the personnel at the gaming site. This suggests that Exception Reports Enable flags will only be configured to a non-zero value when a location (venue) has chosen to install an optional Site Controller which can provide the exception information.

SIGNIFICANT EVENTS:
The terminal is expected to be capable of detecting and reporting the occurrence of all conditions defined in this category. The reporting of exception events in other exception categories will be dependent upon the features of the individual Gaming Machines.

The **Memory Signature Calculation Completed** exception event must be reported whenever a Memory Signature Result has been calculated as an outcome of a non-zero Divisor in Date and Time Data Write Request (not reported when initiated by a Control Data Write Request). This exception event is always reported (not disabled through the Exception Reports Enable flags).

The **Playable** versus **Non-playable State** referred to in exceptions 3, 4, and 5 refer to the availability of games to the player. These exceptions are to be reported at the time the GM makes the transition to the new state, which is not necessarily at the same time as the circumstance which caused the occurrence (such as if the player is in the middle of a Poker game and the GM is not disabled until after the hand is completed).

Non-playable states caused by the GM should include faults that occur at the terminal and other conditions that occur as a result of terminal limitations, such as Events Log Full, all of which would leave the terminal permanently out-of-order if action was not taken by location personnel. This is not intended to include any self-correcting conditions, such as some tilts, or door openings which will typically be closed by attending personnel.

Non-playable states caused by the GMMS include entering Full Configure Enable mode and Disabled by Host mode as well as being disabled by the Poll command (with d bit set). This should also include incidents which are a result of parameters provided by the GMMS, such as Daily Poll Timeouts.

The **Status Response Buffer Full** exception is to be reported whenever enabled exceptions have to be discarded due to an overflow of the exception queue. The exception should not be subsequently reported until the queue has reached at least the half-empty state and refilled again. This exception is always reported (not disabled by the Exception Reports Enable flags).

Once a terminal has been enrolled on the system (no longer in Demo Mode), it is expected to disable game play after the current game is completed whenever an **Events Log Full** or **Daily Poll Timeout** condition occurs. These situations can typically be resolved by informing host personnel, who will initiate host communication with the terminal.

SPECIAL CONDITIONS:
The events included in Special Conditions are the type which typically can be corrected by actions such as adding printer paper, un-jamming coin acceptors, or filling the hopper. Not all events require immediate attention. If a resolution event is appropriate, it is not reported until all active special conditions have cleared.

DOORS:
Each door opening is reported as a separate exception event, but door closings are not reported until all doors are closed. The state of the Audit Key switch is included to distinguish between legal and illegal door openings if a procedure is implemented to require the Audit Key ON when doors are accessed. Note, the state of the Audit key bit should always be set properly in the door access events even if the Administrative/Diagnostics Access event reports are disabled. Door accesses that are detected as occurring with power off are reported when power is restored.

ADMINISTRATIVE/DIAGNOSTICS ACCESS (Audit Key):
The terminal is typically expected to have an external Audit Key switch which provides access to the administrative and diagnostics functions of the terminal. Access to these functions is reported with these exception codes.

SYSTEM FAILURE FAULTS:
In general, action by technical site personnel will typically be required to correct this type of fault. The resolution event is not reported until all Faults in this category are cleared.

Terminal manufacturers are expected to match their diagnostic and detection conditions as appropriately as possible to the defined codes. South Australia regulations may establish some as mandatory troubleshooting aids.

LOCAL SITE ITEMS:
Player Requests are reported once when initiated by the player. The Request Answered event should not be reported if the function is not secure (if a player can toggle the request answered status).

SUBSTANTIAL WIN AWARDED:
Any game win that meets or exceeds the Substantial Win Award provided in the Configuration data type is to be reported with this exception.

SUBSTANTIAL CASHOUT PERFORMED:
This exception is based on the Configuration data field of the same name. Any single cashout (including a printed Cash Ticket, hopper payout, and/or manual pay) that meets or exceeds this value is to be reported as an exception.

DATA LENGTH Field

The "data length" field (nnnnn) specifies the length of the data segment in multiples of eight (8) bytes and is included in all command types that include data. The minimum value of the data length field is 00000 (zero), which indicates eight bytes of data. The maximum value of the data length field is 11111 (31) which indicates 256 bytes of data.

Value of nnnnn = (bytes of data / 8) - 1

The actual number of bytes in a data segment may be computed using the following formula:

Bytes of data = (value of nnnnn + 1) * 8.

Thus:

A nnnnn of 00000 indicates 8 bytes of data.
A nnnnn of 00001 indicates 16 bytes of data.
A nnnnn of 00010 indicates 24 bytes of data.
A nnnnn of 11111 indicates 256 bytes of data.

SOURCE Indicator

The Source Indicator field indicates the origin of the packet. Currently, GMs need only check this field to confirm that the source of a packet was not another GM.

ODD/EVEN Flag

The "Odd/Even Flag" (o) is useful on multi-packet Data Write Request transfers and associated ACK/NAKs. It is also important in the ACK/NAKs sent by the GMMS to Data Read Replies.

The odd/even flag value in an initial Data Write Request packet sent by the GMMS is set to zero (0). In multi-packet transfers, the odd/even flag of the second (continuation) Data Write Request packet is set to one (1). In the third Data Write Request packet (continuation), the odd/even flag value is zero (0), and so on. The ACK from the GM to the GMMS is expected to echo this flag if the packet was received properly. The GMMS may retransmit packets if it does not receive an ACK from the terminal.

When sending a Data Read Reply, the GM must be able to track whether its packet is even or odd during any multi-packet reply. The Odd/Even bit (o) in the ACK or NAK to the GM reflects whether the GMMS is referring to an odd or even packet. In the ACK for the initial packet, the Odd/Even bit will be zero (0). For the first continuation packet, it will be a one (1).

Thus, if the terminal receives an ACK to the first continuation packet with o = 0 (zero), the GMMS apparently either did not receive the continuation packet or has timed out and is re-ACKing the initial packet. The GMMS may do this when it can not tell whether the terminal received the initial ACK. In this situation the terminal must resend the first continuation packet when allowed to do so by the Poll command. In a NAK packet, the odd/even flag from the corresponding packet will be echoed/returned.

This procedure allows the communications handlers in both the GM and GMMS to recognize a new packet as opposed to a repeated/resent packet.

LAST PACKET Flag

The Last Packet Flag (q bit) is used by data-related packets to indicate that either the current packet is the last packet or that there are additional packets of the same type to follow. On the last packet, the bit value must be a one (1); otherwise a zero implies that there are continuation packets to follow.

5 DATA TYPES

Commands in the communications protocol that include data segments use a data type code (tttt, rrrr, www) in the command segment to indicate the type of data in the request/reply packet. The Transaction Request and Transaction Reply commands typically each have a separate data segment structure while the Data Write Request and Data Read Reply commands typically share a common data segment structure for each set of data types.

The data segment of each data type includes a fixed Format Identifier code. If changes are made to the protocol which affect the field format of a particular data type, the Format Identifier code will be changed, thus providing a means for the GMMS to identify and support multiple versions of the protocol within the same jurisdiction. When receiving a particular data type from the GMMS, the GM can also use this code to confirm that it has received the expected version.

TRANSACTION REQUEST AND TRANSACTION REPLY

Transactions are initiated by the GM by generating a Transaction Request. The GMMS responds in turn with a Transaction Reply. The following table shows the transaction data type codes that are currently defined for the (tttt) bit field of the command segment of a Transaction Request/Reply and the Format Identifiers that are currently assigned to the data types:

(tttt)	Data Type	Format Identifier
0000	Cashout	$05

Included in each Transaction data segment is a transaction number. Independent of the Transaction type, the GM assigns the Transaction number in a circular sequential fashion to each new Transaction Request it generates. To aid in tracking and identification, the Transaction number is echoed in the corresponding Transaction Reply from the GMMS.

The Transaction Number (TTN) should only be incremented when the GM receives a Transaction Reply which satisfies the Transaction Request. In all other cases (such as no Reply or an improper Reply received by the GM, or a NAK to the Request), the terminal is expected to leave the TTN intact and use the same TTN in the next transaction regardless of whether it is of the same transaction type or not. If, after the i bit in the Poll command has returned to zero, the Gaming Machine has not received a valid Transaction Reply or a Transaction Error NAK, the terminal should re-queue and re-send the exact same Transaction Request until one of these legitimate responses is received. If a Transaction Error NAK is received, the transaction is to be aborted without incrementing the TTN.

When a Gaming Machine makes the very first Transaction Request after exiting Demo Mode, it is expected to use a TTN of zero (0). After that, a TTN of zero is not to be used again by that GM unless it is reset to Demo Mode and re-enrolled on the system. Thus, the TTN is expected to roll over from 255 to 1 ($FF to $01), not 0.

Of the 16 possible transaction data types, the enable/disable status of the first 8 (tttt = 0 through 7) are controlled by the Transaction Enable Flag field in the GM Configuration data. If the respective bit is cleared, the transaction type is disabled. This field should be defaulted to zero.

CASHOUT OPTIONS:
A Gaming Machine must observe the applicable Transaction Enable Flag in its Configuration data when generating a Cash Ticket or when performing a cancel-credit (manual pay) operation. If the Cashout Transaction bit is set, the terminal must perform a Cashout Transaction before printing a Cash Ticket or cancelling credit on the GM. If this bit is not set, the terminal is allowed to print a Cash Ticket and/or cancel credit without performing a transaction. If no transaction is required, the same Cash Ticket print format should be followed but the terminal is not required to include the Certification and Authorization numbers defined below. Typically, Cashout transactions will not be enabled unless an optional GMMS Site Controller has been installed in the venue.

5.1 CASHOUT REQUEST

A Cashout Request is to be generated by the GM when a player elects to cash out through the Cash Ticket option and/or when a cancel credit (manual pay) operation is performed. The request contains the current terminal credit balance or the amount of the manual pay in cents as the Cash Out Amount. Note, any cash or coin inserted into the terminal after the Cashout Request is initiated must be rejected or tracked separately to give credit after the transaction is complete. When this transaction is enabled, GMs may not print cash tickets or complete manual pays without making a Cashout Request and receiving a valid Cashout Reply from the GMMS. (Note, the GM is not to make Cashout Requests while in the Demo Mode, but may be allowed to print mock Cash Tickets which clear the credit balance as long as the printed ticket is clearly marked as VOID. See Section 5.2 for further details.)

The following table shows the data segment contents of the Cashout Request (Cashout Format Identifier $05):

Byte	**Format**	**Description**
1	Binary1	Cashout Format Identifier ($05)
2		Reserved
3	Binary1	Transaction Number (sequentially assigned by GM)
4	Code1	Cashout Type Code (0 = Cash Ticket, 1 = Manual Pay)
5-8	Binary4	Cashout Amount in cents

5.2 Cashout REPLY

The Cashout Reply is generated by the GMMS after receiving a Cashout Request from a GM. The reply contains information that the GM must print on a cash ticket and/or enter into its Events Log. This information is used to validate the ticket when presented by the player for redemption and to track the cashouts in the GMMS.

The following table shows the data segment contents of the Cashout Reply:

Byte	**Format**	**Description**
1	Binary1	Cashout Format Identifier ($05)
2		Reserved
3	Binary1	Transaction number (echoes number in request)
4-6	Binary3	Venue ID Number
7-10	Binary4	Cashout Amount (echoes amount in request)
11-12	Date(U)	Date of cashout issue
13-14	Time(U)	Time of cashout issue
15-16	Binary2	Cashout Number
17-18	Binary2	Security Number
19-24		Reserved

When the GM receives a valid Cashout Reply (amount and transaction number matches request), it is expected to make a Cashout entry in the Events Log, and, if applicable, print a cash ticket.

Two validation numbers, the Certification and Authorization numbers, must be printed on each ticket to provide unique identification for Cash Ticket Validation and ticket tracking by the GMMS. The Universal date and time, in 24-hour format, and the validation numbers will be printed in decimal form to facilitate the validation process for site personnel.

To create the Certification and Authorization numbers, the GM must convert four binary fields received in the Cashout Reply to decimal numbers:

Venue ID Number (bytes 4-6) => $V_6V_5V_4V_3V_2V_1V_0$ (decimal digits)
Date (bytes 11-12, in Julian days) => $J_4J_3J_2J_1J_0$ (decimal digits)
Cashout Number (bytes 15-16) => $C_4C_3C_2C_1C_0$ (decimal digits)
Security Number (bytes 17-18) => $S_4S_3S_2S_1S_0$ (decimal digits)

These converted numbers are concatenated and printed at the bottom of the ticket in the following format (observe spaces for readability; information in parentheses not printed):

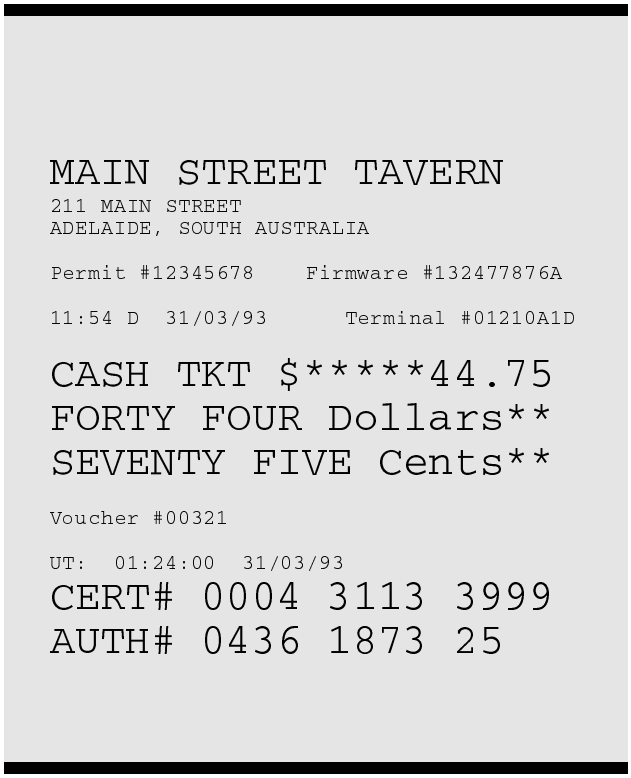
UT hh:mm:ss dd/mm/yy (Universal Time in Hour:Min:Sec Day/Month/Year format)
CERT# xxxx xxxx xxxx ($V_6V_5V_4V_3\ V_2V_1V_0J_4\ J_3J_2J_1J_0$)
AUTH# xxxx xxxx xx ($C_4C_3C_2C_1\ C_0S_4S_3S_2\ S_1S_0$)

Typically, the printed cash ticket will also be required to include the following:

* Venue Name and Address (from Configuration)
* Terminal number

  (The recommended printed number includes the two-digit hexadecimal number assigned to each manufacturer, the South Australia system number ($21), and the four-digit hexadecimal Terminal ID number, in that order.) Example: Terminal Manufacturer $01 and Terminal ID $0A1D gives $01210A1D
* Local Date and Time (corresponding to the date and time of the Terminal Events Log Cashout record; the 'D' below indicates Daylight Savings Time; 'S' for Standard)
* Cashout Amount in both numerical and text form (from Cashout Reply)
* Voucher Number (decimal cash voucher number sequentially assigned by the GM)
  (suggested range of 1 to 65535, with rollover if necessary)
* Permit Number (hex) provided in the Configuration data packet.

An example of a recommended Cash Ticket is provided below:



Note, Cash Tickets printed while in the Demo Mode do not utilize the Cashout transaction process. Terminals in the Demo Mode are typically allowed to print mock cash tickets to clear the credit balance and demonstrate the appearance of a cash ticket, but the following guidelines are typically expected to be followed:

a) Location and address information defaulted to ASCII string (manufacturer-determined).
b) Permit Number defaulted to zero.
c) Voucher Number reset to one (1) when terminal exits Demo Mode.
d) Last three lines (UT, CERT#, & AUTH#) replaced with at least two lines of "VOID VOID VOID VOID" or the like in bold or large, readily-visible letters.

DATA READ REPLY and DATA WRITE REQUEST

The following table summarizes the data types that may be included in the data segment of Data Read Reply and Data Write Request command types. The data type codes (rrrr or www) are included in the command segment of the packet to identify the type of data included (or requested, in the case of the Data Read Request which has no data).

Data Write (www)	**Data Read (rrrr)**	**Data Type**	**Format Identifier**
000	0000	Control	$05
	0001	Monitor - Last Day End	$05
001	0010	Configure	$05
	0011	Statistics - Last Week End	$05
010	0100	Banner / Report	$05
	0101	Monitor - Current	$05
	0111	Statistics - Current	$05
	1001	Terminal Events Log	$05
110	1100	Date and Time / Memory Signature	$05
111	1111	Continuation Packet	None

Note that some data types are read-only, while others are both read and write. Read-only data types are maintained and updated by the GM only; the GMMS has the capability to only read the data field values in these data types. Read/write data types can be both written and read by the GMMS. Typically, when these data types are written by the GMMS, the terminal is expected to change to the new values or indirectly verify that the written values accurately reflect the matching parameters within the GM. When read/write data types are read, they are expected to indicate the values of the data fields at the time they are reported by the GM.

The rrrr bits in Data Read Request packets identify the data type requested. For example, if the GMMS requests the Monitor at Last Day End information, it will send a Data Read Request command to the GM with these data type bits set to 0001. The rrrr bits in the Data Read Reply must correspond. If the GMMS sends new Control information to a GM, it will send a Data Write Request command with the data type code (www) set to 000.

Note: GMs must be able to accept variable length packets, up to the full 256 data bytes (272 bytes total). Continuation packets are used when the amount of data exceeds 256 bytes. For example, in packetizing Events Log data where event entries are 8-byte records, the first 32 Events Log records go into the first packet. If there are additional records to send, the remaining records go in subsequent continuation packets with up to 32 records (256 bytes) in each packet.

SITE INFO REPLY

The Site Info Reply (requested by a Site Info Request) can provide the same data types as the Data Read Reply. With respect to the Data Read Reply, this command differs in two minor ways:

1) When data segment encryption is enabled, the data segment of the Site Info Reply is encrypted with a different encryption key than the Data Read Reply (see Encryption Control and Keys in Section 5.4).

2) A Terminal Events Log Site Info Reply does not determine the starting point of the data contents in the same way as a Data Read Reply. Rather, the Site Info Reply always returns all of the Events Log records that have been generated since the start of the previous day (see the Terminal Events Log introduction in Section 5.8).

The following subsections describe the data segment contents for each data type. All features shown, except those identified as "{FUTURE}", are supported by the GMMS. Fields identified as "{optional}" are beyond the *basic* terminal requirements and support by the GM is not required. If the terminal ignores data received in the optional fields, it must return zeros in these fields when responding to a Data Read Request. "Reserved" bytes in the data segment structure must be ignored when received by the terminal; however, they must be sent as zeros when transmitted by the terminal to the GMMS.

5.3 CONTROL

The Control data type contains basic GM control information which may be changed by the GMMS on a regular basis, typically daily. The Control data segment definition applies to a Data Write Request to the terminal, and both a Data Read Reply and Site Info Reply from the terminal.

The following table shows the contents of the Control data segment (Control Format Identifier $05) and a more detailed description for each Control-related field follows this table:

Byte	**Format**	**Description**
1	Binary1	Control Format Identifier ($05)
2		Reserved
3	Binary1	Control Version Number
4		Reserved
5	Code1	Terminal Mode field:

bits	Description
7-5	Terminal Mode code: 000 - Normal Mode (0) 001 - Full Configure Enable Mode (1) 010 - Demo Mode (2) 011 - Host-Disabled Mode (3)
4-2	Reserved
1	Terminal Status Poll Timeout Enable
0	Daily Poll Timeout Enable

6		Reserved
7	Binary1	Daily Poll Timeout - Hours
8	Flags1	Holiday Flags:

bits	Description
7	Reserved
6-0	Sunday (bit 0) through Saturday (bit 6) flags

9-10	Binary2	Memory Signature Divisor
11		Reserved
12-14	Binary3	Events Log Read Record Number (next read, FROM)
15-22	Flags8	Game Key Enable flags, correspond to game keys 1-64

bits	Description
63	Game Key 1
62	Game Key 2
..	.....
0	Game Key 64

23-26		Reserved
27-		Variable Banner Data (0 to 127 bytes.)

CONTROL VERSION:
The Control Version Number is set by the GMMS host during a Data Write Request of the Control data. The current Control Version Number must be returned by the terminal in Control and Monitor Replies. It is used by the GMMS for monitoring the last Control packet received and accepted by the terminal.

TERMINAL MODE Field:
The Terminal Mode field is a one-byte flag field used to control operational features of the terminal.

bits	Description
7-5	Terminal Mode code: 000 - Normal Mode (0) 001 - Full Configure Enable Mode (1) 010 - Demo Mode (2) 011 - Host-Disabled Mode (3)
4-2	Reserved
1	Terminal Status Poll Timeout Enable - If the bit is set to 1, the entire terminal (all games) is expected to observe the Status Poll Timeout established in the System Configuration field. - If the bit is set to 0, the Status Poll Timeout field is to be observed on a game-by-game basis according to the flags included in the Game Key Configuration section of the Configuration packet.
0	Daily Poll Timeout Enable - If the bit is set to 1, the Daily Poll Timeout function is enabled. - If the bit is set to 0, the Daily Poll Timeout function is disabled.

TERMINAL MODE Code:

The use and definition of the Terminal Modes are explained by describing some of the situations and events that occur during the operation of the Gaming Machine Monitoring System. Note, the Terminal Mode is changed only by a Control Data Write Request from the host.

When a terminal is delivered from the factory and prior to being enrolled on the system, it is expected to be in Demo Mode (2). While in Demo Mode, a gaming jurisdiction's primary requirement and concern is usually that the Gaming Machine must be unable to operate for any type of gambling purpose. Illustrative game play is typically allowed in the Demo Mode to facilitate terminal demonstration and testing. Jurisdictional concerns are usually met by following the following guidelines when in Demo Mode:

a) do not accept or give credit for any money inserted into the terminal.
b) do not give permanent or meaningful credit for the outcome of any games played on the terminal.

While in Demo Mode, terminals should not generate Transaction Requests (Cashout or any other kind) and although GMs are not allowed to generate valid Cash Tickets, they are typically allowed to generate mock Cash Tickets as described in Section 5.2. Terminals are expected to support all other communication protocol while in Demo Mode, but they do not need to shut down or time out for Events Log Full, Status Poll Timeout, or Daily Poll Timeout and, since no meaningful game play occurs, all reported accounting and game play data fields should have zero (0) values. When in Demo Mode and until a valid Configuration is received,

the gaming machine is to consider itself as having no games configured on the machine; this means that, until Game IDs and other game parameters have been properly assigned by the GMMS through the Configuration process, the Game sections of the Monitor or Statistics packets (see 5.5 Monitor and 5.6 Statistics) should not be returned at all when responding to a Data Read Request.

When the host is ready to enroll a Gaming Machine, it will first read and set (if necessary) the time and date of the terminal. After reading data from the terminal to record any activity while in the Demo Mode, the host will then set the terminal to either Full Configure Enable Mode (1) or to a Host Disabled Mode (3). At this time, as the terminal exits the Demo Mode, it must perform an automatic Master Reset.

A Master Reset is expected to zero all of the accounting, performance statistics, and game-related counters, then set various reset dates and times (including the Master Reset Date and Time in the Monitor data type), and initialize the Events Log. Once a terminal exits the Demo Mode, regulations usually dictate that it shall contain no provision to return to the Demo Mode except through the factory or a certified service center after obtaining proper approval from the regulating jurisdiction.

After setting the terminal to Full Configure Enable Mode (1), the host will configure the terminal by sending a valid Configuration packet. An ensuing Control command will set the terminal to Normal Mode (0). **The Normal mode (0), which is the only mode that allows normal game play, must be entered only after receiving the appropriate Control command from the host.**

The Full Configure Enable mode is subsequently set by the host only when it needs to change the selection of games offered or assign the games to different keys on terminals which offer a choice of games. Whenever the terminal receives a full Configuration packet while in Full Configure Enable Mode, it shall zero all of the game-related accounting and statistical counters, but it shall not zero the main (Master or Period) accounting or terminal performance counters. The counters referred to are defined in the Monitor and Statistics data type sections.

If there is reason to either dis-enroll a terminal or suspend play, the GMMS will set the terminal into the Host-Disabled mode (3).

TERMINAL STATUS POLL TIMEOUT ENABLE Flag:

The Terminal Status Poll Timeout Enable function activates or deactivates the Status Poll Timeout detection as it applies to game play on the entire terminal. If the flag is set to 1, game play is to be disabled for the entire Gaming Machine whenever the terminal is not polled within the time provided in the Configuration data field (Status Poll Timeout - Secs). Note, 0 seconds is considered an infinite timeout period, or timeout disabled). If the Status Poll Timeout Enable flag is set to zero, the terminal is still expected to observe the Status Poll Timeout, but on a game-by-game basis as determined by the Game Enable Control flags in the Game Key Configuration sections.

DAILY POLL TIMEOUT ENABLE Flag:

The Daily Poll Timeout Enable function activates and deactivates the automatic Daily Poll Timeout shut down feature. When this bit is set to 1 (activated), the terminal must automatically disable game play if the Daily Poll Timeout Hours have passed without being rewritten by the GMMS host. Please refer to the following field for more explanation.

DAILY POLL TIMEOUT - HOURS:

This field reflects the number of hours remaining before the terminal must automatically disable game play in the case that it does not validly receive a new Control Data Write Request packet. Reception of a new Control Data Write Request will overwrite the remaining time currently reflected in this field and start the timeout process anew. If a daily poll timeout occurs, the terminal is expected to keep game play disabled until it receives a new Control Data Write Request.

The GMMS host provides this parameter during each daily poll and expects the GM to track elapsed time from the point it receives the last Control write packet. When read by the host, the field should accurately indicate the number of hours remaining before shutdown. Note, this is the only field in the Control data type that is updated by the GM. If the field is set to zero, the terminal is expected to disable all game play. The shutdown time is not to be extended by any means other than the GM receiving a Control Data Write Request, including shutting off power to the terminal. To do this, it is recommended that a projected shutdown time is calculated and stored at the time the parameter is received.

HOLIDAY FLAGS:

These bit flags indicate which days of the week have been identified as holidays. They are used in conjunction with the Configuration system and game parameters to determine the allowable hours of game play for each game during holidays.

bits	Description
7	Reserved
6-0	Sunday (bit 0) through Saturday (bit 6) flags

If the bit is set to one (1), that day is considered a holiday and each game must use enable/disable play times according to its Holiday Enable/Disable window periods.

MEMORY SIGNATURE DIVISOR:

This 2-byte field is a number which the terminal is expected to use to compute an error detection number called a Memory Signature Value. The Memory Signature calculation is computed over all program memory (PROM, EPROM, etc.) in the GM and the resulting value based on this field is returned in the Monitor packet. The terminal is required to begin this calculation whenever it receives a Control packet with a non-zero Memory Signature Divisor. Please refer to the Memory Signature description in the Monitor data type section and Appendix A: Memory Signature Value Algorithm for complete details of the expected operation.

Note, when a Memory Signature Value results from a Date and Time/Memory Signature Data Write Request, the value is not recorded in the Monitor field but is reported using a Date and Time/Memory Signature read. Also, the Memory Signature Divisor provided or reported in this Control field is not changed by a Date and Time/Memory Signature Data Write Request.

EVENTS LOG READ RECORD NUMBER (FROM):
This field reflects the record number as set by the GMMS to be used as the first record to be returned in the next Events Log Data Read Reply. When this number is changed by the host, it confirms that the previous records have been read and processed properly. Refer to the Terminal Events Log data type section for a complete description.

Note, this field does not reflect the starting point for the Events Log records to be returned in the next Site Info Reply. Rather, the Site Info Reply is always expected to return all Events Log records that have occurred since the start of the previous Universal day (including the Record Number/Date Marker that signifies the start of the previous day).

GAME KEY ENABLE Flags:
The Game Key Enable field is a 8-byte flag field used to individually enable/disable games which have been Configured into the Gaming Machine. The high order bit of the first byte corresponds to Game Key 1; bits 63 through 0 correspond to Game Keys 1 through 64, respectively. If the corresponding bit is set to 1, the game key may be enabled; if the bit is set to 0, the game key should be disabled. The GMMS currently only supports up to ten separate game key selections on a single terminal, but protocol provisions provide the means for future expansion.

The GMMS uses the Full Configuration function (with the Game Key Configuration sections of the Configuration data type) to assign specific games (of the games resident in the terminal) to each of the available game select keys. Alternatively, a manufacturer may choose to fix the game key assignments within the terminal and inform the regulating jurisdiction accordingly. In either case, the terminals are expected to respond to these Game Key Enable flag bits to allow the host to enable or disable each of the game keys, thus changing the apparent game offering without changing the game selection configuration.

VARIABLE BANNER DATA {optional}:
See Banner/Report data type section for a complete description of this field (127 characters maximum; a tilde ($7E) delimits fields).

5.4 CONFIGURATION

The Configuration data type serves to provide configuration information to selectively establish the identity and game offering of each individual GM. This information is typically changed by the GMMS very infrequently. The data segment consists of system related data as well as game-related data for each game select key on the terminal. The terminal is expected to support all features referenced in the system configuration data section. **If a terminal is incapable of supporting game key configuration parameters as sent by the host, it is expected to send a Configuration error NAK. All Configuration data fields, as sent or when read, must accurately reflect the subsequent or current GM configuration parameters whether fixed in the terminal or programmable by the host.**

A Full Reconfiguration is performed by the GMMS to initially establish the configuration of the terminal or to change the game type assigned to each game key on a terminal which supports this feature. The Full Reconfiguration is accomplished only when the terminal is in the Full Configure Enable mode (as established by the Control packet Terminal Mode field) and a valid Configuration packet is received by the GM. **This establishes a new definition for all games available on the terminal and the terminal is expected to zero all fields in the system that account for individual game keys, specifically:**

Monitor Game Acctg. Data fields: (Games Played and Won, Cents Played and Won)
Statistics Game Statistics Data fields: (all Win/Loss Statistics fields)

The Full Reconfiguration Reset Date and Time in the Statistics packet should also be updated. After the terminal receives the new value for the Beginning Events Log Record Number for Full Reconfigure, it is expected to re-initialize the Events Log and begin anew at the record marker specified. The Monitor Terminal Events Record Number data field should then reflect this record number. Note: The Master Accounting Data and Master Reset Date and Time in the Monitor data type must remain unchanged by any reconfiguration process.

The GM is expected to prevent entry into the Full Configure Enable mode when a non-zero credit balance exists and to NAK the Control Data Write Request with the "non-zero credit balance error" code. Similarly, when a terminal enters Full Configure Enable mode, it must inhibit any further play until another Control packet returns the terminal mode to Normal mode.

When the terminal is not in Full Configure Enable mode, it is expected to accept Configuration Data Write Requests that change system and game parameters, EXCEPT for the following fields:

1) Beginning Events Log Record Number for Full Reconfigure in the System Configuration section. (the terminal must ignore this field in this situation.)
2) Game Type Number in Game Key Configuration sections:

Note, when not in Full Configure Enable mode and a Game Type Number does not match the game type currently assigned to a particular game key, the terminal is expected to NAK the packet with a Configuration error NAK code. Also, when not in the Full Configure Enable mode, the terminal is expected to logically accept Configuration packets that only contain the System Configuration section and/or the Games Key Configuration sections that have fields which are being changed; in other words, some sections may be zero length. Note: Even though game keys are "configured", they may still be disabled/enabled by the Game Key Enable flags of the Control data packet.

The following table shows the Configuration data segment (Configuration Format Identifier $05) format as it applies to a Configuration Data Write Request, and to a Data Read Reply or Site Info Reply from the terminal:

Byte	**Format**	**Description**
Overhead Configuration section:		
1	Binary1	Configuration Format Identifier ($05)
2		Reserved
3	Binary1	Configuration Version
4	Binary1	Number of bytes of System Configuration
5		Reserved
6	Binary1	Count (n) of Game Key Configurations
If (n) = 0, the Overhead Configuration section will end at byte 6. If (n) is non-zero, the length of all Game Key Configuration Sections will be specified in subsequent Overhead Configuration fields. For example, if (n) = 2, the included fields are:		
7	Binary1	Number of bytes of Game Key #1 Configuration
8		Reserved
9	Binary1	Number of bytes of Game Key #2 Configuration
10		Reserved
The last Game Key Configuration will be specified with the following fields:		
(5 + 2*(n))	Binary1	Number of bytes of Game Key #(n) Configuration
(6 + 2*(n))		Reserved
System Configuration section:		
1-20	Ascii(20)	Location Name
21-60	Ascii(40)	Location Address
61-68	Ascii(8)	Permit Number
69-74		Reserved
75-76	Time(R)	Non-Holiday Terminal Enable Time
77-78	Binary2	Non-Holiday Terminal Enable Duration - Secs/2
79-80	Time(R)	Holiday Terminal Enable Time
81-82	Binary2	Holiday Terminal Enable Duration - Secs/2
83-84	Binary2	Hopper Payout Max - Coins
85-86	Binary2	Substantial Cashout Value - Dollars
87-88	Binary2	Substantial Win Value - Dollars
89-90	Time(R)	Day End Snapshot Time
91	Code	Week End Snapshot Day
92	Code	Time Zone Code/Daylight Savings Time Mode
93-94	Date(U)	Date to make Daylight Savings Time change
95-96	Time(U)	Time to make Daylight Savings Time change
97-98	Binary2	Share Percentage
99	Flags1	Transaction Enable Flags
100-102	Binary3	Beginning Events Log Record Number for Full Reconfigure
103		Reserved
104	Binary1	Status Poll Timeout - Seconds
105-108	Flags4	Exceptions Report Enable

Encryption Control subsection (considered part of System Configuration section):

109	Flags1	Encryption Control
110-116		Reserved
117-124	Hex(8)	Encryption Key #1
125-132	Hex(8)	Encryption Key #2
133-140	Hex(8)	Encryption Key #3 - reserved

Game Key Configuration sections (for each Game Key to be changed):

1-4		Reserved
5	Binary1	Game Type Number
6-20	Ascii(15)	Game Name
21-22	Time(R)	Non-Holiday Game Disable Time
23-24	Binary2	Non-Holiday Game Disable Duration - Secs/2
25-26	Time(R)	Holiday Game Disable Time
27-28	Binary2	Holiday Game Disable Duration - Secs/2
29	Binary1	Game ID Number
30	Flags1	Game Enable Control Flags
31-32	Binary2	Game Credit Size in Cents
33-34	Binary2	Max Bet per Game in Cents
35-38	Binary4	Max Award per Game in Cents
39-n		Additional Game-dependent parameters

Note: The Game Key Configuration data sections are sequentially appended, in the order of their key assignments, to the System Configuration data section. The GMMS will not currently support more than 10 game keys, but fields in the Overhead Configuration section provide protocol support for additional games. The terminal is expected to determine the starting point of the System Configuration section from byte 6 of the Overhead Configuration section (there will be 6+2*(n) bytes of Overhead before the System section starts). The length of the System Configuration data is indicated in byte 4 of the Overhead Configuration section and will be 140 bytes if encryption keys are sent. **The Encryption Control subsection is not to be returned in any Data Read Reply or Site Info Reply, so Replies should always have a System Configuration length of 108 bytes.**

OVERHEAD CONFIGURATION Data Section

CONFIGURATION VERSION:
This number is set by the host and must be returned by the GM in response to subsequent Configuration and Monitor read requests. It serves as confirmation of the last Configuration packet accepted.

NUMBER OF BYTES OF SYSTEM CONFIGURATION DATA:
The value of this field during a Configuration Data Write Request will typically be either 0 (System configuration unchanged and not sent), or 108 (no New Encryption Control), or 140 (entire System section contained). The GMMS expects only the first 108 bytes of this section in a Configuration Data Read Reply or Site Info Reply.

COUNT OF GAME KEY CONFIGURATIONS:
This field enumerates the number of Game Keys which are included in the Game Key Configuration sections. In practice, it identifies the number of the last Game Key included. For example, if Configuration data for Game Keys 1,2, and 9 is included, the count will be 9 and the Number of Bytes for Game Keys 3-8 will be set to zero.

NUMBER OF BYTES OF GAME KEY CONFIGURATION DATA:
Each of these bytes (one for each appropriate game key) contains the number of bytes of Configuration data to follow for each respective game key. If not in the Full Configure Enable mode, a Data Write Request from the GMMS may set this field to zero for game keys in which the game parameters are not being changed. On the other hand, a zero size field during a Full Reconfiguration identifies the respective key as having no game assigned to it. All configured Game Keys are to be returned in replies from the Gaming Machine.

SYSTEM CONFIGURATION Data Section

LOCATION NAME:
This ASCII field defines the name of the location (site or venue) as it is to appear on tickets printed by the terminal.

LOCATION ADDRESS:
This ASCII field defines the address of the location as it is to appear on tickets printed by the terminal.

PERMIT NUMBER:
This ASCII field defines the terminal permit number as it is to appear on tickets printed by the terminal.

NON-HOLIDAY TERMINAL ENABLE TIME:
This time field specifies the local time of day that the terminal is allowed to enable game play on days other than holidays (as identified by the Holiday flags in the Control packet).

NON-HOLIDAY TERMINAL ENABLE DURATION:
This field specifies the number of seconds divided by 2 (2 second increments) that the terminal is to be enabled on days other than holidays. Note, the field size provides a span of over 36 hours and the terminal is expected to observe enable periods that started on previous local days.

HOLIDAY TERMINAL ENABLE TIME:
This time field specifies the local time of day that the terminal is allowed to enable game play on days that are considered holidays (as identified by the Holiday flags in the Control packet).

HOLIDAY TERMINAL ENABLE DURATION:
This field specifies the number of seconds divided by 2 (2 second increments) that the terminal is to be enabled on days that are considered holidays. Note, the field size provides a span of over 36 hours and the terminal is expected to observe enable periods that started on previous local days.

HOPPER PAYOUT MAX - COINS:
In terminals that provide the capability to pay out with both a hopper and a printed cash ticket, the GM must restrict the hopper payout option on any individual "cash out" to the number of coins indicated in this field. If the current credit balance on the GM exceeds the value of this field, the only cash out option the terminal is to offer is to be a printed cash ticket. If, on the other hand, the hopper payout option was available and was selected by the player, the terminal is expected to complete the payout in coins even if a hopper refill is required.

If the GM offers only a hopper payout, a "hand pay" (by site personnel) must be executed if the credit balance on the GM exceeds this value.

SUBSTANTIAL CASHOUT VALUE - DOLLARS:
The terminal is expected to report any instance when a single Cashout (Cash Ticket, manual pay/cancel credit, or hopper payout) meets or exceeds the dollar value in this field. These events must be reported both in a Status Response (when the exception code is enabled) and in the Events Log (always).

SUBSTANTIAL WIN VALUE - DOLLARS:
The terminal is expected to report any single game win that matches or exceeds the dollar value in this field. These wins must be reported both in a Status Response (when the exception code is enabled) and in the Events Log (always).

DAY END SNAPSHOT TIME:
This field indicates the **local time** at which one accounting day is considered to end and the next day to begin. The terminal is expected to use this field to determine the time of day when the Day End Monitor (as well as the Week End Statistics) snapshot must be taken. A snapshot retains a separate record of all specified data at that time. The snapshot time may be set for shortly after the end of the business day, possibly 2:10 A.M ($0F3C in seconds divided by 2).

If there is a non-zero credit balance on the GM at the snapshot time, the snapshot shall still be taken without game play interference. In this situation, an entry must be made in the Events Log indicating the credit balance at the time.

If power is off at the time when a snapshot was to occur, the terminal must perform the snapshot at the time power is restored and record the snapshot date and time for the Monitor (and Statistics, if on week-end day) data types as the time the most recent snapshot should have occurred.

Likewise, when a full reconfiguration is completed, snapshots (both Monitor Day End and Statistics Week End) must be forced with the snapshot date and time recorded as the date and time of the reconfiguration.

WEEK END SNAPSHOT DAY:
This field defines on which day of the week (at the Day End Snapshot Time) to snapshot for beginning the next week and, conversely, closing out the previous week. It is used by the terminal to determine on what day **(based on local time)** to take its Week End Statistics snapshot.

Code		Day
$00	=	Sunday
$01	=	Monday
$02	=	Tuesday
$03	=	Wednesday
$04	=	Thursday
$05	=	Friday
$06	=	Saturday

TIME ZONE CODE/DST MODE:
Since the system may be operating over multiple time zones and, at times, under Daylight Savings Time, fields that are used for Date and Time stamps and synchronization settings are transferred in Universal Time. In cases where the terminal must convert from universal time to local time, this field, along with the subsequent "Date and Time to make DST change" fields, provides the parameters necessary for this conversion.

bits	Description
7-1	Signed binary (2's complement) number of half-hours to add to Universal Time to get local Standard Time.
0	Enable (1) daylight savings time, or Cancel (0) daylight savings time at specified date and time.

The flag (bit zero) is provided as the Time Zone Code to indicate whether to change to or from Daylight Savings Time at the Date and Time to Make DST Change field values. **Note:** If the Date and Time to Make DST Change has not yet passed and the flag to enable is set, the terminal is currently not on Daylight Savings Time. Similarly, if the Date and Time to Make DST Change has passed with the flag to enable set, the terminal is currently on DST.

DATE AND TIME TO MAKE DAYLIGHT SAVINGS TIME CHANGE:
These fields contain the Universal date and time on which the change to or from Daylight Savings Time is to occur.

TRANSACTION ENABLE FLAGS:
The eight flags in this field are used to enable or disable a subset of the Transaction types which may be defined for the system. These flag bits correspond to Transaction data type codes 0 through 7 (tttt = 0000 to 0111).

bits	Description
7-1	Reserved
0	Cashout Transactions

If the respective bit is set to 1, that particular transaction type is enabled.

If bit zero is set to 1, the terminal is not allowed to print a valid Cash Ticket or execute a cancel-credit operation (manual pay) without first performing a successful Cashout transaction with the GMMS.

SHARE PERCENTAGE:
This field may be used by the terminal to generate local reports which require a variable percentage rate. For example, South Australia may define it as the government (or venue) share of net revenue. The field format is: Commission Percentage * 100 (Range is limited to: 00.00% through 100.00%).

BEGINNING EVENTS LOG RECORD NUMBER for FULL RECONFIGURE:
The value included in this field during a Configuration Data Write Request is only to be accepted by the terminal if the terminal mode is set to Full Configure Enable. This identifies the 24-bit Events Log record number at which the GMMS expects the first record in the re-initialized Events Log. This field is expected to reflect the last Beginning Events Log Record Number received while the terminal was in Full Configure Enable mode. See Terminal Events Log data type section for further information.

STATUS POLL TIMEOUT - SECONDS:
The terminal is expected to use the value in this field to determine how much time is allowed between received Poll commands before game play must be disabled. The GM is to consider communication "lost" if no Poll command addressed to it is received by the GM within this time.

A zero value in this field is to be interpreted by the GM as an infinite Status Poll Timeout value; that is, the terminal is expected not to time out based on the frequency of Poll commands. Zero is the recommended GM default value for this field.

When this field is non-zero, the Terminal Status Poll Timeout flag in the Control packet determines if the disable condition must be applied to the entire terminal (all games) or only to individual games. If the individual game basis is selected, the Status Poll Timeout Enable - Game flag within each Game Key Configuration section establishes the desired action.

Note, during normal communication, a forced Poll will be interspersed in the poll cycle to verify communication integrity if no other activity has been occurring. After the GMMS perceives a status poll timeout, the terminal will be sent a forced Poll on each subsequent poll cycle until communication is re-established, or until the terminal is removed from or relocated in the Host database. The only time that a GM will be sent multiple consecutive forced Poll commands (f bit set) is during an attempt to establish communication. **Therefore, the receipt of three (3) consecutive forced Polls by the GM is also to be interpreted as a Status Poll Timeout by the GM.** In this way, the GMMS will inform the GM of the occasion when it is not receiving responses, while the GM itself is responsible to determine when it is not being polled within the required time. Note, if the total communication link to a GM fails, the GM will not receive Polls; in the case of either the transmit or receive link experiencing a failure, one of the two conditions above will provide indication of a Status Poll Timeout to the GM.

When a site is so configured, local GMMS components (Site Controller) will assume the real-time task of reporting terminal communication status changes to the central site, but the GM is required to make entries into the Events Log when it perceives communication as lost and when it detects communication as restored or established. The GM is to consider communication established (or re-established) when it receives a Poll command with the forced poll bit reset (f bit = 0).

When the terminal recognizes a Status Poll Timeout, the terminal is typically expected to disable game play at the completion of any current game, make the appropriate entry into the Events Log, and display a suitable message. In the case where Cashout transactions are enabled, the terminal will obviously not be able to complete the printing of a Cash Ticket or cancel credit (manual pay) operation until communication is re-established. To cover cases of extended communication loss, a GM administrative means to lock out all game play and cashouts may be incorporated; this would

allow site personnel to pay the player and later adminstratively re-enable the GM and initiate the cashout when communication is intact. After a Status Poll Timeout, the GM must not re-enable game play until a Poll command indicates an acknowledged response (v bit) and then only if the Disable Flag (d bit) in the Poll is set to allow enabling.

EXCEPTIONS REPORT ENABLE Flags:
The 32 flags in this field are used to select the exception codes which must be reported in Status Responses. A one in the bit field enables the exception code category, while a zero in the bit field indicates that the category does not need to be queued or reported by the Gaming Machine. Exceptions codes that do not have a flag assigned ($00, $02, $0B) are always reported.

bits	Description	Exception Codes
	SIGNIFICANT EVENTS:	
31	Reserved	
30	Playable and Non-Playable State changes	$03 - $05
29-28	Reserved	
27	Events Log Full	$0C
26	Daily Poll Communication Timeout	$0D
25-24	Reserved	
	SPECIAL CONDITIONS:	
23	Tilts	$11, $13, ($10)
22	Paper Low, Paper Out	$15, $16, ($10)
21	Hopper Empty, Hopper Jam	$19, $1B, ($10)
20	Reel Tilt (mechanical reels)	$1D, ($10)
19	Coin Diverter Error, Hopper Excess Coin	$1E, $1F
	DOOR ACCESSES:	
18	Door Accesses with Audit Key OFF	$20 - $24
17	Door Accesses with Audit Key ON	$28 - $2C
	ADMINISTRATIVE/DIAGNOSTICS ACCESS:	
16	Administrative/Diagnostics Access	$27 & $2F
	FAULTS:	
15	All Faults	$30 - $47
14	Reserved	
	LOCAL SITE ITEMS:	
13	Player Requests	$48 - $4B
12-11	Reserved	
	SUBSTANTIAL Categories:	
10	Substantial Wins	$51
9	Substantial Cash Out	$52
8-0	Reserved	

ENCRYPTION CONTROL AND KEYS:

The Encryption Control subsection of the Configuration data segment contains the Data Encryption Standard (DES) encryption keys and flags used to control which segments, if any, of all data packets are encrypted. When any encryption is enabled, the National Bureau of Standards Data Encryption Standard (DES) in Electronic Codebook Mode is used.

Encryption on some data and validation segments can be enabled independently. In addition, encryption of the data in the Encryption Control subsection (bytes 109 - 140 of the System Configuration section) can be separately controlled when encryption on data segments (Encryption Category #1) is not enabled. Which segments are encrypted is determined by the current Encryption Control flags, represented by byte 109 in the System Configuration data segment.

Encryption Category #1 segments include the Data Segments of all Data Read Replies and Data Write Requests except for the Date and Time / Memory Signature data type.

Encryption Category #2 segments include the Validation Segments of ALL data types and the Data Segments of the Transaction Request/Reply, Site Info Request/Reply, the Date and Time / Memory Signature Data Read Reply and Data Write Request.

System Configuration bytes 109 - 140 is the Encryption Control subsection of the Configuration data type when included in a Data Write Request.

According to the Encryption Control flags, encryption is enabled in the following manner:

flag bits 7 1 0	Encryption Category #1 Data Segments	Encryption Category #2 Segments	System Configuration bytes 109-140
1 0 0	OFF	OFF	OFF
0 0 0	OFF	OFF	ON
x 0 1	ON	OFF	(ON with data)
x 1 1	ON	ON	(ON with data)
0 1 0	OFF	ON	ON
1 1 0	OFF	ON	OFF

Note, bit 7 controls (inverse polarity) the Encryption Control section except that its encryption is always On when Category #1 is turned ON; the other five flag bits in byte 109 are reserved.

When the corresponding encryption is enabled, the two eight-byte encryption keys currently used are selected in the following manner:
Encryption Key #1 is used for Encryption Category #1 segments and/or for the Encryption Control subsection if included in a Configuration Data Write Request.
Encryption Key #2 is used for Encryption Category #2 segments.

A pristine terminal is expected to have all Encryption Control flags set to 0 and Encryption Key #1 set to a default key. The 64-bit default value for Encryption key #1 is created by appending the 16-bit Terminal ID to the end of a 16-bit manufacturer-specific number and generating the 64-bit square of that combined number. The standard form of the manufacturer-specific number is a one-byte manufacturer number, assigned by South Australia, followed by a one-byte system number ($21 for South Australia). The default for Encryption keys #2 and #3 is $00. Bit 0 of each byte of the encryption keys is an optional parity bit. The GMMS will ignore this bit and terminals must do likewise.

A terminal must maintain two sets of Encryption Control information (including the flags and keys), a current set and a pending set. The GM always uses the current set for encryption / decryption. The keys received in a Configuration Data Write Request command always go into the pending set.

The pending set of keys are used as shown in Figure 6. If the terminal finds that the validation code check on a validation segment fails with the current keys, then, if there is a set of pending keys, it re-checks the validation using the pending set of keys. If, and only if, the validation code checks, the entire pending set of keys replaces the current set of keys. If the check fails with the current set and there is either no pending set or the pending set also fails, the terminal reports a validation code error.

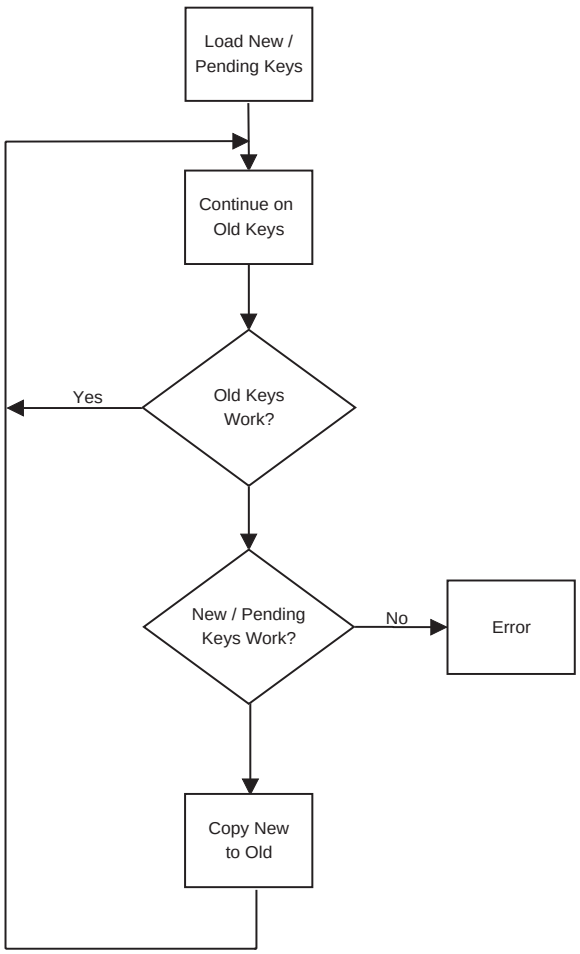


Figure 5: Encryption Keys Flowchart

For recovery purposes, a secure means must be provided at the terminal for a qualified person to force the current keys back to the default values. This feature should require access to the logic area. The preferred implementation is to have this action also move the current keys to the pending keys before setting the current keys to default values. In this way, host access may be successful

if it uses either the previous current keys (now the pending keys) or the default keys (now the current keys).

When the terminal is first enrolled on a system, the host will generally change the encryption keys, set Encryption Category #2 segments for encryption, and establish whether or not the Encryption Category #1 segments are to be encrypted. The terminal must not support any other means for entering or displaying the encryption keys. The only means by which they are entered into the terminal is through communication from the host.

GAME KEY CONFIGURATION Data Sections:

The following game data is supplied for all of the game keys on the terminal (the GMMS is currently limited to a maximum of 10 keys).

GAME TYPE NUMBER:
The game type included in each game segment is the terminal's own program identification code for a particular type of game, such as poker. In terminals that support this, the same game type may be assigned to more than one game select key where each has a different set of parameters, including the game name and/or the credit size, and so on. The GMMS uses an additional field, the Game ID Number, to provide the distinction between game key assignments with the same game type.

The GM manufacturer is free to assign Game Type Numbers as desired (within the permissible range of $01 through $FA) and provide this information to South Australia for entry into the GMMS database. It is recommended that manufacturers use Game Type Numbers $E0 through $FA only for games or game options which have manufacturer-assigned Game ID Numbers (see Game ID Number below and "Non-Configured Games" at the end of this section).

Game Type Numbers $FB through $FF are reserved to configure and control non-game features which have no game accounting, such as logos. If there is no game for a game select key, the Game Type is expected to be zero.

GAME NAME:
This is typically the game name that appears on the game select key.

NON-HOLIDAY GAME DISABLE TIME:
This field specifies a local time of day at which the terminal is expected to start a game play disable period on this individual game for days that are not holidays. These game disable conditions are to take precedence over Non-Holiday Terminal Enable periods; so, when these fields have significance, they will typically establish a disable period within the Non-Holiday Terminal Enable period to provide a split enable period for that game.

NON-HOLIDAY GAME DISABLE DURATION:
This field specifies the number of seconds divided by 2 (2 second increments), starting at the Non-Holiday Game Disable Time, that the individual game is to be disabled on days that are not considered holidays. In combination with the Non-Holiday Terminal Enable period, the option of a split enable period is provided. Note, the field size provides a span of over 36 hours and the terminal is expected to observe disable periods that started on previous local days.

HOLIDAY GAME DISABLE TIME:
This field specifies a local time of day at which the terminal is expected to start a game play disable period on this individual game for days that are considered holidays. These game disable conditions are to take precedence over Holiday Terminal Enable periods; so, when these fields have significance, they will typically establish a disable period within the Holiday Terminal Enable period to provide a split enable period for that game.

HOLIDAY GAME DISABLE DURATION:
This field specifies the number of seconds divided by 2 (2 second increments), starting at the Holiday Game Disable Time, that the individual game is to be disabled on days that are considered holidays. In combination with the Holiday Terminal Enable period, the option of a split enable period is provided. Note, the field size provides a span of over 36 hours and the terminal is expected to observe disable periods that started on previous local days.

GAME ID NUMBER:
The Game ID Number is defined by GMMS administrative personnel to uniquely identify a game and its parameters. This Game ID Number must be returned to the GMMS in Data Read Replies and Site Info Replies of Monitor data and Statistics data. Game ID Numbers assigned by administrative personnel will be restricted to the range of $01 to $DF (with $00 considered as no game). If game options (such as Double-Ups) are financially tracked separately from the Configured games and no Game Key Configuration data section defines the option (such as when 10 Game Keys would be exceeded in the current system), the manufacturer is expected to assign a Game ID Number of $E0 through $FF. This alerts the Central System that no Configuration is associated with this "game" when Game Accounting information is returned in the Monitor packet (see Monitor Section 5.5 and "Non-Configured Games" at the end of this section).

Terminals in which the game name and other parameters are configurable by the host may offer multiple games of the same Game Type Number. That is, they may be using the same firmware program with different parameters. The Game ID Number will provide the distinction.

GAME ENABLE CONTROL FLAGS:
This flag field is defined to provide enable/disable options on a game-by-game basis.

Bit	**Description**
7-5	Reserved
4	Status Poll Timeout Enable - Game 1 = Enable Timeout (Game Key is required to observe Status Poll timeout) 0 = Disable Timeout (Game Key is allowed to operate beyond Status Poll timeout)
3-0	Reserved

The Status Poll Timeout Enable flag is to be observed when the Terminal Status Poll Timeout flag in the Control packet is set to zero. If the entire terminal is required to disable due to a Status Poll Timeout (Control flag set and non-zero Status Poll Timeout - Seconds in the Configuration), this flag is overridden. On the other hand, if the Terminal Status Poll Timeout flag is cleared (with a non-zero Status Poll Timeout - Seconds), the individual games are expected to observe a Status Poll Timeout based on this flag.

GAME CREDIT SIZE:
This field defines the value of each credit in cents for the game.

MAX BET PER GAME:
This field sets and/or reflects the maximum allowable bet on the game. This field is specified in cents.

MAX AWARD PER GAME:
This field sets and/or reflects and the maximum allowable award for the top win category(s) on the game. The award value is specified in cents. Any game award that matches this amount must be reported in the Events Log and as an exception event in a Status Response.

GAME DEPENDENT PARAMETERS {optional}:
Additional programmable game parameters may be specified in additional fields. The manufacturer may define these fields as required by the individual game. They may be used to configure such features as enabling/disabling "Hold Recommendations" in Poker or any other game options or attributes identified by the manufacturer.

Non-Configured Games

Currently, the GMMS has a limitation of 10 configurable games; that is, the Configuration packet can specify or configure up to 10 games or Game Keys on a Gaming Machine and the Statistics packet can only include game statistics for those configured games. VLC has plans to expand the Configuration and Statistics capacity in the future, but for the time being, a method is provided to financially track specific games beyond the 10 configured games on a terminal.

Within the GM, the explicitly configured games may, as a normal course of play, offer the player a chance to risk all or part of the last win in a separate "sub-game" or double-up. If these sub-games are only made available to the player during the play of some configured game, and are never directly accessible, these sub-games may be considered as attached to another game or games. (This attachment relationship may be established, if desired, by using the Game Dependent Parameters in the Game Key Configuration sections).

As a means for the GMMS to identify and separately track the financial activity of these sub-games, a special range of Game Types and Game ID's are to be assigned by the manufacturer. Both of these fields must be unique for each sub-game offered by the manufacturer and are both expected to be in the range of $E0 to $FA. When the GM sends the Monitor data, a Game Accounting Data Section should be sent for each active sub-game in the GM (transmitted in any order after the Configured games).

In summary, sub-games which are not Configured into the terminal must observe the following guidelines:

1) The sub-game is accessible to the player only through some other Configured game.
2) The sub-game complies, where applicable, to the parameters defining the associated Configured game (Credit Size, Max Award, Enable/Disable status, etc.).
3) No statistical information is reported for the sub-game in the Statistics packet.

4) Monitor Game Accounting meters are reset when a Master Reset or Full Reconfiguration takes place.

5.5 MONITOR

The read-only Monitor data type contains the basic accounting and performance data that is read from the terminal on a regular basis, typically daily. There are two Monitor read data types, Current and Last Day End; these can be included in both a Data Read Reply and Site Info Reply:

MONITOR - CURRENT:
The Current Monitor (data type 0101) provides the Monitor data field values at the time the Reply is returned by the terminal.

MONITOR - LAST DAY END:
The Last Day End Monitor (data type 0001) provides accounting Monitor data field values at the time of the last Day-End Snapshot; the snapshot time is set by the Configuration data type.

If power is off at the time when a snapshot was to occur, the terminal must perform the snapshot at the time power is restored and record the snapshot date and time for the Monitor data types as the time the snapshot should have occurred. Likewise, after a full reconfiguration is performed, snapshots (both Monitor Day End and Statistics Week End) must be forced with the snapshot date and time recorded as the date and time of the reconfiguration.

The following table shows the contents of the Monitor data segment (Monitor Format Identifier $05) as it applies to either a Current or Day End Monitor Reply. Immediately following this table is a detailed description for each Monitor-related field.

MONITOR - Current or Last Day End:

(Monitor data always Current data):

Byte	**Format**	**Description**
1	Binary1	Monitor Format Identifier ($05)
2		Reserved
3-4	Code2	System Mode/Failure Code
5	Code1	Active Game ID#
6-8	Binary3	Terminal Events Record Number (BASE)
9-10	Flags2	Status Flags
11-12	Flags2	Subassembly Failure Flags
13-14	Flags2	Operator Flags
15-16	Binary2	Paper Level
17	Binary1	System Firmware Version
18	Binary1	Game Firmware Version
19	Binary1	Control Version
20	Binary1	Configuration Version
21	Binary1	Banner Version
22	Binary1	Alt Banner Version
23	Binary1	Report Version
24-26		Reserved
27-28	Binary2	Memory Signature Value

29-32	Reserved

(Monitor data contingent on request - Current or Last Day End):

(Accounting Date/Time)			
33-34	Date(U)	Date for Accounting Data (Current or Last Day End snapshot)	
35-36	Time(U)	Time for Accounting Data (Current or Last Day End snapshot)	
(Master Accounting Data)		*** (all fields)**	
37-38	Date(U)	Master Reset Date	
39-40	Time(U)	Master Reset Time	
41-44	Binary4	Bills IN - Cents	
45-48	Binary4	Coin IN Drop Box - Cents	
49-52	Binary4	Coin IN Hopper - Cents	
53-56	Binary4	Player (Debit) Card IN - Cents {FUTURE}	
57-60	Binary4	Hopper Fills - Cents (attendant input)	
61-64	Binary4	Product IN - Cents (attendant input) {optional}	
65-68	Binary4	Bills Collected - Cents (collected)	**
69-72	Binary4	Drop Box Collected - Cents (collected)	**
73-76	Binary4	Coin OUT Hopper - Cents	
77-80	Binary4	Player (Debit) OUT - Cents {FUTURE}	
81-84	Binary4	Cash Tickets and Manual Pays - Cents	
85-88	Binary4	Product OUT - Cents {optional}	
89-92	Binary4	Total PLAYED - Cents	
93-96	Binary4	Total WON - Cents	
(Game Accounting Data)			
97	Binary1	Game ID Number for first included Game	
98	Flags	Game Status	
99-101	Binary3	Games Played	* / ***
102-104	Binary3	Games Won	* / ***
105-108	Binary4	Cents Played	* / ***
109-112	Binary4	Cents Won	* / ***
113-128	Same as 97-112 but for 2nd included Game		
129-n	Same as 97-112 but for 3rd through Last included Game		

***** Field Will **Reset** at time of a **Master Reset**
****** Field Will **Update** at time of a **Period Reset**
******* Field Will **Reset** at time of a **Full Reconfiguration**

The monitor packet must include all games that are available on the terminal and the packet may either end after the last included game or have zero for the Game ID# in any additional Game Accounting Data sections that are sent. For example, if a terminal has only one game, the data segment may end after byte 112. If Game Accounting Data is sent for 10 games, the Monitor data segment will be 256 bytes long. The gaming machine is not to include any Game Accounting Data sections (starting at byte 97) in Data Read Replies until games have been "configured" onto the machine by the GMMS.

Although the GMMS cannot currently Configure more than ten games, Game Accounting Data can be sent for more than 10 games through the use of Continuation packets. If Accounting Data is included for games that are not Configured to a Game Key, the Game ID Number is to be assigned by the terminal manufacturer and be in the range of $E0 to $FF (see "Non-Configured Games at the end of the Configuration section).

SYSTEM MODE/FAILURE CODE:
This field is used to show the terminal's current internal operating mode and to report the current existence of System Failure Faults as defined by the Status Response exception codes. A non-zero value in the System Failure Fault Code indicates a condition in which play is not allowed. The standard format is:

byte 1 - System Terminal Mode
$00 = Normal Mode, $01 = Full Configure Enable Mode, $02 = Demo Mode, $03 = Host-Disabled Mode, $04 = other disable condition

byte 2 - System Failure Fault Code ($00 = no System Failures currently exist)
$31 - $47 is to correspond to the System Failure Faults exception codes from the Status Response definition.

ACTIVE GAME ID#:
This field is used to identify the game (if any) which is currently being played at the terminal. The contents are expected to be: Game ID Number ($00 = no game currently being played)

TERMINAL EVENTS RECORD NUMBER (BASE):
This field indicates the record number at which the terminal expects the next Events Log Read to begin (BASE). The GM is expected to update this field after an Events Log Read by the GMMS. See Terminal Events Log data type section for a complete description.

STATUS FLAGS:
These flags indicate the current status of various conditions of the terminal. A one (1) in any of the following bits indicates that the condition exists. The standard assignments are:

Bit	**Description**
15 -	Events Log Full
14 -	Daily Poll Timeout shutdown
13 -	Hopper Out
12 -	Hopper Low
11 -	Special Report Print Pending {optional}
10 -	Paper Out
9 -	Paper Low
8 -	GM-detected disable condition
7 -	Product Empty (such as Instant Tickets) {optional}
6 -	GM Host-Disabled Mode is active
5 -	GM set to Modem Master
4 -	Audit Key On (Terminal in Administrative/Diagnostics state)
3 -	Area 4 Open (Logic Door)
2 -	Area 3 Open (Secondary Door)
1 -	Area 2 Open (Cash Door)
0 -	Area 1 Open (Main Door)

SUBASSEMBLY FAILURE FLAGS:
These flags are used to indicate active failures in a terminal subassembly. A one (1) indicates a failure.

Bit	Description
15-13	Reserved
12	Coin Acceptor (failure, tilt, or jam)
11	Hopper (failure or empty)
10	Expansion Board {optional}
9	Communications
8	Printer
7	Product Dispenser {optional}
6	Bill Acceptor {optional}
5	Optical Mark Sense Reader {optional}
4	Touch Screen {optional}
3	Video
2	Sound
1	I/O Logic {optional}
0	Power Supply

OPERATOR FLAGS {optional}:
These flags can used to report operator-selected conditions to alert the host of mutually-understood specific needs or conditions.

PAPER LEVEL:
This field is used to report the amount of printer paper remaining. The standard format is the number of lines remaining with 0000 indicating "paper OK" and $FFFF indicating out of paper.

SYSTEM FIRMWARE VERSION:
Version number of the manufacturer's firmware currently installed in the terminal. Each variation of system firmware offered by a manufacturer is expected to be assigned a unique non-zero version number. If the terminal's system and game firmware are common, the version number belongs in the System Firmware Version field, and Game Firmware Version should be set to 0.

GAME FIRMWARE VERSION:
Version number of current game firmware in the terminal. Each variation of game firmware offered by a manufacturer is expected to be assigned a unique non-zero version number.

CONTROL VERSION:
Version number of last accepted Control packet. The version number was sent in the last Data Write Request of the Control packet.

CONFIGURATION VERSION:
Version number of last accepted Configuration packet. The version number was sent in the last Data Write Request of the Configuration packet.

BANNER VERSION: {optional}
Version number of current Main Banner loaded in the terminal.

ALT BANNER VERSION: {optional}
Version number of current Alternate Banner loaded in the terminal.

REPORT VERSION: {optional}
Version number of last accepted Report loaded in the terminal.

MEMORY SIGNATURE VALUE:
The Memory Signature Value returned in this field is computed based on the Memory Signature Divisor provided in the Control packet. The Memory Signature Value is the result of a calculation designed to detect memory failures, to prevent tampering, and to ensure general memory integrity. The algorithm for computing the Memory Signature Value is defined in Appendix A.

Whenever the terminal receives a Control packet, it is expected to calculate a Memory Signature Value using the Memory Signature Divisor contained in that packet. Typically during a daily poll cycle, a Control write is followed closely by a Monitor read, and it is recognized that the calculation may require some time. Therefore, when the Memory Signature Divisor was not changed and recalculation is not complete, the terminal is expected to return the previously calculated Memory Signature Value. **However, even if the Memory Signature Divisor sent in the Control packet has not changed, the terminal still must recalculate the Memory Signature Value.**

If the Memory Signature Divisor received in a Control packet is different than the previous one, the terminal is expected to return $FFFF in Monitor Data Read Replies and Site Info Replies while computation using the new Divisor is still underway.

Note: In a similar manner, whenever a non-zero Memory Signature Divisor is received in a Date and Time/Memory Signature Data Write Request, the GM is required to compute an appropriate Memory Signature Value and, when completed, return the result in a subsequent Date and Time / Memory Signature read (Data Read Reply and/or Site Info Reply). In this case, it does not matter whether the Divisor has changed; a new Value must be calculated and any Date and Time / Memory Signature Reply should contain $FFFF while the computation is in progress. Note, a Status Response with Exception Code $02 is to be generated when this calculation has been completed. **The Memory Signature Divisor and Value associated with this process are <u>not</u> to be reflected in the Monitor and Control fields.**

DATE AND TIME FOR MONITOR ACCOUNTING DATA:
These fields reflects the Universal date and time for the accounting data that follows. For Current Monitor data, this is the current date and time. For Day-End Monitor data, this will be the date and time of the last day-end snapshot.

MASTER ACCOUNTING DATA

The following fields hold the master accounting data. Through the use of these values, the total net income for the terminal can be determined. In addition, if proper procedures are followed by site personnel, the following is known about each machine at any given time:

total currency in terminal (bills, coins in hopper and drop box); value of bills in terminal; value of coins in hopper; value of coins in drop box; value of Product (Instant or Scratch tickets, etc.) in terminal.

MASTER RESET DATE AND TIME:
This field indicates the Universal date and time that the master accounting fields were last zeroed. It reflects the date and time the host brought the terminal out of Demo Mode. This is expected to change at a terminal only if it is necessary to install a new main logic board.

Master BILLS IN - CENTS: {optional}
Total value of cash (bills) accepted by the bill acceptor.

Master COIN IN DROP BOX - CENTS:
Total value of coins accepted through the coin acceptor(s) and diverted into the drop box (collected coin area).

Master COIN IN HOPPER - CENTS:
Total value of coins accepted through the coin acceptor(s) and diverted into the hopper(s).

Master PLAYER (DEBIT) CARD IN - CENTS: {FUTURE}
Total value of credit accepted on the terminal through player card or debit card transactions. This field must be zero as long as player cards are not supported.

Master HOPPER FILLS - CENTS:
Total value of coins that have been put directly into the hopper by site attendants.

Note, GM manufacturers are expected to provide a secure yet convenient means for site personnel to record into the terminal the value of each hopper fill.

Master PRODUCT IN - CENTS: {optional}
Total value of product (Instant or Scratch tickets, etc.) that has been loaded into the machine by site attendants. If product is not supported, this field must be zero. If product is supported, GM manufacturers are expected to provide a secure yet convenient means for site personnel to record into the terminal the value of each product fill.

Master BILLS COLLECTED - CENTS: {optional}
Total Value of cash (bills) collected from the terminal by site attendants. This field will reflect the value of the Master BILLS IN field at the time of the last Period Reset. Site attendants are expected to initiate a period reset of the terminal at each cash collection. (The value of bills from each collection should equal BILLS IN minus BILLS COLLECTED prior to Period Reset.)

Master DROP BOX COLLECTED - CENTS:
Total value of coins collected from the drop box of the terminal by site attendants. This field will reflect the value of the Master COIN IN DROP BOX field at the time of the last Period Reset. Site attendants are expected to initiate a period reset of the terminal at each cash collection.
Master COIN OUT HOPPER - CENTS:
Total value of the coins dispensed by the hopper.

Master PLAYER CARD (DEBIT) OUT - CENTS: {FUTURE}
Total value of credit granted through Player Card cash out transactions. This field must be zero as long as player cards are not supported.

Master CASH TICKETS AND MANUAL PAYS - CENTS:
Total value of cash tickets printed and manual pays made (cancel-credit operations performed).

Master PRODUCT OUT - CENTS: {optional}
Total of value of product (Instant or Scratch tickets, etc.) sold or dispensed.

Master TOTAL PLAYED - CENTS:
Total value of credits played or bet on all games combined.

Master TOTAL WON - CENTS:
Total value of credits won or awarded on all games combined.

GAME DATA:

A set of the following game data information must be supplied for each of the games included on the terminal.

GAME ID #:
This field reflects the Game ID Number which is assigned to this game as provided in the Configuration data type. For games that are not Configured, the Game ID Number is to be assigned by the terminal manufacturer and be in the range of $E0 to $FF ("different" games must be assigned numbers which are unique within the manufacturer's game selection).

GAME STATUS:
This field is used to reflect the status of each individual game. If detection is supported by the terminal, the respective bit is set to 1 by the GM to indicate that the condition exists. The standard game status assignments are:

Bit	Description
7 -	Invalid Game Configuration
6,5 -	Reserved
4 -	Game Check Sum Error
3 -	Reserved
2 -	Game-determined Inhibit
1 -	Reserved
0 -	Disabled by Game Key Enable flags (Control parameter)

GAMES PLAYED:
Total number of games played.

GAMES WON:
Total number of games that won awards.

CENTS PLAYED:
Total number of cents bet on the game.

CENTS WON:
Total number of cents awarded by the game.

5.6 STATISTICS

The read-only Statistics data type contains detailed game play statistics and terminal related performance data. Separate data types are provided to read either the Statistics - Current (data type 0111) data or the Statistics - Last Week End (data type 0011) snapshot data. The day for the week-end snapshot is determined by the corresponding Configuration data type field.

If power is off at the time when a snapshot was to occur, the terminal must perform the snapshot at the time power is restored and record the snapshot date and time for the Statistics data types as the time the snapshot should have occurred. Likewise, after a full reconfiguration is performed, snapshots (both Monitor Day End and Statistics Week End) must be forced with the snapshot date and time recorded as the date and time of the reconfiguration.

The following table shows the contents of the Statistics data segment (Statistics Format Identifier $05) as it applies to either a Current or Week-End Statistics Data Read Reply or Site Info Reply. The 44 bytes of System Statistics data is always current data, even for Week-End Statistics.

STATISTICS - Current or Last Week End

OVERHEAD STATISTICS Data Section:

Byte	Format	Description
1	Binary1	Statistics Format Identifier ($05)
2-3		Reserved
4	Binary1	Number of Bytes of System Statistics Data
5		Reserved
6	Binary1	Count (n) of Game Key Statistics

If (n) = 0, the Overhead Statistics section will end at byte 6. If (n) is non-zero, the length of all Game Statistics Sections will be specified in subsequent Overhead Statistics fields. For example, if two Game Keys have been Configured, the (n) will equal 2 and the included fields are:

7	Binary1	Number of bytes of Game Key #1 Statistics
8		Reserved
9	Binary1	Number of bytes of Game Key #2 Statistics
10		Reserved

The last Game Statistics section will be specified with the following fields:

(5 + 2*(n))	Binary1	Number of bytes of Game Key #(n) Statistics
(6 + 2*(n))		Reserved

SYSTEM STATISTICS Data Section (always current data):

1-2	Date(U)	Full Reconfiguration Reset Date (Game Acctg & Stats Reset)
3-4	Time(U)	Full Reconfiguration Reset Time (Game Acctg & Stats Reset)
Tilt Counters		
5-6	Binary2	Tilt Source 1 - Left Coin Slot
7-8	Binary2	Tilt Source 2 - Right Coin Slot
9-10	Binary2	Tilt Source 3 - Bill Acceptor
11-12	Binary2	Tilt Source 4 - Reserved

Ram Fault Counters

13-14	Binary2	RAM 0 Errors - Main Ram
15-16	Binary2	RAM 1 Errors - Backup RAM 1
17-18	Binary2	RAM 2 Errors - Backup RAM 2
19-20	Binary2	RAM 3 Errors - Expansion RAM

Door Access Counters

21-22	Binary2	Door 1 Accesses - Main Door
23-24	Binary2	Door 2 Accesses - Cash Door
25-26	Binary2	Door 3 Accesses - Secondary Door
27-28	Binary2	Door 4 Accesses - Logic Door

Faults Counters

29-30	Binary2	CRC Errors
31-32	Binary2	Validation Time Stamp Errors
33-34	Binary2	Validation Code Errors
35-36	Binary2	System Failure Faults (Status Response exception codes $31-47)

Resets Counters

37-38	Binary2	Full Reconfiguration Resets
39-40	Binary2	Period Resets

Date and Time of Game Statistics Data (contingent on request - Current or Last Week End):

41-42	Date(U)	Date of Game Statistics Data (Current or Last Week End)
43-44	Time(U)	Time of Game Statistics Data (Current of Last Week End)

GAME STATISTICS Data Sections for each game key:

1-2	Binary2	Reserved
3	Binary1	Game ID Number
4	Binary1	Game Type Number
5-x	******* Statistics fields as required to track the occurrence of each win and loss category within the game for Game Key #1.	

******* Will **Reset** at time of a **Full Reconfiguration**

x+1 - - Game Statistics data sections for Game Keys 2-n, respectively.
(n is currently limited to 10 games)

Note: The Game Key Statistics data sections are sequentially appended to the System Statistics data section in the order of their Configuration Game Key assignments. The GMMS will not currently support more than 10 game keys, but fields in the Overhead Statistics section provide protocol support for additional games. The terminal is expected to determine the starting point of the System Statistics section from byte 4 of the Overhead Statistics section. The length of the System Statistics data is indicated in byte 2 of the Overhead Statistics section and is currently defined as 44 bytes.

The gaming machine is not to include any Game Statistics Data Sections in Data Read Replies until games have been "configured" onto the machine by the GMMS.

OVERHEAD STATISTICS Data Section

NUMBER OF BYTES OF SYSTEM STATISTICS DATA:
This field indicates the number of bytes of System Statistics data included and is currently expected to always be 44 bytes.

COUNT OF GAME KEY STATISTICS:
This field enumerates the number of Game Keys which are included in the Game Key Statistics sections. In practice, it identifies the number of the last Game Key included. For example, if Configuration data for game keys 1,2, and 9 is included, the count will be 9 and the Number of Bytes for Game Key 3-8 will be set to zero.

NUMBER OF BYTES OF GAME KEY STATISTICS DATA:
Each of these fields (one byte for each appropriate game key) contains the number of bytes of Game Statistics data that are included for each game key. The number of bytes needed by each game is game-dependent as defined by the GM manufacturer.

SYSTEM STATISTICS Data Section

FULL RECONFIGURATION RESET DATE AND TIME:
These fields show the date and time of the last Full Reconfiguration. This is expected to reflect the date and time the Game Statistics were last zeroed. The games fields are reset whenever the terminal receives a full Configuration packet while in Full Configure Enable Mode.

TILT COUNTS:
These four counters show the total number of tilts detected by cash acceptance devices.

Tilt Source 1 - Left Coin Slot
Tilt Source 2 - Right Coin Slot
Tilt Source 3 - Bill Acceptor
Tilt Source 4 - Reserved

RAM FAULTS:
These four counters show the number of RAM faults, categorized by RAM areas or by fault type. These fields are expected to tally any non-fatal RAM errors that can be corrected by the GM without resulting in a system failure. The standard assignments are:

RAM 0 Errors - Main RAM
RAM 1 Errors - Backup RAM #1
RAM 2 Errors - Backup RAM #2
RAM 3 Errors - Expansion RAM

DOOR ACCESSES:
These counters show by type the total number of door accesses detected. Door assignments are:

Door 1 - Main Door
Door 2 - Cash Door
Door 3 - Secondary Door
Door 4 - Logic Door

FAULTS:
These four counters show the number of faults, categorized by type. The assignments are:

Packet CRC Errors
Validation Time Stamp Errors
Validation Code Errors
System Failure Faults (Status Response exception codes $31-$47)

RESETS:
These two counters indicate the number of specific resets which have occurred. The types are:

Full Reconfiguration Resets
Period Resets

DATE AND TIME OF GAME STATISTICS DATA:
These fields indicate the Universal date and time for the Game Stats data in the reply. For Current Statistics, this is the current date and time. For Week End Statistics, this is the date and time of the last week end snapshot.

GAME STATISTICS Data Section

The following game statistics data is provided for each game select key.

GAME ID:
This is the Game ID number for this game key as set by the host in the Configuration data packet.

GAME TYPE:
This is the Game Type number assigned to this game key. It identifies the actual firmware controlling the game.

GAME PLAY STATISTICS:
For each game, win categories must be defined for each pay table class, plus at least one for a losing category. These counters are intended to allow verification of game play probabilities by win category and can also show game play preferences, such as the number of spots picked in Keno. The sum of the values in these fields should equal the total number of games played since the last full reconfiguration. The GM manufacturer is expected to provide the order and definitions to South Australia for data-entry into the GMMS database.

For each category, a two or four byte binary count field must track the number of game play results that end in that manner. The number and size of these fields is game-dependent, but all fields for a given game must be the same size and the size must be sufficient to handle at least one week's game play.

For example, the categories identified for a Draw Poker game may include:

Royal Flush, Straight Flush, Four of a Kind, Full House, Flush, Straight, Three of a Kind, Two Pair, Pair of Aces, and Non-Win.

(Note, the defined order is the expected order of data transmission and also determines the Win Category number (1 to n, in the order defined for Statistics) for Events Log reporting of Game Win Data. See Events Log data type.)

5.7 BANNER / REPORT {Optional}

The read/write Banner/Report data type allows the host to download a Main Banner, an Alternate Banner, or a Special Report to the terminal. The banner that is displayed on the terminal is built from the Main Banner with the selected text inserted; it is referred to as the Composite Banner and can be read by the host.

A Special Report consists of up to 252 characters of text to be printed by the terminal at the convenience of the operator. A new line is indicated by a tilde,'~' ($7E), in the text. The Monitor data packet Status flag, "Special Report Print Pending", indicates whether a Special Report is waiting to be printed.

A Main Banner of up to 252 characters may be used as the basic text for a scrolling display on the terminal. This banner may include select codes to identify other information to be inserted in the displayed text stream. These select codes are designed to cause any of the following information to be inserted into the banner:

1. The location name as defined in the Configuration data packet.
2. On-site operator-enterable banner.
3. The local time or date.
4. Any of up to 10 Variable Banner data fields as sent in the Control data packet.
5. Any of up to 26 alternate fields as sent in an Alternate Banner.

SPECIAL BANNER CONTROL CODES

Select Codes allowed in the Main Banner are 2 bytes: $7E (~) followed by one of the following characters.

L - Display location name as set in the Configuration data packet, displayed to the end of the text string with trailing spaces ignored.

O - Operator-enterable banner, displayed to the end of the text string with trailing spaces ignored.

T - The local time.

D - The local date.

0-9 - Variable Banner field number as sent with the Control data packet. A $7E (~) separates the fields in the Control data packet and returns control to the next character in the Main Banner.

a-z - Alternate Banner field number as downloaded in a Banner/Report data packet. A $7E (~) separates the fields within the Alternate Banner and returns control to the next character in the Main Banner. Thus '~a' in the Main Banner inserts the first field from the Alternate Banner and '~d' in the Main Banner inserts the fourth field (beginning after the third '~' and continuing to the fourth '~' in the Alternate Banner).

An Alternate Banner consists of up to 126 characters of text partitioned into up to 26 fields. The fields may be selected for insertion into the displayed Composite Banner. Each field is delimited from the next by a tilde character, '~'.

The following table shows the contents of the Banner/Report data segment (Banner/Report Format Identifier #05) as it applies to a Banner/Report Data Write Request.

Byte	Format	Description
1	Binary1	Banner/Report Format Identifier ($05)
2		Reserved
3	Binary1	Banner/Report Version
4	Binary1	Banner Type
5-n	Ascii	Text

BANNER/REPORT VERSION:
This field contains the Banner/Report version number included in the packet. The version number of the current Main Banner and Alternate Banner as well as the version number of the last Special Report received are returned in the corresponding fields of the Monitor data type.

BANNER TYPE:
This field indicates the type of banner or report:

$01 - Main Banner Number
$02 - Alternate Banner
$03 - Special Report

TEXT:
This field contains the actual banner/report text including main banner tilde codes.
The following are maximum lengths allowed for each banner type:

Main banner - max 252 characters
Alternate banner - max 126 characters
Reports - max 252 characters

The following shows the contents of the Banner/Report data segment (Banner/Report Format Identifier $05) as it applies a to Banner/Report Data Read Reply. Note that this may result in multiple packets.

Byte	Format	Description
1	Binary1	Banner/Report Format Identifier ($05)
2		Reserved
3	Binary1	Main Banner Version
4-n	Ascii	Composite Banner Text (as displayed)

MAIN BANNER VERSION:
This field contains the Main Banner version number.

COMPOSITE BANNER TEXT (AS DISPLAYED):
This field contains the actual banner text as displayed on the terminal.

5.8 TERMINAL EVENTS LOG

The read-only Events Log data type contains the Events Log Format Identifier and records that constitute a log of events of particular interest within the terminal. **The Format Identifier is provided in Byte 1 of the Events Log data segment with the following 7 bytes reserved. The records follow and each record is eight (8) bytes long and is associated with a sequential record number.**

Byte	Format	Description
1	Binary1	Events Log Format Identifier ($05)
2-8		Reserved
9-16	Binary8	1st Events Log record (Record Number/Date Marker)

Plus additional 8-byte fields for the remaining Events Log records sent.

When a full reconfiguration is performed on a terminal, the Events Log is initialized to begin at the 24 bit record number specified in the Configuration packet. At that time, the terminal must create a Record Number/Date Marker record as the first (or base) record number in the Events Log. When a specified event occurs, it is entered into the log at the next record number. Also, after each time the Events Log is read by the host (Data Read Reply by the terminal), the terminal is expected to append a Record Number/Date Marker record after the last event which was sent from the log.

Before the host reads a terminal Events Log, it will first send a Control packet that contains an Events Log record number (Events Log Read Record Number (FROM) field) which is one greater than the last record read and recorded at the host. When the host sends an Events Log Data Read Request, the terminal replies with Events Log records beginning at the record number sent in the Control data packet and continuing through the end of the currently logged events. The terminal is expected to retain these records in memory until a subsequent Control packet confirms their acceptance by sending an Events Log Read Record Number (FROM) corresponding to the next consecutive record number in the Events Log.

To provide a cross-reference from the terminal's perspective, the Terminal Events Record Number (BASE) field in the Monitor packet is expected to reflect the record number of the first unread record in the Events Log. The terminal is expected to update this field after every Events Log Data Read Reply and the updated field is to be returned by the terminal in subsequent Monitor Replies. When the terminal receives an Events Log Read Record Number (FROM) value in a Control packet that matches this Monitor field, the terminal may "free" the memory area used by all older (lower number) records. Note, when data is exchanged error-free, each Events Log Data Read Reply from the terminal will begin at the Record Number/Date Marker which was appended after the last read.

Note: It is recommended that the terminal allocate enough memory for a minimum of 256 Events Log records. If the Events Log becomes full, the terminal must automatically shut down and prohibit game play. After the Events Log is read and a subsequent Control command is sent by the host, releasing memory for additional records, the terminal may resume operation. **See the Events Log Implementation dialogue at the end of this subsection for an implementation example, suggestions on handling potential irregular conditions, and the requirements for determining and handling a "full" Events Log.**

The previous discussion refers specifically to Events Log reads performed by the host (Data Read Requests and Data Read Replies). **When an Events Log read is performed with a Site Info Request and Site Info Reply, different rules apply** since this read is performed by GMMS components which do not log the events on the host data base. In response to an Events Log Site Info Request, the Site Info Reply is expected to include all records that are available back through the Time and Date date marker that signifies the start of the previous Universal day. To reiterate, an Events Log Site Info Reply is to start with the Date Marker (if present) that signifies the start of the previous day. No record markers are to be created and no Events Log memory is freed by a Site Info Events Log read, and no pointers used in a host Events Log read are to be moved.

Events Log records are eight (8) bytes long with the first byte identifying the event record type. The following table provides the Event Types associated with Event Log Format Identifier $05:

Byte 1	Event Type
0000 0000	Cashout ($00)
0000 0001 _ _ _ _ / 1111 1010	Game Win Data ($01 - $FA) (Substantial Wins, Max Awards)
1111 1011	Continuation ($FB)
1111 1100	Reserved ($FC)
1111 1101	Notable Events ($FD)
1111 1110	Time & Date ($FE)
1111 1111	Record Number/Date Marker ($FF)

The following tables provide the Events Log data type details of each event type:

Cashout (0000 0000) ($00):
Records every Cash Ticket printed by the terminal and every manual pay (cancel credit) performed at the terminal.

Byte	Format	Description
1	Binary1	Cashout, Type $00
2	Binary1	Cashout Code (Cash Ticket = 0; Manual Pay = 1)
3-6	Binary4	Cashout Amount in Cents (from Cashout Reply)
7-8	Time(U)	Time (like all other records; not from Cashout Reply)

Cashout Continuation:

Byte	Format	Description
1	Binary1	Continuation, Type $FB
2-4	Binary3	Venue ID Number (from Cashout Reply)
5-6	Binary2	Cashout Number (from Cashout Reply)
7-8	Binary2	Security Number (from Cashout Reply)

If the Transaction Enable flag in the Monitor field allows cash tickets and manual pays to be completed without performing a transaction, the Cashout Continuation record is inappropriate and is not included. The Cashout Amount is then provided by the GM rather than from the Cashout Reply.

GAME WIN DATA (0000 0001 - 1111 1010) ($01 - $FA):
Records win category and amount for any game which issues an award that matches or exceeds the Substantial Win Value, or which equals the Maximum Award per Game value set for each game key. Both parameters are provided by the host in the Configuration packet. This event type is also used to report any stand-alone progressive jackpot wins.

Byte	Format	Description
1	Binary1	Game Type Number 1 - 250, Type $01 - $FA respectively
2-4	Binary3	Win Amount in Cents
5	Binary1	Game ID Number
6	Binary1	Win Category number (from Win/Loss fields identified for Game Statistics)
7-8	Time(U)	Time

If the Win Amount exceeds the value that will fit in three bytes, the Win Amount in the first record should be set to zero and a Continuation record should be used to reflect the correct amount.

Game Win Data Continuation:

Byte	Format	Description
1	Binary1	Continuation, Type $FB
2-4		Reserved
5-8	Binary4	Win Amount in Cents

CONTINUATION (1111 1011) ($FB):
These records are continuations of the previous contiguous record. Continuation records are used for Cashout events and are also defined for Game Win Data and Message records.

Byte	Format	Description
1	Binary1	Continuation, Type $FB
2-8		Continuation Data

NOTABLE EVENTS (1111 1101) ($FD):
Records various notable events occurring in the terminal. All Notable Events are Type $FD with Byte 2, the subtype, categorizing the events as follows:

Byte 2	Notable Event Subtype
0xxx xxxx	- Exception Events ($00 - $7F)
1000 0000	- Credit at Snapshot ($80)
1100 0000	- Message to Host ($C0)
1111 0000	- Status Poll Timeout ($F0)
1111 0001	- Firmware Revision Change ($F1)
1111 0010	- Encryption Key Change ($F2)

Exception Events:
Records when an exception event is detected by the terminal. Note, the Exceptions Report Enable flag field in the Configuration data type does **not** enable/disable exception event reporting with respect to the Events Log. These exception events are always reported in the Events Log but note, not all Status Response exception events are required in the Events Log.

Byte	**Format**	**Description**
1	Binary1	Notable Event, Type $FD
2	Binary1	Exception Event Code, (Subtype $00 - $7F)
3-6	Binary4	Event Data **(Data included only when stated in the exception description; otherwise zero)**
7-8	Time(U)	Time

Exception Code	Description
	Significant Events:
	(Max Award reported as Event Type $01 -$FA)
$03	Gaming Machine Entered Playable State
$04	Entered Non-playable State (caused by GM)
$05	Entered Non-playable State (caused by GMMS)
$OB	Status Response Buffer (exception queue) was Filled
$0C	Events Log Full
$0D	Daily Poll Communication Timeout
	Special Conditions:
$10	Returned to All Special Conditions Cleared (applies to codes $11-1D)
$11	Coin Acceptor Tilt (stringing, timing, jam, etc.)
$13	Bill Acceptor Tilt (jam, stacker full, etc.)
$15	Printer Paper Low
$16	Printer Paper Out
$19	Hopper Empty
$1B	Hopper Jam
$1D	Reel Tilt (mechanical reels)
$1E	Coin Diverter Error (coin to wrong location) (NO CLEAR)
$1F	Hopper Excess Coin (one) Dispensed (NO CLEAR)

Door Accesses:

$2x Door accesses: (only closure reported is "all doors closed")
x = addd
ddd = 000 = All doors Closed
ddd = 001 = Main Door Opened
ddd = 010 = Cash Door Opened
ddd = 011 = Secondary Door Opened
ddd = 100 = Logic Door Opened
a = State of Audit Key when doors are opened or all-closed

Door Examples:

$28 Change of state to All Doors Closed (while Audit Key on)
$22 Cash Door Opened (while Audit Key off)
$2C Logic Door Opened (while Audit Key on)

System Failure Faults:

$30 Returned to "no Fault" condition (Applies to $31 - $47)
$34 EPROM Checksum Failure
$35 Fatal RAM Failure
$36 Battery Low
$37 Main Logic Failure
$38 Power Supply Failure
$39 Coin Acceptor Failure
$3B Hopper Failure / Runaway (2 or more excess coins)
$3D Printer Failure
$3E Bill Acceptor Failure
$3F Touchscreen Failure
$40 Over-temperature
$41 Electromechanical Meter Failure
$45 Miscellaneous Firmware failure
EVENT DATA: byte 6 = failure code set or approved by jurisdiction
$46 Miscellaneous Electronics failure
EVENT DATA: byte 6 = failure code set or approved by jurisdiction
$47 Miscellaneous Mechanical failure
EVENT DATA: byte 6 = failure code set or approved by jurisdiction

Game Wins, and Cashouts are reported as different Events Types (not as Exception Events) in the Events Log, and Local Site Item (Exception Events) are not reported in the Events Log.

Credit at Snapshot:
Records if there is a credit balance on the terminal at the time of the Day-End Snapshot.

Byte	**Format**	**Description**
1	Binary1	Notable Event, Type $FD
2	Binary1	Credit at Snapshot Subtype - $80
3-6	Binary4	Credit Balance in Cents
7-8	Time(U)	Time

Message to Host {optional}:
Records text of a message to be sent to the host. Messages longer than 6 characters use continuation records for the additional text at 7 characters per continuation record with a maximum of 9 total packets per message. If implemented, messages are entered through a secure means by site personnel at the terminal.

Byte	Format	Description
1	Binary1	Notable Event, Type $FD
2	Binary1	Message to Host Subtype - $C0
3-8	ASCII(6)	Message Text

Message Continuation:

Byte	Format	Description
1	Binary1	Continuation, Type $FB
2-8	ASCII(7)	Message Continuation Text

Status Poll Timeout:
Records the loss and restoration of polling communication as perceived by the GM.

Byte	Format	Description
1	Binary1	Notable Event, Type $FD
2	Binary1	Status Poll Timeout, Subtype - $F0
3-5		Reserved
6	Binary1	Communication change-of-state $00 = Communication lost (timed-out) $01 = Communication restored or established
7-8	Time(U)	Time

Firmware Revision Change
Records a change in the installed terminal firmware (Game or System Firmware Version as returned in the Monitor packet).

Byte	Format	Description
1	Binary1	Notable Event, Type $FD
2	Binary1	Firmware Revision Change, Subtype - $F1
3	ASCII(1)	New System Firmware Version
4	ASCII(1)	Old System Firmware Version
5	ASCII(1)	New Game Firmware Version
6	ASCII(1)	Old Game Firmware Version
7-8	Time	Time (U)

If only one Firmware Version changes, the other Old and New should reflect the current Version.

Encryption Key Change
New encryption keys have taken effect.

Byte	Format	Description
1	Binary1	Notable Event, Type $FD
2	Binary1	Encryption Key Change, Subtype - $F2
3	Reserved	
4	Binary1	Cause of Change: 0 = Triggered by Normal Communications 1 = Forced Back to Default
5-6		Reserved
7-8	Time	Time (U)

TIME AND DATE (1111 1110) ($FE):
Records any events related to Time and Date. The date in these records is used as the date reference for subsequent records in the log.

Byte	Format	Description
1	Binary1	Time and Date, Type $FE
2	Binary1	Record Reason: $00 = Date Marker (e.g. Calendar rollover) $01 = Power Failure $02 = Date/Time Set by Terminal $03 = Date/Time Set by Host
3-4	Time(U)	Old Time
5-6	Date(U)	New Date
7-8	Time(U)	New Time

Old Time is used only when the date and time are set and for power failures. For Power Failure, Old Time is when the failure occurred; New Time and Date is when the power was recovered. For Date Marker, Old Time is not used (0).

RECORD NUMBER/DATE MARKER (1111 1111) ($FF):
A record of this type will be written in the Events Log when it is initialized and on each Events Log Read executed. Thus, a Record Number/Date Marker will be the first record in an Events Log Data Read Reply if the Events Log Read Record Number in the Control packet is advanced by the GMMS in accord with the last record received in the previous Events Log read. As with the Time and Date records, the date in this event type can serve as a reference for subsequent records.

Byte	Format	Description
1	Binary1	Record Number/Date Marker, Type $FF
2-4	Binary3	Record Number of this Record.
5-6	Date(U)	Date
7-8	Time(U)	Time

EVENTS LOG IMPLEMENTATION: The following is a detailed description of an implementation of the Events Log. It is offered as a guide and as a means to answer by example unique situations that should be considered.

The terminal events log RAM is a circular queue of 8 byte records used to store significant events occurring at the terminal. Three internal record number pointers control this queue:

START - Identifies the lowest record number that must be retained by the terminal.

BASE - Identifies the lowest record number that has never been read under a normal Events Log Read. (Note, the record number of the BASE pointer is the value to be returned in the Monitor data packet. It identifies the record number the terminal expects will be the start of the next Event Log read if the last Data Read Reply was successful.)

END/NEXT - Identifies the next record number to be added to the Events Log.

An additional pointer is provided by the GMMS in each Control write packet (Events Log Read Record Number) to confirm reception of all events before that record number:

FROM - Identifies the beginning record number the host wishes to read from. It is supplied in the Control data packet.

The records from START up to BASE have been read from the terminal but must still be retained by the terminal in case there is a problem in processing the information. The records from BASE to END/NEXT are new records since the last normal Events Log Read. Under routine polling, the host, having processed the records it last read (from START up to BASE), will write a control packet with FROM pointing to BASE as the record number from which to begin the next read. The terminal now moves the START pointer up to FROM. The host requests an Events Log Read and the terminal sends the records inclusive of FROM (equal to START) up to END/NEXT. It then adds a Record Number/Date Marker record to the Events Log, sets BASE to this record, and increments END/NEXT by one. Since the next Events Log Read by the host will normally be from a record number one higher than the last one it read, this assures that the first record of each Event Log Read is a Record Number/Date Marker. This means that START and BASE will always point to Record Number/Date Marker records.

The host refers to the records through a 24 bit record number. This record number can be mapped to a physical RAM location by taking the record number modulo the number of records that can be stored in RAM. This number times 8 bytes is the address offset into the record RAM area. Note that this forms a circular queue. Each new record added is stored 8 bytes higher than the start of the previous record until the end of the allocated memory area is reached. At this time, the next record goes to the beginning of the allocated RAM area. After the first cycle through, records added at END/NEXT will be overwriting the lowest record number still in RAM.

If END/NEXT+32 records maps to the same memory address as START, the Event Log memory has only thirty two entries free. At this time, the terminal must inhibit further game play until a normal Events Log Read causes the START pointer to advance. The next record is used to record the Events Log Full event, which is reported as an exception in a Status Response. The next 30 entries may be used to record any subsequent events, with the last entry reserved for the Record Number/Date Marker resulting from the expected Events Log Read. If END/NEXT+1 maps to START, no further Event Log entries are made until START advances, freeing some memory. This approach is necessary

to assure that the record following the last record number in any Events Log read is a Record Number/Date Marker.

The Events Log is initialized during a full reconfiguration. This is done when the terminal is first placed into the full reconfiguration mode (Control packet sets the terminal mode to Full Configure Enable) and a configuration packet is sent to the terminal. Note, at the start of receiving a Configuration packet while in this mode, the Events Log ceases to record events. After receipt of the Configuration, the Events Log is initialized to begin at the record number specified in the Beginning Events Log Record Number for Full Reconfigure field in the Configuration data packet. That is, the START, BASE, and END/NEXT pointers are all set to point to that record number. A Record Number/Date Marker record is then entered in the log and END/NEXT is incremented to the next record.

Note that if a Configuration packet is received while the terminal is not in the Full Configuration Enable mode, the Events Log is not restarted. In this case the terminal is expected to ignore the Beginning Events Log Record Number value and continue the Events Log as if nothing has happened.

When an Events Log Read Request comes in from the host, the terminal uses the Terminal Events Log Record Number (FROM) field in the last Control packet received as the first record from which to start the read. To prevent anomalies, the terminal is expected to process the Events Log Read depending upon the relationship of FROM to the three internal pointers:

A normal Events Log Read is one in which FROM is greater than or equal to START and less or equal to BASE. In this case only, the BASE pointer is set to END/NEXT immediately following the Events Log Read, prior to adding the next Record Number/Date Marker record.

The Events Log Read begins at the FROM record and goes up to the END/NEXT record, except in the following situations:

a) FROM less than Beginning Record Number for Full Reconfigure ..... the Events Log read reply should begin at the lowest record in RAM and go up to END/NEXT.

b) FROM less than lowest record number in RAM and greater than or equal to the Beginning Record Number for Full Reconfigure ..... the Events Log read reply begins at START and goes up to END/NEXT. Note, the lowest possible record number in RAM is the greater of END/NEXT less the RAM size in records, or the Beginning Record Number for Full Reconfigure.

c) FROM greater than or equal to END/NEXT ..... the Events Log read reply begins at START and goes up to END/NEXT.

After being enrolled on the system and receiving a full reconfiguration, the only time that START is adjusted is when a Control packet is received. At that time, if FROM is greater than or equal to START and less than or equal to BASE and points to a record number/date marker record, START is set to FROM.

Following any Events Log Read (unless the events log is still full after the read), a Record Number/Date Marker record is added to the Events Log at END/NEXT and END/NEXT is incremented by one.

5.9 DATE AND TIME / MEMORY SIGNATURE

The read/write Date and Time / Memory Signature data type is used to read or set the current Universal date and time at the terminal and, in a some cases, to command the terminal to calculate a Memory Signature Value and in turn provide the result of the Memory Signature calculation.

The following table shows the contents of the Date and Time / Memory Signature data segment as defined with Format Identifier $05:

Byte	Format	Description
1	Binary1	Date and Time / Memory Signature Format Identifier ($05)
2-8		Reserved
9-10	Date(U)	Current date
11-12	Time(U)	Current time
13-14	Binary2	Memory Signature Value, $FFFF if calculation in progress
15-16	Binary2	Memory Signature Divisor

A Date and Time / Memory Signature Data Write Request will always have zero in the Memory Signature Value field. Whenever a **non-zero** Memory Signature Divisor is received in a Data Write Request, the terminal, as well as setting the date and time, is expected to begin calculation (aborting any calculation in progress from a previous Date and Time / Memory Signature write) of a Memory Signature Value based on the new Divisor received. The terminal is expected to report the Divisor and resulting Value in response to any subsequent Date and Time / Memory Signature Data Read Requests or Site Info Requests. The Memory Signature Value should be reported as $FFFF while these calculations are in progress. Note, when this type of Memory Signature Calculation is completed, the terminal is also expected to generate a Status Response with exception code $02. Memory Signature Divisors and Values received or generated in this way are not reported in the Monitor data type. See Appendix A for the Memory Signature algorithm.

Note: All Date and Time data pairs exchanged with the host computer are handled in Universal Time. At times, the terminal may need to convert between Universal and local time for such things as printing local time on tickets. To do this, the terminal needs to know the time zone offset and whether daylight savings time is in effect. This information is available in the Configuration data type fields, Time Zone Code/DST Mode and Date and Time to Make Daylight Savings Time Change.

When the appropriate encryption is enabled, all data segments and validation segments of Date and Time / Memory Signature packets (Data Read/Write and Site Info) are encrypted with Encryption Key #2. This is different than other Data Read/Write data types.

Once the host removes a terminal from demo mode, this data type packet must be the only means by which to set the date and time in the terminal.

APPENDIX A: MEMORY SIGNATURE VALUE ALGORITHM

MEMORY SIGNATURE VALUE:

This two-byte number is a calculation by the terminal over all memory (PROM or EPROM or other storage media). The computation algorithm uses a Memory Signature Divisor provided in the Control packet or in the Date and Time/Memory Signature packet. The Memory Signature Value is the calculation result and serves to validate a terminal's chipset. It is meant to detect memory failures, to prevent tampering, and to ensure general memory integrity.

When the terminal receives a Control packet, it is expected to compute a Memory Signature Value using the Memory Signature Divisor contained in the Control packet and return the Memory Signature Value in subsequent Monitor data read replies. It is recognized that the calculation may require some time, so the terminal must return $FFFF in Monitor data read replies only when the Memory Signature Divisor changed and the computation with that Divisor is still underway. If the Memory Signature Divisor sent in a Control packet has not changed, then the Memory Signature Value returned in response to a Data Read Request of the Monitor data must be the previously calculated Signature Value. However, remember that even though the Memory Signature Divisor did not change upon receipt of the current Control packet the Memory Signature Value must still be recalculated.

On the other hand, whenever a non-zero Memory Signature Divisor is transmitted in a Date and Time/Memory Signature Data Write Request, the GM is required to compute the Memory Signature Value and, when completed, generate an appropriate Status Response and return the value to the GMMS in response to a subsequent Request. (In some cases, the terminal may be disabled by the GMMS until this Memory Signature calculation has been completed, so calculation expediency will minimize delays to the player.) The specific Memory Signature Divisor and Value associated with this process are only provided by the GM in Date and Time / Memory Signature Data Read and Site Info Replies and are <u>not</u> reflected in the Monitor and Control fields.

MEMORY SIGNATURE VALUE ALGORITHM:

The host will provide a Memory Signature Divisor number between hexadecimal 0001 and FFFF (decimal 1 and 65535) to be used as the divisor over memory.

To calculate the Memory Signature Value, the first two bytes of memory will be taken as a word (the lower address byte as the most significant byte) and divided by the Memory Signature Divisor number. The quotient will be discarded. The remainder from this division will be shifted left 16 bits (i.e. multiplied by 65536) and logically or'd with the next word of memory to form a 32 bit number. This number is again divided by the Divisor, the remainder shifted left 16 bits, or'd with the next word of memory, and again divided by the Divisor. This process will continue over the entire firmware/data memory whether it contains significant code/data or not, unless it can in no way affect game play or accounting, such as video character generators or memory that is part of peripheral devices. If the memory information does not contain an even number of bytes, the last 16-bit value from the memory will consist of the last byte of information in the upper part of the word and the lower part will be padded with a hexadecimal FF.

The resulting 16-bit remainder from the last divide is the Memory Signature Value.

APPENDIX B: GM COMMUNICATION BUS SPECIFICATIONS

COMMUNICATION BUS DESCRIPTION

To require only one modem within a site and still allow multiple Gaming Machines (terminals) to communicate with the Gaming Machine Monitoring System, a communication bus which is common to all terminals is specified. The defined Gaming Machine Communication Bus is based upon the EIA RS-422 bus standard, with the only deviation from the RS-422 standard being that the drivers and receivers used in the hardware implementation are required to comply with the specifications of most EIA RS-485 compatible devices. This "enhancement" permits more terminals to be connected to a single (unamplified) bus by taking advantage of the lower input current requirements of RS-485 receivers and the higher output current capabilities of typical RS-485 drivers (see Selection of Drivers and Receivers). The device within the site which controls the modem is referred to as a Modem Master. Each terminal is expected to be capable of serving as a Modem Master. In some cases, when an optional VLC Site Controller is installed, all terminals in the site will be selected to function as slave devices. GMs are required to be capable of supporting the bus at a rate of either 2400 or 9600 baud, with a manual set-up means of making this selection. When a GM serves as the Modem Master, bus operation will be at the modem transfer rate of 2400 baud because the modem master simply gates the modem data onto the bus; when a Site Controller serves as the modem master, 9600 baud bus operation will be utilized.

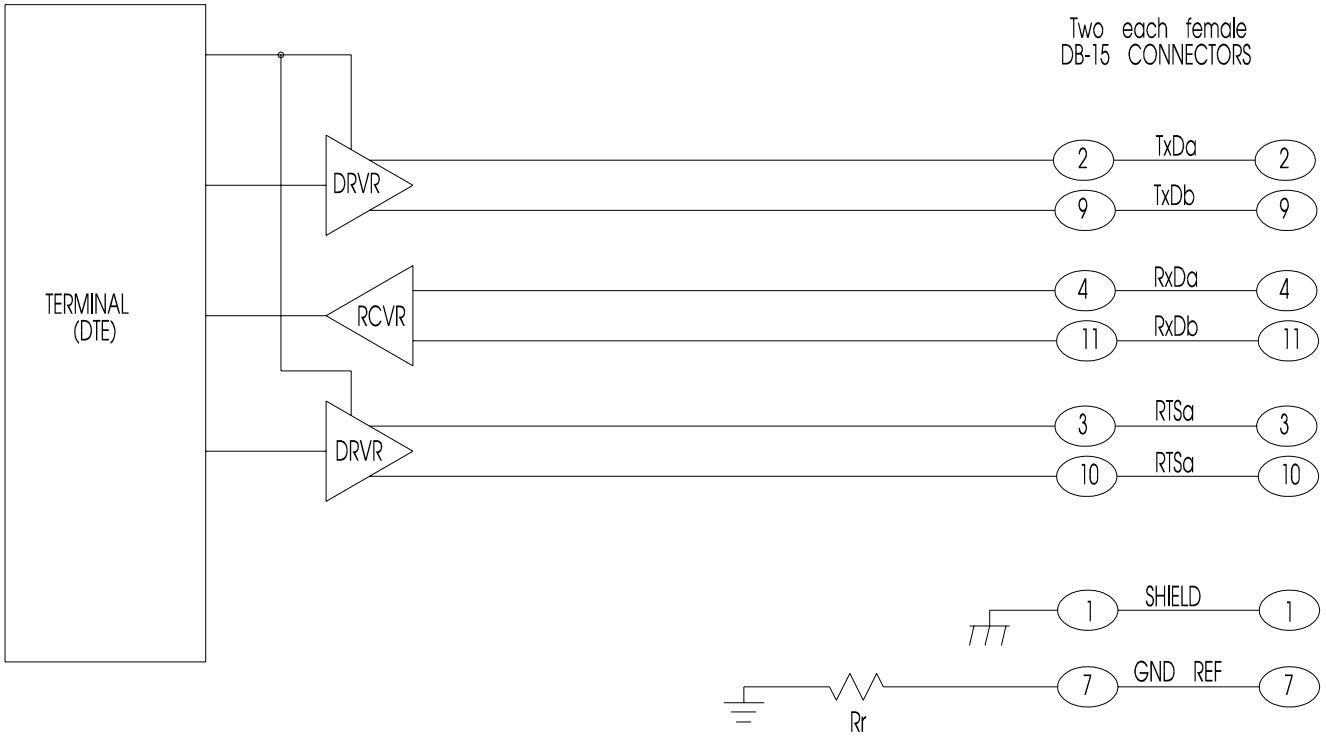
Three lines, each having differential signals for high noise immunity, are required on the Communication Bus. The lines specified are the complimentary TxD (Transmit Data), RxD (Receive Data), and RTS (Request To Send) signals, along with a ground Reference signal (7 signal wires total). To provide compatibility between different manufacturers, a standard connection scheme using two DB-15 connectors (15-pin series D) is also required. These two connectors, which are connected in parallel, make daisy-chaining terminals with identical cables a simple matter.

The **RxD** bus signals are only driven by the modem included in the Modem Master and terminals receive data on this line. All terminals on the bus are to constantly listen to the RxD line, looking for packets addressed to them.

The **TxD** signals are driven by the terminals, and the modem in the Modem Master receives data from this line. The **RTS** signals are also driven by the terminals and function to gate data onto the bus in situations where optional Bus Expanders are required. Since any terminal on the bus must be able to transmit on the TxD and RTS lines, each terminal must tri-state these drivers to put them in a high impedance state when not transmitting. When sending a reply, a terminal is expected to enable its transmit control line, assert RTS, and then drive TxD with the proper data response. After the last stop bit of a packet is transmitted, these TxD and RTS signals must be tri-stated within 8 milliseconds.

The ground **Reference** signal on the bus should be referenced to the logic ground of the bus drivers and receivers in each terminal through a resistor of approximately 270 Ohms. This addresses common mode voltage concerns by providing a pseudo ground reference between drivers and receivers in separate terminals, while at the same time minimizing direct ground loops within the local network. The **Shield** pin specified on the bus connectors is to be tied to earth ground (chassis) within the terminal.

The following figure provides a graphical representation of the required hardware interface when the terminal is not selected as a Modem Master:



RS422 BUS DB-15 Connectors

Pin	Name	Description	Polarity
1	SHIELD	Chassis Ground	
2	TxDa	Transmitted Data	LOW = SPACE; HIGH = MARK
9	TxDb	Transmitted Data	HIGH = SPACE; LOW = MARK
3	RTSa	Request to Send	LOW = Asserted; HIGH = Negated
10	RTSb	Request to Send	HIGH = Asserted; LOW = Negated
4	RxDa	Received Data	LOW = SPACE; HIGH = MARK
11	RxDb	Received Data	HIGH = SPACE; LOW = MARK
5	CTSa	Clear to Send	LOW = Asserted; HIGH = Negated
12	CTSb	Clear to Send	HIGH = Asserted; LOW = Negated
7	GND REF	Ground Reference	

Rr = Reference Resistor = approximately 270 Ohms (to minimize ground loops)

Slave Terminal Hardware Implementation

As depicted in Terminal Implementation and Terminal Interconnection figures in this Appendix, the Communication Bus must be available in each terminal on two DB-15 connectors (15-pin Series D, with female sockets) with common signals (paralleled) on each connector. Note, the two DB-15 connectors on each terminal are interchangeable; there is no "in" or "out".

BUS TERMINATION

When multiple terminals are interconnected, each line of the bus should be terminated with approximately 180 Ohms to further improve noise immunity by creating a lower impedance transmission line. The termination resistor is considered most effective when placed across the inputs of the receiver which is farthest from the related transmitter, or across the inputs of the receiver that is driven by the most remote transmitter. To observe this guideline, termination of the individual lines is to be performed as follows:

* RxD is to be terminated at the last terminal (or other device, such as a Bus Expander) on the bus. This is typically accomplished through the use of a termination plug which plugs into the unused bus connector of that device. The termination plug for the last device contains one resistor (nominally 180 ohms) connected between pins 4 and 11 (RxDa and RxDb).

* TxD is to be terminated at the Modem Master device, since the receiver through which the modem listens is the only receiver on that line. In addition to termination, the Modem Master device is expected to fulfill the responsibility of establishing a known state on the TxD line when the bus is tri-stated. The established standard is to maintain the TxD line in a "Mark" state while tri-stated; this biasing along with the termination can be accomplished as follows:

  Provide a 1K Ohm pullup resistor from TxDa to +5V, a 1K Ohm pulldown resistor from TxDb to Logic Gnd, and a 270 Ohm resistor from TxDa to TxDb. Note, these resistors should only load the signal lines when the terminal has been selected as a Modem Master in the Internal Modem Implementation.

  Since the terminal is required to have a method of manually selecting it as the Modem Master, it is preferable if the biasing termination described above takes place at the same time or in a similar manner as the Modem Master selection.

* RTS will be terminated by the Bus Extender if any of these devices are used.

These termination requirements are depicted in Terminal Implementation and Terminal Interconnection figures in this Appendix,

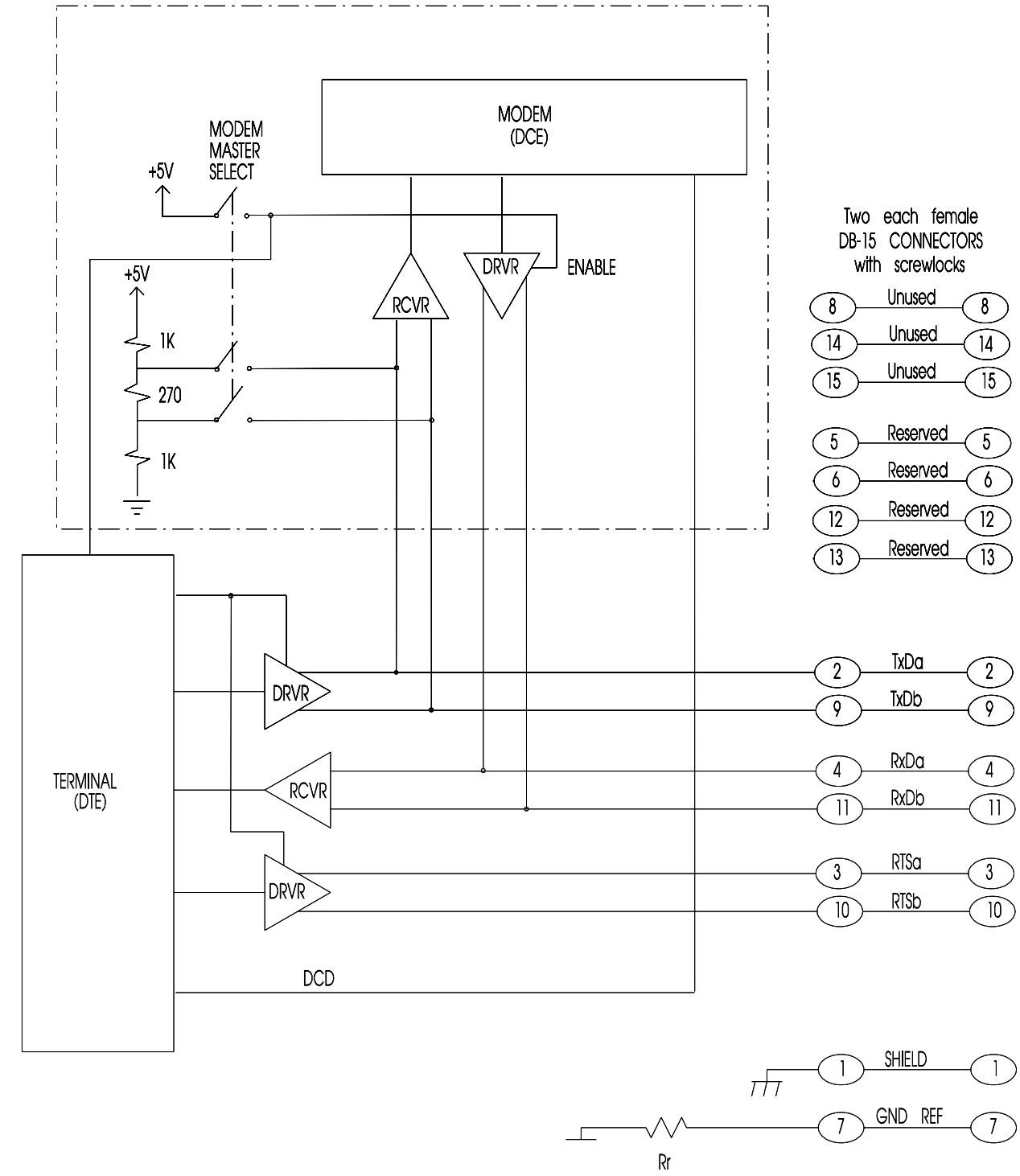
MODEM MASTER IMPLEMENTATION

The basic function of the Modem Master is to link a modem to the communication bus. The specific responsibilities of a Modem Master are:

1) provide the modem.
2) ensure that all incoming calls are properly answered.
3) gate the telecommunication data received by the modem onto the RxD line of the bus.
4) provide an interface for the modem to hear all data transmitted on the TxD line of the local communication bus.
5) terminate and bias the TxD line of the bus.
6) ensure that the phone is placed "on hook" when carrier is lost.

If the modem selected by a manufacturer operates reliably through power cycling, lost carrier, and other functional situations while only utilizing its default set-up mode, there may be no need for the Modem Master terminal to directly communicate with the modem. On the other hand, if it is deemed appropriate to confirm the modem status and/or at times re-establish the modem parameters, a Modem Master terminal is to send commands to the modem only when a Connection is not currently established by the modem. To provide this indication, the DCD (Data Carrier Detect) signal of the modem can usually be set up to follow the Connect/No Connect state of the modem. Terminals that are not designated as the Modem Master are not to send set-up control commands to the modem.

The Modem Master Hardware Implementation drawing on the following page shows the Modem Master circuitry permanently attached to the communication bus. With the depicted implementation, the Modem Master selection switches inform the terminal of when it is selected as a Modem Master, gate the driver onto the RxD lines, and attach the termination/biasing resistor network to the TxD signals. If the Modem Master selection was to be made by selectively attaching the required circuitry, the switches would obviously not be necessary. Probably just as obviously, one of the possible points of attaching the circuitry is on the unused communication bus connector of the Modem Master terminal. If desired, manufacturers are free to use the unused pin (8,14, and 15) of these bus connectors to implement the Modem Master function.



Modem Master Circuitry shown in dashed box.

Rr = Reference Resistor = approximately 270 Ohms (to minimize ground loops)

Modem Master Hardware Implementation

SELECTION OF DRIVERS AND RECEIVERS

DRIVERS:
All drivers must have tri-state outputs to allow multiple terminals to communicate on the same bus; terminals internally drive the state of the outputs through what has been termed the transmit control line. Terminals must be designed to assure that a failed or unpowered terminal will not drive the bus signals. If all drivers within terminals assume the high impedance state when not powered, other terminals on the bus can still communicate. In addition, the differential line drivers are expected to meet the following specifications:

High-level output current, I_{OH}: -60mA min
Low-level output current, I_{OL}: 60mA min
Output current with power off, I_O: ±100þA max
Tri-state output current, I_{OZ}: ±100þA max

Some drivers which have suitable published specifications are:
SN75172 (DS96172, LTC486), SN75174 (DS96174, LTC487)

RECEIVERS:
The differential line receivers are expected to meet the following specifications:

Common-mode input voltage, V_{IC}: ±7V
Input Resistance, r_i: 12K Ohms min

Some receivers which have suitable published specifications are:
SN75ALS193, SN75ALS195, SN75173 (DS96173, LTC488), SN75175 (DS96175, LTC489)

There are also numerous transceivers (which can be used) which have suitable published specifications, including:
SN75176B, SN75ALS176B (LTC1485), DS3695 (TL3695), SN75ALS180 (LTC491), SN751177, SN751178, SN75ALS170, SN75ALS171
The part numbers in parentheses "()" are reportedly equivalent to the previous part.

VLC does not suggest that suitable published specifications constitute acceptable performance, but merely offers the list of possible components as a starting point for terminal manufacturers. Manufacturers are expected to confirm performance acceptability for their individual products.

Proper bus operation results in a minimum differential voltage of 2V on each of the lines when the following conditions exist: Maximum Bus Capacity, Maximum Bus Cable Length, Bus Termination of 180 ohms, and Common Mode voltage of +7V or -7V introduced.

When implementing the driver/receiver interface, manufacturers are encouraged to design and test for device immunity to voltage transients that may potentially be induced onto the site cabling. Effective driver/receiver voltage transient protection has been implemented using bi-directional Zener transient voltage suppressors on each bus signal or by using fast-acting diodes in series with a unidirectional Zener transient voltage suppressors. It is recommended that the protection circuitry be designed to clamp both the positive and negative potentials to a +/- 7V of common-mode voltage swing. If deemed appropriate, optical isolation may also provide further protection for internal circuitry
.

BUS CABLES

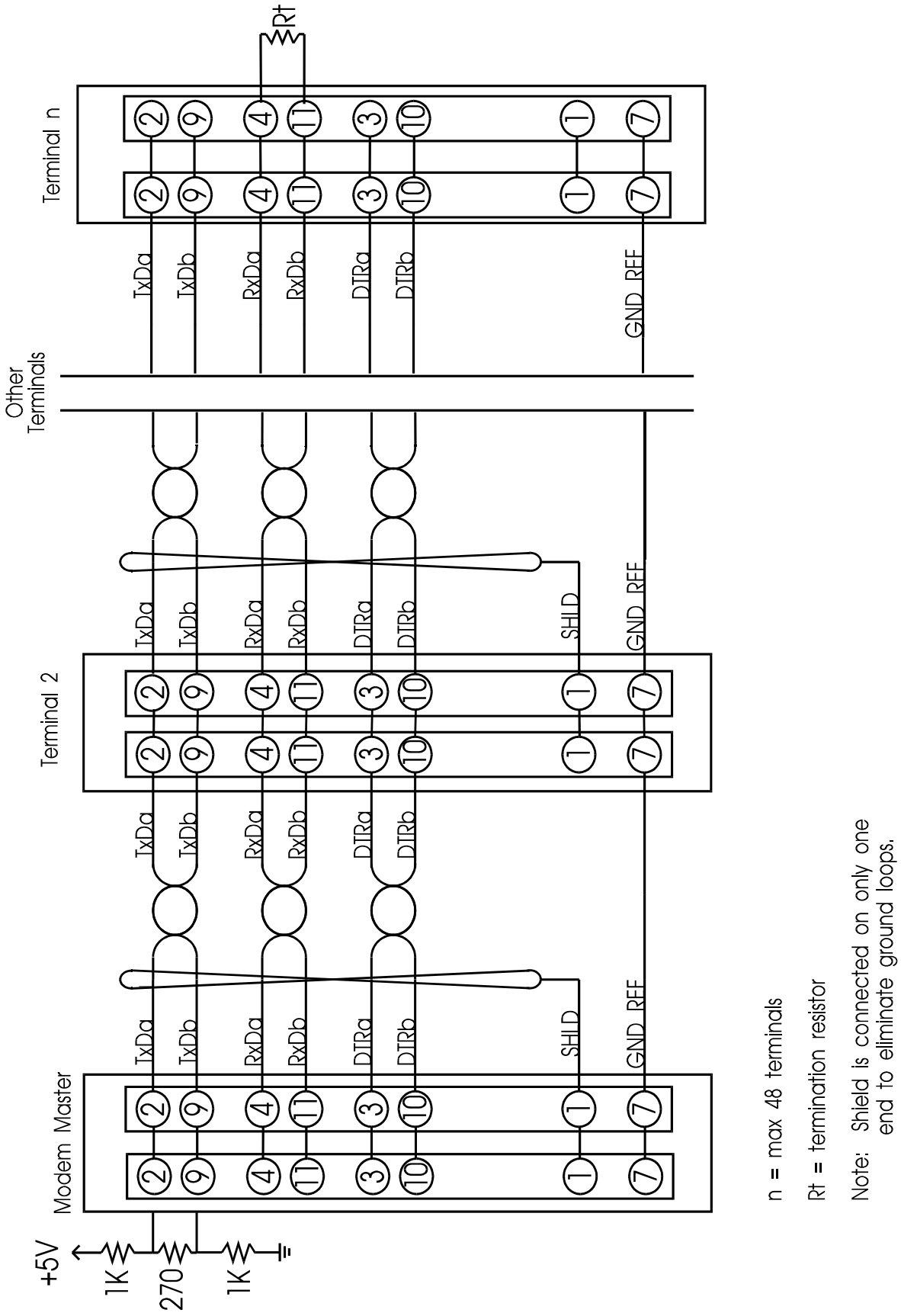
It is recommended that shielded interconnecting cables consist of a twisted pair (26 AWG or larger) for each line (pair of complementary signals); twisted pairs provide increased noise immunity and improved common mode rejection. Identical cables are used to interconnect all Gaming Machines. One cable from the first terminal will connect to the second terminal on the bus, with another cable interconnecting the second terminal to the third, and so on. As represented in the Terminal Interconnection Illustration, this daisy chaining continues to the last terminal on the bus. The last terminal on the bus is expected to include a termination method (plug) which includes a resistor to terminate the RxD signal. Termination of the TxD signals is to take place at the Modem Master. The site is configured with the Modem Master being either the first terminal on the bus, or a Site Controller, in which case none of the terminals is the Modem Master. **Cables are to be made such that the cable shield ties to the Shield pin at only one end of each interconnecting cable.** See the Terminal Interconnection Illustration for cable construction.

The total length of cable on a single (unamplified) bus must not exceed 2000 feet (610 meters) with 26 AWG wires or 4000 feet (1220 meters) with 24 AWG or larger wires in the cables. This total length limitation refers to the sum of the lengths of all the cables which are used to interconnect terminals and the Modem Master. If the site configuration requires a greater length, at least one Bus Expander must be used.

BUS CAPACITY

When the previously described requirements are met (cable lengths and driver/receiver selection), a single Gaming Machine Communication Bus can function reliably (sufficient differential voltage margin and noise immunity) with up to forty-eight (48) Gaming Machines directly interconnected.

In other words, there is no need for a Bus Expander in a site unless more than 48 terminals are installed or the distance between terminals causes the total cable length limitation to be exceeded. If or when Bus Expanders are required, they will be provided by VLC.



Terminal Interconnection Illustration

APPENDIX C: DIAL-UP MODEM SPECIFICATIONS

In the South Australia system, each dial-up modem is to have the following basic specifications:

* CCITT V.22/V.21 compatible
* 2400 baud operation
* Must have any compression and/or error checking Disabled
* Austel-approved

MODEM MASTER SYNOPSIS:

Each Gaming Machine is expected to support the function of Modem Master and the manufacturer is expected to designate and provide the modem to be used when one of their terminals is selected as a Modem Master. Each Gaming Machine must optionally provide the hardware and other support required to interface the modem to the Gaming Machine Communication Bus (see Appendix B).